al-Farabi Kazakh National University

UDC 004.386 On manuscript right

ABYLKASSYMOVA AIZHAN BOLATOVNA

POTENTIAL OF HYBRID OPENMP/MPI PARALLELIZATION STRATEGIES FOR HPC SOFTWARE

6D060200-Computer science

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (PhD)

Supervisors:
Associate Professor, candidate of physico-mathematical Sciences, Mansurova Madina
al-Farabi Kazakh National University

Dr.-Ing. Matthias Meinke
JARA – HPC - RWTH Aachen University (Germany)

Republic of Kazakhstan, Almaty, 2022

Content

Acknowledgements

The author expresses his gratitude to the supervisor Professor Mansurova Madina (Al-Farabi Kazakh National University) for her support in the work, Professor Matthias Meinke (RWTH Institute of Aerodynamics, Aachen, Germany) for the tasks and attention paid to the work, as well as PhD A. Niemöller (RWTH Institute of Aerodynamics, Aachen, Germany) and as PhD Issakhov Alibek (Al-Farabi Kazakh National University) for fruitful discussions and valuable comments.

Aizhan Abylkassymova

Designations and abbreviations

In this dissertation the following terms with appropriate definitions are used:

∇ – the operator of nabla,
Δ – Laplacian vector operator,
t – time,
ν – coefficient of kinematic viscosity, m^2/s
ρ – density,
p – pressure, kg/m^3
$\vec{u} = (u, v)$ – the components of the flow velocity,
$\vec{f}$ – vector field of the mass forces.
$\delta(i, j)$ – is the Kronecker delta
Re - Reynolds number
P – number of processors
E_p – efficiency of parallel algorithm
S_p – speed-up of parallel algorithm
T_{comm} – time of execution of communication operation

INTRODUCTION

Information technology is gaining popularity day by day, as today it is the era of high-performance parallel computing systems. For this reason, the issue of accelerating and processing and analyzing large amounts of data is acute. There are a number of ways to deal with this issue, such as multi-core machines, supercomputers, or grid systems. Also, parallel algorithms are used for processing, modeling and visualization of both initial and transformed data. Depending on the technologies used in the construction of parallel programs, the architecture of computing systems also depends (MPI libraries for message passing [1, 2], high-level parallel programming systems (Unified Parallel C, ParJava, DVM, T-system, Cray Chapel, IBM X10) support for multithreading and development of programs for specialized systems (NVIDIA CUDA, OpenCL, OpenACC, OpenMP)).

As it knows, the use of parallel technologies can be due to such reasons as modeling real physical problems described by systems of differential equations in partial derivatives. For example, flows of a viscous incompressible medium (one of the important problems of mechanics).

Moreover, in real life and in many sciences (medicine, architecture, industry), the problems of the interaction of air flow with various obstacles are very popular. This relevance of the aerodynamic stability of structures appeared after some unfortunate incidents [3-7].

Scientists, who have made a significant contribution to the theory and practice of computer systems and parallel computing technologies, are: J.L. Traf, T. Sterling, M. Snir, R. Rabenseifner, S. Matsuoka, T. Hoeler, W. Gropp, S. Cray, J. Dongarra, P. Balaji, N.N. Yanenko, Yu.I. Shokin, B.N. Chetverushkin, V.G. Khoroshevsky, Ya.A. Khetagurov, A.N. Tomilin, V.B. Smolov, G.G. Ryabov, G.E. Pukhov, D.V. Puzankov, I.V. Prangishvili, D.A. Pospelov, Yu.I. Mitropolsky, V.A. Mellnicks, G.I. Marchuk, I.I. Levin, V.K. Levin, S.A. Lebedev, A.O. Latsis, V.G. Lazarev, L.N. Korolev, V.V. Korneev, Yu.G. Kosarev, I.A. Kalyayev, A.V. Kalyaev, M.B. Ignatiev, V.P. Ivannikov, A.V. Zabrodin, E.V. Evreinov, V.F. Evdokimov, V.M. Glushkov, V.V. Voevodin, V.V. VaSyliev, V.S. Burtsev, V.B. Betelin, E.P. Balashov, S.M. Abramov and etc.

In [8], a parallel algorithm for modeling geological environments is presented. A distinctive feature of this algorithm is the use of several parallel programming technologies, such as MPI, OpenMP, CUDA. This work is aimed at studying the simulation of acoustic wave propagation in an inhomogeneous medium for high-performance systems, which is an important aspect in the field of seismic exploration.

Whereas in [9] a number of studies of the execution time of various types of implementation of collective algorithms were carried out. Moreover, the dependence of energy, memory and execution time of certain algorithms was revealed.

Dalle, in [10-12] papers talk about a natural study of the effectiveness of the parallel seismic migration algorithm in reverse time for Blue Gene/P. That is, a question is raised that arises when using the program on a different number of processors.

As is known, the use of parallel technologies can be due to such reasons as modeling real physical problems described by systems of differential equations in partial derivatives. For instance, the flow of a viscous incompressible medium is one of the important problems of mechanics. In fact, it is very difficult to achieve high efficiency for large-scale parallelized tasks. Since even the slightest imbalance can lead to undesirable results in overall performance. Dynamic load balancing (DLB), in turn, improves the efficiency of complex modeling with non-trivial domain decomposition. This possibility is provided by the Hilbert Space Filling Curve (SFC) at a coarse level. Since it is necessary to take into account various numerical methods, as well as various computational costs this schema automatically assigns weights to estimate the overall workload distribution. Due to the inability of this evaluation procedure to capture local workload changes, the overall SFC-based load balancing approach may not be optimal. Therefore, the DLB incremental diffusion algorithm is based on SFC separation, which allows to customize the domain decomposition. Simulations for various physical processes in complex domains demonstrate the effectiveness of DLB schemes for various large-scale related problems. A detailed performance analysis showed the need to use the DLB method to directly determine load imbalances, which, for example, are caused by individual computational efficiency depending on the composition of the local workload, scalability of individual program codes. In addition, the strong scaling experiment showed performance improvement with increasing degree of parallelism when a priori estimated computational weights are used for the initial split.

Objectives of the study. Improving the efficiency of complex modeling of various physical and technical problems has been one of the hot topics for several years now. But in this paper, the above goal is pursued, but with the use of a dynamic load balancing (DLB) scheme. A method with minimal intervention is proposed, regardless of the problem statement. Furthermore, computational weights are estimated based on performance measurements during the simulation. The approach automatically determines the appropriate weights that can be used to estimate the overall distribution of the workload.

Study object. The object of the study is high-performance computing using the method of dynamic load balancing for various physical tasks.

Research methods. The methods proposed in the thesis are a new tool in the study of the problems of load distribution on various processors. The efficiency of complex modeling with significant domain decompositions are the main point of increase for DLB.

For numerical calculations, parallel numerical algorithms are used in the work, comparison of calculated results and experimental data of other well-known authors have been performed.

Theoretical and practical value. To solve important applied problems related to numerical simulation on high-performance cluster machines, the conclusions and results of this work can be used.

The developed schemes and numerical algorithms make a direct contribution to the development of science in distributed computing and in the field of information

technology in the country. The practical value of the work lies in the fact that the developed dynamic load balancing (DLB) scheme on high-performance systems of great practical importance allows not only to obtain a significantly "fast" result compared to sequential calculations, but also expands the possibilities of implementing labor-intensive methods and algorithms. for solving important applied and fundamental problems.

Scientific novelty. In order to improve the performance efficiency of massively parallel computing in this paper a dynamic load balancing (DLB) scheme has been derived. With the help of the Hilbert space filling curve (SFC) at a rough level it become possible obtain this method with different numerical methods and different computational costs per divided cell.

Using the constructed parallel numerical algorithm, the following were performed:

- numerical study of the efficiency of high-performance computing for flow problems behind a backward-facing step;
- numerical study of the efficiency of high-performance computing when using hybrid parallel algorithms for problems of air flow in a complex nasal region;
- hybrid parallel numerical computation using various methods of domain decomposition;
- hybrid parallel numerical computation using the dynamic load balancing method;
- evaluation of the efficiency of hybrid parallel numerical computation using different methods of domain decomposition;
- efficiency evaluation of the hybrid parallel numerical algorithm using the dynamic load balancing method;
- the comparison of the obtained simulation results with numerical data and experimental data of other authors was carried out;
- the analysis of the obtained results of hybrid parallel numerical computation computation was carried out using the method of dynamic load balancing;

Provisions for Defense. The work contains the following results:

—results of a numerical study of the efficiency of high-performance computing for flow problems behind a backward-facing step

—results of a numerical study of the efficiency of high-performance computing when using hybrid parallel algorithms for problems of air flow in a complex nasal region

—results of hybrid parallel numerical computation using different domain decomposition method

—results of hybrid parallel numerical computation using dynamic load balancing method

—results of evaluating the efficiency of hybrid parallel numerical computation using various domain decomposition methods

—the results of efficiency evaluation of the hybrid parallel numerical algorithm using the dynamic load balancing method;

—the obtained simulation results were compared with numerical data and experimental data of other authors.

—the analysis of the obtained results of hybrid parallel numerical computation using the method of dynamic load balancing.

Publication and approbation of results. The results of the dissertation were published in 11 papers [116-126], of which 2 are from the list, the THOMSON REUTERS database and 4 from the list, the SCOPUS database, 6 articles are from the list recommended by the Committee for Control in Education and Science of the Ministry of Education and Science of the Republic of Kazakhstan, 1 works - in the materials of international and republican conferences.

The structure and scope of the thesis. The dissertation consists of a designation and abbreviation, an introduction, five chapters, a conclusion and a references. It is presented on 122 pages, references contain 126 items.

The main content of the work.

The first chapter provides descriptions of various technologies for high performance computing.

The second chapter has a detailed description of the analytical models of the main characteristics of the parallel algorithm.

The third chapter provides a detailed description of the mathematical formulation of the basic equations for modeling the problem.

In the fourth chapter, numerical studies of the efficiency of high-performance computing for backward-facing flow problems are considered.

In the fifth chapter, numerical studies of the efficiency of high-performance computing using hybrid parallel algorithms for airflow problems in a complex nasal region are considered.

In conclusion, the results of the dissertation work are presented.

1. HIGH PERFORMANCE COMPUTING TECHNOLOGY DESCRIPTIONS

Nowadays the need to solve complex applied problems with a large amount of computation and the fundamental limitation of the maximum speed of the "classical" ones - according to the von Neumann scheme - computers led to the appearance of multiprocessor computing systems (MVS) or supercomputers.

Vector expansion of multi-core processors and their mass production gave impetus to parallel computing. Today, multi-core processors can be found in everything from supercomputers to handheld gadgets. A sequential program written without distributing work between different cores of the central processor and without vectorization cannot reveal the full potential of the computing capabilities of the central processor (Figures 1 and 2). Therefore, mainly for improving the efficiency of the algorithm, in other words means applying parallel computing applications. Obviously there are certain parameters that affect to the choice of parallel algorithms and technical solutions. The main of them is the dimension of the spatial grid. Thus, due to the fact of using rather coarse spatial grid, multiprocessor systems with shared memory are preferable rather than using systems with distributed memory. While for implementation using multiple threads, it is expedient to prefer the implementation of using multiple processes. Moreover, the cost of synchronization of workflows and the problem of its minimization play a significant role on efficiency.

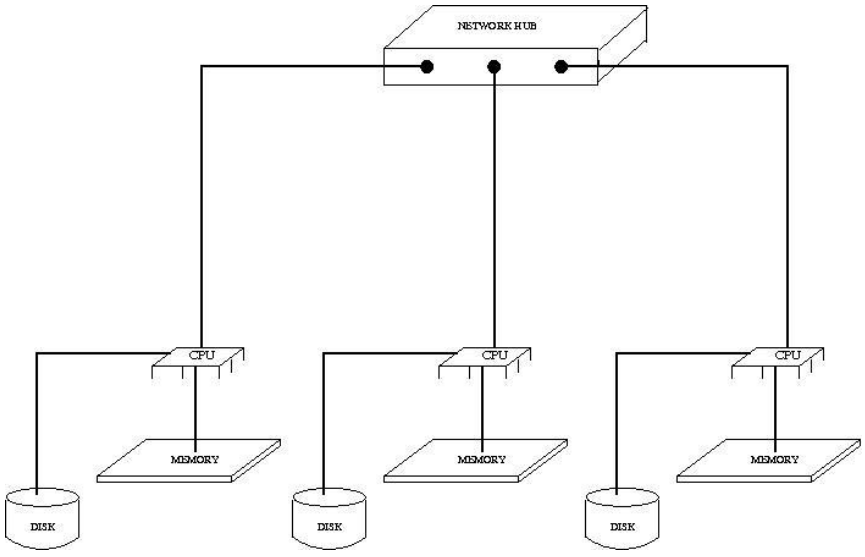


Figure 1 - The disturbed memory multiprocessor architecture.

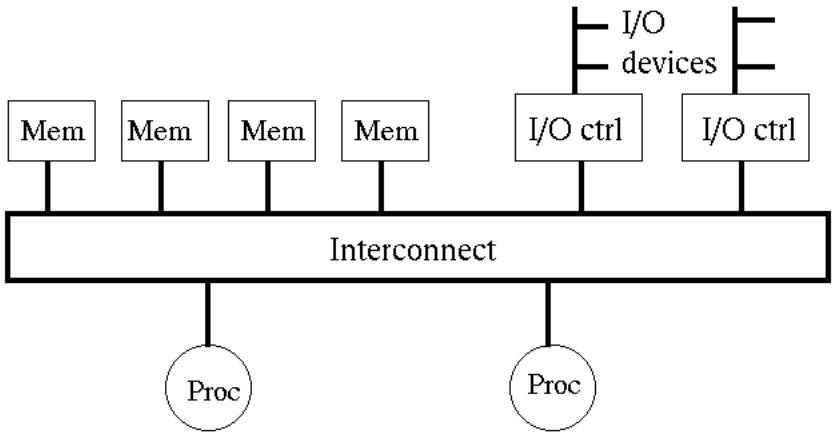


Figure 2 - The shared memory multiprocessor architecture.

It is widely known that not all programs could totally be parallelized. So that any program consists of two domains: parallel and sequential. The difference is in the performance of the processes while generation. Based on the definitions of the sequential field follows that birth, execution and completion of the program happens in the same thread. Whereas generated by the parallelization threads can be executed as well as on different processors, and on a single processor computer system. Nevertheless, the access to the processors are not parallel, sometimes it is a competition between flows. Managing competition is held by the scheduler of the operating system using special algorithms.

One of the most common methods of classification is Flynn's computer taxonomy [10], in which the main point in the analysis of computer architecture is emphasis on the methods of interaction sequences (flows) of executing commands and data to be processed. With this approach the following main types of systems can be distinguished:

	Single instruction	Multiple instruction
Single data	SISD	MISD
Multiple data	SIMD	MIMD

— **SISD (Single Instruction, Single Data)** - a system in which there is a one instruction stream and one data stream. Conventional sequential computers may be included in this type;

— **SIMD (Single Instruction, Multiple Data)** - single system command stream and multiple data streams. Such kind of class consists of multiprocessor systems that at each given time can be running the same command for processing multiple data elements; such architecture can be found in for example, multi-processor systems with a single control device. This approach was widely used in

previous years, recently it has been seldom used, only, for creation of specialized systems;

— **MISD (Multiple Instruction, Single Data)** - a system in which can be found both many instruction stream and one data stream. Still there is no consensus regarding this type of system: some experts believe that specific examples of computers that match the type of computer systems, do not exist and the introduction of this class is taken to complete the classification, while others refer to this type as, systems such as systolic computing or systems with conveyor data processing;

— **MIMD (Multiple Instruction, Multiple Data)** – a system with multiple command stream and multiple data streams. Most parallel multiprocessor systems are concerned with a similar class.

Among others, the next number of common problems encountered when using parallel computing can be considered:

— **Loss of productivity for the organization of overlapping** - according to the hypothesis of Minsky [9], acceleration which is achieved using a parallel system is proportional to the binary logarithm of the number of processors (for instance, at 1000 processors possible acceleration is equal to 10).

At this point the acceleration of parallelism of certain algorithm should be clarified. As there is a similar estimate held, regardless to the number of processors involved and percentage of parallelizing.

— **The existence of sequential computing** - in accordance with Amdahl's Law [9, 10] accelerating the computation using p processors is limited to the value

$$S \le \frac{1}{f + (1 - f)/p}$$

where f is the fraction of sequential calculations in the applicable data processing algorithms (for example, in the presence of only 10% of sequential instructions to perform calculations on the effect of parallelism cannot exceed 10 times the speed of data processing).

Summing up the above it could be declared that there could not be 100% parallel program. Therefor the share of consistent action is mainly depends on the certain percentage of sequential instructions. As a result, correctly chosen parallelization algorithm can significantly reduce the sequential parts.

— **The efficiency of parallelism has diversity of architectural design principles**. The percentage of efficiently directly depend on the percentage of fully utilizing all the features of the equipment. Disadvantage of which is the difficulty of transfer parallel algorithms and programs between the different types of systems.

In response to this observation, "homogeneity" also exists in serial computers, which means that in this case, the properties of the equipment should be taken into account for more efficient use. Moreover, today there are different types of system architectures. But along with them, other methods of parallelization are also popular (pipeline computing, multiprocessor systems, etc.). In addition, a variety of parallel

programs can be achieved using standard software for parallelization (built-in libraries, OpenMp and others)

The current software is geared primarily towards serial computers - hence the lion's share of software is written in a serial algorithm, thereby making the processing of such a number of programs for parallel systems impossible.

OpenMP is an application programming interface on shared memory devices. Therefore, parallelism is possible only where there is access to common data for all parallel processes. By dividing loops into different threads, parallelization can be achieved.

MPI is a method opposite to OpenMP in its logic, which is reflected in the name itself. That is, it is a method of programming on devices with distributed memory. Simply put, each parallel thread has its own space.

These kind of techniques are widely used in algorithmic languages like Fortran and C / C + +, as a result of description of standard and its implementation.

To organize parallel computing in distributed memory conditions (where processes operate independently of each other), there is a need to distribute the computing load and organize information interaction (data transfer) between processors. As a solution to all these options, a data interface (MPI) can be proposed.

1. Generally, these steps could be outlined for the distribution of computation among the processors:
   1) analysis of an algorithm for solving the problem
   2) providing independent information fragments of computation
   3) implementation
   4) received parts distribution of the program on different processors. One of the simple ways of covering all these steps in MPI – development of only program code running at the same time on all available processors! Obviously, this method leads to identity calculations on different processors. In order to avoid this substitute different data for the program on different processors and take into account the difference in calculation by means of identifying the process that executing the program.
2. Appropriate operations for the organization of information exchange between processors like sending/receiving and transmitting data are the most minimal variant. Not least importance should be paid to the sufficient space for communication between processors. There are many data transfer operations in the MPI which are applied by means of communication operations. This opportunity is the most powerful part of MPI (what is particularly shown by the name - MPI).

As a result attempts to create software for data transfer between processors began almost immediately with the advent of local computer networks - a series of such tools is presented, for example, Quinn (2004), Alexey L.Lastoevsky (2009) and many others. However, incompletely and incompatibility of these works were the main disadvantages. Moreover the problem of transferring software to other computer systems becomes acute. Thus lead to the active attempts by scientists to standardize the organization of messages in a multiprocessor system. The bull point in emergence

of MPI was a workshop on standards for transmission of messages among the distributed memory (the Workshop (later transformed into the international community to MPI Forum group) on Standards for Message Passing in a Distributed Memory Environment, Williamsburg, Virginia, USA, April 1992). As a result the 1994 standard MPI version 1.0 was created and even adopted. Later in 1997 MPI standard version 2.0 appeared. MPI becoming the most widely used API for delivering a required datum to another part of a program at some future time by using the future concept. Another reason of MPI popularity is its easy usage in high level languages as Fortran, Java and C++.

So let us give more deep definition for MPI. By past organization of transferring messages, it is also enabling transmission of messages and thus meets all requirements of the standard of MPI. Moreover, these tools should be organized in the libraries and be accessible. But the difference between MPI as a standard and saying MPI as a usage of it as a library in the program should be clearly separated. As these two different interpretations of them.

There are issues related to the development of parallel programs using MPI [10, 11]. Let us outline positive concepts of MPI:

— In addition to cross platform programs with MPI libraries, they greatly alleviate the problem of portability of parallel programs between different computer systems

— Nowadays there are different MPI libraries adopted every type of computer system in order to get the maximum possible extension taking into account computer equipment possibilities of using a computer;

— It could be said that MPI reduces the complexity of developing parallel programs. As standard of MPI provide the most basic operations of data transmission while MPI libraries have a large number of parallel methods.

1.1 High-Performance Computing with MPI Technology

Let us consider the concepts and definitions that are fundamental to the standard MPI.

1.1.1. The concept of a parallel program

Many parallel processes can be considered as a parallel program under MPI tools. Processes can run on different processors, but on a single processor some processes can also be located at the same time (in this case their response will be implemented with time sharing). In the extreme case, only one processor can be used to execute a parallel program - as a rule, this method is used for the initial verification of parallel programs.

Usually the same software package serves as the basis for a parallel program, that is, its copy - the code being executed must be available on all processors during the execution of the parallel program. C or FORTRAN are frequently used languages for source code, using one or another implementation of the MPI library.

At the stage of execution the number of the processors is determined and it cannot be modified during the run time of a program. Thus the number of the processor/process is called rank ranked from 0 till np-1, while the total number of them is usually called size.

1.1.2 Operation Data

The core of the MPI infrastructure is messaging operations. For synchronous operation of several processors during collective communication, there are various paired operations in MPI.

Different transmission modes like synchronous, blocking can be used to perform binary operations.

As mentioned earlier, the MPI standard is designed to implement the most basic collective operations for data transfer [10-13].

1.1.3 The concept of communicators

Combining the processes of a parallel program create groups. The communicator in MPI is used for data transfer operations, while it combines process groups and a number of additional parameters (context) to use. It is also a generating service object.

As a rule, for processes belonging to the same communicator paired data transfer operations are performed. Whereas collective operations apply simultaneously to all communicator processes. As a result, reference to the appropriate communicator is a mandatory instruction for data transfer operations in MPI.

When executing a program process, existing groups of processes and communicators can be destroyed or created as new. Different groups or communicators may perform the same process. All existing parallel software processes are part of the MPI_COMM_WORLD communicator, which is actually the default device.

Moreover to transfer data between processes of different groups, a global communicator (intercommunicator) is required.

1.1.4 Virtual topology

As mentioned earlier, between any processes of the same communicator paired data transfer operations can be performed. And all processes of the communicator have to be included in a collective operation. In this regard, the structure of a complete graph (regardless of the actual physical connections between processors) could be named as the logical topology of connections between processes.

Thus taking into account above, it is advisable to consider the logical representation of the existing communication network in the form of various topologies for the presentation and further analysis of a number of parallel algorithms.

In MPI many processes could be presented in the form of a lattice of arbitrary dimension. In this case, arrays of boundary processes can be declared neighbors and, thus, torus structures can be determined based on the lattice [10, 11].

Moreover, MPI gives a possibility to form a logical (virtual) topology of any desired type.

Sequential programming models have the following characteristics:

— relatively low productivity;
— use of standard programming languages;
— good portability of programs at the source code level.

In their terms parallel programming model is characterized by:

— the ability to achieve higher performance programs;
—use of special programming techniques;
—use of special programming tools;
—more time-consuming programming;
—problems with mobility.

1.2 High-Performance Computing with OpenMP Technology

OpenMP (Open Multi-Processing) is an open standard for parallelizing programs in the C, C ++ and Fortran languages. It describes the set of compiler directives, library routines, and environment variables that are intended for programming multithreaded applications on shared memory multiprocessor systems. The development of the standard specifications is carried out by the non-profit organization OpenMP Architecture Review Board (ARB) [1], which includes all major processor manufacturers, as well as a number of supercomputer laboratories and universities. The first version of the specification was released in 1997, it was intended only for Fortran, the next year a version for C and C ++ was released. OpenMP implements parallel computing using multithreading, in which the master thread creates a set of slave threads, and the task is distributed among them. Threads are assumed to run in parallel on a machine with multiple processors.

Tasks executed by threads in parallel, as well as the data required to perform these tasks, are described using special preprocessor directives of the corresponding language - "pragmas". Thus, Fortran code, which must be executed by several threads, each of which has its own copy of the variable N, is preceded by the following directive:! $OMP PARALLEL PRIVATE (N). The number of threads created can be controlled both by the program itself by calling library procedures, and from the outside, using environment variables.

Key elements of the standard:

— constructs for creating threads (parallel directive),

— constructs for distributing work between threads (DO / for and section directives),

— constructs for managing work with data (shared and private expressions for defining a variable memory class),

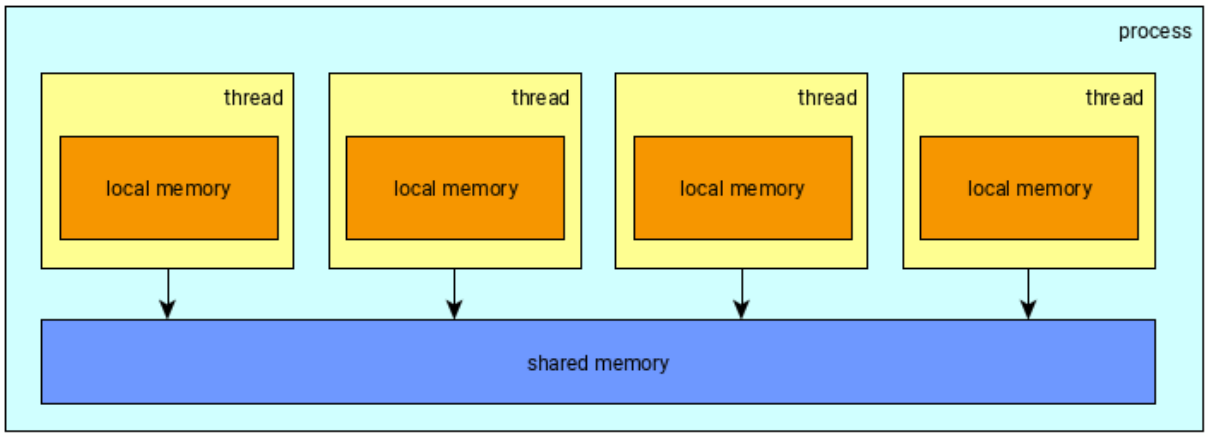
— constructs for synchronizing threads (directives critical, atomic and barrier),

— runtime support library routines (e.g. omp_get_thread_num)

— environment variables (for example, OMP_NUM_THREADS).

1.2.1 OpenMP ideology

There are many types of parallel computing systems - multicore / multiprocessor computers, clusters, systems on video cards, programmable integrated circuits, etc. The OpenMP library is only suitable for programming shared memory systems using thread parallelism. Threads are created within a single process and have their own memory. In addition, all threads have access to the process memory. It can be schematically represented in this form.



1.3 Models of parallel programming

There are various techniques, having a special purpose at different architecture of high-performance computing systems and various tools in the range of parallel programming model. Several of them are listed below.

Message-passing model

Key features of this approach:

— The program outlines several problems.

— Each task has a unique identifier.

— Sending and receiving messages the main tools of interaction.

— Multiple tasks can be performed on one processor.

— New tasks can be generated during the execution of a parallel program.

Concurrency model data

Key features of this approach:

— The program contains a sequence of one operation to the set of data structure elements operations.

— "Grits" computing is low.

— The allocation of the data is made manually.

1.4 Development of parallel algorithms

Development phases of parallel algorithm:

1. **Decomposition.**
   Ignoring architectural features of a particular computer system problem is considered from the point of parallelization. In other words, the program is divided into several sub-programs.
2. **Designing exchange (data communication) between tasks**.
   By defining communication methods for data transfer of source data and intermediate results of sub-programs manage the problem.
3. **Enlargement.**
   Combination to larger units of sub-programs could be considered in order to increase efficiency and decrease the complexity of development.
4. **Scheduling algorithms**.
   Subtask distribution among the processors. Reducing exchange time effectively use of processors – key point for choosing the method of destitution of sub problems.

1.4.1 Decomposition (segmentation, splitting)

There are various methods of decomposition. Implementation of data decomposition could be divided into following steps: data segmentation and the algorithm processing. The divided data is separated into fragments of approximately the same size. Transaction processing is coupled with a certain part of the data from which subtasks will be generated. Then the steps for determining the necessary transmission data should be defined. Duplicate calculations should be outlined in the intersection parts of the program. In order to reduce the number of exchanges number of stacks' overlapping can be increased.

Data structure with the largest size, or those that are accessed more often than others are analyzed first. Various stages of the calculation require data structures. So it can be used for both static and dynamic decomposition schemes of these structures.

1.4.2 Recursive dichotomy

The possibility of dividing into sub regions keeping complexity but decreasing the communication – this is called recursive dichotomy. First of all single dimension is considered and divided into two parts. Then recursively executes the process of portioning for each sub domain with a given a number of subtasks.

For example, for irregular grids recursive coordinate dichotomy likely to be applied. As described above the process first is performed at each step along that dimension. In addition, the method of recursive count dichotomy can be also applied

for this type of grids. In this instance the number of edges crossing the boundaries of subdomains is minimized by means of lattice topology.

1.4.3 Functional decomposition

In the cases when there is no obvious parallelization of the algorithm, functional decomposition method can be useful. By segmenting first the numerical algorithm, later the data decomposition scheme is configured according to the adopted scheme. The efficiency is ensured by the following points:

— After decomposition the number of processors at least for one is less than the number of subtasks' order;

— Needless computations and data manipulation should be avoided;

— Approximately of the same size for the sub tasks;

— Increase in a number of subtasks leading to the increase in problem (while maintaining a constant size of a subtask) is a good point as a characterized segmentation.

The size of sub problems defines granularity of the algorithm. While a number of operations in the block is a measure of the granularity. Three degrees of granularity could be outlined:

1. Fine-grained parallelism - instruction-level (less than 20 teams per block, the number of concurrent sub-tasks - from a few to several thousand, the average scale parallelism of about 5 commands per block)
2. Medium-grained parallelism - at the level of procedures. The block size is 2000 operations. Identification of such parallelism is more difficult to implement, as inter procedural dependence should be considered also. Requirements for communications are less than in the case of instruction-level parallelism.
3. Coarse-grained parallelism – level of programs (tasks). It corresponds to the execution of independent programs on a parallel computer. Coarse-grained parallelism requires the support of the operating system.

By the way the most important condition of decomposition - independence of subtasks. There are main types of independence:

— Independence of data - data that are processed by one part of the program is not modified in another part of it.

— Independent management - the order of execution of the program may be determined only at a run time (the sequence of execution is determined depending on availability management).

— The independence of the resource – provided by the sufficient computing resources.

— Independence of the conclusion - occurs if two subtasks do not produce an entry in the same variable, and the independence of I / O, if the operators input-output of two or more subtasks do not apply to a single file (or variable). For example, we can use it for solving whole Navier – Stokes equations system.

— Complete independence is usually unreachable to achieve.

1.4.4 Design Communications

Nowadays following basic types of communications could be outlined:

— local - each sub problem is associated with a small set of other sub-tasks;

— global - each sub problem is associated with a large number of other sub problems;

— structured - each sub-task and subtasks associated with it, form a regular structure (for instance, the topology of the lattice);

— unstructured - subtasks associated arbitrary graph;

— static - communication scheme does not change over time;

— dynamic - the scheme of communication varies during program execution;

— synchronous - the sender and recipient of the data coordinate an exchange;

— asynchronous - sharing data is not coordinated.

— Moreover the following recommendations for communication design could be given:

— In order to avoid poor scalability subtasks should have approximately the same number of communications;

— Constantly use local communication as much as possible;

— And try to parallelize communication as much as possible.

1.4.5 Enlargement

Taking into account the results of the previous two steps, at the stage of agglomeration combination of them occurs so that their number corresponds to the number of processors.The agglomeration demands the following requirements:

— overheads of communication must be reduced;

— if the consolidation requires duplication of the calculations or data, this should not lead to loss of performance and scalability of the program;

— the resulting problems should have roughly the same complexity;

— scalability must be maintained;

— the possibility of parallel execution must be maintained;

— labor cost of development must be reduced.

1.5 High-Performance Computing with OpenMP/MPI Technology

Almost linear scaling in the number of MPI processes and in time can be noted regarding the efficiency of algorithms with natural data distribution (explicit schemes for parabolic equations, separation of variables for elliptic equations, domain decomposition methods, etc.). To improve efficiency on one computing node with OpenMP, it is necessary to introduce local arrays for each of the sweats, as shown, for example, in [1, 2]. The above described can be achieved by applying an algorithm of technology similar to MPI, but do not forget about the need for shared memory on the node.

The concept of data distribution between computing nodes, similar to MPI, is used in this hybrid implementation of MPI/OpenMP. Which means there is one MPI

process per node. At each of the computing nodes, an important point is the exchange of data and the organization of the overlay of calculations. Three timed data exchanges occur at each step of the circuit. Moreover, each previous exchange consists of more runs to find the necessary stream components. The main idea is to send small arrays partly during the passages, and not in one large array after all passes in a certain direction. Moreover, while one part is busy sending already processed data, the other part is busy processing. That is, the solver threads are busy performing runs of the relevant data.

Hybrid scheme concept: with postmen for the case of two MPI processes with n+1 OpenMP threads (postman + n solvers) on each of the processes. Data corresponding to MPI processes numbered 0 and 1 are separated by a horizontal plane. In this example, on each of the processes, the data is divided into eight blocks, each block corresponds to the execution of part of the runs. According to the proposed idea, one postman and n solvers work simultaneously on each of the processes, which together perform sweeps for each of the blocks. Numbers 0 denote blocks processed by solvers at process 0, and 1 - blocks processed by solvers at process 1. After several blocks are processed in parallel at each of the two processes, the postmen (one at each of the processes) simultaneously send the already processed blocks, and the solvers process the next blocks, etc. All available blocks that have been processed by all MPI processes by the current moment are involved in the data exchange each time. The order of processing blocks is chosen in such a way that at the moment when the last blocks are exchanged, the solvers could already perform the next step of the algorithm, in this example, these are sweeps along the other direction. To do this, either during calculations, or during data exchanges, or at the stage of assembling the right-hand side, it is necessary to change the local numbering within each block of attacks so that the corresponding data has already been “sliced” along the required direction.

1.6 High-Performance Computing Performance Assessment

This relation can express the speedup of the parallel algorithm:

$$Sp = \frac{T1}{Tp},$$

where $T1$– is the execution time of the program on one processor, Tp – the execution time of the program on p processors.

Due to the acceleration it is possible to compare the behavior of algorithm with one and p processors. The effectiveness of the parallel algorithm is closely related to the acceleration. The efficiency of a parallel algorithm:

$$E_p = \frac{Sp}{p}$$

Since acceleration $Sp \le p$, the efficiency of the algorithm $E_p \le 1$. A comparison of the acceleration values obtained numerically with theoretical calculations based on the Amdal and Gustafson-Barsis acons was also carried out. The application of Amdahl's law makes it possible to determine the maximum theoretical acceleration of a parallel solution relative to a serial code (Figure 3). It follows from this law that the acceleration of program execution by parallelizing the program on a certain number of processors is limited by the time required to perform its sequential operations.

$$Sp \le \frac{1}{\alpha + \frac{1-\alpha}{n}}$$

where α – the proportion of work that can be parallelized; n - number of computational processors.

Whereas Gustafson's law gives an estimate of the theoretical acceleration of the calculation of the problem with an increase in the volume of the problem, which leads to a decrease in the share of the subsequent part of the problem (Figure 4). For Gustafson's law, the acceleration is called the scaling acceleration.

$$Sp = n + (1-n) * \alpha,$$

where α – the proportion of work that can be parallelized; n – number of computational processors.

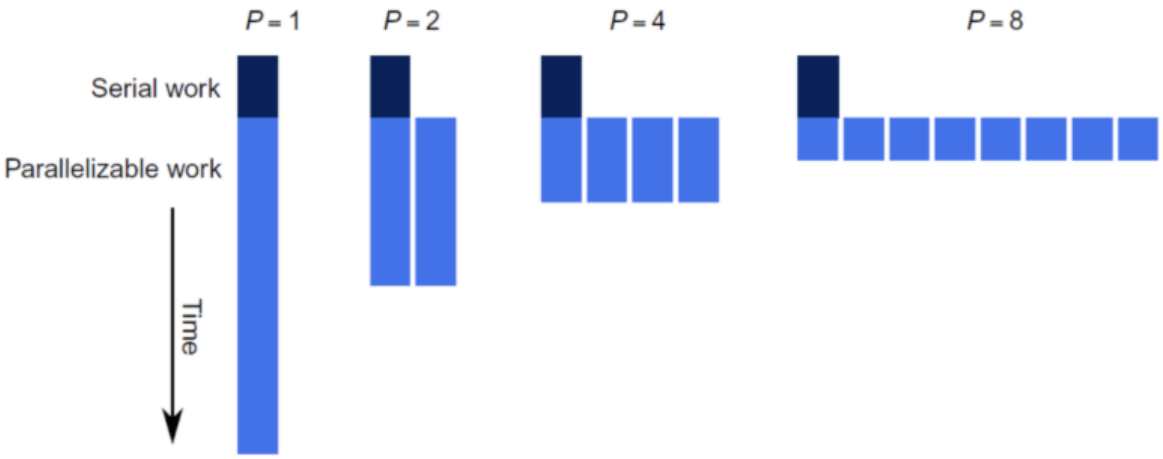


Figure 3 – Use of law of Amdahl.

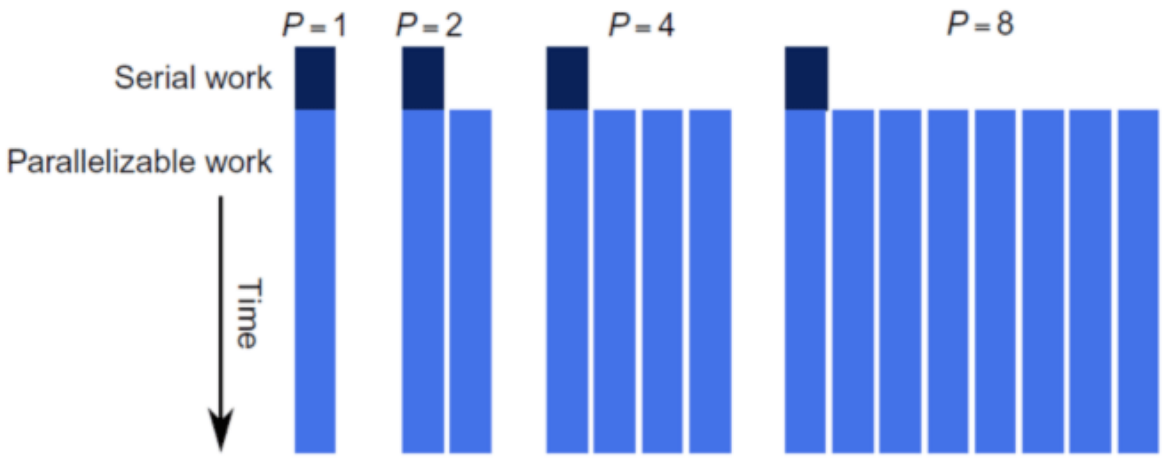


Figure 4 – Gustafson-Barsis' Law; by increasing the problem size with P, the serial portion increases slowly and speedup increases as processes are added.

When making a conclusion about the laws of Amdahl and Gustafson-Barsis, it is necessary to note that they are both correct. Amdahl's law should be applied if you need to run a program faster while maintaining the same load. Whereas at the same time, but with a greater load, one can notice the influence of the Gustafson-Barsis law.

2 ANALYTICAL MODELS OF THE MAIN PARALLEL ALGORITHM CHARACTERISTICS

2.1 Topology of data transfer processes

As you know, a point-to-point operation is a function for the interaction of no more than two program processes.

In this work MPI MPI_Sendrecv (communication function) was used to send and receive messages from one processor to another.

C/C++:

int MPI_Sendrecv (void * sendbuf, int sendcount, MPI_Datatype sendtype, int dest, int sendtag, void * recvbuf, int recvcount, MPI_Datatype recvtype, int source, MPI_Datatype recvtag, MPI_Comm comm, MPI_Status * status*)*,

Input Parameters

sendbuf
Initial address of send buffer (choice).
sendcount
Number of elements to send (integer).
sendtype
Type of elements in send buffer (handle).
dest
Rank of destination (integer).
sendtag
Send tag (integer).
recvcount
Maximum number of elements to receive (integer).
recvtype
Type of elements in receive buffer (handle).
source
Rank of source (integer).
recvtag
Receive tag (integer).
comm
Communicator (handle).

Output Parameters

recvbuf
Initial address of receive buffer (choice).
status
Status object (status). This refers to the receive operation.

By itself, the MPI_Sendrecv procedure sends sendcount and sendtype elements from the sendbuf array. This sends a message identifier with the dest number in the communicator during the sendtag process. Moreover, in the comm communicator, no more than recvcount elements of the recvtype type with the identifier of the received message recvtag are received into the recvbuf array. The state parameter is stored for the received message. Whereas the same processor can be the recipient and the sender at the same time. However, an intersection in the process address space of received and transmitted data is not allowed. The MPI_SENDRECV and MPI_SENDRECV commands can be used to receive and send in the usual way. In this case, different data types and different lengths of transmitted messages are allowed.

The topology of data transfer processes is built from certain communications between the processors of a computer system. The following schemes of communication processes can be noted:

1) designed for a small number of processors - a complete graph (any pair of processes has a direct connection);

2) clear numbering and communication with only two neighboring processors (except for the first and last) - a ruler;

3) when connecting the first to the last processors from the previous topologies, a ring graph was taken;

4) if all processors have a connection with some control processor, it is called a star topology;

5) The most easily implemented topology. It is effectively used in parallel computations of numerical algorithms, as well as in partial derivatives. The structure itself is a two- or three-dimensional rectangular grid - a lattice.

6) A particular version of the lattice structure is a hypercube. The peculiarity is that there are only two processors for each dimension of the grid (i.e., the hypercube contains 2N processors of dimension N).

For this work, the topology of the lattice was analyzes. The Hockney model [14] was invoked to estimate the execution time of passing byte messages between two processes:

$$T\_comm = \alpha + \frac{n*l}{\beta},$$

where α – is the latency (the length of the message preparation for transmission), the length of the transmission path (byte), and β – the bandwidth of the data link (byte/second).

The length of the data transfer path depends on the type of process topology. For the lattice topology $l = \sqrt{p}$.

2.2 Analytical models for estimating the of energy consumption, memory and time of collective operations characteristics

Information exchange can be divided into three types [2]:

1. Two-way exchanges (differentiated exchanges, point-to-point exchanges). Two processes are involved in this exchange: the message sending function (MPI_Send) and the message receiving procedure (MPI_Recv).

2. One-way connections. Obviously, here only one of the two branches actively participates in the operation and implements remote access to the memory of the second process for reading or writing (Remote Memory Access ĮI RMA). MPI_Put and MPI_Get are prime examples of such operations. Moreover, there are similar operations in the SHMEM, BSP, and Uniied Parallel C libraries. In some circumstances, such functions make it possible to avoid unwanted process synchronization.

