Pergamon

Int. J. Heat Mass Transfer. Vol. 38, No. 6, pp. 999–1008, 1995

0017-9310/95 $9.50 + 0.00

0017-9310(94)00223-1

A block-correction aided strongly implicit solver for the five-point formulation of elliptic differential equations

CHARN-JUNG KIM and SUNG TACK RO†

Department of Mechanical Engineering, Seoul National University, Seoul 151-742, Korea

(*Received* 18 *February* 1994 *and in final form* 13 *July* 1994)

Abstract—An efficient and robust iterative solver was proposed to solve the five-diagonal matrix equations that arise from implicit discretization of two-dimensional thermal and fluid flow problems. By regarding the well-known ADI solver and the direct solver as the lowest and the highest levels of matrix decomposition, a systematic investigation was carried out to identify which level of matrix decomposition is then most suitable for the five-point formulation currently in use. Also, the existing block-correction procedure is implemented by deliberately evaluating the residual. In this way, both the high- and low-frequency errors are efficiently reduced so that the new iterative solver provides savings of about an order of magnitude in the computational cost. As the grid size increases, the new solver presented here turns out to be more powerful and more robust than other iterative solvers previously available.

INTRODUCTION

Numerical investigation in the field of thermal and fluid engineering frequently requires the solution of partial differential equations that reflect the conservation principles of physical quantities. Application of implicit discretization techniques then poses the task of solving a large set of algebraic equations that replace the original partial differential equations. There is no need to say that it is very important to adopt an *efficient* solver.

Solvers for simultaneous linear algebraic equations are normally classified into two groups, e.g. direct solvers and iterative solvers [1]. There exist in the literature a number of direct solvers, including the earlier methods in algebra such as the Cramer's rule and the Gaussian elimination. Two fast, direct methods for the solution of the Poisson and Laplace equations are the even–odd reduction method [2] and the fast Fourier transform method [3]. These two direct methods have been generalized later in refs. [4, 5]. A highly competitive direct block Gaussian elimination method is also available in ref. [6] for discrete Poisson problems. Nevertheless, the use of direct solvers is efficient only when the coefficient matrix is prescribed once and for all, as encountered in linear problems. For nonlinear problems, it is unwise to spend an excessive amount of effort on solving equations that are based only on tentative coefficients [7]. Therefore, under these circumstances, the iterative solvers are more widely used, in which the tentative coefficients are updated after sufficient iterations of the equation solver. Up to now, many iterative solvers have been developed, such as the Gauss–Seidel point-iterative method, the ADI (*A*lternating *D*irection *I*mplicit), the SIP (*S*trongly *I*mplicit *P*rocedure) [8], the MSI (*M*odified *S*trongly *I*mplicit) [9], the CSIP (*C*oupled *S*trongly *I*mplicit *P*rocedure) [10], the RL (right-to-left) and LR (left-to-right) solvers [11], and the SIS (*S*trongly *I*mplicit *S*olver) [12]. Other iterative solvers are also available for certain limited cases, e.g. the ICGM (*I*ncomplete *C*holesky-conjugate *G*radient *M*ethod) [13] is useful when the coefficient matrix is symmetric and positive definite. A brief review of the performance characteristics of these iterative solvers is given below.

The Gauss–Seidel point-iterative solver poses a low convergence rate because the boundary-condition information is transmitted at a rate of one grid interval per iteration. To overcome such a slow convergence, ADI employs the line-by-line TDMA (*T*ri-*D*iagonal *M*atrix *A*lgorithm) in alternating directions. For two-dimensional situations, a single iteration of ADI involves four sweeping directions (i.e. west-to-east, east-to-west, south-to-north, and north-to-south). Depending on problems, one may choose a smaller number of sweeping directions to improve the convergence rate, because the rate of propagation of information is dependent on the sweeping direction. Generally, the otherwise effective ADI solver shows poor performance when the sweeping direction is opposite to the flow.

The SIP [8] precludes the need for sweeping, but it is sensitive to a parameter for the matrix operation, called the cancellation parameter α. In the SIP, the five-diagonal coefficient matrix is approximately decomposed into a product of an upper triangular matrix and a lower one through the partial can-

†Author to whom correspondence should be addressed.

NOMENCLATURE

A	influence coefficient
$[A]$	coefficient matrix
CPU	computational cost unit, equal to the actual computing time for a single iteration of the SIS solver
entry	a set of nonzero elements in a matrix, equation (6)
$[G]$	auxiliary coefficient matrix
$\{\mathbf{H}\}$	modified source vector, equation (16)
i, j	index for the node of interest
I, J	number of nodes in x, y coordinates
$[L]$	lower triangular matrix
N	number of total grid points, $I \times J$
$\{\mathbf{q}\}$	source vector, equation (1)
Pr	Prandtl number
$\{\mathbf{R}\}$	residual vector, equation (5)
$\{\tilde{\mathbf{R}}\}$	residual vector for $\{\tilde{\phi}\}$
$R_{\max}$	maximum residual, $\mathrm{Max}_{i,j}\lvert R_{i,j}\rvert$
Ra	Rayleigh number
u, v	Cartesian velocities
$[U]$	upper triangular matrix.

Greek symbols

α	cancellation parameter
$\{\delta\}$	increment vector, equation (19)
$\delta_{\max}$	maximum change, $\mathrm{Max}_{i,j}\lvert\delta_{i,j}\rvert$
θ	temperature
ϕ	unknown field variable
$\{\phi^*\}$	guessed field vector
$\{\tilde{\phi}\}$	overrelaxed field vector, equation (18)
ψ	streamfunction
ω	relaxation factor or vorticity.

Superscripts

e	east
n	north
p	main grid point
s	south
Tr	transpose
w	west
*	guessed values
$^-$	additive-correction
$^\sim$	overrelaxation.

cellation procedure. To eliminate the effect of the partial cancellation, iteration is necessary and it is crucial to evaluate the residual for the next iteration. The RL and LR solvers [11], as variants of the SIP, can accommodate the nine-diagonal coefficient matrix, but they are also sensitive to the cancellation parameter α. Although the MSI [9] appears insensitive to the α value, contradicting results are reported on its performances [11]. The CSIP [10] is particularly applicable to the coupled set of algebraic equations and needs further improvement in efficiency.

