SPEEDING-UP COLLISION PROBABILITY CALCULATIONS

Dimitar Altiparmakov
AECL – Chalk River Laboratories
Chalk River, Ontario, Canada, K0J 1J0

Abstract

The requirements for extensive computing time and large computer memory have been recognized as major deficiencies of the collision probability method. This paper presents two methods for speeding up the calculations in the above two areas. One method is concerned with the calculation of collision probabilities, while the other allows a faster solution of resulting systems of linear algebraic equations. In two-dimensional geometry, the collision probabilities are expressed as linear combinations of Bickley functions, $Ki_3(x)$, the evaluation of which is the main time consumer for small and medium size problems. The new method presented here applies a numerical integration of the polar angle instead of Bickley function calculation. As a result, the algorithm is more robust and twice as fast. The solution of the collision probability equation is usually obtained by direct methods of matrix decomposition. The computing time is proportional to the third degree of the number of unknowns and increases rapidly with the increase of the problem size. To speed-up the calculation, the within-group matrix is subdivided into a number of blocks. The solution is obtained iteratively by block-matrix iteration using the traditional Gauss-Seidel method. Test results show a decrease in computing time by more than one order of magnitude.

1. Introduction

Owing to both geometric flexibility and efficiency in treating strong absorbers, integral transport methods have been used since the early era of reactor physics code development. The collision probability method was the main engine for solution of the neutron transport equation in a series of computer codes developed in 1960's. The spectrum code THERMOS [1] is credited with conceiving the use of the collision probability method as an efficient transport solver, while the code WIMS [2] is the first code that encompassed the full functionality of reactor lattice calculation by collision probability method. A few collocation methods based on Gaussian quadrature formulae have been also developed [3], [4] for one-dimensional geometry. However, the collision probability method prevailed because of the simplicity of mathematical apparatus that allows an easy extension to two-dimensional geometry.

The size and the density of collision probability matrices were recognized as major deficiencies requiring extensive computing time and large computer memory. Over time, the progress in computer capabilities diminished the importance of these deficiencies, so that in 1990's a new generation of codes emerged, which extended the use of collision probability method from the lattice cell level to the fuel assembly level. Typical representatives are: DRAGON [5] with selected three-dimensional capabilities, GTRAN2 [6] with anisotropic scattering and parallel processing, and HELIOS [7] that combines the current coupling and collision probability

techniques. With today's computer power it is to be expected that the area of application of integral transport methods can be significantly extended.

WIMS-AECL is a two-dimensional multi-group collision probability code routinely used for lattice calculations of CANDU®-type reactors. The code evolved from the UK code WIMS, a copy of which was distributed to Chalk River Laboratories, AECL, in 1971. Over the years the code has undergone extensive modifications, and today, it is recognized as an independent version of the original UK code. The code capabilities were significantly extended with the development of the version WIMS-AECL 3.1 [8].

In order to speed up calculations, new methods for both collision probability integration and solution of large systems of equations have been developed and presented in this paper. The new methods were implemented in a developmental version of WIMS-AECL, which, for the sake of simplicity will be further referred to as "WIMS". Section 2 reviews the basic mathematical apparatus of the collision probability method and describes the new algorithm for numerical integration. Section 3 presents the new method for solution of large systems of linear algebraic equations generated by the collision probability method. Section 4 describes several test problems and presents the numerical results.

2. Collision probability method

Assuming isotropic neutron scattering, the within-group integral transport equation can be cast in either of the following two equivalent forms:

$$\varphi(\boldsymbol{r}) = \frac{1}{4\pi}\int_V d^3r' \frac{\exp[-\tau(\boldsymbol{r},\boldsymbol{r}')]}{|\boldsymbol{r}-\boldsymbol{r}'|^2}\Big[\,\Sigma_{\mathrm{s}}(\boldsymbol{r}')\,\varphi(\boldsymbol{r}') + S(\boldsymbol{r}')\Big] \tag{1}$$

$$\varphi(\boldsymbol{r}) = \frac{1}{4\pi}\int_{4\pi} d\Omega \int_0^{\infty} ds\,\exp[-\tau(\boldsymbol{r}',\boldsymbol{r})]\Big[\,\Sigma_{\mathrm{s}}(\boldsymbol{r}')\,\varphi(\boldsymbol{r}') + S(\boldsymbol{r}')\Big]\,,\quad \boldsymbol{r}' = \boldsymbol{r} - s\boldsymbol{\Omega} \tag{2}$$

in which, the boundary term is omitted for the sake of simplicity, assuming that the spatial domain V is either infinite or bounded by a boundary of convex shape surrounded by vacuum. The group index is also omitted for the sake of clarity, while the quantities have the following meanings:

$\boldsymbol{r}, \boldsymbol{r}'$ – Position vectors specified by spatial coordinates (x, y, z) and (x', y', z')

$\boldsymbol{\Omega} = \frac{\boldsymbol{r}-\boldsymbol{r}'}{|\boldsymbol{r}-\boldsymbol{r}'|}$ – Neutron path direction specified by azimuth and polar angles (ϕ, θ)

$s = \boldsymbol{r}' - \boldsymbol{r}$ – Distance along the neutron path

$d^3r' = s^2 ds\, d\Omega$ – Elementary volume

$\varphi(\boldsymbol{r})$ – Scalar neutron flux

$S(\boldsymbol{r})$ – Neutron source (fission and neutron scattering from other groups)

$\Sigma_{\mathrm{s}}(\boldsymbol{r})$ – Macroscopic scattering cross section

$\Sigma_{\mathrm{t}}(\boldsymbol{r})$ – Total macroscopic cross section

$\tau(\boldsymbol{r},\boldsymbol{r}') = \int_0^{|r-r'|} \Sigma_{\mathrm{t}}\,(\boldsymbol{r} - s\frac{r-r'}{|r-r'|})ds$ – Optical distance between points $\boldsymbol{r}$ and $\boldsymbol{r}'$

Eqs. (1) and (2) represent a Fredholm-type integral equation of the second kind **[**9]. The kernel is continuous, square-integrable and symmetric. It is worth mentioning that the denominator in the kernel of Eq. (1) might be misleading. In the past, a number of authors considered the transport equation as a singular integral equation. However, Eq. (2) shows that the kernel is continuous and cannot have infinite values.

According to the kernel properties, the integral transport equation has a non-trivial solution for any square-integrable function S that represents the neutron source. The solution belongs to the class of square-integrable functions. In contrast, the solution of the integro-differential transport equation belongs to the class of continuous and differentiable functions. Thus, the main benefit of using the integral equation is that the approximate solution is sought in a much less restrictive class of functions than the class of functions to which the solution of the original problem belongs. Another benefit is that the number of degrees of freedom of the solution (independent variables) is reduced by two degrees, i.e. the angular variables (ϕ, θ) are eliminated by angular integration over the unit sphere. In simple terms, the approximate solution of the integral equation is equivalent to P_n spherical harmonics or S_n discrete ordinates solutions when n tends to infinity.

