AAEC/E192

AAEC/E192

8

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

SUBROUTINE SOK - ITERATIVE SOLUTION OF LINEAR EQUATIONS BY THE METHOD OF AVERAGING FUNCTIONAL CORRECTIONS

by

J.P. POLLARD

August 1968

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

SUBROUTINE SOK – ITERATIVE SOLUTION OF LINEAR EQUATIONS BY THE METHOD OF AVERAGING FUNCTIONAL CORRECTIONS

by

J. P. POLLARD

ABSTRACT

The subroutine SOK solves a set of N simultaneous linear equations by an essentially iterative method. For the method to converge at a reasonable rate (or at all) the user must choose $K(\leqslant N)$, the order of subsidiary equations which are to be obtained from the N given equations. The matrix of coefficients of the K subsidiary equations is inverted using the direct method of Gauss-Jordan. The method is most effective on large sparse matrices that have dominant diagonal terms and for this situation it should be possible to choose K a lot less than N.

The method is most advantageous compared to other iterative methods when the trial investigation of typical matrices is worthwhile. For large matrix problems, details are given of a possible compact matrix storage arrangement. For very large matrices, which even when compacted cannot fit in core, the solution procedure is feasible, provided the matrix is available from disk or tape a column at a time.

The subroutine is written in FORTRAN for the IBM 360/50 computer.

CONTENTS

1. INTRODUCTION

Many problems of physics and engineering require solution of linear equations. In the field of numerical solution of differential and integral equations arising from reactor physics problems, matrices of coefficients appear that have dominant diagonal terms. Much of the literature is concerned with iterative solution of these equations using say the method of successive over relaxation (Wachspress 1966). Here we are concerned with the Sokolov method, the method of averaging functional corrections (Luchka 1965), which does not require diagonal terms to dominate in the matrix of coefficients in order for the method to converge, although the speed of convergence depends on this property.

The basis of the Sokolov method is completely different from other iterative methods and a knowledge of the method does not appear to be widespread. For this reason the work reported here was undertaken and a FORTRAN subroutine SOK was written.

The Sokolov method requires the selection of two sets of base vectors ψ_k and ϕ_k, $k = 1,2,\ldots,K$, which are used in a moment approximation (Kantorovich and Akilov 1964) to differences of solution of the problem from one iteration to the next. In the subroutine SOK the choice of base vectors and their order K is the responsibility of the user. A basis for the selection of these vectors for a "one-off" calculation is not clear, but when a recurring type of problem is being tackled, an initial trial investigation with typical problems does not seem unreasonable. We would then seek the base vectors which keep computer time as small as possible. In a sense the user can develop SOK into a subroutine which is "tailor-made" for a specific type of problem.

Let A be the matrix of real coefficients a_{ij} $(i,j = 1,2,\ldots,N)$, c the column vector of real constants c_i $(i = 1,2,\ldots,N)$ and x the column vector of unknowns x_j $(j = 1,2,\ldots,N)$; then the equations to be solved may be written

$$Ax = c \quad , \tag{1}$$

or

$$\sum_{j=1}^{N} a_{ij} x_j = c_i \quad , \qquad i = 1,2,\ldots,N \quad .$$

Decomposing A into a non-singular diagonal matrix D, not necessarily the matrix obtained from the diagonal terms of A, and a matrix T such that

$$A = D + T \quad , \tag{2}$$

then Equation 1 may be written

$$Dx = c - Tx \quad , \tag{3}$$

or

$$x = D^{-1} c - D^{-1} Tx \quad . \tag{4}$$

Using a superscript n to designate the iteration cycle, the Jacobi method of successive approximation amounts to using the iterative scheme

$$Dx^{(n)} = c - Tx^{(n-1)}, \qquad n = 1,2,\ldots, \tag{5}$$

where $x^{(o)}$ is a trial solution, say, a vector of all zero elements. A simple extension of this method is the Gauss-Seidel method which uses the scheme

$$Dx^{(n)} = c - T_U x^{(n-1)} - T_L x^{(n)} \quad , \qquad n = 1,2,\ldots, \tag{6}$$

where T_U and T_L respectively denote upper and lower triangular matrices obtained from T with the diagonal terms of T included in T_L. The Sokolov method, as pursued here, uses a combination of the Jacobi and Gauss-Seidel methods as the basis for an iterative procedure and uses the Gauss-Jordan direct method for solution of an associated moments approximation of order K ($<<N$). Convergence of Equation 5 is improved by the addition of an easily determined moments approximation to $(x^{(n)} - x^{(n-1)})$ appearing on the right hand side. With a particular choice of base vectors ψ_k and ϕ_k $(k = 1,2,\ldots,K)$ for some partitions of the matrix A (Section 2.3.1). Equations 5 or 6 may be used without further modification.

2. MATHEMATICAL DETAILS

2.1 The Sokolov Method

The equation to be solved, Equation 3, is

$$Dx = c - Tx \quad .$$

Here we introduce the Sokolov method in the following way. Let U be an NXN matrix as yet unprescribed; then we write instead of the Jacobi scheme, Equation 5, the following iterative scheme:

$$Dx^{(n)} = c - Tx^{(n-1)} - U(x^{(n)} - x^{(n-1)}) \quad , \quad n = 1,2,\ldots \tag{7}$$

If we choose U=T the solution of Equation 7 for $x^{(n)}$ is the same as the solution of Equation 3 for x. Of course we are no better off as the solution of either equation involves a method as yet not forthcoming. Let us suggest however that for this choice of U we solve Equation 7 by a direct method such as the Gauss-Jordan method as implemented in the subroutine SID (Davids 1967). On the other hand if we choose U=0 we obtain the Jacobi method and if we choose $U=T_L$ we obtain the Gauss-Seidel method. What we require in general is a selection of U which in some sense is between the two extremes U=0 (or $U=T_L$) and U=T.

In Section 2.2 we will investigate projection matrices P_K, $K=0,1,\ldots,N$, which form a non-unique set

$$P_0\ (=0),\ P_1, P_2, \ldots,\ P_{N-1},\ P_N\ (=I) \quad , \tag{8}$$

where 0 and I denote the zero and unit N×N matrices respectively. Let us associate a diagonal matrix H_K with P_K which contains elements 0 or 1 along the diagonal (Section 2.3.1) and which is such that

$$P_K H_K = 0 \quad . \tag{9}$$

This selection of U made in this work is

$$U_K = TD^{-1} P_K D + H_K T_L \quad , \tag{10}$$

where a subscript has been appended to U to remind us that a set of possibilities exists. The user of the SOK subroutine is required to make a choice from the set given as Equation 10 through a selection of the diagonal matrix D, the projection matrix P_K and the matrix associated with the projection matrix H_K.

Using Equation 10 and the following definitions:

$$\delta^{(n)} = x^{(n)} - x^{(n-1)} \tag{11}$$

$$\alpha^{(n)} = D^{-1} P_K D\, \delta^{(n)} \quad , \tag{12}$$

we obtain the Sokolov iterative scheme

$$Dx^{(n)} = c - T(x^{(n-1)} + \alpha^{(n)}) - H_K T_L \delta^{(n)}, \quad n = 1,2,\ldots \tag{13}$$

Discussion of the method for determining $\alpha^{(n)}$ is now postponed until Section 2.4 to allow a closer look at the projection matrices, P_K.

2.2 The Projection Matrices, P_K

A matrix P is called a projection (Halmos 1958) if

$$P = P^2 \quad . \tag{14}$$

Obviously P=0 and P=I are projections and there are many others.

To understand the role of projection matrices we digress slightly. Our vectors, x, may be thought of as elements of an N-dimensional linear space $\mathcal{E}_N$ (Halmos ibid.). Every $x \in \mathcal{E}_N$ may be expressed in the following form

$$x = \sum_{i=1}^{N} x_i e_i \ , \tag{15}$$

where e_i denotes the i^{th} column of the NXN unit matrix. Given two vectors x and $y \in \mathcal{E}_N$ we may define a scalar product of x and y as

$$(x,y) = \sum_{i=1}^{N} x_i y_i \ , \tag{16}$$

and an associated norm of a vector x as

$$\|x\| = (x,x)^{1/2} \ . \tag{17}$$

Two vectors x and y are said to be orthogonal if $(x,y) = 0$.