The SIS [12] is based on a combination of SIP and ADI, taking the advantages of both while avoiding the disadvantages of each. In the SIS, no sweeping is necessary while the convergence rate is less sensitive to the iteration parameter. Compared to the SIP family of solvers, it does not require the residual to be evaluated and thus saves considerable CPU. The SIS showed good performances over other iterative solvers. Unfortunately, the performance test of the SIS was made only for relatively coarse grid systems of 21×21 resolution. As a result, it is uncertain that the SIS also works well with higher grid resolution.

In the present work, a new solver is proposed for the solution of simultaneous linear algebraic equations that arise from implicit discretization of two-dimensional field equations. The new solver proposed here is based on an elegant combination of the block-correction procedure of Settari and Aziz [14] and the matrix decomposition technique—for convenience it will be called the BASIS solver (*B*lock-correction *A*ided *S*trongly *I*mplicit *S*olver) hereinafter. Although not inherently limited, the five-diagonal coefficient matrix is considered here, since it is more prevalent among the numerical formulations currently in use. Depending on the choice of nonzero diagonals of the two factored matrices, three levels of matrix decomposition are considered and the convergence rate of each is examined. In the present BASIS, the matrix decomposition alleviates the short-wavelength residual components, whereas the block-correction procedure attenuates the long-wavelength residual components. Overall, the new BASIS solver shows performances an order of magnitude faster than other iterative solvers.

TASK TO BE UNDERTAKEN

Consider a two-dimensional domain composed of a set of $I \times J$ inner grid points (excluding the grid points on the boundaries of the domain) so that each point in the domain is designated by an index array (i, j) where $i = 1, 2, \ldots, I$ and $j = 1, 2, \ldots, J$. Implicit discretization of the differential equations frequently yields the following well-structured linear algebraic equations:

$$A^{\mathrm{w}}_{i,j}\phi_{i-1,j} + A^{\mathrm{s}}_{i,j}\phi_{i,j-1} + A^{\mathrm{p}}_{i,j}\phi_{i,j} + A^{\mathrm{n}}_{i,j}\phi_{i,j+1} + A^{\mathrm{e}}_{i,j}\phi_{i+1,j} = q_{i,j} \quad (1)$$

where

$$A^{\mathrm{e}}_{I,j} = A^{\mathrm{w}}_{1,j} = 0 \quad \text{for} \quad j = 1, 2, \ldots, J$$

$$A^{\mathrm{n}}_{i,J} = A^{\mathrm{s}}_{i,1} = 0 \quad \text{for} \quad i = 1, 2, \ldots, I. \quad (2)$$

Each index is repeated so as to generate a total of

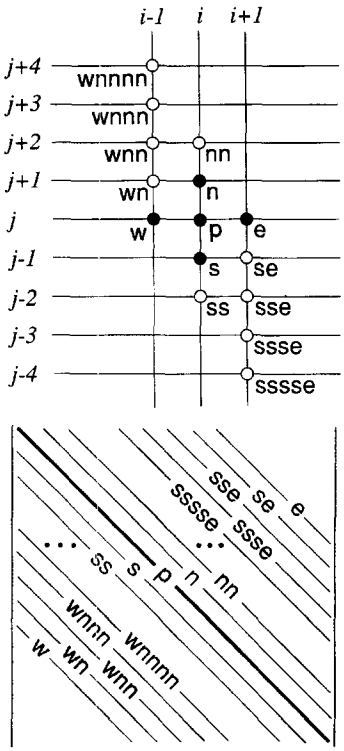

Fig. 1. The designations used for the node molecules and their matrix representation.

$N = I \times J$ linear algebraic equations. In compliance with the matrix operation, equation (1) is conveniently expressed in a form of matrix equation as

$$[A]\{\phi\} = \{\mathbf{q}\} \tag{3}$$

where $[A]$ is an $N \times N$ matrix with five diagonal coefficients; and both $\{\phi\}$ and $\{\mathbf{q}\}$ are $N \times 1$ column vectors, e.g. $\{\phi\} = (\phi_{1,1}, \phi_{1,2}, \ldots, \phi_{2,1}, \ldots, \phi_{I,J})^{\mathrm{Tr}}$. Given $[A]$ and $\{\mathbf{q}\}$, the mission is to find a solution vector $\{\phi\}$ that makes the residual defined as

$$R_{i,j} = q_{i,j} - A^{\mathrm{p}}_{i,j}\phi_{i,j} - A^{\mathrm{e}}_{i,j}\phi_{i+1,j} - A^{\mathrm{w}}_{i,j}\phi_{i-1,j} - A^{\mathrm{n}}_{i,j}\phi_{i,j+1} - A^{\mathrm{s}}_{i,j}\phi_{i,j-1} \tag{4}$$

or

$$\{\mathbf{R}\} = \{\mathbf{q}\} - [A]\{\phi\} \tag{5}$$

to vanish sufficiently over the entire domain.

For later reference, Fig. 1 shows the designations used for the node molecules and their corresponding diagonal locations when represented as matrix components. In addition, an *entry* of a matrix is defined as a set of nonzero diagonals of a given matrix, e.g. the five-diagonal coefficient matrix $[A]$ will be simply denoted by

$$\text{entry}\,[A] = (A^{\mathrm{w}}, A^{\mathrm{s}}, A^{\mathrm{p}}, A^{\mathrm{n}}, A^{\mathrm{e}}) \tag{6}$$

where A in the parentheses stands for the matrix of interest and the superscripts, such as w, refer to the locations of nonzero diagonals in the matrix $[A]$.

