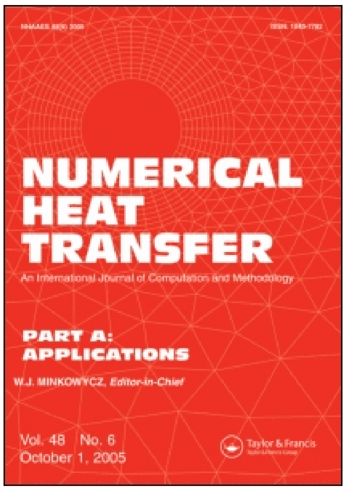


Numerical Heat Transfer, Part A: Applications
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713657973

KRYLOV SUBSPACE SOLVERS IN PARALLEL NUMERICAL COMPUTATIONS OF PARTIAL DIFFERENTIAL EQUATIONS MODELING HEAT TRANSFER APPLICATIONS

B. V. Rathish Kumar[a]; Bipin Kumar[a]; Shalini[a]; Mani Mehra[a]; Peeyush Chandra[a]; V. Raghvendra[a]; R. K. Singh[b]; A. K. Mahindra[b]
[a] Indian Institute of Technology, Kanpur, U.P., India [b] BARC, Mumbai, India

To cite this Article Kumar, B. V. Rathish , Kumar, Bipin , Shalini, Mehra, Mani , Chandra, Peeyush , Raghvendra, V. , Singh, R. K. and Mahindra, A. K.(2004) 'KRYLOV SUBSPACE SOLVERS IN PARALLEL NUMERICAL COMPUTATIONS OF PARTIAL DIFFERENTIAL EQUATIONS MODELING HEAT TRANSFER APPLICATIONS', Numerical Heat Transfer, Part A: Applications, 45: 5, 479 — 503

To link to this Article: DOI: 10.1080/10407780490269094

URL: http://dx.doi.org/10.1080/10407780490269094

Numerical Heat Transfer, Part A, 45: 479–503, 2004
Copyright © Taylor & Francis Inc.
ISSN: 1040-7782 print/1521-0634 online
DOI: 10.1080/10407780490269094

KRYLOV SUBSPACE SOLVERS IN PARALLEL NUMERICAL COMPUTATIONS OF PARTIAL DIFFERENTIAL EQUATIONS MODELING HEAT TRANSFER APPLICATIONS

B. V. Rathish Kumar, Bipin Kumar, Shalini, Mani Mehra, Peeyush Chandra, and V. Raghvendra
Indian Institute of Technology, Kanpur, U.P., India

R. K. Singh and A. K. Mahindra
BARC, Mumbai, India

In this study, parallel numerical algorithms for Krylov methods such as GMRES(k), Bi-CGM, Bi-CGSTAB, etc., for handling large-scale linear systems resulting from finite-difference analysis (FDA) and finite-element analysis (FEA) of coupled nonlinear partial differential equations (PDEs) describing problems in heat transfer applications are discussed. Parallel code has been successfully implemented on an eight-noded cluster under ANULIB message-passing library environment. Bi-CGM and ILU-GMRES(k) are found to give good performance for linear systems resulting from FEA, whereas Bi-CGSTAB is seen to give good performance with linear systems resulting from FDA.

1. INTRODUCTION

In numerical modeling of heat transfer and fluid flow, one generally employs numerical schemes based on methods such as finite differences, finite elements, spectral elements, etc., to solve the associated coupled nonlinear partial differential equations (PDEs). In such numerical studies PDEs are converted to algebraic systems and iterative methods are commonly used to solve these large-scale linear systems. These iterative schemes are classified as stationary and nonstationary methods. In stationary iterative methods the current value of a variable depends only on the immediate previous level, whereas in nonstationary methods current variable values are updated based on several of the previous iteration values. Under the first category we have methods such as Gauss-Seidel, Jacobi, successive overrelaxation (SOR), alternating direction implicit (ADI) scheme, etc. Methods such as conjugate gradient (CGM), conjugate gradient squared (CGS), conjugate gradient residual (CGR), biconjugate gradient (Bi-CGM), biconjugate gradient stabilized (Bi-CGSTAB), generalized minimal residual [GMRES(k)], and quasi minimal residual (QMR)

Received 13 December 2002; accepted 18 August 2003.

Address correspondence to B. V. Rathish Kumar, Department of Mathematics, Indian Institute of Technology, Kanpur 208016, India. E-mail: bvrk@iitk.ac.in

NOMENCLATURE

For Algorithms

A	matrix of order $n \times n$ in $Ax = b$ system
b	right-hand-side vector (dimension $n \times 1$)
$k_{\max}$	maximum number of iterations
p	search direction used for new approximation for solution vector x
PE_i	Processing element i
r	residual vector ($= b - Ax$)
u	residual vector used in GMRES(k) algorithm ($= b - Ax$)
x	solution vector (dimension $n \times 1$)
$\lVert y \rVert$	2-norm of vector y of dimension $n \times 1$, $\left[= \left(\sum_{i=1}^{n} \lvert y_i^2 \rvert\right)^{1/2}\right]$
α	step size used in updating the solution vector in new search direction
β	parameter used in updating the search direction
$\tilde{\beta}$	parameter used in GMRES(k) for updating residual vector
ε	given tolerance level
ρ	residual norm ($= \lVert r \rVert^2$)

Subscripts

k	indicates value at the kth iterate
0	indicates initial value
$\hat{}$	indicates dual vector

For Test Problems

Gr*	modified Grashof number
Nu	Nusselt number $\left[= \int_0^1 (\partial T/\partial x)\, dx\right]$
Ra	Rayleigh number (thermal forces/viscous forces)
T	temperature
x, y	Cartesian coordinates
ψ	stream function

methods, etc., are based on Krylov subspaces [1, 2] and are nonstationary. Iterative methods are easy to implement, but the computational efficiency is less robust and more unpredictable than those of direct methods. To improve the speed of convergence, multigrid methods have been developed [3]. Multigrid methods use several sets of grids with different coarseness to accelerate the propagation of iteration information. However, multigrid methods are difficult to implement in programming, particularly when a complex computational domain is considered. Krylov subspace-based methods [4] can be implemented independent of grid setup and hence are desirable in several engineering applications [5–12]. The convergence rates of these iterative methods can be improved by preconditioning and/or scaling of the equations. Incomplete L-U (ILU) factorization [1] is one of the popular preconditioners, which is widely in use. It is well known that not all Krylov subspace-based methods are equally applicable to all linear systems resulting from finite-element analysis of PDEs modeling flow and heat transfer processes in heat transfer applications. Generally the solutions of very large-scale systems resulting in the numerical study of 3-D heat transfer related problems are computationally very demanding and time-consuming on sequential computing systems. Parallel or distributed computations of the solution on a cluster of PCs (i.e., on Beowolf clusters) are one of the effective ways to carry out such large-scale computations. Different Krylov subspaces methods are amenable to different degrees of parallelism. Hence one has to develop suitable parallel algorithms and measure their performance.

In this study we investigate the power of various Krylov subspace solvers such as the conjugate gradient method (CGM), biconjugate gradient method (Bi-CGM), biconjugate gradient stabilized method (Bi-CGSTAB), generalized minimal residual method [GMRES(k)] etc., for parallel computation of linear systems resulting from finite-difference and finite-element analysis of partial differential equations modeling heat transfer applications. For this purpose, three different test problems in heat transfer applications are considered. The first of these test problems is related to the nonlinear reaction-diffusion problem and the last two problems are related to convection in porous enclosures under Darcian and non-Darcian assumptions. All the parallel computations are carried out on eight-noded PC cluster under ANULIB, a parallel communication interface library. Parallel algorithms and parallel implementation strategies essential for the distributed computation on a PC cluster are evolved. The performance of various Krylov subspace solvers is measured in terms of speed-up and efficiency factors. Further streamline and isotherm contours depicting the flow and temperature fields are presented. The article is organized under the following six sections: (1) introduction, (2) Krylov subspace methods, (3) parallel computing environments, (4) parallelization strategies and algorithms, (5) test problems and numerical schemes with results and discussion, and (6) conclusions.

2. KRYLOV METHODS AND THE MINIMIZATION PROPERTY

Stationary iterative methods can be expressed in the simple form as $x^k = Bx^{k-1} + C$, where neither B nor C depends on iterative count k. Nonstationary methods minimize at the kth iteration some measure of error over the affine space $x_0 + K_k$, where x_0 is the initial iterate and K_k is the kth Krylov subspace method defined as

$$K_k = \text{span}\,\{r_0, Ar_0, \ldots, A^{k-1}r_0\} \qquad \text{for } k \geq 1$$

where, the residual r_0 is given by $r_0 = b - Ax_0$. Details of some commonly used Krylov methods are provided below.

Conjugate Gradient Method

The CGM is an effective method for symmetric positive-definite systems. The method proceeds by generating vector sequences of iterates (i.e., successive approximations to the solution), residuals corresponding to the iterates, and search directions used in updating the iterates and residuals. In every iteration of the method, the inner products are found in order to update the scalars which are defined to make the sequences satisfy certain orthogonal conditions. Details of the iteration scheme are provided below:

Given A, b, $K_{\max}$ (maximum number of iterations), ε (tolerance) do the following:

```
1. Set $r_0 = b - Ax_0$, $\rho_0 = \|r_0\|_2^2$, $k = 0$.
2. Do while $(\rho_k)^{1/2} > \varepsilon\|b\|_2$
   (a) If $k = 0$ then $p_0 = r_0$,
       else $\beta_k = \dfrac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$ and $p_k = r_k + \beta_k\, p_{k-1}$
   (b) $w_k = Ap_k$
   (c) $\alpha_k = \dfrac{r_k^T r_k}{p_k^T w_k}$
   (d) $x_{k+1} = x_k + \alpha_k p_k$
```

```
    (e) $r_{k+1} = r_k - \alpha_k w_k$
    (f) $\rho_{k+1} = \|r_{k+1}\|_2^2$
    (g) $k = k + 1$
```

Note that the matrix A itself need not be formed or stored; only a routine for the matrix–vector product is required. It is for this reason that Krylov subspace methods are often called matrix-free methods.