The user of the SOK subroutine is required to provide information, detailed in Section 2.3.1, from which is generated two sets of K bi-orthogonal vectors of dimension N, ψ_k and ϕ_k with $(\psi_i,\phi_k) = 0$ if $i \neq k$, $k = 1,2,\ldots,K$. The sets of vectors $\{\psi_k\}$ and $\{\phi_k\}$ are used as bases for K-dimensional subspaces $\tilde{\mathcal{E}}_K$ and $\bar{\mathcal{E}}_K$ of $\mathcal{E}_N$. Every $x \in \tilde{\mathcal{E}}_K$ and $y \in \bar{\mathcal{E}}_K$ may be written

$$x = \sum_{k=1}^{K} \tilde{x}_k \psi_k \text{ and } y = \sum_{k=1}^{K} \bar{y}_k \phi_k \ , \tag{18}$$

where $\tilde{x}_k$ and $\bar{y}_k$ are respectively the components of x and y relative to the bases.

The projection matrices P_K we require in this work are to be such that for every $x \in \mathcal{E}_N$ we have $P_K^T x \in \tilde{\mathcal{E}}_K$ and $P_K x \in \bar{\mathcal{E}}_K$, where the subscript T denotes the transpose. We must thus be able to write

$$P_K^T x = \sum_{k=1}^{K} \tilde{x}_k \psi_k \text{ and } P_K x = \sum_{k=1}^{K} \bar{x}_k \phi_k \ , \tag{19}$$

where $\tilde{x}_k = (P_K^T x, \phi_k)/(\psi_k,\phi_k)$ and $\bar{x}_k = (\psi_k, P_K x)/(\psi_k,\phi_k)$.

Since $\psi_k \in \tilde{\mathcal{E}}_K$ and $\phi_k \in \bar{\mathcal{E}}_K$, and we have

$$(P_K^T x, \phi_k) = (x, P_K \phi_k) = (x, \phi_k) \ ,$$

with a similar result for the scalar product $(\psi_k, P_K x)$, then the components simplify to

$$\tilde{x}_k = (x,\phi_k)/(\psi_k,\phi_k) \quad \text{and} \quad \bar{x}_k = (\psi_k, x)/(\psi_k,\phi_k) \ . \tag{20}$$

As a simple example, say we have the following:

$$N=3, K=2, \ \psi_1 = (1,1,0)^T, \ \psi_2 = (0,0,1)^T, \ \phi_1 = (2,3,0)^T \text{ and } \phi_2 = (0,0,4)^T \ ,$$

then $\tilde{x}_1 = \frac{2}{5} x_1 + \frac{3}{5} x_2$, $\tilde{x}_2 = x_3$, $\bar{x}_1 = \frac{1}{5} x_1 + \frac{1}{5} x_2$ and $\bar{x}_2 = \frac{1}{4} x_3$. We thus identify the matrix P_2 as

$$P_2 = \begin{pmatrix} \frac{2}{5} & \frac{2}{5} & 0 \\ \frac{3}{5} & \frac{3}{5} & 0 \\ 0 & 0 & 1 \end{pmatrix} \ .$$

In general the identification is not required, as P_K does not appear explicitly in the computational procedures to follow, but rather components $\bar{x}_k$.

2.3 The Base Vectors for $\tilde{\mathcal{E}}_K$ and $\bar{\mathcal{E}}_K$

The subroutine SOK only allows a simple choice to be made for the base vectors $\{\psi_k\}$ and $\{\phi_k\}$. Considering, in general, that we are unlikely to have available any information which could be used to advantage, such as the eigenvectors corresponding to the largest eigenvalue of A and A^T (should they be real), the desire to minimize computer time suggests the use of single vectors containing elements of 0 and 1 where possible.

2.3.1 The base vectors $\{\psi_k\}$

The user of the SOK subroutine is required to provide a vector m of K' integers, K of which are positive, and which satisfy

$$|m_1| + |m_2| + \ldots + |m_{K'}| = N . \tag{21}$$

From the vector m, K base vectors $\{\psi_k\}$ are generated in such a way that the elements of a base vector consist of either 0 or 1 with the unit elements all grouped together. A simple example best illustrates the way the base vectors are described. Say N=10, K=3 and we require the base vectors

$$\psi_1 = (0,0,1,1,1,0,0,0,0,0)^T,$$
$$\psi_2 = (0,0,0,0,0,0,0,1,0,0)^T,$$
$$\psi_3 = (0,0,0,0,0,0,0,0,1,1)^T,$$

then $m = (-2, 3, -2, 1, 2)^T$ would be used.

In general the base vectors $\{\psi_k\}$ for $\tilde{\mathcal{E}}_K$ are generated from m in the following way:

(i) (a) if $m_1 > 0$, then elements 1 to m_1 of ψ_1 are 1 and the rest are 0,

(b) if $m_1 < 0$, then elements 1 to $|m_1|$ of ψ_1 are 0 with the remaining elements yet to be described,

(ii) (a) if $m_2 > 0$, then elements of ψ_1, if $m_1 < 0$, or ψ_2, if $m_1 > 0$, from $|m_1| + 1$ to $|m_1| + m_2$ are 1 and the rest are 0,

(b) if $m_2 < 0$, then elements of ψ_1, if $m_1 < 0$, or ψ_2, if $m_1 > 0$, from $|m_1| + 1$ to $|m_1| + |m_2|$ are 0 with the remaining elements yet to be described, etc.

To let m describe the matrix H_K as well as P_K, zero elements are permitted in m. (These are ignored in the preceding interpretation). Whenever zeros appear in the same element of every member of the set $\{\psi_k\}$, zeros appear in the corresponding positions along the diagonal of P_K. For this situation we would normally choose H_K to have unit elements in the corresponding diagonal positions since the Gauss-Seidel method would then be used for partitions of A. If however $m_{i-1} = 0$, $m_i < 0$ then instead of taking $|m_i|$ neighbouring unit elements along the diagonal of H_K we take instead $|m_i|$ neighbouring zeros. Returning to the example above we would have $h_{11} = h_{22} = h_{66} = h_{77} = 1$ with all other elements zero, using an obvious notation for an element of H_K. If however we used instead $m = (2,3,0,-2,1,2)^T$ then the elements h_{66} and h_{77} would be zero. The simplest example of the use of this feature is the way the SOK subroutine may follow the Gauss-Seidel method, $m = (-N)^T$, or the Jacobi method, $m = (0,-N)^T$. For these methods however we should note that the advantage of the Sokolov method is lost since $P_o = 0$.

When K=N the base vectors can only be $\psi_k = e_k$ and hence $P_N = I$. The two extremes P_o and P_N are thus uniquely defined. We saw above that the extreme P_o has no particular merit in this work and the same applies to P_N since with this choice it is easier to calculate the solution of Equation 1 directly.

2.3.2 The base vectors $\{\phi_k\}$

The choice of base vectors $\{\phi_k\}$ for $\bar{\mathcal{E}}_K$ made in this work is

$$\phi_k = D^{1-Q}\psi_k \ , \quad k=1,2,...,k, \tag{22}$$

where $$Q = 0 \text{ or } 1 \ , \tag{23}$$

and consequently the relation between base vectors is either $\phi_k = D\psi_k$ or $\phi_k = \psi_k$. We see in Section 2.4 that the selection Q=0 or Q=1 corresponds to selection of a moments approximation for $\alpha^{(n)}$ using the Galerkin method (Q=0) or the least squares method (Q=1) .

2.4 Determination of $\alpha^{(n)}$

Returning to the problem of determination of the vector $\alpha^{(n)}$ required in Equation 13 we may write the equation as

$$D\delta^{(n)} = \epsilon^{(n-1)} - T\alpha^{(n)} - H_K T_L \delta^{(n)} \ , \tag{24}$$

where $$\epsilon^{(n-1)} = c - Ax^{(n-1)} . \tag{25}$$

Hence employing Equation 12, the definition of $\alpha^{(n)}$, we obtain from Equations 9 and 24 the result

$$(D + P_K T)\,\alpha^{(n)} = P_K \epsilon^{(n-1)} \tag{26}$$

Since $D\alpha^{(n)} = P_K D\delta^{(n)}$, it is an element of $\bar{\mathcal{E}}_K$, and we may seek coefficients $b_k^{(n)}$ such that

$$D\,\alpha^{(n)} = \sum_{k=1}^{K} b_k^{(n)} \phi_k \ ,$$

and hence

$$\alpha^{(n)} = \sum_{k=1}^{K} b_k^{(n)} D^{-Q}\psi_k \ , \tag{27}$$

from Equation 22. Substituting Equation 27 into Equation 26 and taking a scalar product of the result with ψ_i we get

$$\sum_{k=1}^{K} (\psi_i, (D + P_K T) D^{-Q}\psi_k)\, b_k^{(n)} = (\psi_i, P_K \epsilon^{(n-1)}) \ , \qquad i=1,2,...,K.$$

Since $P_K^T \psi_i = \psi_i$ we may further simplify the above to give

$$\sum_{k=1}^{K} (\psi_i, A D^{-Q} \psi_k)\, b_k^{(n)} = (\psi_i, \epsilon^{(n-1)}) \ , \qquad i=1,2,...,K, \tag{28}$$

and then $$b_i^{(n)} = \sum_{k=1}^{K} \bar{w}_{ik} (\psi_i, \epsilon^{(n-1)}) \ , \quad i=1,2,...,K, \tag{29}$$

where $\bar{w}_{ik}$ $(i,k=1,2,...,K)$ are elements of the array $\bar{W}_K$, which is the inverse of the matrix of coefficients $(\psi_i, A D^{-Q} \psi_k)$, provided the inverse exists.