PREAMBLE: BLOCK-CORRECTION

Since the block-correction procedure [14] is one of the essential parts of the BASIS, its features are briefly described below. Subsequently, we explain how it can be implemented into the BASIS. The block-correction procedure was originally developed in an attempt to improve an intermediate solution $\{\tilde{\phi}\}$ by adding uniform corrections, e.g. $\bar{\phi}_i$ along lines of constant i, such that

$$\phi_{i,j} = \tilde{\phi}_{i,j} + \bar{\phi}_i. \tag{7}$$

Substituting this into equation (1) and summing over j, a tridiagonal matrix equation is obtained:

$$x^{\mathrm{p}}_i\bar{\phi}_i + x^{\mathrm{e}}_i\bar{\phi}_{i+1} + x^{\mathrm{w}}_i\bar{\phi}_{i-1} = x^{\mathrm{r}}_i \tag{8}$$

where

$$\begin{aligned} x^{\mathrm{p}}_i &= \Sigma_j(A^{\mathrm{p}}_{i,j} + A^{\mathrm{n}}_{i,j} + A^{\mathrm{s}}_{i,j}) \\ x^{\mathrm{e}}_i &= \Sigma_j A^{\mathrm{e}}_{i,j} \\ x^{\mathrm{w}}_i &= \Sigma_j A^{\mathrm{w}}_{i,j} \\ x^{\mathrm{r}}_i &= \Sigma_j \tilde{R}_{i,j}. \end{aligned} \tag{9}$$

In the above, $\tilde{R}_{i,j}$ is the residual defined by equation (4). Note that the boundary corrections, $\bar{\phi}_0$ and $\bar{\phi}_{I+1}$, are unnecessary since $x^{\mathrm{e}}_I = 0$ and $x^{\mathrm{w}}_1 = 0$. Hence, equation (8) can be solved efficiently by the TDMA. Then, the new field $\{\phi\}$ available through equation (7) exactly satisfies the overall conservation over the control-volume blocks of each constant-i line. In a similar manner, uniform corrections along constant-j lines can be found as well; therefore, there exist two uniform corrections, say $\bar{\phi}_i$ and $\bar{\phi}_j$.

The foregoing additive-correction is particularly useful in reducing the long-wavelength components of the residual that span, with same sign, many grid points. In the presence of long-wavelength residual components the convergence rate is dramatically reduced [15] and its influence is more pronounced for grid systems of high resolution.

Unfortunately, this useful additive-correction procedure has not been tested in the SIS or in the SIP family of solvers. For instance, a major advantage of SIS lies in that it needs no evaluation of the residual, which thus is incompatible with the block-correction procedure making use of the evaluated residual. Another efficient way of eliminating the long-wavelength residual components is using the multi-grid method [15, 16]. This technique utilizes the fact that a long wavelength relative to a fine mesh is short to a coarser mesh. Apparently, the multi-grid method needs further standardization for easier use. Due to a building-block character of a matrix solver, the present BASIS solver can also be incorporated into the multi-grid method to enhance the convergence rate further.

FORMULATION OF THE BASIS SOLVER

In this section, the new BASIS solver is presented with the aid of the block-correction procedure [14]

Table 1. The three levels of the L–U decomposition considered in this study: the entries of each matrix in equation (10) are listed below, where entry $[A] = (A^{w}, A^{s}, A^{p}, A^{n}, A^{e})$

Level of decomposition	Entry $[L]$	Entry $[U]$	Entry $[G]$
Level 1	(L^{w}, L^{s}, L^{p})	$(U^{p}=1, U^{n}, U^{e})$	(G^{wn}, G^{se})
Level 2	$(L^{w}, L^{wn}, L^{s}, L^{p})$	$(U^{p}=1, U^{n}, U^{se}, U^{e})$	(G^{wnn}, G^{sse})
Level 3	$(L^{w}, L^{wn}, L^{wnn}, L^{ss}, L^{s}, L^{p})$	$(U^{p}=1, U^{n}, U^{nn}, U^{sse}, U^{se}, U^{e})$	$(G^{wnnn}, G^{wnnnn}, G^{sssse}, G^{ssse})$

described above and the matrix decomposition technique. As a first step, let us examine how an auxiliary $N \times N$ matrix $[G]$ can be constructed such that $[A]+[G]$ allows for the well-known L–U decomposition, i.e.

$$[A]+[G] = [L][U] \tag{10}$$

where $[L]$ designates a lower triangular matrix and $[U]$ an upper one. In this respect, ADI may be interpreted to have

$$\text{entry}\,[G] = (-A^{w}, -A^{e}) \tag{11}$$

and $[A]+[G]$ thereby becomes a tridiagonal matrix. Such a situation may be regarded as Level-0 decomposition since ADI provides the lowest level of decomposition possible. Another limiting case of $[G] = [0]$ corresponds to direct matrix decomposition, which may be symbolically designated by Level-∞. Meanwhile, the decomposition levels associated with the available iterative solvers can be thought to lie in between the above two limiting cases. For an iterative solver to be efficient, it might be good for the sparse nature of each matrix to be retained as far as possible. However, there must be a trade-off between the total number of iterations required and the CPU spent per each iteration (e.g. direct solvers need only one iteration but at a large cost of CPU and storage). Unfortunately, this issue has not been investigated systematically in the previous studies.

In order to investigate which level of decomposition is most suitable for five-diagonal formulation of field equations, we decided to consider three different levels of decomposition, namely, levels 1–3. The entries of $[L]$, $[U]$ and $[G]$ matrices corresponding to each level of decomposition are shown in Table 1. It can be seen that, as the decomposition level increases, each matrix gradually loses its sparsity. This implies that the CPU required for a single iteration goes up at a higher decomposition level. The explicit expressions for the entries of $[L]$, $[U]$ and $[G]$ can be derived by term-by-term matching between the matrix elements of $[A]+[G]$ and $[L][U]$. The final results are summarized below.

Level-1 decomposition

$$\begin{aligned}
L^{w}_{i,j} &= A^{w}_{i,j}\\
L^{s}_{i,j} &= A^{s}_{i,j}\\
L^{p}_{i,j} &= A^{p}_{i,j} - L^{s}_{i,j}U^{n}_{i,j-1} - L^{w}_{i,j}U^{e}_{i-1,j}\\
U^{n}_{i,j} &= A^{n}_{i,j}/L^{p}_{i,j}\\
U^{e}_{i,j} &= A^{e}_{i,j}/L^{p}_{i,j}\\
G^{wn}_{i,j} &= L^{w}_{i,j}U^{n}_{i-1,j}\\
G^{se}_{i,j} &= L^{s}_{i,j}U^{e}_{i,j-1}.
\end{aligned} \tag{12}$$