Biconjugate Gradient Method

The CGM is not suitable for nonsymmetrical systems because the residual vectors cannot be made orthogonal with short recurrences. The Bi-CGM replaces the orthogonal sequences of residuals by two mutually orthogonal sequences at the cost of no longer providing a minimization. Here the update relations are based on A as well as on A^T. Details of the Bi-CGM are provided below.

```
1. $r_0 = b - Ax_0$
2. Select $\hat{r}_0$
3. For $k = 0, 1, 2, \ldots$ until convergence do
   If $k = 0$ then
      (a) $p_0 = r_0$
      (b) $\hat{p}_0 = \hat{r}_0$
   else
      (c) $\beta_k = \frac{\hat{r}_k^T r_k}{\hat{r}_{k-1}^T r_{k-1}}$
      (d) $p_k = r_k + \beta_k p_{k-1}$
      (e) $\hat{p}_k = r_k + \beta_k \hat{p}_{k-1}$
   end if
4. $\alpha_k = \frac{\hat{r}_k^T r_k}{\hat{p}_k^T A p_k}$
5. $x_{k+1} = x_k + \alpha_k p_k$
6. $r_{k+1} = r_k - \alpha_k A p_k$
7. $\hat{r}_{k+1} = \hat{r}_k - \alpha_k A^T \hat{p}_k$
   end for
```

Biconjugate Gradient Stabilized Method

The Bi-CGSTAB was developed to solve nonsymmetrical linear systems while avoiding the often-irregular convergence pattern of the conjugate gradient squared method. This method also produces iteratively sequences of approximations x_k, residuals r_k, and search directions p_k. The scalar α_k, β_k are computed such that both Ax_k and r_k are orthogonal to the Krylov subspace $K_k(A^T; r_0)$ of order k. Details of the algorithm are provided below.

```
1. Compute $r_0 = b - Ax_0$ for some initial guess $x_0$
2. Choose $\hat{r}_0$ (for example, $\hat{r}_0 = r_0$)
3. For $k = 0, 1, 2, \ldots$ until convergence do
   If $(k = 0)$ then
```

```
        (a) $p_0 = r_0$
    else
        (b) $\beta_k = \frac{\alpha_{k-1}}{\omega_{k-1}} \frac{\hat{r}_0^T r_k}{\hat{r}_0^T r_{k-1}}$
        (c) $p_k = r_k + \beta_k(p_{k-1} - \omega_{k-1} A p_{k-1})$
4. $\alpha_k = \frac{\hat{r}_0^T r_k}{\hat{r}_0^T A p_k}$
5. $r_{k+1/2} = r_k - \alpha_k A p_k$
6. $\omega_k = \frac{r_{k+1/2}^T A r_{k+1/2}}{(A r_{k+1/2})^T (A r_{k+1/2})}$
7. $r_{k+1} = r_{k+1/2} - \omega_k A r_{k+1/2}$
8. $x_{k+1} = x_{k+1/2} + \alpha_k p_k + \omega_k r_{k+1/2}$
    end for
```

Generalized Minimal Residual Method [GMRES(k)]

The GMRES(k) is an extension of MINRES [1, 2] (which is applicable only to symmetric systems) to unsymmetrical systems. Like the MINRES, it generates a sequence of orthogonal vectors, but in the absence of symmetry this can no longer be done with short sequences; instead all previously computed vectors in the orthogonal sequence have to be retained. For this reason, "restarted" versions of the method are used. In the conjugate gradient method, the residuals form an orthogonal basis for the space Span $\{r_0, Ar_0, A^2 ro, \ldots\}$. In the GMRES(k), this basis is formed explicitly:

```
$w_i = Av_i$
For $k = 1$ to $i$
    $w_i = w_i - (w_i, v_k)v_k$
end
    $v_{i+1} = \frac{w_i}{\|w_i\|}$
```

This is modified Gram-Schmidt orthogonalization.

Applied to Krylov sequence (A^k;r_0), this orthogonalization is called the Arnoldi method. The inner product (w_i, v_k) and $\|w_i\|$ are stored in a Hessenberg matrix. The GMRES(k) iterations are constructed as

$$x_i = x_{i-1} + y_1 v_1 + y_2 v_2 + \cdots + y_i v_i$$

where the coefficient y_i have been chosen to minimize the residual norm $\|b - Ax_i\|$. The GMRES(k) algorithm has the property that this residual norm can be computed without the iterate having been formed. Thus, the expensive action of forming the iterate can be postponed until the residual is deemed small enough. Details of the GMRES(k) algorithm are provided below.

Given A, b, L, U, k, ε_{tol} and $\ell_{\max}$, proceed as follows:

```
    $\varepsilon = \varepsilon_{\text{tol}} \|b\|$
    $x_0 = 0$
  (GMRES(k) cycles)
    for $\ell = 1, 2, \ldots, \ell_{\max}$
    $u_1 = b - Ax_{\ell-1}, \bar{e}_1 = \|u_1\|$
    $u_1 \leftarrow \frac{u_1}{\|u_1\|}$
  (GMRES(k) iteration)
    for $i = 1, 2, \ldots, k$
    $u_{i+1} = Au_i$
  (Modified Gram-Schmidt orthogonalization)
    for $j = 1, 2, \ldots, i$
    $\tilde{\beta}_{i+1J} = (u_{i+1}, u_i)$
    $u_{i+1} \quad u_{i+1} - \tilde{\beta}_{i+1,j} u_j$
  (End of Gram-Schmidt orthogonalization)
    $\bar{h}^{(i)} = \{\tilde{\beta}_{i+1,1}, \tilde{\beta}_{i+1,2}, \ldots, \tilde{\beta}_{i+1,i}, \|u_{i+1}\|\}^T, u_{i+1} = \frac{u_{i+1}}{\| u_{i+1} \|}$
  (Q-R algorithm)
    for $j = 1, 2, \ldots, i-1$
    $\begin{Bmatrix} \bar{h}_j^{(i)} \\ \bar{h}_{j+1}^{(i)} \end{Bmatrix} \leftarrow \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{Bmatrix} \bar{h}_j^{(i)} \\ \bar{h}_{j+1}^{(i)} \end{Bmatrix}$
    (end of $j$ loop)
    $r = \left[ \left(\bar{h}_i^{(i)}\right)^2 + \left(\bar{h}_{i+1}^{(i)}\right)^2 \right]^{\frac{1}{2}}$
    $c_i = \frac{\bar{h}_i^{(i)}}{r}, \; s_i = \frac{\bar{h}_{i+1}^{(i)}}{r}$
    $\bar{h}_i^{(i)} \leftarrow r, \; \bar{h}_{i+1}^{(i)} \leftarrow 0$
    $\bar{e}_{i+1} = -s_i \bar{e}_i, \; \bar{e}_i \leftarrow c_i \bar{e}_i$
  (end of Q-R algorithm)
    Convergence check: if $|\bar{e}_{i+1}| \leq \varepsilon$ exit $i$ loop
  (end of GMRES(k) iteration)
    Solve for $y$:
    $\begin{bmatrix} \bar{h}_1^{(1)} & \cdots & \bar{h}_1^{(i-1)} & & \bar{h}_1^{(i)} \\ 0 & \cdots & \cdots & & \cdot \\ \cdot & \cdots & \cdots & & \cdot \\ 0 & \cdots & \bar{h}_{i-1}^{(i-1)} & & \bar{h}_{i-1}^{(i)} \\ 0 & \cdots & \cdot & 0 & \bar{h}_i^{(i)} \end{bmatrix} \begin{Bmatrix} y_1 \\ \cdot \\ \cdot \\ y_{i-1} \\ y_i \end{Bmatrix} = \begin{Bmatrix} \bar{e}_1 \\ \cdot \\ \cdot \\ \bar{e}_{i-1} \\ \bar{e}_i \end{Bmatrix}$
    Updated solution:
    $x_\ell \leftarrow x_{\ell-1} + \sum_{j=1}^{i} y_j u_j$
    Convergence check: if $|\bar{e}_{i+1}| \leq \varepsilon$, exit $\ell$ loop
  (end GMRES(k) cycle)

Return
```

3. PARALLEL COMPUTING ENVIRONMENT

The ANUPAM cluster at the Parallel Computing Lab in the Department of Mathematics at IIT Kanpur is a homogeneous cluster of eight Intel Pentium III processors. The configuration of each of these eight PEs is as follows:

Intel Pentium III, Infinity 2000 BT @ 800 MHz Processors
Intel i815e Chipset
256 Kb on dir Cache (ATC)
256 MB SDRAM PC 133 MHz
20 GB Hard disk
10/100 Mbps PCI Ethernet card
ANULIB (Parallel Library)

All eight PCs are sitting on Star-Network provided by Dlink Switch (DES-3225G), which is a 24-port fast Ethernet switch. Layout of the parallel system is presented in Figure 1*a*.

ANULIB is a set of message-passing library calls that are used for communication among the processors sitting on the Dlink-Switch network. These library calls were developed for Linux OS by the computer division of BARC, Mumbai, India, in 1992. ANULIB environment supports master–slave paradigms of parallel programming. In master–slave paradigms, one of the processing elements (PEs) acts as the master node and the rest of the PEs act as slaves. Under ANULIB there are two separate sets of programs, one for the master node and another for the slave nodes. A sample of master–slave programs will be provided after introducing the ANULIB library calls. The set of library calls under ANULIB can be divided into four categories: (1) initialization calls, (2) termination calls, (3) communication calls, and (4) miscellaneous calls. These calls are explained below.