3. Participation of all program processes - collective operations (global, group operations, collective communications). There are two types of root and non-root operations. The first type includes: broadcast transmission (one-to-all, one-to-all) and collector reception (all-to-one, all-to-one). Whereas to non-root (unrooted) exchanges of the type "all-to-all" (all-to-all). For most types of algorithms, the execution time of collective operations plays a very important role and their scalability directly depends on this. To implement collective operations, a two-way exchange is used. There are a number of ways for collective exchanges in the MPI libraries: by ring, recursive doubling, recursive halving (recursive halving), Brook [23]. Moreover, for pairwise exchange algorithms and algorithms, the ordering of branches into trees of different types: binomial trees (binomial tree); balanced -ary trees, plane trees (lat-tree, linear tree), chains (pipeline, -chains) [24].

The most important advantage of MPI is the existence of more complex types of interactions, in which all processes associated with a particular communicator take part. It is these interactions that are called collective.

When writing MPI programs, collective operations are often used. For example, if it is necessary to distribute a variable or an array from one processor to all other processors, or vice versa, it is not advisable to collect information in the same place using point operations. While the use of collective - the best option.

Below are some of the distinguishing features of collective operations:

1) Lack of intersection of point-to-point operations with collective ones;

2) After the completion of its part, the subroutine in each process returns from the collective operation, but this is not at all a sign of the completion of the entire collective operation. It follows from this that collective operations are a blocking operation.

3) the need for the same number of sent and received elements;

4) similar to the point above, but the point is in data types;

5) absence of message identifiers.

There is also the concept of global computational operations. These include operations of addition, finding the maximum, etc. MPI_Reduce, MPI_Allreduce, MPI_Scan, MPI_Reduce_scatter, can be attributed to the above type.

The syntax of MPI_Allreduce on C/C++:
int MPI_Allreduce(const void *sendbuf, void *recvbuf, int count,
MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

Input Parameters

sendbuf
starting address of send buffer (choice)
count
number of elements in send buffer (integer)
datatype
data type of elements of send buffer (handle)
op
operation (handle)
comm
communicator (handle)

Output Parameters

recvbuf
starting address of receive buffer (choice)

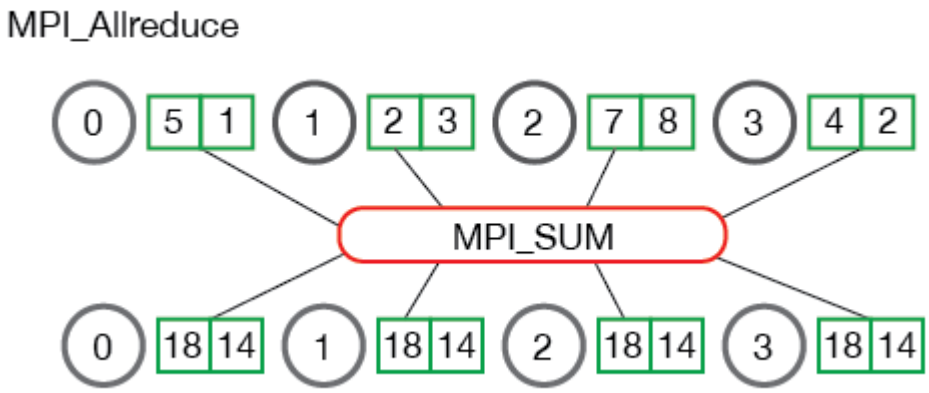


Figure 5 – The usage of MPI_AllReduce function.

There is a possibility of deadlocks given circular dependencies. Since when sending-receiving both can be the same processor. Naturally, there are ways to avoid this situation. For example, the use of various blocking techniques (it is possible to divide processors into even and odd ones). In the collective exchange of information, analytical models such as time, power consumption, and memory models can be used to evaluate the performance of an algorithm.

Thus, for the analytical study of the execution time of the collective work algorithm, the LogGP communication model [17] was used. Plus, the model applies to both short and long transmitted messages. The LogGP collective operation execution time estimation model is described by the following parameters:

L – is the maximum value of the latency of the message transmission of one processor to another (latency);

o – the time interval during which the processor can not perform other operations, i.e. When the processor is busy transmitting or receiving a message (overhead);

g – is the minimum time interval between consecutive transmissions or reception of a short message, 1/g is the capacity of the communication channel for short messages (gap);

G – is the minimum time interval between consecutive transmissions or reception of one byte of a long message, 1/G is the capacity of a communication channel for long messages (gap per byte);

P – is the number of processors in the system.

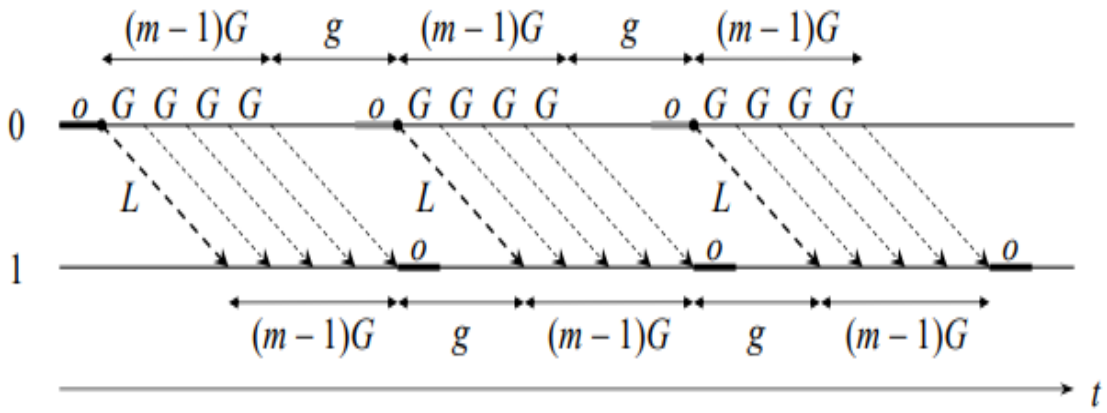


Figure 6 - Spatial-temporal scheme of processor interactions in LogGP model: processor 0 executes three message transferring to processor 1.

On figure 6 shows a space-time diagram of processor 0 sending three long messages in sequence to processor 1 using the LogGP model. That is, at time o processor 0 has completed the transmission of the first byte of the message to the network, at time o + G the second byte of the message has been sent to the network. Moreover, the last byte of the first message will be sent to the network.

o + (m - 1) G. Delivery of each byte of the message from processor 0 to processor 1 takes L units of time. The first byte of message processor 1 arrives from the network at time o + L. Reception of the entire message processor 1 ends at time o + L + (m - 1)G + o.

The sum of dynamic and static energy [18, 19] can be represented as a model of energy consumption. Everyone knows that static energy is the energy consumed by the system when the mode is on, but in an inactive state. Whereas the energy consumption for computing, or receiving - sending messages, is called dynamic. Considering only the energy for message transmission, but not spent on calculations

on local processors, studies were carried out. To create communication energy, it was assumed that each message spends a fixed energy e. Let's introduce the notation for the energy required to transfer each byte from the source memory to the target memory as E.

So, the total energy consumption of the collective operation

$$L = T \cdot P + D,$$

where *T* is the collective operation time (based on *LogGP*), *P* is the leakage power, *D* is the dynamic energy consumption model. In this paper, dynamic energy models are the sum of all the dynamic energies consumed by each processor.

To predict the used memory, a simple memory model is considered. It assumes that each sent message is explicitly specified as a descriptor in the upload operation. For example, that these descriptors have a constant size d. This descriptor size is independent of the actual size of the message being sent (received). The maximum memory required for any process is calculated.

2.3 Optimization of collective operation algorithms

The exchange of messages between processors is carried out by a directed graph. In other words, a certain virtual topology. The most popular are distribution algorithms, specialized algorithms, and trees in various forms. The latter can be used to implement any collective communications and the transmission of messages is standard from parent to child.

Consider a flat tree (FT) [20], where messages are sent directly. This algorithm is one of the simplest. Figure 6 shows an example of such a tree for a personalized or personalized operation. Green squares on the edges of the connection denote the transmitted data of size s. Annotations in this and the following figures indicate the end time of the processes in the example. In all figures, it was assumed that the data is sent to the children of the process in order, starting from the very left. Although the simplicity of the algorithm is a clear advantage, its sequential communication limits performance.

To calculate the execution time for such an operation, the LogGP model uses the formula:

$$T\_FT = L + oP + sG\,(P - 1) = (o + sG)\,P - O\,(s)$$

Dynamic energy consumption for this algorithm is presented as

$$D\_FT = (P - 1)\,(e + sE) = P\,(e + sE) - O\,(s)$$

The maximum storage in the root of the tree is (Figure 7)

$$M\_FT = d\,(P - 1)\,.$$

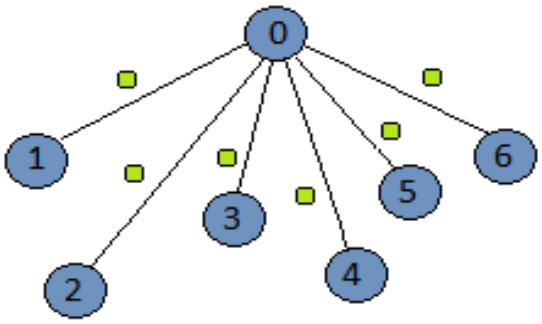


Figure 7 – The implementation of the collective algorithm of a flat tree (FT) with the number of processes (P=7).

Tree topologies are popular for root collective operations. In most cases, ordinary trees are used (when the number of child nodes is the same). This view gives superior to flat tree topics, performing collective operations, processes simultaneously perform interactions and, thus, achieve higher performance. Figures 8a and 8b show the transmission of personalized and non-personalized messages using the binary tree algorithm. Whereas the execution time of the k-dimensional tree algorithm of a non-personalized operation in the LogGP model is calculated by the formula

$$T\_KT = (L + k\,(o + sG) + o)[\log_k P] = (L + ko + ksG)\log_k P - O(s) + \log_k P \cdot O(1)$$

Dynamic energy consumption for this algorithm is presented as

$$D\_KT = (P - 1)\,(e + sE) = P\,(e + sE) - O\,(s)$$

The maximum required storage in the root node of the tree is estimated by the formula

$$M\_KT = kd,$$

since each process sends a message to at most k-child computing nodes.

Moreover, the execution time of the algorithm in the LogGP model for personalized messaging using the k-dimensional tree algorithm is calculated by the formula

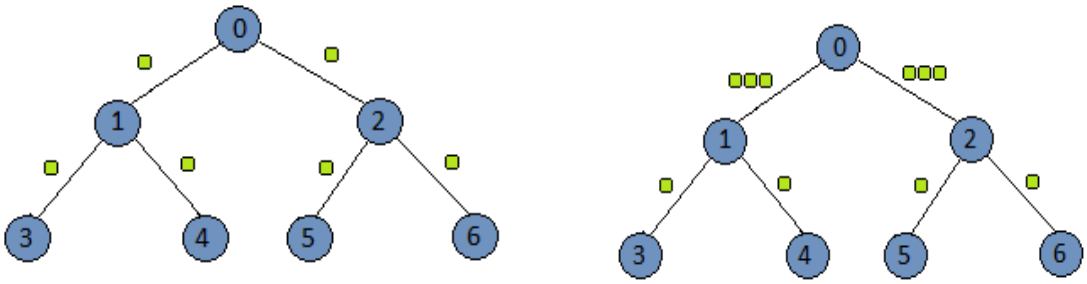
$$\overline{T\_KT} = [\log_k P](L + o(k + 1)) + sG\sum_{i=0}^{[\log_k P]} ([\log_k P) - j)k^i = (L + ko)\log_k P + sGP \cdot O(1) + O(\log P - s)$$

Dynamic energy consumption (for large k)

$$\overline{(D\_KT)} = e(P - 1) + sE \cdot k^{[\log_k P]}\sum_{i=0}^{[\log_k P]-1} ([\log_k P] - i)\frac{1}{k^i} \approx P\,(e + sE\log_k P) + O\,(sP),$$

and memory consumption is the same as non-personalized data transfer

$$\overline{M\_KT} = kd,$$



a) b)

Figure 8 – The implementation of the collective binary tree algorithm (KT): a) personalized transfer; b) non-personalized transfer.

The very famous Butterfly Graph (BF) [21] demonstrates a binary scheme for fast data exchange between all processes, with the number of processes P being a power of two. The implementation of the BF algorithm consists of steps, where P is the number of processors. At level k, one can observe data exchange between streams spaced 2k apart. On figure 9b and 9c, one can see the scheme of the BF algorithm with the number of nodes P = 8 for personalized and personalized messaging, respectively.

The execution time of the BF algorithm of non-personalized data transfer in the LogGP model can be represented by the formula

$$T\_BF = (2o + sG + L)\log_2 P$$

dynamic energy consumption

$$D\_BF = (e + sE)P\log_2 P$$

with consumed memory $M\_BF = d\log_2 P$.

The recursive doubling algorithm [22] as well as the Brooks algorithm [23] implement the BF scheme for personalized messaging. The execution time of these personal algorithms can be represented by the formula below:

$$\overline{(T\_BF)} = (2o + L)\log_2 P + Gs(P-1) = (2o + L)\log_2 P + sGP - O(s),$$

dynamic energy consumption can be modeled as

$$\overline{D\_BF} = eP\log_2 P + sE(P-1)P = P(e\log_2 P + sEP) - O(sP)$$

and their need for memory

$$\overline{M\_BF} = d\log_2 P.$$

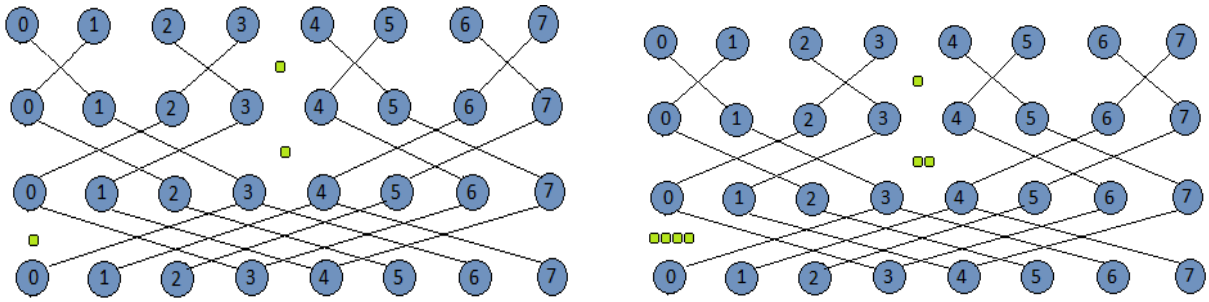
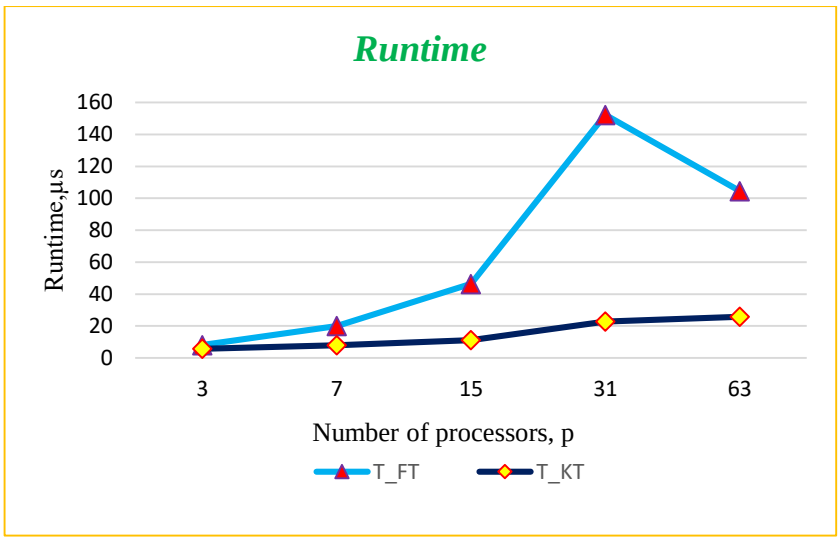


Figure 9 – The implementation of the collective algorithm Butterfly (BF): a) personalized transfer; b) non-personalized transfer.

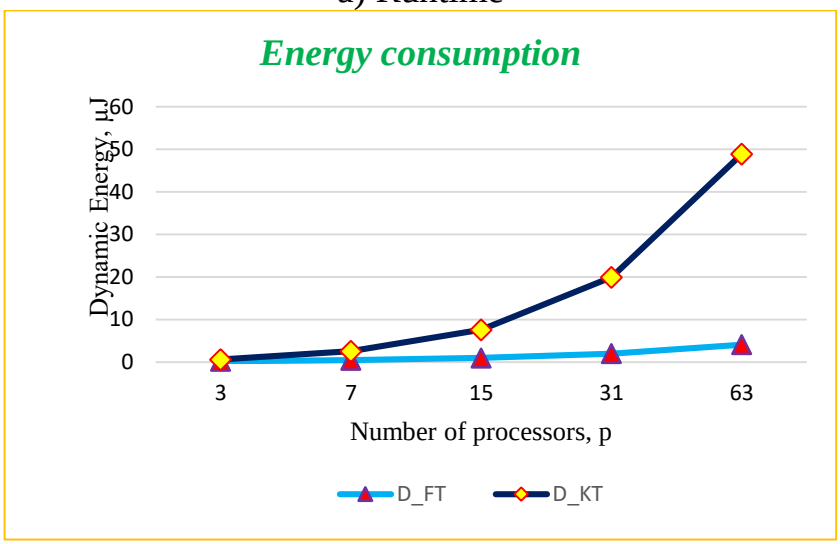
Based on collective algorithms, to be more precise, a flat tree (FT) and a regular tree (KT), the time, energy and memory costs for the execution of the collective MPI_Reduce procedure were considered. The LogGP parameters used the previously described characteristics measured for the Intel Phi processor using parallel MPI technology: L=6 μs, o=4.7 μs, G=0.73 ns/B [24]. The parameter e, which is responsible for the expenditure of the necessary energy for the exchange of messages between the processors, has the value e=16.5 pJ. Moreover, the energy E required to transfer each byte of a message from the source memory to the destination memory was equal to E=8.1 nJ/B [25-30].

Summing up, it can be said that the optimal collective algorithms with respect to the execution time of the algorithm demonstrate non-optimal dynamic energy costs. Moreover, the energy cost of the fastest algorithm is asymptotically higher compared to the energy cost of the slower algorithm. In addition, it is shown that large amounts of memory are needed for optimal algorithms in relation to the energy consumed. On figure 10 shows graphs of execution time, power consumption, and memory for a non-personalized MPI_Allreduce collective operation using the Flat Tree (FT) and Binary Regular Tree (KT) algorithms. From the results of the runtime graphs, it can be said that the binary regular tree (KT) algorithm is a more rational option regarding the execution time of the flat tree (FT) algorithm. Nevertheless, the binary regular tree (RT) algorithm performed worse than planar trees in terms of the energy model for a large number of processes. But the opposite can be said in the issue of memory consumption by algorithms. This can be explained in direct proportion to the number of child computations k nodes. As a result, it can be admitted that each of the algorithms has its own advantages and depends on certain metrics [11, 31–36].

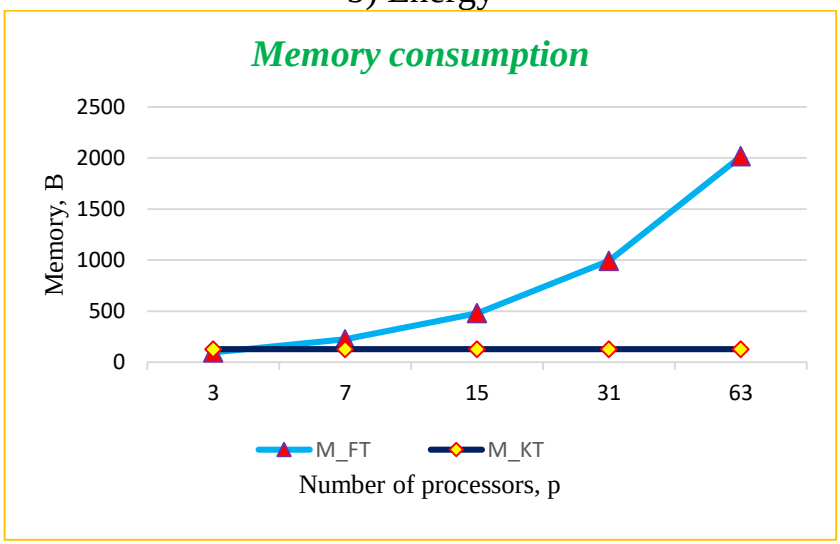
Schedules were compiled to perform the collective data exchange of the MPI_Allreduce operation using two different collective algorithms (Figure 11).



a) Runtime

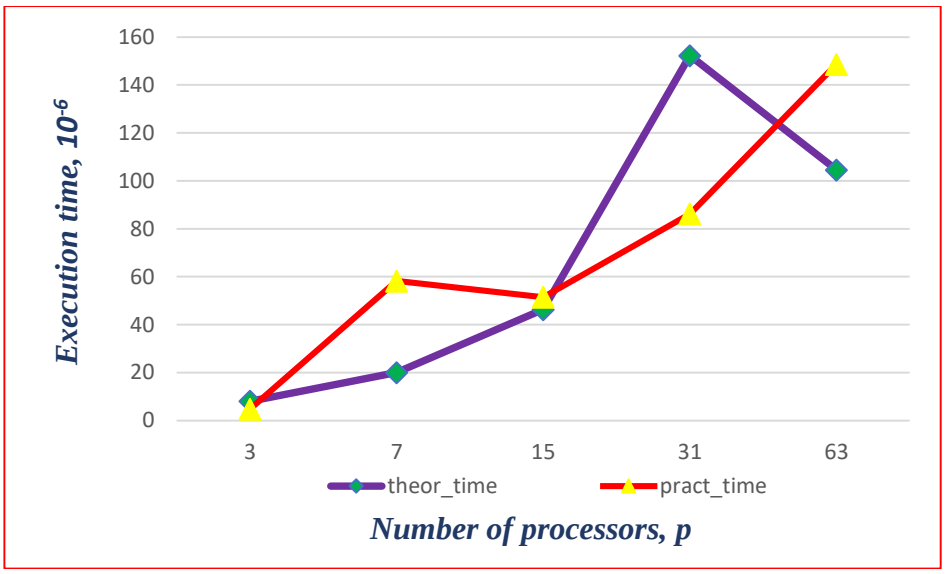


b) Energy

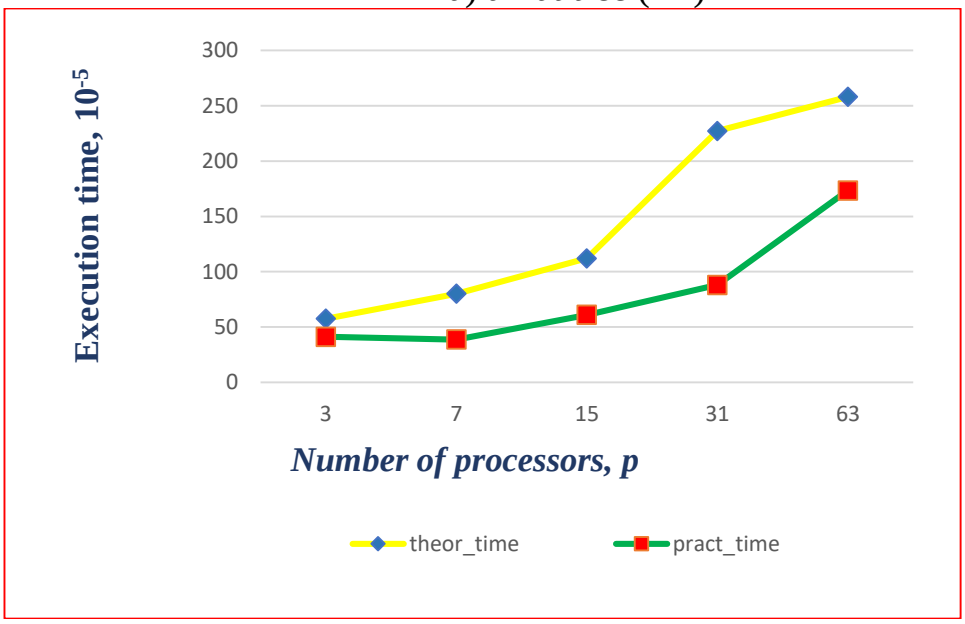


c) Memory

Figure 10 – The execution time, energy consumption and memory dependence of the non-personalized MPI_Allreduce function by using the flat tree (FT) algorithm and the binary regular tree (KT).



a) a flat tree (FT)



b) a binary regular tree (KT)

Figure 11 – The non-personalized MPI_Allreduce function: a) a flat tree algorithm (FT); b) a binary regular tree (KT).

Figure 11 compares the theoretical data with the obtained collective work time for two different algorithms based on the LogGP communication model. So it can be noted the smaller time spent on the MPI_Allreduce operation for the flat tree algorithm.

Based on the data in Table 1, it can be noted the dependence of the main characteristics of the parallel algorithm, such as execution time, power consumption, and memory of the non-personalized MPI_Allreduce collective operation using the flat tree (FT) algorithm and the binary regular tree (BT). Moreover, in this table, it is necessary to emphasize the merits of each algorithm, depending on the choice of evaluation metric.

Table 1 – Comparative analysis of the simulation time, energy and memory of sequential and parallel calculations for different sizes.

	Flat tree algorithm			k-regular tree algorithm		
Number of processors	Runtime	Energy	Memory	Runtime	Energy	Memory
3	8.09669	0.19445	96	5.74493	0.617093	128
7	20.0272	0.453715	224	8.010156	2.55021	128
15	46.348	0.972248	480	11.20013	7.60487	128
31	152.254	2.000931	992	22.783	19.9296	128
63	104.542	4.08344	2016	25.8021	48.8656	128

3 BASIC EQUATIONS OF HYDRODYNAMICS FOR MATHEMATICAL MODELING OF PHYSICAL PROCESSES

By now, it is becoming obvious that all the problems that arise in aerodynamics and hydrodynamics in the numerical solution of the Navier-Stokes equations are unlikely to be solved even when using the most powerful computers with tens and even hundreds of billions of operations per second. Therefore, in connection with their ever-increasing use in solving scientific and technical problems, it is important to provide the greatest possible scientific and practical base. This is facilitated by the deep penetration of mathematical modeling methods into a particular subject area.

3.1 Law of mass conservation. Continuity equation.

In the dynamics of a discrete system of material points, dynamic characteristics, including inertial ones, were attributed to individual points of the system. In the case of an absolutely rigid body, both total characteristics, but related to sharply defined areas of the rigid body, and forces concentrated at specific points of the rigid body, moments of forces were considered. In the dynamics of continuums (liquids, gases, elastic and other "solid" deformable bodies), this approach is replaced by the assignment of continuous distributions of dynamic and, in general, physical quantities over a continuum, characterized by the distribution density of these quantities.

The first example of such a problem can be the mass distribution density in the form of the limit of the ratio Δm of the mass of a small volume $\Delta\tau$, containing a given point M, to the volume $\Delta\tau$, when the latter tends to zero, shrinking to the point M. The ratio $\frac{\Delta m}{\Delta\tau}$ is called the average density in the volume $\Delta\tau$, and the limit of this ratio $\Delta\tau \to \infty$ at is the density of the medium at a given point M and is denoted by the letter ρ, so that

$$\rho = \lim_{\Delta\tau \to 0} \frac{\Delta m}{\Delta\tau}$$

It contracts to the point M and get

$$\rho = \lim_{\Delta\tau \to 0} \frac{\Delta m}{\Delta\tau} = \frac{\delta m}{\delta\tau} \qquad (1)$$

This equation (1) is followed by expressions for the mass of an elementary volume in terms of the density ρ

$$\delta m = \rho \delta\tau \qquad (2)$$

and the mass τ of the finite volume allocated in the medium τ

$$m = \int_{\tau} \rho \delta \tau$$

Differentiating equation (2) with time t, it obtains

$$\frac{d}{dt}(\delta m) = \frac{d}{dt}(\rho \delta \tau) = \frac{d\rho}{dt}\delta_{\tau} + \rho \frac{d}{dt}(\delta \tau) = 0$$

Considering that the mass does not change with time.
Let's use the definition

$$\frac{d}{dt}(\delta \tau) = \mathrm{div} V \cdot \delta \tau$$

Given the above expression, it will be possible to rewrite the equation in this form

$$\frac{d\rho}{dt}\delta \tau + \rho \mathrm{div} V \delta \tau = 0$$

Finally, it can be written in the form

$$\left(\frac{d\rho}{dt} + \rho \mathrm{div} V\right)\delta \tau = 0 \Rightarrow \frac{d\rho}{dt} + \rho \mathrm{div} V = 0$$

Expanding the equations taking into account the total derivative, it will be possible to write in the form

$$\frac{d\rho}{dt} = \frac{\partial \rho}{\partial t} + V \mathrm{grad} \rho = \frac{\partial \rho}{\partial \mathrm{t}} + u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y} + w\frac{\partial \rho}{\partial \mathrm{z}}$$

Then the continuity equation can be written in the form

$$\frac{\partial \rho}{\partial t} + V \mathrm{grad} \rho + \rho \mathrm{div} V = 0.$$

Now, it uses these properties

$$\mathrm{div}(\rho V) = V \mathrm{grad} \rho + \rho \mathrm{div} V$$

In the final form, the continuity equation for the compressible case will look like

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho V) = 0.$$

For the incompressible case, using the property $\frac{\partial \rho}{\partial t} = 0.$, the continuity equation can be rewritten in the form

$$\operatorname{div}\rho V = 0. \text{ or } \operatorname{div} V = 0.$$

In the final case, the continuity equation for the incompressible case looks like this:

$$\operatorname{div} V = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0.$$

3.2 The momentum theorem. Momentum equation.

According to the general theorem on the change in the main vector of the system's momentum: the time derivative of the main vector of the system's momentum is equal to the main vector of external forces applied to the system. The individual derivative of the main vector is the amount of motion of the liquid volume and surface forces applied to particles located respectively in the volume and on the surface bounding it.

$$\delta \mathrm{k} = \rho V \delta \tau$$

$\delta \mathrm{k}$ - the amount of movement.

The main momentum vector in the entire volume τ is equal to the sum or integral of the elementary quantities of motion

$$\mathrm{k} = \int_{\tau} \rho V \delta \tau$$

The main vectors of external volumetric and surface forces will be respectively equal to

$$\int_{\tau} \rho F \delta \tau \quad \text{and} \quad \int_{\sigma} p_n \delta \sigma$$

Thus, just by the momentum change theorem, it leads to the equality

$$\int_{\tau} \rho V \delta\tau = \int_{\tau} \rho F \delta\tau + \int_{\sigma} p_n \delta\sigma$$

Differentiate with respect to time the first term on the left of the equation and obtain

$$\frac{d}{dt}\int_{\tau} \rho V \delta\tau = \int_{\tau} \rho F \delta\tau + \int_{\sigma} p_n \delta\sigma \qquad (3)$$

On the left side of equation (3), it performs the transformation

$$\frac{d}{dt}\int_{\tau} \rho V \delta\tau = \int_{\tau} \frac{d}{dt}(\rho V \delta\tau) = \int_{\tau} \rho \frac{dV}{dt}(\delta\tau) + \int_{\tau} V \frac{d}{dt}(\rho \delta\tau)$$

Using the property for the last term $\frac{d}{dt}(\rho\delta\tau) = \frac{d}{dt}(\delta m) = 0.$, it rewrites the above equation in this form

$$\frac{d}{dt}\int_{\tau} \rho V \delta\tau = \int_{\tau} \rho \frac{dV}{dt} \delta\tau$$

In the last term on the right-hand side of equation (3), it makes a change and, according to the Gauss theorem, it has results equivalent to each other

$$\int_{\sigma} p_n \delta\sigma = \int_{\sigma} (n_1 p_1 + n_2 p_2 + n_3 p_3) \delta\sigma$$

It uses the Gauss theorem and obtain

$$\int_{\sigma} (n_1 p_1 + n_2 p_2 + n_3 p_3) \delta\sigma = \int_{\tau} \left(\frac{\partial p_1}{\partial x_1} + \frac{\partial p_2}{\partial x_2} + \frac{\partial p_3}{\partial x_3} \right) \delta\tau$$

or the above equation can be written in this form

$$\int_{\sigma} p_n \delta\sigma = \int_{\sigma} nP \delta\sigma = \int_{\tau} \mathrm{div} P \delta\tau$$

It substitutes the result obtained into equation (3) and obtain

$$\int_{\tau} \rho \frac{dV}{dt} \delta\tau = \int_{\tau} \rho F \delta\tau + \int_{\tau} \mathrm{div} P \delta\tau$$

Making some transformations

$$\int_{\tau} (\rho \frac{dV}{dt} - \rho F - divP)\delta\tau = 0$$

taking into account the arbitrariness of the volume of integration τ, it arrives at the following two possible expressions for the momentum equation of a continuous medium with stresses:

$$\rho \frac{dV}{dt} = \rho F + \frac{\partial p_1}{\partial x_1} + \frac{\partial p_2}{\partial x_2} + \frac{\partial p_3}{\partial x_3}$$

In tensor form, the momentum equations look like this:

$$\rho\left(\frac{\partial V}{\partial t} + V_k \frac{\partial V_i}{\partial x_k} \right) = \rho F_i + \frac{\partial p_{ki}}{\partial x_k}$$

In a rectangular Cartesian coordinate system, the momentum equations in expanded form has the following form

$$\rho\left(\frac{\partial V_1}{\partial t} + V_1 \frac{\partial V_1}{\partial x_1} + V_2 \frac{\partial V_1}{\partial x_2} + V_3 \frac{\partial V_1}{\partial x_3} \right) = \rho F_1 + \frac{\partial p_{11}}{\partial x_1} + \frac{\partial p_{21}}{\partial x_2} + \frac{\partial p_{31}}{\partial x_3}$$

$$\rho\left(\frac{\partial V_2}{\partial t} + V_1 \frac{\partial V_2}{\partial x_1} + V_2 \frac{\partial V_2}{\partial x_2} + V_3 \frac{\partial V_2}{\partial x_3} \right) = \rho F_2 + \frac{\partial p_{12}}{\partial x_1} + \frac{\partial p_{22}}{\partial x_2} + \frac{\partial p_{32}}{\partial x_3}$$

$$\rho\left(\frac{\partial V_3}{\partial t} + V_1 \frac{\partial V_3}{\partial x_1} + V_2 \frac{\partial V_3}{\partial x_2} + V_3 \frac{\partial V_3}{\partial x_3} \right) = \rho F_3 + \frac{\partial p_{13}}{\partial x_1} + \frac{\partial p_{23}}{\partial x_2} + \frac{\partial p_{33}}{\partial x_3}$$

3.3 Navier-Stokes equations. Newton's hypothesis.

The generalized Newton's law gives a linear relationship between the stress tensor and the strain rate tensor, which is expressed in the case of an isotropic medium by the tensor relation

$$P = a\dot{S} + bE \qquad (4)$$

the expression of the generalized Newton's law for an incompressible viscous fluid looks like

$$P = 2\mu\dot{S} - pE = -pE + 2\mu \operatorname{def} V$$

or in component form,

$$P_{ij} = \begin{cases} \mu\left(\dfrac{\partial V_i}{\partial x_j} + \dfrac{\partial V_j}{\partial x_i}\right) & if \;\; i \neq j \\ -p + 2\mu\dfrac{\partial V_i}{\partial x_j} & if \;\; i = j \end{cases}$$

Equation (4) is the rheological equation of a Newtonian incompressible viscous fluid. Expanded form the formulas of the accepted connection (4) in the Cartesian coordinate system

$$P_{11} = -p + 2\mu\frac{\partial V_1}{\partial x_1}$$

$$P_{22} = -p + 2\mu\frac{\partial V_2}{\partial x_2}$$

$$P_{33} = -p + 2\mu\frac{\partial V_3}{\partial x_3}$$

$$P_{12} = P_{21} = \mu\left(\frac{\partial V_1}{\partial x_2} + \frac{\partial V_2}{\partial x_1}\right)$$

$$P_{23} = P_{32} = \mu\left(\frac{\partial V_2}{\partial x_3} + \frac{\partial V_3}{\partial x_2}\right)$$

$$P_{13} = P_{31} = \mu\left(\frac{\partial V_3}{\partial x_1} + \frac{\partial V_1}{\partial x_3}\right)$$

Substituting these expressions into the equations of motion and using the property, it obtains

$$\frac{\partial}{\partial x_1}\left(-p + 2\mu\frac{\partial V_1}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(\mu\left(\frac{\partial V_1}{\partial x_2} + \frac{\partial V_2}{\partial x_1}\right)\right) + \frac{\partial}{\partial x_3}\left(\mu\left(\frac{\partial V_3}{\partial x_1} + \frac{\partial V_1}{\partial x_3}\right)\right) =$$

$$= -\frac{\partial p}{\partial x_1} + \mu\left(\frac{\partial^2 V_1}{\partial x_1^2} + \frac{\partial^2 V_1}{\partial x_2^2} + \frac{\partial^2 V_1}{\partial x_3^2}\right) + \mu\frac{\partial}{\partial x_1}\left(\frac{\partial V_1}{\partial x_1} + \frac{\partial V_2}{\partial x_2} + \frac{\partial V_3}{\partial x_3}\right)$$

At the end, it obtains the Navier-Stokes equations for an incompressible viscous fluid:

$$\rho\left(\frac{\partial V_1}{\partial t} + V_1\frac{\partial V_1}{\partial x_1} + V_2\frac{\partial V_1}{\partial x_2} + V_3\frac{\partial V_1}{\partial x_3}\right) = \rho f_1 - \frac{\partial p}{\partial x_1} + \mu\left(\frac{\partial^2 V_1}{\partial x_1^2} + \frac{\partial^2 V_1}{\partial x_2^2} + \frac{\partial^2 V_1}{\partial x_3^2}\right)$$

$$\rho\left(\frac{\partial V_2}{\partial t} + V_1\frac{\partial V_2}{\partial x_1} + V_2\frac{\partial V_2}{\partial x_2} + V_3\frac{\partial V_2}{\partial x_3}\right) = \rho f_2 - \frac{\partial p}{\partial x_2} + \mu\left(\frac{\partial^2 V_2}{\partial x_1^2} + \frac{\partial^2 V_2}{\partial x_2^2} + \frac{\partial^2 V_2}{\partial x_3^2}\right)$$

$$\rho\left(\frac{\partial V_3}{\partial t} + V_1\frac{\partial V_3}{\partial x_1} + V_2\frac{\partial V_3}{\partial x_2} + V_3\frac{\partial V_3}{\partial x_3}\right) = \rho f_3 - \frac{\partial p}{\partial x_3} + \mu\left(\frac{\partial^2 V_3}{\partial x_1^2} + \frac{\partial^2 V_3}{\partial x_2^2} + \frac{\partial^2 V_3}{\partial x_3^2}\right)$$

3.4 Discretization of governing equations

For the numerical implementation of the set mathematical model, discretization is carried out using the finite volume method. To use the finite volume method, the formulated equations are written in the integral form

$$\int_S \rho\phi\upsilon \cdot ndS = \int_S \Gamma grad\phi \cdot ndS + \int_\Omega q_\phi d\Omega. \qquad (5)$$

Further use of the finite volume method depends on the type of computational grid (structured or unstructured).

3.5 Incompressible Navier-Stokes equations. Dynamic similarity (dimensionless).

Let us proceed to the consideration of conditions for the similarity of two isothermal flows of Newtonian viscous incompressible fluids with different, non-constant densities and viscosities. Let us reduce the Navier-Stokes equations to a dimensionless form, choosing as the scales of time, length (in particular, coordinates), velocity, pressure and body forces, respectively, some constant values characteristic of the flow T, L, U_0, P.

Denoting the dimensionless values of time, position coordinates, velocity, pressure, and forces with an upper underline, it was assumed (here it is more convenient to use letter indexing: x, y, z, u, v, w) that

$$\bar{u} = \frac{u}{U_0}, \qquad \bar{v} = \frac{v}{U_0}, \qquad \bar{w} = \frac{w}{U_0}, \qquad \bar{p} = \frac{p}{P}$$

$$\bar{x} = \frac{x}{L}, \qquad \bar{y} = \frac{y}{L}, \qquad \bar{z} = \frac{z}{L}, \qquad \bar{t} = \frac{t}{T}.$$

Let us transform these expressions to the form

$$u = \bar{u}U_0, \qquad x = \bar{x}L, \qquad v = \bar{v}U_0,$$

$$y = \bar{y}L, \qquad w = \bar{w}U_0, \qquad z = \bar{z}L, \qquad (6)$$

$$p = \bar{p}P, \qquad t = \bar{t}T,$$

where

$$T = \frac{L}{U_0}$$

Consider the Navier-Stokes equations

1) $$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \quad (7)$$

2) $$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -\frac{1}{\rho}\frac{\partial P}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) \quad (8)$$

3) $$\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}=-\frac{1}{\rho}\frac{\partial P}{\partial z}+\nu(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}) \quad (9)$$

4) $$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \quad (10)$$

It substitutes (6) into equations (7), (8), (9) and (10) and for simplicity, it omits the underscores and write in the form.

1) $$\frac{U_0^2}{L}\frac{\partial u}{\partial t}+\frac{U_0^2}{L}(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z})=-\frac{P}{\rho L}\frac{\partial P}{\partial x}+\frac{U_0}{L^2}\nu(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2})$$

2) $$\frac{U_0^2}{L}\frac{\partial v}{\partial t}+\frac{U_0^2}{L}(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z})=-\frac{P}{\rho L}\frac{\partial P}{\partial y}+\frac{U_0}{L^2}\nu(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2})$$

3) $$\frac{U_0^2}{L}\frac{\partial w}{\partial t}+\frac{U_0^2}{L}(u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z})=-\frac{P}{\rho L}\frac{\partial P}{\partial z}+\frac{U_0}{L^2}\nu(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2})$$

4) $$\frac{U_0}{L}(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z})=0$$

Reducing the momentum equations by $\frac{U_0^2}{L}$, and the continuity equation by $\frac{U_0}{L}$, it obtains

1) $$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}=-\frac{P}{\rho U_0^2}\frac{\partial P}{\partial x}+\frac{\nu}{LU_0}(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2})$$

2) $$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}=-\frac{P}{\rho U_0^2}\frac{\partial P}{\partial y}+\frac{\nu}{LU_0}(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2})$$

3) $$\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}=-\frac{P}{\rho U_0^2}\frac{\partial P}{\partial z}+\frac{\nu}{LU_0}(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2})$$

4) $$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0$$

It can introduce the following dimensionless one-term complexes, called "similarity numbers":

$$Eu = \frac{P}{\rho U_0^2}, \qquad \mathrm{Re} = \frac{LU_0}{\nu}$$

Eu is the Euler number, Re is the Reynolds number.

