Parallel Computations of Incompressible Viscous Flow in a Lid-driven Square Cavity and Program Performance on the NWT Computer System*

Shigeki HATAYAMA*[1]

ABSTRACT

The NWT computer system available at the NAL since February 1993 comprises two system administrators, *n* processing elements (where *n* was 140 at the beginning, and is 166 at present) and a crossbar network, and operates as a distributed-memory message-passing MIMD computer. Each processing element itself is a vector computer. This paper reports parallel computations of incompressible viscous flow in a lid-driven square cavity on the NWT computer system. In order to obtain numerical solutions of this flow, consistent finite-difference approximations on non-staggered grids and four iterative solution methods are used. Computations are performed on the Reynolds number range of $Re = 0 \sim 10^5$, and effects of the Reynolds number, number of processing elements (*pe*) in the parallel processing, solution method and grid size on the computational results are examined. Actual rates of the parallelised square cavity programs on the NWT computer system are measured, and two characteristic parameters of these programs are estimated for the cases that the actual rate is considered as a function of *pe* and that the actual rate is considered as a function of the grid size. Measurements of the maximum actual rate and estimations of the speedup and efficiency against *pe* on the NWT computer system are indicated as well.

Keywords: square cavity problem, non-staggered grid, consistent finite-difference approximation, iterative method/Jacobi/red-black/CG/ADI, SIMD/MIMD computing, synchronization/data-transfer overhead, program performance, maximum performance, half-performance grain size, characteristic parameters of parallelised program, speedup/efficiency, NWT computer system

概　要

1993年2月に運用を開始した航空宇宙技術研究所の数値風洞(NWT) 計算機システムは世界に類例を見ない新機種の並列計算機システムであって，2台の制御計算機，n台の要素計算機（nの値は運用開始時において140台，現在は166台，並びにクロスバーネットワークから構成されていて，分散メモリ型メッセージパッシング方式のMIMD計算機として動作する。各要素計算機自体はベクトル計算機である。

従って，本研究の主たる動機はこの新しい計算機上の並列処理によって，(1)実際に流れ場の数値計算がどの程度の精度で可能であるか，(2)そのプログラム処理速度は使用台数を増していくとどの程度まで向上させることができるか，(3)その処理速度に限界をもたらす各種のシステム・オーバヘッドはどの程度のものか，そして最後に（4)他の計算機と容易に比較可能なように，一般的に用いられているプログラム処理性能を表すNWTの特性パラメータを推定することにある。

* 平成10年4月8日受付 (received 8 April 1998)

*[1] Computational Sciences Division

本稿では，上記(1)の目的のために，長い研究歴史のあるリッド駆動型平方キャビティ内の非圧縮粘性流れの問題を題材として，それを非スタガードグリッド上でのコンシステントな有限差分スキームで数値解を求める。その際使用する解法として，ヤコビー法／レッド・ブラック法／CG法／ADI法の４つの反復解法を取り上げる。その結果を示すために，Re = 0～10^5の範囲のReの値の変化に応じた平方キャビティ内の流れの変化の様相を詳細に調べる。またその並列処理精度の程度を，使用要素計算機台数を変化させて詳細に調べるとともに，使用反復解法及び使用グリッドサイズに応じても詳細に調べる。

また上記(2)の目的のために，各種プログラム処理速度を使用要素計算機の台数を増大させることによって詳細に調べる。なおこの際，一般的なFORTRANで記述された各種プログラムはそれぞれNWT FORTRANを用いて，可能な限りの最高速処理を実現させ得るように手直しがなされる。ここでNWT FORTRANとはFORTRANの仕様にコンパイラ・ディレクティブ群を付け加えたものである。こうすることによって，物理的には分散メモリ型並列計算機が，ユーザーにとっては論理的に共有メモリ型並列計算機に見え，FORTRANプログラムを大幅に手直しすることなしに，高速処理を可能とする並列プログラムに変更させることが出来る。ただし一般的に言えることであるが，データ転送を実行するコード箇所については十分な注意を持って手直ししなければならない。不注意をおかすと，まったく処理速度を向上させることができない。

