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”

ITERATIVE METHODS FOR SOLVING LINEAR SYSTEMS

Practice 1. Solving sparse linear systems by iterative methods:

problem of heat diffusion in a plate

Nizhni Novgorod

2014

OBJECTIVES

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.

ABSTRACT

The subject of this laboratory work is one of iterative methods of solving linear systems – the SOR method as applied to a block five-diagonal matrix. The matrix above results from solving a stationary problem of heat diffusion in a rectangular plate at given temperature conditions at the plate edges. We shall review parallel method implementations using Cilk and TBB.

GUIDELINES

The subject of this laboratory work is one of iterative methods of solving linear systems – the SOR method as applied to a block five-diagonal matrix. The matrix above results from solving a stationary problem of heat diffusion in a rectangular plate at given temperature conditions at the plate edges.

The model problem of this laboratory work is a linear system resulting from numerical solution of the Dirichlet problem posed for Poisson equation. Poisson equation is a partial differential equation for an unknown function which models heat diffusion in a rectangular plate. The finite difference method converts the equation to the linear system $Ax=b$ with a block diagonal sparse symmetric positive definite matrix.

Within the scope of this laboratory work, it is proposed to implement the successive over-relaxation (SOR) method for the purpose of solving $Ax=b$. This method is based on splitting the matrix A into three matrices $A=L+D+R$, where D is the matrix A diagonal, L is the lower and R is the upper triangle, respectively. The SOR method in the form of a matrix is described by

$$\frac{(D+\omega L)(x^{(s+1)}-x^{(s)})}{\omega}+Ax^{(s)}=b\,.$$

As A is a block diagonal matrix, it is advisable to store and process it as a band. This laboratory work reviews a number of band matrix formats, such as rowed, column and profile matrices.

Having studied the formats above, it is proposed to implement the SOR method for each such format. To check the resulting solution for consistency, we shall compare it with the exact linear system solution obtained by means of **`dpbtrf()`** and **`dpbtrs()`** of the MKL LAPACK library. The method convergence check will confirm the implementation consistency.

The following part of the work consists in implementing a parallel version of the SOR method based on Intel® Cilk Plus. In case of a brute-force approach to parallelization, the parallel program run results will not be in line with those of the sequential version. Indeed, the SOR method iterations are data-dependent; computation of the next solution approximation depends on the previous one. Thus, let us parallelize computations within a single method iteration. Computation will result in mixed approximations whose elements have been obtained using mixed new and old components without keeping strictly to the method formula. This results in a reduced method convergence rate. There are experimental results confirming this.

Therefore, a parallel computation scheme is to be developed to find the same solution as the sequential one does. For this purpose, let us implement a pipelined scheme of the SOR method using the Intel® TBB library.

According to the method computing formulae, computation of the next element $x_i^{(s+1)}$ requires approximation elements $x^{(s)}$ whose numbers are greater than i. Therefore, for each grid node, its upper and right neighbours in the cross stencil will be taken from the previous approximation, while the left and bottom ones – from the current one. The number of nodes to be computed at each line is equal to $n-1$ for the grid (n,m). Thus, computing $x_i^{(s+1)}$ requires computation of the previous approximation elements $x_j^{(s)},\ j=\overline{1,i+n-1}$. Let us use this relation to construct the pipelined method scheme. If for several approximations $x^{(s)}, x^{(s+1)}, \dots, x^{(s+p)}$ the previous approximation has at least $n-1$ more computed elements, further approximations may be computed in parallel and in an in-sync manner with a difference of $n-1$ element.

Parallel computation will be based on the Manager - Worker scheme. The Manager will coordinate computation of approximations, distribute the load among the Workers and check the method stop criteria. Workers will find the approximate solutions. The pipelined scheme will be iterative, i. e. each iteration will be further divided into three stages:

- The Manager will initialize data for the current computation step.
- Having distributed the load between the Workers, the Manager will wait for completion. The Workers will compute a certain number of elements, each for the respective approximation.
- The Manager will check the method stop criterion.

The number of Workers and physical flows of the application may not coincide. To better balance the load, each physical flow must take the load from several Workers.

The experimental results show a good (up to 6 for 8 flows) speedup of a parallel version of the pipelined algorithm.

RECOMMENDATIONS FOR STUDENTS

SOR is a classical iterative method. See [1] for a detailed description of its features. A brief method description, including pseudocode algorithms, can be found in [2]. The parallel programming technologies used for this laboratory work are described in [3 – 5].

REFERENCES

1. David R. Kincaid and E. Ward Cheney. Numerical analysis : mathematics of scientific computing. Brooks/Cole Publishing Company, 1991.
2. J. Dongarra et al. Templates for the solution of linear systems: building blocks for iterative methods. SIAM, 1994.
3. Intel® Cilk Plus documentation [http://software.intel.com/sites/products/documentation/hpc/composerxe/en-us/cpp/mac/index.htm#cref_cls/common/cilk_bk_using_cilk.htm].
4. TBB library page on the Intel website [http://software.intel.com/en-us/articles/intel-tbb/].
5. Intel® Math Kernel Library documentation [http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/].

PRACTICE

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.

TEST

1. What is the purpose of initialize(…) method of tbb::task_scheduler_init?
    a. It enables setting a minimum grain size for iterative space partition in the course of cycle parallelization.
    b. +Activation of a tbb::task_scheduler_init object; the optional parameter also enables determining the number of threads.
    c. Enables determining the number of threads.
2. If for a tbb::task_scheduler_init object the deactivation method terminate() is not explicitly called,
    a. +the object will be deactivated when the corresponding destructor is called.

b. the object will not be deactivated.
c. the object will be deactivated upon completion of the parallel program section.

3. How can one set the number of handlers in a parallel program using the capabilities of Intel® CilkPlus?
    a. Using __cilkrts_set_param(…) within the program.
    b. Using the CILK_NWORKERS environment variable.
    c. Using the respective Microsoft Visual Studio option.
    d. +All the above is possible.
4. Can the Intel® CilkPlus extension to C/C++ be used in the absence of the Intel compiler?
    a. +Yes.
    b. No.
5. Can the SOR method be considered as a direct method of solving linear systems?
    a. Yes
    b. Yes, but only for well-conditioned matrices.
    c. +No
6. For a linear system with a SPD matrix, the SOR method will converge at
    a. $+\omega \in (0, 2)$
    b. $\omega \in (-2, 2)$
    c. any ω value
7. How many nonzero diagonals has the matrix of a linear system resulting from numerical solution of the Poisson's equation in a two-dimensional region?
    a. 3
    b. + 5
    c. 7
8. SOR method
    a. Can be effectively parallelized for sparse matrices of any structure
    b. + Can be effectively parallelized for large block diagonal sparse matrices
    c. Cannot be effectively parallelized for sparse matrices