2.1 Collision probability equations

The class of square-integrable functions allows a large degree of freedom in the choice of trial functions as approximate solution of the transport equation. The simplest form is the step function. Suppose the spatial domain V is subdivided into n subdomains V_i. Regardless the geometric shape, a step function $\omega_i(\boldsymbol{r})$ can be associated to each subdomain, i.e.

$$\omega_i(\boldsymbol{r}) = \begin{cases} 1, & \boldsymbol{r} \in V_i \\ 0, & \boldsymbol{r} \notin V_i \end{cases} \tag{3}$$

The collision probability method was originally formulated by means of physics arguments. From the mathematical point of view, however, it is an application of the Galerkin method of moments. The functions representing the neutron flux $\varphi(\boldsymbol{r})$ and neutron source $S(\boldsymbol{r})$ can be expanded over the basis of step functions $\omega_i(\boldsymbol{r})$ as follows:

$$\varphi(\boldsymbol{r}) = \sum_{i=1}^{n} \bar{\varphi}_i\, \omega_i(\boldsymbol{r})\,, \qquad S(\boldsymbol{r}) = \sum_{i=1}^{n} \bar{S}_i\, \omega_i(\boldsymbol{r}) \tag{4}$$

where the expansion coefficients $\bar{\varphi}_i$ and $\bar{S}_i$ represent region-averaged value of neutron flux and neutron source, respectively, i.e.

$$\bar{\varphi}_i = \frac{1}{V_i}\int_{V_i} \varphi(\boldsymbol{r}) d^3r\,, \qquad \bar{S}_i = \frac{1}{V_i}\int_{V_i} S(\boldsymbol{r}) d^3r \tag{5}$$

Multiplying Eq. (1) or (2) by each basis function $\omega_i(\boldsymbol{r})$ and integrating over the spatial domain V, one gets a system of linear algebraic equations that determines the approximate solution $\bar{\varphi}_i$. Multiplying each equation by the volume and the total cross section of the corresponding subdomain, the system of equations can be expressed in the classical form of collision probability equations:

$$\Sigma_{t,i} V_i \bar{\varphi}_i = \sum_{j=1}^{n} V_j P_{j\to i}\left(\Sigma_{s,j}\bar{\varphi}_j + \bar{S}_j\right)\,, \qquad i = 1,2,\ldots n \tag{6}$$

2.2 Collision probability expression

A matrix coefficient $P_{j\to i}$ represents the probability that a neutron born in region *j* will suffer its next collision in region *i*. Its explicit form is:

$$\begin{aligned} P_{j\to i} &= \frac{\Sigma_{t,i}}{4\pi V_j}\int_{V_i} d^3r \int_{V_j} d^3r'\,\frac{\exp[-\tau(\boldsymbol{r},\boldsymbol{r}')]}{|\boldsymbol{r}-\boldsymbol{r}'|^2} \\ &= \frac{\Sigma_{t,i}}{4\pi V_j}\int_{V_i} d^3r \int_{4\pi} d^2\,\Omega \int_{s_j-\Delta s_j}^{s_j} ds\,\exp[-\tau(\boldsymbol{r},\boldsymbol{r}')] \end{aligned} \qquad \boldsymbol{r}\in V_i,\ \ \boldsymbol{r}'\in V_j \tag{7}$$

where s_j denotes the distance from the point $\boldsymbol{r}$ in region V_i to the exit point of direction $\boldsymbol{\Omega}$ from the region V_j, while Δs_j is the track length size of direction $\boldsymbol{\Omega}$ through the region V_j.

Owing to axial invariability (two-dimensional geometry), the integral over the volume V_i reduces to a double integral over the horizontal cross section of V_i. Also, the geometric distance $\boldsymbol{s} = \boldsymbol{r} - \boldsymbol{r}'$ and the optical distance $\tau(\boldsymbol{r},\boldsymbol{r}')$ can be expressed by their projections *x* and *t* onto the horizontal plane:

$$s = \frac{x - x'}{\sin\theta}\,, \qquad \tau = \frac{t(\boldsymbol{r},\boldsymbol{r}')}{\sin\theta} \tag{8}$$

Assuming that that the *x*-*y* Cartesian coordinate system rotates with the azimuth angle ϕ so that the *x*-axis is parallel to the projection of $\boldsymbol{\Omega}$ onto the horizontal plane, the collision probability takes the form of the following five-fold integral:

$$P_{j\to i} = \frac{\Sigma_{t,i}}{4\pi V_j}\int_0^{2\pi} d\phi \int_{y_{min}}^{y_{max}} dy \int_{x_i-\Delta x_i}^{x_i} dx \int_{x_j'-\Delta x_j'}^{x_j'} dx' \int_0^{\pi} d\theta\, e^{-t(x,x')/\sin\theta} \tag{9}$$

The integration along direction $\boldsymbol{\Omega}$ is carried out over parallel trajectories that intersect both regions V_i and V_j. The limits of y-integration $y_{min} = min(Y_{i,1}, Y_{j,1})$ and $y_{max} = max(Y_{i,2}, Y_{j,2})$ are illustrated in Figure 1. Because of complicated geometric shapes that usually occur, integration with respect to azimuth angle ϕ and Cartesian coordinate y cannot be carried out analytically. Instead, it is evaluated numerically, i.e

$$\int_0^{2\pi} d\phi \int_{y_{min}}^{y_{max}} dy\ F(\phi, y) \approx \sum_{k=1}^{K} w_k \sum_{\ell=1}^{L} w_\ell\, F(\phi_k, y_\ell) \tag{10}$$

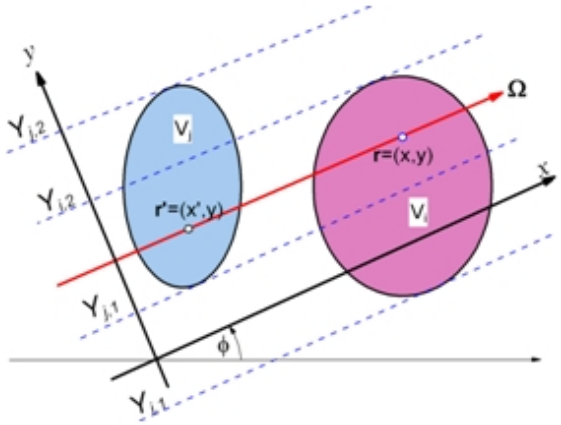

Figure 1. Limits of y-integration in collision probability calculation.

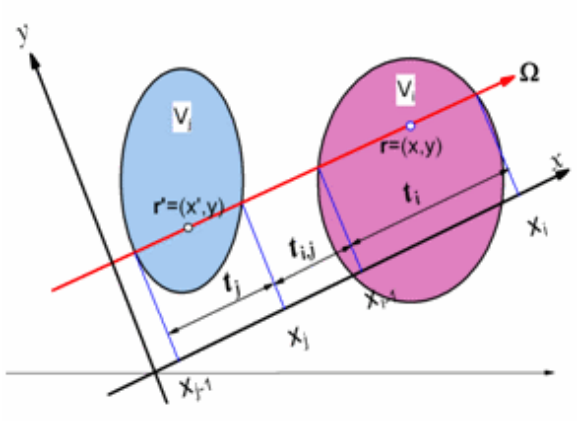

Figure 2. An illustration of analytical integration along neutron trajectory.