And using these dimensionless numbers, it rewrites the momentum equations and continuity equation in this form

1) $$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -Eu\frac{\partial P}{\partial x} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

2) $$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -Eu\frac{\partial P}{\partial y} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right)$$

3) $$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -Eu\frac{\partial P}{\partial z} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right)$$

4) $$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

3.6 Incompressible Navier-Stokes equations for curvilinear coordinates

Carrying out numerical simulation for complex areas, the transformation of coordinates from the Cartesian to the generalized coordinate system is carried out. So the transformation between the Cartesian system ($\bar{x} = (x, y, z)$) and the curvilinear coordinate system ($\bar{\xi} = (\xi, \eta, \varsigma)$) occurs using such relations $x_\xi, x_\eta, ..$ and $\xi_x, \eta_x, ...$

$$\begin{bmatrix} x_\xi & x_\eta & x_\varsigma \\ y_\xi & y_\eta & y_\varsigma \\ z_\xi & z_\eta & z_\varsigma \end{bmatrix} \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \xi_x & \xi_y & \xi_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The Jacobian is defined in this form

$$J = \frac{\partial(x, y, z)}{\partial(\xi, \eta, \varsigma)} = \begin{bmatrix} x_\xi & x_\eta & x_\varsigma \\ y_\xi & y_\eta & y_\varsigma \\ z_\xi & z_\eta & z_\varsigma \end{bmatrix}$$

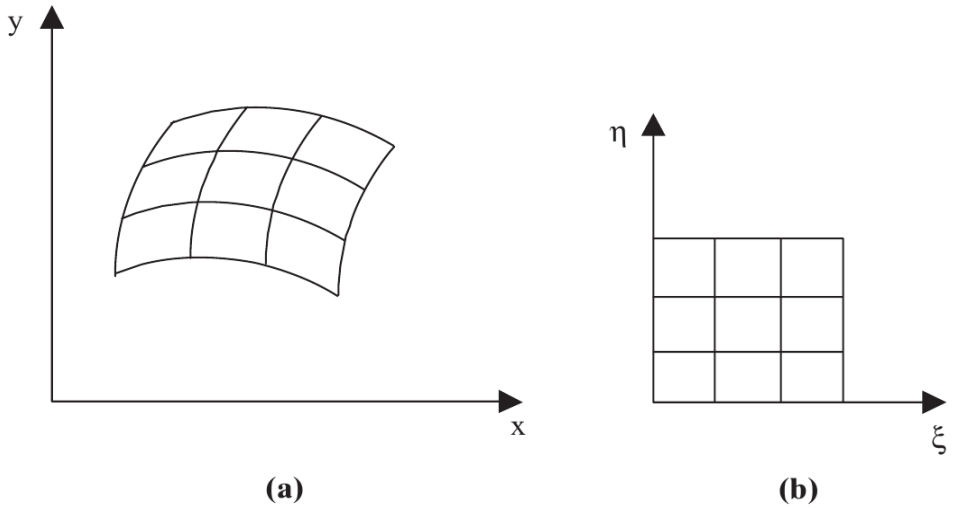


Figure 12 (a) - Curvilinear coordinate system; (b) Cartesian coordinate system.

Now it will be necessary to replace the derivatives with respect x, y, z to ξ, η, ς, then it turns out in this form

$$\frac{\partial}{\partial x_i} = \frac{\partial \xi_j}{\partial x_i} \frac{\partial}{\partial \xi_j}$$

then

$$\nabla = \nabla \xi_j \frac{\partial}{\partial \xi_j}$$

Given this ratio, it can be written as

$$J\nabla \phi = \frac{\partial}{\partial \xi_i} J\nabla \xi_j \phi$$

$$J\nabla^2 = \frac{\partial}{\partial \xi_i}\left[J g_{ij} \frac{\partial}{\partial \xi_i} \right]$$

$$u_i \frac{\partial}{\partial x_i} = U_i \frac{\partial}{\partial \xi_j}$$

where

$$g_{ij} = \nabla\xi_i \nabla\xi_j$$

Using these expressions, the Navier-Stokes equation can be rewritten in this form

$$\frac{\partial}{\partial t}(JU_i) + \frac{\partial}{\partial \xi_j}(JU_jU_i) - JU_j u \frac{\partial}{\partial \xi_j}\nabla\xi_i = -g_{ij}\frac{\partial p}{\partial \xi_j} +$$

$$\nu\left[\frac{\partial}{\partial \xi_j} g_{ij} \frac{\partial}{\partial \xi_k}(U_i) - \frac{\partial}{\partial \xi_j}(J\frac{\partial \xi_j}{\partial x_i}\frac{\partial}{\partial \xi_k}(\frac{\partial \xi_i}{\partial x_i})U_k) - J\frac{\partial^2 \xi_i}{\partial x_k \partial x_i}\frac{\partial u_k}{\partial x_i}\right]$$

$$\frac{\partial}{\partial \xi_j}(JU_j) = 0$$

where

$$\frac{\partial u_k}{\partial x_i} = \frac{\partial \xi_i}{\partial x_i}\frac{\partial}{\partial \xi_j}(U_m \frac{\partial x_k}{\partial \xi_m})$$

4. NUMERICAL INVESTIGATION OF THE EFFICIENCY OF HIGH PERFORMANCE COMPUTING FOR BACKWARD STEP FLOW PROBLEMS

4.1 Problem statement

Construction is one of the fastest growing areas today. But the natural aerodynamics between buildings is always taken into account. To do this, you must first take into account ventilation before the start of the project itself. One of the optimal solutions is to build a mathematical model of the air flow. Below are numerical solutions for the wind flow around architectural obstacles, taking into account vertical buoyancy forces. For the calculation, the Navier-Stokes equation, the control volume method for approximation and the projection method for the numerical solution were used. The Jacobi method for each time step by iterative method was used for the Poisson equation satisfying the discrete continuity equation. The test problem was solved to check the correctness of the mathematical model and the numerical algorithm. The results of other authors in the numerical form were used for comparison with the numerical solutions of the reverse flow step with vertical buoyancy forces. For parallelization of the numerical algorithm, 1D, 2D and 3D decompositions were used. Having made a theoretical analysis of the efficiency for the above types of decomposition, the best method of domain decomposition was determined. And also real computational experiments were carried out for this problem. Moreover, the resulting numerical algorithm with the best domain decomposition method and mathematical model can be widely used for various complex flows with vertical buoyancy forces.The increased pace of construction in modern large cities and, in particular, Almaty, leads to a tightening of architectural structures. Due to the increase in the population of cities and to save space, mostly high-rise buildings are being built. As a consequence, this entails such consequences as a violation of the natural aerodynamics of the city, which in turn leads to increased gas contamination of the city, the accumulation of heavy metals in the lower atmosphere, and to the violation of the local climate. The building codes and norms currently used in the construction and design of buildings do not contain aerodynamic criteria and coefficients indicating the optimal distance between buildings of different heights. When determining these standards, various natural and climatic features are taken into account, such as wind loads, insolation, etc. Fire safety requirements are also taken into account. However, the above-mentioned documents do not take into account the factor of natural aerodynamics of space between neighboring buildings. The distance between buildings and structures is considered to be the distance between the outer walls or other structures. As a result, when designing, the distances between building objects are laid, which can not provide free movement of the wind vortex, which leads to a disturbance of the natural air flow. In this thesis, a model of aerodynamics between two high-rise buildings is considered. Parameters such as the optimal distance between two buildings, climate change are taken into account in this mathematical model. Stream splitting with sudden geometry expansion or subsequent

reconnection is quite common in many technical streams. The separation zone and flow recirculation has a significant impact on the performance of heat exchange devices, such as cooling devices in electrical engineering, cooling channels for turbine blades, combustion chambers, and many other heat exchange surfaces that occur in equipment. There are many works without taking into account the forces of buoyancy, which are devoted to the movement of fluid with separation and reconnection of flows. Nevertheless, this process is very important, which is confirmed in various works. For example, the study of flows with separation zones and their development of experimental and theoretical methods [37-45, 119, 122, 123, 125]. As well as construction equipment [37-39]. Isothermal flows in fluid flows are considered in [46-48]. Such as Aung [49, 50], Aung et al. [51], Aung and Worku [52], Sparrow et al. [53, 54] and Vorobey and Chuck [55] also investigated heat transfer in flows. However, the heat transfer characteristics and the buoyant force on the streamlined flow are not taken into account. At low speed and with a large temperature difference, these effects are significantly noticeable. Moreover, the finite element method was studied when applied to a two-dimensional square resonator with different heater positions and sizes [62]. In turn, the same finite volume method was used by Oztop and Abu-Nada [63] to numerically study convection in rectangular shells partially heated from the side wall. The paper also shows the influence of buoyant forces on the characteristics of the flow and heat transfer in individual flows. Figure 13 shows the numerical results for a laminar mixed-convective air flow (Pr=0.7) in a vertical two-dimensional channel with a reverse step to preserve the buoyancy effect. The buoyancy force affects different parameters. Of great interest are the distributions of velocity and temperature, ligation lengths and coefficients of friction shown to demonstrate the effect of the buoyancy force.

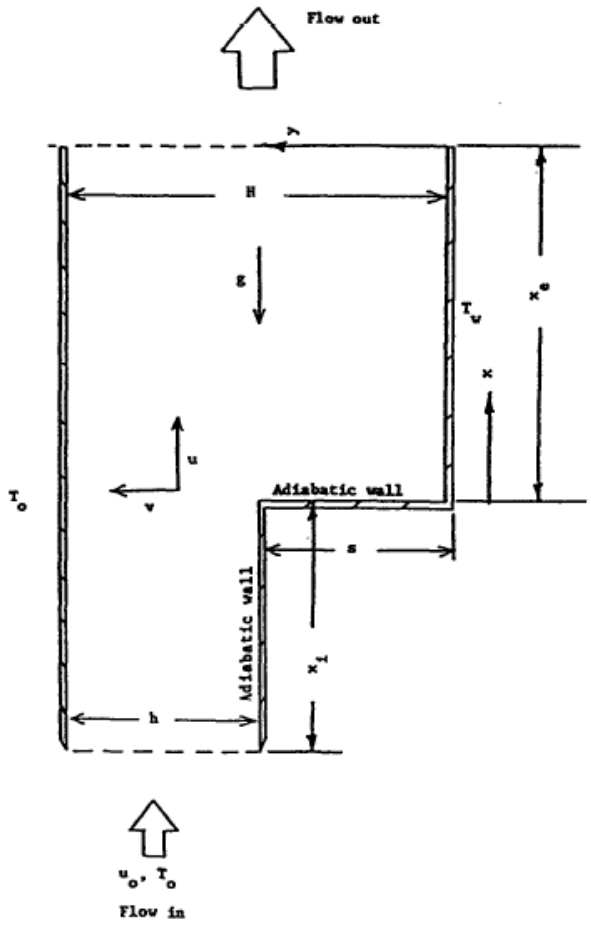


Figure 13 - Schematic representation of the backward-facing step flows.

4.2 Mathematical Formulation of the Problem

Figure 13 shows a two-dimensional laminar convective flow in a vertical channel with a sudden expansion behind a backward ledge of height s. A constant temperature similar to the intake air temperature T0 exists permanently on a straight duct wall. Tw desired heating temperature for the stepped wall below the stage, constant temperature. The upper part of the stepped wall and the reverse side are taken as the adiabatic surface. xi input length and xe output lower channel length have the corresponding dimensions. Despite the assumption that these lengths are infinite, the length limit is Le = xe + xi. H = h + s is the height of the large area below the stage, while the smaller area is just height. Moving up the channel, the air has an average speed u0 and a constant temperature T0. The gravitational force g is directed vertically downward. The Boussinex approximation and the assumption of constant properties were used to describe this physical problem. This system of equations in an unenclosed form can be written as:

1) $$\frac{\partial U}{\partial X} + \frac{\partial U}{\partial Y} = 0 \qquad (15)$$

2) $$\frac{\partial U}{\partial t}+U\frac{\partial U}{\partial X}+V\frac{\partial U}{\partial Y}=-\frac{\partial P}{\partial X}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 U}{\partial X^2}+\frac{\partial^2 U}{\partial Y^2}\right)+\frac{Gr}{\mathrm{Re}^2}\theta \tag{16}$$

3) $$\frac{\partial V}{\partial t}+U\frac{\partial V}{\partial X}+V\frac{\partial V}{\partial Y}=-\frac{\partial P}{\partial Y}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 V}{\partial X^2}+\frac{\partial^2 V}{\partial Y^2}\right) \tag{17}$$

4) $$\frac{\partial \theta}{\partial t}+U\frac{\partial \theta}{\partial X}+V\frac{\partial \theta}{\partial Y}=\frac{1}{\mathrm{Pr}\,\mathrm{Re}}\left(\frac{\partial^2 \theta}{\partial X^2}+\frac{\partial^2 \theta}{\partial Y^2}\right) \tag{18}$$

The dimensionless parameters in the equations given above are defined by the formula:

$U=u/u_0$, $V=\upsilon/u_0$, $X=x/s$, $Y=y/s$,

$\theta=(T-T_0)/(T_w-T_0)$, $P=p/\rho_0 u_0^2$,

$\mathrm{Pr}=\nu/\alpha$, $\mathrm{Re}=u_0 s/\nu$, $Gr=g\beta(T_w-T_0)s^3/\nu^2$.

Where α – the temperature diffusion, ν – the kinematic viscosity, and β – the thermal expansion coefficient are estimated at the film temperature $T_f=(T_0+T_w)/2$ (Figure 14).

Boundary conditions:

(a) Inlet conditions: At the point $X=-X_i$ and $1\le Y\le H/s$: $U=u_i/u_0$, $V=0$, $\theta=0$, $\frac{\partial p}{\partial x}=-\frac{Gr}{\mathrm{Re}^2}\theta$.

where u_i is the local distribution of velocities at the inlet, which is assumed to have a parabolic profile and u_i/u_0 an average inlet velocity, that is, given by formula

$$u_i/u_0=6\left[-y^2+(H+s)y-Hs\right]/(H-s)^2$$

(b) Outlet conditions: At the point $X=X_e$ and $0\le Y\le H/s$: $\partial U/\partial X=0$, $\partial^2\theta/\partial X^2=0$, $\partial V/\partial X=0$, $\frac{\partial p}{\partial x}=-\frac{Gr}{\mathrm{Re}^2}\theta$.

(c) on the top wall: At the point $Y=H/s$ and $-X_i\le X\le X_e$: $U=0$, $V=0$, $\theta=0$, $\frac{\partial p}{\partial y}=0$.

(d) on the wall of the upper stage: At the point $Y=1$ and $-X_i \le X < 0$: $U=0$, $V=0$, $\partial\theta/\partial Y=0$, $\frac{\partial p}{\partial y}=0$.

(e) on the wall of the lower stage: At point $X=0$ and $0 \le Y \le 1$: $U=0$, $V=0$, $\partial\theta/\partial X=0$, $\frac{\partial p}{\partial x}=0$.

(f) on the wall below the stage: At the point $Y=0$ and $0 \le X \le X_e$: $U=0$, $V=0$, $\theta=1$, $\frac{\partial p}{\partial y}=0$.

The last term on the right-hand side of equation (16) is the contribution of the buoyancy force. The length of the downstream flow from the simulation area was chosen to be 70 steps (X_e=70). The upper length of the design area was chosen to be 5 steps (i.e. X_i=5), and the velocity profile at the input area was set as parabolic profile, like $u_i/u_0 = 6\left[-y^2+(H+s)y-Hs\right]/(H-s)^2$, and temperature was chosen as uniform T_0.

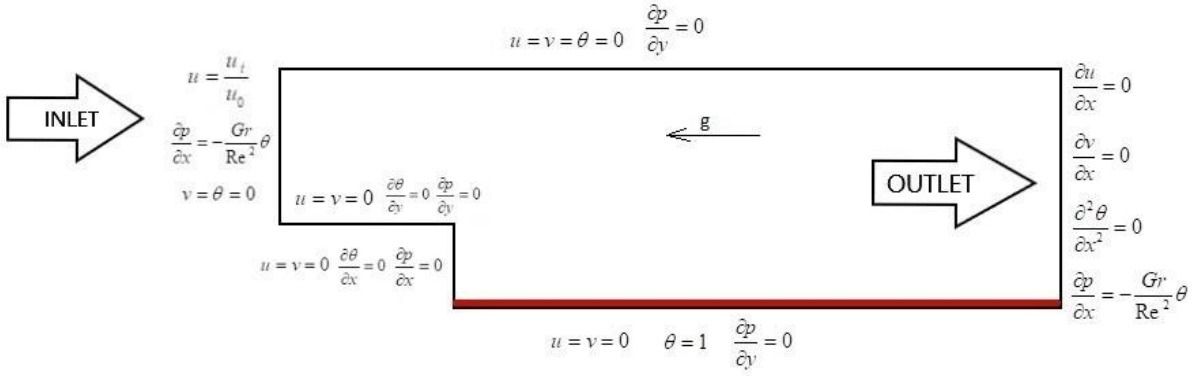


Figure 14 - Boundary conditions.

4.3 Numerical algorithms

The projection method [57-60] was used to solve this system of equations numerically. The finite volume method [56-59] was used for approximation. At the first stage, it is assumed that momentum transfer occurs only due to convection and diffusion, and the intermediate velocity field is calculated by the fourth-order Runge-Kutta method [56-59]. At the second stage, according to the found intermediate velocity field, there is a pressure field. The Poisson equation for the pressure field is

solved by the Jacobi method. At the third stage, it is assumed that the transfer is carried out only due to the pressure gradient. At the fourth stage, the equations for temperature are calculated by the fourth-order Runge-Kutta method [56-59].

$$I. \quad \int_{\Omega} \frac{\vec{u}^{*} - \vec{u}^{n}}{\Delta t} d\Omega = -\oint_{\partial\Omega} (\vec{u}^{n}\,\vec{u}^{*} - \frac{1}{\mathrm{Re}} \nabla \vec{u}^{*}) n_i d\Gamma - \int_{\Omega} \frac{Gr}{\mathrm{Re}^2} \theta d\Omega,$$

$$II. \quad \oint_{\partial\Omega} (\nabla p) d\Gamma = \int_{\Omega} \frac{\nabla \vec{u}^{*}}{\Delta t} d\Omega,$$

$$III. \quad \frac{\vec{u}^{n+1} - \vec{u}^{*}}{\Delta t} = -\nabla p,$$

$$IV. \quad \int_{\Omega} \frac{\theta^{*} - \theta^{n}}{\Delta t} d\Omega = -\oint_{\partial\Omega} (\vec{u}^{n}\,\theta^{*} - \frac{1}{\mathrm{Re}\,\mathrm{Pr}} \nabla \theta^{*}) n_i d\Gamma,$$

4.4 Parallelization algorithm

For numerical simulation was constructed a computational mesh by using the PointWise software. The problem was launched on the ITFS-MKM software using a high-performance computing. The equations are approximated by the finite volume method (FVM) and used collocated grid, because it makes parallelization of numerical algorithm simple and efficient to use domain decomposition method. For pressure velocity coupling is used Rhie-Chow interpolation. This relation can interpolate the surface pressure of the cell. The scheme of the central pressure difference was built on the basis of the above pressure. In turn, this scheme makes it possible to avoid the checkerboard effect, and also connects adjacent pressures. a first-order upstream flow scheme was used for convective flow. 1D, 2D and 3D decompositions were used to fully parallelize this numerical algorithm. Geometric partitioning of the computational grid was chosen as the main approach. Given this, it can be said that there are three ways to exchange the values of the grid function on the computing nodes of a one-dimensional, two-dimensional and three-dimensional grid. After the domain decomposition stage, when parallel algorithms are built on separate blocks, a transition is made to the relationships between the blocks, the simulations on which will be executed in parallel on each processor. For this purpose, a numerical solution of the equation system was used for an explicit scheme, since this scheme is very efficiently parallelized. In order to use the domain decomposition method as a parallelization method, this algorithm uses the boundary nodes of each subdomain in which it is necessary to know the value of the grid function that borders on the neighboring elements of the processor. To achieve this goal, at each compute node, ghost points store values from neighboring computational nodes, and organize the transfer of these boundary values necessary to ensure homogeneity of calculations for explicit formulas [61].

The procedures of the MPI library [61] were used for data transfer.

Some theoretical analysis was carried out to determine the effectiveness of different decompositions. For this, Tcalc was taken as the sequential program time divided by the number of processors plus the transfer time Tp = Trast/r + Tcom. While transfers for various domain decomposition methods can be roughly expressed in terms of throughput [59, 61]:

$$T_{com}^{1D} = t_{send}\, 2N^2 x2$$

$$T_{com}^{2D} = t_{send}\, 2N^2 x4p^{1/2} \qquad (19)$$

$$T_{com}^{3D} = t_{send}\, 2N^2 x6p^{2/3}$$

where N – the number of nodes in the computational mesh, p - the number of processors (cores), t_{send} – the time of sending one element (number). It should be noted that for different decomposition methods, the data transmission cost can be represented as $T_{com}^{1D} = t_{send}\, 2N^2 xk(p)$ in accordance with the formula (19), where k(p) is the proportionality coefficient, which depends on the domain decomposition method and the number of processing elements used [59, 61]. At the first stage, one common program was used, the size of the array from start to run did not change, and each element of the processor was numbered by an array of elements, starting from zero. For the test simulation well known problem – 3D cavity flow was used. Nevertheless, despite the fact that theoretical analysis showed that 3D decomposition gives the best parallelization option (Figure 15), judging by the computational experiments, it can be said that the best results were obtained using 2D decomposition, when the number of processes varies from 25 to 144 (Figure 15) [59, 61].

Moreover, after a preliminary theoretical analysis of the graphs, the following characteristics can be given. If communication time is not taken into account, the simulation time should be approximately the same for the same number of processors and decrease by Tcalc/p.

In fact, judging by the calculated data, the use of 2D decomposition with different computational grids gives the minimum cost of modeling and cost graphs are much higher, depending on the simulation time, on several processors taken with Tcalc/p [59, 61].

To substantiate these results, it is necessary not to forget the assumptions that were made in the preliminary theoretical analysis of the effectiveness for this task. The first assumption: regardless of the distribution of data over the elements of the processor, the same amount of computational load is performed. This assumption should lead to the same time costs. The second assumption is that the time taken to interprocessor transfer of any degree of the same amount of data does not depend on their choice of memory. For a deeper understanding, the following sets of computational simulation tests were carried out. For evaluation, the sequence of the first approach was considered, when the program is executed in a single-processor

version and, thus, models various methods of data decomposition in the geometric domain with the same amount of computation performed by each processor [59, 61].

4.5 Results of numerical calculation

On Figure 13 shows the geometric parameters: channel length L=75, channel height H=2, step height S=1. Dimensionless numbers equal to Re=50, Pr=0.7, Gr=19.1 [45] were used in the numerical results.

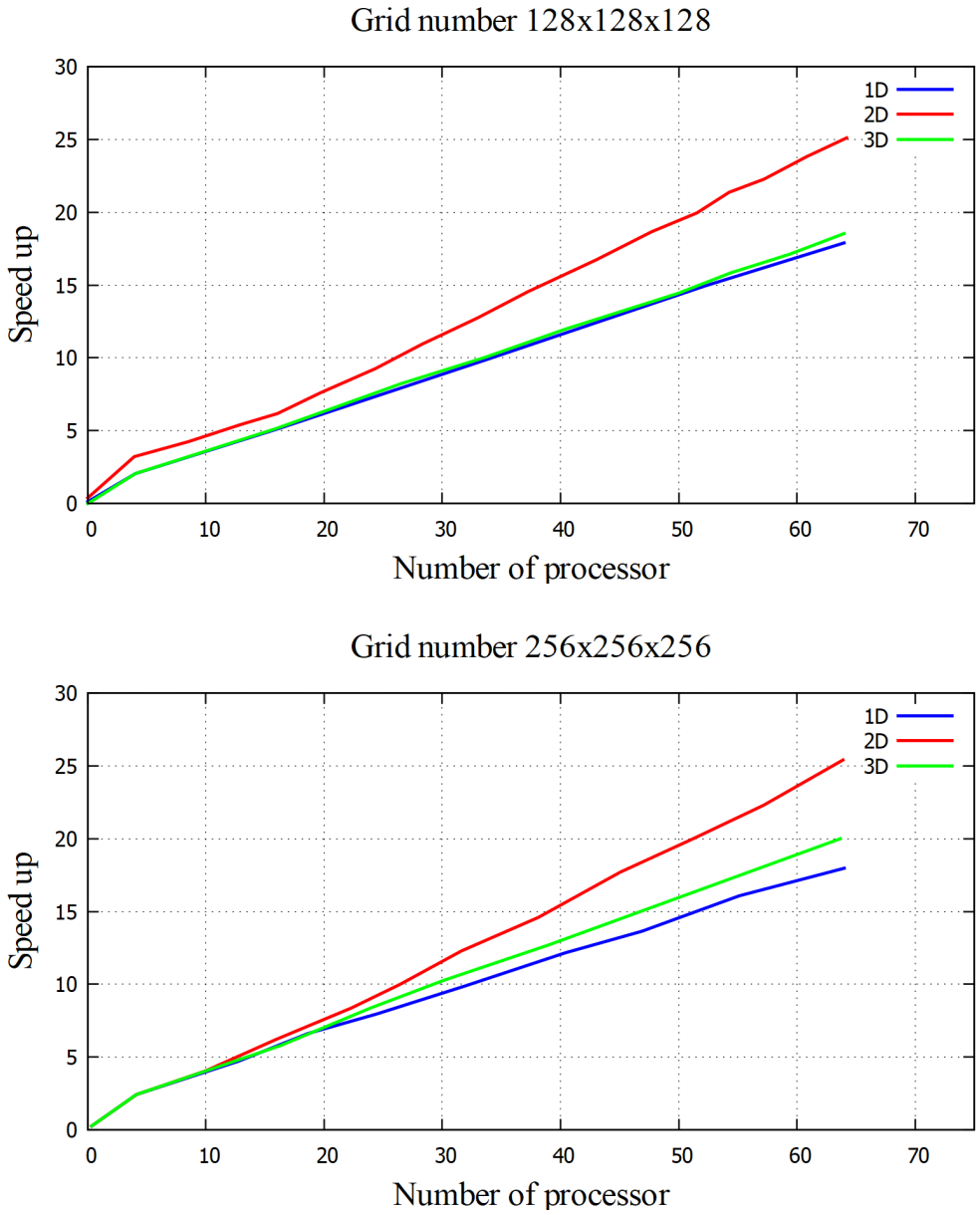


Figure 15 - Speed-up for various domain decomposition methods of the computational domain.

Comparison of numerical data with data from Lin et al. [45] at the point $x/x_f=0.5$, where $x_f=2.91$ is shown in Figure 16 for the longitudinal velocity profile. Whereas comparison of temperature profiles with numerical data from Lin et al. [45] at the

point x/x_f=0.5, where x_f=2.91 is shown in Figure 17. Judging by the figures, it can be said that there is a similarity between the results of this work and those of Lin et al. [45]. Moreover, in Figure 18 it can be seen the streamlines and the contour of the horizontal velocity for the dimensionless numbers Re=50, Pr=0.7 and Gr=19.1. With the same dimensionless quantities in Figure 20 shows the temperature profile. For a better understanding of this process, from Figure 18, 19 and 20 one can see the development of a reverse stepwise flow with a vertical buoyancy force: the origin and development of the flow reconnection region, taking into account the buoyancy forces.

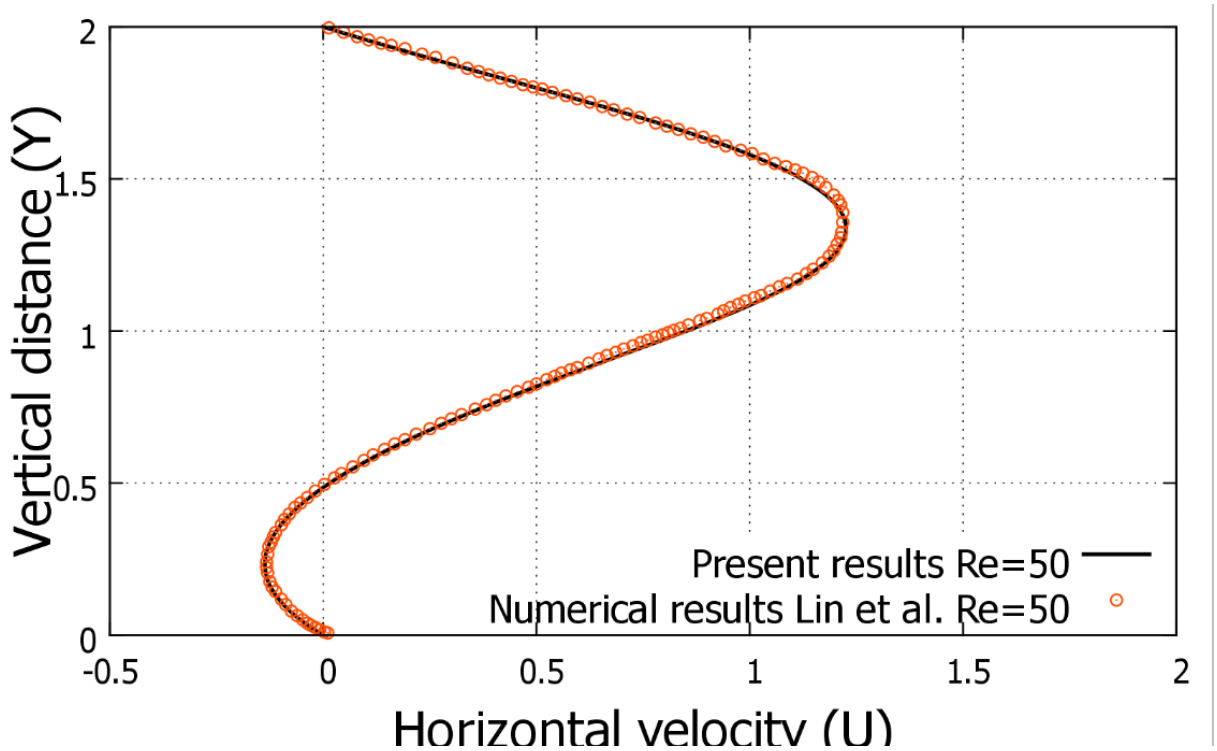


Figure 16 - Velocity profile with vertical buoyancy forces for dimensionless number Re = 50, ΔT = 1 ºC, x/x_f = 0.5, where x_f = 2.91.

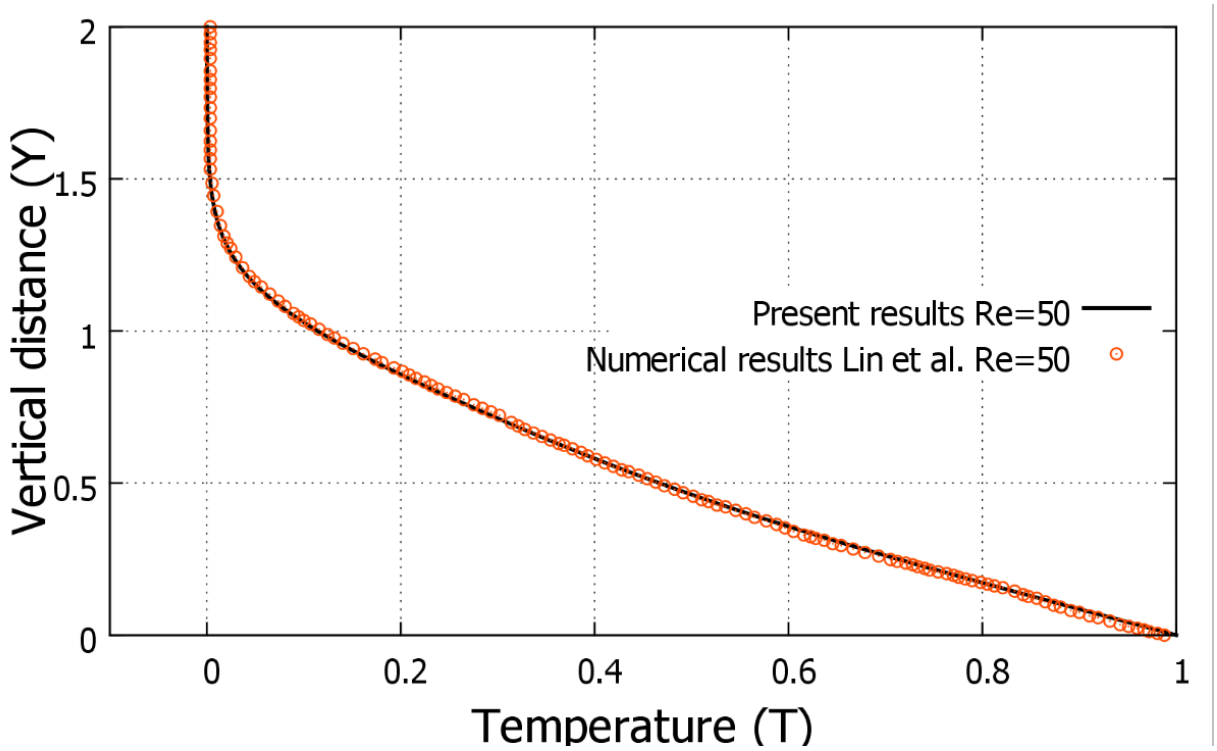


Figure 17 - Temperature profile with vertical buoyancy forces for dimensionless number Re = 50, ΔT = 1 ºC, x/x_f = 0.5, where x_f = 2.91.

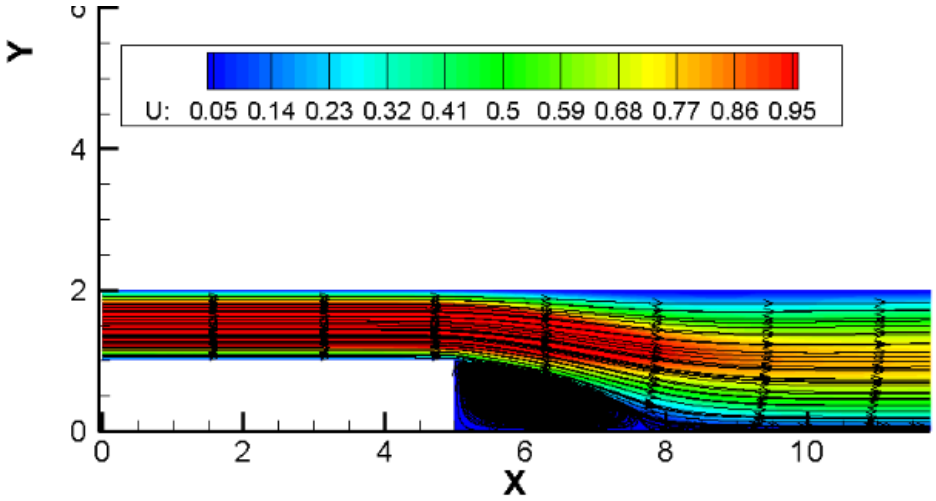


Figure 18 - The contour of the horizontal velocity component with streamlines for dimensionless numbers Re=50, Pr=0.7 and Gr=19.1.

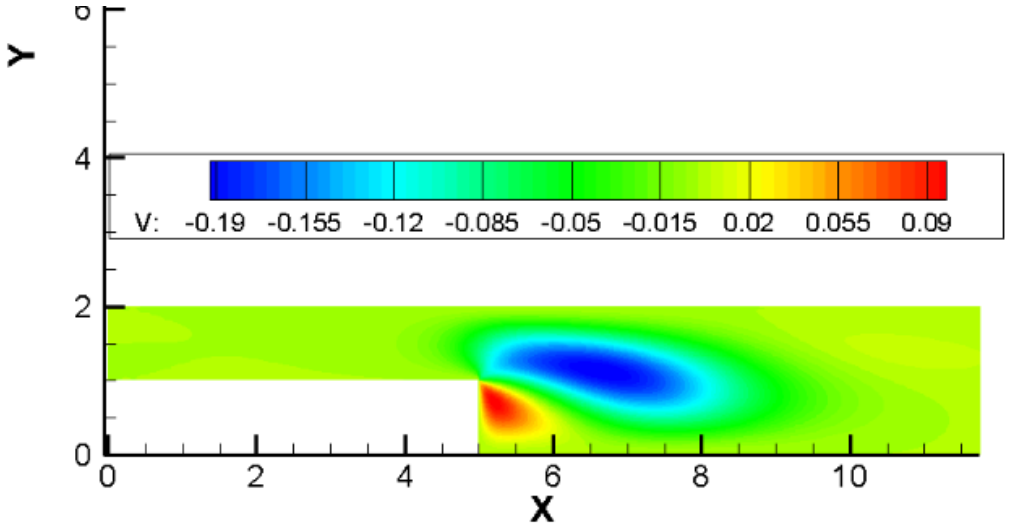


Figure 19 - The contour of the vertical velocity component for dimensionless numbers Re=50, Pr=0.7 and Gr=19.1.

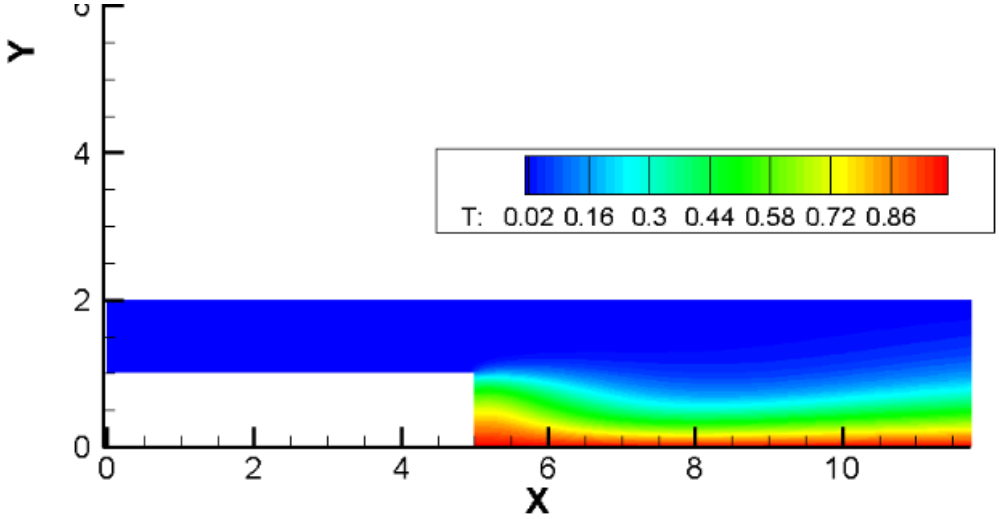


Figure 20 - Temperature contour for dimensionless numbers Re=50, Pr=0.7 and Gr=19.1.

4.6 Conclusion

The zone of flow confluence behind the reverse step, taking into account the buoyancy forces, was used for numerical studies of the laminar flow. Thus, it was possible to obtain some clarification in the internal flow behind the reverse scarp and in the processes of reconnection of flows under the influence of temperature effects. Moreover, data were obtained for a deep understanding of the appearance of secondary zones. For a booming study of reverse stepwise flows, taking into account the buoyancy forces [45], the distance from the ledge to the channel boundary is 4 times greater than the channel height. From the numerical data of the distribution of velocities, the zones of formation of the primary reattachment of reverse stepwise flows were obtained. Thus, the projection method was used to numerically solve the system of Navier-Stokes equations. Moreover, it can be argued that the numerical results have a small error. Since they were compared with the dimensionless numbers

Re=50, Pr=0.7 and Gr=19.1 by the numerical results of other authors [45]. It was obtained by fully parallelizing the numerical algorithm to speed up the process. As mentioned earlier, different types of decompositions (1D, 2D and 3D) were used. For 250 processors or less, 3D domain decomposition does not need as much time as 2D. These conclusions were drawn on the basis of the numerical results of the test problem of flow in a 3D resonator, in which the decomposition method of 1D, 2D, and 3D domains was used. In theory, more time-consuming work is required for the decomposition of the subject area. But for a given scale of the problem, a 2D decomposition of the subject area is sufficient. That's why for backward-facing step flow with vertical buoyancy force is used 2D domain decomposition. It should also be noted that setting the boundary conditions is an important process. In the future, this mathematical model and a parallel numerical algorithm can be applied to various complex flows taking into account the buoyancy forces.

5. NUMERICAL INVESTIGATION OF THE EFFICIENCY OF HIGH-PERFORMANCE COMPUTING USING HYBRID PARALLEL ALGORITHMS FOR AIRFLOW PROBLEMS IN A COMPLEX NASAL REGION

The current trend in the development of high-performance computers opens up new opportunities for developing highly effective methods for modeling complex problems using multi-level decomposition and hierarchical parallelization of computations. For most real physical processes with a large computational grid this approach is the only practical way to create an adequate computing model of control objects. In addition, traditional serial computers and computational schemes have come to their technological limit. At the same time, a technological breakthrough in the field of creating means for interprocessor and intercomputer communications makes it possible to realize effective control in the distribution of computations for various components by an integrated computer, which in turn is one of the key properties of parallelism. Through the nasal cavity there is a primary recognition of odors, through it the air is breathed, which passes into the alveolar state (it heats there to physiologically normal temperature and is completely saturated with water vapor). They serve as regulators of all air circulation, create a normal air temperature and are completely saturated with water vapor, cleaned and disinfected. Normally, the airflow passes through the nose at a speed of 6 L/min, this figure can be increased to 10 L/min. However, the nasal cavity depending on the cause of occurrence has curvatures and can be divided as: Physiological Compensatory Traumatic. With the aforementioned character of curvature, they negatively affect primarily the difficulty of breathing. Nasal breathing is very important, a systemic part of our body's vital activity and any of its disturbances sooner or later cause negative consequences for the human body. The main method of eliminating the curvature of the nasal cavity is a surgical operation—septoplasty. However, it should be noted that the success of a surgical operation at the best does not exceed 80%, which leads to a repeated surgical operation. And also the surgical operation will depend on the experience and skill of the surgeon. Naturally, to increase the percentage of success of a operation, it will be necessary to accurately make nasal cavity corrections. Before the surgical intervention due to X-ray images it is possible to evaluate the nature of the curvature and with the help of numerical modeling it will be possible to correct and optimize the nasal cavity in advance. Knowing the preliminary accurate correction of the nasal sinus, the surgeon can increase the percentage of success of the operation that will accordingly reduce the percentage of the reoperation [124].

The nasal cavity balances the inhaled air with the internal state of the body with surprising efficiency. In the papers of Cole [64], Inglessted [65] and Webb [66], it was generally agreed that the inhaled air through the nasal cavity reaches up to the alveolar state (completely saturated with water vapor and at normal body temperature) by the time it reaches the pharynx. And it practically does not depend on a condition of ambient air which has arrived through nostrils. These results were also obtained in the paper of Farley and Patel [67] who collected data in natural conditions

with air temperature readings along the upper respiratory tract, as well as Hannah and Scherer [68], reflecting measurements of local mass transfer coefficients on the gypsum model the human upper respiratory tract. Nevertheless, McFadden [69] noted that the conclusions are valid for quiet breathing, in some circumstances at high ventilation levels, conditioning of additional air should occur in the intrathoracic airways in order to completely determine the inhaled air in the alveolar state [124].

Numerous studies have been aimed at assessing the hydration and temperature regulation of the nasal cavity. However, mathematical models were based on axisymmetric tubes or occupied quasistationary flows [70]. As a rule, these studies confirmed the opinion that under normal conditions there is enough time for heating and humidifying air in the nasal cavity. In addition, medications as well as surgical procedures are being used with increasing speed to restore the structure and functions of the nasal cavity [71]. For example, aromatic inhalations are used to improve airflow and to reduce congestion, as well as rhinoplasty procedures are used to overcome trauma or aesthetic deformities. These artificial interventions cause local changes, and can affect the efficiency of transport phenomena of air. However, precise intranasal characteristics and distribution of transport phenomena are not yet known even for a normal (or healthy) state [72-76, 117, 118, 120, 124, 126].

Experimental examination of the nasal cavity is practically impossible, due to the complex internal structure and dimensions, i.e. The introduction of any measuring device or probe causes additional disturbance of the flow. Therefore, mathematical modeling is one of the only approaches for studying the flow of air in the nasal cavity [124].

5.1 Statement of the physical problem

Air flow that passes through the structure of the nasal cavity has a very difficult path. The complex structure of the nasal cavity and complete three-dimensional analysis of the steam flow, heat transfer in the inner part of the nasal mucosa requires significant computational resources that prevent a systematic analysis of the relevant factors (Figure 21) [124].

