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

ITERATIVE METHODS FOR SOLVING SYSTEM OF LINEAR EQUATIONS

Bachelor program "Fundamental Informatics and Information Technologies"

Specialization

General part

Professional block

Elective disciplines

Б3.В.ДВ.9

Degree

Bachelor

Education form

Full-time attendance

Nizhni Novgorod

2014

1. Course objective

This course explains popular iterative algorithms of solving linear systems and deals with a number of issues related to their parallelization on shared memory systems.

The course is based on materials developed in UNN with the support of Intel from 2011 through 2013 (http://www.hpcc.unn.ru/?doc=491). In 2014, within the competitive growth program, part of course materials were modified. The modifications mostly consisted in a detailed independent work planning. The key course elements have been translated into English.

The main course objective is to study basic iterative algorithms of solving linear systems, gain experience in developing parallel numerical algorithms for efficient use on shared memory systems.

It involves solving the following **problems**:

1. Mastering parallel programming on shared memory systems (OpenMP, TBB, Cilk Plus).
2. Studying basic iterative methods of solving linear systems (Jacobi, Seidel, SOR and direct iteration methods).
3. Studying Krylov subspace iterative methods (conjugate gradient, biconjugate gradient and generalized minimum residual methods).
4. Studying basic preconditioning algorithms (methods based on incomplete LU-factorization).
5. Studying general approaches to memory algorithm optimization and load balancing in case of parallelization.
6. Insight into efficient implementations of studied algorithms (in the course of laboratory works).

The course is intended for engineers, teaching staff, scientists and both graduate and post-graduate students.

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 of the course is oriented on the students familiar with basics of linear algebra and numerical methods to the extent of bachelor courses in natural sciences.

The students are supposed to have basic skills of designing C/C++ programs and parallel programming using OpenMP in order to perform the laboratory works. Competence in TBB and MKL as well as the other components of the Intel Parallel Studio XE package is encouraged, but not mandatory for studying the course. All the information about using Intel Parallel Studio XE is included in the course materials.

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:

- use parallel algorithms to solve linear systems with both general and special matrices;
- analyze and split algorithms into parts allowing for parallel execution;
- develop parallel programs for shared memory computing systems based on OpenMP, TBB, Cilk Plus;
- perform computational experiments on high performance computing systems;
- evaluate efficiency of the performed parallel computation.

4. Course outline

The course consists of 1 credit, 36 hours, including 8 lecture hours and 12 practice hours. 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	Introduction. Basic iterative methods.	6	1	2	–	–	1	Test
2	Solving sparse linear system by iterative methods: problem of heat diffusion in a plate		2	–	–	2	1	Test
3	Solving symmetric sparse linear systems using SOR		3-4	–	–	2	2	Test

	method with Chebyshev's acceleration							
4	Preconditioning methods		4-5	2	–	–	1	Test
5	Preconditioning using incomplete LU-factorization		6-7	–	–	2	2	Test
6	Krylov subspace iterative methods		7-10	4	–	–	4	Test
7	Solving sparse linear systems using the preconditioned conjugate gradient method		11-12	–	–	2	2	Test
8	Solving sparse linear systems using the preconditioned generalized minimum residual method		13-14	–	–	2	2	Test
9	Solving sparse linear systems using the preconditioned biconjugate gradient method		14-16	–	–	2	1	Test
	TOTAL:			**8**	–	**12**	**16**	**Final assessment form – exam**

4.2. Course description

Course content is as follows:

1. **Introduction. Basic iterative methods.**

   The objective of this lecture is to review general concepts of solving linear systems using basic iterative methods (Jacobi, Seidel and Successive Over Relaxation (SOR) methods)

2. **Solving sparse linear system by iterative methods: problem of heat diffusion in a plate**

   The purpose of this laboratory work is to see how linear systems with sparse matrices are solved using iterative methods via example of a stationary problem of heat diffusion in a rectangular plate at given temperature conditions at the plate edges.

3. **Solving symmetric sparse linear systems using SOR method with Chebyshev's acceleration**

   The purpose of this laboratory work is to implement the SOR method for solution of sparse linear systems and study ways to accelerate iterative methods by the example of the symmetric successive over relaxation.

4. **Preconditioning methods**

   The objective of this lecture is to review approaches to reduce the condition number of a matrix based on preconditioning. Preconditioning is important for iterative methods as it improves their convergence rate.

5. **Preconditioning using incomplete LU-factorization**

   The purpose of this laboratory work is implementation of preconditioner construction methods based on incomplete LU-factorization.

6. **Krylov subspace iterative methods**

   The objective of this lecture is to review a general approach to construction of Krylov subspace iterative methods. The generalized minimum residual method, conjugate gradient method and biconjugate gradient method are used as examples. The lecture also reviews preconditioning for the above algorithms.

7. **Solving sparse linear systems using the preconditioned conjugate gradient method**

   The purpose of this work is to demonstrate practical implementation of the conjugate gradient method for symmetric sparse matrices using preconditioning and study influence of the computation error on the solution accuracy.