The result of integration over the polar angle θ is expressed as the Bickley function of the first order $Ki_1(t)$ [10].

$$\int_0^{\pi/2} d\theta\, e^{-t(x,x')/\sin\theta} = Ki_1(t(x,x')) \tag{11}$$

Thus, by analytical integration of Cartesian coordinates x and x', the collision probability expression takes the following form:

$$P_{j\to i} = \frac{1}{2\pi\Sigma_{t,j}V_j} \sum_{k=1}^{K} w_k \sum_{\ell=1}^{L} w_\ell\, F_{j,i}(\phi_k, y_\ell) \tag{12}$$

where w_k and w_ℓ represent integration weights of the quadrature formulae applied, while the function $F_{j,i}$ represents a linear combination of Bickley functions of third order:

$$F_{j,i} = \begin{cases} \Sigma_{t,i} V_i - [Ki_3(0) - Ki_3(t_i)] , & i = j \\ Ki_3(t_{ij}) - Ki_3(t_{ij} + t_j) - Ki_3(t_{ij} + t_i) + Ki_3(t_{ij} + t_i + t_j) , & i \neq j \end{cases} \quad (13)$$

For a given neutron trajectory, the quantities $t_i = \Sigma_{t,i}\Delta x_i$ and $t_j = \Sigma_{t,j}\Delta x_j$ denote the optical thickness of regions *i* and *j*, while t_{ij} is the minimum optical distance between these two regions (Figure 2).

$$t_{ij} = \sum_{n=i+1}^{j-1} t_n \ , \ \ i < j \ \ \ or \ \ \ t_{ij} = \sum_{n=j+1}^{i-1} t_n \ , \ \ i > j \quad (14)$$

Thus, the contribution to collision probability of a given neutron path trajectory is expressed as two-term ($i = j$) or four-term ($i \neq j$) linear combinations of Bickley functions. Its evaluation is usually the most time consuming part of the collision probability integration algorithm.

2.3 Evaluation of Bickley functions

The Bickley function $Ki_n(t)$ is a special function defined [10] as *n*-times repeated integral of the modified Bessel function of the second kind, $K_0(x)$, i.e.

$$Ki_n(x) = \int_x^\infty Ki_{n-1}(t)dt, \quad n = 1,2,3, \ldots \quad and \quad Ki_0(x) = K_0(x) \quad (15)$$

A general approach to numerical evaluation of special functions is to subdivide the argument range in the following three parts:

- Range of small arguments, in which a convergent series is used. The upper bound of the range is chosen according to the radius of convergence.
- Range of regular (medium size) arguments, in which integral formulae are applied, often coupled with recurrence relations in order to get functions of various order *n*.
- Range of large arguments, in which calculations are carried out using an asymptotic expansion. This range, however, is out of interest here, because, for such arguments the contribution to collision probabilities is negligible.

In order to speed up collision probability codes, the calculation of Bickley functions is usually carried out by polynomial or rational approximations [11]. The lattice code WIMS-AECL uses look-up tables of piecewise polynomials. The code version 2-5d applies a table of quadratic polynomials specified over a set of 2,500 equidistant intervals that cover the range from *x*=0 to *x*=15, while the code version 3.1 uses linear polynomials over a set of 15,000 equal width intervals of length size 0.001.

2.4 Numerical integration instead of Bickley functions calculation

To explain the new method of collision probability calculation, it is useful to present the accuracy of the Gauss quadrature formula [12] applied to Bickley function calculation. Figure 3 presents the absolute value of the relative error in $Ki_3(x)$ evaluation over the range $x \in [0,15]$. It is calculated varying the number of quadrature nodes from 3 to 8. Due to logarithmic scale, sharp minimum values denote argument values at which the sign of the relative error changes.

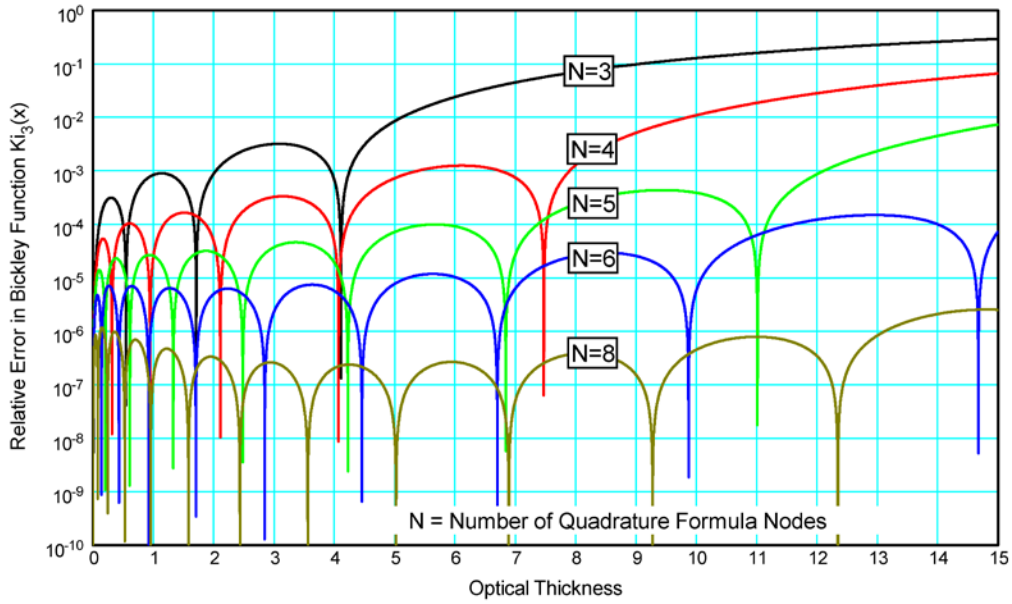

Figure 3. Accuracy of Bickley function calculation using Gauss quadrature formula

The Gauss quadrature formula can be applied directly into Eq. (9) instead of the analytical integration of the polar angle. In this case, the collision probability can be expressed as follows:

$$P_{j\to i} = \frac{1}{2\pi\Sigma_{t,j}V_j}\sum_{k=1}^{K} w_k \sum_{\ell=1}^{L} w_\ell \sum_{m}^{M} w_m \sin^2\theta_m \, G_{j,i}(\phi_k, y_\ell, \theta_m) \tag{16}$$

where θ_m and w_m are abscisas and ordinates of the Gauss quadrature formula, and the function $G_{j,i}$ is obtained by analytical integration along the trajectory specified by the polar angle θ_m

$$G_{j,i} = \begin{cases} \Sigma_{t,i}\Delta x_i - [1-\exp(-\tau_i)]\,, & i = j \\ [1-\exp(-\tau_i)]\cdot\exp(-\tau_{ij})\cdot\left[1-\exp(-\tau_j)\right], & i \neq j \end{cases} \tag{17}$$

One would expect that using numerical integration instead of analytical one, the computing time will increase. As shown in Section 4, however, the effect is opposite for low order quadrature formulae. This is mainly due to the simplicity of Eq. (17). Owing to the properties of the exponential function, the attenuation factor (the trajectory contribution to the transmission probability) between regions i and j can be expressed as a product of attenuation factors of individual regions, i.e.

$$\exp(-\tau_{ij}) = \prod_{n=i+1}^{j-1} \exp(-\tau_n) \tag{18}$$

Increasing the index j, the new attenuation factor is obtained by a single multiplication. On the other hand, the analytical approach requires calculation of four Bickley functions. Using linear look-up tables, the calculation of each function implies two arithmetic operations (one

multiplication and one addition), plus three arithmetic operations to locate the subinterval in the look up table.

For a large optical distance between regions *i* and *j*, the contribution to the collision probability becomes a very small quantity. Thus, the calculation of Eq. (13) is usually truncated when the argument of the Bickley function reaches a cut-off value. In WIMS, it is an input specified parameter limited to 15 neutron mean free paths – the maximum value for which the look-up table is calculated. Instead of Bickley functions, the new method uses exponential functions that are calculated by intrinsic computer procedure. Thus, the cut-off parameter in calculation of Eq. (17) is not limited. In this case, it is more convenient to specify the truncation by the magnitude of the transmission probability, Eq. (18), instead of the optical distance.

Another advantage of the numerical integration is reduced round-off error. The exponential function is evaluated by intrinsic computer procedure, so that the contribution to the collision probability is calculated accurately and always positive. This is not necessarily the case for the analytical approach. The related expression in Eq. (13) represents a sum of two differences. Due to the limited accuracy of the look up tables, a loss of significant figures may occur for optically thin regions. As a result, the round off error accumulates in the sum of the trajectory contributions and impairs the accuracy of collision probability values.