Having the available computational resources, a complex study of transport mechanisms was carried out in two-dimensional form, through the cross sections of the nose [124].

In addition, the following assumptions are made for numerical modeling [124]:

· The walls of the nasal cavity and nasal concha are assumed to be immovably hard.

· The air flow in the nasal cavity is considered as a laminar flow, and the air as an incompressible medium (since the Reynolds and Mach numbers are very small).

· The velocities on the walls of the cavity are taken as zero ($u = 0$, $v = 0$).

The walls of the nasal cavity are considered fully saturated with water vapor and the temperature near the body due to the moist mucous layer reaches the vascular vessels of the nasal wall.

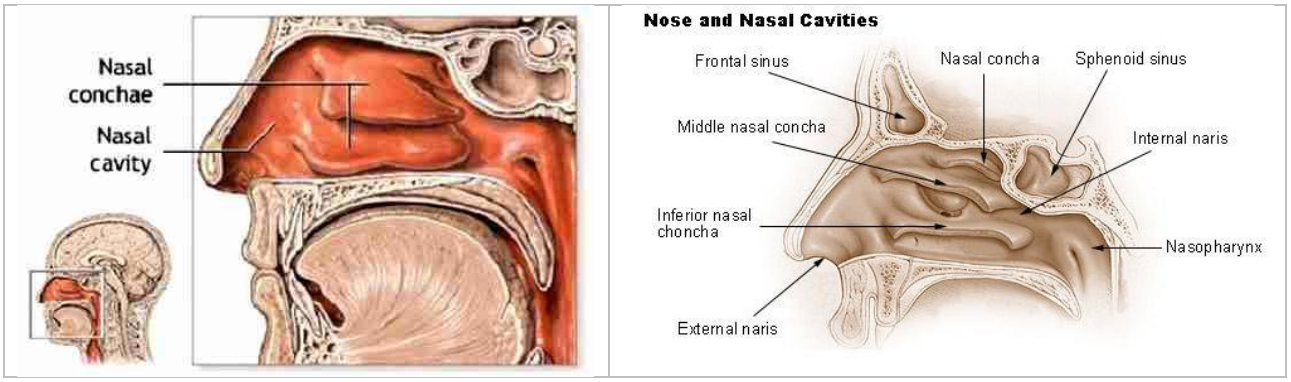


Figure 21 - Nose model with a longitudinal section.

Thin features of the nose do not have exact dimensions, because there are differences in the structure of the nasal cavity in healthy people, so it is almost impossible to determine the exact model of a "normal nose". Thus, a simplified model of the nose is developed, where the main essential signs of the nasal cavity are revealed. The dimensions are taken from the averaged data of the human nasal cavity (Figure 22). The physical area of the problem is the second cross-section (Figure 22c "-2-"), which is important for the study, because it is an area where a significant proportion of the air flow takes place, and also it has a complex structure, through which the basic functions of the nasal cavity are performed [124].

The mathematical model is constructed on the basis of the Navier-Stokes equations, including the continuity equation, the momentum equation, and also the energy (temperature) equation also relative humidity equations are used [77-79, 124].

$$\nabla U = 0,$$

$$\frac{\partial U}{\partial t} + (U \cdot \nabla)U = -\frac{1}{\rho}\nabla p + v\nabla^2 U ,$$

$$\frac{\partial T}{\partial t} + (U \cdot \nabla)T = \frac{k}{\rho c_p}\nabla^2 T ,$$

$$\frac{\partial C}{\partial t} + (U \cdot \nabla)C = D\nabla^2 C$$

where U is the velocity vector, t is the time, p is the pressure, v is the kinematic viscosity, T is the temperature, C is the humidity, c_p is the specific heat of the medium at constant pressure, k is the coefficient of thermal conductivity, ρ is the density, D is the coefficient of molecular diffusion.

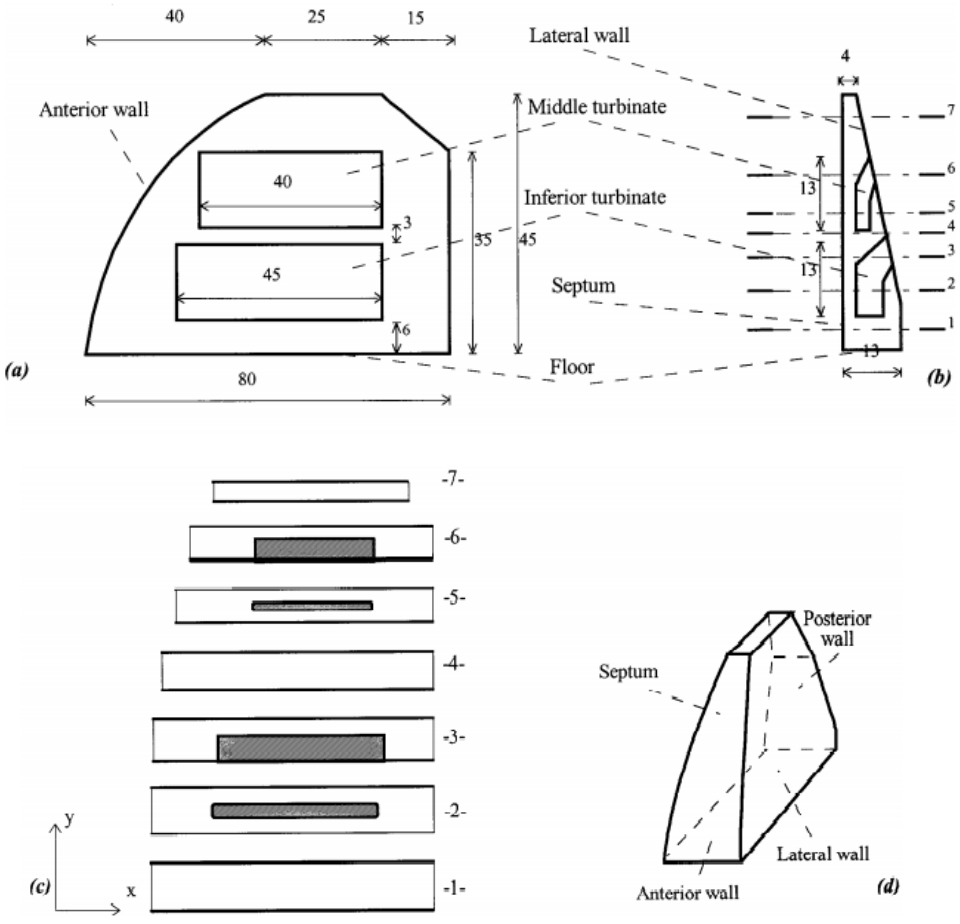


Figure 22 - Simplified nose model: a longitudinal section, b coronary section, c cross sections at height h = 3, 13, 17, 20, 26, 33, 40 mm from the bottom point of the nasal cavity, d perspective view.

The instantaneous velocity at the inlet in each cross section is assumed to have a parabolic profile with a maximum velocity (UM)max that varies during the breathing cycle. In the paper of Girardin et al. [80] measurements were made using laser anemometry in the model of the human nose and it was found that the field flow basically have layered parabolic velocity profiles in any cross section. At rest, a normal adult breathes a volume about VT = 0.5 L (inhaling and exhaling) once a minute at an average flow rate of about 0.125 L/s to each nostril. Accordingly, the instantaneous velocity distribution at the input UM in the direction is given in the following form [124]:

$$u_{in}(t, x=0, y) = (U_{in}^{M})_{\max}\left[2\sin^2\frac{\pi t}{2} - 1\right] \times \frac{(12y - y^2)}{36}$$

The inlet boundary conditions for the temperature and relative humidity of the external air are given in the following form:

$$T_{in}(t, x=0, y) = 25°C\,,\ C_{in}(t, x=0, y) = 0.0047\ kg\ H_2O / m^3$$

On the walls of the nasal cavity and nasal concha:

$$u_{wall}(t,x,y)=0,\ v_{wall}(t,x,y)=0,\ T_{wall}(t,x,y)=37°C,\ C_{in}(t,x,y)=0.0438\ kg\ H_2O/m^3$$

The initial conditions are given in this form:

$$u_0(t=0)=0,\ T_0(t=0)=32°C.,\ C_0(t=0)=0.0235\ kg\ H_2O/m^3.$$

5.2 Numerical algorithm

The projection method is used for a numerical solution of this system of equations [81, 82]. Equations are discretized by the finite volume method [81, 83, 84]. At the first stage it is assumed that the transfer of the momentum is carried out only through convection and diffusion, and an intermediate velocity field is calculated by the fourth-step Runge-Kutta method [79]. In the second stage, according to the found intermediate velocity field, there is a pressure field. The Poisson equation for the pressure field is solved by the Jacobi method. Then at the third stage, it is assumed that the transfer is carried out only due to the pressure gradient. Further at the fourth stage, the equations for the temperature are calculated by the fourth-step Runge- Kutta method. And in finally, equations for the relative humidity are calculated, and also solved by the fourth-step Runge-Kutta method [72, 73, 79, 124].

5.3 Parallel implementation

Well known three dimensional lid-driven cavity problem is used to check various geometric decompositions method. A computational grid was constructed using the Pointwise software to carry out numerical simulation. The problem was launched on the ITFS-MKM software complex using a high-performance cluster. A cluster system (Intel(R) Xeon(R) CPU E5645 2.40 GHz CPU, 26 nodes with two processors per node and totally number of cores are 312, 624 GB RAM) is used to decrease CPU time. This numerical algorithm is completely parallelized using various geometric decompositions (1D, 2D and 3D). Geometric partitioning of the computational grid is chosen as the main approach of parallelization. In this case, there are three different ways of exchanging the values of the grid function on the computational nodes of a one-dimensional, two-dimensional, and three-dimensional grid. After the decomposition stage, when parallel algorithms are built on separate blocks, it was proceed to the relations between the blocks, the calculations on which will be performed in parallel on each processor. For this purpose, a numerical solution of the equation system was used for an explicit scheme, since this scheme is very well parallelized. In order to use the decomposition method as a parallelization method, this algorithm uses the boundary nodes of each subdomain in which it is necessary to know the value of the grid function that borders on the neighboring elements of the processor. To achieve this goal, at each computational node, fictitious

points store values from neighboring computational nodes, and organize the transfer of these boundary values necessary to ensure homogeneity of calculations for explicit formulas (Figure 23) [124].

Data transmission is performed using the procedures of the MPI library [61, 124].

By doing preliminary theoretical analysis of the effectiveness of various methods of the computational domain decomposition for this problem, it will be estimated that the time of the parallel program as the time of the sequential program divided by the number of processors used Tcalc, plus the transmission time Tp = Tcalc⁄p + Tcom. While transmissions for different decomposition methods can be approximately expressed through bandwidth [124]:

$$T_{com}^{1D} = t_{send}\, 2N^2 x2$$

$$T_{com}^{2D} = t_{send}\, 2N^2 x4p^{1/2} \qquad (20)$$

$$T_{com}^{3D} = t_{send}\, 2N^2 x6p^{2/3}$$

where N^3 - the number of nodes in the computational grid, p - the number of processors (cores), t_{send} - the time of sending one element (number) [124].

It should be noted that for different decomposition methods, the data transmission cost can be represented as $T_{com}^{1D} = t_{send}\, 2N^2 xk(p)$ in accordance with the formula (20), where $k(p)$ is the proportionality coefficient, depending on the decomposition method and the number of processing elements used [124].

Table 2 shows numerical values $k(p)$. It can be seen that if p> 5 and using 3D decomposition this algorithm is more efficient, and for p> 11 and using 3D decomposition, the necessary time of sending between the processors of the value of the function $u_{i,j,k}^{n+1}$, $v_{i,j,k}^{n+1}$, $w_{i,j,k}^{n+1}$, $p_{i,j,k}^{n+1}$ in a node with a smaller number of elements, it will be expected that the time spent on data transmission will be minimal (Figures 24-26) [124].

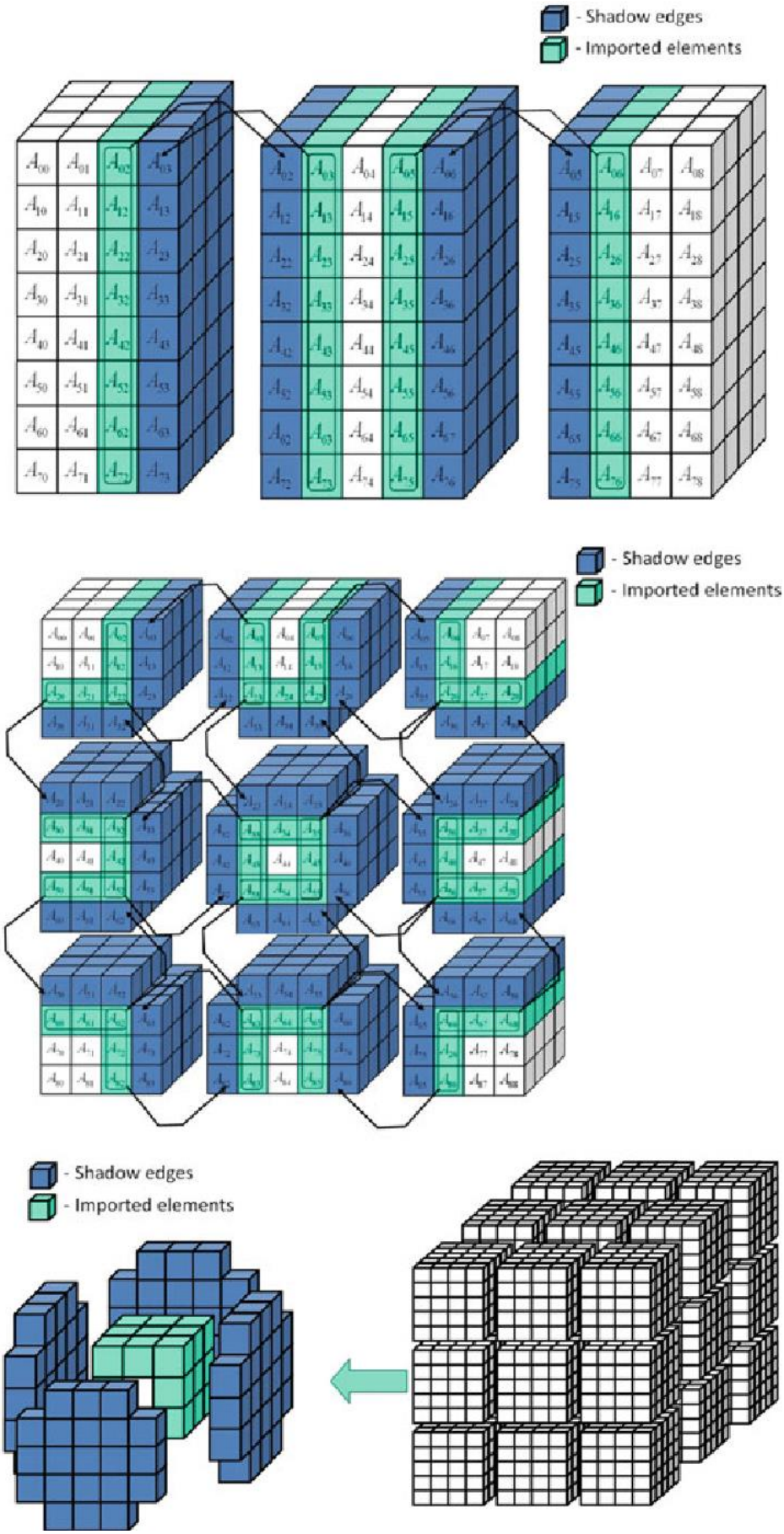


Figure 23 - Different methods of decomposition. Schemes of mechanisms for the exchange of 1D, 2D and 3D decompositions.

Table 2. Dependence of the proportionality $k(p)$ coefficient on the number of processor elements and the decomposition method.

Number of processes	3	4	5	6	10	11	12	16	60	120	250
1D Decomposition	2	2	2	2	2	2	2	2	2	2	2
2D Decomposition	2,31	2,00	1,79	1,63	1,26	1,20	1,15	1,00	0,51	0,36	0,25
3D Decomposition	2,88	2,38	2,05	1,82	1,29	1,21	1,14	0.94	0,39	0,24	0,15

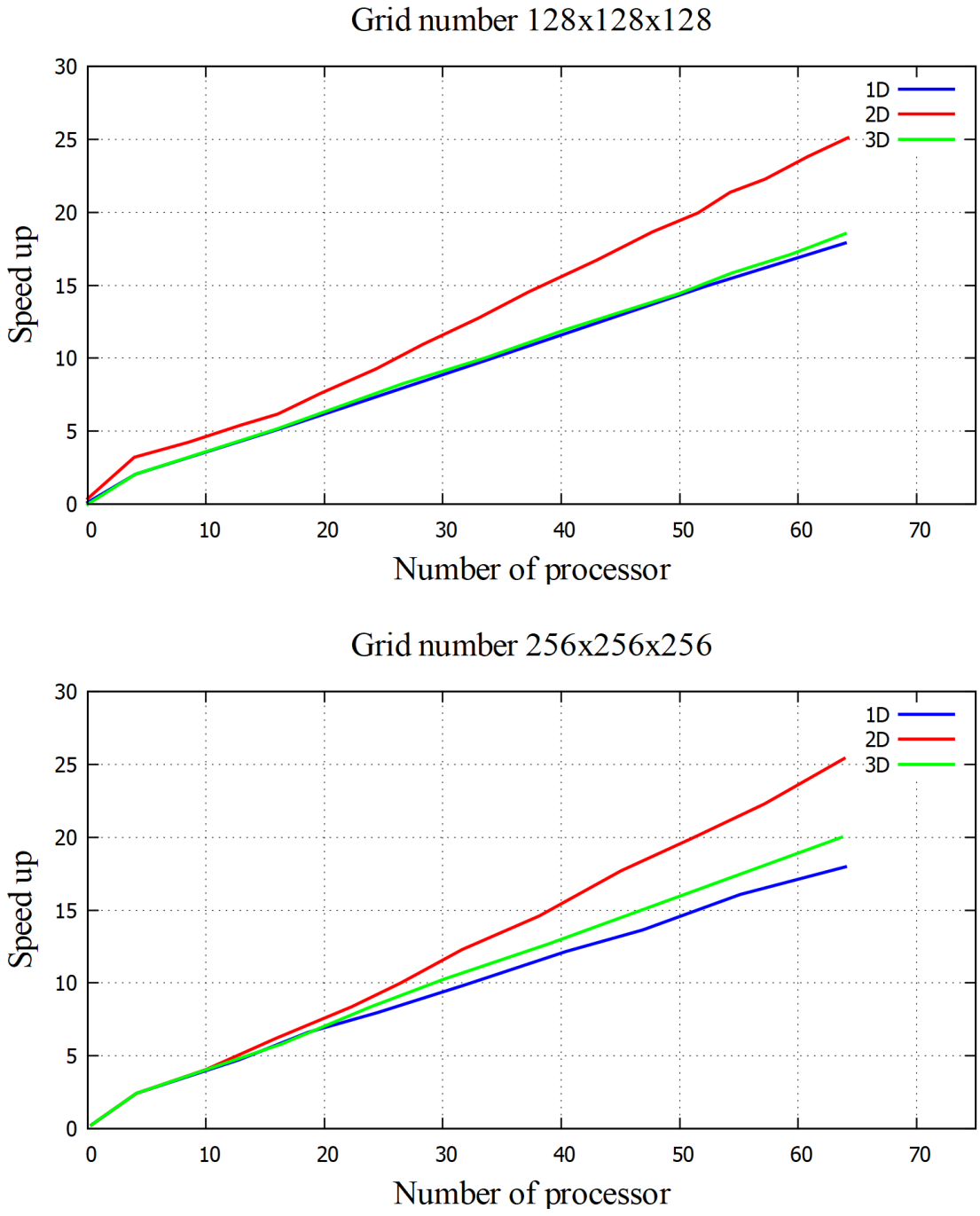


Figure 24 - Speed-up for various methods of decomposition of the computational domain.

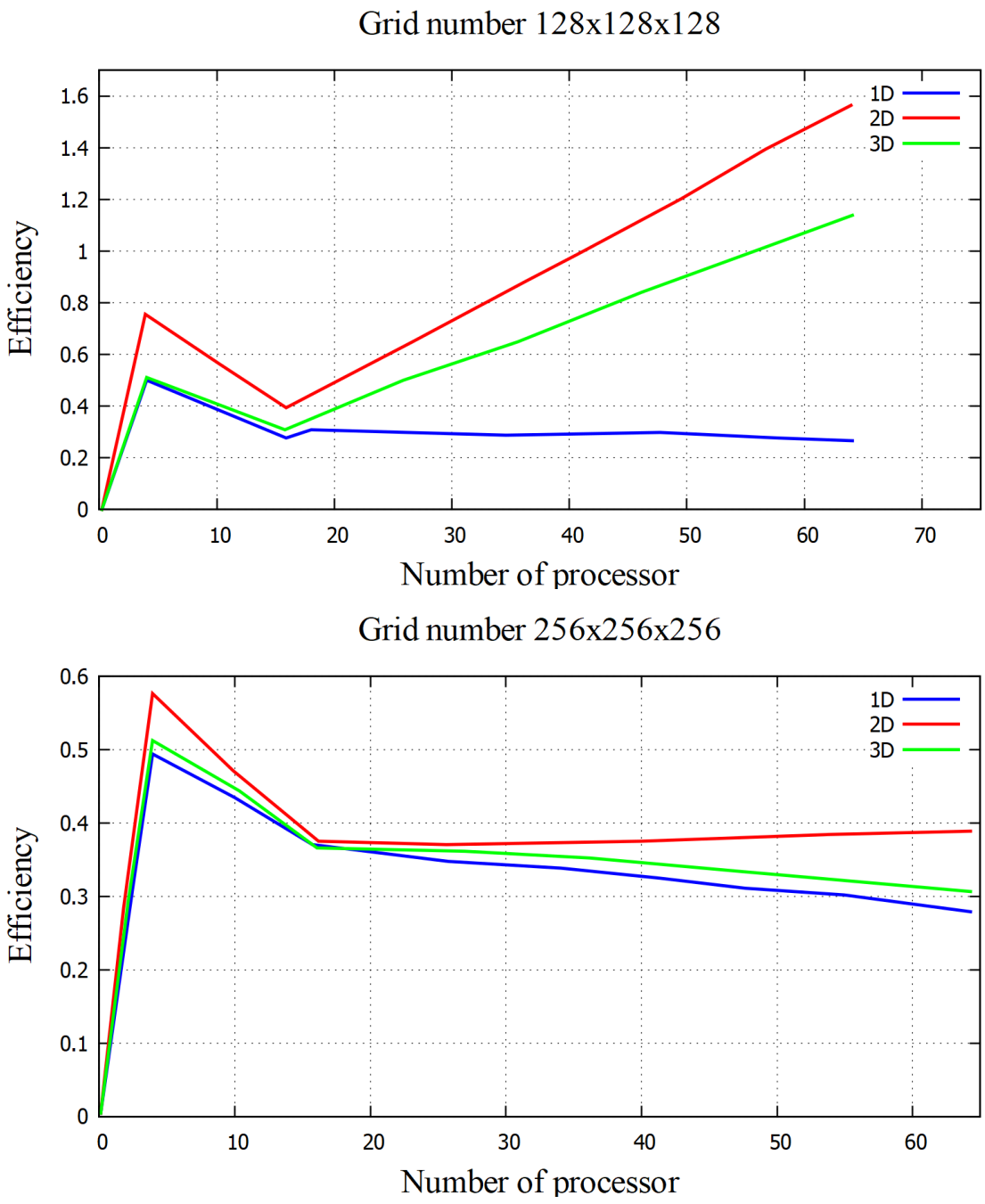


Figure 25 - Efficiency for various methods of decomposition of the computational domain.

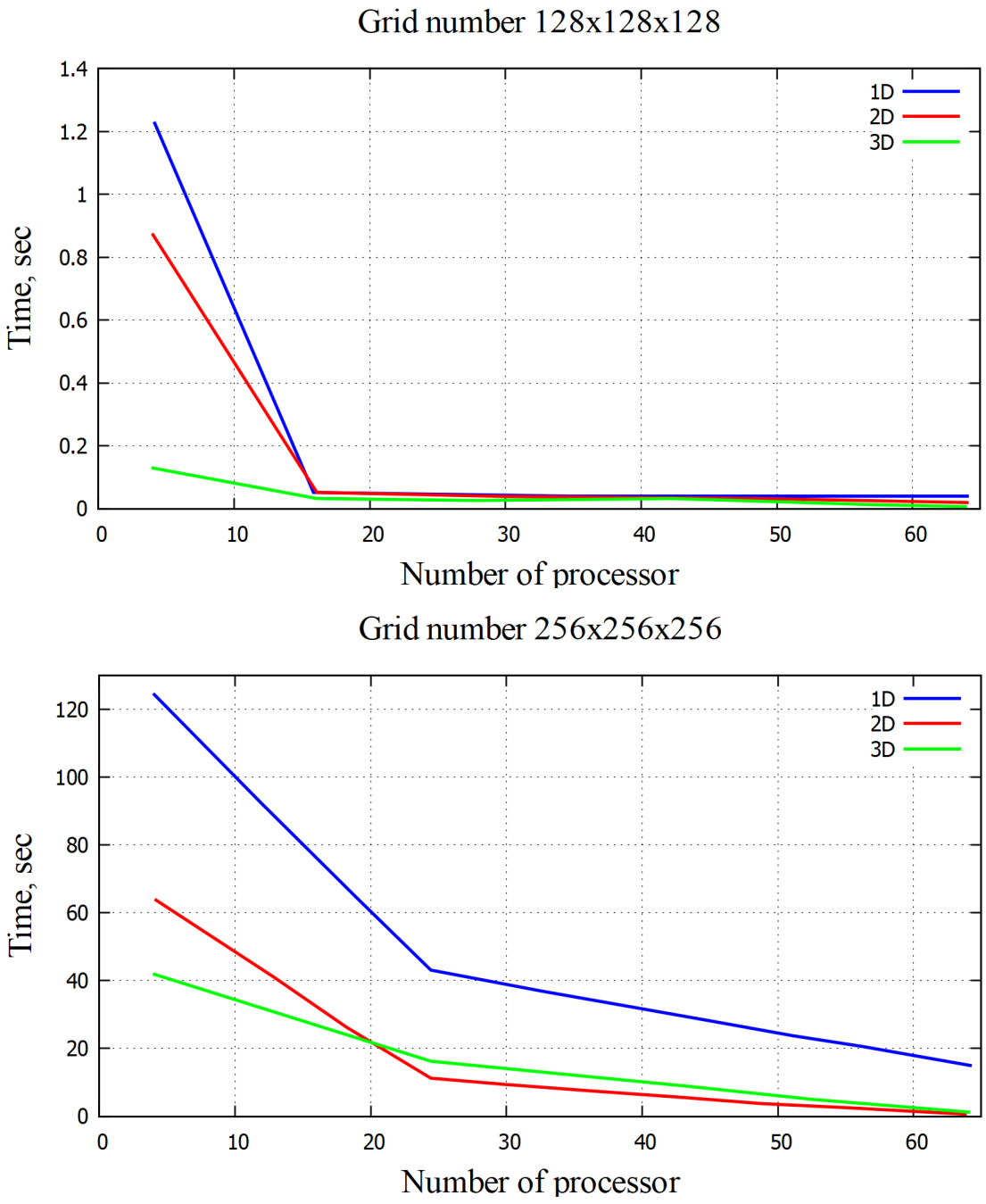


Figure 26 - Time of calculation without taking into account the cost of data transfer for different decomposition methods.

Comparison of the results of parallelization by simple MPI procedures and the hybrid OPENMP/MPI model of the Fourier method for solving the Poisson equation for calculating the pressure field is presented in Figure 27. The problem was solved with the distribution of the computational rectangular grid into computational subdomains for each processor. The exchange of the required calculated values between the processors is implemented using the MPI library for a simple MPI procedure and the MPI and OPENMP libraries for the OPENMP/MPI hybrid model.

All calculations were carried out on cluster systems T-Cluster and URSA at the Faculty of Mechanics and Mathematics, al-Farabi Kazakh National University using

128 × 128 × 128 and 256 × 256 × 256 computational grids. Computational experiments were conducted using up to 250 processors. The results of the computational experiment showed the presence of a good speed in solving problems of this class. They are mainly focused on additional transmissions and time calculations for various decomposition methods [124].

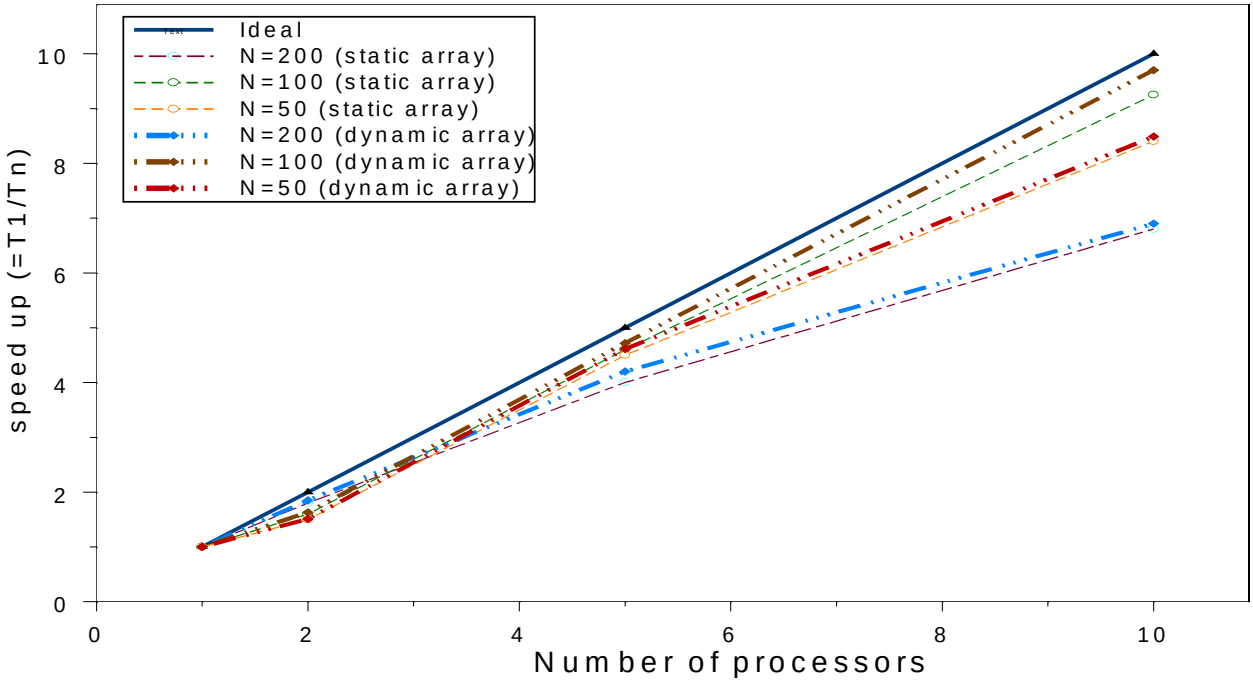


Figure 27 - Dependence of acceleration on the number of processors used for simple MPI and hybrid OPENMP/MPI model.

At the first stage, one common program was used, the size of the array from start to run did not change, and each element of the processor is numbered by an array of elements, starting from zero. Despite the fact that according to the theoretical analysis of 3D decomposition is the best option for parallelization (Figure 24), computational experiments showed that the best results were achieved with 2D decomposition when the number of processes varies from 25 to 144 (Figure 24) [124].

Based on a preliminary theoretical analysis of the graphs, the following character can be noted. The calculation time without the cost of inter-processor communications with different decomposition methods should be approximately the similar for the equal number of processors and shrink by Tcalc⁄p. In fact, the calculated data (Figure 25) show that using 2D decomposition on various computational grids gives a minimal cost for the calculation and the cost graphs are significantly higher depend- ing on the computation time on several taken processors Tcalc⁄p (Figure 26) [124].

To explain these results, it is necessary to pay attention to the assumptions made in the preliminary theoretical analysis of the effectiveness for this task. First, it was assumed that, regardless of the distribution of data per one processor element, the same amount of computational work was performed, which should lead to the same time expenditure. Secondly, it was assumed that the time spent on interprocessor sendings of any degree of the same amount of data does not depend on their choice of

memory. In order to understand what is actually happening, the following sets of test computational calculations were carried out. For the evaluation, the sequence of the first approach was considered, when the program is executed in a version with one processor, and thus simulates various ways of geometric decomposition of data for the same amount of calculations performed by each processor [124].

5.4 Results of numerical calculation

As a result of numerical modeling of the aerodynamics of the human nasal cavity, the following data were obtained. Also, to verify this numerical algorithm, the calculation data from paper [77] has been used, which describes the profiles of the longitudinal component of velocity and temperature in three cross sections: at a distance x1=17 mm, x2=49 mm and x3=80 mm from the entrance (Figure 28). For the numerical simulation, the corresponding parameters for air constants were used: ρ=1.12 kg/m3, μ=1.9× 10−5 kg/ms, cp=1005.5 J/kgK, k=0.0268 W/mK, D=2.6×10−5 m^2/s [124].

Figure 29 shows the comparison of profiles for x1, x2 and x3 the longitudinal velocity component of the calculation results and data from the article Naftali et al. [77]. Figure 30 shows a comparison of temperature profiles for cross sections x1, x2 and x3. Figure 31 shows the relative humidity profiles for cross sections x1, x2 and x3. In all the figures, numerical results were shown to be dimensionless [124].

It can be seen from the figures that when passing through narrow areas of the nasal cavity air is heated downstream, and relative humidity also increases. And also from Figure 30 it can be noted that behind the nasal septum the temperature increases and the air temperature is heated to the alveolar state when reaching to the nasopharynx. And at low ambient temperatures, relative humidity plays a very important role. Figure 32 shows the longitudinal velocity component in the cross section for time t=1 s. Figure 33 shows the transverse velocity component in the cross section for time t=1 s. It can be seen from the figures that vortex currents appearing from the nasal conchas, which play an important role in the process of heating the air. Figures 34 and 35 show the temperature components, and the relative humidity in the calculated area for time t = 1 s. It can be seen from the figures that when passing through narrow areas of the nasal cavity air is heated downstream, and relative humidity also increases [124].

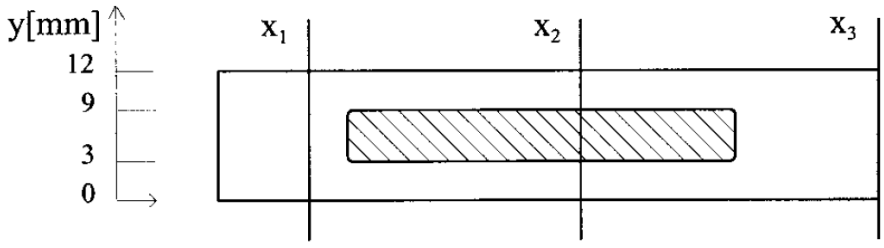


Figure 28 - Evaluation in three locations for temperature and velocity for a cross section.

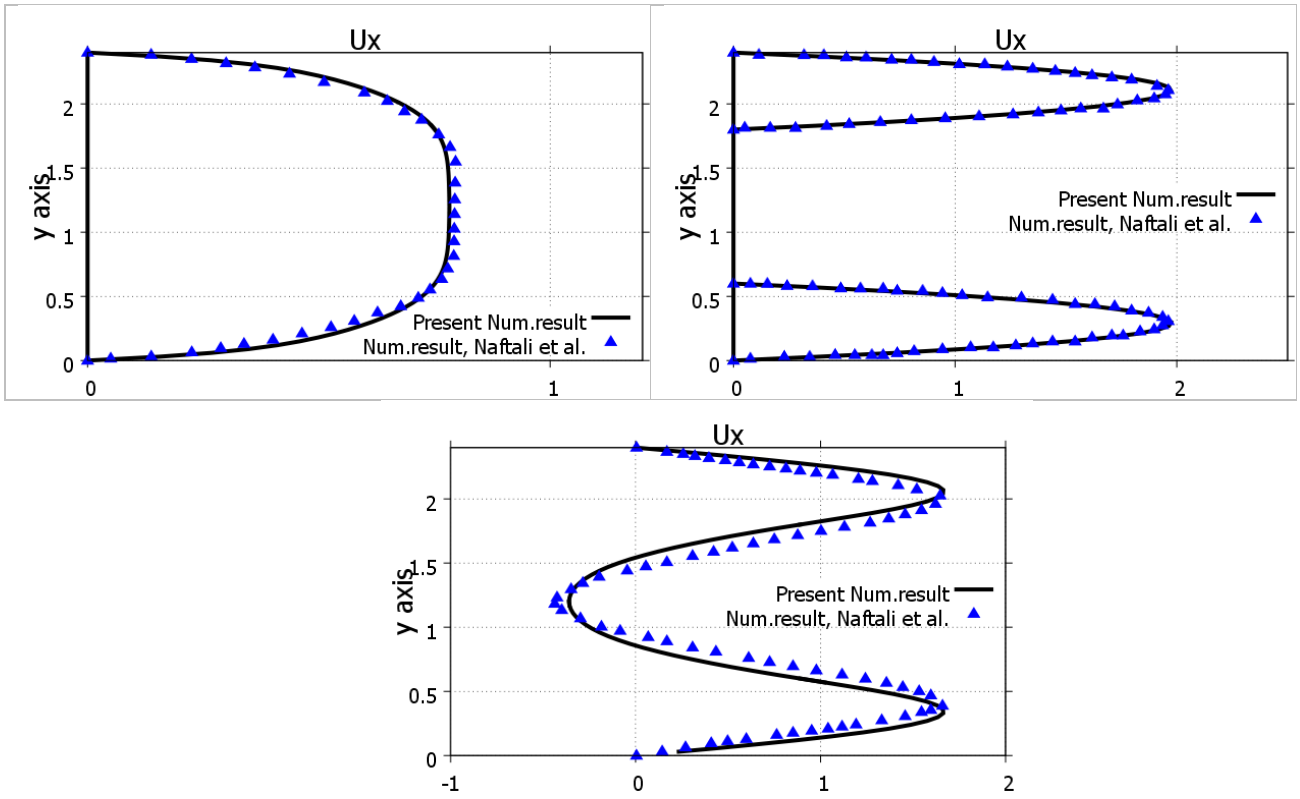


Figure 29 - Comparison of the profile of the velocity component for the cross sections $x_1 = 17$ mm, $x_1 = 49$ mm and $x_1 = 80$ mm with the results of calculations from [77].

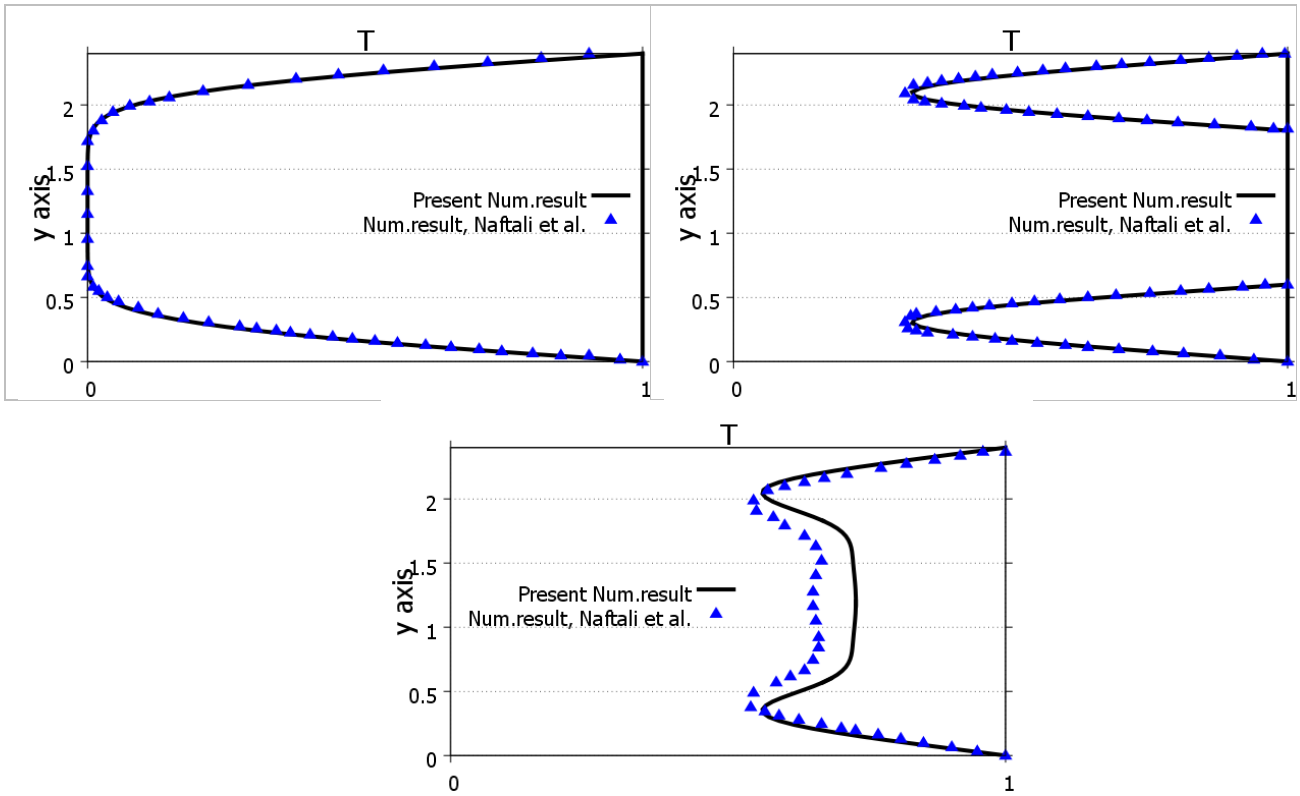


Figure 30 - Comparison of temperature profiles for cross sections $x_1 = 17$ mm, $x_1 = 49$ mm and $x_1 = 80$ mm with the results of calculations from [77].