Level-2 decomposition

$$\begin{aligned}
L^{w}_{i,j} &= A^{w}_{i,j}\\
L^{wn}_{i,j} &= -L^{w}_{i,j}U^{n}_{i-1,j}\\
L^{s}_{i,j} &= A^{s}_{i,j} - L^{w}_{i,j}U^{se}_{i-1,j}\\
L^{p}_{i,j} &= A^{p}_{i,j} - L^{s}_{i,j}U^{n}_{i,j-1} - L^{w}_{i,j}U^{e}_{i-1,j} - L^{wn}_{i,j}U^{se}_{i-1,j+1}\\
U^{n}_{i,j} &= (A^{n}_{i,j} - L^{wn}_{i,j}U^{e}_{i-1,j+1})/L^{p}_{i,j}\\
U^{se}_{i,j} &= -L^{s}_{i,j}U^{e}_{i,j-1}/L^{p}_{i,j}\\
U^{e}_{i,j} &= A^{e}_{i,j}/L^{p}_{i,j}\\
G^{wnn}_{i,j} &= L^{wn}_{i,j}U^{n}_{i-1,j+1}\\
G^{sse}_{i,j} &= L^{s}_{i,j}U^{se}_{i,j-1}.
\end{aligned} \tag{13}$$

Level-3 decomposition

$$\begin{aligned}
L^{w}_{i,j} &= A^{w}_{i,j}\\
L^{wn}_{i,j} &= -L^{w}_{i,j}U^{n}_{i-1,j}\\
L^{wnn}_{i,j} &= -L^{w}_{i,j}U^{nn}_{i-1,j} - L^{wn}_{i,j}U^{n}_{i-1,j+1}\\
L^{ss}_{i,j} &= -L^{w}_{i,j}U^{sse}_{i-1,j}\\
L^{s}_{i,j} &= A^{s}_{i,j} - L^{w}_{i,j}U^{se}_{i-1,j} - L^{wn}_{i,j}U^{sse}_{i-1,j+1} - L^{ss}_{i,j}U^{n}_{i,j-2}\\
L^{p}_{i,j} &= A^{p}_{i,j} - L^{s}_{i,j}U^{n}_{i,j-1} - L^{ss}_{i,j}U^{nn}_{i,j-2}\\
&\quad - L^{w}_{i,j}U^{e}_{i-1,j} - L^{wn}_{i,j}U^{se}_{i-1,j+1} - L^{wnn}_{i,j}U^{sse}_{i-1,j+2}\\
U^{n}_{i,j} &= (A^{n}_{i,j} - L^{s}_{i,j}U^{nn}_{i,j-1}\\
&\quad - L^{wn}_{i,j}U^{e}_{i-1,j+1} - L^{wnn}_{i,j}U^{se}_{i-1,j+2})/L^{p}_{i,j}\\
U^{nn}_{i,j} &= -L^{wnn}_{i,j}U^{e}_{i-1,j+2}/L^{p}_{i,j}\\
U^{sse}_{i,j} &= (-L^{s}_{i,j}U^{se}_{i,j-1} - L^{ss}_{i,j}U^{e}_{i,j-2})/L^{p}_{i,j}\\
U^{se}_{i,j} &= -L^{s}_{i,j}U^{e}_{i,j-1}/L^{p}_{i,j}\\
U^{e}_{i,j} &= A^{e}_{i,j}/L^{p}_{i,j}\\
G^{wnnn}_{i,j} &= L^{wn}_{i,j}U^{nn}_{i-1,j+1} + L^{wnn}_{i,j}U^{n}_{i-1,j+2}\\
G^{wnnnn}_{i,j} &= L^{wnn}_{i,j}U^{nn}_{i-1,j+2}
\end{aligned}$$

$$G^{\text{sssse}}_{i,j} = L^{\text{ss}}_{i,j} U^{\text{sse}}_{i,j-2}$$

$$G^{\text{ssse}}_{i,j} = L^{\text{s}}_{i,j} U^{\text{sse}}_{i,j-1} + L^{\text{ss}}_{i,j} U^{\text{se}}_{i,j-2}. \tag{14}$$

Now, with all the ingredients described so far, we are at the point of proposing the solution procedure of the BASIS solver. Suppose $\{\phi^*\}$ is an unconverged field at the previous iteration. By adding $[G]\{\phi\} = [G]\{\phi^*\}$ to equation (3), we have the following recursive relation:

$$[A+G]\{\phi\} = \{\mathbf{q}\} + [G]\{\phi^*\} \tag{15}$$

or

$$[L][U]\{\phi\} = \{\mathbf{H}\} \equiv \{\mathbf{q}\} + [G]\{\phi^*\} \tag{16}$$

which can be efficiently solved by applying the forward followed by backward elimination, i.e.

$$\{\phi\} = [U]^{-1}[L]^{-1}\{\mathbf{H}\}. \tag{17}$$

Since the column vector $\{\mathbf{H}\}$ contains the guessed solution $\{\phi^*\}$, the correct solution should be iteratively determined by updating $\{\mathbf{H}\}$ until a convergence criterion is satisfied. To further speed up the convergence rate, it is also good to employ the successive overrelaxation factor ω, as in other iterative solvers:

$$\{\tilde{\phi}\} = \{\phi^*\} + \omega\{\delta\} \tag{18}$$

where

$$\{\delta\} = \{\phi - \phi^*\} \tag{19}$$

is the increment vector and the vector $\{\phi\}$ is the solution obtained from equation (17). Note that in the SIS the relaxed vector $\{\tilde{\phi}\}$ is simply used to update $\{\mathbf{H}\}$ for the next iteration.

Due to the sparse nature of the matrices, the solution from equation (17) mainly reduces the short-wavelength residual components, while the long-wavelength residual components may survive up to a large number of iterations. As was mentioned earlier, such an unwelcomed situation can be avoided by employing the block-correction procedure, which, however, calls for the evaluation of the residual. Lee [12] points out that evaluating the residual is equivalent to performing a complete sweep of Gauss–Seidel point iteration and, therefore, is very time-consuming. In the present study, the computational cost for evaluating the residual is cut off by deliberately rearranging equation (15) as

$$\{\mathbf{q}\} - [A]\{\phi\} = [G]\{\phi - \phi^*\}$$

or

$$\{\mathbf{R}\} = [G]\{\delta\}. \tag{20}$$

A use of equation (20) possesses two notable advantages over using equation (4). First, there is no need to store the coefficient matrix $[A]$, thereby saving the storage considerably. Second, for Levels 1 and 2 decomposition, only two multiplications and one addition are necessary for each (i, j) location, since the matrix $[G]$ has only two nonzero diagonals (see Table 1). Compare this with using equation (4), where five multiplications and five additions are required. But for Level-3 decomposition, there exists about 20% savings in CPU for evaluating the residual, and therefore its use would be desirable only when the total number of iterations can be substantially reduced.