3. Block-iteration of collision probability equations

The collision probability method approximates the neutron transport equation by a set of linear algebraic equations. The overall solution is obtained by a two-level iteration. The outer iteration is carried out by the power iteration method to get the eigenvalue (neutron multiplication factor) and the related fission source. The inner iteration is performed on the thermal groups to determine the thermal scattering source.

3.1 Within-group solution

Dividing each collision probability equation by $\Sigma_{t,i}V_i$, the neutron flux can be expressed by the so-called modified collision probabilities $P^*_{j\to i}$, so that Eq. (6) takes the following form:

$$\bar{\varphi}_i = \sum_{j=1}^{n} P^*_{j\to i}\left(\Sigma_{s,j}\bar{\varphi}_j + \bar{S}_j\right) , \qquad i = 1,2,\dots n. \tag{19}$$

Denote by $\boldsymbol{\varphi}$ and $\boldsymbol{s}$ the vectors of region averaged fluxes $\bar{\varphi}_j$ and sources $\bar{S}_j$, respectively, $\boldsymbol{P}$ is the matrix of modified collision probabilities, $\boldsymbol{\Sigma}_s$ is a diagonal matrix of scattering cross sections, and $\boldsymbol{I}$ is a unity matrix. The matrix of Eq. (19) is dense and the solution is obtained either by matrix decomposition or matrix inversion. To do this, Eq. (19) is rearranged as follows:

$$\boldsymbol{A\varphi} = \boldsymbol{f}, \qquad \boldsymbol{A} = \boldsymbol{I} - \boldsymbol{\Sigma}_s\boldsymbol{P}, \qquad \boldsymbol{f} = \boldsymbol{Ps} \tag{20}$$

The matrix $\boldsymbol{A}$ can be decomposed as a product of lower $\boldsymbol{L}$ and upper $\boldsymbol{U}$ triangular matrices, so that the solution is obtained by forward elimination and backward substitution as follow:

$$\boldsymbol{A = LU}, \qquad \boldsymbol{\xi = Lf}, \qquad \boldsymbol{\varphi = U\xi} \tag{21}$$

where $\boldsymbol{\xi}$ is an auxiliary vector. This procedure is usually used to calculate the within-group flux in fast energy groups. In thermal groups, however, it is necessary to calculate the flux by direct matrix inversion, i.e.

$$\boldsymbol{\varphi = A^{-1}f}, \qquad \boldsymbol{B = LI}, \qquad \boldsymbol{A^{-1} = UB} \tag{22}$$

The number of arithmetic operations of the solution procedures (21) and (22) is proportional to the third degree of the number of unknowns per energy group. Evidently, the computing time increases rapidly with the increase of the number of unknowns.

The total computing time of the collision probability method consists of three components: (1) calculation of collision probabilities, (2) matrix decomposition/inversion, and (3) thermal flux iteration. For small and medium-size problems (below one thousand equations per group), the major consumer of computing time is the calculation of collision probabilities. A new method was presented in Section 2 that, as shown in Section 4, speeds up the calculation by a factor of 2. Increasing the problems size above one thousand equations per group, the dominant time consumer is the matrix inversion. The computing time is proportional to the third degree (n^3) of the number of unknowns n, so that the method becomes inefficient for models that are represented by several thousand equations. This section presents a new solution method for this type of problems, which speeds up the calculation of large cases by at least an order of magnitude.

3.2 Block-iteration method

The vectors of unknown flux $\boldsymbol{\varphi}$ and source values $\boldsymbol{s}$ and, accordingly, the matrix $\boldsymbol{P}$ of modified collision probabilities, can be subdivided into a number m of blocks as follows:

$$\boldsymbol{\varphi} = \begin{vmatrix} \boldsymbol{\varphi}_1 \\ \boldsymbol{\varphi}_2 \\ \vdots \\ \boldsymbol{\varphi}_m \end{vmatrix}, \qquad \boldsymbol{s} = \begin{vmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \vdots \\ \boldsymbol{s}_m \end{vmatrix}, \qquad \boldsymbol{P} = \begin{vmatrix} \boldsymbol{P}_{11} & \boldsymbol{P}_{12} & \cdots & \boldsymbol{P}_{1m} \\ \boldsymbol{P}_{21} & \boldsymbol{P}_{22} & \cdots & \boldsymbol{P}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{P}_{m1} & \boldsymbol{P}_{m2} & \cdots & \boldsymbol{P}_{mm} \end{vmatrix} \tag{23}$$

Suppose that each block has n_i elements so that $n = \sum_{i=1}^{m} n_i$. Eq. (19) can be rewritten now in matrix form as follows:

$$\boldsymbol{\varphi}_i = \sum_{j=1}^{m} \boldsymbol{P}_{ij}\left(\boldsymbol{\Sigma}_{s,j}\boldsymbol{\varphi}_j + \boldsymbol{s}_j\right), \qquad i = 1,2,\ldots m. \tag{24}$$

One may apply block iteration in order to solve Eq. (24). Using the Gauss-Seidel method, the iteration scheme can be cast as follows where ℓ denotes the iteration index:

$$\boldsymbol{\varphi}_i^{(\ell+1)} = \sum_{j=1}^{i} \boldsymbol{P}_{ij}\left(\boldsymbol{\Sigma}_{s,j}\boldsymbol{\varphi}_i^{(\ell+1)} + \boldsymbol{s}_j\right) + \sum_{j=i+1}^{m} \boldsymbol{P}_{ij}\left(\boldsymbol{\Sigma}_{s,j}\boldsymbol{\varphi}_i^{(\ell)} + \boldsymbol{s}_j\right), \qquad i = 1,2,\ldots m. \tag{25}$$

It can be rearranged as follows in order to express the unknown block $\boldsymbol{\varphi}_i^{(\ell+1)}$

$$\boldsymbol{\varphi}_i^{(\ell+1)} - \boldsymbol{P}_{ii}\boldsymbol{\Sigma}_{s,i}\boldsymbol{\varphi}_i^{(\ell+1)} = \sum_{j=1}^{i-1} \boldsymbol{P}_{ij}\left(\boldsymbol{\Sigma}_{s,j}\boldsymbol{\varphi}_i^{(\ell+1)} + \boldsymbol{s}_j\right) + \sum_{j=i+1}^{m} \boldsymbol{P}_{ij}\left(\boldsymbol{\Sigma}_{s,j}\boldsymbol{\varphi}_i^{(\ell)} + \boldsymbol{s}_j\right) \tag{26}$$

Inverting the matrix on the left hand side of Eq. (26), the iteration scheme takes the following form:

$$\boldsymbol{\varphi}_i^{(\ell+1)} = \sum_{j=1}^{i-1} \boldsymbol{A}_{ii}^{-1}\boldsymbol{P}_{ij}\left(\boldsymbol{\Sigma}_{s,j}\boldsymbol{\varphi}_i^{(\ell+1)} + \boldsymbol{s}_j\right) + \sum_{j=i+1}^{m} \boldsymbol{A}_{ii}^{-1}\boldsymbol{P}_{ij}\left(\boldsymbol{\Sigma}_{s,j}\boldsymbol{\varphi}_i^{(\ell)} + \boldsymbol{s}_j\right) \tag{27}$$

where

$$\boldsymbol{A}_{ii}^{-1} = \left(\boldsymbol{I} - \boldsymbol{P}_{ii}\boldsymbol{\Sigma}_{s,i}\right)^{-1} \tag{28}$$

Instead of inverting the full matrix $\boldsymbol{A}$ of dimensions $n \times n$, the above scheme requires inversion of m matrices $\boldsymbol{A}_{ii}$ of much lower dimensionality. Another advantage is the significant reduction in memory requirements. Matrix blocks can be stored in a temporary file and retrieved in the memory one-by-one as necessary.