Consider the need to calculate $b_i^{(1)}$, $b_i^{(2)}$,... $(i=1,2,...,K)$. It is worthwhile calculating the inverse once to give $\bar{w}_{ik}$ which may then be used for every iteration step, rather than repeatedly solve Equation 28. In the subroutine SOK the inverse is calculated with the Gauss-Jordan direct method as implemented in the subroutine SID (Davids 1967). The coefficients $b_k^{(n)}$ are substituted into Equation 27 to give $\alpha^{(n)}$ which is then available for use in the main iteration.

To understand the significance of the preceding analysis we readily verify that Equations 27 and 28 correspond to the equation

$$(\psi_i, A\alpha^{(n)} - \epsilon^{(n-1)}) = 0, \qquad i=1,2,...,K$$

and hence $$(\psi_i, D\alpha^{(n)} - D\,\delta^{(n)}) = 0, \qquad i=1,2,...,K \ .$$

Both these equations correspond to the method of moments (Kantorovich and Akilov 1964). The first equation shows $\alpha^{(n)}$ to be an approximation for $\Delta^{(n-1)}$, the solution of the equation

$$A\,\Delta^{(n-1)} = \epsilon^{(n-1)} ,$$

and the second equation shows $D\alpha^{(n)}$ to be an approximation for $D\delta^{(n)}$. The selections Q=0 or Q=1 in Equation 27 correspond to different moment approximations.

(i) Q=0. Equation 27 becomes

$$\alpha^{(n)} = \sum_{k=1}^{K} b_k^{(n)} \psi_k ,$$

hence the first equation shows that the residual $(A\alpha^{(n)} - \epsilon^{(n-1)})$ is orthogonal to the vectors in the expansion of $\alpha^{(n)}$. This is the Galerkin method.

(ii) Q=1. Equation 27 becomes

$$\alpha^{(n)} = \sum_{k=1}^{K} b_k^{(n)} D^{-1} \psi_k ,$$

hence the second equation shows that $\| D\alpha^{(n)} - D\delta^{(n)} \|^2$ is stationary with respect to variation of the coefficients in the expansion of $D\alpha^{(n)}$. This is the method of least squares.

2.5 Main Iteration

We now have available all required terms on the right hand side of Equation 24, which reads

$$(D + H_K T_L)\delta^{(n)} = \epsilon^{(n-1)} - T\alpha^{(n)} .$$

Provided the premultiplying triangular matrix in the above equation is non-singular the solution for $\delta^{(n)}$ is trivial. In general we use

$$x^{(n)} = x^{(n-1)} + \delta^{(n)} \tag{30}$$

to obtain the latest estimate of the solution of Equation 1, although for certain matrices A an extrapolation procedure may be applied spasmodically (Section 2.9).

2.6 Condition for Convergence

To express the Sokolov method in a form suitable for further analysis (as distinct from a form suitable for computational use) Equation 13 may be written

$$Dx^{(n)} = c - Tx^{(n-1)} - U_K(x^{(n)} - x^{(n-1)}), \quad n = 1,2,\ldots \tag{31}$$

Writing x* for the solution of the equation to be solved, Equation 1, and defining

$$\Delta^{(n)} = x^* - x^{(n)}, \tag{32}$$

then Equation 31 becomes

$$\Delta^{(n)} = S_K \Delta^{(n-1)}, \quad n = 1,2,\ldots, \tag{33}$$

where

$$S_K = (U_K + D)^{-1} (U_K - T) . \tag{34}$$

Continued application of Equation 33 is equivalent to

$$\Delta^{(n)} = S_K^n \Delta^{(o)}, \quad n = 1,2,\ldots, \tag{35}$$

As is usual in analysis of convergence of repeated transformations we reduce the study to one of a study of convergence of a sequence of numbers. Numbers are associated with a vector x and a matrix B through a vector norm $\| x \|$, Equation 17, and a matrix norm $\| B \|$, defined as the smallest number g such that

$$\| Bx \| \leq g \| x \| \quad \text{for all } x \in \mathcal{E}_N . \tag{36}$$

Applying the above inequality to Equation 35 we obtain the inequality

$$\|\Delta^{(n)}\| \leqslant \|S_K^n\| \ \|\Delta^{(o)}\| \ , \qquad n=1,2,... \tag{37}$$

If there exists an n such that

$$\|S_K^n\| < 1 \ , \tag{38}$$

then for all $\ell \geqslant n$ we may write

$$\ell = \ell^* + i \ , \tag{39}$$

where ℓ^* is exactly divisible by n and is such that $0 \leqslant i < n$, and hence

$$\|S_K^{\ell}\| \leqslant \|S_K^n\|^{\ell^*/n} \|S_K^i\| \ . \tag{40}$$

Introducing the quantities

$$M = \max \{1, \|S_K\|, \|S_K^2\|, ..., \|S_K^{n-1}\|\} \tag{41}$$

and

$$r = \|S_K^n\|^{1/n}, \tag{42}$$

which are bounded thus:

$$M < \infty \tag{43}$$

$$r < 1 \ , \tag{44}$$

then the inequality (40) may be strengthened to give

$$\|S_K^{\ell}\| \leqslant M r^{\ell^*} \ , \tag{45}$$

hence the inequality (37) now becomes

$$\|\Delta^{(\ell)}\| \leqslant M r^{\ell^*} \|\delta^{(o)}\| \ .$$

Since as $\ell \to \infty$ we have $\ell^* \to \infty$ and $r^{\ell^*} \to 0$, from inequality (44), then

$$\|\Delta^{(\ell)}\| \to 0 \ . \tag{46}$$

From a fundamental property of a norm the above implies

$$\Delta^{(\ell)} \to 0 \ ,$$

and hence

$$x^{(\ell)} \to x^* \ . \tag{47}$$

Convergence is thus assured if the inequality (44) is satisfied, that is if there exists an n such that

$$\|S_K^n\|^{1/n} < 1 \ . \tag{48}$$

In view of the complicated form of the matrix S_K, Equation 34, and the complicated problem of calculating a matrix norm (Fox 1964), verification that the inequality (48) holds is never pursued in practice. It is shown by Halmos (1967) that

$$r(B) = \lim_{n \to \infty} \|B^n\|^{1/n} \ ,$$

is the spectral radius of B, that is, the eigenvalue of B for which $|\lambda|$ is a maximum. The convergence condition (48) may then be replaced by the more acceptable condition.

$$r(S_K) < 1 \ . \tag{49}$$

Even so we are content to use the analysis of this section simply as a guide to our selection of convergence criteria to be discussed in the next section. Mathematical rigour will then be set aside in favour of computational feasibility.

2.7 Analysis of Successive Iterations

Rather than calculate $r(S_K)$ in order to say whether the Sokolov method would converge or not if the method were applied, here we assume the method is applied and we attempt to ascertain whether the Sokolov method has converged or not. We do this from analysis of the norms $\| \delta^{(1)} \|$, $\| \delta^{(2)} \|$, $\| \delta^{(3)} \|$,... From the definitions of $\delta^{(n)}$ (Equation 11) and $\Delta^{(n)}$ (Equation 32)

we have $\delta^{(n)} = \Delta^{(n-1)} - \Delta^{(n)}$,

and then using Equation 35 we obtain

$$\delta^{(n)} = S_K \Delta^{(n-2)} - S_K \Delta^{(n-1)},$$

and hence $$\delta^{(n)} = S_K \delta^{(n-1)}, \qquad n = 2,3,\ldots, \tag{50}$$

which is the basis for this study. Ignoring the first few iterations in the derivation below, continued application of Equation 50 is equivalent to

$$\delta^{(n)} = S_K^{n-3} \delta^{(3)}, \qquad n = 4,5,\ldots, \tag{51}$$

from which we notice that

$$\{ \| \delta^{(n)} \| / \| \delta^{(3)} \| \}^{1/(n-3)} \leq \| S_K^{n-3} \|^{1/(n-3)}, \qquad n = 4,5,\ldots, \tag{52}$$

which leads us to define

$$r_n = \{ \| \delta^{(n)} \| / \| \delta^{(3)} \| \}^{1/(n-3)}, \quad n = 5,6,\ldots \tag{53}$$

The convergence condition (48) is then taken as: if there exists an n such that

$$r_n < 1, \qquad n = 5,6,\ldots, \tag{54}$$

then the method will probably converge. The inequality above strictly speaking will certainly hold if the method is convergent but we cannot draw any rigorous conclusions for the converse. The word "probably" has been added to cover this lapse in rigour. Assuming that we can find an n such that the condition (54) holds we must still decide when to terminate the iteration procedure.