8. **Solving sparse linear systems using the preconditioned generalized minimum residual method**

   The purpose of this laboratory work is to demonstrate practical implementation of the generalized minimum residual method and study influence of ILU(0)-preconditioning on the method convergence rate.

9. **Solving sparse linear systems using the preconditioned biconjugate gradient method**

   The purpose of this laboratory work is to demonstrate practical implementation of the biconjugate gradient method and study influence of preconditioning on the method convergence rate.

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 fixed point iteration method to solve a linear system with a SPD matrix (both its sequential and parallel versions). Estimate how the method parameter influences the convergence rate. Study the parallel algorithm scalability.

2. Implement the Jacobi and Seidel methods to solve a system of linear equations with a SPD matrix. Compare the respective convergence rates. Propose parallel implementations of the above algorithms.

3. Implement the SOR method to solve a system of linear equations with a SPD matrix. See how the method parameter influences the convergence rate. Propose a parallel implementation and study its scalability.

Module 2

1. Implement the Jacobi method as applied to a block five-diagonal matrix mentioned in this laboratory work. Think about a possible parallelization scheme.
2. Implement the Seidel method as applied to a block five-diagonal matrix mentioned in this laboratory work. Think about a possible parallelization scheme.
3. Conduct a computational experiment having found the best pipelined scheme parameter values using Intel® TBB for test grid dimensions.

Module 3

1. Study the SOR method convergence rate depending on the method parameter.
2. Implement the Conjugate Gradient (CG) method and apply it to the test problem. Compare the number of iterations for the CG and SOR methods.
3. Apply Chebyshev's acceleration procedure to the conjugate gradient method. Compare the number of iterations for the initial and accelerated method.

Module 4

1. Construct the symmetric Gauss-Seidel preconditioner. Estimate the matrix A condition number reduction using this preconditioner.

2. Construct the $ILU(p)$-preconditioner. Estimate the matrix A condition number reduction using this preconditioner with various degrees of p filling.

3. Implement a parallel version of the $ILU(p)$-preconditioner. Check your program scalability.

Module 5

1. Implement a parallel *ILU*(*p*) algorithm version and analyze its scalability.
2. Implement a block modification of the *ILU*(*p*) algorithm to increase the algorithm effectiveness for large matrices.

Module 6

1. Implement the GMRes(m) method (generalized minimal residual method with restart). Compare the number of method iterations for various restart parameter *m* values.
2. Implement the BiCG-Stab method (stabilized biconjugate gradient method) using the pseudocode indicated in [1]. Compare convergence rates of the initial and stabilized methods.
3. Use the ILU(0)-preconditioner for the methods from previous tasks. Compare the number of iterations for the initial and preconditioned methods.

Module 7

1. Apply the preconditioner *ILU(p)* to the conjugate gradient method. How will it influence the convergence rate?
2. Modify your program to implement the biconjugate gradient method. Compare convergence rates of the conjugate and biconjugate gradient methods.

Module 8

1. Use the *ILU*(*p*)-preconditioner for the GMRes method. Analyze the convergence, compare the number of iterations and method convergence for the *ILU*(0) and *ILU*(*p*) preconditioners.
2. Implement the GMRes(m) method (generalized minimum residual method with restart). Compare the number of method iterations for various restart parameter *m* values.

Module 9

1. Analyze the BiCG method convergence rate depending on precision of the floating point arithmetic. Use the float, double and long double data types and the mpfr library of real numbers.
2. Implement the BiCG-Stab method or stabilized biconjugate gradient method using the pseudocode indicated in [1]. Compare convergence rates of the initial and stabilized methods.

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. David R. Kincaid and E. Ward Cheney. Numerical analysis : mathematics of scientific computing. Brooks/Cole Publishing Company, 1991.
2. Golub, G.H., Ortega, J.M. Scientific Computing. An Introduction with Parallel Computing. Academic Press, Inc., 1993.
3. J. Dongarra et al. Templates for the solution of linear systems: building blocks for iterative methods. SIAM, 1994.
4. Gene H. Golub, Charles F. Van Loan. Matrix Computations. The John Hopkins University Press, 1996.
5. James W. Demmel. Applied Numerical Linear Algebra. SIAM, 1997.
6. Richard L. Burden, J. Douglas Faires. Numerical Analysis. Brooks Cole, 2000.
7. Joe D. Hoffman. Numerical Methods for Engineers and Scientists. CRC Press, 2001.
8. Y. Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.
9. G. Karniadakis, R. Kirby. Parallel scientific computing in C++ and MPI. Cambridge university press, 2003.
10. M. Quinn. Parallel programming in C with MPI and OpenMP. McGraw-Hill, 2004.
11. William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. Numerical Recipes. The Art of Scientific Computing. Cambridge University Press, 2007.
12. Intel Math Kernel Library https://software.intel.com/en-us/intel-mkl/

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.

9 Authors

English version of the course is developed by K. Barkalov. V. Gergel, E. Kozinov, A. Pirova, V. Kustikova participated in the Russian version preparation.

Associate prof. ____ Konstantin Barkalov

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