4. Numerical results

4.1 Test problems

In order to study the properties of the new methods, the following three sets of problems are considered:

A. CANDU-6 lattice cell with fresh fuel.
   1. Regular lattice cell at normal operating condition (Figure 4).
   2. Lattice cell with destroyed bundle in a crept pressure tube (Figure 5).

B. Defueled channel in a periodically repeating lattice region.
   1. Periodically repeating lattice region of 2x2 cells (Figure 6).
   2. Periodically repeating lattice region of 3x3 cells.
   3. Periodically repeating lattice region of 4x4 cells.
   4. Periodically repeating lattice region of 5x5 cells.

C. Core-reflector interface.
   1. Model of 3+2 cells (3 fuel cells + 2 reflector cells) (Figure 7).
   2. Model of 4+2 cells.
   3. Model of 5+2 cells.
   4. Model of 6+2 cells.

Figure 4 shows the geometric model of the regular lattice cell along with the mesh subdivision that is used for collision probability calculations. Because of the symmetry, the number of

unknowns per group reduces to the number of meshes in one quadrant of the lattice cell, which is in this case equal to 142 unknowns per group.

Figure 5 presents the geometric model of the lattice cell with destroyed fuel bundle in a sagged pressure tube, as used usually in safety analysis. In this case, there is no symmetry in the model. Moreover, due to the asymmetry of the pressure tube and fuel pins, additional mesh refinement should be applied to the pressure tube interior. As a result, the problem is represented by 825 unknowns per group.

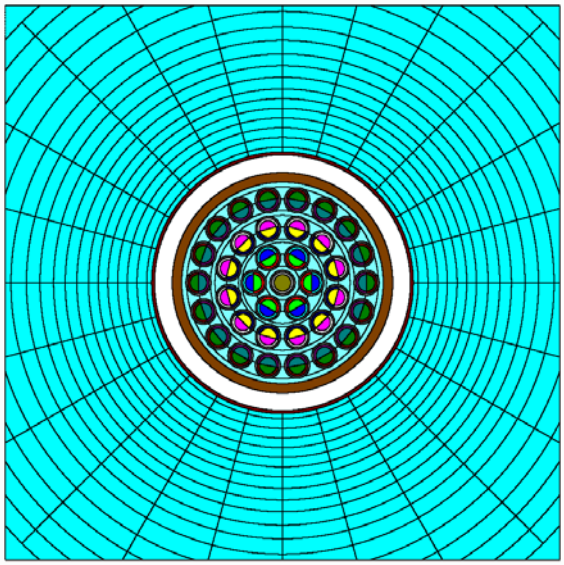

Figure 4. Geometric model and mesh subdivision of the regular lattice cell.

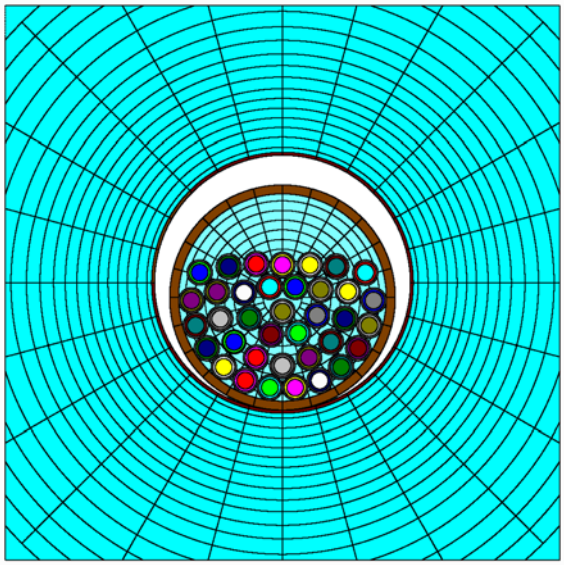

Figure 5. Lattice cell with destroyed fuel bundle in a sagged pressure tube..

Figure 6 shows the geometric model of the defueled channel problem in 2×2 cells environment, while Figure 7 represents the geometric model of a core-reflector interface problem with three fueled cells and two reflector cells (3+2 cells). In related models, only the number of regular lattice cells is greater; there is again one defueled cell or two reflector cells.

4.2 Collision probability calculation

As presented in Section 2, there are two free parameters in the new method for collision probability calculation. They are: the number of quadrature nodes for polar angle integration and the truncation (cut-off) criterion in the calculation of the trajectory contribution to collision probability. Calculations of test problems A.1 and A.2 were carried out varying these parameters as follows: the number of quadrature nodes varies from 2 to 8, while truncation parameter varies over the following set of values: 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, and 10^{-8}. Reference calculations were performed using 32 quadrature nodes and truncation parameter of 10^{-18}. Table 1 and Table 2 present the results of calculations obtained for both coolant states (cooled and voided) of the test problems A.1 and A.2, respectively. For each parameter value (the number of nodes and the cut-off parameter), the following results are given: the infinite neutron multiplication factor (k-inf),

the absolute error Δk-inf expressed in pcm = 0.01 mk, the root mean square (RMS) error (%) in neutron flux distribution, and the computing time for calculation of collision probabilities. Calculations were carried out on a Hewlett Packard PC with Intel Core2, a 32-bit processor, at 3 GHz clock. The results show that for the cooled lattice state even the very coarse approximation (2 nodes and cut-off value of 10^{-4}) produces rather low errors (20 pcm = 0.2 mk in k-infinity and less than 1% in flux distribution). However, for the voided coolant state the flux error of the low order approximation rises to about 15%. Thus, 3 quadrature nodes and a cut-off value of 10^{-6} seem to be a reasonable choice for routine calculations. The error in neutron multiplication factor is within ±2 pcm, the RMS error in flux distribution is below half a percent, while the computing time is decreased by more than a factor of 2

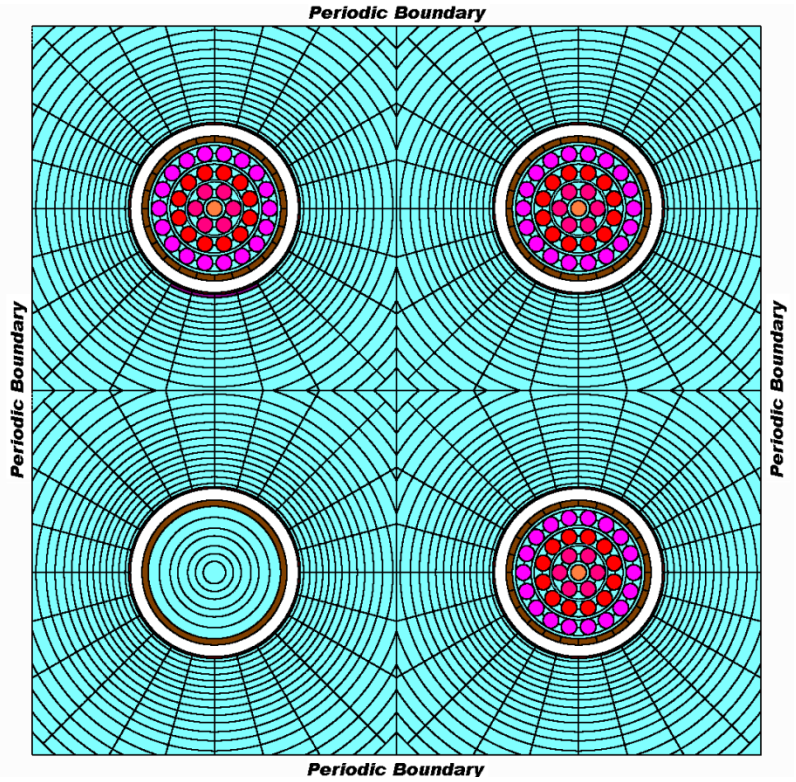

Figure 6. Geometric model of defueled channel problem represented by 2x2 lattice cells.