Ideally the iteration would terminate when

$$\| \Delta^{(n)} \| / \| x^* \| \leq \epsilon, \tag{55}$$

where ϵ is the maximum fractional error assigned by the user of the SOK subroutine. It is clear that we need a means of approximately anticipating the iteration steps we have yet to pursue and we note that

$$\Delta^{(n)} = \sum_{j=n+1}^{\infty} \delta^{(j)}, \tag{56}$$

if the method converges. If in addition to demanding that the condition (54) holds, we also require n to be such that r_n is approximately constant as n increases, then we will assume

$$\ldots = r_{n+2} = r_{n+1} = r_n. \tag{57}$$

If the assumption (57) were true we would in fact have

$$r_n = r(S_K), \tag{58}$$

and the condition (54) would be bona-fide.

Returning to Equation 56 we obtain the bound

$$\| \Delta^{(n)} \| \leq \sum_{j=n+1}^{\infty} \| \delta^{(j)} \|, \tag{59}$$

which in view of the assumption (57) becomes

$$\| \Delta^{(n)} \| \leqslant \sum_{j=1}^{\infty} r_n^j \| \delta^{(n)} \| \quad ,$$

and then

$$\| \Delta^{(n)} \| \leqslant \frac{r_n}{1-r_n} \| \delta^{(n)} \| \quad . \tag{60}$$

If we define

$$f_n = \frac{r_n}{1-r_n} \frac{\| \delta^{(n)} \|}{\| x^{(n)} \|} \quad , \qquad n = 5,6,\ldots \, , \tag{61}$$

we may replace the ideal terminating condition expressed as (55) by the computationally feasible, but approximate, condition

$$f_n \leqslant \epsilon \quad . \tag{62}$$

2.8 Convergence and Divergence Criteria

Two numbers ϵ and L are required by the SOK subroutine for use with convergence and divergence criteria. On entry, ϵ indicates the maximum permitted fractional error in the solution and L indicates a limit to the number of iterations. On exit, ϵ indicates an estimate of the maximum fractional error in the solution, f_n , and L indicates the number of iterations taken.

(i) The method is said to have converged if

(a) $r_n < 1$

and (b) $n \leqslant L$

and (c) $f_n \leqslant \epsilon$.

(ii) The method is said to be divergent if

(a) $r_n \geqslant 1$ for $n > \frac{L}{5}$ (the user supplies an estimate of r_n for $n \leqslant \frac{L}{5}$)

or (b) $n > L$.

Although these tests are not strictly correct they are considered to be adequate.

2.9 Extrapolation Procedure

In Section 2.7 we investigated a means of extrapolating the information on hand in order to estimate $\| \Delta^{(n)} \|$ which was required in deciding when the iteration procedure should terminate. The analysis rested on an approximate means of estimating the spectral radius of the iteration matrix, $r(S_K)$ (Equation 58). We now carry out a similar analysis to ascertain whether the extrapolation information may be put to use in the solution process. We start by defining

$$R_n = \| \delta^{(n)} \| / \| \delta^{(n-1)} \| \, , \qquad n = 3,4,\ldots \tag{63}$$

and

$$E_n = \delta_i^{(n)} / \delta_i^{(n-1)} , \qquad n = 4,5,\ldots \, , \tag{64}$$

where i is an index for which $|\delta_i^{(n-3)}|$ is biggest and which does not change in the set of equations below. If there exists an n such that

$$\left.\begin{aligned} & R_n \simeq R_{n-1} \simeq R_{n-2} \\ & |E_n| \simeq R_n \, , \; |E_{n-1}| \simeq R_{n-1} \\ & E_n \simeq E_{n-1} \end{aligned}\right\} \tag{65}$$

hold to within 5 per cent (this may be changed – see Section 3.4) and $|E_n| < 1$ then we will assume

$$\ldots = E_{n+2} = E_{n+1} = E_n \quad , \tag{66}$$

where now $i = 1,2,\ldots,N$ in Equation 64. If the condition (66) were true we would in fact have

$$E_n = \lambda \quad ,$$

where λ is real and is the only eigenvalue of S_K lying on the spectral radius. Unlike the corresponding result in Section 2.7 the extrapolation procedure discussed here only applies to particular iteration matrices S_K. The set of approximate equalities (65) has been chosen to select out these matrices from those likely to be met in practice.

Following the course of earlier analysis we now substitute Equation 66 into Equation 56 to give

$$\Delta^{(n)} = \sum_{j=1}^{\infty} E_n^j \delta^{(n)} \quad , \tag{67}$$

and we immediately obtain the desired extrapolation procedure

$$\Delta^{(n)} = \frac{E_n}{1-E_n} \delta^{(n)} \quad . \tag{68}$$

Equation 30 of Section 2.5 expresses the manner of obtaining the latest solution of Equation 1 for most steps, namely

$$x^{(n)} = x^{(n-1)} + \delta^{(n)},$$

but now we use as well

$$x^{*(n)} = x^{(n)} + \Delta^{(n)} \quad , \tag{69}$$

which would be the solution we require if our extrapolation procedure is exact. Following the application of Equation 69 the normal iteration procedure is continued, except that we take

$$r_n = R_n \quad , \tag{70}$$

for all subsequent steps. After an extrapolation is applied, several more iterations would be necessary before the method may be applied again. This is why in Section 2.5 the extrapolation procedure was said to be spasmodic.

2.10 Change of Convergence with Change of Base Vectors

In general little can be said about change of convergence with change of K and the set $\{\psi_k\}$. If storage permitted then we could take $K=N$ and hence $S_K=0$ and $r(S_K)=0$, which implies that the method must converge. This limiting behaviour is of little interest since when $K=N$ no iteration is really necessary as we are solving Equation 1 directly. Ideally we would like

$$r(S_{K+\ell}) < r(S_K) \quad , \qquad \ell = 1,2,\ldots \quad , \tag{71}$$

for some choice of base vectors (Section 2.3), although we might expect that if we chose the base vectors well for use with S_K the inequality (71) may break down. When using SOK for solving the homogeneous multigroup neutron flux equation for a reactor model, as employed in GYMEA (Pollard and Robinson 1966) we are in a similar position to that of attempting to choose the best number of groups and arrangement of collapsed groups for a particular space dependent calculation to follow. This problem remains unsolved at present, although fortunately we can proceed by taking other than the optimum choice.

Returning to the problem in hand, again an optimum choice is not essential and certainly in this case, provided our choice is such that the method converges, we will obtain the solution of Equation 1 to a specified accuracy. Experience with solving the types of equations that arise in GYMEA with $N \simeq 70$ (upscattering groups only) indicates that the inequality (71) mostly holds when the base vectors $\{\psi_k\}$ roughly contain the same number of unit elements and that $K=10$ is a suitable choice (Section 4.1).

The need to choose K and a set of base vectors may be considered a disadvantage of the Sokolov method. In fact it is an advantage since we are able to obtain a subroutine specifically designed to solve a particular problem simply by making this choice.

3. THE SOK SUBROUTINE

3.1 Subroutine Details

The calling sequence for the subroutine consists of the following...

CALL SOK(N,K,M,IQ,C,AA,X,W,Z,L,E,R)

where

N is the order of the set of equations (Section 1),

K is the order of the subsidiary set of equations (Section 2.1) – this should be not greater than 50 using the standard version of SID (Davids 1967),

M is the vector m of integers, K of which are positive, which describe the base vectors $\{\psi_k\}$ and which must satisfy Equation 21 (Section 2.3),

IQ is the indicator for selection of the particular method of moments required, IQ=0 for the Galerkin method and IQ=1 for the method of least squares (Section 2.3.2),

C is the vector c of N constants (Section 1),

AA is the name of a subroutine provided by the user for off-loading elements of the array A which would normally be stored in a compacted form (Section 3.3),

X is a trial solution $x^{(o)}$ of order N (say all zero elements) on entry and is the final solution $x^{(n)}$ on exit (Section 1),

W is a working array with DIMENSION W(K,K+2) which is used by the subroutine to store $\overline{W}_K$ (Equation 29), $(\psi_i, \epsilon^{(n-1)})$ (Equation 28) and $b_l^{(n)}$ (Equation 29),

Z is a working array with DIMENSION Z(N,2) which is used by the subroutine to store D when $h_{ii} = 0$ (Equation 2) and $\delta^{(n)}$ (Equation 24),

L is a limit to the number of iterations on entry and is the number of iterations taken on exit (Section 2.8),

E is the maximum fractional error permitted for the solution on entry and is an estimate of the fractional error of the solution on exit, f_n (Equation 61),

and R is an estimate of $r(S_K)$ (Equation 58) on both entry (the user's estimate) and exit (the subroutine's estimate). (If the user has no knowledge of $r(S_K)$ then R=0.8 should suffice). If R is zero on entry then all previously defined variables will be used from the previous entry. This entry option only makes sense when convergence of the one problem is being investigated and then only for the one choice of base vectors.