また上記(3)の目的のために，システムオーバヘッドの主要因である，FORTRANプログラムに挿入されるコンパイラ・ディレクティブの１つ１つについて，それぞれのオーバヘッドを詳細に調べる。これらの得られた結果はたいへん貴重なものと言え，実際これらのオーバヘッドの値を見ることによってNWTシステム・ソフトウェアの性能改善が精力的になされていった。

そして上記(4)の目的のために，並列プログラムの実効処理性能が使用要素計算機台数の関数として見た場合の特性パラメータと，並列プログラムの実効処理性能がグリッド・サイズの関数として見た場合の特性パラメータとを推定する。これらのパラメータを一般的に使用されている記号で表わすならば，それぞれ(τ_∞, $pe_{1/2}$, $\tilde{pe}$, n)と($\tilde{\tau}_\infty$, $s_{1/2}$) と表現される。

最後に，本稿で得られたほとんどの結果には1993年4月から6月にかけて利用できたNWTのシステム・ソフトウェアと，その１年後に当たる1994年の4月に利用できたNWTのシステム・ソフトウエアに基づく２つの測定値が示されている。結果はこの１年間におけるシステム・ソフトウェアの改善努力が目覚しく，数値的に述べると平均約3倍の処理速度の向上が得られていることを付言する。

1. INTRODUCTION

The incompressible viscous flow in a lid-driven square cavity has been investigated from more than thirty years ago as a representative problem of closed separated flows. Experimental studies of this flow are found chronologically in [15, 16, 23, 19, 6], and analytical studies by means of the finite Fourier series expansions found in [15, 23, 5]. Studies on numerical solutions of this flow are found in [19, 8, 21, 4, 20, 3, 2, 7] which deal with only the vorticity (ζ) and stream-function (ψ), and found in [14, 16, 5, 6, 22, 17, 1] which deal with the pressure (p) as well as ζ and ψ or the primitive variables. In general, as the Reynolds number is larger, the time required to obtain numerical solutions for ζ and ψ is much longer, but the time required to obtain numerical solutions for p is shorter. However there becomes more difficulty in treating the pressure Poisson equation with Neumann boundary conditions. By the way the maximum Reynolds number dealed with in previous studies mentioned above is $Re = 10^4$, and the maximum grid size is 257*257 [3, 7], where we note that [8] and [17] had dealed with $Re = 10^5$ and $Re = 3*10^4$, respectively, but that both numerical solutions could not capture the secondary and tertiary vortices located at the corners of the square cavity due to a rather small grid size.

We have extensively examined the difference between numerical solutions based on marching the unsteady equations in time and on treating the steady-state equations, and the difference between numerical solutions based on staggered grids and on non-staggered grids during this investigation. In this report we indicate numerical solutions based on consistent finite-difference approximations on non-staggered grids for steady-state Navier-Stokes equations [1], which look like to be most superior. We deal with $Re = 10^5$ as the maximum Reynolds number, $16k*16k$ ($1k = 1024$) as the maximum grid size, and the Jacobi/red-black/CG/ADI methods as the iterative solution methods [18].

All the computations are performed in parallel on the Numerical Wind Tunnel (NWT) computer system available at the National Aerospace Laboratory since February 1993 [9, 10, 11, 12]. This system consists of two system administrators, n processing elements (where *n* was 140 at the beginning, and is 166 at present) and a crossbar network, and operates as a distributed-memory message-passing MIMD computer. Each processing element itself is a vector computer. The language to bescribe parallel processing is the NWT Fortran. Main memories of the NWT are physically distributed across the processing elements, but to ease programming, the logical model of the NWT assumed a hierarchical memory parallel computer system for programming offers the virtual global space (or global memory) shared by the selected processing elememts to users. On the other hand, each local space (or local memory) is the memory specific to each processing element. Details about the machine architecture and logical model of the NWT, the NWT Fortran, communication and synchronization are shown in [10].

The prime motive of this study is to make clear the following problems which originate from parallel processing on the NWT computer system:

(1) Is the numerical computation of flows possible to how degree of accuracy practically?