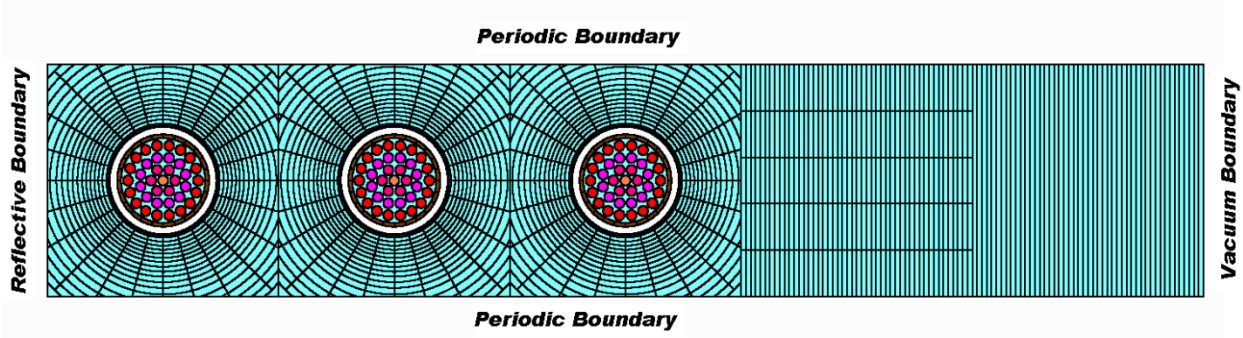

Figure 7. Geometric model of core-reflector interface problem represented by three fuel cells and two reflector cells

Table 1
Results of calculation of the regular lattice cell

Number of Nodes	Cut Off Value	Cooled				Voided			
		k-inf	Δk-inf (pcm)	RMS (%)	CPU (s)	k-inf	Δk-inf (pcm)	RMS (%)	CPU (s)
2	1.E-4	1.11467	19	0.79	4.7	1.13585	33	3.8	4.7
	1.E-5	1.11465	17	0.32	5.3	1.13575	23	0.60	5.3
	1.E-6	1.11465	17	0.33	5.9	1.13574	22	0.35	5.9
	1.E-7	1.11465	17	0.33	6.5	1.13574	22	0.36	6.5
	1.E-8	1.11465	17	0.33	7.0	1.13573	21	0.36	7.1
3	1.E-4	1.11450	2	0.77	6.1	1.13559	7	3.9	6.1
	1.E-5	1.11448	0	0.061	7.0	1.13550	-2	0.55	7.1
	1.E-6	1.11447	1	0.043	7.9	1.13550	-2	0.076	7.9
	1.E-7	1.11448	0	0.043	8.7	1.13550	-2	0.050	8.8
	1.E-8	1.11448	0	0.043	9.7	1.13550	-2	0.05	9.6
4	1.E-4	1.11451	3	0.77	7.7	1.13562	10	3.9	7.5
	1.E-5	1.11449	1	0.046	8.8	1.13552	0	0.55	8.8
	1.E-6	1.11449	1	0.011	9.9	1.13552	1	0.059	9.9
	1.E-7	1.11449	1	0.010	10.9	1.13552	0	0.012	11.1
	1.E-8	1.11450	2	0.010	11.9	1.13551	-1	0.012	12.1
5	1.E-4	1.11451	3	0.76	9.0	1.13562	10	3.9	8.9
	1.E-5	1.11448	0	0.044	10.6	1.13552	0	0.55	10.5
	1.E-6	1.11448	0	0.005	11.9	1.13551	-1	0.058	12.0
	1.E-7	1.11449	1	0.004	13.2	1.13551	-1	0.006	13.4
	1.E-8	1.11449	1	0.004	14.4	1.13551	-1	0.003	14.6
6	1.E-4	1.11451	3	0.77	10.5	1.13562	10	3.89	10.5
	1.E-5	1.11448	0	0.044	12.2	1.13553	1	0.55	12.3
	1.E-6	1.11449	1	0.004	13.9	1.13552	0	0.058	14.0
	1.E-7	1.11449	1	0.002	15.4	1.13551	-1	0.005	15.6
	1.E-8	1.11448	0	0.002	16.9	1.13552	0	0.002	17.1
7	1.E-4	1.11450	2	0.76	12.0	1.13563	11	3.9	11.9
	1.E-5	1.11449	1	0.044	14.0	1.13552	0	0.55	14.0
	1.E-6	1.11448	0	0.003	16.1	1.13552	0	0.058	16.1
	1.E-7	1.11449	1	0.001	17.7	1.13551	-1	0.005	17.9
	1.E-8	1.11448	0	0.001	19.4	1.13552	0	0.001	19.7
8	1.E-4	1.11449	1	0.77	13.5	1.13562	10	3.9	13.4
	1.E-5	1.11448	-0	0.04	15.7	1.13552	0	0.55	15.8
	1.E-6	1.11449	1	0.003	17.9	1.13551	-1	0.058	18.1
	1.E-7	1.11448	0	0.001	19.9	1.13551	-1	0.005	20.2
	1.E-8	1.11448	0	0.001	21.8	1.13552	0	0.001	22.9
32	1.E-18	**1.11448**	**Reference**		122.3	**1.13552**	**Reference**		124.0
Bickley func.		1.11448	0	0.001	19.5	1.13541	-11	0.003	20.1