An example of the use of the subroutine is given in Table 2.

When the subroutine is unable to proceed with the solution of Equation 1, L is returned with a non-positive value indicative of the error. The user should test L on exit from SOK and take appropriate action. The error conditions are indicated by:

(i) L=0; the limit to the number of iterations specified on input has been reached,

(ii) L = – (small number), $r_n \geq 1$ for the $-L^{th}$ iteration step (the test for $r_n \geq 1$ is not carried out until $n > \max(L/5,4)$),

(iii) L=–777, an attempt has been made to continue a solution, using R=0 on entry, which has previously terminated with an error,

(iv) L=–888, the matrix $(D+H_K T_L)$ is singular,

(v) L=–999, the matrix composed of elements $(\psi_i, AD^{-Q}\psi_k)$ is singular.

3.2 Off-loading Elements of A

The user of the SOK subroutine is required to provide a subroutine with a name corresponding to the 6th argument of the list supplied as argument to SOK. The subroutine so named is then called by SOK when elements of the matrix A are required (Section 1). Using this feature it is possible to use a compact storage arrangement for the matrix (Section 3.3) and to by-pass calculations associated with zero elements. The subroutine, say AA, would start, for example,

SUBROUTINE AA(I,J,AIJ,DJJ) ,

where I is the row index of the matrix A, except as mentioned below,

J is the column index of the matrix A (an input variable only),

AIJ is the returned value of a_{ij} ,

and DJJ is the returned value of d_{jj} , a diagonal element of D, which need only be set on first entry and which is used to terminate processing of a column as mentioned below.

The SOK subroutine has been written in a way which processes a column of A at a time in order to make full use of a possible compact storage arrangement. Although some rows and columns of A may be skipped during the process of off-loading elements, the columns are always processed in natural order. Details of (i) the start and (ii) the termination of a column follow.

(i) On first entry for a particular column J the row index is set to –I by the SOK subroutine to indicate to the subroutine AA that a new column is to be processed. The subroutine AA must change I from that implied on entry to the first non-zero element a_{ij} or if processing of other than the first element is indicated, the subroutine must simply change the sign of I. Subsequent entry to AA for the one column is with I increased by 1 although I may be further changed in AA.

(ii) Should an index I be supplied which is beyond all non-zero elements of column J then DJJ=0 should be used to bypass subsequent processing of that column. The SOK subroutine checks the index to prevent the condition I > N from arising.

In addition to the significance attached to negative values of the index I mentioned above we have a further interpretation for I=0. After the last required element of A has been off-loaded for one complete pass through the matrix, the off-loading subroutine is called with I=0. The purpose of this is to enable a disk or tape data set to be rewound if A is being off-loaded from this type of medium. Two complete passes are made through the matrix A for every iteration step unless $x^{(o)}=0$ in which case only one pass is required. In addition one complete pass is required in the once off calculation of $\overline{W}_K$.

As an example of a suitable subroutine for off-loading elements of A we will ignore any possible compact storage arrangement (however see Section 3.3) and we will assume that A is available from disk FORTRAN unit 9 a column at a time. We could write:

```
      SUBROUTINE AA(I,J,AIJ,DJJ)
      DIMENSION AJ(500)
      COMMON /SOKI/N,K,IQ,IT,ITT  (N is the order – see Section 3.5)
      IF(I)1,2,3
    1 READ(9)J1,(AJ(I1),I1=1,N)
      IF(J1.NE.J)GO TO 1
      I=-I
      DJJ=AJ(J)      (DJJ need not be the element ajj – see Section 1)
    3 AIJ=AJ(I)
      RETURN
    2 REWIND 9
      RETURN
      END
```

3.3 A Possible Compact Storage Arrangement for A

Given an NXN matrix A the array is stored in successive core locations a column at a time. The storage could thus be visualized as consisting of

$$(a_{11},a_{21},\ldots,a_{N1},a_{12},a_{22},\ldots,a_{N2},\ldots,a_{1N},a_{2N},\ldots,a_{NN}),$$

or $$(A_1,A_2,\ldots,A_N),$$

where A_i, $i=1,2,\ldots,N$, are the columns of A. When the columns of A consist of many neighbouring zero elements before and after a group of neighbouring elements which are non-zero and which cluster around diagonal terms then the use of the compact storage arrangement described here becomes feasible. Similar storage arrangements could be designed to suit other particular types of matrices. The types of matrices we have in mind appear in the multigroup representation of the Boltzmann transport equation and arise through the limited energy loss or gain possible in a neutron collision. A compacted form of neutron scattering matrices $\sigma_{j\to i}$ is used on input to the code WDSN (Francescon 1963) and a similar form is used in GYMEA (Pollard and Robinson 1966).

The compact storage arrangement to be discussed here consists of a rectangular I(<N)XN array

$$(A^*_1,A^*_2,\ldots,A^*_N).$$

where $a^*_{1j} = \ell_j$, an integer indicating the number of elements of A_j that follow later,

$a^*_{2j} = k_j$, an integer indicating the position of the diagonal element of A_j in the array to follow ($a_{jj} = a^*_{k_j+2,j}$),

and $a^*_{3j}, a^*_{4j}, \ldots, a^*_{\ell_j+2,j}$ are the neighbouring non-zero elements of A_j.

From this information we may ascertain the value of a typical element a_{ij} thus:

$a_{ij}=0$ if $i<j-k_j+1$ or $i>j-k_j+\ell_j$

and $a_{ij} = a^*_{i-j+k_j+2,j}$ otherwise.

An example of a subroutine to off-load elements from this compact storage arrangement follows.

```
      SUBROUTINE AC(I,J,AIJ,DJJ)
      COMMON /ACC/A(133,150)
      DIMENSION LK(133,150)
      EQUIVALENCE (A,LK)
      IF(I)1,2,3
    1 I=-I
      NI=LK(1,J)+2
      IJ=J-LK(2,J)+1
      II=I-IJ
      IF(II.LT.0)II=0
      I=II+IJ
      II=II+2
      DJJ=A(LK(2,J)+2,J)
    3 II=II+1
      IF(II.GT.NI)GOTO 4
      AIJ=A(II,J)
    2 RETURN
    4 DJJ=0.
      RETURN
      END
```

3.4 Additional Options

In Section 2.9 the conditions (65) for applying the extrapolation procedure, Equation 69, were required to hold to within 5 per cent. The required accuracy may be changed to say 2 per cent by using

```
CALL SOKOPT(0.02,0)
```

before a call to SOK. If the extrapolation procedure is definitely to be avoided then

```
CALL SOKOPT(0.,0)
```

should be used.

The second argument of the SOKOPT entry is used to set the SOK subroutine to take particular advantage of a possible saving of computer time for matrices arising in homogeneous multigroup neutron flux calculations (Section 4.1). If the matrix A may be partitioned thus

$$A = \left(\begin{array}{c|c} A_{11} & O \\ \hline A_{21} & A_{22} \end{array}\right) ,$$

where A_{11} is a square JXJ matrix which is of lower triangular form, then Equation 1 may be written

$$\left(\begin{array}{c|c} A_{11} & O \\ \hline A_{21} & A_{22} \end{array}\right) \left(\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right) = \left(\begin{array}{c} c_1 \\ \hline c_2 \end{array}\right) , \qquad (72)$$

where the vectors have been partitioned in like manner to the matrix A. If our selection of the vector m (Section 2.3.1) should start $m_1 = -J$, and this is essential, the solution for x_1 is available after the first iteration and for subsequent iterations we need only solve

$$A_{22}\, x_2 = c_2' , \qquad (73)$$

where

$$c_2' = c_2 - A_{21}\, x_1 . \qquad (74)$$

If we require this option to be available we should use

```
CALL SOKOPT(0.05,1).
```

The transformation given as Equation 74 is carried out by storing c_2' back in the array C (Section 3.1) and hence C is effectively destroyed for subsequent use when this option is used.