1. Initialization calls:

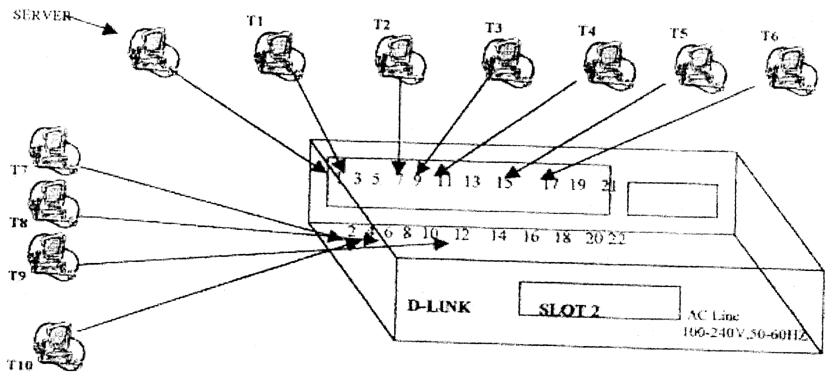
```
m_init (<number of slave nodes>, <slave executable file name>)
    (for master program)
s_init( )      (for slave program)
```

These commands are used to start parallel environment in the program and are called in master and slave programs, respectively, before making any of the communication calls.

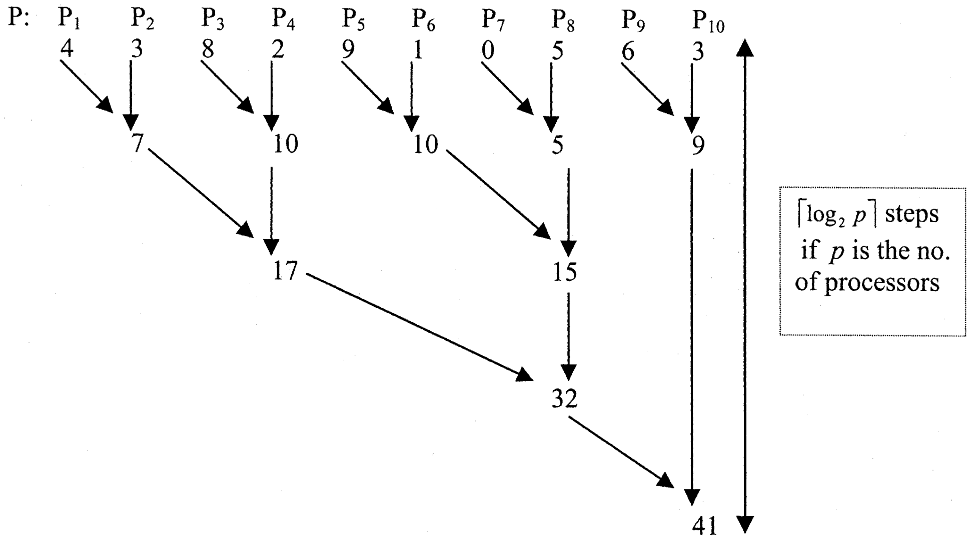
2. Termination calls:

```
m_end( )      (for master program)
s_end( )      (for slave program)
```

These calls are used to close the communication channel and to do some cleanup action.



(a)


P: P1 P2 P3 P4 P5 P6 P7 P8 P9 P10
4 3 8 2 9 1 0 5 6 3
7 10 10 5 9
17 15
32
41
⌈log2 p⌉ steps if p is the no. of processors

(b)

Figure 1. (*a*) Detailed layout of eight-noded PC cluster. (*b*) Pictorial representation of fan-in algorithm.

3. Communication calls:

```
send_element(<variable to be sent>, <destination CPU identity>,
    data type>)
b_send_element(<variable to be sent>, <data type>)
receive_element(<variable to be sent>, <source CPU identity>,
    <data type>)
```

```
send_els(<destination CPU identity>, <number of variables to be sent>,
    <variable1>, <data type>, <variable2>, <data type>, ...)
b_send_els(<number of variables to be sent>, <variable1>,
    data type> , <variable2> , <data type>,...)
receive_els(<source CPU identity>, <number of variables to be
    received>, <variable1>, <data type>, <variable2>,
    <data type>, ...)
send_data(<data buffer>, <buffer size>, <destination CPU identity>,
    <data type>)
b_send_data(<data buffer>, <size of the buffer>, <data type>)
receive_data(<data buffer>, <size of the buffer>, <source CPU
    identity>, <size check>, <data type>).
```

These communications calls are common to both master and slave programs. An include file "mincl.inc", should be included in the master file which contains certain declarations used by these communication calls. The corresponding include file for slaves is "sincl.inc".

4. Miscellaneous calls:

```
get_cpu_id( )
```

This call returns an integer value that represents the ID of the processing element.

```
second( ) ; second1( )
```

These functions return a double-precision value that represents the CPU time used by this process from the start.

Sample Fortran Program

(In this program, the master program sends a real number to the slave processors. Each of the slave processors, after receiving the number, adds its processor identity to this number and sends it back to the master. The master processor then displays it.)

```
C     MASTER PROGRAM
      include 'mincl.inc' ! include file
      real a, b
      print*, "Type a real number"
      read*, a
      print*, "Number of processors:"
      read*, nproc
      call m_init(nproc-1, 's_sample')    ! initialization call
      call b_send_element(a,S_REAL)    ! broad sending element 'a' to all slave
processors
```

```
      do i = 1, nproc-1
            call receive_element(b, i, S_REAL)      ! receiving modified number
from all slave processors
        print*, " Element received from slave processor:", i, "is:",b
      end do
      call m_end( )      ! termination call
      stop
      end

C       SLAVE PROGRAM (slave executable file is written as s_sample)
        include 'sincl.inc' !include file
        real a, b
        call s_init( )      ! initialization call
        call receive_element(a, 0, S_REAL)      ! receiving element a from master
processor whose identity is '0'
      myid = get_cpu_id( )
      print*, "I'm slave processor no.:", myid, "Number received from master
         processor is:", a
      b= a + myid
      call send_element(b, 0, S_REAL)      ! sending modified element 'b'
        to master
      call s_end( )
      end
```

The master and the slave programs are compiled using ANUG77 compiler to generate executable code for master and slave nodes. A machine file called hostname.par containing the details of the PEs and the corresponding executable files is prepared prior to execution of the code on PEs.

4. PARALLEL STRATEGIES AND ALGORITHMS

The computations which are amenable to parallelization in the algorithms discussed earlier are (1) inner product of two vectors, (2) matrix–vector multiplication, (3) L_2 norm of a vector, (4) summation of two vectors, etc. So, to begin with, the parallelization of these basic components will be discussed, and later the actual algorithmic parallelization will be dealt with.

1. Inner Product of Two Vectors (Vector–Vector Multiplication)

Suppose $\bar{u}, \bar{v}$ are two vectors of dimension $n \times 1$. Then, to find the inner product $\langle \bar{u}, \bar{v} \rangle = \sum_{i=1}^{n} u_i v_i$ on a cluster of p nodes, the vectors are partitioned into p segments and are distributed to p processors as follows:

```
do $i = 1, 2, \dots, p$
    if ($i \leq$ ($n$ modulo $p$)) then
        Load_on_PE$_i$ = $\left\lceil \frac{n}{p} \right\rceil$
        else
            Load_on_PE$_i$ = $\lfloor \frac{n}{p} \rfloor$
        endif
    enddo
```

Here PE$_i$ denotes the *i*th processor. Now each of the *p* processors has vectors local_*u*, local_*v* of size (Load_on_PE$_i$). The inner products of these segments residing in various processors are determined concurrently by executing the sequential code in the respective processors as follows:

```
set local_product = 0
do $i$ = 1, Load_on_PE$_i$
    local_product = local_product + local_$u(i) \times$ local_$v(i)$
enddo
```

Once local_product is determined in each of the *p* processors, these are communicated to the master node, wherein they are summed to give the actual inner product.

If *p* (number of processors) is very large, one may use fan-in strategy or a parallel reduction algorithm to find the global sum from the local_sums evaluated on individual processors.

Parallel reduction algorithm (fan-in-strategy). In the fan-in algorithm, data flow from the leaves of the tree to the root. Given a set of n values $a_1, a_2, \ldots, a_n$ and an associative binary operator $(+)$, reduction is the process of computing $a_1(+)a_2(+)a_3(+)\cdots(+)a_n$. For example, consider the set of integers $\{4, 3, 8, 2, 9, 1, 0, 5, 6, 3\}$. Say that each these integers resides in distinct PEs. Then parallel reduction process may be described pictorially as in Figure 1*b*.

2. Matrix–Vector Multiplication

To find the product of $A_{n\times n}$ and vector $u_{n\times n}$, the matrix $A_{n\times n}$ is row-wise block partitioned into segments A_i of size (Load_on_PE$_i$) $\times\, n$ and distributed to various PEs. Also, the whole of the vector u_n is broadcast by the master PE to all slave PEs. Now in each of the slaves matrix vector multiplication of the respective matrix blocks and the vector are carried out.

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ \cdot \\ \cdot \\ \cdot \\ A_n \end{bmatrix}_{A_{n\times n}} \times \begin{pmatrix} u_{n\times 1} \\ \\ \text{In} \\ \text{all} \\ \text{PEs} \\ \\ \end{pmatrix}_{u_{n\times 1}} \rightarrow \begin{pmatrix} Au_1 \\ Au_2 \\ Au_3 \\ \cdot \\ \cdot \\ \cdot \\ Au_n \end{pmatrix}_{Au_{n\times 1}}$$

3. L_2 Norm of a Vector

The L_2 Norm of a vector $\bar{v} = (v_1, v_2, v_3, \ldots, v_n)^T$ is denoted by

$$||\bar{v}||_2 = \left(\sum_{i=1}^{n} v_i^2\right)^{1/2} = (\bar{v}.\bar{v})^{1/2}$$

It is determined in the same manner as the inner product of vectors, i.e., based on block partition of the vectors.