When the relaxation parameter ω differs from the unity, an exact expression for the residual pertaining to the relaxed vector $\{\tilde{\phi}\}$ can be derived by combining equations (3), (15) and (20):

$$\{\tilde{\mathbf{R}}\} = (1-\omega)\{\mathbf{R}^*\} + \omega[G]\{\delta\} \tag{21}$$

where $\{\mathbf{R}^*\} = \{\mathbf{q}\} - [A]\{\phi^*\}$ or $\{\mathbf{R}^*\} = [G]\{\delta^*\}$. But, due to the iterative nature of the solution procedure, being *exact* with these residuals seems to be unwise and hence $\{\tilde{\mathbf{R}}\}$ is approximated as

$$\{\tilde{\mathbf{R}}\} \cong \omega[G]\{\delta\}. \tag{22}$$

Then, by determining $\{\tilde{\mathbf{R}}\}$ as above, the relaxed vector $\{\tilde{\phi}\}$ from equation (18) is corrected with the aid of two uniform additive-corrections, $\bar{\phi}_i$ and $\bar{\phi}_j$. Accordingly, the previously guessed field vector $\{\phi^*\}$ is updated such that

$$\phi^{\text{new}}_{i,j} = \phi^*_{i,j} + \omega\delta_{i,j} + (\bar{\phi}_i + \bar{\phi}_j) \tag{23}$$

where the first correction $\omega\delta_{i,j}$ deals with the high-frequency errors and the second additive-correction $(\bar{\phi}_i + \bar{\phi}_j)$ eliminates low-frequency errors. It is noted that the coefficients necessary for the additive-correction procedure (such as $x^{\text{p}}_i, x^{\text{e}}_i, x^{\text{w}}_i$, etc.) need to be determined only once, as is the case with the L–U decomposition.

In the following, the basic steps for the BASIS solver are summarized.

Factorization stage:

(i) select decomposition level;
(ii) evaluate the coefficients necessary for additive corrections;
(iii) determine $[G]$, $[L]$ and $[U]$ matrices.

Iteration stage:

(1) evaluate $\{\mathbf{H}\} = \{\mathbf{q}\} + [G]\{\phi^*\}$;
(2) find $\{\phi\} = [U]^{-1}[L]^{-1}\{\mathbf{H}\}$ and $\{\tilde{\delta}\} = \omega\{\phi - \phi^*\}$;
(3) find $\{\mathbf{R}\} = [G]\{\tilde{\delta}\}$;
(4) solve for $\bar{\phi}_i$ and $\bar{\phi}_j$;
(5) find $\phi^{\text{new}}_{i,j} = \phi^*_{i,j} + \tilde{\delta}_{i,j} + (\bar{\phi}_i + \bar{\phi}_j)$;
(6) take $\{\phi\}^{\text{new}}$ as a newly-guessed field $\{\phi^*\}$ and go back to step 1.

The above iteration continues until a preselected convergence criterion is satisfied. Lee [12] suggested the convergence criterion

$$\frac{\delta_{\max}}{|\Delta\phi|_{\max}} \leqslant \text{TOL} \tag{24}$$

where $\delta_{\max} = \text{Max}_{i,j}|\delta_{i,j}|$; TOL is a prescribed tol-

erance; $|\Delta\phi|_{\max}$ the difference between the maximum and minimum values of $\phi_{i,j}$. However, for the problems involving strongly anisotropic coefficients (e.g. $A^{e}_{i,j} \gg A^{n}_{i,j}$), $\delta_{\max}$ may remain very small even when the corresponding solution is far from converged [17]. In such situations, the use of equation residual, defined as $R_{\max} = \mathrm{Max}_{i,j}|R_{i,j}|$, is more appropriate to declare the convergence. For comparison with the available results of other iterative solvers, both the criteria of $\delta_{\max}$ and $R_{\max}$ are adopted here for the measure of the convergence.

COMPARISON WITH OTHER SOLVERS

In this section, the implication of different levels of decomposition is addressed and the difference between BASIS and other iterative solvers is outlined.

It can be first stated that both the SIP [8] and SIS [12] basically involve Level-1 decomposition defined here, i.e. entry $[G] = (G^{wn}, G^{se})$, whereas the MSI [9] is based on Level-2 decomposition with entry $[G] = (G^{wnn}, G^{sse})$. In the SIP family, the partial cancellation procedure is employed to nullify the influence of $[G]$ and, therefore, the matrix $[G]$ is never used during iteration procedure. In fact, the SIP family make use of the following recursive equation:

$$[A+G]\{\phi\} = [A+G]\{\phi^*\} + \{\mathbf{q}\} - [A]\{\phi^*\} \tag{25}$$

or

$$[A+G]\{\delta\} = \{\mathbf{R}^*\}. \tag{26}$$

It should be emphasized that the above iteration strategy of the SIP family completely disregards the otherwise attractive sparsity of the $[G]$ matrix. By contrast, in the SIS and BASIS, the sparse $[G]$ matrix is fully utilized with the successive overrelaxation factor. Although BASIS stems from SIS, BASIS is distinguished from SIS due to its necessity of evaluating the residual and its flexibility in the choice of decomposition levels. Also, BASIS contains all the beneficial features of SIS, but excludes disadvantages of SIS. For instance, the L–U decomposition needs to be performed only once, even though the relaxation parameter may change from iteration to iteration. In addition, the storage can be saved by replacing the elements of $[A]$ by the corresponding ones in $[L]$ and $[U]$. As can be seen later, SIS shows poor performance for high grid resolution, but BASIS becomes more powerful in such cases.

In Level-1 decomposition, the values at the northwest and the southeast points (i.e. ϕ^{wn} and ϕ^{se}) are either partially cancelled (SIP) or guessed (SIS and Level-1 BASIS). For strongly convective problems, these two points would fairly affect the main ϕ^{p} value and hence the effect of poorly guessed values should be eliminated via many iterations.

But for the case of Level-2 decomposition (MSI and Level-2 BASIS), the guess points are further removed from the (i, j) location so that both the ϕ^{wnn} and ϕ^{sse} values are of major interest. Therefore, Level-2 decomposition is considered to be more strongly implicit than Level-1 decomposition, thereby requiring fewer iterations. Level-2 decomposition, however, accompanies an increased CPU per each iteration, since the entries of $[L]$ and $[U]$ have one more element than Level-1 decomposition.

