The Ministry of Education and Science of the Russian Federation

Lobachevsky State University of Nizhni Novgorod

Computing Mathematics and Cybernetics faculty

The competitiveness enhancement program
of the Lobachevsky State University of Nizhni Novgorod
among the world's research and education centers

Strategic initiative

Achieving leading positions in the field of supercomputer technology and high-performance computing

Computational mathematics and cybernetics faculty

APPROVED

Dean of the Computational mathematics
and cybernetics faculty

_____ V. Gergel

"_____"_____ 2014.

Course program

Numerical Methods For Solving Differential Equations

Bachelor program "Fundamental Informatics and Information Technologies"

Specialization

General part

Professional block

Elective disciplines

Б3.В.ДВ.8

Degree

Bachelor

Education form

Full-time attendance

Nizhni Novgorod

2014

1. Course objective

The course explains numerical methods for solving ordinary and partial differential equations and approaches to their parallelization for shared memory systems.

The course is based on the materials developed in UNN with the support of Intel in 2011-2013 years. (http://www.hpcc.unn.ru/?doc=491). In 2014, a partial modification for the course materials was made as a part of the UNN competitiveness enhancement program. The changes mainly consist in the detailed planning of individual student work. In addition, the key course elements were translated into English.

The main objective of this course is to study numerical methods for solving ordinary and partial differential equations and approaches to their parallelization for shared memory systems.

This involves solving the following **problems**:

1. Studying numerical methods for solving systems of ordinary differential equations
2. Studying numerical methods for solving systems of stochastic differential equations
3. Studying numerical methods for solving systems of partial differential equations
4. Studying approaches to checking correctness and convergence of experimental results to theoretical data.
5. Studying principles of parallel algorithm construction for solving differential equations.
6. Mastering parallel programming on shared memory systems (OpenMP, TBB, Cilk Plus).
7. Studying functionality of libraries to solve auxiliary problems such as random number generation and Fourier transform (Intel MKL, FFTW).

The course targets engineers, teachers and research assistants as well as postgraduate students and students in higher education.

2. Course position in the bachelor program

The course is developed for the 4-th year students and is given in the 8-th semester. This course belongs to the elective disciplines of the professional block.

The lecture part is intended for graduates familiar with basics of numerical methods to the extent of Bachelor's program.

For practice, students are required to have basic C/C++ software development skills and experience in OpenMP parallel programming. TBB, MKL and other Intel Parallel Studio XE

capability skills will be useful but are not mandatory. Course materials will provide students will all relevant information about their use.

3. Learning outcomes and requirements

In the framework of this course, the following **competencies** are formed:

- Possessing the general culture of thinking, the ability to perceive, compile and analyze information (General Competency 1 - GC1). Students will be able to:
  - construct oral and written arguments in a logical and clear manner.
- The ability of intellectual, cultural, moral, physical, and professional self-development and self-improvement (GC 2). Students will be able to:
  - constantly improve their professional and cultural level.
- The ability to understand and apply in practice the theory of information as a fundamental scientific basis of information technology (Professional Competency 1 – PC 1). Students will be able to:
  - understand the content side of the information process, know the techniques for sending, receiving, processing, analyzing and storing data.
- The ability to understand, develop and apply modern information technology (PC 4). Students will be able to:
  - understand the concepts and implement the functionality of the following core technologies:
    - at the level of technological literacy:
      - computer systems architecture;
    - at the level of in-depth knowledge:
      - basic programming;
      - parallel and distributed computing.
  - develop and use professionally the software for supporting information systems and processes, to be able to use modern instrumental computing equipment.
- The ability to conduct research (PC 5). Students will be able to:
  - develop new algorithmic and methodological and technological solutions;
  - collect, process and interpret the data of modern research necessary to develop approaches, decisions and conclusions on appropriate scientific and professional issues.
- The ability to conduct analytical activities (PC 8). Students will be able to:

- analyze and select modern technologies and methodologies for implementing an information system.

As a result of education graduates from the course will know and be able to:

- apply the parallel algorithms for solving systems of linear equations with both matrices of general type and matrices of special form;
- analyze and decompose into parts the algorithms allowing the parallel execution;
- develop the parallel programs for shared memory computing systems using the technologies OpenMP, TBB, Cilk Plus;
- conduct computational experiments on the high performance computing systems;
- estimate efficiency of the performed parallel computations.

4. Course outline

The course consists of 1 credit, 36 hours, including 12 hours of lectures and 8 hours of practice. Practice classes can be held as lab works (students carry out assignments step-by-step under supervision) or master class (supervisor demonstrates and explains step-by-step solutions). 16 hours are allocated for individual work. The authors encourage additional work.