3.5 Additional Variables

Two labelled COMMON areas are set aside for (i) passing information to the off-loading subroutine provided by the user (Section 3.2) and (ii) returning information about convergence of a problem. The first,

```
COMMON /SOKI/N,K,IQ,IT,ITT ,
```

consists of the variables

N, the order of the set of equations to be solved,

K, the order of the subsidiary set of equations,

IQ, the indicator for the method of moments to be used,

IT, the current iteration count for the present entry to SOK,

and ITT, the total iteration count which includes previous iterations when SOK is entered with R=0 (Section 3.1).

The second,

COMMON /SOKC/XN,DELN,IRN,EN ,

consists of the variable

XN= $\| x^{(n')} \|$, where n' is the last index for which $f_{n'} > 0.05$,

DELN= $\| \delta^{(n)} \|$, where n is the exit iteration index,

IRN= the number of times the extrapolation procedure was used,

and EN= E_n for the last extrapolation if one was carried out.

4. THE USE OF SOK

4.1 Homogeneous Multigroup Neutron Flux Calculations

To gain some experience with possible ways to use the SOK subroutine, a series of calculations was undertaken for the multigroup neutron flux in an infinite homogeneous reactor system. A sample of the results is shown in Table 1. All data for the calculations were prepared by GYMEA (Pollard and Robinson 1966) and the equations solved are the same as those solved by GYMEA. The calculations were carried out on the systems detailed below:

1. All systems used scattering determined by D_2O at 300 °K. The concentration of D_2O was taken as 3×10^{-2} molecules/$(10^{-8}$ cm$)^3$.

   (i) Four spectrum calculations were carried out with different amounts of ZZ^{999}, a pseudo nuclide with an absorption cross section of 1 barn at a neutron velocity of 2,200 m/sec. These were, using a synonym for the type of spectrum

   (a) FAST , ZZ^{999} 3×10^{-1} atoms/$(10^{-8}$ cm$)^3$,

   (b) INTERMEDIATE , ZZ^{999} 3×10^{-2} atoms/$(10^{-8}$ cm$)^3$,

   (c) THERMAL , ZZ^{999} 3×10^{-3} atoms/$(10^{-8}$ cm$)^3$,

   and (d) VERY THERMAL , ZZ^{999} 3×10^{-4} atoms/$(10^{-8}$ cm$)^3$.

   (ii) Again, four spectrum calculations were carried out but this time with different amounts of natural uranium (taken as having atomic proportions 0.993 U^{238} and 0.007 U^{235}) chosen so that the absorption cross sections at a neutron velocity of 2,200 m/sec were the same as (a) to (d) above.

2. The number of groups used in each calculation was N=120 of which J=47 did not have any up-scatter from lower groups, that is, the matrix may be partitioned as mentioned in Section 3.4.

3. (i) For most runs the base vectors ψ_k, k=1,2,...,K were chosen so that approximately the same number of unit elements appeared in each vector with the Gauss-Seidel method being applied to the fast part of the matrix of coefficients (down to about 1 eV). For example the following were the selections of the vector m (Section 2.3.1).

   K=10 , m=(−47,−23,10*5) (10*5=5,5,...,5,5)

   and K=20 , m=(−47,−23,10*3,10*2).

   (ii) To gain some insight into the effect of a change of base vectors, but with the order fixed, the following additional vectors m were used

(a) K=10, m=(-47,-33,10*4)

and (b) K=10,m=(-47,3*8,7*7).

(iii) The Galerkin method was used for all calculations (Q=0, Section 2.4).

4. (i) The diagonal matrix D was chosen to consist of all the diagonal terms of the matrix A for all calculations (Section 1).

(ii) To make the comparison between the calculations as simple as possible, extrapolation was not permitted (Section 3.4).

(iii) Advantage was taken of the time saving resulting from the partition of A for J=47 which makes A_{11} lower triangular (Section 3.4).

(iv) A maximum fractional error of $\epsilon = 10^{-4}$ was used in all calculations.

(v) The trial solution $x^{(o)}=0$ was used in all calculations.

From the results of the calculations given in Table 1 we see that the choice K=10, and m as given in Part 3. (i) above, is acceptable. Unfortunately the method is not as fast as was originally hoped. (The same calculations carried out in GYMEA are at least 50 per cent faster although the trial solution is taken in GYMEA as a Wescott flux which could make a slight difference). The reason for this is probably twofold, (i) two accesses are required of the matrix A for each iteration step and (ii) the off-loading of elements is slow, particularly using a FORTRAN subroutine. The difficulty (i) could be overcome by using a working array Y with DIMENSION Y(N,K) to store the information required in applying Equation 24. In general the extra storage that this step requires makes the change impractical. The difficulty (ii) could be overcome to some extent by writing the off-loading subroutine in assembly language. In general this is a serious disadvantage of the method. Of course both difficulties could be overcome if the matrix A were available in core, stored in the normal way, only then no advantage could be taken of the computer time and storage saving possible when using an off-loading subroutine with sparse arrays.

4.2 A Simple Example

To show the way in which the subroutine SOK is used, a listing of a simple example, and all subroutines required, is given in Table 2. The simple example does not show the method to particular advantage since it makes no use of a compact storage arrangement.

5. CONCLUDING REMARKS

Experience with the Sokolov method, as applied in this work, for solving the homogeneous multigroup neutron flux equations, shows the method to be quite acceptable although not as fast as it should be. The flux equations, however, are such that their array of coefficients is not very sparse. For sparse matrices the advantages of the method should become more apparent.

6. REFERENCES

Davids, R.E. (1967). – Abstracts of computer programmes available locally and through 'SHARE'. Unpublished A.A.E.C. report (AM/CP11).

Fox, L. (1964). – An Introduction to Numerical Linear Algebra. Clarendon Press, London.

Francescon, S. (1963). – The Winfrith DSN programme. AEEW-R273.

Halmos, P.R. (1958). – Finite-Dimensional Vector Spaces. Van Nostrand, Princeton, N.J.

Halmos, P.R. (1967). – A Hilbert Space Problem Book. Van Nostrand, Princeton, N.J.

Kantorovich, L.V. and Akilov, G.P. (1964). – Functional Analysis in Normed Spaces. Pergamon Press, Oxford.

Luchka, A.Yu. (1965). – The Method of Averaging Functional Corrections. Academic Press, N.Y.

Pollard, J.P. and Robinson, G.S. (1966). – GYMEA – A nuclide depletion, space independent, multigroup neutron diffusion, data preparation code. AAEC/E147.

Wachspress, E.L. (1966). – Iterative Solution of Elliptic Systems. Prentice-Hall, Englewood Cliffs, N.J.

TABLE 1

PERFORMANCE OF SOK FOR HOMOGENEOUS MULTIGROUP

NEUTRON FLUX CALCULATIONS

Details of the calculations are given in Section 4.1. The table entries r_n, n and time relate to, respectively, an estimate of the spectral radius of the iteration matrix, the number of iterations and the time (in seconds) taken for convergence on the IBM 360/50 computer.

Systems of D_2O/ZZ^{999}

SPECTRUM	FAST			INTERMEDIATE			THERMAL			VERY THERMAL		
K	r_n	n	time	r_n	n	time	r_n	n	time	r_n	n	time
0	0.36	8	24	0.85	> 20	–	0.97	> 20	–	0.98	> 20	–
5	0.08	5†	16	0.30	10	32	0.36	12	38	0.36	12	38
10	0.04	5	16	0.11	6	19	0.19	8	26	0.19	8	26
20	0.04	5	17	0.09	6	20	0.14	7	24	0.14	7	24
30	0.04	5	19	0.05	5	19	0.08	6	22	0.08	6	22
40	0.04	5	22	0.05	5	22	0.05	5	22	0.05	5	22
50	0.04	5	27	0.04	5	27	0.04	5	27	0.04	5	27
10(a)	0.09	5	15	0.15	7	21	0.19	8	25	0.19	8	25
10(b)	0.29	7	26	0.24	7	26	0.25	8	30	0.25	8	30

Systems of D_2O/natural uranium

SPECTRUM	FAST			INTERMEDIATE			THERMAL			VERY THERMAL		
K	r_n	n	time	r_n	n	time	r_n	n	time	r_n	n	time
0	0.37	7	21	0.86	>20	–	0.97	>20	–	0.98	> 20	–
5	0.08	5	16	0.29	9	29	0.36	12	38	0.36	12	38
10	0.04	5	16	0.11	6	19	0.19	8	26	0.19	8	26
20	0.04	5	17	0.09	6	20	0.14	7	24	0.14	7	24
30	0.04	5	19	0.05	5	19	0.08	6	22	0.08	6	22
40	0.04	5	22	0.05	5	22	0.05	5	22	0.05	5	22
50	0.04	5	27	0.04	5	27	0.04	5	27	0.04	5	27
10(a)	0.09	5	15	0.12	6	18	0.19	8	25	0.19	8	25
10(b)	0.51	13	48	0.21	7	26	0.25	8	30	0.25	8	30

† For $n<5$ the estimate of the spectral radius of the iteration matrix was taken as 0.8. This prevented early anticipation of convergence of the calculations with very low spectral radius.