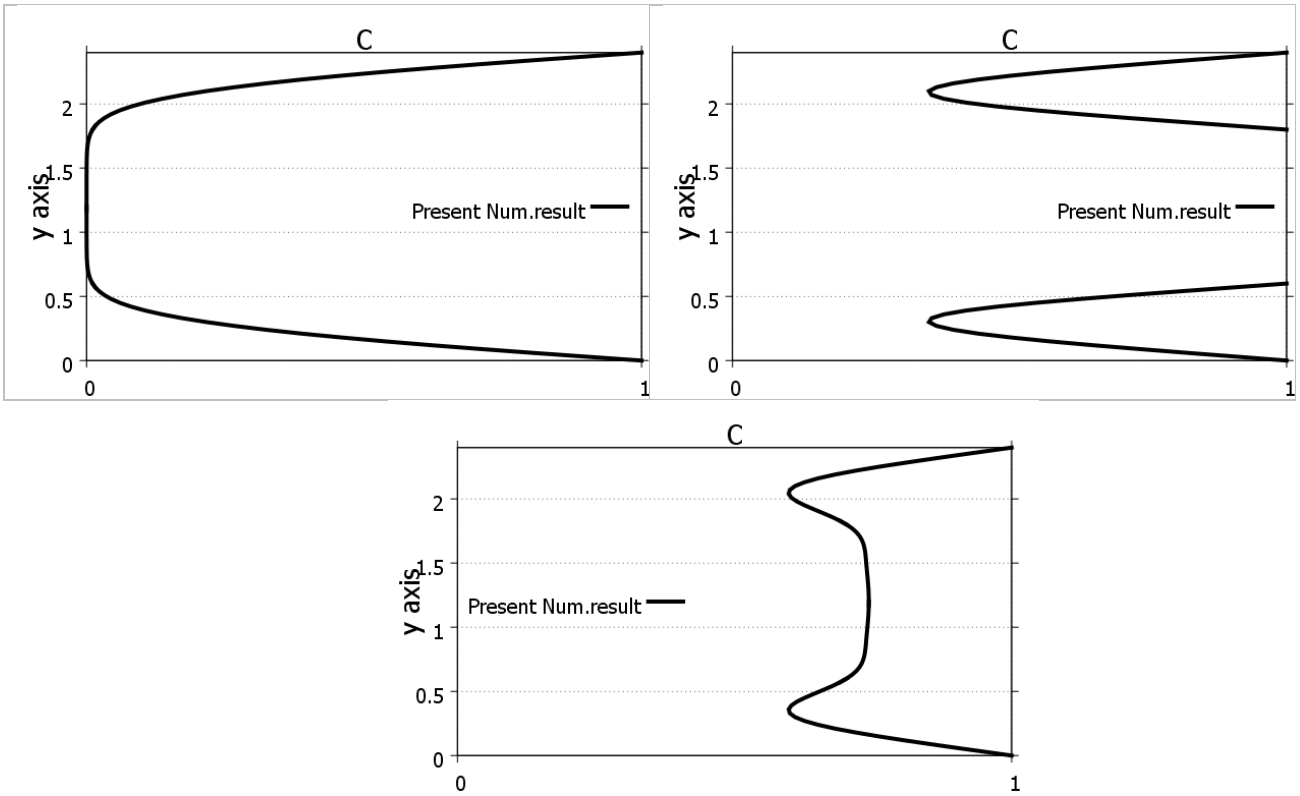


Figure 31 - Relative humidity profiles for sections $x_1 = 17$ mm, $x_1 = 49$ mm and $x_1 = 80$ mm.

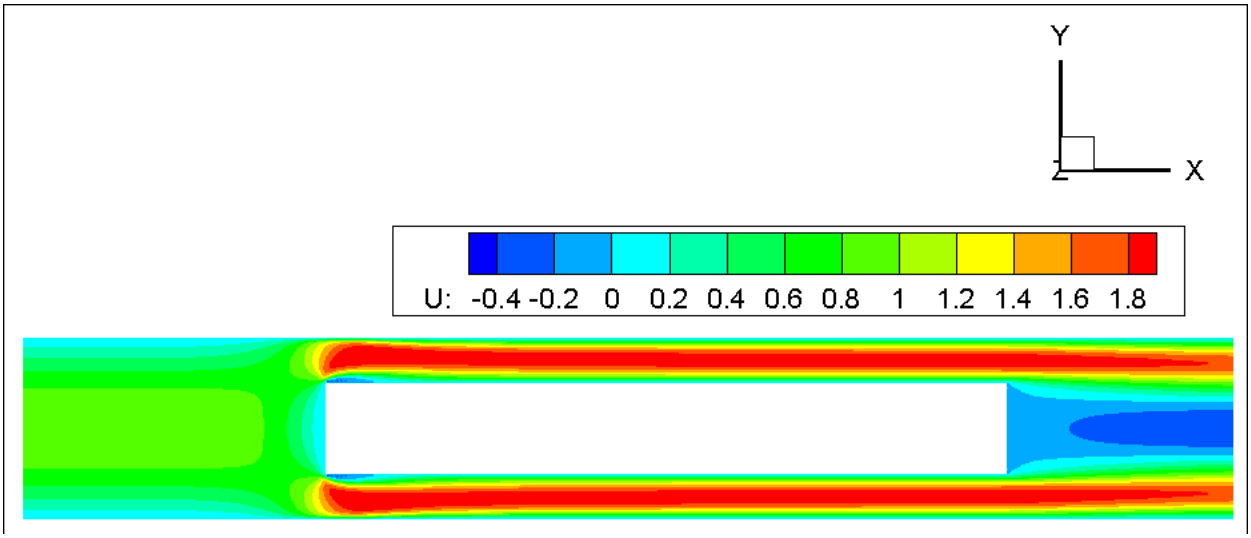


Figure 32 – Longitudinal components of the flow velocity for a cross section.

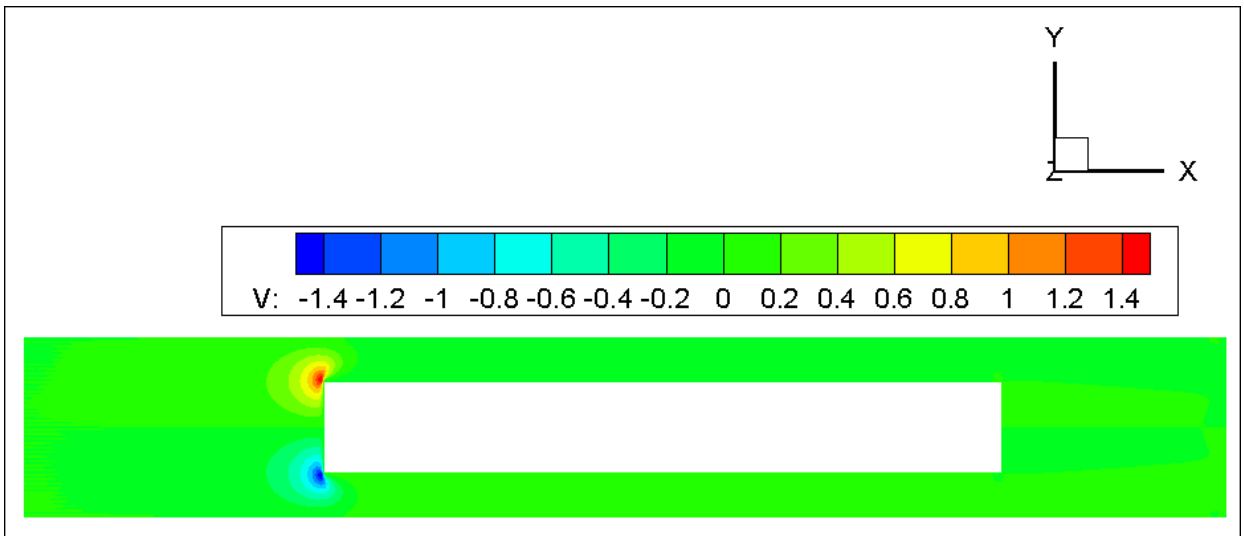


Figure 33 – Transverse flow velocity components for a cross section.

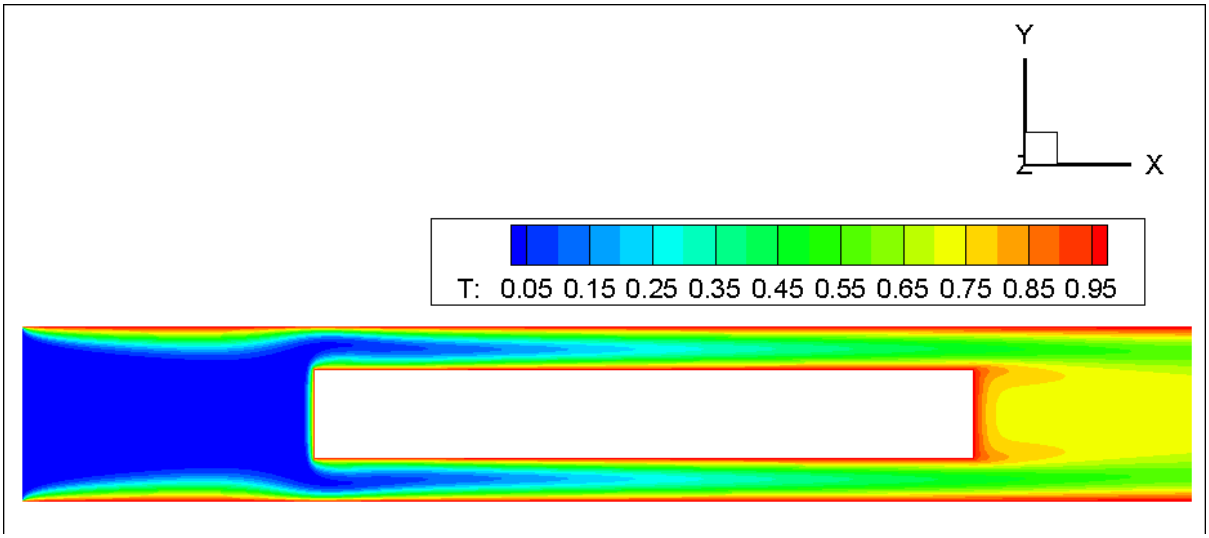


Figure 34 – Flow temperature for cross section.

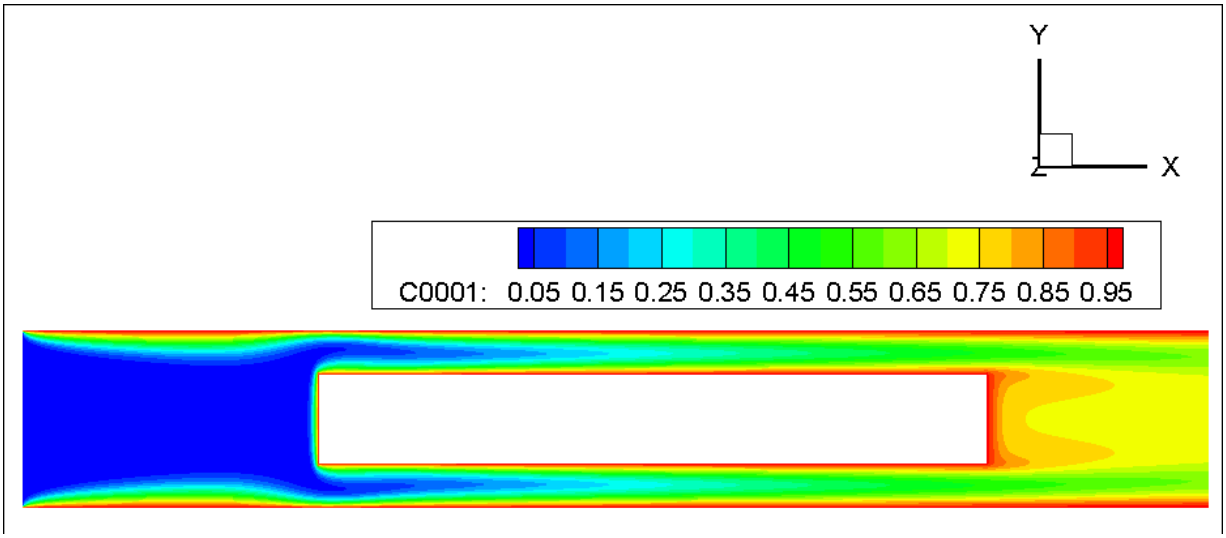


Figure 35 – Relative flow humidity for cross section.

5.5 Statement of the 3D Physical Problem

To approximate the obtained numerical results to the real result, it is necessary to use three-dimensional models of the nasal cavity, since the heating and humidification of the inhaled air strongly depends on the structure of the walls of the nasal cavity. Therefore, computational models must include more realistic 3D geometry descriptions in order to determine the effect of complex geometry on various characteristics such as speed, heat and mass transfer, and relative humidity [125].

In [79] (Hahn et al., 1993), experimental measurements of air flow in a prototype nose were carried out using a hot wire anemometer in a large-scale physical model of the nasal cavity, built on the basis of computed tomography of the right nasal cavity of a healthy man. The experiments used different breathing rates equivalent to 180 ml/s, 560 ml/s and 1100 ml/s in a real human nose, which corresponds to calm breathing, medium inhalation and intense inhalation, respectively. Figure 36 shows the experimental setup and the relative position of the measurement sites on three slices that were used to verify the numerical values obtained [125].

$$\nabla U = 0,$$

$$\frac{\partial U}{\partial t} + (U \cdot \nabla)U = -\frac{1}{\rho}\nabla p + \nu \nabla^2 U ,$$

where U is the velocity vector, t is the time, p is the pressure, ν is the kinematic viscosity, ρ is the density.

The computational model included the region from the anterior tip of the nose to the posterior end of the turbinate. As shown in Figure 36, the nasopharynx was dilated to fit the experimental setup [125].

The geometry of the human nasal cavity was created by aligning and processing 40 computed tomography (CT) scans of the airways of a healthy man. Using the AutoCAD software package, intermediate geometric shapes of the nasal cavity were created, which correspond to the average physical parameters of the human nasal cavity. From these idealized 2-D images (Figure 37), a 3-D complex human nasal cavity was created. The locations of the anemometers in the study area are shown in Figure 38 [125].

As can be seen from Figure 38, on sections 1 and 2, 4 lines were located to measure the speed at these points of the section. The computational grid of the area under study is shown in Figure 39. The final computational grid of the nasal cavity consisted of 6,876,463 elements (Figure 39). Comparisons of the velocity profiles with experimental data [79] (Hahn et al., 1993) on lines 1-4 are shown in Figure 40 [125].

Velocity profiles were dimensioned according to the value of the local maximum velocity. As shown in Figure 40, the predictions of the laminar model are in good agreement with experimental data [79] (Hahn et al., 1993). The directions of the air flow in the nasal cavity and two-dimensional contours in sections 1-3 are shown in Figures 41 and 42. As expected, in the narrow channels of the nasal cavity the air flow accelerated and reached a maximum value of 3.33 m/s (Figure 41). You can also notice that due to the deep conchas of the nasal cavity, vortices arise, which in the subsequent can have a good effect on heat and mass transfer. The two-dimensional velocity contours presented in Figure 42 confirm the statements about the acceleration of the air flow due to the narrowing of the turbinates and, as a consequence, the occurrence of vortices [125].

Despite some irregularities in the flux field profiles, it can be seen that the proposed model achieves good agreement with the experimental results [79] (Hahn et al., 1993) under calm breathing conditions. It can also be noted that the obtained numerical results show values much closer to the experimental data [79] (Hahn et al., 1993) than the numerical results obtained in [86] (Li et al., 2017). However, the obtained inaccuracies in this work can be explained by the fact that the fine features of the nasal cavity cannot be accurately measured due to the limited resolution of existing imaging methods. Accordingly, a model similar to the nasal cavity was developed (Figure 36), in which the dimensions are taken from the average data of the human nasal cavities. The model used allows a comprehensive study of a large number of functions of the nasal cavity in relation to the structural components of the nasal cavity and the corresponding heat and mass transfer. It should be noted that in real conditions the walls of the nose may not be the same or equal to the alveolar conditions, especially during exercise, and future models should refine these assumptions [125].

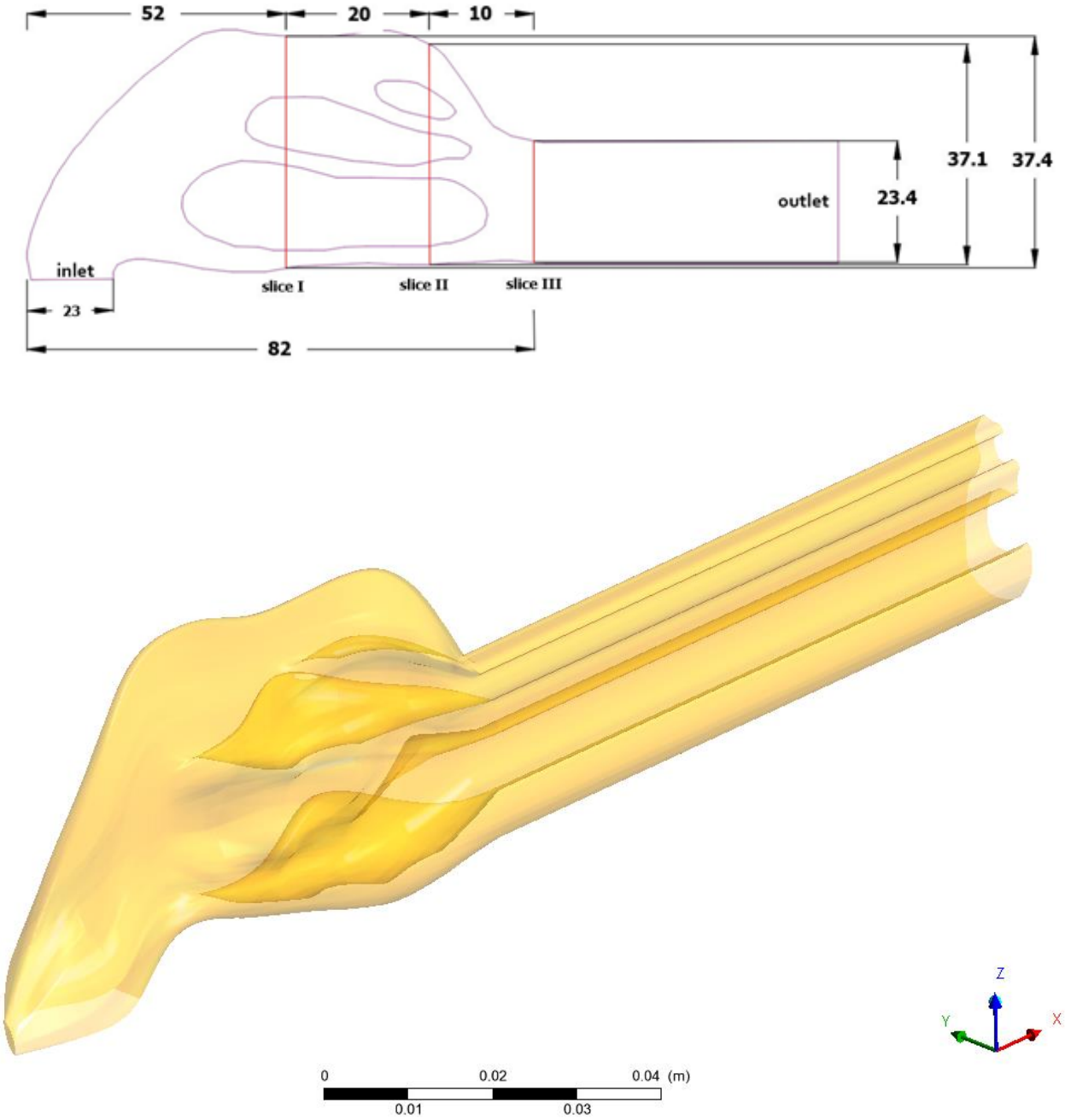


Figure 36 - Physical geometry of the study area (all dimensions in mm).

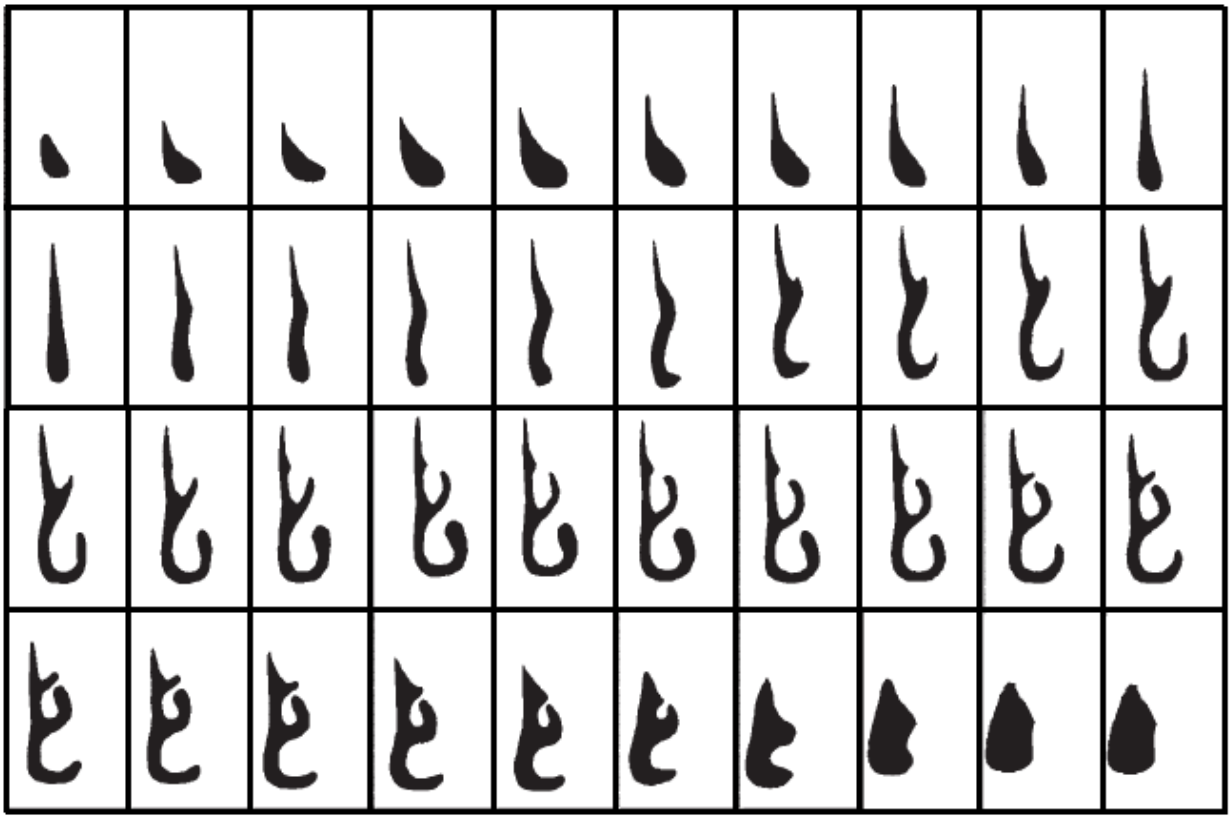


Figure 37 - Digitized 2D sections of computed tomography (CT) images of the human nasal cavity.

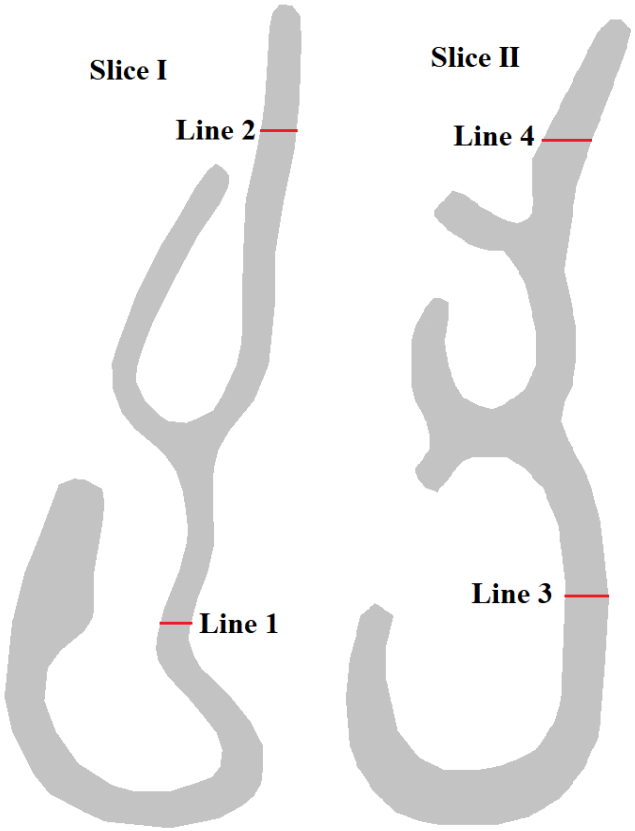


Figure 38 - Locations of line 1-4 on sections 1 and 2.

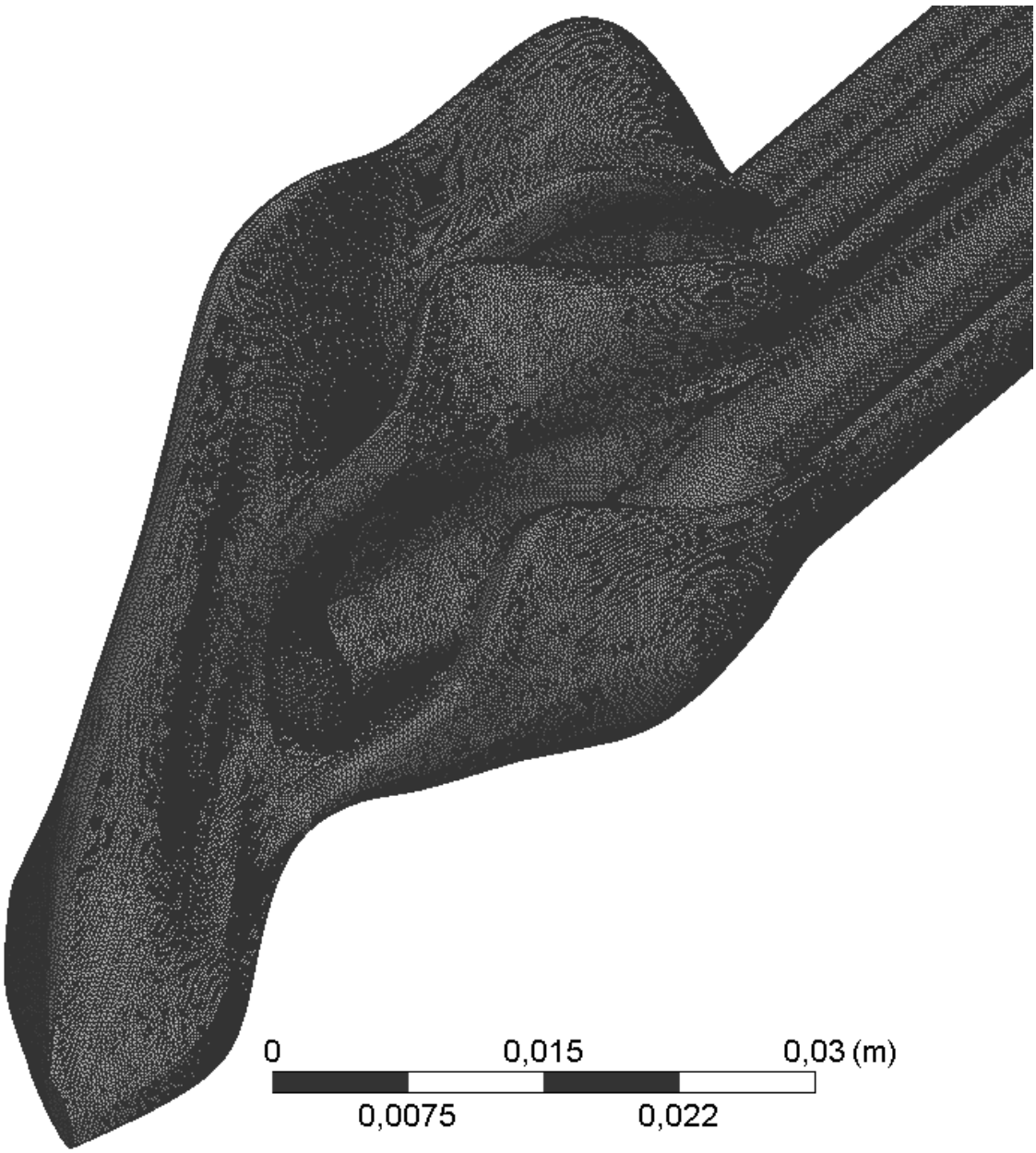
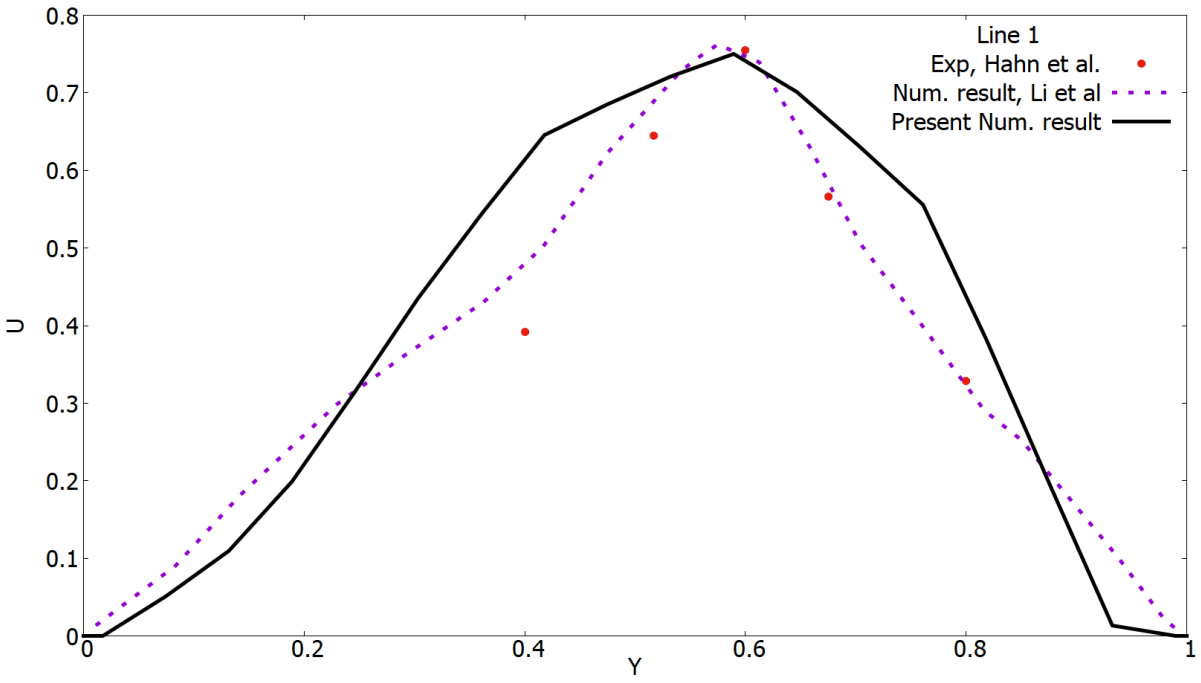
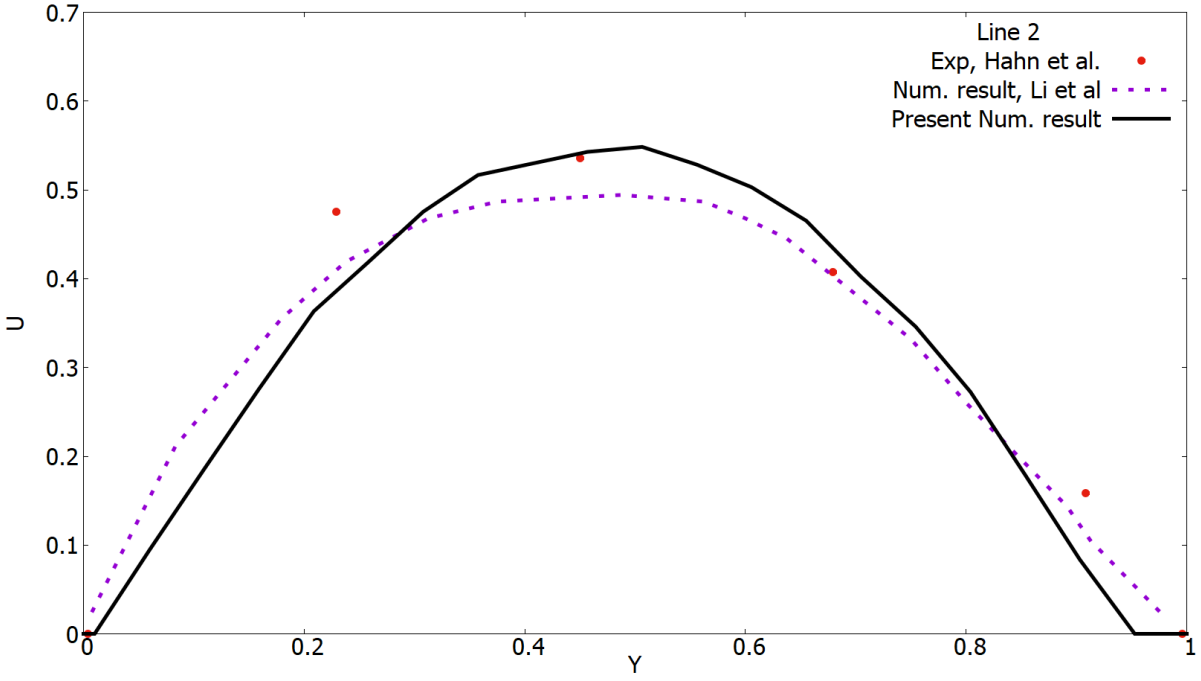


Figure 39 - Computational grid of a complex human nasal cavity.


Line 1
Exp, Hahn et al.
Num. result, Li et al
Present Num. result
U
Y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
0.2
0.4
0.6
0.8
1


Line 2
Exp, Hahn et al.
Num. result, Li et al
Present Num. result
U
Y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
0.2
0.4
0.6
0.8
1

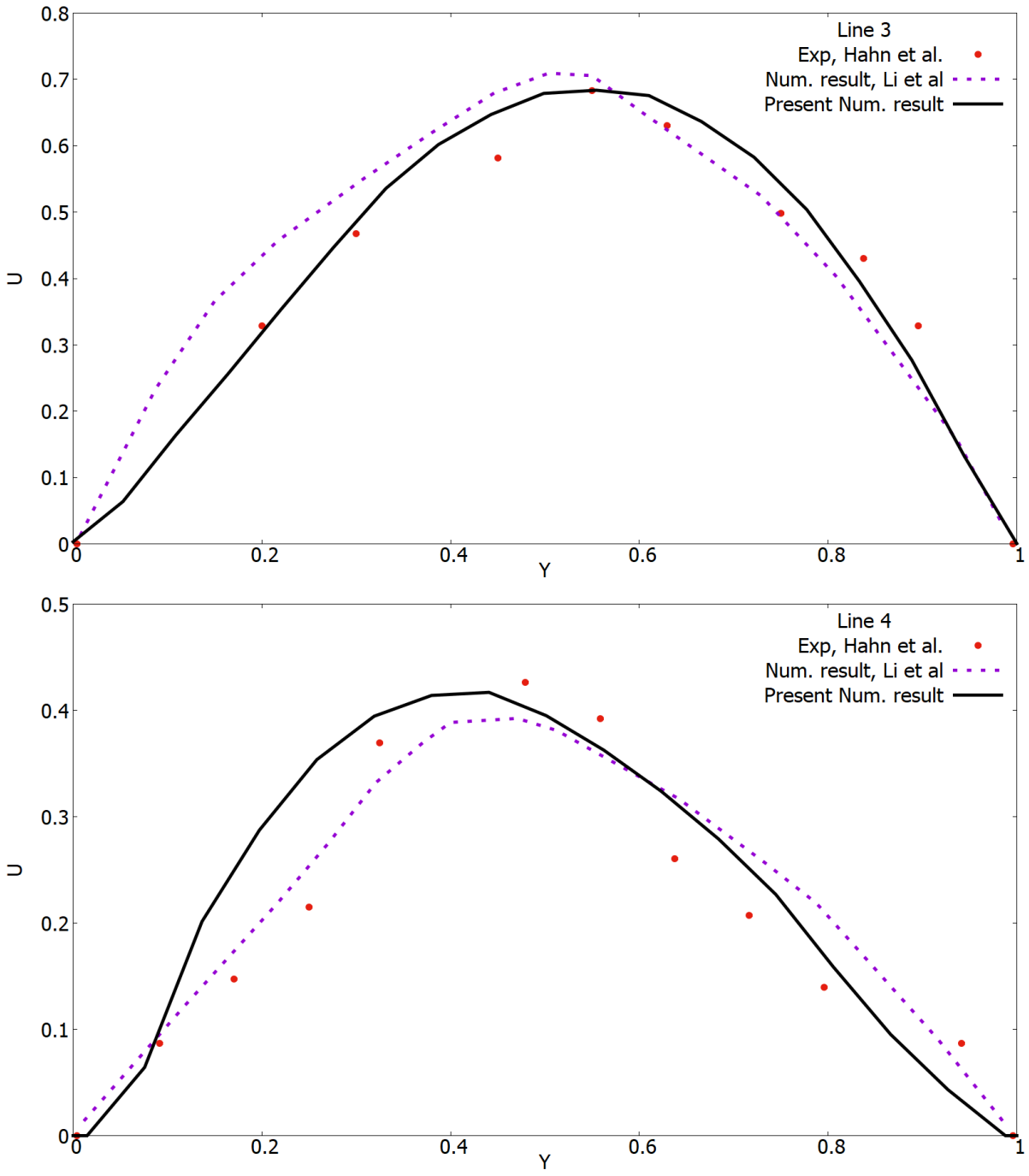


Figure 40 - Comparison of the profiles of the horizontal velocity component on lines 1-4 with the numerical results of other authors [86] (Li et al., 2017) and experimental data [79] (Hahn et al., 1993).

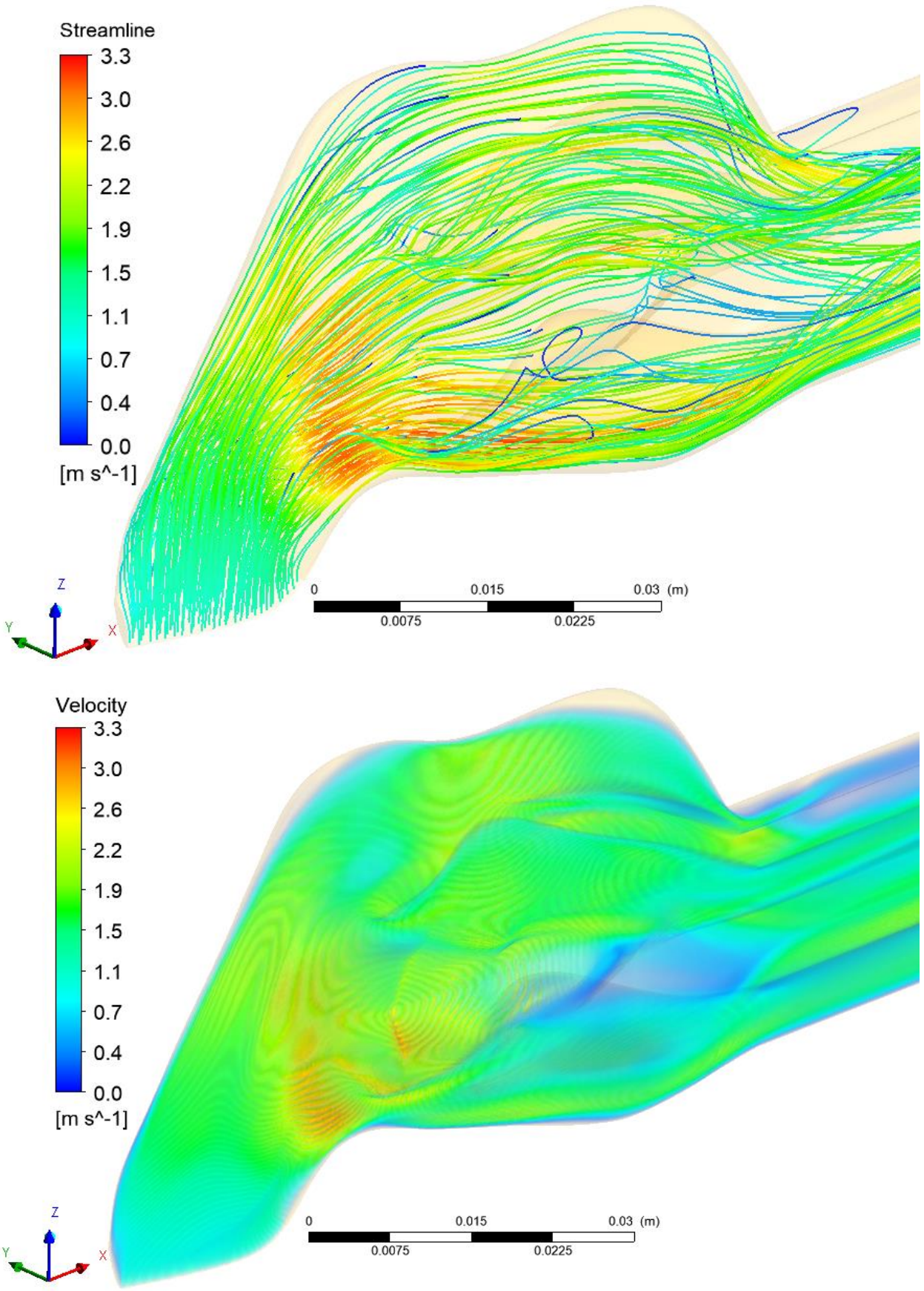


Figure 41 - Longitudinal components of the air flow velocity with delineation of streamlines in the nasal cavity.

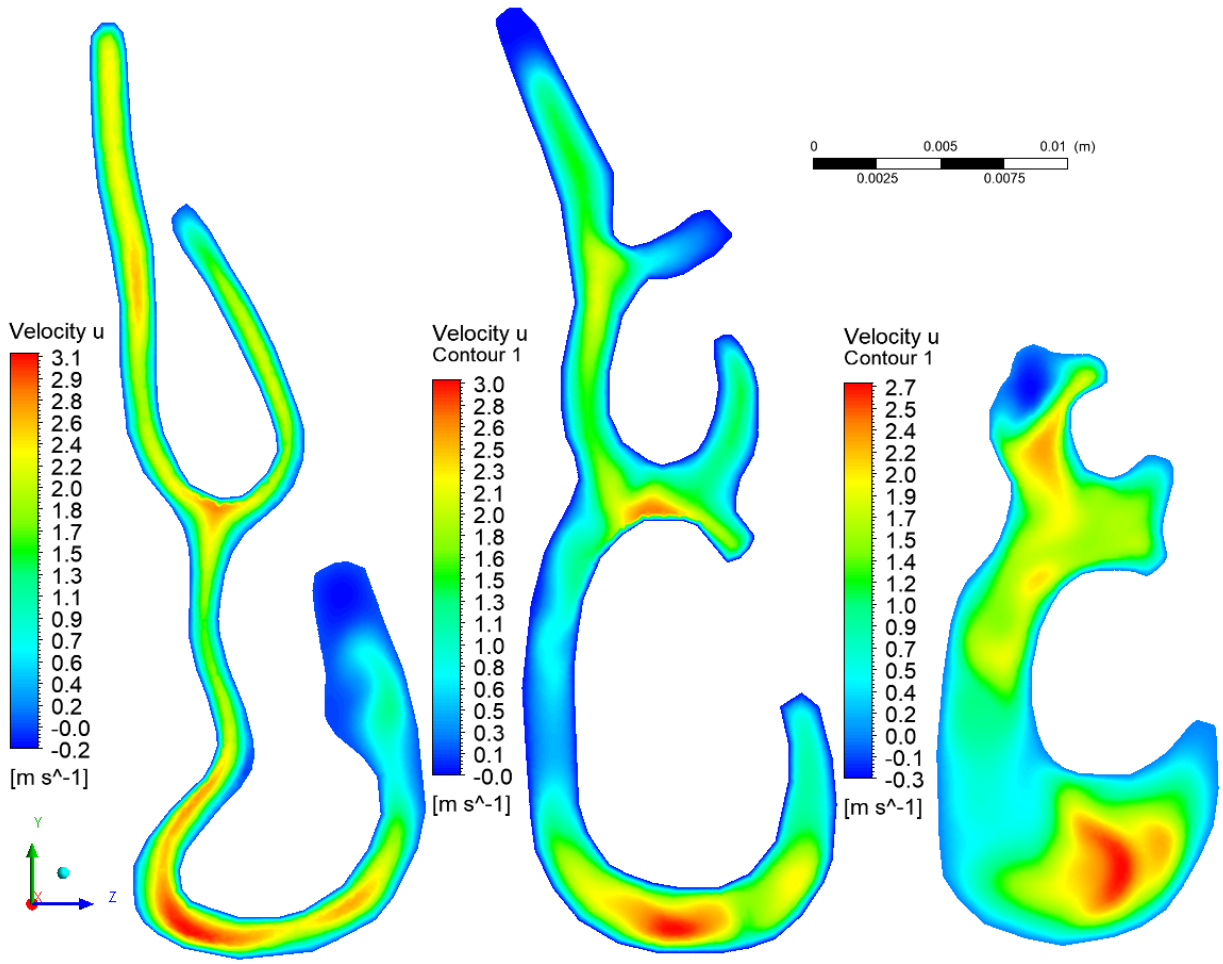


Figure 42 - Two-dimensional contours of the longitudinal components of the cross-sectional velocity 1-3.

