Parallel computations to solve steady-state transport equations

S.N.Lebedev, E.M.Romanova*, O.V.Stryakhnina

RFNC-VNIITF, Snezhinsk, Russia

A study is done of paralleling methods likely to be applied to solve neutron transport equation using a steady-state equation as an example. The code is used to estimate an effective neutron multiplication factor. A difference scheme based on the rhomb model of the DSn-method is used for numerical solution. A parallel version of the code can run on a set of multi-processor computers both with shared and distributed memory. Different methods of geometry decomposition have been evaluated as well as paralleling by groups or by neutron directions. Some qualitative and quantitative assessments of the computation speed-up have been done.

1. Paralleling methods for solving a steady-state transport equation

A code solving steady-state neutron transport equation is a part of the software package for emergencies at the fast reactors [1,2]. In particular, it is used to calculate the effective neutron multiplication factor. The code solves the steady-state transport equation in a 2D axisymmetrical statement in the multi-group anisotropic approximation. A difference model based on the rhomb DSn-method is built for numerical solution. A discrete ordinate method is used in the space of the neutron directions. Zeidel method or simple iteration method is applied to calculate the collision integral.

The parallel version of the code can run on a set of multi-processor computers with both shared and distributed memory. Based on the study performed, we could evaluate the advantages and disadvantages of the geometry decomposition method, paralleling by groups and by neutron directions and obtain qualitative and quantitative characteristics of the computation speed-up.

Statement of the problem to assess a critical parameter Keff.

A closed axi-symmetric region D = {r,z}, being a section of an axi-symmetric body by a plane containing the z-axis, is used to resolve the following system of transport equations for the energy groups (g=1,2,…G) in the multi-group kinetic approximation:

$$\mathrm{div}(\Omega N_g) + \alpha_g N_g = \frac{1}{2\pi}\sum_{g'=1}^{G}\int \beta^{s}_{gg'} N_g d\Omega' + \frac{1}{K_{eff}}\frac{1}{2\pi}\sum_{g'=1}^{G}\int \beta^{f}_{gg'} N_g d\Omega'.$$

Here boundary conditions are zero, $N_g(r, \Omega)$ is density of angular flux of the g-group neutrons flying in the particle direction, Ω is a unity vector in the neutron direction, α_g is a neutron absorption coefficient, $\beta^{s}_{gg'}$ is a macroscopic neutron scattering cross-section, $\beta^{f}_{gg'}$ is a cross-section of neutron scattering on the fission reactions.

Under the above conditions there exists the only positive eigenfunction up to a factor $\{N^0_g$, g=1,2…G$\}$, corresponding to a simple real eigenvalue $K_{eff} = K^0_{eff} > 0$, which is less than modulus of any other eigenvalue K^m_{eff}.

Different methods are applied to solve the system [3], for instance, iteration method for fission neutron source. An internal iteration loop implies iterations of the scattering source calculation, and the fission neutron source is assumed known from the previous external iteration loop:

$$LN_g^{(i,j)} + \alpha_g N_g^{(i,j)} = \frac{1}{2\pi}\sum_{g'=1}^{G}\beta^{s}_{gg'} SN_{g'}^{(i,j-1)} + \frac{1}{2\pi}\frac{1}{K_{i-1}}\sum_{g'=1}^{G}\beta^{f}_{gg'} SN_{g'}^{(i-1)}$$

* Corresponding author. E-mail: ye.m.romanova@vniitf.ru

Here i is external iteration number, j is internal iteration number, L is difference approximation of the transport operator div(ΩN_g), SN_g is a scalar flux of the g-group neutrons.