TABLE 2

A LISTING OF A SIMPLE EXAMPLE, INCLUDING ALL SUBROUTINES

```
C       A SIMPLE EXAMPLE
        COMMON /AAA/ A(120,120)
        DIMENSION M(30),W(25,27),Z(120,2),X(120),C(120)
        EXTERNAL AA
        IQ=0
C       IQ FOR GALERKIN METHOD
        NO=1
    1   READ(1,2)N,K,KD,L
C                       KD INCLUDES ANY NON-POSITIVE NUMBERS
    2   FORMAT(10I5)
        READ(1,2)(M(I),I=1,KD)
        READ(1,3)(X(I),I=1,N)
    3   FORMAT(5E15.6)
        READ(1,3)(C(I),I=1,N)
        DO 4 J=1,N
    4   READ(1,3)(A(I,J),I=1,N)
        E=1.E-4
        R=0.8
        CALL SOK(N,K,M,IQ,C,AA,X,W,Z,L,E,R)
        WRITE(3,5)NO,L,E,R
    5   FORMAT('0PROBLEM',I3,' L,E,R AND X FOLLOW',I5,1P2E15.6)
        WRITE(3,6)(X(I),I=1,N)
    6   FORMAT(1X1P5E15.6)
        NO=NO+1
        GO TO 1
        END
        SUBROUTINE AA(I,J,AIJ,DJJ)
        COMMON /AAA/ A(120,120)
        IF(I)1,2,3
    1   I=-I
        DJJ=A(J,J)
    3   AIJ=A(I,J)
    2   RETURN
        END
```

```
      SUBROUTINE SOK(N,K,M,IQ,C,SOKA,X,W,Z,L,E,R)
C     SOLVES THE EQN  AX=C
C     USING THE METHOD OF AVERAGING FUNCTIONAL CORRECTIONS
      EXTERNAL SOKA
      DIMENSION M(1),C(3),X(3),W(K,2),Z(N,2)
      COMMON /SOKI/ NJ,KJ,IQQ,IT,ITT
      COMMON /SOKC/ XN,DELN,IRN,EN
      DATA ACE/0.05/,ILL/0/,ACXN/0.05/,ITS/0/
      NJ=N
      KJ=K
      IQQ=IQ
      IT=777
      IF(R.EQ.0..AND.ITS.NE.0)GO TO 1
      IF(R.EQ.0.)GO TO 22
      IRN=0
      ITT=0
      LA=L/5
      IF(LA.LT.4)LA=4
      IL=0
      IF(ILL.GT.0)IL=-M(1)
      IJ=1
      IK=1
      RN=R
      RNC=R
      RR=RNC/(1.-RNC)
      FN=0.
      EN=0.
      XN=1.E-40
      DO 75 I=1,N
   75 Z(I,1)=1.
      IF(K.LE.0)GO TO 1
      CALL WKBAR(M,SOKA,W,Z)
      IF(IT.GE.999)GO TO 24
    1 DO 46 I=1,N
      IF(X(I).NE.0.)GO TO 47
   46 CONTINUE
      IX=0
      GO TO 48
   47 IX=1
   48 DO 2 IT=1,L
      ITT=ITT+1
      IF(IL.LE.0)GO TO 65
      IF(ITT-2)65,66,67
   66 IK=IL+1
      IX=2
      IF(IK.LE.N)GO TO 65
      FN=0.
      RNC=0.
      GO TO 23
   67 IJ=IK
   65 DO 3 I=IK,N
    3 Z(I,2)=C(I)
      IF(IX.EQ.0)GO TO 49
      DO 64 J=IJ,N
      I=-IK
    5 IF(I.GT.N)GO TO 4
```

```
      CALL SOKA(I,J,W2,WD)
      IF(WD.EQ.Ø.)GO TO 4
      W2=W2*X(J)
      Z(I,2)=Z(I,2)-W2
      IF(IX.NE.2)GO TO 68
      C(I)=C(I)-W2
   68 I=I+1
      GO TO 5
    4 IF(IX.EQ.2.AND.J.EQ.IL)IX=3
   64 CONTINUE
      I=Ø
      CALL SOKA(I,J,W2,WD)
      GO TO 7Ø
   49 IX=1
   7Ø IF(K.LE.Ø)GO TO 59
      CALL SCALPK(M,Z(1,2),W(1,K+1))
      DO 6 I=1,K
      W1=Ø.
      DO 7 J=1,K
    7 W1=W1+W(I,J)*W(J,K+1)
    6 W(I,K+2)=W1
   59 J=1
      J2=Ø
      J3=Ø
      IF(IX.LE.1)GO TO 61
      J4=Ø
      GO TO 9
   69 J=J+1
   61 J4=1
    9 J1=J2+1
      IF(J1.GT.N)GO TO 18
      J3=J3+1
      MJ3=M(J3)
      J2=J2+IABS(MJ3)
      IF(MJ3)5Ø,6Ø,51
   6Ø J4=Ø
      GO TO 9
   5Ø IF(J4.LE.Ø)GO TO 61
      DO 52 JJ=J1,J2
      I=-JJ
   55 IF(I.GT.J2)GO TO 52
      CALL SOKA(I,JJ,W2,WD)
      IF(WD.EQ.Ø.)GO TO 52
      IF(I.NE.JJ)GO TO 53
      IF(W2.NE.Ø.)GO TO 54
      IT=888
      GO TO 24
   54 Z(I,2)=Z(I,2)/W2
      GO TO 56
   53 Z(I,2)=Z(I,2)-W2*Z(JJ,2)
   56 I=I+1
      GO TO 55
   52 CONTINUE
      GO TO 61
   51 W1=W(J,K+2)
      DO 12 JJ=J1,J2
      I=-IK
```

```
10 IF(I.GT.N)GO TO 12
   CALL SOKA(I,JJ,W2,WD)
   IF(WD.EQ.0.)GO TO 12
   IF(I.EQ.JJ)W2=W2-WD
   IF(IQ.EQ.1)W2=W2/WD
   Z(I,2)=Z(I,2)-W2*W1
   I=I+1
   GO TO 10
12 CONTINUE
   GO TO 69
18 DO 76 I=1,N
76 Z(I,2)=Z(I,2)/Z(I,1)
   I=0
   CALL SOKA(I,JJ,W2,WD)
72 DELN=ANORM(IK,N,Z(1,2))
   IF(ITT.EQ.1)GO TO 42
   RN=DELN/DELN1
   IF(ITT.LE.LA)GO TO 30
   RNC=(DELN/DEL3)**(1./(ITT-3))
   IF(RNC.GE.1.)GO TO 22
   RR=RNC/(1.-RNC)
30 IF(ABS(RN/RN1-1.).GT.ACE)GO TO 42
   IF(IRC.GE.1)GO TO 43
   IRC=1
   BD=-1.
   DO 40 I=IK,N
   W1=ABS(Z(I,2))
   IF(W1.LE.BD)GO TO 40
   BD=W1
   IBD=I
40 CONTINUE
44 BD=Z(IBD,2)
   EN1=EN
   GO TO 14
43 IRC=IRC+1
   EN=Z(IBD,2)/BD
   IF(ABS(ABS(EN)-RN).GT.ACE)GO TO 42
   IF(IRC.NE.3)GO TO 44
   IF(ABS(EN-EN1).GT.ACE)GO TO 42
   IF(ABS(EN).GE.1.)GO TO 24
   EXTRAP=1./(1.-EN)
   DO 45 I=IK,N
45 X(I)=X(I)+Z(I,2)*EXTRAP
   IRC=-1
   IRN=IRN+1
   IF(IRN.GT.1)GO TO 29
   LA=99999
   RNC=RN
   RR=RNC/(1.-RNC)
   GO TO 29
42 IRC=0
14 DO 13 I=IK,N
13 X(I)=X(I)+Z(I,2)
29 IF(FN.LE.ACXN)GO TO 15
   XN=ANORM(1,N,X)
15 FN=RR*DELN/XN
   DELN1=DELN
```

```
      IF(ITT.EQ.3)DEL3=DELN
      RN1=RN
      IF(FN.LE.E)GO TO 23
    2 CONTINUE
      IT=0
      GO TO 23
   24 RNC=EN
   22 L=-IT
      ITS=0
      GO TO 25
   23 L=IT
      ITS=1
   25 E=FN
      R=RNC
      RETURN
      ENTRY SOKOPT(RACE,ILR)
      ACE=RACE
      ILL=ILR
      RETURN
      END
```

```
      SUBROUTINE WKBAR(M,SOKA,W,Z)