(2) Can the processing speed of parallelised programs improve to how degree along with increase of *pe*?

(3) Are all kinds of system overhead bringing the processing speed bounds how degree?

(4) How degree of value are the characteristic parameters of the NWT computer system?

In this paper, for making clear the above problem (1), we adopt a lid-driven square cavity problem that there is the long study history mentioned above, and obtain numerical solutions of this flow by the consistent finite-differrence schemes on non-staggered grids. As the solution method used at this juncture, we adopt four iterative solution methods of Jacobi/red-black/CG/ADI. From the obtained results, we show the process of stabilization of the eddy system and the location of the center of the primary vortex in the square cavity on the Reynolds number range of $Re = 0 \sim 10^5$, and examine in detail effects of the Reynolds number, the selected number of the processing elements (*pe*) for MIMD computings, the solution method and the grid size on numerical solutions of ζ, ψ and p.

Next for making clear the above problem (2), we examine in detail processing speeds of all kinds of parallelised programs by increase of *pe*, and make its bounds clear. Next for making clear the above problem (3), we examine in detail overhead of each compiler directive of the NWT Fortran which is inserted into the general Fortran program. Finally for making clear the above problem (4), we measure actual rates of the parallelised square cavity program adopted each solution method, and estimate in detail two characteristic parameters, $(\tau_\infty, \tilde{pe}_{1/2}, pe, n)$ and $(\tilde{\tau}_\infty, s_{1/2})$ of these programs on the NWT computer system with respect to the cases that the actual rate is considered as a function of *pe* for a fixed grid size [13] and that the actual rate is considered as a function of the grid size for a fixed *pe*, respectively.

For almost all results for the actual rate on the NWT computer system, we show both values when applied to the system software had been available at the NAL during the period April to June 1993 and during the period April to May 1994. Hence the degree of improvement on the NWT system software becomes self-evidently, which is the result had been vigorously continuing subsequent improvement during one year.

2. FORMULATION

We consider an incompressible viscous flow in a square cavity by a uniformly moving upper surface as shown in Figure 1, where *H* is the depth of cavity, *L* the width of cavity and $H = L = 1$. The dimensionless stream-function vorticity conservation form of the two-dimensional incompressible Navier-Stokes equations is as follows:

$$\zeta_t = -\psi_y\zeta_x + \psi_x\zeta_y + \frac{1}{Re}(\zeta_{xx} + \zeta_{yy}), \qquad (1)$$

$$\psi_{xx} + \psi_{yy} = -\zeta, \qquad (2)$$

where ζ is the vorticity, ψ the stream-function, *Re* the Reynolds number, *t* the time, and *x* and *y* the axtial and normal coordinates, respectively. The subscripts *t*, *x* and *y* refer to partial derivatives with respect to *t*,

x and y, respectively.

The boundary conditions on the stream-function equation (2) for flow in a lid-driven cavity with the upper surface translating to the right with uniform velocity $u = 1$ and with no flow at the other boundaries are at the upper surface

$$\psi_y = 1,\ \psi_x = 0,\ \psi = 0, \tag{3}$$

and at the bottom, left and right surfaces

$$\psi_y = 0,\ \psi_x = 0,\ \psi = 0. \tag{4}$$

The boundary conditions on the vorticity equation (1) are obtained by applying the boundary conditions for the stream-function at the solid boundaries as follows:

$$\zeta = -(\psi_{xx} + \psi_{yy}),\ at\ x = 0,\ 1\ and\ y = 0,\ 1. \tag{5}$$

The steady-state Navier-Stokes equations are deduced from (1) and (2) if we set $\zeta_t = 0$. When we could obtain the steady-state solutions ζ and ψ, the primitive variables can be computed from the following equations:

$$u = \psi_y,\ \upsilon = -\psi_x, \tag{6}$$

$$p_{xx} + p_{yy} = (\upsilon\zeta)_x - (u\zeta)_y = \sigma, \tag{7}$$

where p, u and υ are the total pressure, the velocity component in the x-direction and the velocity component in the y-direction, respectively.