External iteration recalculates the fission neutron sources and parameter K^i:

$$K^{i+1} = \frac{K^i \sum_g \sum_{g'} \int_D \beta^f_{gg'} SN^{i+1}_{g'} dV}{\sum_g \sum_{g'} \int_D \beta^f_{gg'} SN^i_{g'} dV}$$

Convergence criterion for internal iteration loop is as follows $| N^i - 1 | < \varepsilon_1$, where ε_1 is an accuracy of internal iteration termination, $N^i = \frac{\| N^{i-1} \|}{\| N^i \|}$; and for the external iterations the criterion is $|K^i - K^{i-1}| < \varepsilon_2$, where ε_2 is an accuracy of K_{eff} calculation.

There are a lot of mathematical ways to speed up K_{eff} calculation though, unfortunately, none of them is universal. Practice shows that for each way there is a problem where this particular way is the longest.

The transport equation in the above statement, though simulating 2D systems, is actually solved in the multidimensional phase space. For anisotropic problems with large number of groups and neutron directions and a number of grid points of an order of 100 000, time needed to calculate one iteration of transport equation can reach several minutes. We deal with the problems where, first, the neutron transport equation is to be solved hundreds and even thousands of times and, second, a huge amount of random access memory is needed and the problems often claim an exclusive usage of the computer. Paralleling of the problems like these will not only speed up the solving process but will enable the enlargement of the problems since each processor will calculate only some part of them. For a small computer complex, paralleling is absolutely necessary since it will in addition enable other problems to get some real time for computing. The above reasons motivated the efforts made to parallel the calculation of neutron transport equation.

1.1. Geometry decomposition method

Geometry decomposition method as applied to our code splits the closed 2D area into computational regions. The main criterion of the splitting is an approximately equal number of points in each computational region. This makes the computational load on the processors almost equal, and the paralleling efficiency almost the highest. In our case the whole system is divided into the computational regions – zones. The principle of block-regular mesh construction is used afterwards. Zone size and number depend on the problem statement. Each zone is described with its own regular rectangular mesh. If some zones differ significantly in the number of points, the code can additionally split "large" computational regions, thus forming additional computational regions.

When a problem has a small number of zones, additional splitting of the system into larger number of computational regions will speed up the computation. The larger the number of computational regions differs from the initial zone number, the more iterations the problem computation will require. Thus, even in the case of acceptable accuracy of the result and high speed-up factor of an iteration, the actual speed-up factor of the whole code is as much lower, as much higher is the number of paralleled iterations compared to the number of iterations in the sequential code version. The same disadvantage is typical of all problem zones unbalanced by the number of computational points.

One more disadvantage of the geometry decomposition method is that the result of the parallel code version does not coincide with that of the sequential code version – no matter how small the difference is. Having studied the code results, we considered this method inefficient for the problems with complex geometry and currently we use it only for testing and numerical simulation of the simple problems.

1.2 Method of paralleling by the neutron directions.

To refresh your memory, in DSn-method the space where neutrons fly is covered by a special mesh, so called "tortoise", described in the literature devoted to neutron transport equations. The mesh is built by angular variables in compliance with ES_M – quadrature as shown in Fig.1:

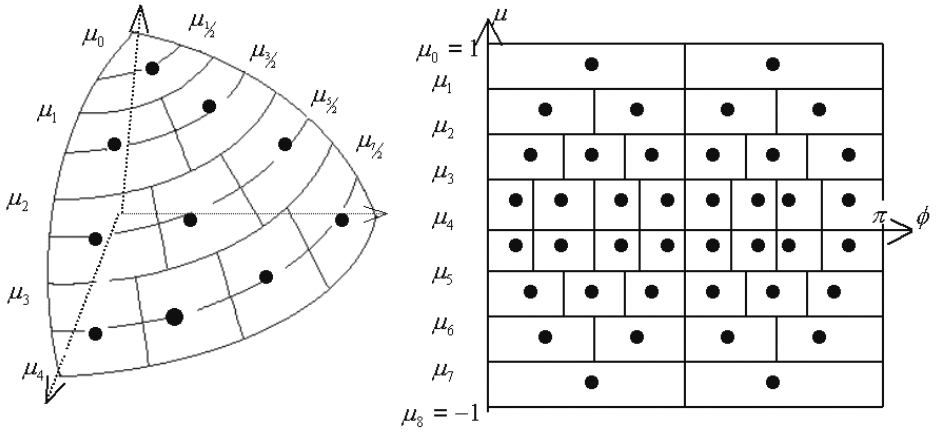

Fig.1. Mesh in the space of neutron directions.
ES_8 - quadrature.