To the authors' knowledge, no iterative solvers based on Level-3 decomposition have appeared up to date. In the present Level-3 BASIS, the guess points are located farther away from the main node—i.e. ϕ^{wnnn}, ϕ^{wnnnn}, ϕ^{ssse}, and ϕ^{sssse}. Unfortunately, the $[G]$ matrix has twice as many nonzero elements and the CPU for evaluating the residual is doubled. Despite an increase in CPU per iteration, Level-3 decomposition poses a more strongly implicit formulation than Levels 1 and 2 decomposition. It can be anticipated that there should exist a trade-off between the total number of iterations and the CPU per iteration. The performance of each level of BASIS is examined and compared with those of other iterative solvers in the next section.

TEST OF THE BASIS SOLVER

As was emphasized earlier, the selection of the iterative solution procedure is based on the strategy of extracting sufficient information from the algebraic equations with tentative coefficients. Therefore, the convergence behavior, especially at an early stage of iterations, say within the first 10 iterations, needs to be examined between different solvers. This kind of performance test is presented in example 2 where the vorticity equation is selected as the target equation.

Example 1. *Heat conduction problem*

Consider a two-dimensional heat conduction problem on a rectangular domain. The top and bottom walls are insulated, while the left and right walls are maintained isothermal at $\theta = 1$ and $\theta = -1$, respectively. The same problem has been considered to test the RL and LR solvers [11] and the SIS solver [12], taking account of the effect of the inclined side walls. Since the present study is mainly concerned with the five-point formulation, only the case with the vertical side walls is considered to test the performance of BASIS against other iterative solvers. Specifically, the width-to-height ratio is fixed to be 1.25. As in refs. [11, 12], the governing equation ($\partial^2\theta/\partial x^2 + \partial^2\theta/\partial y^2 = 0$) is discretized with the central difference scheme on several uniform grid systems. The grid resolutions were varied from 20×20, 45×45, 100×100 to 200×200. In comparing the performances between different solvers, the number of iterations itself is not important because the CPU per iteration differs from one another [12]. Therefore, the actual CPU is selected as a measure of the performance. For convenience, one CPU is defined as the computational cost required by a single iteration of SIS. Figure 2 shows the variation

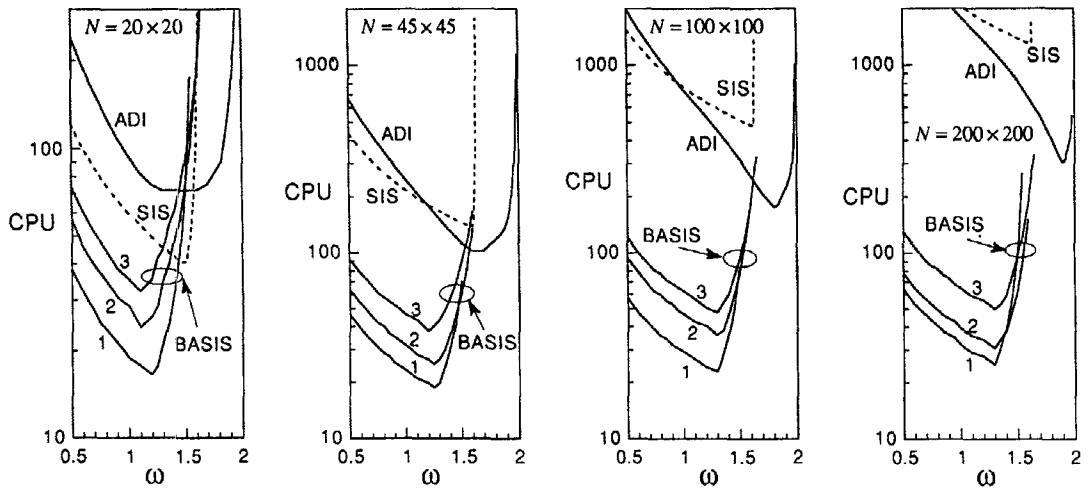

Fig. 2. The CPU required by ADI, SIS and BASIS solvers with various grid sizes. The convergence criterion used is $\delta_{\max} \leqslant 5 \times 10^{-5}$. The numbers in each plot denote the corresponding level of decomposition in BASIS.

of CPU with respect to the relaxation parameter for all the grid resolutions examined. Convergence was assumed when the condition $\delta_{\max} \leqslant 5 \times 10^{-5}$ was satisfied. Since for the present problem the SIS solver was known to be superior to other iterative solvers [12], comparison is made here only between BASIS, SIS and ADI. Lee [12] has claimed that SIS shows better performance than ADI. The results shown in Fig. 2 substantiate his opinion, but only for coarse grid resolution. It is evident that at higher grid resolution the performance of SIS deteriorates and is worse than that of ADI for the case of 200×200 grid system. By contrast, it can be seen that all the three levels of BASIS converge much faster than SIS or ADI. However, BASIS has a narrower range of effective ω than SIS and ADI. This is due to the fact that for large ω the intermediate field variables are exposed to severe additive-corrections, thereby resulting in a shrinking range of effective ω. The minimum CPU of BASIS occurs at values of ω less than 1.3, while ADI possesses a wide range of effective ω. Nevertheless, even when compared with the optimized CPUs of SIS and ADI, an order of magnitude saving in CPU is experienced by BASIS near $\omega = 1$. Note that Level-1 BASIS shows best performance over Levels 2 and 3. In this example, the values of θ^{wn} and θ^{wnn} are close to each other during iteration. For this reason, Level-2 and Level-3 decompositions, though more implicit than Level-1 decomposition, are not at their full power. The performance of each level of BASIS is investigated in the next example in the presence of a large variation in the field variable due to strong convection.