4.1. Course outline

Course outline is as follows:

#	Module	Semester	Week	Module type				Assessment
				Lecture	Seminar	Lab	Individual work	
1	Numerical Methods for Solving Systems of Ordinary Differential Equations	8	1	2	-	-	1	Test
2	Numerical Solution of Ordinary Differential Equations as Illustrated by Brain Modeling Problem		2-3	-	-	2	2	Test
3	Numerical Methods for Solving Systems of Stochastic Differential Equations		4	2	-	-	1	Test

4	Methods for Solving Systems of Stochastic Differential Equations as Illustrated by Financial Market Modeling		5-6	-	-	2	2	Test
5	Solving Partial Differential Equations as Illustrated by Wave Equation and Heat Transfer Equation		7-8	4			1	Test
6	Solving Partial Differential Equations as Illustrated by the Problem of Compound Option Price Computation		9-11	-	-	2	4	Test
7	Solving Partial Differential Equations as Illustrated by the Dirichlet Problem Posed for Poisson's Equation		12-14	4	-	-	1	Test
8	Fast Fourier Transform for the Problem of Heat Diffusion in a Plate		15-17		-	2	4	Test
	TOTAL:			**12**	**0**	**8**	**16**	**Final assessment form – exam**

4.2. Course description

Course content is as follows:

1. **Numerical Methods for Solving Systems of Ordinary Differential Equations**

   Investigation of the numerical methods of solving systems of ordinary differential equations and approaches to their parallelization. This lecture is dedicated to solving systems of ordinary differential equations using the Adams, Euler and Runge–Kutta methods. It discusses the issues of local error control. The lecture also describes the numerical methods to solve systems of ordinary differential equations and the method of partial discretization of partial differential equations. It also describes approaches to parallelization of methods to solve ODE systems.

2. **Numerical Solution of Ordinary Differential Equations as Illustrated by Brain Modeling Problem**

   An introduction into the problems of brain modeling, defining the respective mathematical model in a form of an ODE system and summarizes the Euler method for solving ODE systems. The practice describes a sequential and OpenMP-based parallel implementation, gives scalability analysis and features an implementation modified using

the Intel MKL random number generator. It lists the experimental results and gives a conceptual interpretation of the model problem.

3. **Numerical Methods for Solving Systems of Stochastic Differential Equations**

   An introduction to the methods for SDE numerical integration illustrated by financial market modeling. The lecture also gives background information about SDE and the respective classical explicit solutions methods, brings up for discussion the issues of method convergence and correctness of numerical modeling results.

4. **Methods for Solving Systems of Stochastic Differential Equations as Illustrated by Financial Market Modeling**

   An implementation of the numerical solution of a SDE describing a stock market. It involves numerical modeling of the Wiener process using the pseudo-random generator from the Intel MKL library. This is followed by implementation of the Euler, Milstein and Burrage-Platen methods and subsequent evaluation of the numerical solution error.

5. **Solving Partial Differential Equations as Illustrated by Wave Equation and Heat Transfer Equation**

   Investigation of solving second-order partial differential equations using the finite difference method. Solving the wave and heat equations is given in detailed description. For each example, the lecture states the problem, determines the difference scheme, describes the way of creating computation architecture and lists the experimental results.

6. **Solving Partial Differential Equations as Illustrated by the Problem of Compound Option Price Computation**

   Computation of the convertible option price whose behavior is described by means of a partial differential equation. To solve such an equation, the Crank-Nicolson scheme is used which involves solving a sequence of linear systems with a special tridiagonal matrix. The practice proposes to implement the tridiagonal matrix algorithm and parallelize the cyclic reduction method using OpenMP and TBB. Comparing efficiency of sequential implementation of methods and analyze scalability of parallel versions of the cyclic reduction method. Describing the example how Intel Parallel Amplifier XE is used to find bottlenecks of the parallel version.

7. **Solving Partial Differential Equations as Illustrated by the Dirichlet Problem Posed for Poisson's Equation**

   The lecture describes numerical solution of the Dirichlet problem posed for Poisson's equation. It defines the problem and derives a difference scheme to solve it. To solve a linear system resulting from the difference scheme, the SOR method will be used. This lecture describes a parallel numerical solution algorithm based on wavefront

computational grid traversal. To optimize the said algorithm, perform computations per grid block as part of wavefront grid processing. The lecture describes a method of solving the Poisson's equation that combines the tridiagonal matrix algorithm and Fourier transform, and an approach to its parallelization. It gives experimental results for the described algorithms.