Table 2
Results of calculation of the lattice cell with destroyed fuel bundle

Number of Nodes	Cut Off Value	Cooled				Voided			
		k-inf	Δk-inf (pcm)	RMS (%)	CPU (s)	k-inf	Δk-inf (pcm)	RMS (%)	CPU (s)
2	1.E-4	1.11835	17	0.44	7.2	1.13708	31	14.4	7.0
	1.E-5	1.11837	19	0.31	8.0	1.13699	22	3.81	7.9
	1.E-6	1.11838	20	0.32	8.8	1.13698	21	0.62	8.8
	1.E-7	1.11837	19	0.32	9.5	1.13699	22	0.32	9.6
	1.E-8	1.11837	19	0.32	10.2	1.13697	20	0.32	10.3
3	1.E-4	1.11816	-2	0.38	9.1	1.13685	8	14.6	8.9
	1.E-5	1.11817	-1	0.05	10.3	1.13676	-1	3.87	10.2
	1.E-6	1.11816	-2	0.04	11.3	1.13676	-1	0.57	11.4
	1.E-7	1.11818	0	0.04	12.4	1.13675	-2	0.07	12.5
	1.E-8	1.11817	-1	0.04	13.3	1.13675	-2	0.04	13.5
4	1.E-4	1.11818	0	0.38	11.0	1.13687	10	14.6	10.9
	1.E-5	1.11818	0	0.03	12.6	1.13679	2	3.85	12.5
	1.E-6	1.11817	-1	0.01	14.0	1.13678	1	0.57	14.1
	1.E-7	1.11819	1	0.01	15.3	1.13678	1	0.06	15.4
	1.E-8	1.11818	0	0.01	16.5	1.13678	1	0.01	16.8
5	1.E-4	1.11818	0	0.38	13.0	1.13688	11	14.6	12.8
	1.E-5	1.11818	0	0.03	14.7	1.13678	1	3.87	14.8
	1.E-6	1.11818	0	0.01	16.6	1.13678	1	0.57	16.7
	1.E-7	1.11817	-1	0.01	18.2	1.13678	1	0.06	18.4
	1.E-8	1.11818	0	0.01	19.9	1.13678	1	0.01	20.1
6	1.E-4	1.11816	-2	0.38	14.9	1.13688	11	14.6	14.6
	1.E-5	1.11818	0	0.03	17.1	1.13679	2	3.86	17.0
	1.E-6	1.11816	-2	0.00	19.3	1.13678	1	0.57	19.3
	1.E-7	1.11817	-1	0.00	21.1	1.13678	1	0.06	21.4
	1.E-8	1.11816	-2	0.00	23.1	1.13678	1	0.01	23.3
7	1.E-4	1.11815	-3	0.38	16.9	1.13687	10	14.6	16.5
	1.E-5	1.11818	0	0.03	19.4	1.13679	2	3.86	19.3
	1.E-6	1.11818	0	0.00	21.7	1.13677	0	0.57	21.9
	1.E-7	1.11817	-1	0.00	24.1	1.13678	1	0.06	24.3
	1.E-8	1.11817	-1	0.00	26.2	1.13678	1	0.01	26.5
8	1.E-4	1.11815	-3	0.38	18.7	1.13687	10	14.6	18.4
	1.E-5	1.11818	0	0.03	21.7	1.13678	1	3.86	21.6
	1.E-6	1.11817	-1	0.00	24.4	1.13678	1	0.56	24.7
	1.E-7	1.11818	0	0.00	27.0	1.13678	1	0.06	27.2
	1.E-8	1.11818	0	0.00	29.5	1.13677	0	0.01	29.7
32	1.E-18	**1.11818**	**Reference**		157.5	**1.13677**	**Reference**		159.5
Bickley function		1.11815	-3	0.00	25.4	1.13668	-11	0.08	26.2

4.3 Solution of large systems of collision probability equations

In order to study the efficiency of the block-iteration method presented in Section 3, calculations were carried out by varying the block size of matrix partition from 5, 10, 20, … unknowns per block up to the full size matrix. Figure 8 presents the computing time of matrix inversion as a function of the block size. The right-most point of each curve represents the full matrix case. It is evident that as the block size increases, the computing time increases rapidly. The time ratio of the full matrix case versus small blocks of 5 to 20 unknowns ranges from 100 for the model A.2 to 1000 for model C.

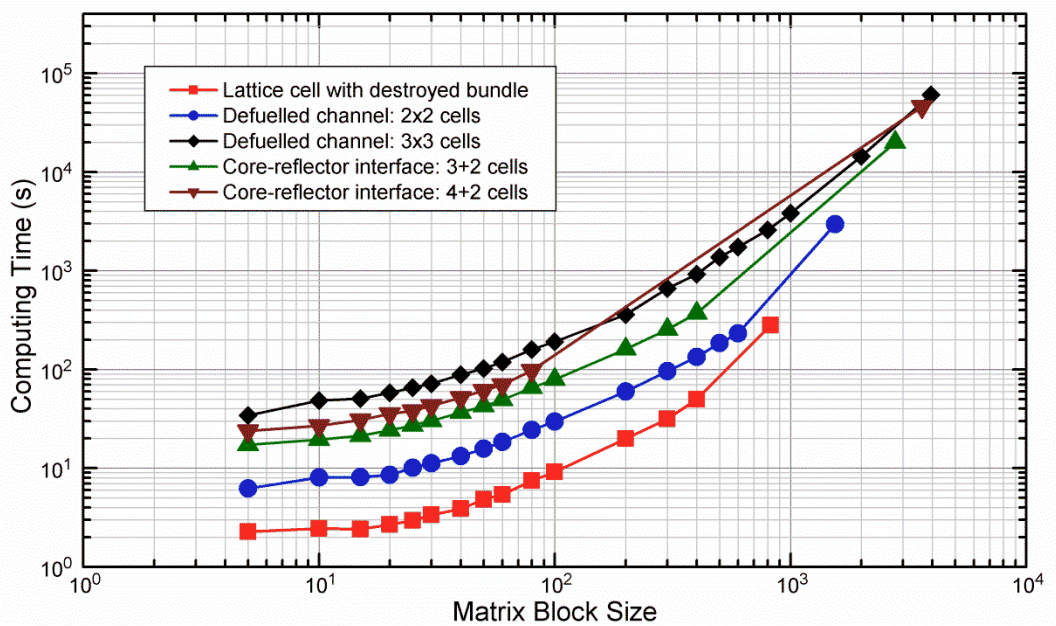

Figure 8. CPU time of matrix inversion as a function of block size

Figure 9 presents the iteration computing time as a function of the block size. A slight increase of the CPU time with the decrease of the block size can be observed for the model A. However, for other models, the dependence on the block size is weak with a slight minimum in the block size range of 20 to 100 unknowns.

Figure 10 presents the total computing time as a function of the block size. For each curve, a flat region can be observed in the block size range of 20 to 100 unknowns. Compared to the full matrix results, the related computing time is decreased by a factor 10 to 80, depending on the problem considered.

According to the presented results, the block size of 20 unknowns seems to be a reasonable choice as the default value.

Table 3 summarizes the calculation of large collision probability problems. As a check of the validity, results of MCNP calculations are included as well. The results of both full-matrix solution and block-matrix solution are given for models represented by up to 4000 unknowns. When the number of unknowns exceeds this value, the memory requirements of the full-matrix

method exceed the 2 GB indexing limit on a 32 bit computer so that such problems could not be solved. The new solution method speeds up the calculation significantly and allows solution of large problems the dimensions of which may go far above 4000 unknowns per group. On the other hand, the comparison with MCNP shows a very good agreement between the two codes.

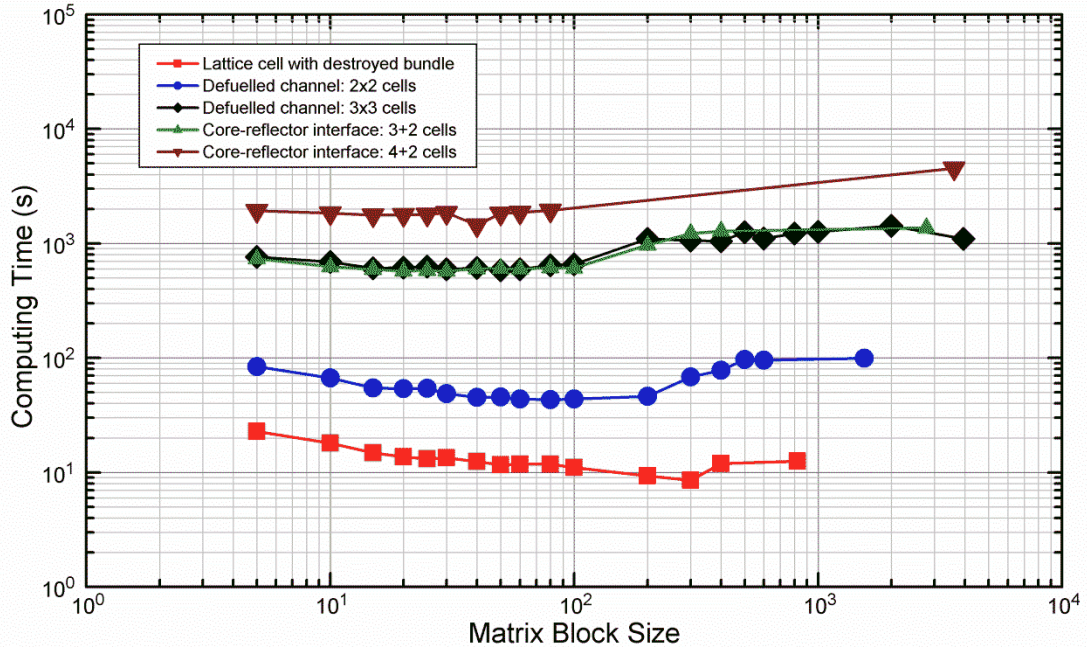

Figure 9. Iteration CPU time as a function of block size.