4. Summation of Two Vectors

If $\bar{u}$ and $\bar{v}$ are two vectors, then, to determine $\bar{w} = \bar{u} + \bar{v}$, we do the block partition of the two vectors and distribute the segments to various processors. Now the summation of these segments is carried out concurrently on all the processors.

Parallelization of CGM Solver under Master–Slave Programming Paradigm

Given A, b, $K_{\max}$ (max no. of iterations), ε (tolerance), do the following:

1. Master PE reads the order of matrix n, the matrix $A_{n\times n}$, vector $b_{n\times 1}$, $x_{0\,n\times 1}$, $K_{\max}$, and ε.
2. Master node broadcasts the control data, i.e., order of matrix (n), number of processors (nprocs), and vector x_0, to all the slave nodes (nprocs-1 slave nodes).
3. Based on the block partition strategy, both master and slave nodes calculate load distribution on various nodes of the parallel system
4. Master node segments the matrix $A_{n\times n}$ and vector $b_{n\times 1}$ as per the load distribution identified in step 3 and sends the segmented blocks of A $(A = [A_1\, A_2 \cdots A_i \cdots A_{\text{nprocs}-1} A_0]^T)$ and $b(b = [b_1 b_2 \cdots b_i \cdots b_{\text{nprocs}-1} b_0]^T)$, where each segment A_i and b_i are of size Load_on_PE$_i \times n$ and Load_on_PE$_i \times 1$, respectively.
5. Calculate $\bar{r}_i = \bar{b}_i - A_i \bar{x}_0$ on PE$_i$, where $0 \le i \le \text{nprocs} - 1$, and

$$\bar{r} = \left[\bar{r}_1 \bar{r}_2 \cdots \bar{r}_i \cdots \bar{r}_{\text{nprocs}-1} \bar{r}_0\right]^T.$$

6. Observe $\rho_0 = (\bar{r}.\bar{r})^{1/2} = \left(\sum_{i=0}^{\text{nprocs}-1} (\bar{r}_i.\bar{r}_i)\right)^{1/2} = \left(\sum_{i=0}^{\text{nprocs}-1} \rho_{0_i}\right)^{1/2}$. Now to calculate ρ_0, calculate $\rho_{0_i} = (\bar{r}_i\, \bar{r}_i)^{1/2}$ on PE$_i$ and use the fan-in algorithm to get ρ_0 on master node.
7. Set $k = 0$ on all nodes.
8. Let $\bar{p} = [\bar{p}_1\ \bar{p}_2 \cdots \bar{p}_{\text{nprocs}-1}\ \bar{p}_0]^T$, where $\bar{p}_i$ is the segment block of $\bar{p}$ residing in PE$_i$.

```
If ($k = 0$) then
        set $\bar{p}_i = \bar{r}_i$ i.e. $\bar{p}_i(j) = r_i(j)$ on $i$th PE for $1 \le j \le$ Load_on_PE$_i$
else
```

calculate $\beta_k = r_k^T r_k / r_{k-1}^T r_{k-1}$ on master node and broadcast β_k to all nodes.

Now set $\bar{p}_i = \bar{r}_i + \beta\, \bar{p}_i$ i.e. $\bar{p}_i(j) = \bar{r}_i(j) + \beta\, \bar{p}_i(j)$ on all nodes.

Gather the segments of update vector $\bar{p} = [\bar{p}_1\, \bar{p}_2 \cdots \bar{p}_{\text{nprocs}-1}\, \bar{p}_0]^T$ on master and broadcast $\bar{p}$ to all slaves.

9. Calculate $\bar{w}_i = A_{\text{Load\_on\_PE}_i \times n} \times \bar{p}_{n\times 1}$ on PE$_i$ so that $\bar{w}_{k_{n\times 1}} = [\bar{w}_1 \bar{w}_2 \cdots \bar{w}_{\text{nprocs}-1} \bar{w}_0]^T$ is available in segments on various PEs
10. Calculate $\bar{p}_i^T \bar{w}_i = \sum_{j=1}^{\text{Load\_on\_PE}_i} \bar{p}_i(j) \bar{w}_i(j)$ on PE$_i$, $0 \le i \le \text{nprocs} - 1$ and communicate the same to master node.
11. Calculate $\bar{p}_i^T \bar{w}_i = \sum_{i=0}^{(\text{nprocs}-1)} \bar{p}_i \bar{w}_i$ on master node and set $\alpha_k = \frac{r_k^T r_k}{p_k^T w_k}$. Now broadcast α_k to all slaves.
12. Set $\bar{x}_i = \bar{x}_i + \alpha \bar{p}_i$ on PE$_i$, $0 \le i \le \text{nprocs} - 1$ so that the solution vector $\bar{x}_{k+1} = [\bar{x}_1\, \bar{x}_2 \cdots \bar{x}_{\text{nprocs}-1} \bar{x}_0]^T$ is updated by concurrently updating segments $\bar{x}_{\text{Load\_on\_PE}_i \times 1}$ on PE$_i$.
13. Set $\bar{r}_i = \bar{r}_i - \alpha\, \bar{w}_i$ on PE$_i$, $0 \le i \le \text{nprocs} - 1$ so that the residual vector $\bar{r}_{k+1} = [\bar{r}_1\, \bar{r}_2 \cdots \bar{r}_{\text{nprocs}-1} \bar{r}_0]^T$ is updated by concurrently updating segments $\bar{r}_{i_{\text{Load\_on\_PE}_i \times 1}}$ on PE$_i$.
14. Calculate $\rho_{k+1_i} = \sum_{j=1}^{\text{Load\_on\_PE}_i} (r_i(j) \cdot r_i(j))$ on PE$_i$ and communicate the same to master node.
15. Calculate $\rho_k = \sum_{i=0}^{\text{nprocs}-1} \rho_{k_i}$ on the master node.
16. Set $k = k + 1$ on all PEs
17. Test for convergence of solution, i.e., $(\rho_{k+1})^{1/2} > \varepsilon ||b||_2$ on master node. If the solution has converged, communicate the message to all slave nodes and terminate the slave and master programs after storing the solution. Otherwise communicate the message of program execution to all the nodes and go to step 8 in all nodes. Continue the execution until convergence is reached.

Parallelization of GMRES(k) Solver under Master–Slave Paradigm

Let $A_{n\times n}, b_{n\times 1}, \ell_{\max}, k, \varepsilon_{\text{tol}}$ be given. The various steps involved in the parallelization of GMRES(k) algorithm are as follows:

1. Master node reads the order of matrix (n), k (re-start value of GMRES(k) iteration), $\ell_{\max}$ (max number of GMRES(k) cycles), and the number of processing elements (nprocs) and broadcasts the same to slaves.
2. Master node reads the asymmetric matrix $A_{n\times n}$ and $b_n \times 1$ and partitions A and b such that $A = [A_1\, A_2 \cdots A_i \cdots A_{\text{nprocs}-1}\, A_0]^T, b = [b_1\, b_2 \cdots b_i \cdots b_{\text{nprocs}-1}\, b_0]^T$. Here each segment of A_i and b is of size Load_on_PE$_i \times n$, Load_on_PE$_i \times 1$, respectively.
3. Master node sends the block partitioned $A_{\text{Load\_on\_PE}_i \times n}, b_{\text{Load\_on\_PE}_i \times 1}$ to node PE$_i$.
4. Calculate $||\bar{b}||^2$ concurrently with appropriate master–slave communications.
5. Set $\bar{x} = (0, 0, \ldots, 0)$ on all PEs.
6. For $\ell = 1, 2, \ldots, \ell_{\max}$ (GMRES(k) cycle)

```
6.1. Calculate $\bar{u}_1 = \bar{b} - A\bar{x}_{\ell-1}$ concurrently on all PEs based on the
     respective blocks of data $A_{\mathrm{Load\_on\_PE_i}\times n}, b_{\mathrm{Load\_on\_PE_i}\times 1}$ and the vector $\bar{x}$.
6.2. Evaluate $\bar{e}_{1_i} = (\bar{u}_{1_i}, \bar{u}_{1_i})$ on PE$_i$ for $0 \le i \le \mathrm{nprocs} - 1$ and send the
     same to PE$_0$ so that $\bar{e}_1 = (\bar{e}_{1_1}, \bar{e}_{1_2}, \ldots \bar{e}_{1_{\mathrm{nprocs}-1}}, \bar{e}_{1_0}) =$
     $\left[\sum_{i=0}^{\mathrm{nprocs}-1} (\bar{u}_{1_i}, \bar{u}_{1_i})\right]^{1/2} = \|\bar{u}_1\|$ can be evaluated on PE$_0$, where $\bar{u}_{1_i}$ and
     $\bar{e}_{1_i}$ denote the segments of vector $\bar{u}_1$ and $\bar{e}_1$, respectively, of size Loa-
     d_on_PE$_i$ on processor PE$_i$.
6.3. Broadcast $\|\bar{u}_1\|$ to all PEs and set $\bar{u}_{1_i}(j) = \bar{u}_{1_i}(j)/\|\bar{u}_1\|$ for
     $1 \le j \le$ Load_on_PE$_i$ and $0 \le i \le \mathrm{nprocs} - 1$, i.e., $u_1 = u_1/\|u_1\|$ con-
     currently on all PEs based on respective blocks of data.
6.4. Gather $\bar{u}_1$ on the master node and broadcast the same to all nodes.
6.5. For $i = 1, 2, \ldots$, k (GMRES(k) iteration begins)
     6.5.1. Calculate $\bar{u}_{i+1_{\hat{i}}} = A_{\hat{i}}\bar{u}_{i_{\hat{i}}}, 0 \le hati \le$ nprocs-1
            (modified Gram Schmidt orthogonalization)
            for $j = 1, 2, \ldots, i$
               Calculate $\tilde{\beta}_{i+1,j_{\hat{i}}} = \left(\bar{u}_{i+1_{\hat{i}}}, \bar{u}_{j_{\hat{i}}}\right)$ on PE$\hat{i}$ for $0 \leqslant \hat{i} \leqslant$ nprocs-1
     6.5.2. Gather $\tilde{\beta}_{i+1,j_{\hat{i}}}$ from all PEs on master and calculate $\tilde{\beta}_{i+1,j} =$
            $\sum_{\hat{i}=0}^{\mathrm{nprocs}-1} \tilde{\beta}_{i+1,j_{\hat{i}}}$ and broadcast $\tilde{\beta}_{i+1,j}$ to all PEs.
     6.5.3. Calculate on each PE$\hat{i}$, for $0 \le \hat{i} \le$ nprocs-1, $\bar{u}_{i+1_{\hat{i}}} =$
            $\bar{u}_{i+1_{\hat{i}}} - \tilde{\beta}_{i+1,j}\bar{u}_{j_{\hat{i}}}$, where each of vectors $\bar{u}_{i+1_{\hat{i}}}$ is of size
            (Load_on_PE$\hat{i} \times 1$).
            (end of Gram-Schmidt orthogonalization)
6.6. On each PE$\hat{i}, 0 \le \hat{i} \le$ nprocs-1 calculate $\|u_{i+1_{\hat{i}}}\| = (\bar{u}_{i+1_{\hat{i}}}.\bar{u}_{i+1_{\hat{i}}})$ and
     gather $u_{i+1_{\hat{i}}}$ from all PEs on master node.
6.7. Now calculate $\|u_{i+1}\| = \left(\sum_{\hat{i}=0}^{\mathrm{nprocs}-1} \|u_{i+1_{\hat{i}}}\|^2\right)^{1/2}$ on master node and
     broadcast $\|u_{i+1}\|$ to all nodes.
6.8. In the master node set $\bar{h}^{(i)} = \left\{\tilde{\beta}_{i+1,1}, \ldots, \tilde{\beta}_{i+1,i}, \|u_{i+1}\|\right\}^T$.
6.9. On each PE$\hat{i}, 0 \le \hat{i} \le$ nprocs-1, set $\bar{u}_{i+1_{\hat{i}}} = (\bar{u}_{i+1_{\hat{i}}}/\|u_{i+1}\|)$
6.10. (Q-R algorithm)
     (Carry out sequentially in master node)
     6.10.1. for $j = 1, 2, \ldots, i-1$
             $$\begin{Bmatrix} \bar{h}_j^{(i)} \\ \bar{h}_{j+1}^{(i)} \end{Bmatrix} \leftarrow \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{Bmatrix} \bar{h}_j^{(i)} \\ \bar{h}_{j+1}^{(i)} \end{Bmatrix}$$
             (end j loop)
     6.10.2. $r = \left[\left(\bar{h}_i^{(i)}\right)^2 + \left(\bar{h}_{i+1}^{(i)}\right)^2\right]^{1/2}$
     6.10.3. $c_j = \bar{h}_i^{(i)}/r$
     6.10.4. $s_j = \bar{h}_{i+1}^{(i)}/r$
     6.10.5. $\bar{h}_i^{(i)} \leftarrow r$
     6.10.6. $\bar{h}_{i+1}^{(i)} \leftarrow 0$
     6.10.7. $\bar{e}_{i+1} \leftarrow -s_i\bar{e}_i$
     6.10.8. $\bar{e}_i \leftarrow -c_i\bar{e}_i$
     (end of Q-R algorithm)
```

6.11. Carry out convergence check, i.e., check if $|\bar{e}_{i+1}| < \varepsilon$ in master node. If yes, then convey the message to all PEs and exit the GMRES(k) iteration loop from all PEs. Otherwise continue the GMRES(k) iteration loop execution in all PEs.
(end of GMRES(k) iteration)

6.12. Solve for $\bar{y}$ in master node by back substitution of the following upper triangular system:

$$\begin{bmatrix} \bar{h}_1^{(1)} & \cdots & \bar{h}_1^{(i-1)} & \bar{h}_1^{(i)} \\ \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdots & \cdot & \cdot \\ 0 & \cdots & \bar{h}_{i-1}^{(i-1)} & \bar{h}_{i-1}^{(i)} \\ 0 & \cdots & 0 & \bar{h}_i^{(1)} \end{bmatrix} \begin{Bmatrix} y_1 \\ \cdot \\ \cdot \\ y_{i-1} \\ y_i \end{Bmatrix} = \begin{Bmatrix} \bar{e}_1 \\ \cdot \\ \cdot \\ \bar{e}_{i-1} \\ \bar{e}_i \end{Bmatrix}$$

This step can be executed out concurrently using slave PEs, but it is not scalable. No big gain has been noticed in parallelizing this step, as the size of the system is $k \times k$ and a generally practical choice of is $k \leq 25$.

6.13. Broadcast $\bar{y}$ from master node to all PEs.

6.14. Update $\bar{x}_{\hat{i}}$ on all PE$\hat{i}$ for $0 \leq \hat{i} \leq$ nprocs-1 as follows:

$$\bar{x}_{\hat{i}} \leftarrow \bar{x}_{\hat{i}} + \sum_{j=1}^{i} y_j \cdot \bar{u}_{j_{\hat{i}}} \qquad 1 \leq i \leq \text{Load\_on\_PE}_{\hat{i}}$$

6.15. Check for convergence, i.e., if $|\bar{e}_{i+1}| \leq \varepsilon$ on master node, and broadcast the result to all PEs. If the convergence criterion is satisfied, then exit from GMRES(k)-cycle loop in all PEs and terminate the parallel execution after storing the solution. Otherwise continue the parallel execution of GMRES(k)-cycle loop.

5. TEST PROBLEMS

Three heat transfer problems, namely, (1) the non-linear reaction-diffusion problem [13], (2) free convection in a vertical porous enclosure under Darcian assumptions [14, 15], and (3) free convection in a vertical porous enclosure under non-Darcian assumptions [16, 17], are considered to test the parallel algorithms, implementation strategies, and to study the performance of Krylov subspace solvers in solving linear systems obtained by either finite-difference or finite-element analysis of partial differential equations modeling these heat transfer applications. The details of the mathematical model and the numerical schemes employed in reducing the PDEs to algebraic systems are given below.

Problem 1: Nonlinear Reactive-Diffusive Problem

The governing equation describing the reactive-diffusive process in $\Omega = \{(x,y)\,|\,0 \le x,y \le 1\}$ is

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + A\exp\left(-\frac{E}{RT}\right) = 0 \tag{1}$$

with the boundary conditions

$$\frac{\partial T(x,0)}{\partial y} = 0 \qquad T(x,1) = 500 \qquad \frac{\partial T(0,y)}{\partial x} = 0 \qquad T(1,y) = 500 \tag{2}$$

The reaction parameters are chosen as in [13], i.e., $A = 10$, $E/R = 10{,}000$. The governing equation (1) is simplified and rewritten as

$$T\nabla^2 T + AT = \frac{AE}{R} \tag{3}$$

A finite-difference scheme, based on a five-point stencil at a typical grid point (i,j), $1 \le i,j \le N$, is given by

$$(-4T^n_{i,j} + Ah^2)T^{n+1}_{i,j} + T^n_{i,j}(T^{n+1}_{i+1,j} + T^{n+1}_{i-1,j} + T^{n+1}_{i,j+1} + T^{n+1}_{i,j-1}) = \frac{AE}{R}h^2 \tag{4}$$

At the grid point $(1,j)$, $(i,1)$ corresponding to the boundaries $x = 0, y = 0$, scheme (4) is suitably modified in accordance with the Neumann boundary conditions. Dirichlet boundary conditions prescribed on the boundaries $x = y = 1$ are incorporated by setting temperature values at the grid points (i,N), (N,j) for $1 \le i,j \le N$, to the corresponding Dirichlet prescriptions.

The linear system resulting from (4) is solved by Bi-CGM, Bi-CGSTAB, and GMRES(k) methods on an eight noded PC cluster based on the parallelization strategies. Performance of these Krylov subspace methods is measured in terms of speedup and efficiency factors, and the results are provided in Figures 2*a* and 2*b*. Here one may note that the speedup is defined as the ratio of time taken to run the code on a single machine to the time taken to run the parallel code on *n* machines (i.e., $S_p = T_1/T_n$). Efficiency factor is defined as $E_f = S_p/n$, i.e., (speedup/no. of nodes).