8. **Fast Fourier Transform for the Problem of Heat Diffusion in a Plate**

   The practice deals with the stationary problem of heat diffusion in a square plate with zero temperature conditions at its edges. Solving the difference equation is based on expansion in basic functions with Fourier coefficients. The computational method consists in multiple use of the tridiagonal matrix algorithm and Fourier transform. This practice involves studying Fourier transform implementation in Intel MKL and FFTW. Developing a sequential implementation of the proposed problem solution algorithm and evaluate efficiency of the Intel MKL and FFTW sequential solvers. Developing a parallel version of the program and analyze its scalability.

5. Learning technologies

During course we use the following learning technologies: lectures, lab works, individual work, assessment techniques. Powerpoint presentations for all lectures and practical lessons are used.

6. Individual work and assessment techniques

Individual work consists of mastering theoretical and practical material according to the given references, solving practical problems, and answering on the given questions. Individual work can be done in both classes and at home. Control of individual work is performed by electronic tests. In the end of the program there is a final test.

6.1 Assessment forms

Monitoring of progress in studies is performed by tests in class that consist of assignments from the list of questions and practical problems (given below).

The final attestation is done based on the results of the final test. This test includes questions from all sections of the course.

6.2 Individual work: Questions and Practical problems

Module 1

1. Implement the Euler method and the Runge-Kutta second-order and fourth-order methods. Implement the methods to solve a first-order test equation, compare the solutions and errors. Compare runtimes.
2. Implement the Adams-Bashforth and Adams-Moulton methods. Implement the methods to solve a first-order test equation, compare the solutions and errors. Compare the results to those of Task 1.
3. Implement the Euler method and the Runge-Kutta second-order and fourth-order methods to solve ODE systems. Use these methods to solve a first-order test system. Compare the runtime and truncation errors.
4. Parallelize the implemented ODE system numerical solution methods. Evaluate efficiency of the implementations.
5. Implement the partial discretization method to solve the heat transfer problem. Apply it to the test problem. Parallelize the program using a pipelined scheme. Evaluate efficiency of the parallel version.

Module 2

1. Implement the uniform distribution generators MCG31 and MCG59 and the Skip-ahead technique of their use for parallel computation. Check the generators for correct results.
2. Develop a parallel implementation using Intel Cilk Plus. Perform computational experiments. Analyze results correctness and speed up.
3. Develop a parallel implementation using IntelTBB. Perform computational experiments. Analyze results correctness and speed up.
4. Use Intel Amplifier XE to identify the most time-consuming code sections. Think about possible optimization.

Module 3

1. Implement the Euler, Milstein and Burrage-Platen methods to solve SDEs for the purposes of finance market modeling. Check computational experiments for compliance with the theoretical method convergence orders.
2. Study other methods of SDE numerical integration, make their software implementation and analyze the convergence order.

Module 4

1. Implement parallel versions of the studied methods.

2. Implement a generator of random numbers subject to normal distribution using the Box-Muller algorithm. Check the implementation for correctness. Replace the PRNG in the financial market modeling program, perform numerical experiments and compare them to the results obtained earlier.

Module 5

1. Implement the algorithm of solving the wave equation with Type 1 boundary conditions. Parallelize computations by layer and calculate the speed up.
2. Use the algorithm to solve the heat equation with Type 1 boundary conditions using the difference scheme with the weight $\sigma = 0$ and $\sigma = 1$. Compare the computational error for the two weight values. Parallelize computations by layer and calculate the speed up.

Module 6

1. Substantiate applicability of the tridiagonal matrix algorithm for solving linear systems with a tridiagonal matrix resulting from the Crank-Nicolson scheme construction for the problem of CB pricing.
2. Evaluate the approximation error for the Crank-Nicolson scheme constructed for the problem of CB pricing.
3. Add the respective Intel MKL function to solve the tridiagonal system. Evaluate efficiency of the library functions as compared to the tridiagonal matrix algorithm and the sequential cyclic reduction method.
4. Implement the parallel tridiagonal matrix algorithm. Evaluate the algorithm efficiency as compared to the ordinary tridiagonal matrix algorithm and parallel cyclic reduction method. Explain the experimental results.
5. Implement the block tridiagonal matrix algorithm. Evaluate the algorithm efficiency as compared to all the above methods for solving tridiagonal systems. Explain the experimental results. Evaluate scalability of the implemented algorithm.