Example 2. Natural convection problem

In testing SIS, Lee [12] considered steady-state natural convection in a square cavity as one of the example problems. In the present work, the same problem is also adopted to test the performance of BASIS. Consider the governing equations in the streamfunction–vorticity formulation:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega \tag{27}$$

$$u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = Pr\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) - Pr \cdot Ra \frac{\partial \theta}{\partial x} \tag{28}$$

$$u\frac{\partial \theta}{\partial x} + v\frac{\partial \theta}{\partial y} = \frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2} \tag{29}$$

where

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}. \tag{30}$$

The boundary conditions are the same as those studied in Lee [12]:

$$\psi(0,y) = \psi(1,y) = \psi(x,0) = \psi(x,1) = 0$$

$$\theta(0,y) - 1 = \theta(1,y) = 0$$

$$\frac{\partial \theta}{\partial y}(x,0) = \frac{\partial \theta}{\partial y}(x,1) = 0. \tag{31}$$

The governing equations are discretized using the power law scheme [7] and the resulting set of algebraic equations are solved on uniform grids for two different Ra numbers and $Pr = 0.7$. Once the correct solutions for ψ, ω and θ are obtained, the vorticity field is set to zero and solved again using the correct ψ and θ fields. This means that only the vorticity equation (28) is adopted for the CPU test. The same approach has been employed in Lee [12], mainly because computational efforts for solving each governing equation are not necessarily the same.

For a convergence criterion of $\delta_{\max} \leqslant 5 \times 10^{-5}$ and $Ra = 10^3$, Fig. 3 exposes the CPU required by various

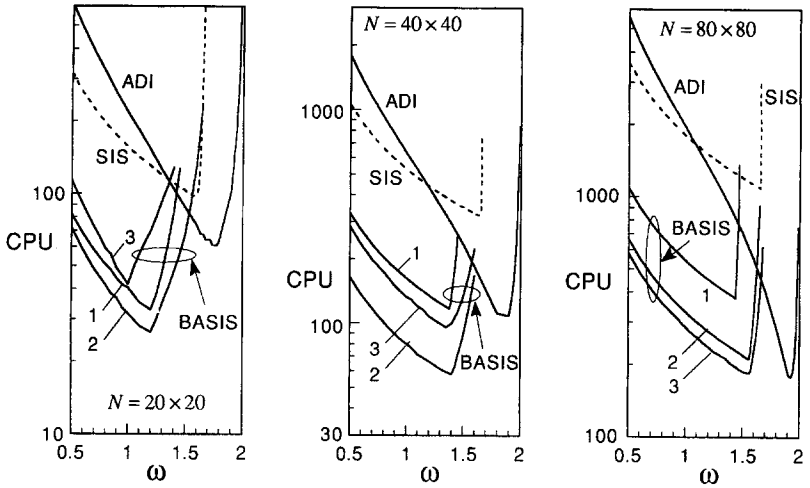

Fig. 3. The CPU required by ADI, SIS and BASIS solvers for example 2 with $\delta_{max} \leqslant 5 \times 10^{-5}$ and $Ra = 10^3$.

solvers with respect to the relaxation parameter ω. Three grid systems of coarse, moderate and high resolutions (i.e. 20×20, 40×40 and 80×80) are considered.

Unlike the case of heat conduction, each level of BASIS shows different performance characteristics, depending on the grid resolution. For coarse to moderate resolution, Level-2 BASIS appears to be the most efficient solver. However, Level-3 BASIS turns out to be best at high resolution. In the presence of convection, the values at the guess points would significantly affect the convergence rate. When the guess points are located farther away, the convergence would be accelerated. Remember that Level-3 BASIS, however, requires twice as much CPU to evaluate the residual. As a result, the overall performance of Level-3 BASIS is only about 15% better than that of Level-2 BASIS, even with 80×80 grid resolution. It is noted that ADI works very well, especially at large values of ω. For the 80×80 case, ADI seems to be comparable to BASIS when both solvers are optimized. However, in using ADI, the relaxation parameter is normally set small enough to avoid divergence. For example, for problems involving strong convection, smaller values of ω are required in ADI [17]. It is interesting to note that the effective range of ω for BASIS becomes wider as the grid size increases. This is the most favorable feature of BASIS, namely that BASIS is more robust at higher grid resolution.

Figure 4 displays the performance test for the strong convection case with $Ra = 10^6$. The convergence condition is the same as in Fig. 3. A feature worthy of note is that the range of effective ω does not change much for the BASIS solver, compared with the case of $Ra = 10^3$. This means that BASIS is less sensitive to the effect of strong convection and remains robust. On the contrary, the optimum value of ω in ADI is dramatically reduced to $\omega \cong 1.3$ from $\omega \cong 1.8$, as the Rayleigh number changes from $Ra = 10^3$ to $Ra = 10^6$. Therefore, in using ADI care should be exercised in selecting the ω value, while BASIS remains robust regardless of strengths of convection. Although SIS has a similar merit, its convergence rate is far from satisfactory, as is evident in Figs. 3 and 4.

Interestingly, decomposition beyond Level 2, in general, does not improve the convergence rate much. For coarse to moderate grids the reduction in the total number of iterations gained by Level-3 BASIS was too small to compensate for the increased CPU per iteration. Also for a high-resolution grid, the remoteness of the guess points (ϕ^{wnnn}, ϕ^{wnnnn}, ϕ^{ssse} and ϕ^{sssse}) from the main node tends to deteriorate as the convection strength increases (see Fig. 4). Therefore, we are led to conclude that matrix decomposition of Level-2 is most suitable for the five-point formulation in view of computational efficiency.

As was mentioned, the use of iterative solver is desirable in handling nonlinear problems, because the algebraic equations are based on tentative coefficients.

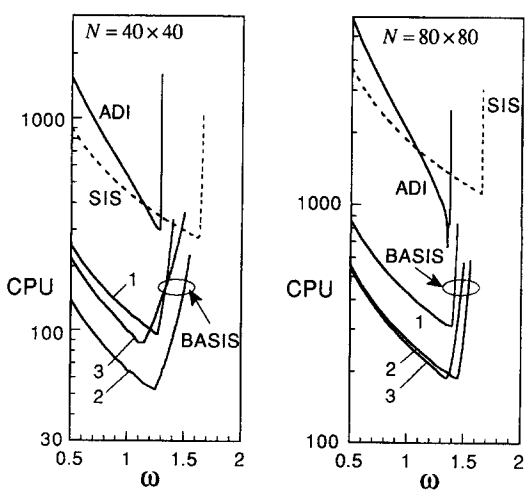

Fig. 4. The CPU test of various solvers for example 2 with $\delta_{max} \leqslant 5 \times 10^{-5}$ and $Ra = 10^6$.