Surface of the unit sphere describing the space of neutron directions is divided into the cells of equal area. Each "tortoise" belt, i.e. a set of cells of one and the same latitude on the unit sphere can be calculated regardless of the other belts during one iteration. Having distributed the belts among the processors so that to load the processors similarly and having then summed up the integral values of the neutron fluxes obtained at every specific point of the system by all processors, we obtain a solution coinciding with that of the sequential code version.

The method is free of the disadvantages of the geometry decomposition method, though it should be kept in mind that the number of the paralleled processes in this case cannot exceed the number of processors working with shared memory.

1.3 Paralleling by groups

The above disadvantages of the geometry decomposition method were studied with respect to the system regions and the results obtained made us draw a conclusion on the inefficiency of this method for our problems. We started paralleling by groups, not fully rejecting the geometry decomposition method. Paralleling by groups is some sort of decomposition method as well. It's implemented on distributed memory and has its own advantages and drawbacks.

Being paralleled, each group of the system is computed on a dedicated computer node. Within the node the paralleling by the neutron directions still works on shared memory. This approach was formed historically – paralleling of the transport equation solution had started long before we gained an opportunity to use several connected multi-processor computers for solving problems. We believe that difficulties are unlikely to happen during transition from paralleling on the shared memory to paralleling on the distributed one.

After an iteration is done for each group, it is necessary to calculate the source parameters which imply node-to-node exchange of huge data arrays. For anisotropic problems these arrays can comprise tens of megabytes that will decrease the paralleling efficiency.

The second and initially deemed more important drawback of this method is the use of simple iterative method instead of Zeidel iterations. However, the numerical experiments showed that the number of transport equation solutions increased slightly at the same specified accuracy. In any case, the speed-up of computations due to paralleling is much higher than deceleration caused by the increased number of the transport equation solutions.

The undoubted and important for us advantages of the method include its good balance (each processor performs the same amount of arithmetic operations) and coinciding solutions of

the parallel code version and the sequential one, to be more precise, the code version where Zeidel iterations were replaced by the simple ones.

2. Technologies of code paralleling

2.1 Paralleling on distributed memory

Since geometry decomposition method implies an arbitrary large number of the paralleled processes, paralleling on the shared memory, i.e. within the node, is not suitable for this method because of the limited number of processors in the node. Therefore, geometry decomposition method was implemented using MPI-tools.

Processor-to-processor exchanges are implemented as follows:

1. During internal iteration the computations are done within every computational region independently, regardless of other regions. These computations are done on each computer node. In each computational region Zeidel iterations are done for each group. Then each computational region transmits the values of neutron fluxes at the boundary for each group and each direction in the group to the neighboring regions. For this purpose, blocking-free operators of array transmission are used, i.e. execution of the operators does not block the other code operations, so the code continues its work, which does not need the boundary conditions from the neighboring regions – calculation of norms, etc.
2. To calculate norms for the whole system, each processor with the help of special MPI-operator sums up the norms obtained for each computational region. After that each processor has full information needed to check the convergence of iterations and, if necessary, to estimate the current value of K_{eff}. Practice shows that this redundant work is many times more effective than collection of all data on one of the processors, calculation of the norm and K_{eff} and follow-up transmission of these values to the other processors.
3. Paralleling by groups is done for each region, computations of each region being then serial. On all nodes the internal iterations for each region group are done independently from the other groups. After the transport equation is solved for each group of the region, the obtained values of the neutron fluxes needed to calculate the source in each group are transmitted from each node to all other nodes where new source values are calculated and then used during the next internal iteration. The blocking-free operators of array transmission are again used for transmission. In this case there is no need to transmit the norms since each node has all values needed to calculate the norms by the end of the region computation cycle.