5.6 Numerical study of heating and humidification of air in the human nose

The flow of air in the human nasal cavity plays an important role in many physiological functions of the nose, such as heating and humidifying the air flow, and others. In this section, the proposed model is used to predict airflow and related transport phenomena in the human nasal cavities [125].

It is assumed that the flow of heat and water vapor is released from the inside of the nasal mucosa. Normal respiration was chosen as a reference base, and then the effect of changes in ambient temperature was investigated. This study serves as a basis for a better understanding of nasal transport phenomena (heat, mass), which are the main functions of the nose [125].

$$\nabla U = 0,$$

$$\frac{\partial U}{\partial t} + (U \cdot \nabla)U = -\frac{1}{\rho}\nabla p + v\nabla^2 U \quad ,$$

$$\frac{\partial T}{\partial t} + (U \cdot \nabla)T = \frac{k}{\rho c_p} \nabla^2 T \quad ,$$

$$\frac{\partial C}{\partial t} + (U \cdot \nabla)C = D \nabla^2 C$$

where U is the velocity vector, t is the time, p is the pressure, ν is the kinematic viscosity, T is the temperature, C is the humidity, c_p is the specific heat of the medium at constant pressure, k is the coefficient of thermal conductivity, ρ is the density, D is the coefficient of molecular diffusion.

The study area was identical to the second test problem in Figure 36. It is believed that the walls of the nasal cavity are completely saturated with water vapor and, due to the moist mucous layer and the rich underlying vascular bed, the temperature values are close to body temperature. The temperature value on the stacks of the nose is taken equal to 37°C, the humidity on the walls is taken to be 100%. The environmental conditions were taken as in the work of Naftali et al. [77], the temperature of the inhaled air is 25°C and the relative humidity is 20%. Figure 43 shows two-dimensional and three-dimensional distributions of the longitudinal components of the flow velocity for various sections (sections 1-3) with the given conditions. From the results obtained, it can be seen that the global behavior of the air flow has not been changed, but, however, the maximum longitudinal velocity has increased to 3.47 m/s (Figure 43). The increase in the maximum longitudinal velocity was influenced by the conditions of heating and humidification on the walls of the nasal cavity. Figure 44 shows the process of heating the inhaled air for different sections. From the results in Figure 44, it can be seen that the inhaled air in section 3 heats up, and the air temperature is 34-37°C. The concentration of water vapor at 3 cross-section reaches a value of 0.66-0.99 (Figure 45). This structure of the nasal cavity increases the local rate of heat and moisture transfer by narrowing the nasal passages for air. These constrictions in the nasal cavities result in turbulence downstream [125].

To investigate the effect of respiration under various temperature and humidity environments, several simulations were carried out using the proposed model. To simulate heat and mass transfer in the nasal cavity for normal inhalation in extreme environments, three modes were chosen: at an ambient temperature of 40°C and humidity on the walls of the nasal cavity 90%, at an ambient temperature of 5°C and humidity on the walls of the nasal cavity 20%, at ambient temperature 5°C and humidity on the walls of the nasal cavity 90%. The air velocity in the nasal cavity was used the same in all cases, however the transfer process differs depending on the conditions. Figures 46-48 show the simulation results at an ambient temperature of 5°C and a humidity on the walls of the nasal cavity of 20%. As shown in Figure 46, at a temperature of 5°C and a humidity of 20%, the maximum speed reached 3.56 m/s. The temperature of the inhaled air per section 3 ranges from 28-37°C. However,

it should be noted that a temperature of 28°C occurs only in a small area, the average temperature is 35°C. As shown in Figure 48, the inhaled air with a humidity of 20% is humidified with water vapor from 0.62-0.99 until it reaches the nasopharynx. In general, the distributions of water vapor are the same with the results from Figure 45 [125].

Figures 49-51 show the results of numerical simulations at an ambient temperature of 5°C inhaled air with 90% humidity on the walls of the nasal cavity. The flow behavior at 5°C and 90% humidity is the same as in Figure 46. The results in Figures 50 and 47 are the same since the conditions for the temperature of the inhaled air are the same. From the results of Figure 51 it can be seen that the concentration of inhaled air per section 3 is in the range of 0.78-1.0. Figures 52-54 show the results of numerical simulations at an ambient temperature of 40°C inhaled air with a nasal humidity of 90%. At an inspired air temperature of 40°C and a humidity of 90%, the maximum flow rate reaches 3.4 m / s. The nasal cavity not only heats the inhaled air, but in some critical cases it can also cool it. As shown in Figure 53, inhaled air at 40°C is cooled to body temperature. The concentration of inhaled air passing through cross section 3 is 0.79-1.0, which is a near alveolar condition [125].

From the above results, it can be concluded that the nasal cavity balances the inhaled air with the internal conditions of the body with remarkable efficiency, and is practically independent of the state of the surrounding air. As a comparison, the effect of external conditions in Figures 55 and 56 shows the profiles of temperature and concentration on lines 1-4 from section 1-2 (Figure 38) [125].

The results in Figure 55 show the difference in temperature propagation on measurement lines 1-4. In all the results, it can be seen that the temperature of the inhaled air tends to the conditions of the walls of the nasal cavity, that is, to a body temperature of 37°C. As shown in Figure 56, the concentration profiles on the measuring lines are the same under the same conditions of humidity of the inhaled air. Like the temperature values, the concentration of water vapor also tends to the conditions of the walls of the nasal cavity, that is, to a concentration equal to 1.0 [125].

In this work, the realistic geometry of the human nasal cavity was used, but, however, it should be borne in mind that this geometry is not universal, since each person has its own structure of the turbinate. However, the created three-dimensional geometry of the human nasal cavity can be of immense value as a standard nasal replica for testing various conditions of ambient temperature, relative humidity and inhaled air flow rate [125].

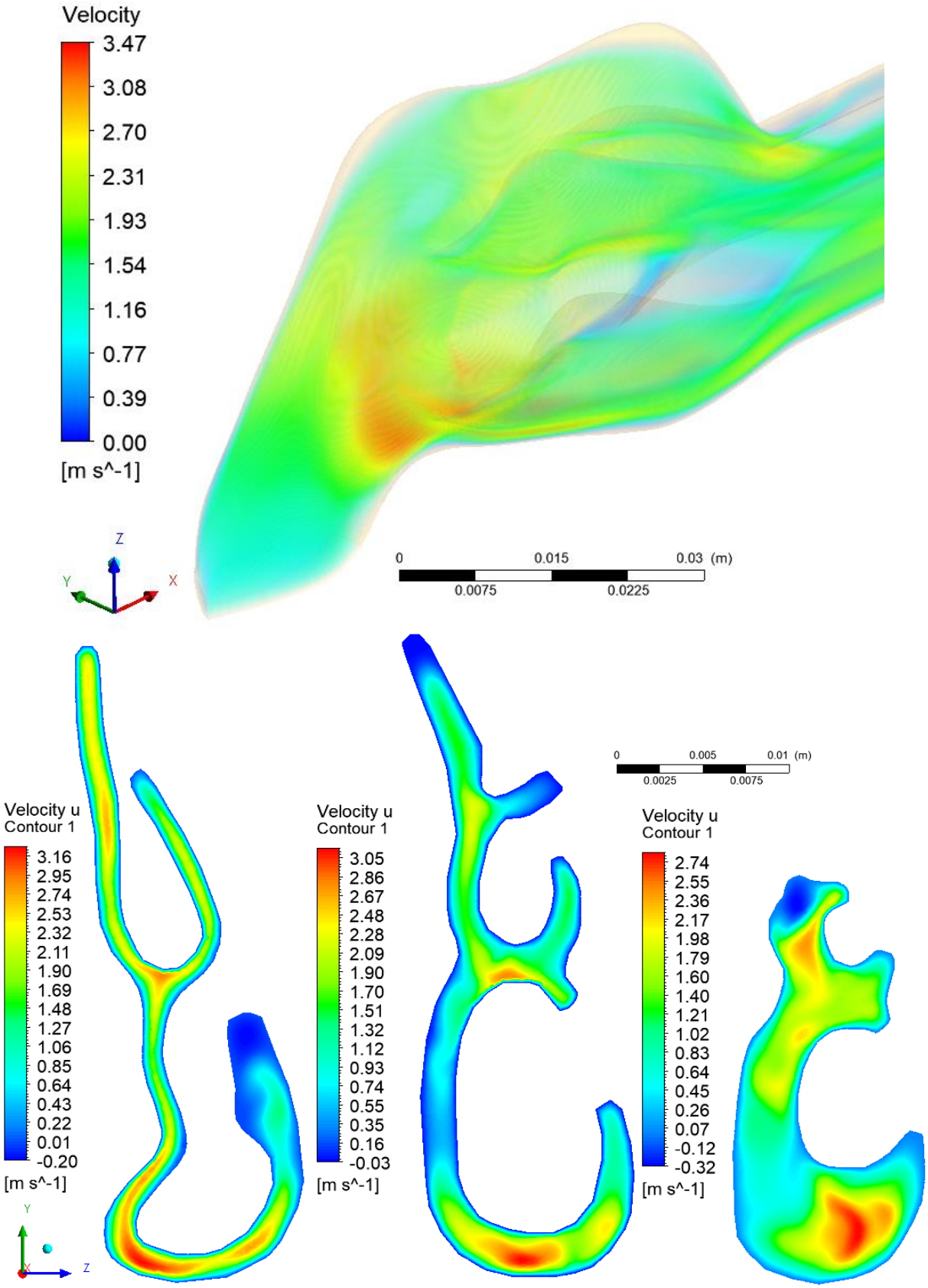


Figure 43 - Two-dimensional and three-dimensional distribution of longitudinal components of the flow velocity at an ambient temperature of 25 ° C and humidity on the walls of the nasal cavity 20% for various sections (sections 1-3).

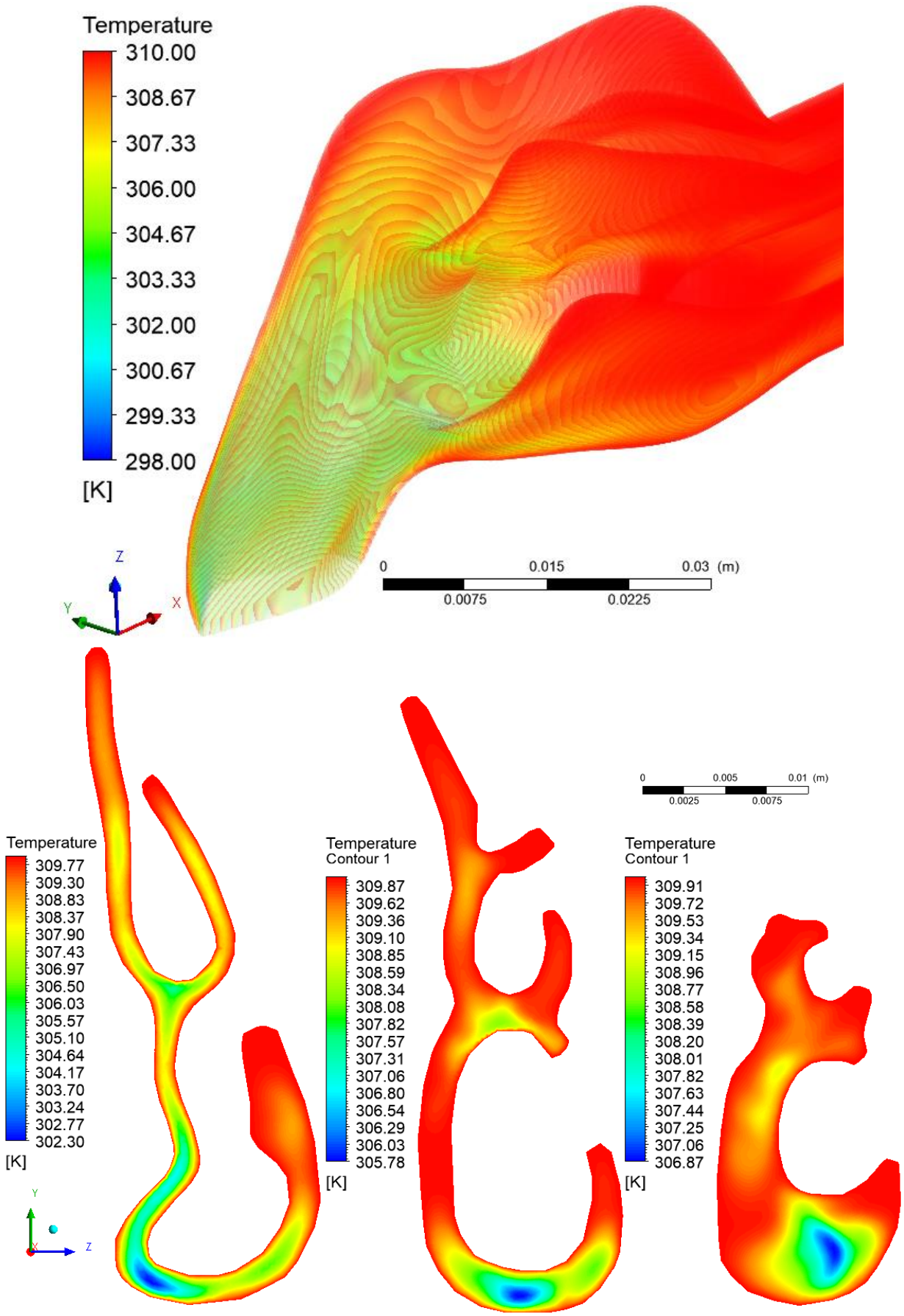


Figure 44 - Two-dimensional and three-dimensional heat distribution at an ambient temperature of 25 ° C and humidity on the walls of the nasal cavity 20% for various sections (sections 1-3).

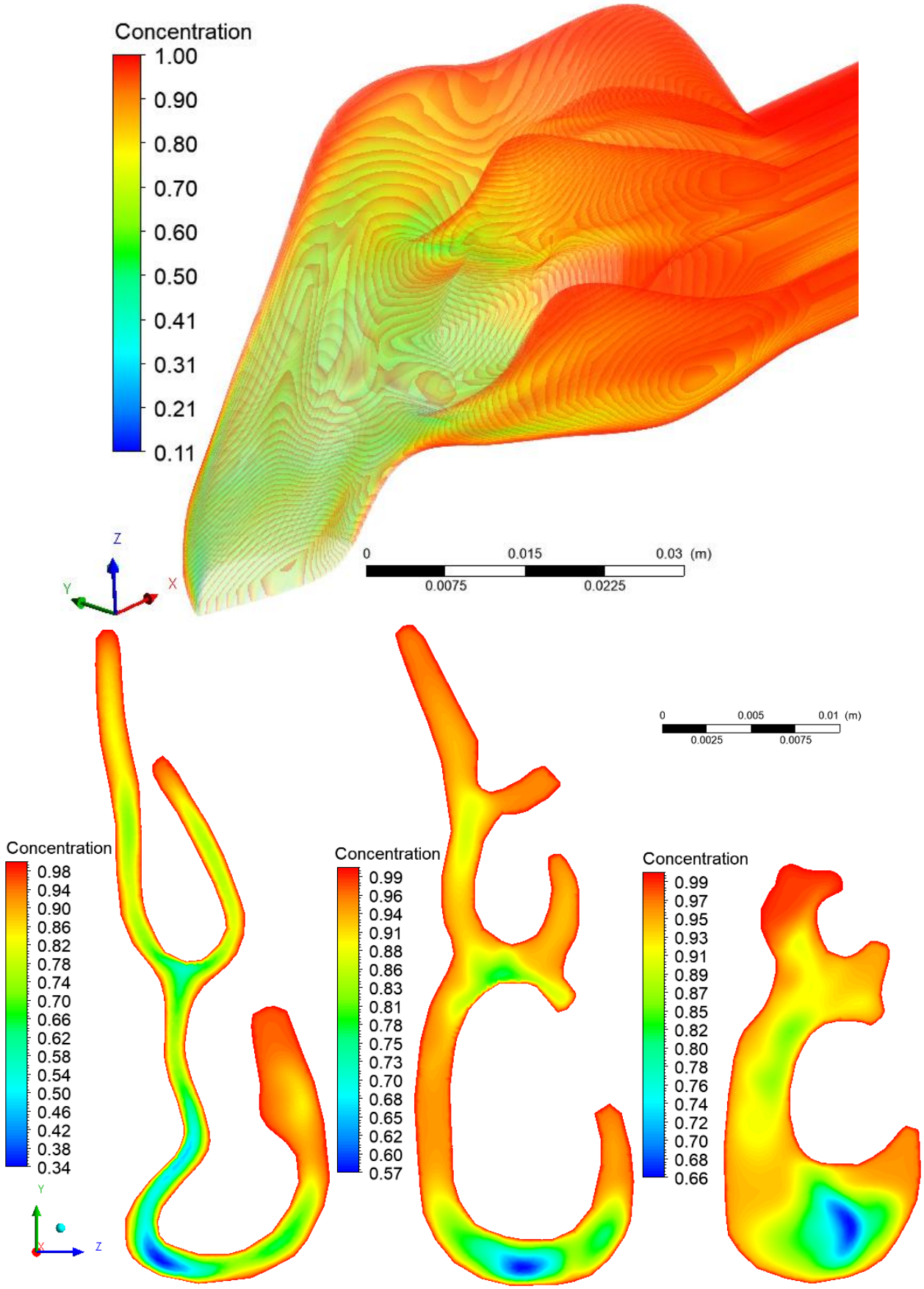


Figure 45 - Two-dimensional and three-dimensional distributions of concentration at an ambient temperature of 25°C and humidity on the walls of the nasal cavity 20% for various sections (sections 1-3).

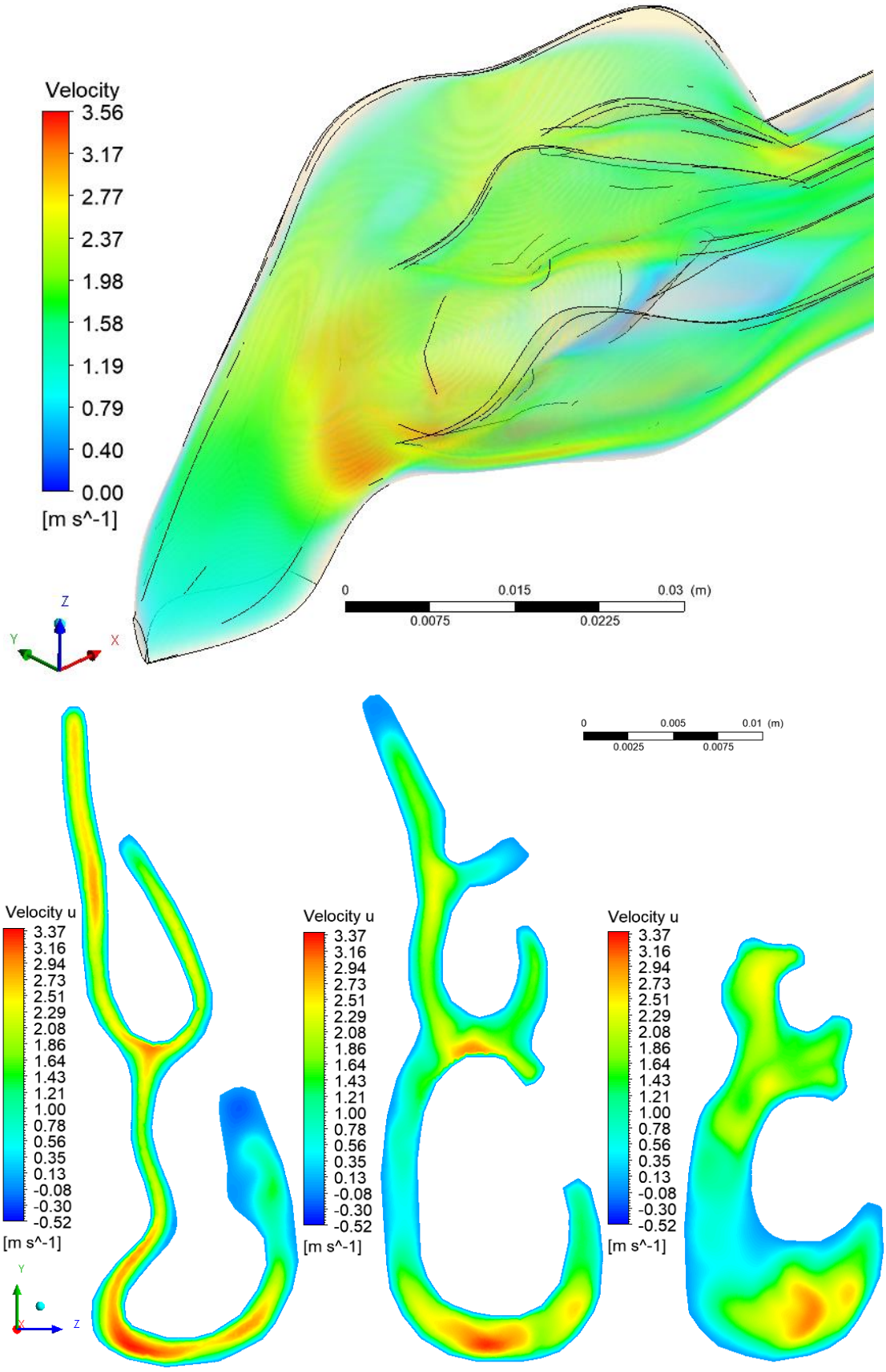


Figure 46 - Two-dimensional and three-dimensional distribution of longitudinal components of the flow velocity at an ambient temperature of 5 ° C and humidity on the walls of the nasal cavity 20% for different sections (sections 1-3).

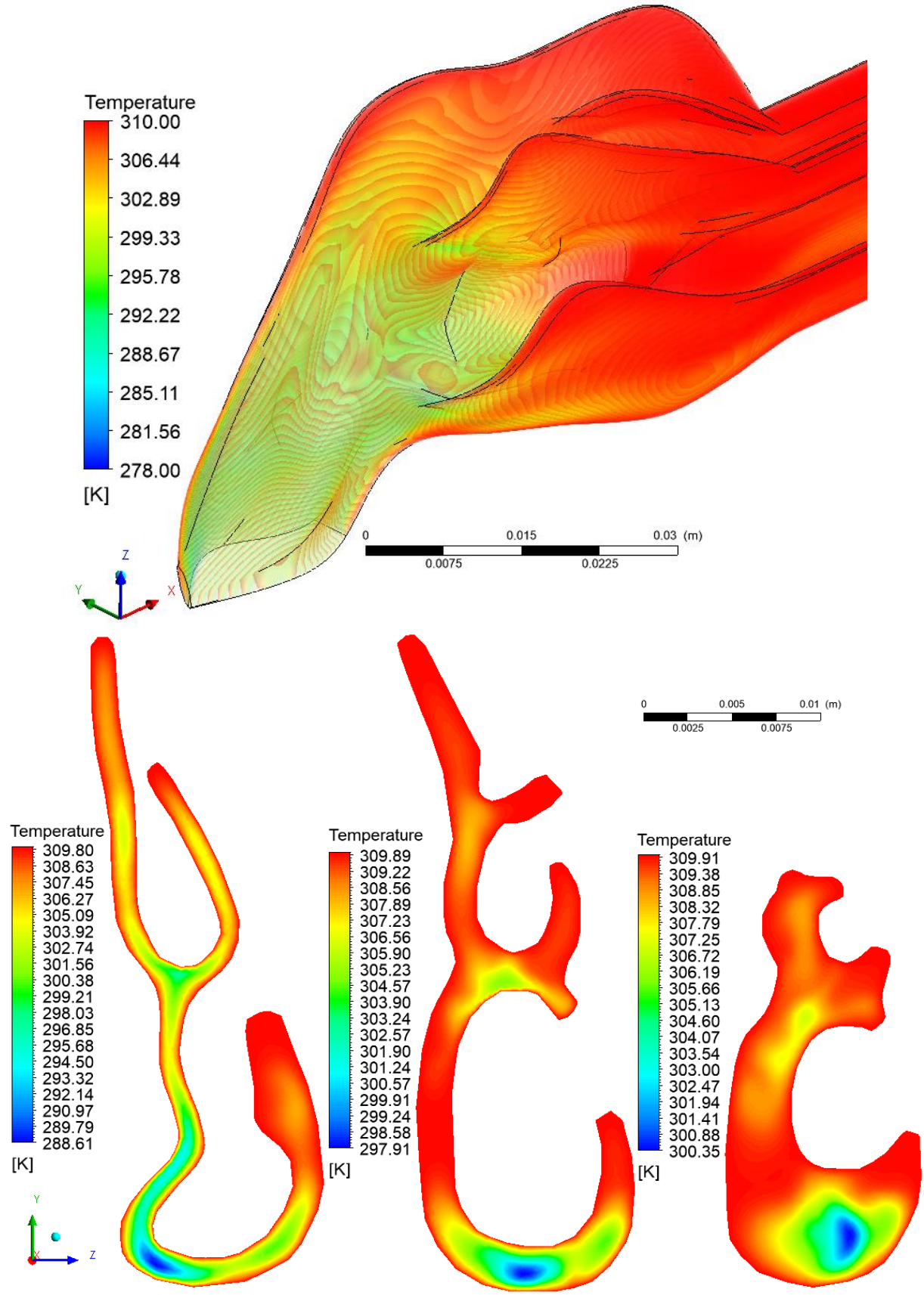


Figure 47 - Two-dimensional and three-dimensional temperature distributions at an ambient temperature of 5 ° C and humidity on the walls of the nasal cavity 20% for various sections (sections 1-3).

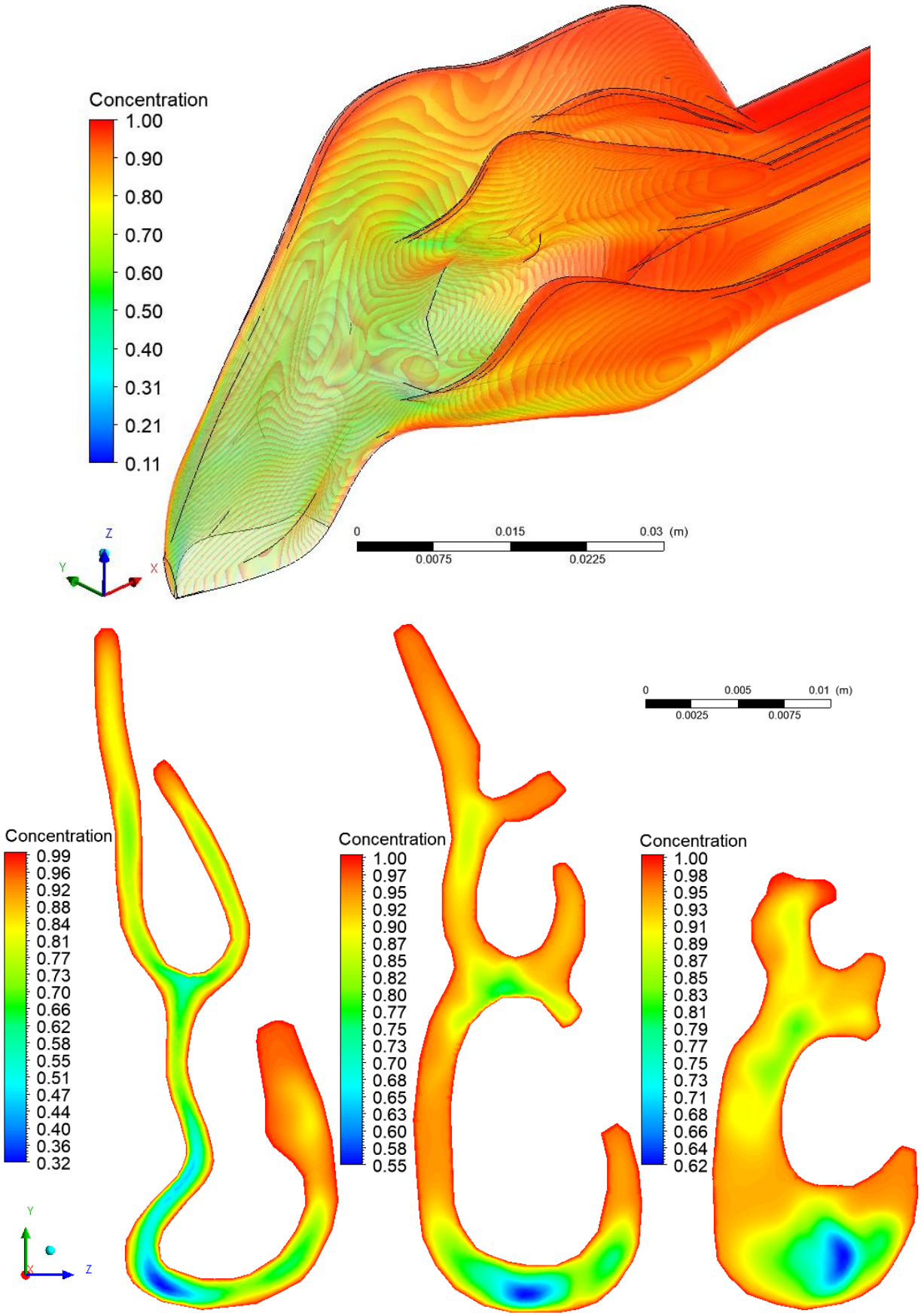


Figure 48 - Two-dimensional and three-dimensional concentration distributions at an ambient temperature of 5 ° C and humidity on the walls of the nasal cavity 20% for various sections (sections 1-3).

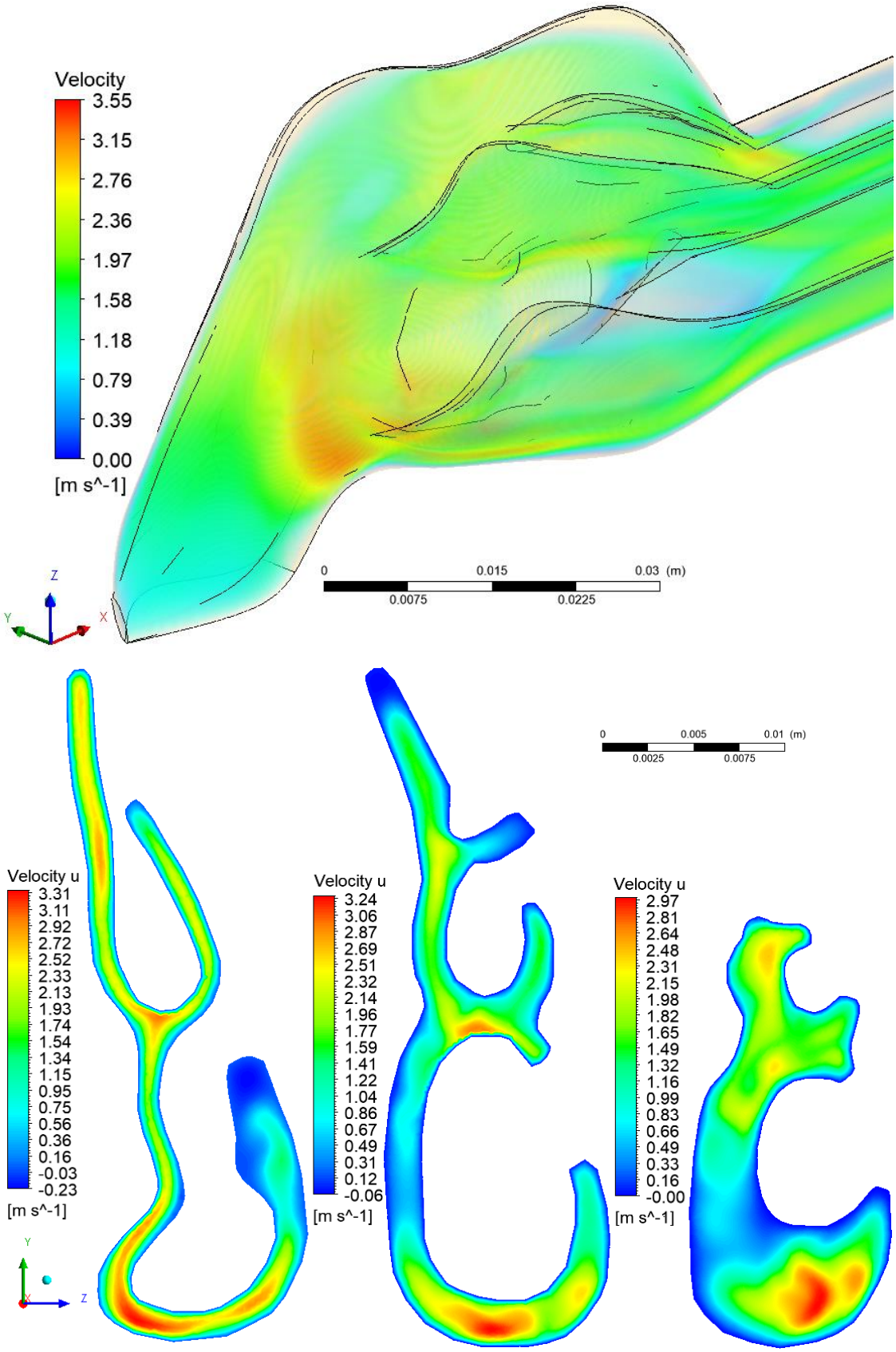


Figure 49 - Two-dimensional and three-dimensional distributions of longitudinal components of the flow velocity at an ambient temperature of 5 ° C and humidity on the walls of the nasal cavity 90% for various sections (sections 1-3).

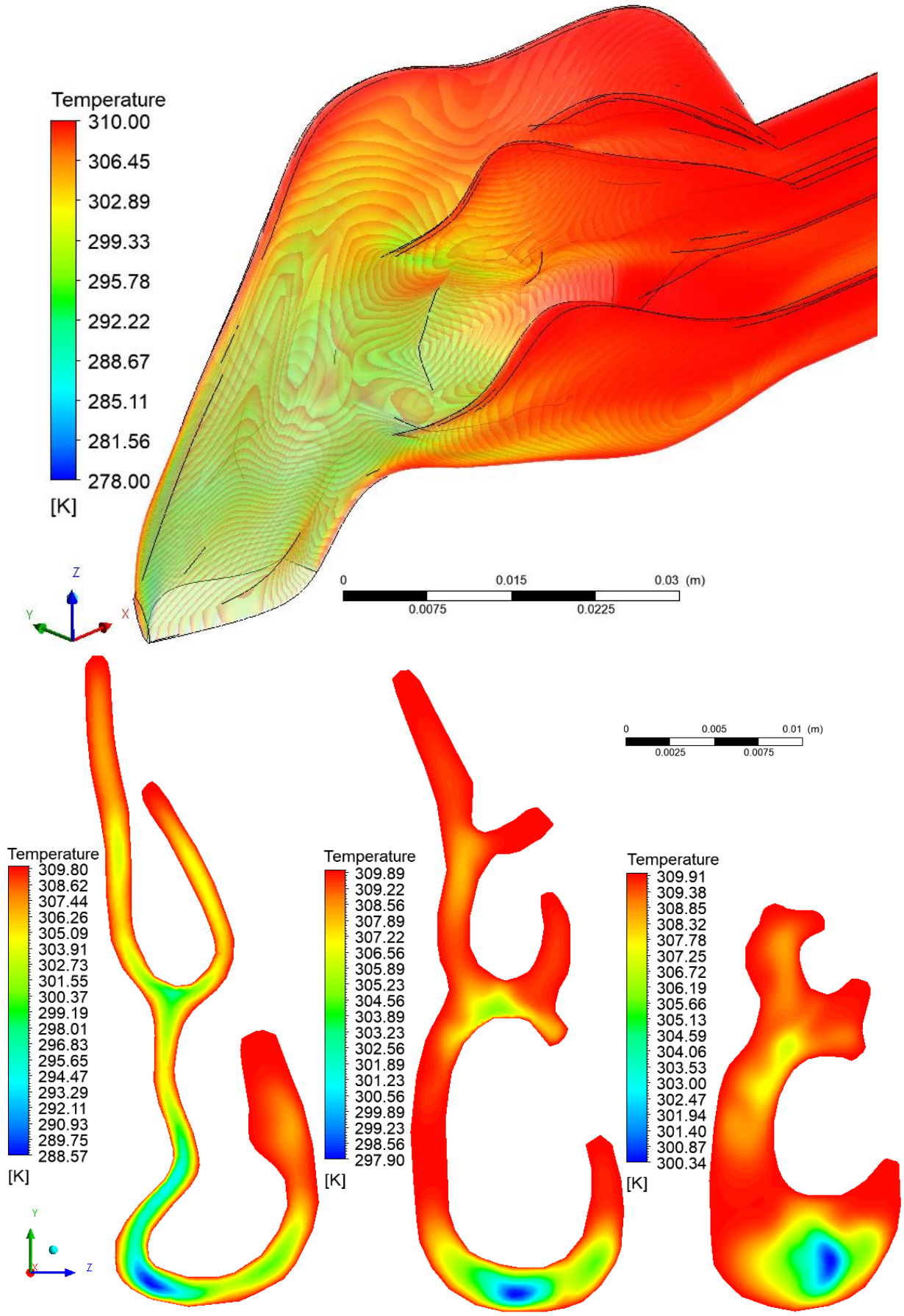


Figure 50 - Two-dimensional and three-dimensional temperature distributions at an ambient temperature of 5°C and humidity on the walls of the nasal cavity 90% for various sections (sections 1-3).

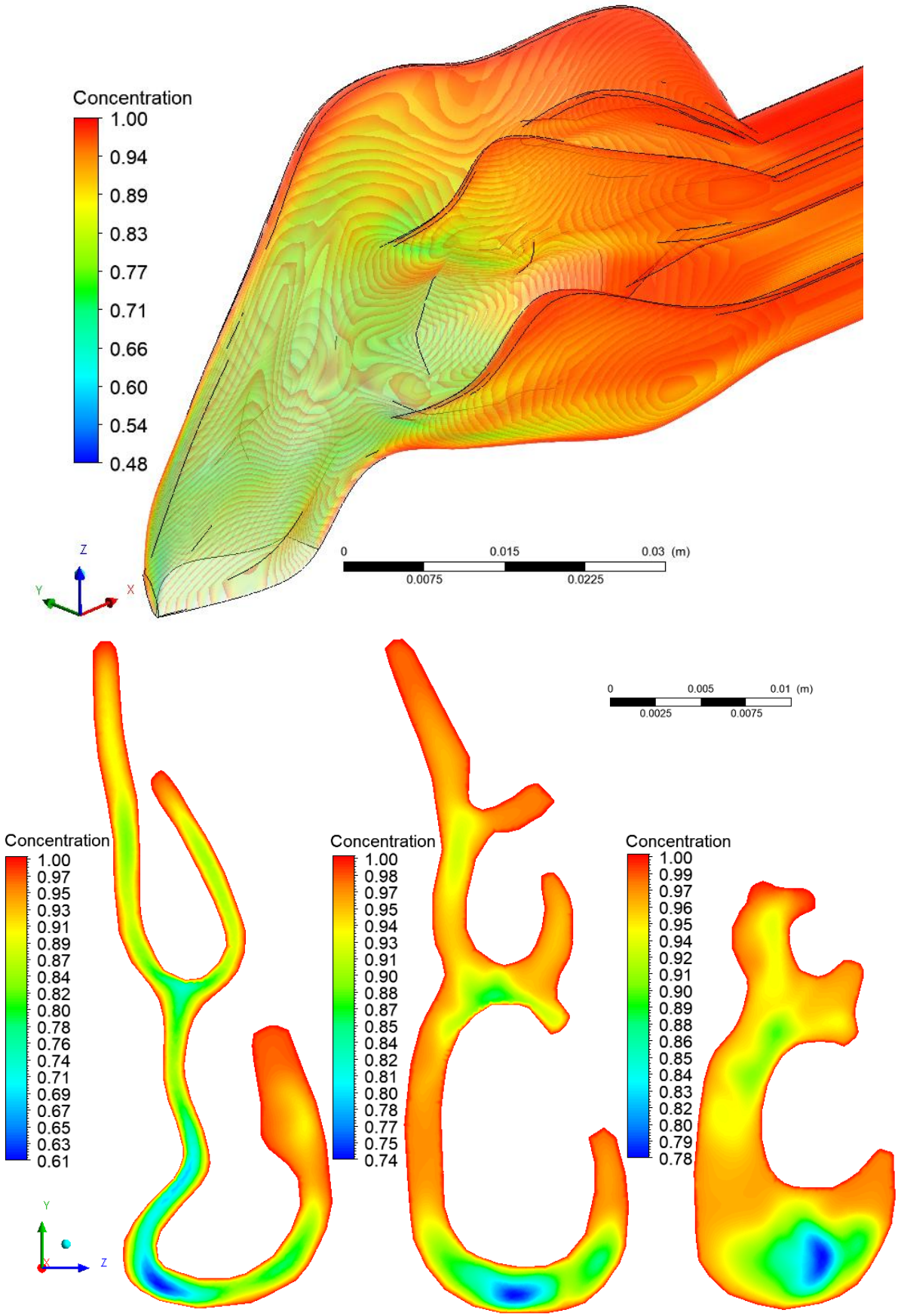


Figure 51 - Two-dimensional and three-dimensional concentration distributions at an ambient temperature of 5 ° C and humidity on the walls of the nasal cavity 90% for various sections (sections 1-3).