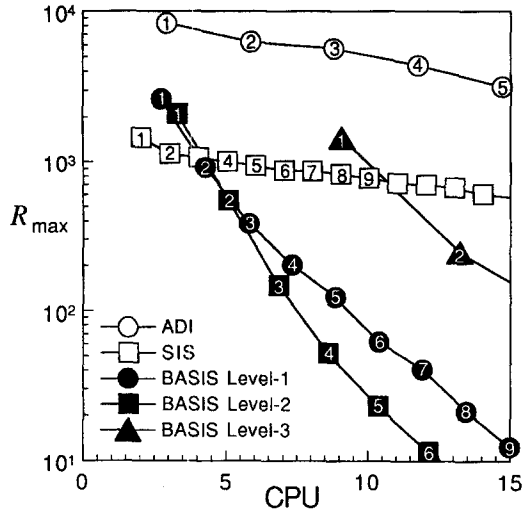

Fig. 5. The convergence history at an early stage of iteration for example 2: $Ra = 10^6$ and $N = 40 \times 40$. The numbers inside each symbol denote the performed iterations with each solver.

In order to evaluate the performance of each solver at an early stage of iterations (say within 10 iterations), the equation residual at each iteration step was printed for the case of $Ra = 10^6$ and $N = 40 \times 40$. Figure 5 exhibits the early history of convergence corresponding to each solver. The results corresponding to ADI and SIS are obtained with their optimized values of relaxation parameter (i.e. $\omega = 1.28$ and $\omega = 1.62$, respectively; see Fig. 4). However, the results of BASIS correspond to the case of $\omega = 1$. For SIS and BASIS, the CPU spent on decomposition is included in the first iteration number. Table 2 lists the CPU required by ADI, SIS and BASIS in the L–U decomposition and in performing a single iteration. When the first iteration is finished, Level-3 BASIS spends the largest CPU as can be expected, while SIS needs the smallest CPU. However, SIS provides a decreasing rate of residual as slow as ADI. In fact, the decreasing rate of the residual per iteration would serve as a valuable index to determine the performance of an equation solver. In this regard, about 2–3, or a few more, iterations with Level-2 BASIS seems to be suitable for treating nonlinear problems from the standpoint of numerical efficiency.

Table 2. The CPU required by ADI, SIS and BASIS in the L–U decomposition and in a single iteration for the case of $N = 40 \times 40$

Solvers	Decomposition	One iteration
ADI	—	2.941
SIS	1.045	1.000
Level-1 BASIS	1.219	1.529
Level-2 BASIS	1.568	1.759
Level-3 BASIS	4.966	4.129

CONCLUSION

A new BASIS solver was proposed to solve the simultaneous linear algebraic equations that frequently arise from implicit discretization of field equations in thermal and fluid engineering. A systematic investigation was carried out to identify which level of matrix decomposition was most suitable for the five-point formulation of two-dimensional field equations. This was done by regarding the ADI solver and the direct solver as the lower and upper bounds of decomposition levels. In this respect, three different levels of the L–U decomposition were examined. Overall, Level-2 decomposition showed good performance compared with other levels of decomposition. Conceptually, the Level-2 BASIS solver is indebted to the previously available solution techniques, being an elegant combination of ADI, MSI, SIS and block-correction procedure. When tested for two example problems using scalar computers, the present BASIS solver showed a performance about an order of magnitude faster than other iterative solvers.

Acknowledgments—The authors are grateful for the financial support from the Turbo and Power Machinery Research Center at Seoul National University.

REFERENCES

1. R. H. Pletcher, W. J. Minkowycz, E. M. Sparrow and G. E. Schneider, Overview of basic numerical methods. In *Handbook of Numerical Heat Transfer* (Edited by W. J. Minkowycz *et al.*), pp. 1–88. John Wiley, New York (1988).
2. O. Buneman, A compact non-iterative Poisson solver, Institute for Plasma Research, SUIPR Report 294, Stanford University, CA (1969).
3. R. W. Hockney, A fast direct solution of Poisson's equation using Fourier analysis, *J. Ass. Comput. Mach.* **12**, 95–113, (1965).
4. B. L. Buzbee, G. H. Golub and C. W. Nielson, On direct method for solving Poisson's equations, *SIAM J. Numer. Anal.* **7**, 627–656 (1970).
5. J. A. George, The use of direct methods for the solution of the discrete poisson equation on non-rectangular regions, Computer Science Report 70–159, Stanford University, CA (1970).
6. G. A. Osswald and K. N. Ghia, Study of unsteady incompressible flow using non-uniform curvilinear grids, time marching and a direct method. In *Multigrid Methods* (Edited by H. Lomax). NASA CP-2202 (1981).
7. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*. Hemisphere, New York (1980).
8. H. L. Stone, Iterative solution of implicit approximations of multidimensional partial differential equations, *SIAM J. Numer. Anal.* **5**, 530–558 (1968).
9. G. E. Schneider and M. Zedan, A modified strongly implicit procedure for the numerical solution of field problems, *Numer. Heat Transfer* **4**, 1–19 (1981).
10. M. Zedan and G. E. Schneider, A coupled strongly implicit procedure for velocity and pressure computation in fluid flow problems, *Numer. Heat Transfer* **8**, 537–557 (1985).
11. M. Peric, Efficient semi-implicit solving algorithm for nine-diagonal coefficient matrix, *Numer. Heat Transfer* **11**, 251–279 (1987).

12. S.-L. Lee, A strongly implicit solver for two-dimensional elliptic differential equations, *Numer. Heat Transfer* **16B,** 161–178 (1989).
13. D. S. Kershaw, The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations, *J. Comput. Phys.* **26,** 43–65 (1978).
14. A. Settari and K. Aziz, A generalization of the additive correction methods for the iterative solution of matrix equations, *SIAM J. Numer. Anal.* **10,** 506–521 (1973).
15. A. Brandt, Multi-level adaptive solutions to boundary-value problems, *Math. Comput.* **31,** 333–390 (1977).
16. B. R. Hutchinson and G. D. Raithby, A multigrid method based on the additive correction strategy, *Numer. Heat Transfer* **9,** 511–537 (1986).
17. G. D. Raithby and G. E. Schneider, Elliptic systems: finite difference method II. In *Handbook of Numerical Heat Transfer* (Edited by W. J. Minkowycz *et al.*), pp. 241–291. John Wiley, New York (1988).