In Figure 2*a*, the speedup factor achieved by Bi-CGM, GMRES(k), and Bi-CGSTAB methods on an eight-noded PC cluster is presented. For these computations, a linear system of size $(2\times10^3)\times(2\times10^3)$ has been considered. Here one may note that for current test data size, speedup factors associated with Bi-CGSTAB are relatively larger when the number of processing elements is less than five. For PEs greater than five, speedup factors associated with GMRES(k) are relatively larger. The tapering nature of speedup factor curves and the decrease in efficiency level with increasing number of processors may be attributed to the smallness in the

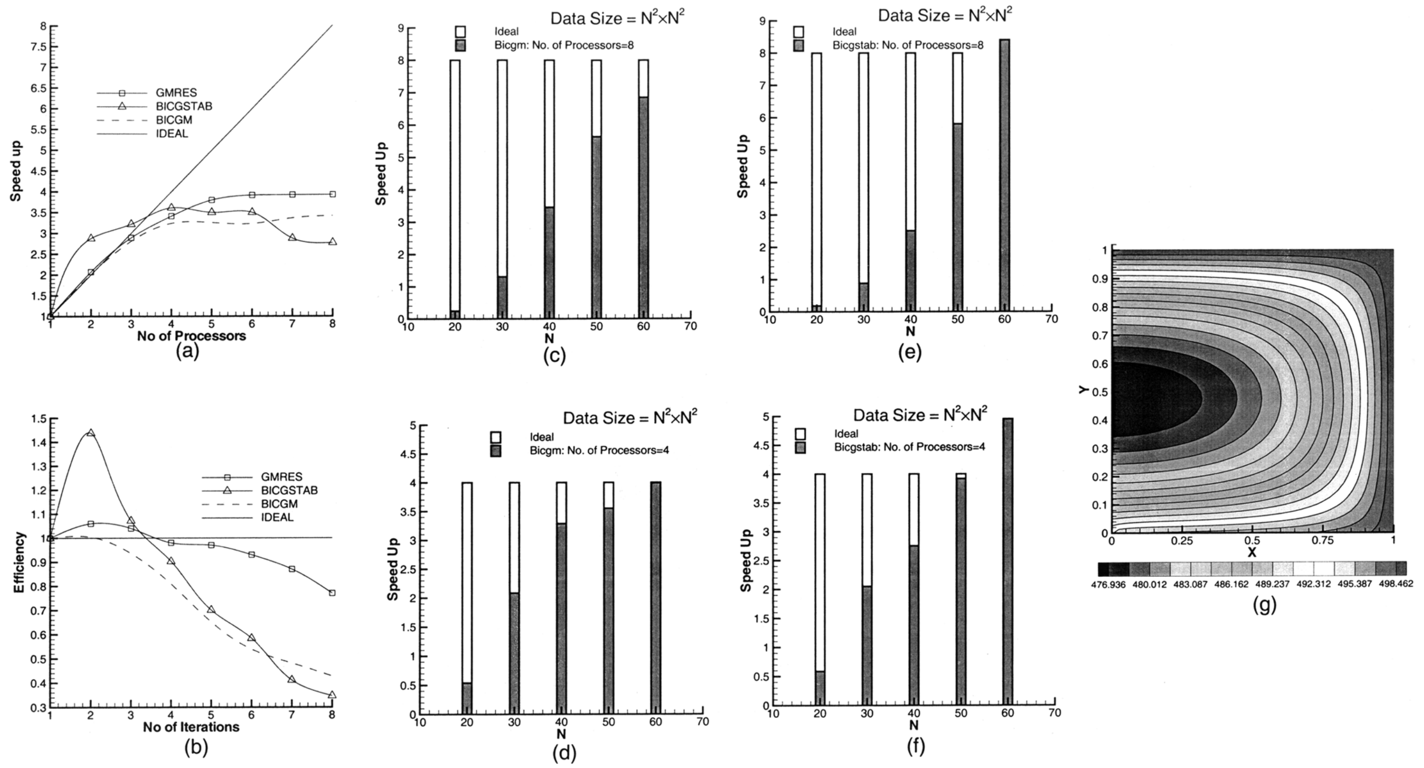


Figure 2. (*a*) Speedup and (*b*) efficiency comparison plots for nonlinear reaction diffusion problem. (*c*, *d*) Speedup plots with increasing data size for Bi-CGM. (*e*, *f*) Bi-CGSTAB. (*g*) Temperature distribution.

data size. In Figure 2*b*, the efficiency factors are presented. Here one may notice that with the current data size on eight nodes, GMRES(k) works at an efficiency of 78%. However, the efficiency of other two solvers falls drastically, due to the smallness in data size. To investigate performance of the solvers with increasing data size, linear systems of size $1{,}600 \times 1{,}600$, $2{,}500 \times 2{,}500$, and $3{,}600 \times 3{,}600$ are considered and parallel computations are carried out on four and eight nodes. Speedup factors resulting from these parallel computations have shown that the performance improves with increasing data size. In Figures 2*c*–2*f*, speedup factors with increasing data size are presented through histogram plots. In Figures 2*c* and 2*d* performance of the Bi-CGM solver with increasing data size on eight and four processors is depicted. In Figures 2*e* and 2*f*, performance of Bi-CGSTAB with increasing data size on eight and four PEs is depicted. In these plots, the lengths of the boxes represent the expected speedup factors whereas the lengths of the shaded portion of the boxes represent the achieved speedup factors. Clearly, with increasing data size, performance of the Bi-CGM and Bi-CGSTAB solvers improves dramatically due to effective cache management. For large data size, Bi-CGSTAB gives super linear speedup. Performance of GMRES(k) also increases with data size. As the problem size grows, the rate of computation to communication will grow and thus the parallel efficiency will increase.

In Figure 2*g*, temperature distribution obtained from parallel simulation on eight nodes is presented. Here the color bar denotes the temperature values. As expected, the temperature distribution is quite symmetric about the left vertical wall of the geometry. Solution is in agreement with the expected isotherm pattern.

Problem 2: Free Convection in a Vertical Porous Enclosure Under Darcian Assumptions

Study of natural convection in a vertical porous enclosure has generated great interest among researchers due to its significance in several scientific and engineering applications, e.g., thermal insulation, geothermal reservoirs, nuclear waste management, grain storage, etc. The nondimensional form of equations governing the conservation of momentum and energy for steady two-dimensional flow in a homogeneous and isotropic porous medium under Darcian assumptions are [13, 14]

$$\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}=\frac{\partial T}{\partial y} \tag{5}$$

$$\frac{\partial\psi}{\partial y}\frac{\partial T}{\partial x}-\frac{\partial\psi}{\partial x}\frac{\partial T}{\partial y}=\frac{1}{Ra}\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right) \tag{6}$$

with boundary conditions

$$\begin{aligned}
&\psi=0, T=1 &&\text{on } y=0 \text{ (on left vertical wall)}\\
&\psi=0, T=0 &&\text{on } y=1 \text{ (on right vertical wall)}\\
&\psi=0, \frac{\partial T}{\partial x}=0 &&\text{on } x=0,1 \text{ (top and bottom walls)}
\end{aligned} \tag{7}$$

The coupled nonlinear partial differential equations (5), (6) together with (7) are solved by the Galerkin finite-element method. The linear system resulting by assembling element matrices is solved by GMRES(k), ILU-GMRES(k), Bi-CGM, and Bi-CGSTAB Krylov subspace methods.

Owing to the nonlinear nature of the problem, both sequential and parallel code take about 28 global iterations. Each global iteration is associated with inner iterations. These inner iteration counts differ from solver to solver. For instance, Bi-CGM has about 40–80 inner iterations during each of its 28-outer/global iterations. GMRES(k) solver is seen to run with about 20–25 inner iterations per GMRES(k) cycle and 3–4 GMRES(k) cycles per outer/global iteration. Each of the global iterations in the current investigation is associated with a linear system of order $1{,}922 \times 1{,}922$. Thus matrices of order (2×10^3) are repeatedly solved 28 times to achieve a solution to an accuracy of 10^{-4} on relative error of field variables. To check the correctness of sequential code, results of the sequential code are compared with those from literature and found to be in good agreement [18, 19]. Next, to validate the parallel code, error data from inner iterations of sequential computation are compared with those from parallel computation and they are found to be in perfect agreement. As a sample, in Figures 3*a*–3*d*, error data from sequential and parallel execution associated with Bi-CGM and GMRES(k) solvers are compared.

In Figures 3*a* and 3*b*, error data associated with the second outer iteration of BICGM solver for $\mathrm{Ra} = 100$ is presented. Figure 3*a* shows data corresponding to the first 50 iterations and Figure 3*b* shows data corresponding to the next 32 iterations. At all inner iteration levels, sequential and parallel error data match perfectly. In Figures 3*c* and 3*d*, sequential and parallel execution error data associated with the 1st, 7th, and 28th outer/global iterations of GMRES(k) solver are compared. Again we find that the error data from both the executions are in perfect agreement. Further, one may also note that each of these outer/global iterations, under both sequential and parallel executions, has about five GMRES(k) cycles. These calculations corresponding to GMRES(k) solver are carried out for $\mathrm{Ra} = 75$. Next, the error data corresponding to sequential and parallel execution of ILU-GMRES(k) solver are compared in Figure 3*d*. Here the data corresponding to the 1st, 7th, and 28th outer iterations are compared. Again a perfect match in the data from sequential and parallel execution can be noted.

Finally, the temperature and flow field corresponding to results from parallel execution on eight nodes are presented in Figures 4*a*–4*d*. In Figures 4*a* and 4*b*, isotherms corresponding $\mathrm{Ra} = 75$, 250 are presented, and in Figures 4*c* and 4*d*, streamlines corresponding to $\mathrm{Ra} = 75$, 250 are presented. Isotherm plots 4*a* and 4*b* clearly show the manifestation of a thermal boundary layer along the left vertical wall as Ra is increased from 75 to 250. Streamline plots in Figures 4*c* and 4*d* depict a change in flow structure with an increase in Ra. These results are very much in accordance with the results reported earlier in the literature [13, 14].