2.2 Paralleling with PTHREAD tools

There is a set of special procedures in FORTRAN, which implement paralleling of computations on the shared memory. The fewer the parameters to be transmitted to each processor, the higher the speed of execution. In our case we managed to distribute data so that each processor works only with its own memory and makes no requests for reading or writing to its neighbors.

3. Estimate of paralleling efficiency

We present simulations for the problems with few groups only since there are only four eight-processor SMP Power Challenge L nodes available for us. Multi-group problems use the code version where paralleling is implemented only on shared memory, therefore speed-up factor cannot exceed the number of processors within the node, i.e. eight. We estimate the speed-up factor of the code only by a ratio of the real execution time of its parallel version to that of the sequential version. To assess the performance of the computer complex, we also estimate the speed-up of transport equation calculation since the parallel code version gives a bit more solutions to the equation and this was discussed above.

Assume, K is a speed-up factor for the problem, P is efficiency of the code paralleling:

$K=\frac{T_{посл}}{T_{пар}}$, where $T_{посл}$ is execution time of the serial code version, $T_{пар}$ is execution time of the parallel code version.

$P=\frac{K}{N}$, where N is a number of parallel processes.

K_1 and P_1 are a speed-up factor and efficiency of paralleling for a transport equation, respectively:

$K_1=\frac{t_{посл}}{t_{пар}}$, where $t_{посл}$ is execution time of one transport equation with the serial code version, $t_{пар}$ - is execution time of one transport equation with the parallel code version.

$P_1=\frac{K_1}{N}$, where N is a number of parallel processes.

The code efficiency can be illustrated by the results of 2D calculation of a steady-state transport equation in multi-group statement for a multi-layered cylinder.

A grid for angular variables consisted of 160 directions (ES_{16} - quadrature). The system contained 24 000 points. There were 10 computational regions. Within the computational regions the grid consisted of regular quadrangles. There were 6 moments of anisotropy.

The problem was solved in 3-group and 4-group statement (using 3 and 4 eight-processor computers, respectively). The table shows the execution time of the problem in seconds, speed-up factors and efficiency of the problem paralleling.

Table 1.

Number of groups	3	4
N – number of processors	24	32
K –speed-up factor of the problem	16.56	21.76
K_1 – speed-up factor of one transport equation	18.36	24.05
P – efficiency of the problem paralleling	69%	68%
P_1 – efficiency of paralleling of one transport equation	76%	75%

Conclusion

Different methods of paralleling were studied which can be applied to solve the steady-state neutron transport equations. Methods of paralleling by groups and particle directions were shown to be most effective. Geometry decomposition method can also be rather effective but for large systems with well-balanced number of points in the computational regions. Paralleling is done on distributed memory. If paralleling is by the particle directions, shared memory can be used also.

We also paralleled the solution of non-stationary neutron transport equation. We neither faced any difficulties nor introduced significant changes in the paralleling algorithm described above.

References

1. A.D.Gadzhiev, V.V.Gadzhieva, S.N.Lebedev, V.N.Pisarev, et al. Software package SINARA for simulation of the dynamics of emergency processes in nuclear-power fast-neutron facilities at the nuclear power stations. Issues of Nuclear Science and Engineering. Ser. Techniques and codes for numerical solution of the mathematical-physics problems, 2000, iss. 3, pp. 25-35.
2. V.M.Gribov, A.O.Ignatiev, A.V.Kim, E.M.Romanova, et al. Software package SINARA for simulation of the dynamics of emergency processes in nuclear-power fast-neutron facilities at the nuclear power stations. Implementation of parallel computations in LAN connecting different types of computers. Preprint of Russian Federal Nuclear Center – VNIITF N131, 1998.
3. G.J.Bell, S.Glasstone. Nuclear reactor theory. Van Nostrand Reinhold Company, 1970.