C     INVERSE OF K*K MATRIX WITH ELEMENTS (PSI(I),A/D**IQ*PSI(K))
      EXTERNAL SOKA
      COMMON /SOKI/ N,K,IQ,IT,ITT
      DIMENSION M(1),W(K,2),Z(N,2)
      J2=0
      J3=0
      DO 1 J=1,K
    9 J1=J2+1
      J3=J3+1
      MJ3=M(J3)
      J2=J2+IABS(MJ3)
      IF(MJ3.LE.0)GO TO 9
      DO 7 I=1,N
    7 Z(I,2)=0.
      DO 6 JJ=J1,J2
      I=-1
    5 IF(I.GT.N)GO TO 6
      CALL SOKA(I,JJ,W2,WD)
      IF(WD.EQ.0.)GO TO 6
      WD1=WD
      IF(IQ.EQ.1)W2=W2/WD
      Z(I,2)=Z(I,2)+W2
      I=I+1
      GO TO 5
    6 Z(JJ,1)=WD1
    1 CALL SCALPK(M,Z(1,2),W(1,J))
      I=0
      CALL SOKA(I,JJ,W2,WD)
      CALL SID(W,-1,K,K,K,DET)
      IF(DET.NE.0.)RETURN
      IT=999
      RETURN
      END
```

```
      SUBROUTINE SCALPK(M,V,P)
C     SCALAR PRODUCTS, P(I)=(PSI(I),V), OF N-DIM VECTORS
      DIMENSION M(1),V(3),P(1)
      COMMON /SOKI/ N,K,IQ,IT,ITT
      J2=0
      J3=0
      DO 1 J=1,K
    9 J1=J2+1
      J3=J3+1
      MJ3=M(J3)
      J2=J2+IABS(MJ3)
      IF(MJ3.LE.0)GO TO 9
      W1=0.
      DO 2 JJ=J1,J2
    2 W1=W1+V(JJ)
    1 P(J)=W1
      RETURN
      END
```

```
      FUNCTION ANORM(IK,N,X)
C     EUCLIDEAN NORM OF VECTOR X(DIM N)
      DIMENSION X(3)
      W1=0.
      DO 1 I=IK,N
    1 W1=W1+X(I)*X(I)
      ANORM=SQRT(W1)+1.E-40
      RETURN
      END
```

```
      SUBROUTINE SID(SIDW,LSID,NROW,MSID,NSID,SIDET)
C     SUBROUTINE TO SOLVE INVERSE AND DETERMINANT OF M*M MATRIX
C     WITH SOLUTION MATRIX M*(N-M) IF REQD. USES GAUSS-JORDAN METHOD.
C     EG. TO SOLVE ((A))(X)=(B), A=3 X 3, B=3 X 1,
C     THEN STORE (A1),(A2),(A3),(B) IN NEIGHBOURING COLS.
C     RESULT     (R1),(R2),(R3),(X).
C     IF LSID=+1 THEN ONLY DET AND (X) SENSIBLE.
C     IF LSID=0 THEN ONLY DET SENSIBLE.
C     IF LSID=-1 THEN INVERSE WILL ALSO BE IN ((R))
C     NOTE THAT THE ORIGINAL MATRIX IS DESTROYED
C     AS THE RESULT OCCUPIES LOCATIONS PREVIOUSLY ((A)).
C     'NROW' MUST BE THE NO. OF ROWS IN THE DIMENSIONED VARIABLE FOR
C     WHICH 'SIDW' IS A SUBSTITUTE.
C     IF MSID IS MORE THAN 50, CHANGE IFSID(50), ETC.
C     TO THE NUMBER OF ROWS REQD.
C     DIMENSION IFSID(50),ILSID(50),IGSID(50), SIDW(50,50)
      DIMENSION IFSID(50),ILSID(50),IGSID(50),SIDW(2)
      SIDET=1.
      ISID1=MSID-1
      ISIDR=NSID-MSID
      IF(ISIDR)8802,8851,8851
 8851 DO 8822 KSID=1,MSID
      ILSID(KSID)=0
 8822 IGSID(KSID)=KSID
      DO 8803 KSID=1,MSID
      IF(LSID)8852,8853,8853
 8852 KSID1=1
      GO TO 8854
 8853 KSID1=KSID+1
 8854 RSID=0.
      DO 8804 ISID=1,MSID
      IF(ILSID(ISID))8804,8805,8804
C8805 WSID=SIDW(ISID,KSID)
 8805 JRSID=ISID+(KSID-1)*NROW
      WSID=SIDW(JRSID)
      XSID=WSID
      IF(XSID)8806,8807,8807
 8806 XSID=-XSID
 8807 IF(XSID-RSID)8804,8808,8808
 8808 RSID=XSID
      PSID=WSID
      KFSID=ISID
 8804 CONTINUE
      IFSID(KSID)=KFSID
      ILSID(KFSID)=KFSID
      SIDET=SIDET*PSID
      IF(SIDET)8810,8802,8810
 8810 DO 8815 ISID=1,MSID
      JRSID=ISID+(KSID-1)*NROW
      IF(ISID-KFSID)8855,8856,8855
C8856 SIDW(ISID,KSID)=1./PSID
 8856 SIDW(JRSID)=1./PSID
      GO TO 8815
C8855 SIDW(ISID,KSID)=-SIDW(ISID,KSID)/PSID
 8855 SIDW(JRSID)=-SIDW(JRSID)/PSID
 8815 CONTINUE
```

```
      IF(KSID1-NSID)8870,8870,8803
 8870 DO8825JSID=KSID1,NSID
      IF(JSID-KSID)8858,8825,8858
C8858 WSID=SIDW(KFSID,JSID)
 8858 JRSID=KFSID+(JSID-1)*NROW
      WSID=SIDW(JRSID)
      IF(WSID)8821,8825,8821
 8821 DO 8820 ISID=1,MSID
      JRSID=ISID+(JSID-1)*NROW
      IF(ISID-KFSID)8823,8824,8823
C8824 SIDW(ISID,JSID)=WSID/PSID
 8824 SIDW(JRSID)=WSID/PSID
      GO TO 8820
C8823 SIDW(ISID,JSID)=SIDW(ISID,JSID)+WSID*SIDW(ISID,KSID)
 8823 JDSID=ISID+(KSID-1)*NROW
      SIDW(JRSID)=SIDW(JRSID)+WSID*SIDW(JDSID)
 8820 CONTINUE
 8825 CONTINUE
 8803 CONTINUE
      DO 8840 KSID=1,ISID1
      KFSID=IFSID(KSID)
      KLSID=ILSID(KFSID)
      KGSID=IGSID(KSID)
      IF(KFSID-KGSID)8841,8840,8841
 8841 IF(LSID)8842,8843,8844
 8844 IF(ISIDR)8802,8843,8846
 8842 DO 8861 ISID=1,MSID
      JRSID=ISID+(KFSID-1)*NROW
      JDSID=ISID+(KGSID-1)*NROW
C     RSID=SIDW(ISID,KFSID)
      RSID=SIDW(JRSID)
C     WSID=SIDW(ISID,KGSID)
      WSID=SIDW(JDSID)
C     SIDW(ISID,KFSID)=WSID
      SIDW(JRSID)=WSID
C8861 SIDW(ISID,KGSID)=RSID
 8861 SIDW(JDSID)=RSID
 8846 DO 8860 JSID=1,NSID
      JRSID=KSID+(JSID-1)*NROW
      JDSID=KLSID+(JSID-1)*NROW
C     RSID=SIDW(KSID,JSID)
      RSID=SIDW(JRSID)
C     WSID=SIDW(KLSID,JSID)
      WSID=SIDW(JDSID)
C     SIDW(KSID,JSID)=WSID
      SIDW(JRSID)=WSID
C8860 SIDW(KLSID,JSID)=RSID
 8860 SIDW(JDSID)=RSID
 8843 ILSID(KFSID)=KSID
      ILSID(KGSID)=KLSID
      IGSID(KLSID)=IGSID(KSID)
      IGSID(KSID)=KFSID
      SIDET=-SIDET
 8840 CONTINUE
 8802 RETURN
C     END OF SUBROUTINE
      END
```