Module 7

1. Implement a sequential program of solving the test Dirichlet problem posed for the Poission's equation. Compare efficiency of your implementation with the experimental results mentioned in the lectures.
2. Implement a parallel program of solving the test Dirichlet problem posed for the Poission's equation using the wave and block wave computation scheme. Find the best block size for the latter. Evaluate efficiency of the implementations and make conclusions.

3. Do the laboratory work “Solving Sparse Linear Systems By Iterative Methods: Problem of Heat Diffusion in a Plate”. Compare efficiency of your implementation with the experimental results mentioned in the lectures. How can you reduce the computation time?

Module 8

1. Develop an implementation that uses ordinary single-dimension arrays instead if pointer arrays. Evaluate its efficiency.
2. Modify the developed program for a $l_1 \times l_2$ rectangular plate.
3. Solve the initial problem subject to inhomogeneous boundary conditions.
4. Develop your own FFT implementation that will work for a random length input sequence.
5. Implement the FFT algorithm to base 4. Evaluate the efficiency of this implementation as compared to implementations described as part of this practice. Optimize and parallelize the algorithm.

6.4 Assessment criteria

Perfect	Correct answers on >95% of the number of test questions
Excellent	Correct answers on 80-95% of the number of test questions
Very good	Correct answers on 70-79% of the number of test questions
Good	Correct answers on 60-69% of the number of test questions
Satisfactory	Correct answers on 50-59% of the number of test questions
Unsatisfactory	Correct answers on 25-49% of the number of test questions
Bad	Correct answers on <25% of the number of test questions

7. References

1. Butcher J.C. Numerical Methods for Ordinary Differential Equations. New York: John Wiley & Sons, 2003.
2. Fastest Fourier Transform in the West (FFTW) official page [http://www.fftw.org/].
3. Fastest Fourier Transform in the West documentation [http://www.fftw.org/fftw3.pdf].
4. Golub G.H., Van Loan Ch. F. Matrix Computations. The John Hopkins University Press, 1996.
5. Gong. P, He. Z and Zhu. SP. Pricing convertible bonds based on a multi-stage compound option model, Physica A, 336, 2006, 449-462.

6. Higham D.J. An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations // SIAM review, Vol. 43, No 3. – pp. 525–546.
7. Hoffman J.D. Numerical Methods for Engineers and Scientists, 2nd Edition. New York: CRC Press, 2001.
8. Intel Math Kernel Library Reference Manual. [http://software.intel.com/en-us/mkl_11.2_ref_pdf].
9. Intel Vector Statistical Library Notes. [http://software.intel.com/en-us/mkl_11.1_vslnotes_pdf]
10. Izhikevich E.M. Dynamical systems in neuroscience: geometry of excitability and bursting. – MIT Press, 2006.
11. Kincaid D.R., Cheney E.W. Numerical Analysis: Mathematics of Scientific Computing, 3rd Edition. Pacific Grove: Brooks Cole, 2001.
12. Kloeden P. E., Platen E., Schurz H. Numerical solution of SDEs through computer experiments. – Berlin: Springer, 1997.
13. Kloeden P.E., Platen E. Numerical solution of stochastic differential equations. – Berlin: Springer, 1992.
14. Nicholls J.G., Martin A.R., Fuchs P.A., Brown D.A., Diamond M.E., Weisblat D. From Neuron to Brain, 5th edition. Sunderland: Sinauer Associates Inc., 2011.
15. Oksendal B.K. Stochastic Differential Equations: An Introduction with Applications. – Berlin: Springer, 2003.
16. Rabinovich M.I., Varona P., Selverston A.I., Abarbanel H.D.I. Dynamical Principles in Neuroscience // Review of Modern Physics, vol. 78(4), 1213(53), 2006.
17. Schafter T. Numerical Integration of SDEs: A Short Tutorial, Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland, 2010. – Unpublished manuscript. – [http://infoscience.epfl.ch/record/143450/files/sde_tutorial.pdf]

8 Course support

The following software and hardware are used during course study:

Hardware

"Lobachevsky" supercomputer with the Intel Xeon Phi coprocessors is used.

Software

Intel Parallel Studio XE (C/C++ Compiler for Intel Xeon Phi, Intel Amplifier, Intel MKL) is used. Open FFTW library is used.

9 Authors

English version of the course is developed by A. Pirova participated in the Russian version preparation.

Teaching assistant _____ Anna Pirova

Course program is discussed by Software Department members.

«_____» _____ 2014; Document # _____

The head of the Software Department, prof. _____ Roman Strongin

Course program is approved by methodical commission of Computational Mathematics and Cybernetics Faculty of UNN,

«_____» _____ 2014; Document # _____

The head of the commission _____ Natalia Shestakova