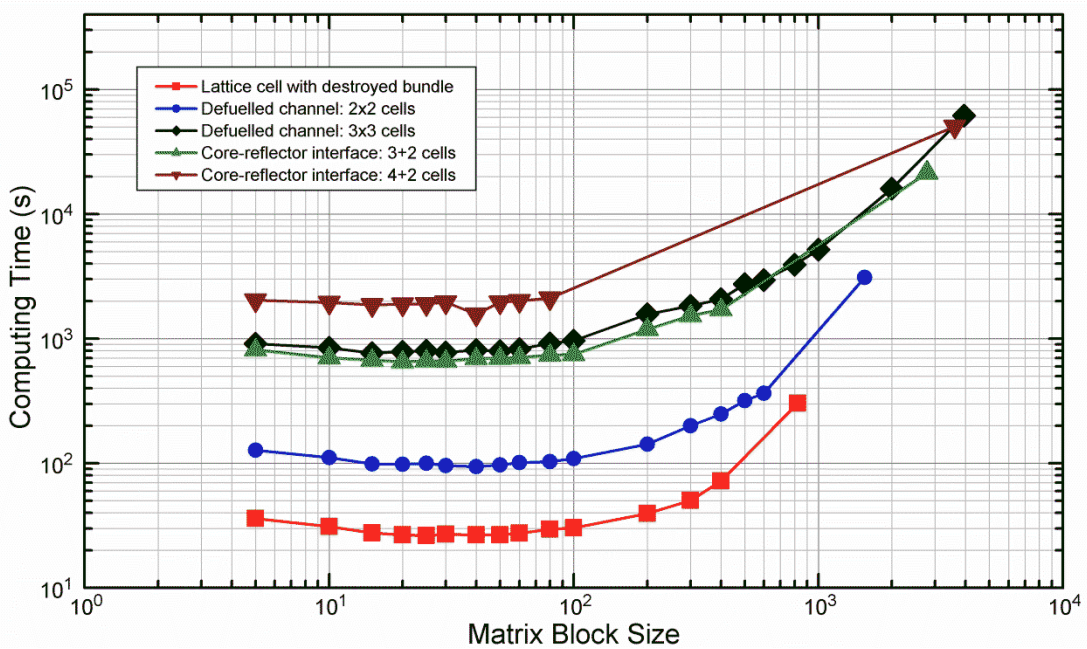

Figure 10. Total CPU time as a function of block size

Table 3
Summary of large problem calculations.

Model	Code	Number of unknowns	Matrix	k-infinity	CPU (min)	Speed-up factor
A.2	WIMS	825	Full	1.11809	5.03	
			Block	1.11813	0.45	11
	MCNP			1.11769		
B: 2x2	WIMS	1547	Full	1.09814	51.60	
			Block	1.09815	1.63	32
	MCNP			1.09771		
B: 3x3	WIMS	3942	Full	1.10799	1029.20	
			Block	1.10804	13.05	79
	MCNP			1.10801		
B: 4x4	WIMS	7295	Block	1.11096	63.22	NA
	MCNP			1.11055		
B: 5x5	WIMS	11606	Block	1.11217	259.33	NA
	MCNP			1.11191		
C: 3+2	WIMS	2781	Full	1.06366	356.83	
			Block	1.06362	10.88	33
	MCNP			1.06387		
C: 4+2	WIMS	3612	Full	1.07933	845.45	
			Block	1.07933	31.43	27
	MCNP			1.07985		
C: 5+2	WIMS	4443	Block	1.08846	60.07	NA
	MCNP			1.08926		
C: 6+2	WIMS	5274	Block	1.09431	155.63	NA
	MCNP			1.09508		

NA – Full-matrix method exceeds 2GB indexing limit of 32-bit machine.

5. Conclusions

A new method for collision probability calculation is presented that applies numerical integration of the polar angle instead of analytical integration of the Bickley functions. The algorithm is flexible concerning the accuracy and eliminates round-off error that may occur in standard collision probability calculations. Test results show that compared to the standard approach, the computing time can be reduced by a factor 2.

For a direct method of solution of within-group collision probability equations, the number of unknowns is a limiting factor with respect to both, memory requirements and computing time. As presented in this paper, the block iteration method reduces significantly the memory requirements and speeds up the calculations by more than one order of magnitude. Another advantage is that the block-partition approach is suitable for parallel processing. Calculations of each diagonal block can be carried out on a separate processor. In that case, however, the Jacobi iteration should be used instead of Gauss-Seidel iteration.

5. Acknowledgements

Appreciation is extended to B. Bromley, R. Davis and A. Trottier who reviewed this paper and provided valuable comments.

6. References

[1] H.C. Honeck, "THERMOS, A thermalization transport theory code for reactor lattice calculations", BNL-5826, 1961.

[2] J.R. Askew, F.E. Fayers, and P.B. Kemshell, "A general description of the lattice code WIMS', *J. British Nucl. Energy Soc.*, 5 (1966) 564.

[3] K. Kobayashi and H. Nishihara, "The solution of the integral transport equations in cylindrical geometry using the Gaussian quadrature formula", *J. Nucl. Energy*, A/B 18 (1964) 513-522.

[4] T. Bosevski, "An improved collision probability method for thermal neutron flux calculation in a cylindrical reactor cell", *Nucl, Sci. Eng.*, 42 (1970) 23.

[5] R. Roy, A. Hebert, and G. Marleau, "A transport method for treating three-dimensional lattices of heterogeneous cells", *Nucl. Sci. Eng*., 101 (1989) 217.

[6] J.L. Vujic and W.R. Martin, "Two-dimensional collision probability method with anisotropic scattering for vector and parallel processing", *Proc. Int. Conf. Physics of Reactors: Operation, Design and Computation*, Marseille, France, April 28-27, 1990, Vol. 3. p. P.IV-104.

[7] Villarino, E.A., Stamm'ler, R.J.J., Ferri, A.A., Casal, J.J., "HELIOS: Angularly dependent collision probabilities", *Nucl. Sci. & Eng.*, 112, pp. 16-31, 1992.

[8] D. Altiparmakov, "New capabilities of the lattice code WIMS-AECL", *Proceedings of PHYSOR 2008, the International Conference on Reactor Physics, Nuclear Power: A Sustainable Resource*, Interlaken, Switzerland, September 14-19, 2008.

[9] G.A. Korn and T.M. Korn, "*Mathematical handbook for scientists and engineers*", McGraw-Hill, New York, 1968.

[10] W.G. Bickley, "Some solutions of the problem of forced convection", *Philos. Mag*., 20 (1935) 322-343.

[11] J.M. Blair, C.A. Edwards and J.H. Johnson, "Rational Chebyshev approximations for the Bickley functions $Ki_n(x)$", *Mathematics of Computation*, 32, 143 (1978) 876-886.

[12] M. Abramowitz and I.A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, US Department of Commerce, National Bureau of Standards, 1970.