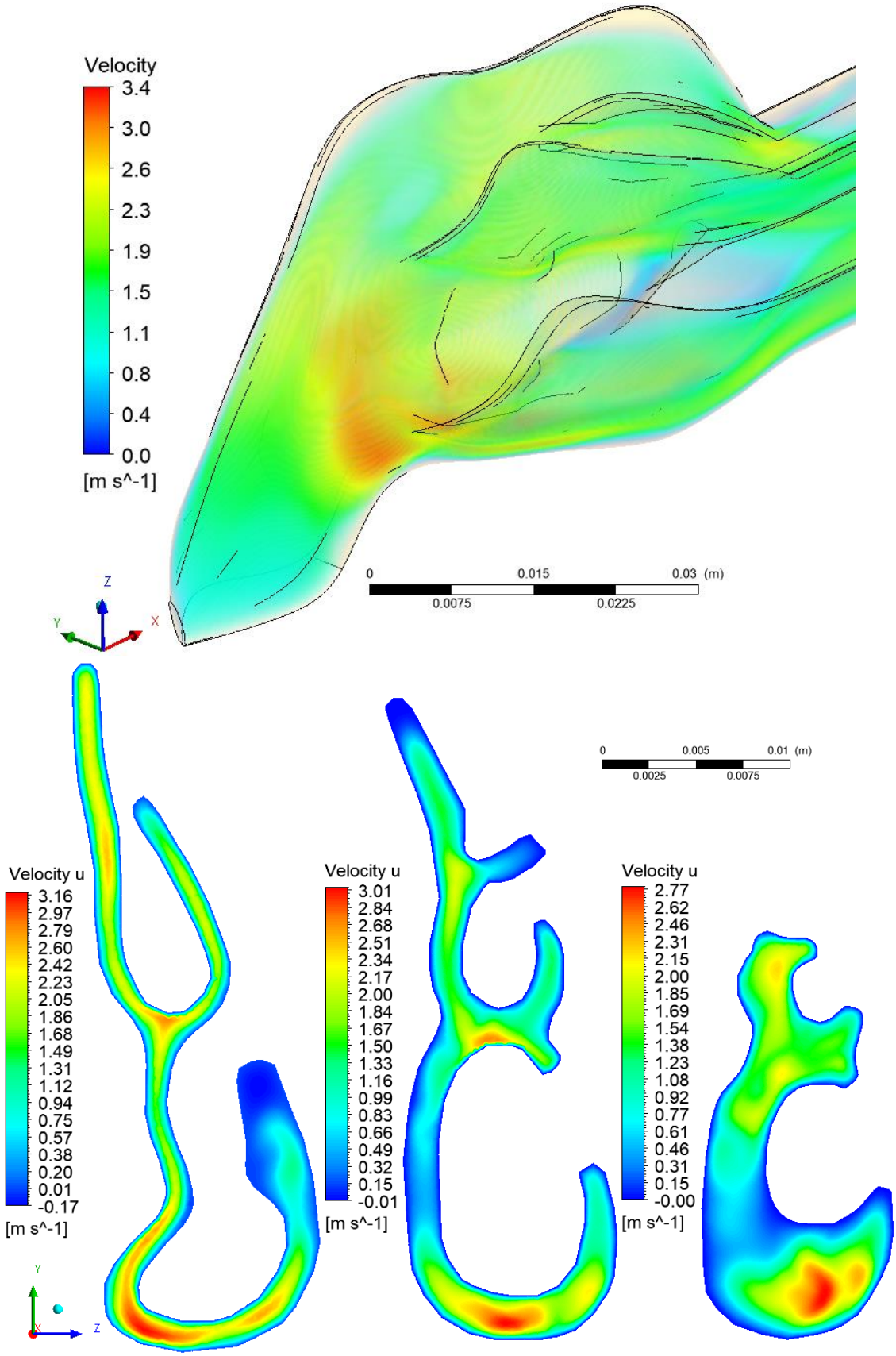


Figure 52 - Two-dimensional and three-dimensional distributions of longitudinal components of the flow velocity at an ambient temperature of 40 ° C and humidity on the walls of the nasal cavity 90% for different sections (sections 1-3).

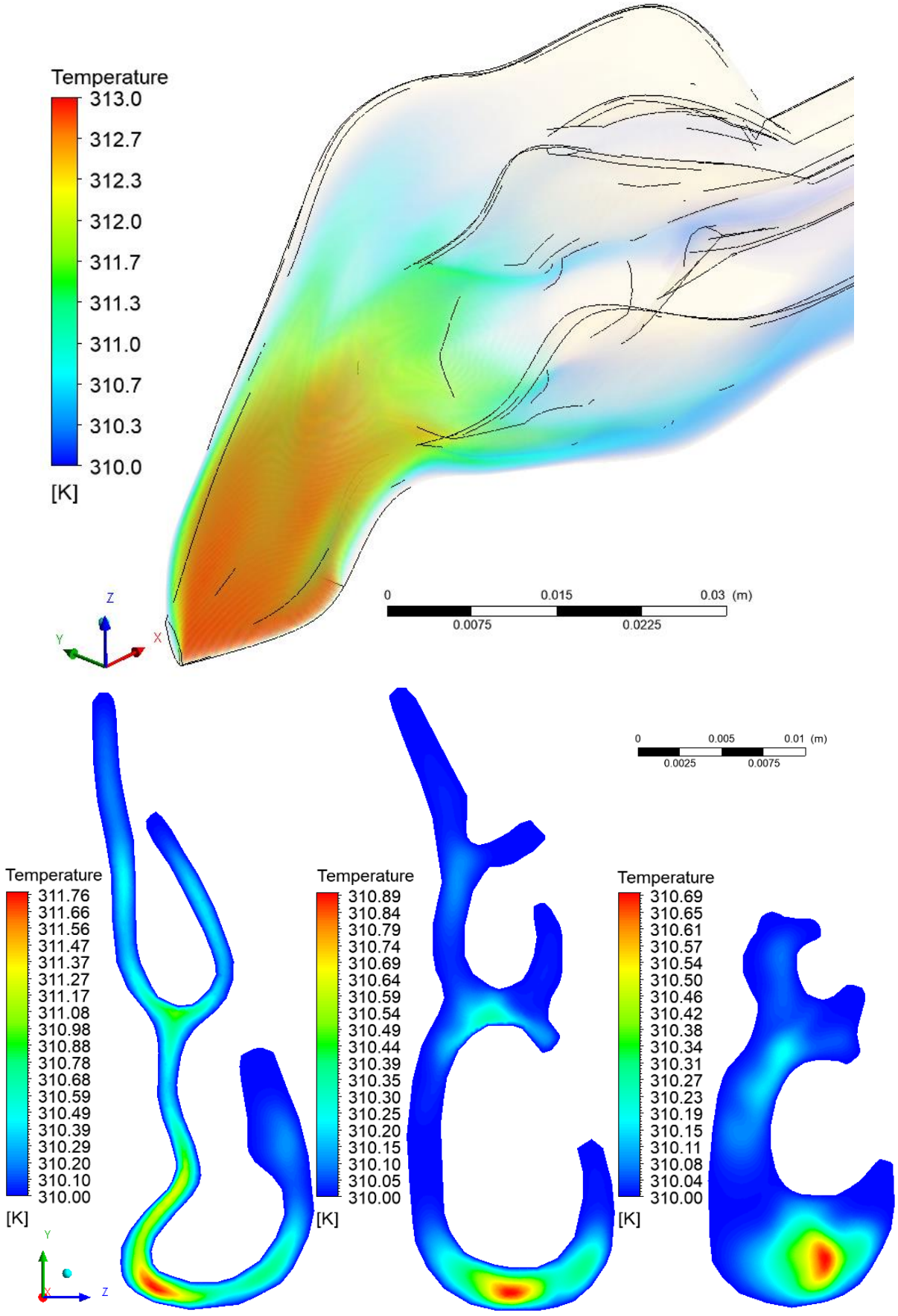


Figure 53 - Two-dimensional and three-dimensional temperature distributions at an ambient temperature of 40 ° C and humidity on the walls of the nasal cavity 90% for various sections (sections 1-3).

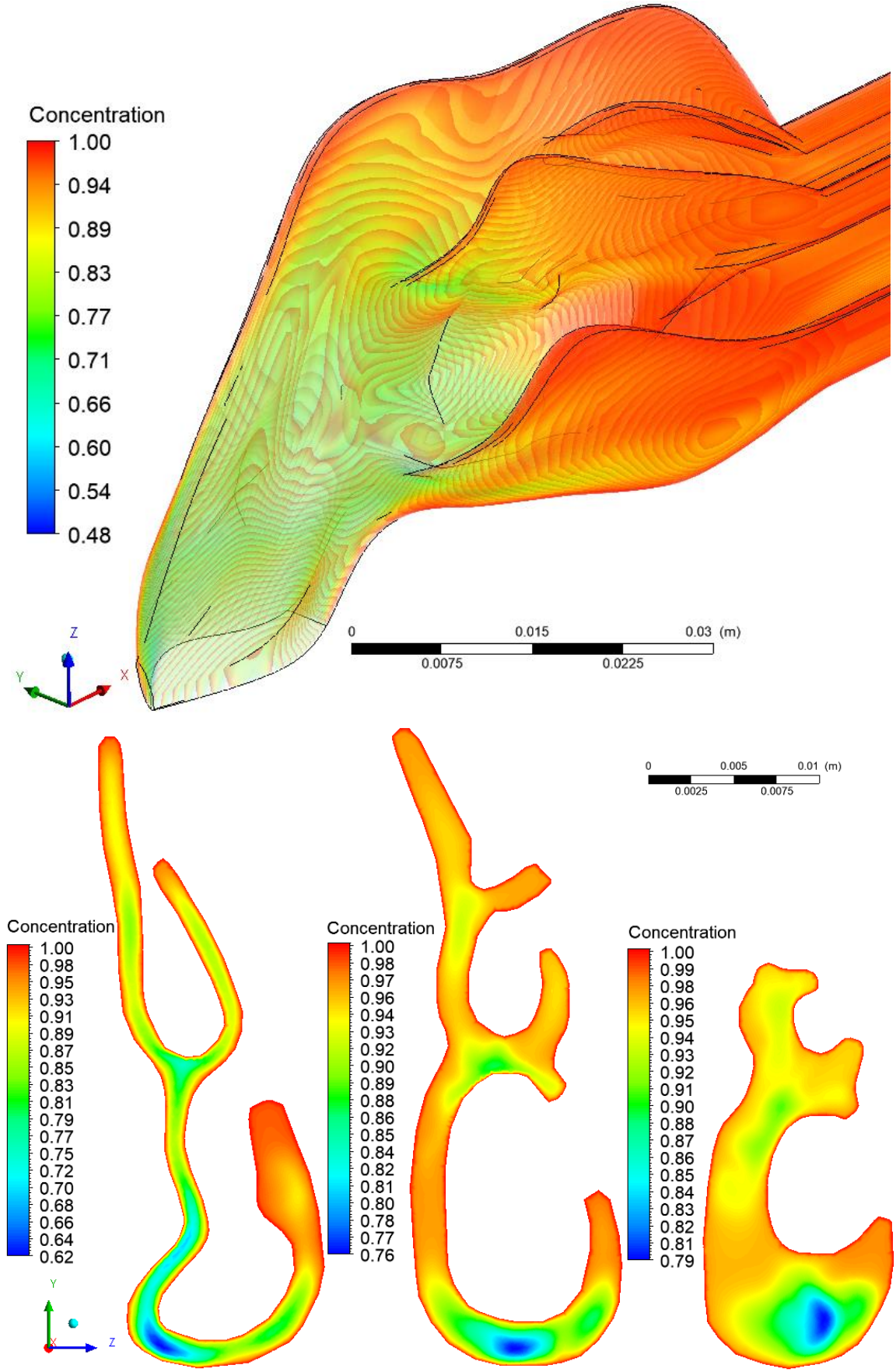
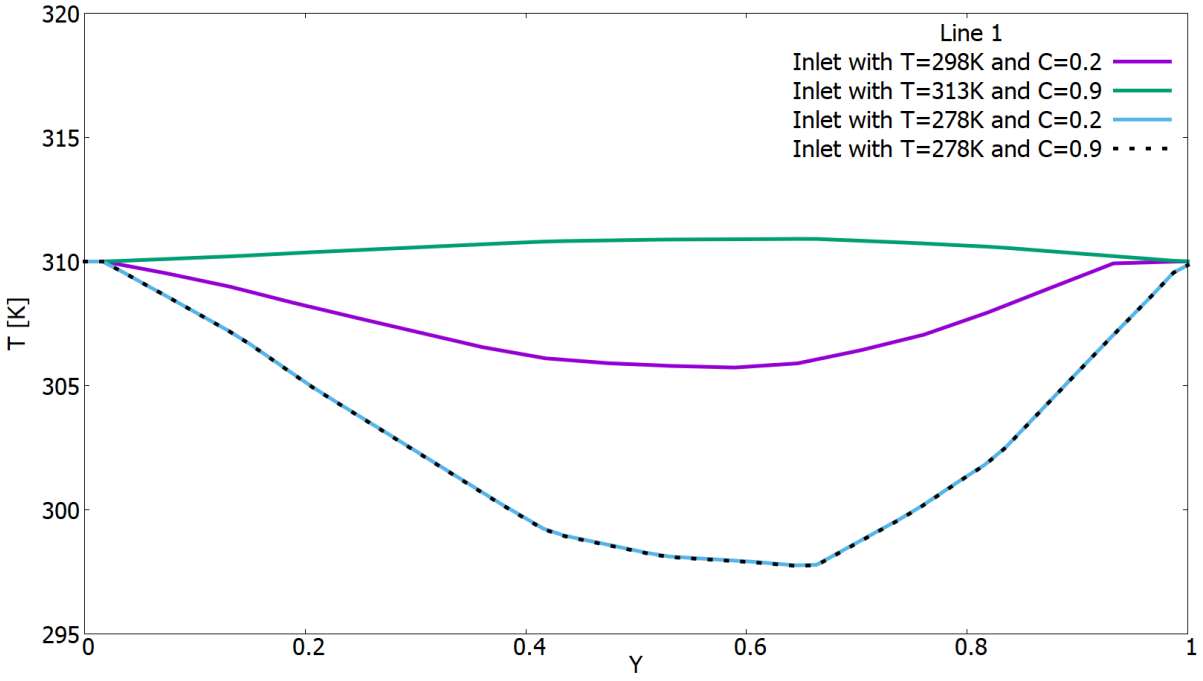
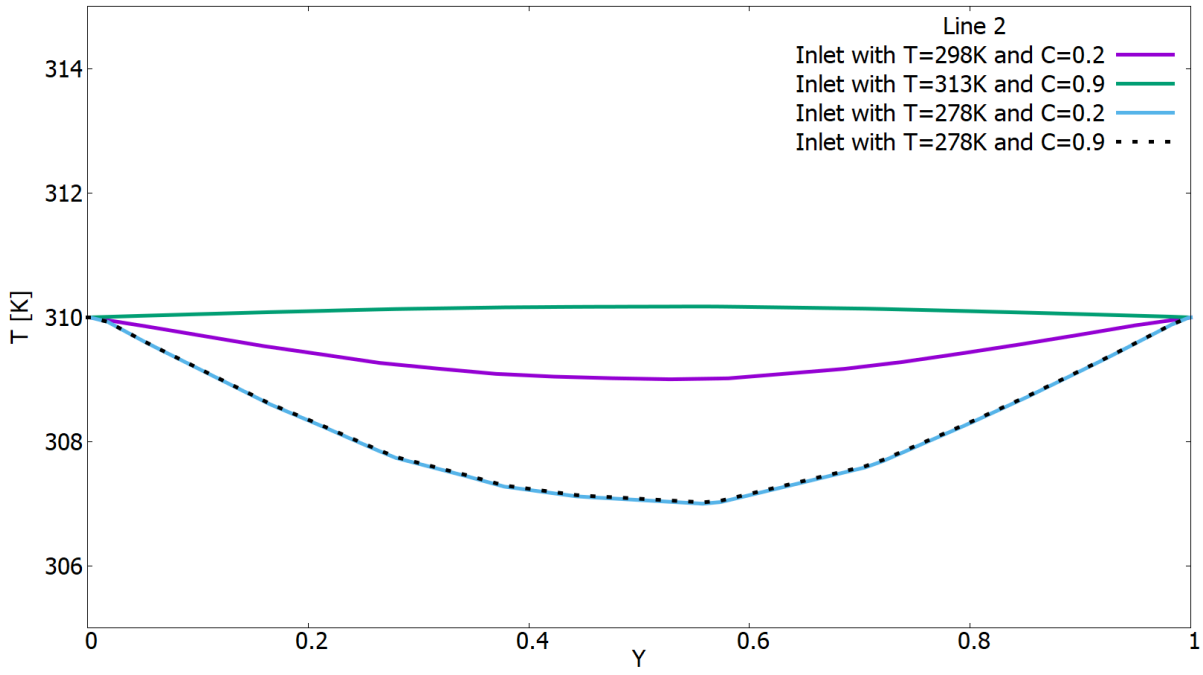


Figure 54 - Two-dimensional and three-dimensional concentration distributions at an ambient temperature of 40 ° C and humidity on the walls of the nasal cavity 90% for various sections (sections 1-3).


Line 1
Inlet with T=298K and C=0.2
Inlet with T=313K and C=0.9
Inlet with T=278K and C=0.2
Inlet with T=278K and C=0.9
T [K]
Y
320
315
310
305
300
295
0
0.2
0.4
0.6
0.8
1


Line 2
Inlet with T=298K and C=0.2
Inlet with T=313K and C=0.9
Inlet with T=278K and C=0.2
Inlet with T=278K and C=0.9
T [K]
Y
314
312
310
308
306
0
0.2
0.4
0.6
0.8
1

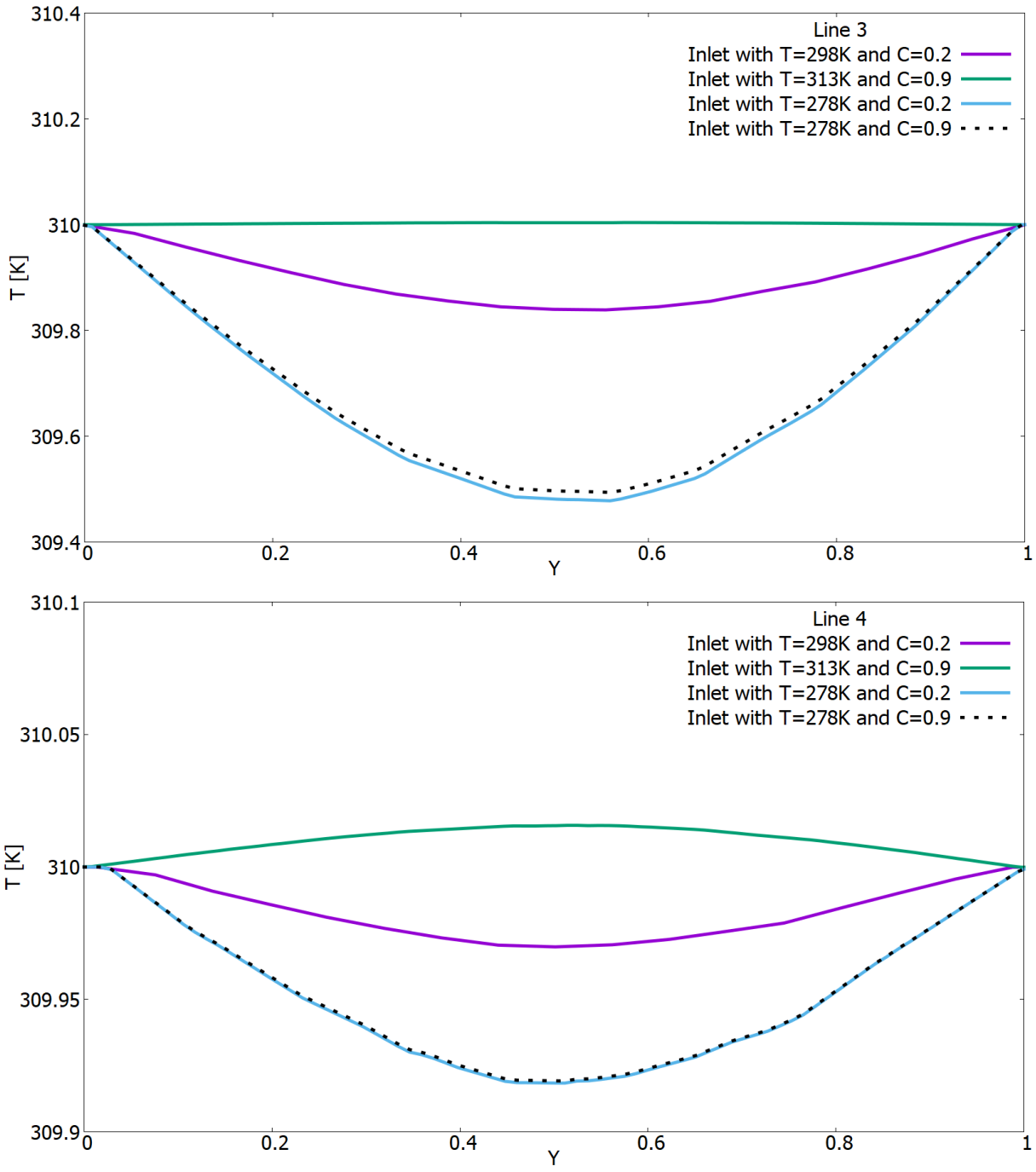
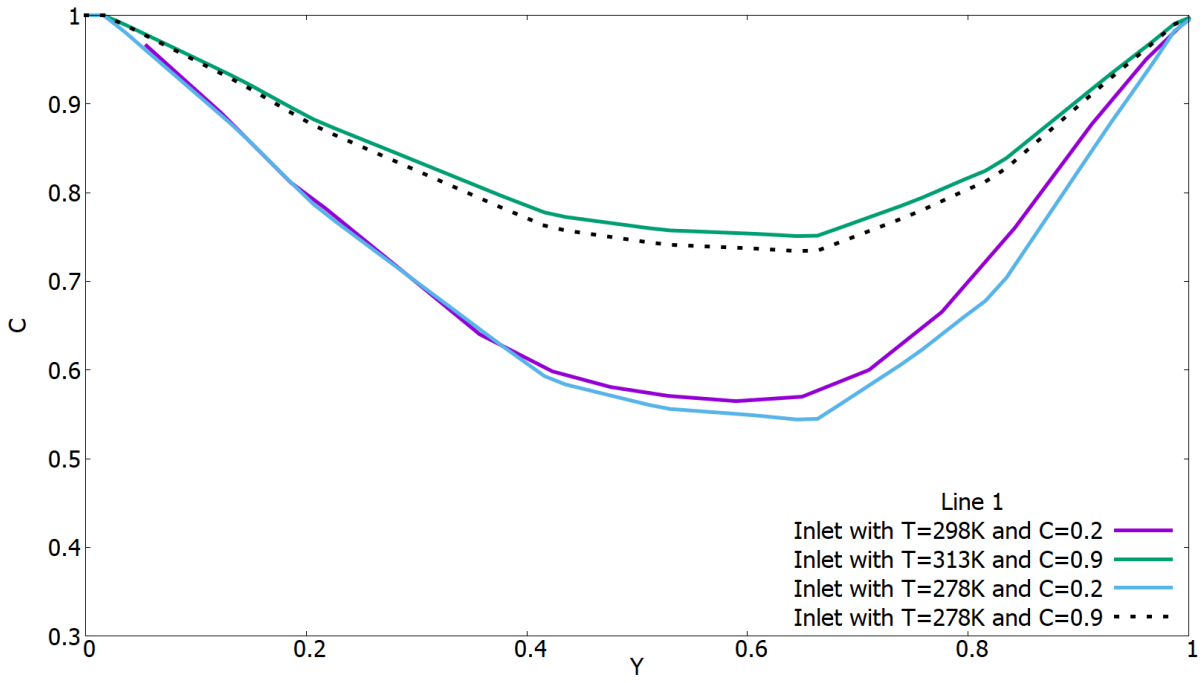
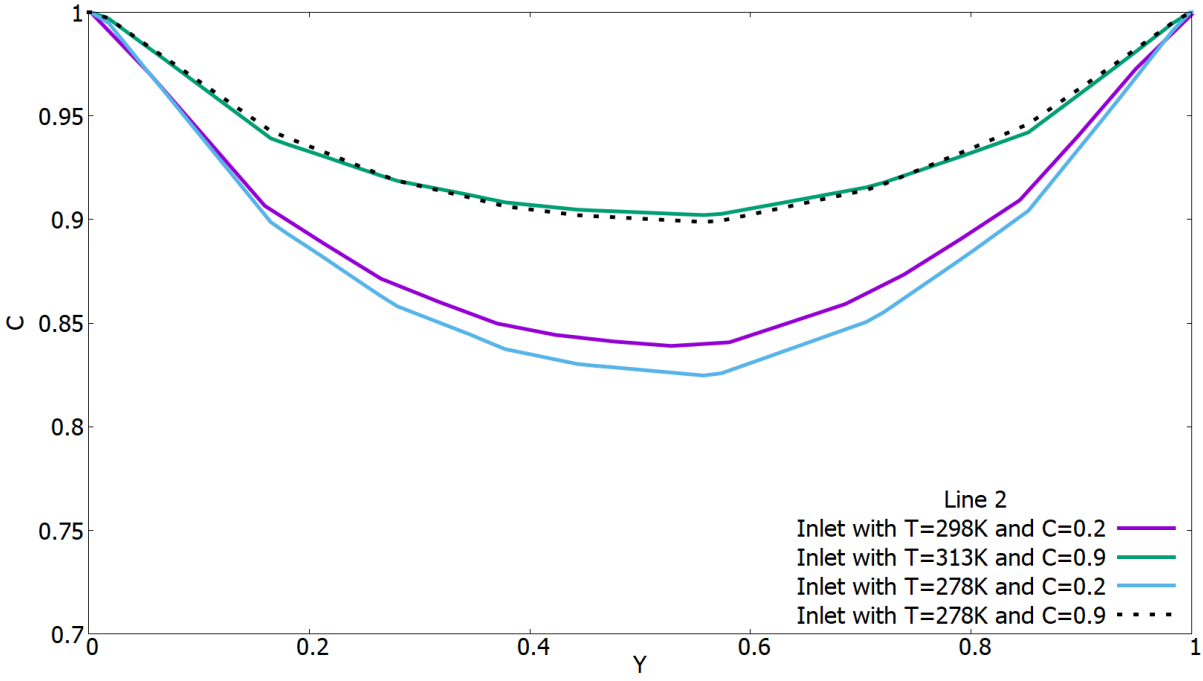


Figure 55 - Comparison of temperature profiles on lines 1-4 under different environmental conditions.


C
Y
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
0.2
0.4
0.6
0.8
1
Line 1
Inlet with T=298K and C=0.2
Inlet with T=313K and C=0.9
Inlet with T=278K and C=0.2
Inlet with T=278K and C=0.9


C
Y
1
0.95
0.9
0.85
0.8
0.75
0.7
0
0.2
0.4
0.6
0.8
1
Line 2
Inlet with T=298K and C=0.2
Inlet with T=313K and C=0.9
Inlet with T=278K and C=0.2
Inlet with T=278K and C=0.9

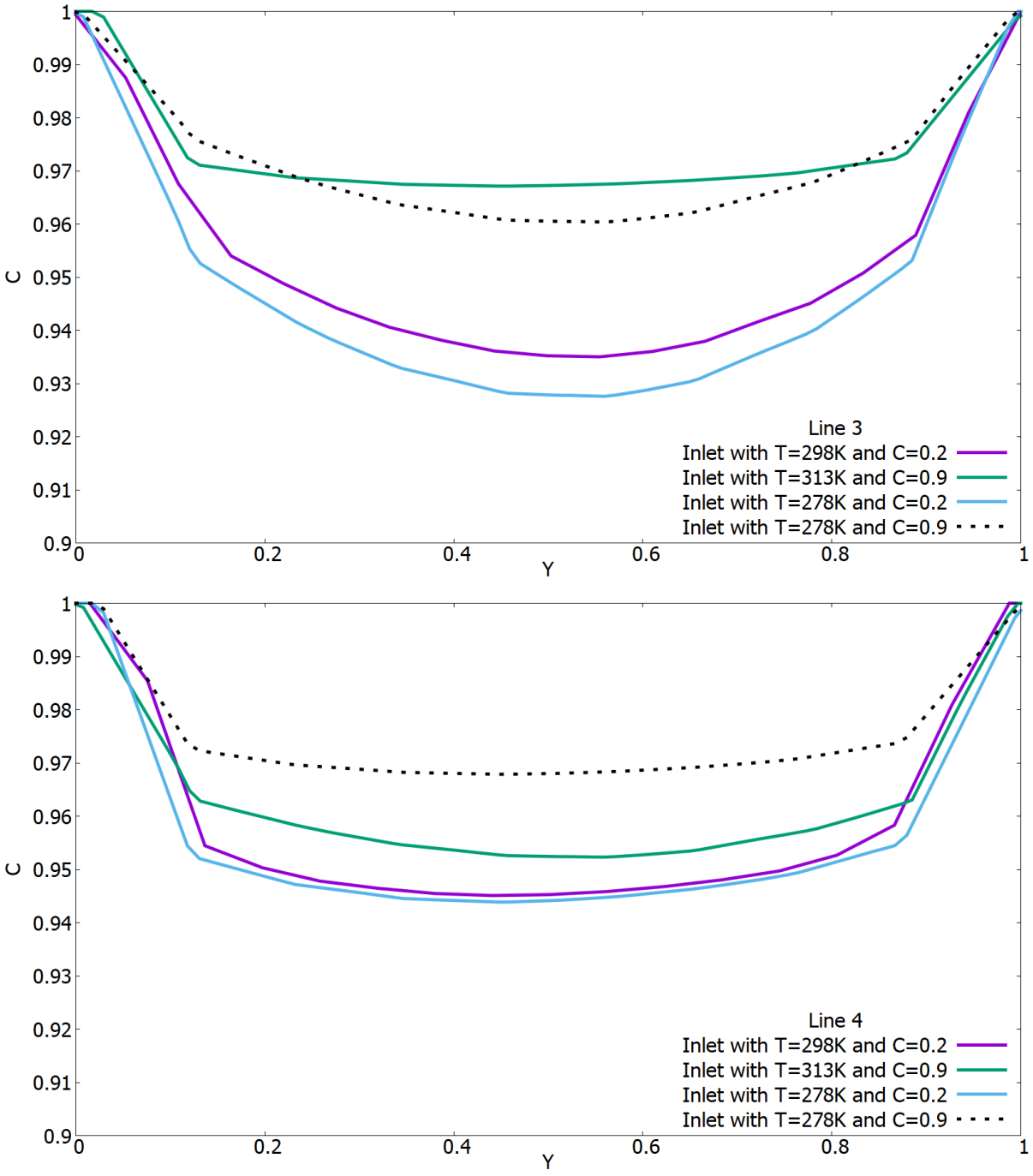


Figure 56 - Comparison of concentration profiles on lines 1-4 under different environmental conditions.

5.7 Hybrid parallel numerical algorithm using dynamic load balancing

The development of technology and the increase in the volume of data in almost all areas have given impetus to a new problem. Thus, increasing efficiency and processing data in real time has become a key aspect for specialists. The first thing that comes to mind is parallel programming. As you know, when a large number of processors work simultaneously, this is called a parallel architecture. Moreover, if the processors have different characteristics, such a system is called heterogeneous (otherwise homogeneous). Variable memory sizes and data rates make it difficult to

distribute work efficiently. While with the same capabilities, this task is more feasible.

Long-term use of parallel algorithms to date, a large selection of implementations. But the key point is the correct distribution of the load on the processors (balancing). Obviously, improper load balancing negatively affects performance, thereby increasing execution time.

It follows from the above that with the help of proper load distribution, significant results can be achieved in reducing the processing time for a single processor. There are two types of balancing, static and dynamic. In this situation, it is more appropriate to use dynamic load balancing, taking into account the different capabilities of each processor.

It is possible to consider two types of balancing application: manual and automated. One disadvantage of manually applying balancing is a big waste of time. Whereas the big advantage of automated balancing is its simple and fast implementation to an already existing parallel algorithm.

As previously mentioned, there are two types of balancing. Static balancing dates back to 1972 (R. L. Graham.) and its main principle is the distribution of balancing for tasks immediately before starting work. Sometimes the lack of accurate information about the characteristics of processors can be a disadvantage, as it requires comparing them with each other. As you know, compilers play an important role in a parallel algorithm. And here you can notice another equally important drawback - the need to match the task with the processors. This can be circumvented by knowing the exact characteristics of the system and by providing independent tasks. The situation is aggravated by the fact that with parallel execution it is not possible to predict the course of communication, nor the execution itself, and the lack of data about the task being performed. Thus, the clear superiority of using dynamic load balancing can be emphasized. In dynamic data allocation, assigning tasks directly on execution avoids the above problems.

5.8 Parallelization algorithm using dynamic load balancing

Using the full power of the processor due to the correct distribution of the load leads to increased efficiency, which is the main goal of dynamic load balancing [96, 97]. It is well known that even a minimal failure in load distribution can have a dramatic effect on the overall picture. Which again emphasizes the importance of proper balancing [105]. Assuming that the delay due to imbalance occurs at each iteration, one can imagine what the magnitude of the impact on efficiency due to synchronization will be. Since the MPI subdomains very often communicate with each other [106]. It is also known that delaying even one processor greatly affects the overall time and efficiency. Since everyone else is waiting for the belated processor to finish executing. At the same time, underutilized processors do not contribute enough to increase efficiency, even if the process is executed ahead of schedule. Different types of imbalances occur under different circumstances. For example, as the workload changes over time, dynamic imbalances occur. Whereas static load

distribution leads to both constant and equal imbalance [106].

Different parameters can be used in determining the imbalance. For example, the percentage of imbalance [107] can be expressed by the formula:

$$I_{\%} = \frac{t_{\max} - t_{avg}}{t_{\max}} \frac{N}{N-1} \quad (21)$$

t_{max} - the maximum and t_{avg} - average time, N - the number of parallel processes. It should be noted here that a result of 0% corresponds to the absence of imbalance. Whereas 100% result is the same if the parallel program was executed by one processor. As a result, it can be said that this formula is an indicator of the percentage of misuse of resources. As you know, in numerical simulation, the work is divided into steps. So here is the idle time of all processors waiting for the completion of the slowest one and can be seen as the result of this formula at one-time step. If there is the possibility of achieving an ideal balance, then the savings in working time can be calculated according to the It=tmax-tavg formula [107]. In addition, IT = N It can be used to express the effect of distribution time, which is the maximum indicator of the amount of resources spent [106].

Having many DLB schemes in science, one can say that the choice depends on several factors: the type of grid, its partitioning, and the numerical method. In short, it is necessary to first determine the amount of load to be transferred, then the processor, which is directly underloaded and can accept additional volume in this amount, and finally perform the transfer.

As a rule, this process is not so simple. For large-scale problems, DLB can be problematic (multi-stage computations and interleaved computations with inherent communication barriers) [108]. Even with static allocation, load imbalances can occur during computation, such as in the direct hybrid method. Estimating various computational costs for the initial domain decomposition is the main parameter for estimating the efficiency of computation in the absence of load balancing. Forgetting to take into account the waiting time and boundary conditions, one can be mistaken when predicting the expected load. As a rule, it is calculated by comparing the measured time from independent models. These parameters display and explain the difference between predicted and actual load [109].

Moreover, the performance situation can deteriorate when considering a heterogeneous system or a large-scale simulation with relatively small field workloads [110]. Thus, as a way out of this situation, one can imagine DLB, automatic distribution of balance in real time simulation. Moreover, the simulation environment can be extended using DLB and hybrid parallelization. Since there are many exceptions when using the direct hybrid method, such as changing the size of the grid when an unassigned task changes, redistributing calculation cells. By calculating new computational weights in all subdomains on the data obtained from the parallel process time count and cell allocation, DLB determines new domain divisions.

5.9 Estimating Computational Weights

Giving an estimate of the computational costs of various objects in the simulation, it is possible to achieve the correct distribution of the load and its balancing for related tasks [102]. The standard method is considered to be a uniform distribution of tasks in each parallel subdomain. For all that, it is assumed that the environment is homogeneous with one method and the computational cost is constant and unchanged for each object. That is, if deal with different computational costs. One way to avoid this problem is to define precomputed weights for each type of problem. Again, there may be exceptions in the form of a failure in decomposition due to the lack of some algorithmic parts or the heterogeneity of processors. One way to solve this situation is to know the real time for each object [110]. But given the likelihood of using different tools and significant efforts, the above method is not suitable for scientific applications. It can be said about the need for a minimally intrusive and reliable method of estimating computational weights to apply the correct load distribution.

A feature of the DLB algorithm in this work is the estimation of weights due to time measurements in each parallel subdomain. And under different types of load, different methods can be taken into account. The time measured for each step in the simulation adds up to the total computation time. To take into account external factors in each subdomain i, its average value ri was determined, truncated by 25% [106, 111]. With N parallel processes, the following is the global average computation time:

$$\bar{r} = \frac{1}{N}\sum_{i=0}^{N-1} r_i \tag{22}$$

where l_i - the local computational load and could be expressed as:

$$l_i = \frac{r_i}{\bar{r}} \tag{23}$$

Later, solving the problem using the least squares method $Ac - l$, you can calculate the calculated weights. As is known, in the least squares method where the right side is determined by the load vector l, while the left matrix A represents the current load distribution among all subdomains. From this it can be summarized that the average workload can be represented by a linear combination of the individual workload contributions.

$$\left\|Ac - l\right\|_2 = \min_v \left\|Av - l\right\|_2 \qquad \text{и} \ \left\|v\right\|_2 \tag{24}$$

The DGELSD procedure in LAPACK [112] given in formula (24) was used in this work. Moreover, formula (25) gives an example of this procedure with two types of

load for N=4 parallel subdomains. A difference of 2.61 was obtained by comparing the two types of loads under an overdetermined system of linear equations using the least squares method. From which it follows that the average calculation time for the first type of objects is 2.6 times lower than for the second type.

$$N\left\{\begin{bmatrix}10 & 7\\ 13 & 4\\ 12 & 2\\ 5 & 8\end{bmatrix}\begin{bmatrix}c_0\\ c_1\end{bmatrix}=\begin{bmatrix}1.2\\ 0.9\\ 0.8\\ 1.1\end{bmatrix}\right. \quad \Leftrightarrow Ac=l \quad \underset{least\ squares\ solution}{\rightarrow} \quad c=\begin{bmatrix}0.0420\\ 0.1097\end{bmatrix} \tag{25}$$

$$\frac{c_1}{c_0}=2.61$$

5.10 Split approach

To put it simply, computations with domain decomposition on the basis of the simulation leads to obtaining results from the assigned CCP task. Essentially, dynamic load balancing calculates new weights for different task types to define them as inputs. In this case, all actions occur during simulation in a one-dimensional decomposition of the application area. However, this does not guarantee a positive performance impact, as it is difficult to track local changes in the workload. Therefore, one can consider a more rigid withdrawal of the imbalance. This is necessary to assign new domain offsets for splitting to the simulation base. Naturally, at the initial stage, it is advisable to use the above method in order to guarantee a good start for the next steps. Further, on the basis of the received data, such as the percentage of imbalance and weights, individual offsets are automatically determined. The cumulative load imbalance can be represented by the formula below as a starting point:

$$s_j=\sum_{i=0}^{j-1} l_i-1, \forall j\in\{1,...,N\} \quad \text{c} \quad s_0=0 \tag{26}$$

Every meaning quantifies the total area load imbalance on both sides (left and right) of the split point. Then it is necessary to make a shift oj during the simulation to reduce the cumulative imbalance. Thus, a general load imbalance emerges as a consequence of determining each offset separately. If to consider large-scale problems [113], based on CFD separation, the above method can be taken as an incremental DLB method based on diffusion [108]. According to formula (26), one can obtain the total imbalance sj, having li as the computational load for each parallel subdomain. As a result, the imbalance difference on both sides (left and right) of each position is quantified. What shows with the value s2=0.45 in the example, the other two processors have a computational load significantly lower than the first two. It follows from this that the load must be shifted to the left from overloaded processors to underloaded ones. As you can see, in this way, the global imbalance is minimized

as a result of reducing the shifting local domain's offset.

$$d_j = -sign(s_j) \tag{27}$$

For cell wk of section k in domain i, the load distribution can be represented as follows:

$$\tilde{w}_k = l_i \overline{w}_k \quad \text{c} \quad \overline{w}_k = \frac{w_k}{W_i}$$

this can be represented as li load layout about the cells of the local section. Equalizing the total imbalance sj by the crossed shares of the load on the partition cells

$$s_j^k = s_j^{k-1} + d_j \cdot f_{penalty} \cdot \tilde{w}_m(k) \quad \text{for } k \geq 1 \text{ c } m(k) = o_j + d_j k - \frac{1}{2}(d_j + 1) \tag{28}$$

and $s_j^0 = s_j$

the required displacement of each offset oj can be estimated from k, which is the index of the sequence. For this index, the statement $s_j^k \approx 0$ is also true. Take fpenalty≥1 as an additional penalty factor to avoid outliers and limit biases. This method is shown using Equation (22) for the offset o2 of the second region at fpenalty=1.25. According to formula (28), s2=0.45, taken as the initial cumulative imbalance, is minimized due to a shift to a new domain by shifting by two separating cells.

$$s_2^0 = 0.45 \rightarrow s_2^1 = 0.45 - 1.25 \cdot 1.2 \cdot 0.1 = 0.3 \quad \rightarrow s_2^2 = 0.3 - 1.25 \cdot 1.2 \cdot 0.15 = 0.075 \tag{29}$$

Moreover, taking into account the performance indicators that evaluate processing speed on different computing nodes, this method can be scaled to special calculations on different computing equipment. By comparing the local load with the reciprocal of the relative domain workload, the relative processing performance can be obtained. In order to quantify the relative computational speeds [114], the power weight has the following form:

$$p_j = l_i \frac{\overline{W}}{W_i}$$

The transfer of excess load to faster processors from slower ones and vice versa can be done by using an additional value in the iterative process as the corresponding computing power, according to formula (28). For instance, having an upper bound on

processing power as pi-1=0.8 and a lower bound as pi=1.2, the workload redistributed from fast to slow processors is weighted as pi/pi-1=1.5. As a result, it can be seen that this method is suitable for both types of loads (static and dynamic). In the dynamic case, keep in mind that frequent reallocation should have a positive effect on performance.

That is, it is necessary to catch a certain balance between these processes. In this work, the DLB algorithm is used in serial format, which in each domain collects data from cell workloads. The hierarchical parallel version can be extended to minimize communication costs and memory requirements. However, in the case where only one value per calculation cell is considered, the size of the section is quite small in comparison, for instance with a huge amount of loading information [115, 116].

5.11 Parallel performance analysis

Figure 57 shows the results of the load obtained from the problem of air transport in the human respiratory system. From 28 to 84 cores were used to obtain data. The results are compared with and without balancing applications. The time spent on calculations and communication between all parallel subdomains was taken into account to calculate the average time spent on modeling a step. A more uniform distribution between parallel subdomains can be observed in the overall computational load under the condition of using DLB. With CFD, more ranks are captured. Moreover, there is a decrease in the imbalance of I from 1% to 9%. Whereas the usage was at 22%. Based on this indicator, even more significant savings can be achieved by an ideal balance. And also improve the initial partition. But due to some complex modeling complexities, such as tasks with different workloads, can be a hindrance.

A significant advantage of the DLB scheme is the possibility of its implementation, regardless of the problem statement (with minimal information). In Figure 57, you can see the above distribution of the share of execution time, with different computing cores. However, for cores with significant computational dominance, the computational simulation load was overestimated, while for domains with a higher proportion, it was underestimated. It can be concluded that, under the condition of exact computational weights, the decomposition of the domain as a result cannot be ideally optimal. This is due to differences in load compositions with different computation times. Which gives an improvement in productivity with time savings. The required total resources for one simulation time step under the condition of strong scaling (from 28-84 cores) with and without load balancing are shown in Figure 58. Moreover, the average downtime for each configuration is shown. With a small number of computing cores, there is a slight imbalance. This fact can be explained by the low degree of parallelism, in which a very large load falls on each processor, with inaccurate computational weights. This is what keeps the imbalance in check. Further, you can notice improvements in the overall performance of parallelization when using DLB. At 28 cores, the standard simulation imbalance starts at I%=9.6% and grows to I%=31% at 84 cores. By reducing local tasks, a 20%

increase in the total amount of computing resources can be achieved. And with DLB, the imbalance is I%=4.6% for 28 cores and I%=22% for 84 cores.