In Figure 5*a*, speedup factors corresponding to Bi-CGM, Bi-CGSTAB, GMRES(k), and ILU-GMRES(k) are presented. Here one may note that Bi-CGM solver is superior to other solvers. ILU-GMRES(k) is also seen to perform well. However, owing to the smallness in data size, speedup factor curves slowly taper as the number of processing elements increases. In the case of ILU-GMRES(k), such tapering has not been noted. This is because with the ILU-GMRES solver,

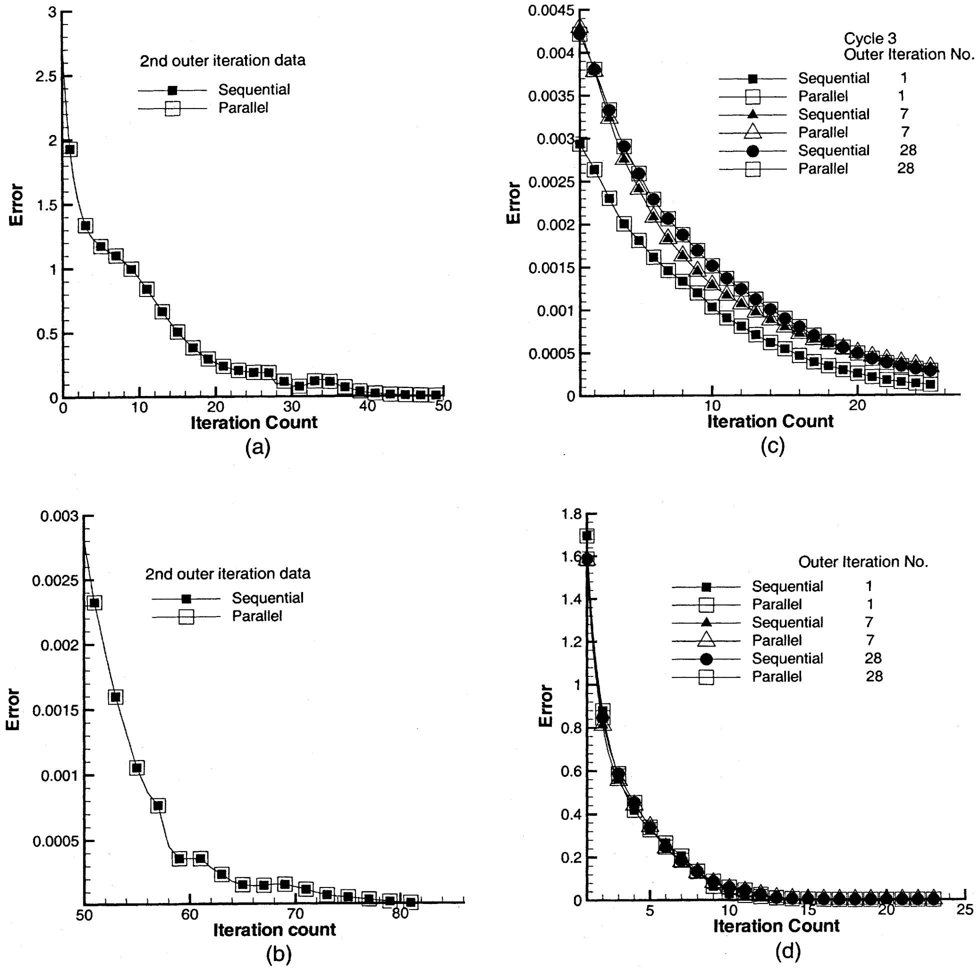


Figure 3. Error data for sequential and parallel executions for (*a*) Bi-CGM fixing Ra = 100 up to 50 inner iterations; (*b*) continuation of plot in (*a*) after 50 inner iterations choosing small scale for error axis; (*c*) GMRES for Ra = 75; (*d*) ILU-GMRES for Ra = 75.

ILU-factorization cost is rather large. In Figure 5*b*, efficiency factors associated with the parallel execution of the four solvers are presented. Clearly, Bi-CGM solver is seen to perform with super linear speedup with over 100–150% efficiency. ILU-GMRES(k) is seen to perform with nearly a linear speedup factor and at a level of 100% efficiency. The other two solvers are about 70% and 40% efficient.

Problem 3: Free Convection in a Vertical Porous Enclosure Under Non-Darcian Assumptions:

Equations governing the flow and convection in a non-Darcian porous enclosure are [16]

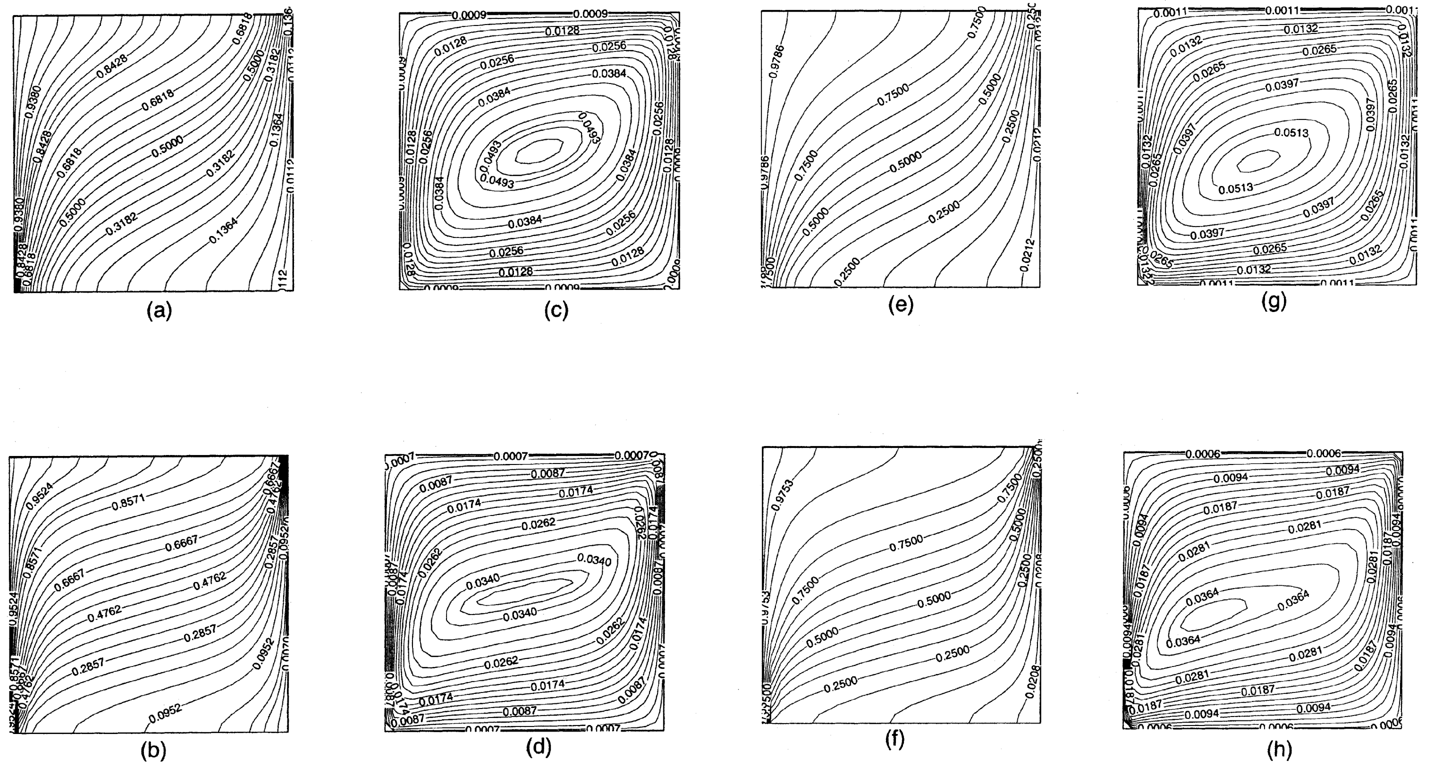


Figure 4. Temperature contours in Darcian case when (*a*) Ra = 75, (*b*) Ra = 250 with (*c*, *d*) corresponding streamlines. Temperature contours in non-Darcian case (Gr* = 0.75) when (*e*) Ra = 75, (*f*) Ra = 250, with (*g*, *h*) corresponding streamlines.