It can be said that the degree of parallelism and the potential of DLB are directly proportional. That is, it is possible to further improve the dynamic load distribution. With a large number of cores, the worst performance of the standard simulation of calculated weights is observed. For low degrees of parallelism, the weighted design factor is close to the calculated prior that is being used, or the initial partition.

There is a decrease in this ratio with a large number of cores. This is due to the decrease in solver efficiency when simulating in each parallel subdomain of the problem with a smaller size, based on the calculations of observations of time per cell and time step. This ratio of weights, due to its inconsistency, leads to a deterioration in the percentage of imbalance. Moreover, on the Figures 58a and 58b imbalances are presented in the ratio time to the nodes. These diagrams also reinforce the above results.

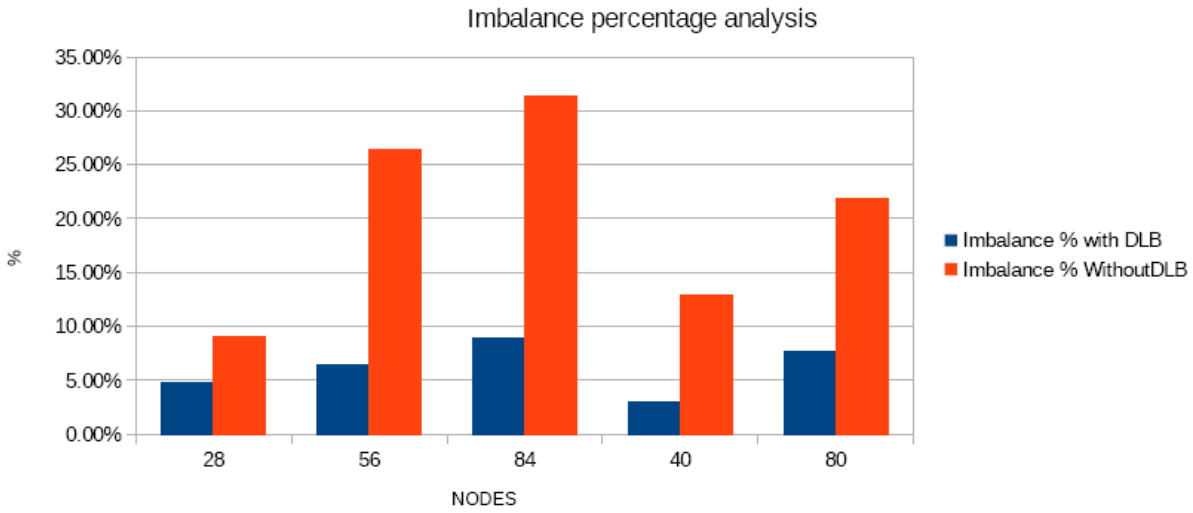


Figure 57 - Distribution of the share of execution time with different computing cores.

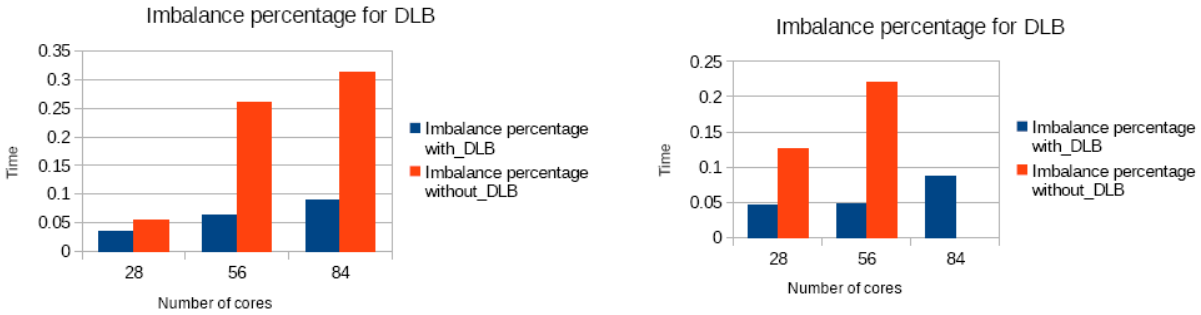
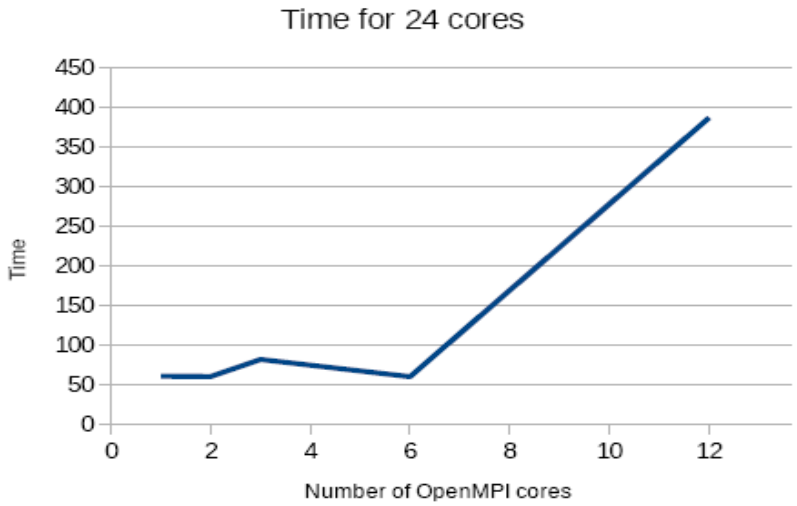


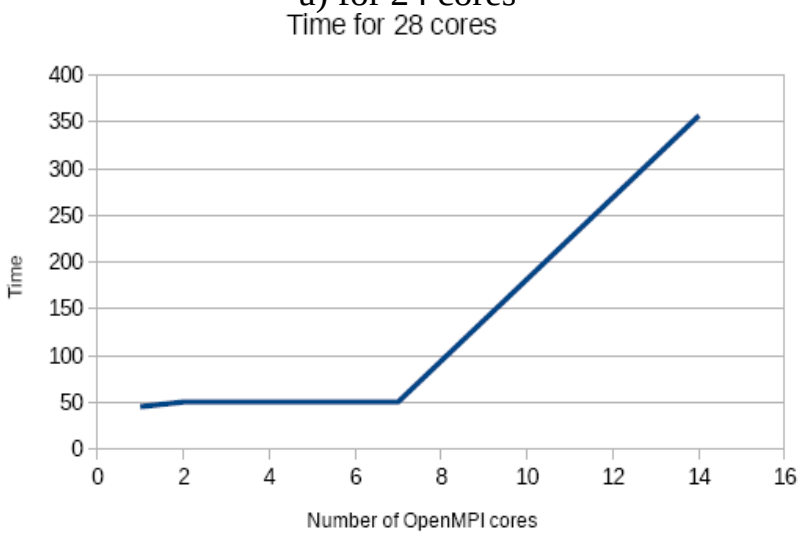
Figure 58 - Imbalances percentage in the ratio time to the nodes.

Furthermore, the research was done in order to find out the impact of the number of OpenMPI cores to the execution time. Let us consider Figure 59b time for 24 cores. Couple OpenMPI cores do not make any scene from the point of dynamic load balancing. Indicators of time worsened a little on three OpenMPI nodes.

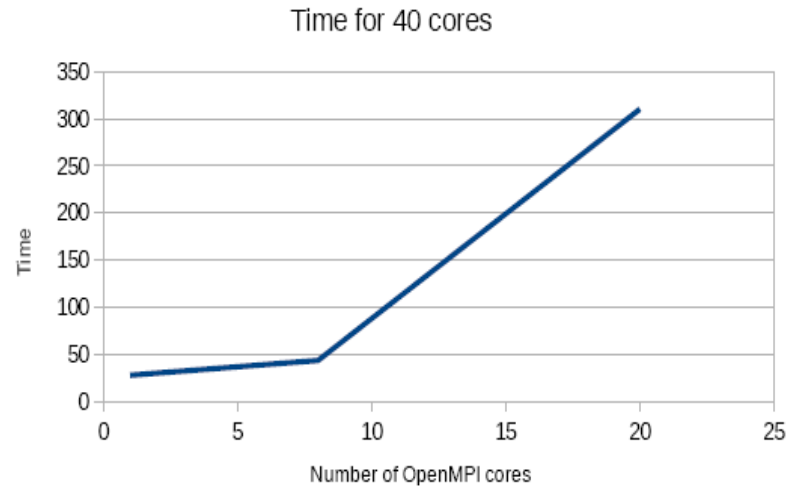
Nevertheless, after 4 nodes time returned to the initial value. But the critical point was 6 OpenMPI cores, as the line gradually went up. Next Figure 59c for 28 cores showed stable time until 7 OpenMPI cores. After which the time increased dramatically. Research for 40 cores first showed subtle changes in time under 7 OpenMPI cores, but then leap in time value was from 50 to more than 300 while increasing ten nodes with OpenMPI. Uniform increase in temporal parameters was observed 48, 56 and 80 cores.



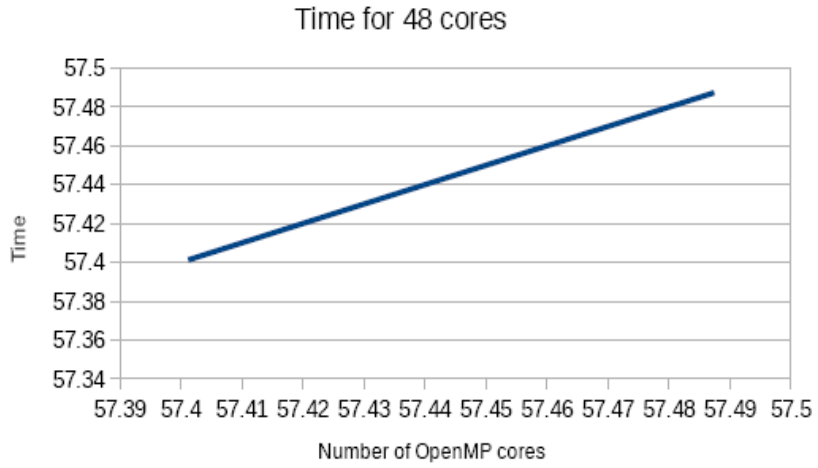
a) for 24 cores



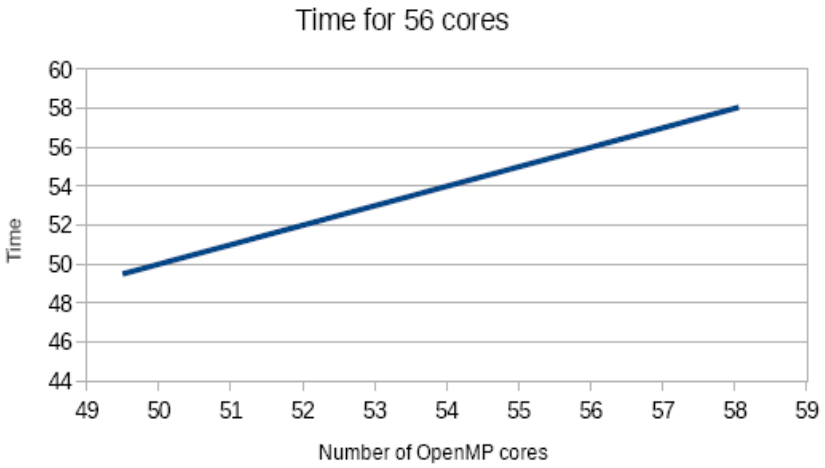
b) for 28 cores



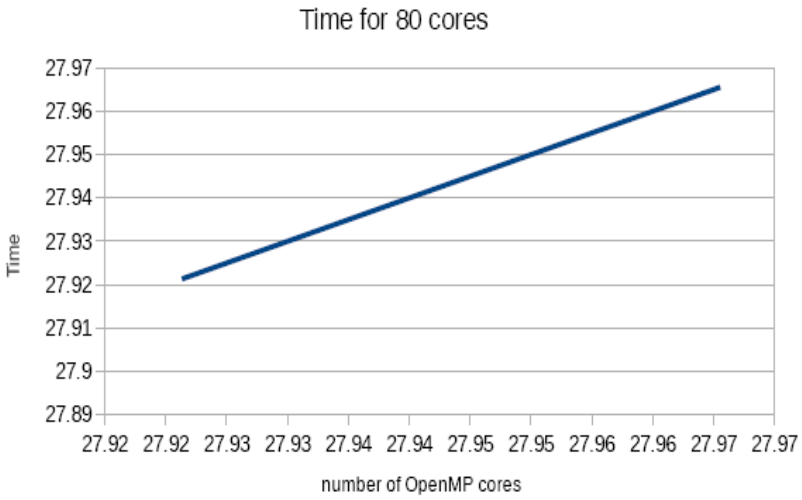
c) 40 cores



d) for 48 cores



e) for 56 cores



f) for 80 cores

Figure 59 - Distribution of the execution time with different cores.

5.12 Conclusion

In fact, it is very difficult to achieve high efficiency for large-scale parallelized tasks. Since even the slightest imbalance can lead to dramatic results in the picture of overall performance. DLB, in turn, makes it possible to increase efficiency in complex modeling with non-trivial domain decomposition. This possibility is provided by the Hilbert Space Filling Curve (HFC) at a coarse level. Since it is necessary to take into account various numerical methods, as well as various computational costs. This scheme automatically appropriate weights to estimate the overall workload distribution. Due to the inability of this assessment procedure catch local changes in workload, a general approach to load balancing SFC based may not be optimal. Therefore, the incremental diffusion DLB algorithm based on SFC separation, which allows customize domain decomposition. Modeling air movement in the nasal cavity demonstrates efficiency DLB schematics for various large-scale related tasks. Detailed analysis performance showed the need for the DLB method for direct determination of load imbalances, which, for example, are caused by individual computational efficiency depending on the composition of the local workload, and scalability of individual program codes. Also, an experiment with a strong scaling has shown an improvement in performance with increasing degree parallelism when a priori estimated computational weights are used for initial partition. Thus, studies of the nasal cavity made it clear that their walls contribute to the heating of the air and the occurrence of vortices. What is important in the transition of incoming air to the alveolar state, before entering the nasopharynx. Moreover, the importance of humidity must be taken into account. Since there is a need for heating at low ambient temperatures. Due to the increase in the number of people with nose problems, the relevance of this topic is increasing every year. To solve this problem, a surgical method can be proposed that provides optimal control for the normal functioning of the nose.

CONCLUSION

The paper developed a dynamic load balancing (DLB) scheme to improve the performance efficiency of massively parallel computing. So this method with different numerical methods and different computational costs per divided cell is obtained using the Hilbert space filling curve (SFC) at a rough level. In order to give a complete assessment, this scheme was applied for various tasks, such as flows behind a backward-facing step and air flows in a complex nasal region.

Comparison of the obtained simulation results with numerical data and experimental data of other authors was carried out, and to evaluate the parallel numerical algorithm, a comparison of the parallel algorithm, hybrid parallel algorithm and an algorithm using a dynamic load balancing (DLB) scheme was carried out.

Brief conclusions based on the results of dissertation research.

— a numerical study of the efficiency of high-performance computing for flow problems behind a backward-facing step has been carried out

— a numerical study of the efficiency of high-performance computing when using hybrid parallel algorithms for problems of air flow in a complex nasal region was carried out

— a hybrid parallel numerical computation was carried out using various methods of domain decomposition

— a hybrid parallel numerical computation was carried out using the dynamic load balancing method

— an estimate is given of the efficiency of hybrid parallel numerical computation using various methods of domain decomposition

— an estimate of the efficiency of the hybrid parallel numerical algorithm using the dynamic load balancing method is given

— Comparison of the obtained simulation results with numerical data and experimental data of other authors was carried out.

— the analysis of the obtained results of hybrid parallel numerical computation and hybrid parallel numerical computation was carried out using the method of dynamic load balancing

It should be noted that the results obtained gave rise to new problems. Since the dynamic load balancing (DLB) scheme does not require any modification, in the future it will be possible to apply this approach to simulate various three-dimensional physical and technical problems. In the future, it is planned to consider these tasks.

The results obtained can be used to solve a number of applied problems.

For numerical calculations, adequate numerical methods and algorithms are used in the work, and the results obtained are compared with the calculated and experimental data of other authors. Thus, the results of this work can find wide application in the field of distributed computing and information technology.

Evaluation of the completeness of the solution of the assigned tasks. In the work, a dynamic load balancing (DLB) scheme has been built, which allows to increase the efficiency of the complex, on the basis of which mathematical modeling

was performed, and the numerical calculation of three-dimensional various physical and technical problems, this allows us to conclude that the goal of the work has been achieved and the research tasks have been completely solved.

Development of recommendations and baseline data for the specific use of the results. The results obtained in this work are recommended to be used for the implementation of projects related to solving problems in the field of distributed computing and information technology.

Assessment of the scientific level of the work performed in comparison with the best achievements in the field. The thesis contains new science-based results that solve an important scientific problem, namely, the problem of high performance computing for various problems using a dynamic load balancing (DLB) scheme.

REFERENCES

1 Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, and S. Tuecke. Wide-Area Implementation of the Message Passing Interface // Parallel Computing. – 1998. - 24(12-13). – P.1735–1749.

2 MPI Forum. MPI: A Message-Passing Interface standard. Version 3.0, 2012.

3 Sohankar, A., Norberg, C. & Davidson, L. Simulation of three-dimensional flow around a square cylinder at moderate Reynolds numbers // Phys. Fluids. - 11, 1999. –P.288–306.

4 Saha, A. K., Biswas, G. & Muralidhar, K. Three-dimensional study of flow past a square cylinder at low Reynolds number // Intl J. Heat Fluid Flow. – 2003. - 24. – P. 54–66.

5 Robichaux, J., Balachandar, S. & Vanka, S. P. Three-dimensional Floquet instability of the wake of square cylinder // Phys. Fluids. – 1999. - 11. – P.560–578.

6 Lighthill M. J. Laminar boundery layers. Chap. II. – London: Oxford University Press, 1963. – P. 355.

7 Davis R. W., Moor E. F. A numerical study of vortex shedding from rectangles // J. Fluid Mech.– 1982.– 116. - P. 475–506.

8 Performance analysis of MPI collective operations [Text] / Jelena Pjesivac-Grbovic, Thara Angskun, George Bosilca [et al.] // Cluster Computing. — 2007. — Vol. 10, no. 2. — P. 127–143.

9 Torsten Hoefler and Dmitry Moor Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations. - ETH Z¨urich, 2014.

10 Perrone Michael, Fast Scalable Reverse Time Migration Seismic Imaging on Blue Gene/P. / Michael Perrone, Lurng-Kuo Liu, Ligang Lu, K Magerlein, Changhoan Kim, I. Fedulova, A. Semenikhin, IBM TJ Watson Research Cente, NY, IBM Russia Systems and Technology Lab. - Moscow, 2011.

11 Issakhov A., Shaibekova A. Mathematical modelling of flow around obstacles with complex geometric configuration in a viscous incompressible medium // International Journal of Mathematics and Physics. - 2016. - Volume 7. Number 1. . - P. 40-45.

12 Anderson D., Tannehill J., and Pletcher R. Computational fluid dynamics and heat transfer. - New York: McGraw-Hill Book Company, 1984.

13 Hirsch C. Numerical computation of Internal and External flows. - New York: John Wiley&Sons, 1988. - volime 1,2.

14 Hockney W. The communication challenge for mpp: Intel paragon and meiko cs-2 // Parallel computing. – 1994. - vol. 20 no. 3. - P. 389–398.

15 Chung T.J. Computational fluid dynamics. – 2002. - P.1034.

16 Amdahls and Gustafson laws. http://www.drdobbs.com/parallel/amdahls-law-vs-gustafson-barsis-law/240162980?pgno=2

17 Alexandrov A., Ionescu M. F., Schauser K. E., and Scheiman C. LogGP: Incorporating Long Messages into the LogP Model // One Step Closer Towards a Realistic Model for Parallel Computation. In Proceedings of the Seventh Annual

ACM Symposium on Parallel Algorithms and Architectures, SPAA ’95. – New York, NY, USA 1995. – P. 95–105.

18 Korthikanti A. and Agha G. Towards optimizing energy costs of algorithms for shared memory architectures // InF.M.aufderHeideandC.A.Phillips,editors,SPAA, - 2010. - P.157–165.

19 Korthikanti V. A. and Agha G. Energy-performance trade-off analysis of parallel algorithms for shared memory architectures. In Sustainable Computing: Informatics and Systems. - In Press. - 2011.

20 Kielmann T., Hofman R. F. H., Bal H. E., Plaat A., and Bhoedjang R. A. F. MagPIe // MPI’s Collective Communication Operations for Clustered Wide Area Systems. In Proceedings of the Seventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP ’99. - New York: NY, USA, 1999. ACM. - P. 131–140.

21 Brooks E. D. III. The butterfly barrier // Int. J. Parallel Program. – Oct. 1986. - 15(4) - P.295–307.

22 Thakur R., Rabenseifner R., and Gropp W. Optimization of Collective communication operations in MPICH // International Journal of High Performance Computing Applications. - 2005. - 19. – P.49–66.

23 Bruck J., Ho C.-T., Kipnis S., Upfal E., and Weathersby D. Efficient algorithms for all-to-all communications in multiport message-passing systems. IEEE Transactions on Parallel and Distributed Systems, - 1997. – 8 - P.1143 – 1156.

24 Hoefler T., Lichei A., and Rehm W. Low-Overhead LogGP Parameter Assessment for Modern Interconnection Networks. In Proceedings of the 21st IEEE International Parallel & Distributed Processing Symposium, PMEO’07 Workshop. IEEE Computer Society, Mar. 2007.

25 Kim E. J., Link G., Yum K. H., Vijaykrishnan N., Kandemir M., Irwin M., and Das C. A holistic approach to designing energy-efficient cluster interconnects. Computers, IEEE Transactions on. - Jun 2005. - 54(6). – P.660–671.

26 Madhavan S. Transition to three-dimensional models for flow past a confined square cylinder. PhD thesis: 2011. - University of Alberta, Edmonton, AB, Canada – P.68.

27 Thakur R., Rabenseifner R., and Gropp W. Optimization of Collective communication operations in MPICH. International Journal of High Performance Computing Applications, - 2005 – 19 – P.49–66.

28 Traff J. L. and Ripke A. Optimal broadcast for fully connected networks. In Proceedings of the First International Conference on High Performance Computing and Communications // HPCC’05. – Berlin, Heidelberg, 2005. Springer-Verlag. – P. 45–56.

29 Vadhiyar S. S., Fagg G. E., and Dongarra J. Automatically tuned collective communications // In Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, SC ’00, Washington, DC, USA. – 2000. - IEEE Computer Society.

30 Wagner A., Buntinas D., Panda D., and Brightwell R. Application-bypass reduction for large-scale clusters // In Cluster Computing. – Dec, 2003. - Proceedings. 2003 IEEE International Conference. – P. 404–411.

31 Watts J. and Van De Geijn R. A pipelined broadcast for multidimensional meshes // Parallel Processing Letters – 1995. - 5(02). – P.281–292.

32 Yu W., Panda D. K., and Buntinas D. Scalable, High-performance NIC-based All-to-all Broadcast over Myrinet/GM // In Proceedings of the 2004 IEEE International Conference on Cluster Computing - Washington, DC, USA, 2004. IEEE Computer Society.- P. 125–134.

33 Sur S., Bondhugula U. K. R., Mamidala A., Jin H. W., and Panda D. K. High performance, based all-to-all broadcast for infiniband clusters // In Proceedings of the 12th International Conference on High Performance Computing, HiPC'05. - Berlin, Heidelberg, 2005.Springer-Verlag. - P. 148–157.

34 Thakur R. and Gropp W. Improving the performance of collective operations in MPICH // In Recent Advances in Parallel Virtual Machine and Message Passing Interface. - Springer Verlag, 2003. - Number 2840 in LNCS, Springer Verlag (2003) 257267 10th European PVM/MPI Users Group Meeting. – P. 257–267.

35 Vadhiyar S. S., Fagg G. E., and Dongarra J. Automatically tuned collective communications // In Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, SC '00. - Washington, DC, USA, 2000. IEEE Computer Society.

36 Wagner A., Buntinas D., Panda D., and Brightwell R. Application-bypass reduction for large-scale clusters. In Cluster Computing, 2003 // Proceedings. 2003 IEEE International Conference. - Dec 2003. – P. 404–411.

37 Abbott D.E., Kline S.J. Experimental investigations of subsonic turbulent flow over single and double backward-facing steps // J. Basic Engng. – 1962. - V.84. – P. 317.

38 Sebanr A. Heat transfer to the turbulent separated flows of air downstream of a step in the surface of a plate // J. Heat Transfer. – 1964. – V.86. – P. 259.

39 Goldsteinr J., Eriksenv L., Olsonr M., Eckerte R.G. Laminar separation, reattachment and transition of flow over a downstream-facing step // J. Basic Engng. – 1970. – V.92. – P. 732.

40 Durst F., Whitelawj H. Aerodynamic properties of separated gas flows: existing measurements techniques and new optical geometry for the laser-Doppler anemometer // Prog. Heat Mass Transfer. – 1971. – V.4. – P. 311.

41 Gosmana D., Punw M. Lecture notes for course entitled: 'Calculation of recirculating flow' // Heat Transfer Rep. – 1974. – V.74. – P. 2.

42 Kumara, Yajnikk S. Internal separated flows at large Reynolds number // J. Fluid Mech. – 1980. – V.97. – P. 27.

43 Chiang T.P., Tony W.H., Sheu, Fang C.C. Numerical investigation of vortical evolution in backward-facing step expansion flow // Appl. Math. – 1999. – V.23. – P. 915-932.

44 Fletcher C.A.J. Computational techniques for fluid dynamics 2 // Springer-Verlag New York. – 1988. – V.1. – P. 387.

45 Lin J.T., Armaly B.F., Chen T.S. Mixed convection in buoyancy-assisting, vertical backward-facing step flows. International Journal of Heat and Mass Transfer. - October 1990. - Volume 33, Issue 10. – P. 2121-2132.

46 Armaly B. F. and Durst F, Reattachment length and recirculation regions downstream of two dimensional single backward facing step // In Momentum and Hear Transfer Process in Recirculating Flows, ASME HTD. - ASME, New York (1980). - Vol. 13. - P. l-7.

47 Eaton J. K. and Johnson J. P. A review of research on subsonic turbulent flow reattachment // AfAA J. – 1981. – 19. – P.1093- 1100.

48 Simpson R. L. A review of some phenomena in turbulent flow separation, // J. Fluid Engng. – 1981. – 103. – P.520-530.

49 Aung W. An experimental study of laminar heat transfer downstream of backsteps // J. Heat Transfer. – 1983. – 105. - P.823-829.

50 Aung W. Separated forced convection // Proc. ASMEIJSME Thermal Enana Joint Con. - ASME. New York (1983). - Vol. 2. DD. – P.499-515.

51 Aung W., Baron A. and Tsou F. K. Wall independency and effect of initial shear-layer thickness in separated flow and heat transfer // Int. J. Hear Muss Transfer. – 1985. – 28. – P.1757-1771.

52 Aung W. and Worku G. Theory of fully developed. combined convection including flow reversal // J. Hear Transfer. – 1986. – 108. – P.485-488.

53 Sparrow E. M., Chrysler G. M. and Azevedo L. F. Observed flow reversals and measured-predicted Nusselt numbers for natural convection in a one-sided heated vertical channel // J. Heat Transfer. – 1984. – 106. – P.325-332.

54 Sparrow E. M., Kang S. S. and Chuck W. Relation between the points of flow reattachment and maximum heat transfer for regions of flow separation // Int. J. Hear Mass Transfer. – 1987. – 30. – P.1237-1246.

55 Sparrow E. M. and Chuck W. PC solutions for heat transfer and fluid flow downstream of an abrupt, asymmetric enlargement in a channel // Numer. Hear Transfer. – 1987. – 12. – P. 1940.

56 Chung T.J. Computational fluid dynamics. - 2002. - P.1034.

57 Issakhov A., Mathematical modeling of the discharged heat water effect on the aquatic environment from thermal power plant // International Journal of Nonlinear Science and Numerical Simulation, – 2015. - 16(5). - P. 229–238, doi:10.1515/ijnsns-2015-0047.

58 Issakhov A., Mathematical modeling of the discharged heat water effect on the aquatic environment from thermal power plant under various operational capacities // Applied Mathematical Modelling, –2016. - Volume 40, Issue 2. - P. 1082–1096 http://dx.doi.org/10.1016/j.apm.2015.06.024.

59 Issakhov A. Large eddy simulation of turbulent mixing by using 3D decomposition method // *J. Phys.: Conf. Ser.* – 2011. - 318(4). - P. 1282-1288, doi:10.1088/1742-6596/318/4/042051.

60 Chorin A.J. Numerical solution of the Navier-Stokes equations // Math. Comp. –1968. – 22. - P. 745-762.

61 Karniadakis G. E., Kirby II R. M. Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation. - Cambridge University Press, 2000. - P. 630.

62 Ngo I., Byon C., Effects of heater location and heater size on the natural convection heat transfer in a square cavity using finite element method. - J. Mech. Sci. Technol, 2015. - 29 (7) – P.2995.

63 Oztop H. F., Abu-Nada E. Numerical study of natural convection in partially heated rectangular enclosures filled with nanofluids // Int. J. Heat. Fluid Fl. – 2008. - 29 (5). – P.1326 - 1336.

64 Cole, P. Some aspects of temperature, moisture and heat relationships in the upper respiratory tract // J. Laryngol. Otol. – 1953. – 67. – P.449–456.

65 Ingelstedt, S. Studies on conditioning of air in the respiratory tract // Acta Oto-Laryngol. - Suppl. – 1956. – 131. – P.1–80.

66 Webb, P. Air temperatures in respiratory tracts of resting subjects. J. Appl. Physiol. – 1951. - 4. - P.378–382.

67 Hanna, L. M., and P. W. Scherer. Measurement of local mass transfer coefficients in a cast model of the human upper respiratory tract // J. Biomech. Eng. . – 1951. – 108. – P.12–18.

68 McFadden, E. R. Respiratory heat and water exchange: Physiological and clinical implications // J. Appl. Physiol. – 1983. – 54. – P.331–336.

69 Doorly, D., Taylor, D., Schroter, R., Mechanics of airflow in the human nasal airways // Respir. Physiol. Neurobiol. – 2008 – 163. – P. 100–110.

70 Girardin, M., E. Bilgen, and P. Arbour. Experimental study of velocity fields in a human nasal fossa by laser anemometry // Ann. Otol. Rhinol. Laryngol. – 1983. – 92. – P.231–236.

71 Anderson, N.J., Cassidy, P.E., Janssen, L.L., Dengel, D.R. Peak inspiratory flows of adults exercising at light, moderate and heavy work loads // J.-Int. Soc. Respir. Prot. – 2006. – 23. – P. 53.

72 Rennie, C.E., Gouder, K.A., Taylor, D.J., Tolley, N.S., Schroter, R.C., Doorly, D.J. Nasal inspiratory flow: at rest and sniffing // Int. Forum Allergy Rhinol. – 2011. – 1. – P. 128– 135.

73 Doorly, D., Taylor, D., Franke, P., Schroter, R., Experimental investigation of nasal airflow // Proc. Inst. Mech. Eng. Part H: J. Eng. Med. – 2008. – 222. – P. 439–453.

74 Croce, C., Fodil, R., Durand, M., Sbirlea-Apiou, G., Caillibotte, G., Papon, J.-F., Blondeau, J.-R., Coste, A., Isabey, D., Louis, B. In vitro experiments and numerical simulations of airflow in realistic nasal airway geometry // Ann. Biomed. Eng. – 2006 – 34. – P.997–1007.

75 Keyhani, K., Scherer, P., Mozell, M. Numerical simulation of airflow in the human nasal cavity // J. Biomech. – 1995. - Eng. 117. – P. 429–441.

76 Wang, T., Chen, D., Wang, P., Chen, J., Deng, J. Investigation on the nasal airflow characteristics of anterior nasal cavity stenosis. Braz // J. Med. Biol. Res. – 2016. – P.49.

77 Naftali, S., Schroter R. C., Shiner R., J., Elad D., Transport Phenomena in the Human Nasal Cavity: A Computational Model // Annals of Biomedical Engineering. – 1998. – 26. – P. 831-839.

78 Naftali S, Rosenfeld M, Wolf M, Elad D The air-conditioning capacity of the human nose // Ann Biomed Eng. – 2005. – 33. – P.545–553.

79 Hahn, I., Scherer, P.W., Mozell, M.M. Velocity profiles measured for airflow through a large-scale model of the human nasal cavity // J. Appl. Physiol. - 2005 – 75. – P. 2273–2287.

80 Mylavarapu, G., Murugappan, S., Mihaescu, M., Kalra, M., Khosla, S., Gutmark, E. Validation of computational fluid dynamics methodology used for human upper airway flow simulations // J. Biomech. – 2009. – 42. – P. 1553–1559.

81 Weinhold, I., Mlynski, G. Numerical simulation of airflow in the human nose // Eur. Arch. Oto-Rhino-Laryngol. Head Neck – 2004. – 261. – P. 452–455.

82 Ball, C., Uddin, M., Pollard, A. High resolution turbulence modelling of airflow in an idealised human extra-thoracic airway // Comput. Fluids. – 2008. – 37. – P.943–964.

83 Chen, J., Gutmark, E. Numerical investigation of airflow in an idealized human extra-thoracic airway: a comparison study // Biomech. Model. Mechanobiol. – 2014. – 13. – P. 205–214.

84 Zhang, Z., Kleinstreuer, C. Laminar-to-turbulent fluid–nanoparticle dynamics simulations: model comparisons and nanoparticle-deposition applications // Int. J. Numer. Methods Biomed. – 2011. - Eng. 27. – P. 1930–1950.

85 Doorly, D., Taylor, D., Gambaruto, A., Schroter, R., Tolley, N. Nasal architecture: form and flow // Philos. Trans. R. Soc. Lond. A: Math. Phys. Eng. Sci. - 2008b – 366. – P.3225–3246.

86 Li, C., Jiang, J., Dong, H., Zhao, K. Computational modeling and validation of human nasal airflow under various breathing conditions // Journal of Biomechanics. – 2017. – 64. – P. 59-68.

87 Lin, C.-L., Tawhai, M.H., McLennan, G., Hoffman, E.A. Characteristics of the turbulent laryngeal jet and its effect on airflow in the human intra-thoracic airways // Respir. Physiol. Neurobiol. – 2007. – 157. – P.295–309.

88 Varghese, S.S., Frankel, S.H., Fischer, P.F. Direct numerical simulation of stenotic flows. Part 1. Steady flow // J. Fluid Mech. – 2007. – 582. – P. 253–280.

89 Wang, Y., Elghobashi, S. On locating the obstruction in the upper airway via numerical simulation // Respir. Physiol. Neurobiol. – 2014. – 193. – P. 1–10.

90 Lindemann J, Leiacker R, Rettinger G, Keck T Nasal mucosal temperature during respiration // Clin Otolaryngol. -2002. – 27. – P.135–139.

91 Lindemann J, Kühnemann S, Stehmer V, Leiacker R, Rettinger G, Keck T Temperature and humidity profile of the anterior nasal airways of patients with nasal septal perforation // Rhinology. -2001. – 39. – P.202–206.

92 Pérez-Mota, J., Solorio-Ordaz, F., & Cervantes-de Gortari, J.. Flow and air conditioning simulations of computer turbinectomized nose models // Medical & Biological Engineering & Computing – 2018. - 56 (10). – P. 1899-1910.

93 Garcia G.J.M., Schroeter J.D., Kimbell J.S. Olfactory deposition of inhaled nanoparticles in humans, Inhal. Toxicol. -2015. - 27 – P.394–403.

94 Na Y., Chung K.S., Chung S.K., Kim S.K. Effects of single-sided inferior turbinectomy on nasal function and airflow characteristics // Respir. Physiol. Neurobiol. -2012. -180 – P. 289–297.

95 Vinchurkar S., De Backer L., Vos W., Van Holsbeke C., De Backer J., De Backer W. A case series on lung deposition analysis of inhaled medication using functional imaging based computational fluid dynamics in asthmatic patients: effect of upper airway morphology and comparison with in vivo data // Inhal. Toxicol. – 2012. - 24. – P.81–88.

96 Teresco J. D., Devine K. D., Flaherty J. E. Partitioning and Dynamic Load Balancing for the Numerical Solution of Partial Differential Equations, ser. // Lecture Notes in Computational Science and Engineering. – 2006. - vol. 51. – P. 55–88. doi: 10.1007/3-540-31619-1_2

97 Hendrickson B., Devine K. Dynamic load balancing in computational mechanics, Comp. Methods in Appl. Mech. Eng. - Apr. 2000. - vol. 184, no. 2–4. - P. 485–500,. doi: 10.1016/S0045-7825(99)00241-8

98 Pinar A., Aykanat C. Fast optimal load balancing algorithms for 1D partitioning, J. Parallel Distrib. Comput., Aug. 2004. - vol. 64, no. 8, - P. 974–996. doi: 10.1016/j.jpdc.2004.05.003

99 Pilkington J. R., Baden S. B. Dynamic partitioning of non-uniform structured workloads with spacefilling curves, IEEE Trans. Parallel Distrib. Syst. - Mar. 1996 - vol. 7. - no. 3, pp. 288–300. doi:10.1109/71.491582

100 Lieber M., Nagel W. E. Scalable high-quality 1D partitioning // HPCS - Jul. 2014. - P. 112–119. doi: 10.1109/HPCSim.2014.6903676

101 Pinar A., Kartal Tabak E., Aykanat C. One-dimensional partitioning for heterogeneous systems: Theory and practice // J. Parallel Distrib. Comput. - Nov. 2008. - vol. 68. - P. 1473–1486. doi: 10.1016/j.jpdc.2008.07.005

102 Menon H., Jain N., Zheng G., Kale L. Automated load balancing invocation based on application characteristics // IEEE Cluster Comput. – 2012. - P. 373–381. doi: 10.1109/CLUSTER.2012.61

103 Issakhov A. Mathematical modeling of the discharged heat water effect on the aquatic environment from thermal power plant // International Journal of Nonlinear Science and Numerical Simulation, – 2015 - 16(5) - P. 229–238, doi:10.1515/ijnsns-2015-0047.

104 Issakhov A. Mathematical modeling of the discharged heat water effect on the aquatic environment from thermal power plant under various operational capacities // Applied Mathematical Modelling, –2016 - Volume 40, Issue 2. - P. 1082–1096 http://dx.doi.org/10.1016/j.apm.2015.06.024.

105 Ashby S. The Opportunities and Challenges of Exascale Computing. ASCAC subcommittee // US - DOE Report. - 2010.

106 Böhme D. Characterizing Load and Communication Imbalance in Parallel Applications // ser. IAS.Forschungszentrum. - Jülich, 2014. - vol. 23. doi: 10.1109/IPDPSW.2012.321

107 DeRose L., Homer B., Johnson D. Detecting Application Load Imbalance on High End Massively Parallel Systems, Parallel Processing. - Aug. 2007. - pp. 150–159. doi: 10.1007/978-3-540-74466-5_17

108 Watts J., Taylor S. A practical approach to dynamic load balancing // IEEE Trans. Parallel Distrib.Syst. – 1998 - vol. 9, no. 3. - P. 235–248. doi: 10.1109/71.674316.

109 Jetley P., Gioachin F., Mendes C., Kalé L., Quinn T., Massively parallel cosmological simulations with ChaNGa // IEEE IPDPS. - Apr. 2008. - P. 1–12. doi: 10.1109/IPDPS.2008.4536319.

110 Phillips J. C., Zheng G., Kumar S., Kalé L. V. NAMD: Biomolecular simulation on thousands of processors // ACM/IEEE Supercomputing. – 2002. - P. 1–18. doi: 10.1109/SC.2002.10019.

111 Petrini F., Kerbyson D. J., Pakin S. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q // ACM/IEEE Supercomputing. - Nov. 2003. doi: 10.1145/1048935.1050204.

112 Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D. LAPACK Users' Guide, 3rd ed. - SIAM, 1999.- ISBN 0-89871-447-8.

113 Lieber M., Gößner K., Nagel W. E. The potential of diffusive load balancing at large scale // EuroMPI. – 2016. - P. 154–157. doi: 10.1145/2966884.2966887.

114 Zhang X., Yan Y. Modeling and characterizing parallel computing performance on heterogeneous networks of workstations // Parallel Distrib. Comput. - Oct. 1995 - P. 25–34. doi:10.1109/SPDP.1995.530661.

115 Zheng G., Bhatelé A., Meneses E., Kalé L. V. Periodic hierarchical load balancing for large supercomputers // Int. J. High Perform. Comput. Appl. - Nov. 2011. - vol. 25, no. 4. - P. 371–385. doi:10.1177/1094342010394383

116 Исахов А.А., Абылкасымова А.Б., Мансурова М.Е. Применение метода балансировки нагрузки на высокопараллельных вычислительных кластерных системах // Вестник КБТУ. – 2021, – № 1 (18) – С.117-125

117 х Исахов А.А., Абылкасымова А. Исследование движения воздуха в респираторной системе человека методами математического моделирования // Известия КГТУ им. И. Раззакова. - 2016, – № 3 (39) – С.116 – 121.

118 Исахов А.А., Абылкасымова А. Свойства переноса воздуха в респираторной системе человека с помощью численного моделирования // Вестник КазНУ. - 2017. – № 1 (93) – С.105 – 118.

119 Исахов А.А., Абылкасымова А., Сакыпбекова М. Применение параллельных вычислительных технологий для моделирования процесса отрыва течения за обратным уступом в канале с учетом сил плавучести // Вестник КазНУ. - 2018. – № 1 (97) – С.143 – 158.

120 Issakhov A., Abylkassymova A. Numerical study of identification of the main characteristics of air transport in the human nasal cavity // International journal of biology and biomedical engineering. – 2017. - Volume 11. - P. 80-87 (Scopus).

121 Исахов А.А., Абылкасымова А. Применения параллельных вычислительных технологий для численного моделирования переноса воздуха в респираторной системе человека // Вестник КазНПУ – 2017. – № 1(57). – С.219-229.

122 Issakhov A.A., Abylkassymova A., M. Sakypbekova Applications of parallel computing technologies for modeling of the wind flow around the architectural obstacles with the vertical buoyancy forces // Известие НАН РК – 2018 Серия физ.-мат. – № 4(320) – С.48-57.

123 Issakhov A.A., Abylkassymova A., M. Sakypbekova Applications of parallel computing technologies for modeling the mixed convection in backward-facing step flows with the vertical buoyancy forces // International Journal of Mathematics and Physics. – 2017. - Volume 8. Number 2 (4). - P. 43-50.

124 Issakhov A.A., Abylkassymova A., Application of Parallel Computing Technologies for Numerical Simulation of Air Transport in the Human Nasal Cavity. Innovative Computing, Optimization and Its Applications // Studies in Computational Intelligence. - vol 741. Springer, Cham. – P.131-149 In: Zelinka I., Vasant P., Duy V., Dao T. (eds).

125 Issakhov A.A., Zhandaulet Y., Abylkassymova A., Issakhov As. A numerical simulation of air flow in the human respiratory system for various environmental conditions // Theoretical Biology and Medical Modelling . - 2021. – 18. - Article number: 2, doi.org/10.1186/s12976-020-00133-8 (Impact Factor: 1.68)

126 Issakhov A.A., Mardieyeva A., Zhandaulet Y., Abylkassymova A. Numerical study of air flow in the human respiratory system with rhinitis // Case Studies Thermal Engineering. Available online. - 19 May 2021. - 101079, 10.1016/j.csite.2021.101079.