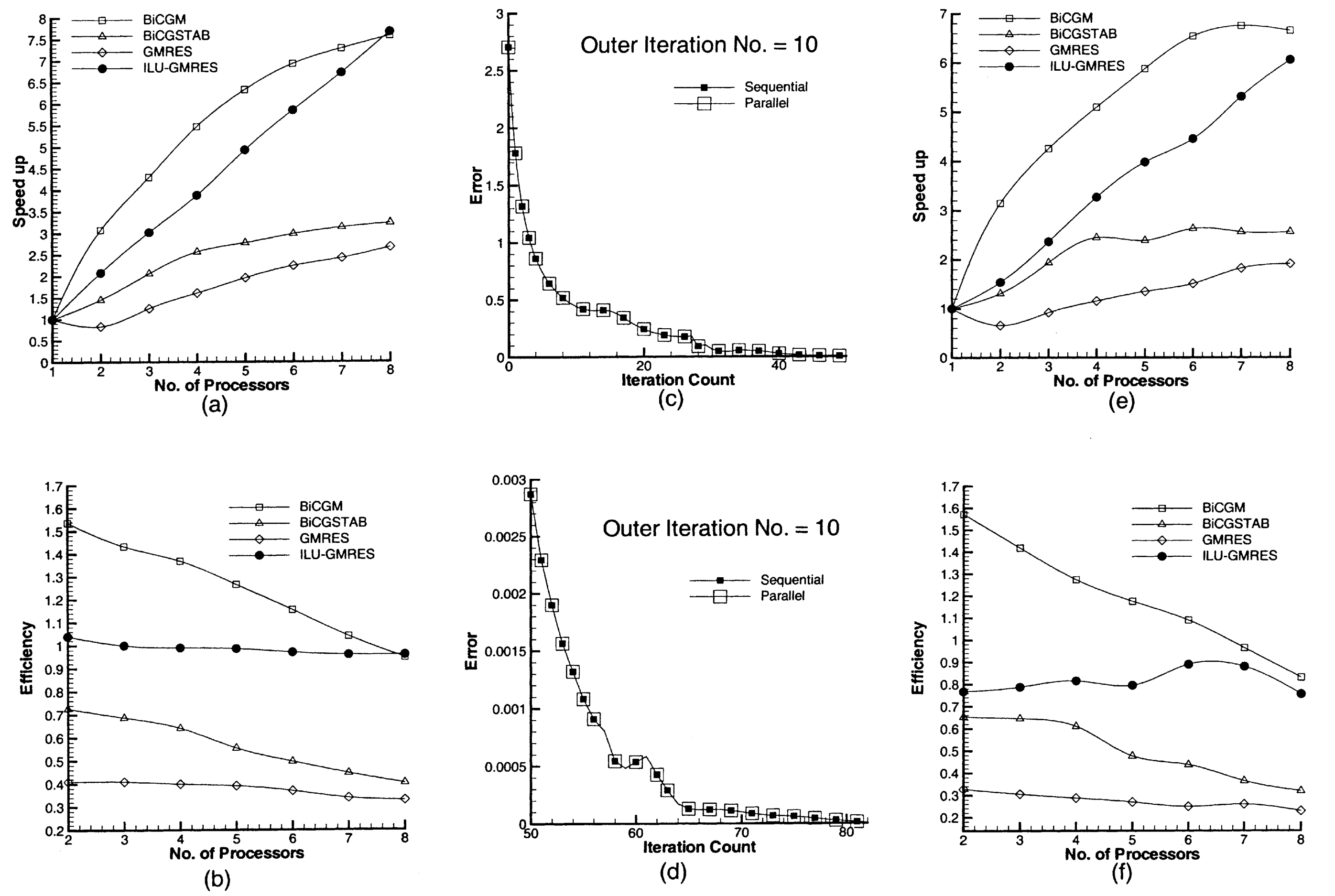


Figure 5. (*a*) Speedup and (*b*) efficiency factor comparison plots when Ra = 100, Gr* = 0.0, error data for Bi-CGM fixing Gr* = 0.25, Ra = 50 for 10th outer iteration, (*c*) up to 50 inner iterations, and (*d*) after 50 iterations. (*e*) Speedup and (*f*) efficiency comparison plots when Gr* = 0.5, Ra = 100.

$$\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\mathrm{Gr}^*\left[-\frac{\partial}{\partial x}\left(\frac{\partial\psi}{\partial x}\right)^2+\frac{\partial}{\partial y}\left(\frac{\partial\psi}{\partial y}\right)^2\right]=\frac{\partial T}{\partial y} \tag{8}$$

$$\frac{\partial\psi}{\partial y}\frac{\partial T}{\partial x}-\frac{\partial\psi}{\partial x}\frac{\partial T}{\partial y}=\frac{1}{\mathrm{Ra}}\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial x^2}\right) \tag{9}$$

with boundary conditions

$$\left.\begin{array}{ll}\psi=0, T=1 & \text{on } y=0\\ \psi=0, T=0 & \text{on } y=1\\ \psi=0, \dfrac{\partial T}{\partial x}=0 & \text{on } x=0,1\end{array}\right\} \tag{10}$$

Equations (8)–(10) are solved by the Galerkin finite-element method. Here again the global matrix of size $1{,}922\times 1{,}922$ resulting from assembly of element matrices is solved by GMRES(k), Bi-CGM, Bi-CGSTAB, and ILU-GMRES(k).

To begin with, the error data from sequential and parallel executions are compared. As a sample, in Figures 5*c* and 5*d*, sequential and parallel execution error data from the 10th global/outer iteration of the Bi-CGM solver are compared. One may note that they are in perfect agreement. Such a comparison is made for the error data from other solvers too, and they are found to be in good agreement. Next the results in the form of isotherm and streamlines for $\mathrm{Ra}=75{,}250$, $\mathrm{Gr}^*=0.75$ as obtained from parallel computation on eight nodes are presented in Figures 4*e*–4*h*. Figures 4*e* and 4*f* clearly depict the manifestation of a thermal boundary layer along the left vertical wall. Figures 4*g* and 4*h* depict a variation in flow structure with increasing Ra. Further, one may also compare Figures 4*a* and 4*b* with Figures 4*e* and 4*f*, and similarly Figures 4*c* and 4*d* with Figures 4*g* and 4*h* to check the influence of Gr*. Clearly, Gr* is seen to intensify the flow circulation and also sharpen the thermal boundary layer. These results are very much in tune with those reported in literature [15, 16].

In Figures 5*e* and 5*f*, speedup and efficiency factors associated with the parallel executions corresponding to $\mathrm{Ra}=100$, $\mathrm{Gr}^*=0.5$ are presented. Here again one can notice that the Bi-CGM is seen to perform better than the other solvers, with super linear speedup factors. ILU-GMRES(k) is seen to have a nearly linear speedup factor and is seen to scale very well with the increase in number of processing elements. It may also be noted that speedup factors associated with Bi-CGM are seen to saturate with the increase in number of nodes, and this may be attributed to the smallness in size of the data. However, ILU-GMRES(k) is seen to scale up linearly. GMRES(k) in the absence of preconditioners is found to be not so efficient. In the presence of ILU preconditioner, it is seen to be 90% efficient. The performance of Bi-CGSTAB is better than that of GMRES(k) but is clearly inferior to that of ILU-GMRES(k) and Bi-CGM. The deterioration in the efficiency of BICGM may be attributed to the smallness in the data size. Again, an increase in data size enhances the speedup factors as observed in the case of the reaction-diffusion problem.

6. CONCLUSIONS

Distributed or parallel algorithms together with parallel implementation strategies for a cluster of eight PCs under ANULIB message-passing libraries has been developed for Krylov methods for linear systems such as GMRES(k), Bi-CGM, Bi-CGSTAB, and ILU-GMRES(k). Parallel code has been developed for all these iterative solvers and has been successfully employed in solving linear systems resulting from finite-difference/finite-element analysis of heat transfer-related mathematical models. For the linear systems resulting from FEA of coupled nonlinear PDE models governing convection in porous enclosure, Bi-CGM and ILU-GMRES(k) solvers are seen to give good speedup factors with an efficiency over 90%. For the linear system resulting from finite-difference analysis of the reaction-diffusion problem, with large data size, Bi-CGSTAB is seen to perform well. The relative merit of linear solvers can perhaps be more explained by looking at the eigenvalue distributions of the different linear systems. In general, with increase in data size, speedup and efficiency factors scale very well with the increase in the number of processing elements.

REFERENCES

1. Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1995.
2. C. Pommerell, Solution to Large Unsymmetric Systems of Linear Equations, Hartung-Goore Verlag, Konstanz, 1992.
3. A. Brandt, Multi-level Adaptive Solutions to Boundary Value Problems, *Math. Comput.*, vol. 31, pp. 333–390, 1977.
4. Y. Saad, Krylov Subspace Methods on Super Computers, *SIAM J. Sci. Stat. Comput.*, vol. 10, no. 6, pp. 1200–1232, 1989.
5. F. Shakib, T. J. R. Hughes, and Z. Johan, A Multi-element Group Preconditioned GMRES(K) Algorithm for Nonsymmetric Systems Arising in Finite Element Analysis, *Comput. Meth. Appl. Mech. Eng.*, vol. 75, pp. 415–456, 1989.
6. T. J. R. Hughes, I. Levit, and J. Winget, Element-by-Element Solution Algorithm for Problems of Structural and Solid Mechanics, *Comput. Meth. Appl. Mech. Eng.*, vol. 36, pp. 241–254, 1983.
7. Y. Eguchi and G. Yagawa, A Conjugate-Residual FEM for Incompressible Viscous Flow Analysis, *Comput. Mech.*, vol. 3, pp. 59–72, 1988.
8. P. Carriere and D. Jeandel, A 3D Finite Element Method for the Simulation of Thermo Convective Flows and Its Performance on a Vector Parallel Computer, *Int. J. Numer. Meth. Fluids*, vol. 12, pp. 929–946, 1991.
9. A. Farcy and T. A. de Roquefort, Chebyshev Pseudospectral Solution of the Incompressible Navier-Stokes Equations in Curvilinear Domains, *Comput. Fluids*, vol. 16, no. 4, pp. 459–473, 1988.
10. G. B. Deng, J. Piquet, P. Queutey, and M. Visonneau, Three-Dimensional Full Navier-Stokes Solvers for Incompressible Flows past Arbitrary Geometries, *Int. J. Numer. Meth. Eng.*, vol. 31, pp. 1427–1451, 1991.
11. K. Chen, Conjugate Gradient Methods for the Solution of Boundary Integral Equations on a Piecewise Smooth Boundary, *J. Comput Phys.*, vol. 97, pp. 127–143, 1991.
12. H. W. Lin, Numerical Simulation of the Dynamics and Instability of Flame Flicker when Subject to Perturbed Boundary Conditions, Ph.D. dissertation, Department of Mechanical Engineering, University of Iowa, Iowa City, IA, 1993.

13. Hsiao-Wen Lin and Lea-Der Chen, Application of the Krylov Subspace Method to Numerical Heat Transfer, *Numer. Heat Transfer A*, vol. 30, pp. 249–270, 1996.
14. B. V. Rathish Kumar and Shalini, Natural Convection in a Thermally Stratified Wavy Porous Enclosure, *Numer. Heat Transfer A*, vol. 43, pp. 753–776, 2003.
15. D. Angirasa, G. P. Peterson, and I. Pop, Combined Heat and Mass Transfer by Natural Convection in a Saturated Thermally Stratified Porous Medium, *Numer. Heat Transfer A*, vol. 31, pp. 255–272, 1997.
16. F. C. Lai and F. A. Kulacki, Non-Darcy Convection from Horizontal Impermeable Surfaces in Saturated Porous Medium, *Int. J. Heat Mass Transfer*, vol. 30, no. 1, 2189–2192, 1987.
17. B. V. Rathish Kumar and Shalini, Natural Convection in a Thermally Stratified Non-Darcian Vertical Porous Enclosure, *J. Porous Media* (In press).
18. K. L. Walker and G. M. Homsy, Convection in a Porous Cavity, *J. Fluid Mech.*, vol. 87, pp. 449–474, 1978.
19. O. V. Trevisan and A. Bejan, Natural Convection with Combined Heat and Mass Transfer Buoyance Effects in Porous Medium, *Int. J. Heat Mass Transfer*, vol. 28, pp. 1596–1611, 1985.