The boundary conditions on the velocity equations (6) are at the upper surface

$$u = 1,\ \upsilon = 0, \tag{8}$$

and at the bottom, left and right surfaces

$$u = 0,\ \upsilon = 0. \tag{9}$$

The following Neumann boundary conditions on the pressure equation (7) are obtained by applying the momentum equations at the solid boundaries [1]:

$$p_x = \upsilon\zeta - \frac{1}{Re}\,\zeta_y,\ at\ x = 0,\ 1, \tag{10}$$

$$p_y = -u\zeta + \frac{1}{Re}\,\zeta_x,\ at\ y = 0,\ 1. \tag{11}$$

Solutions to (7) with (10) and (11) are unique within an arbitrary constant, which can be determined by using the relation

$$\int_{y=0}^{1}\int_{x=0}^{1} p\,dx\,dy = constant. \tag{12}$$

The existence of a solution for (7) with (10) and (11) requires the satisfaction of the following compatibility condition:

$$\int_{y=0}^{1}\int_{x=0}^{1} \sigma\,dx\,dy = \int p_n\,dS, \tag{13}$$

where n is the outward normal to the boundary contour S, enclosing the solution domain.

3. FINITE-DIFFERENCE APPROXIMATIONS

In this Section we discuss the numerical method that we used to obtain the steady-state solutions to (1)-(11), on non-staggered and uniform grids. Let $h = 1/N$ be the mesh size in both the x- and y-directions so that the plane (x, y) is discretized as $x = ih$ and $y = jh$ where $i, j = 0, 1, ..., N$. We denote $\zeta_{i,j} = \zeta\,(ih, jh)$, and so on. All partial derivatives are approximated using second order accurate formulas.

3.1 Finite-difference approximation for steady-state equation (1)

$$(\psi_{i+1,j} - \psi_{i-1,j})(\zeta_{i,j+1} - \zeta_{i,j-1}) - (\psi_{i,j+1} - \psi_{i,j-1})(\zeta_{i+1,j} - \zeta_{i-1,j})$$
$$= -\frac{4}{Re}(\zeta_{i+1,j} + \zeta_{i-1,j} + \zeta_{i,j+1} + \zeta_{i,j-1} - 4\zeta_{i,j}), \tag{14}$$

where $i, j = 1, 2, ..., N-1$.

3.2 Finite-difference approximation for equation (2)

$$\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} - 4\psi_{i,j} = -h^2\zeta_{i,j}, \tag{15}$$

where $i, j = 1, 2, ..., N-1$.

3.3 Finite-difference approximations for equations (3) and (4)

$$\psi_{0,j} = \psi_{N,j} = \psi_{i,0} = \psi_{i,N} = 0, \tag{16}$$

where $i, j = 0, 1, \ldots, N$.

3.4 Finite-difference approximations for equation (5)

The finite-difference approximations for (5) are obtained from (15) and (16) by enforcing reflection at the boundaries as follows: at the upper and bottom surfaces

$$\zeta_{i,N} = -\frac{2}{h^2}(\psi_{i,N-1} + 1),\ \zeta_{i,0} = -\frac{2}{h^2}\psi_{i,1}, \qquad (17)$$

where $i = 1, 2, \ldots, N-1$, and at the left and right surfaces

$$\zeta_{0,j} = -\frac{2}{h^2}\psi_{1,j},\ \zeta_{N,j} = -\frac{2}{h^2}\psi_{N-1,j}, \qquad (18)$$

where $j = 1, 2, \ldots, N-1$.

3.5 Finite-difference approximations for equation (6)

$$u_{i,j} = \frac{1}{2h}(\psi_{i,j+1} - \psi_{i,j-1}),\ \upsilon_{i,j} = -\frac{1}{2h}(\psi_{i+1,j} - \psi_{i-1,j}), \qquad (19)$$

where $i, j = 1, 2, \ldots, N-1$.

3.6 Finite-difference approximation for equation (7)

In order that the compatibility condition (13) is exactly satisfied on a non-staggered grid, we must use the following consistent finite-difference approximation [1]:

$$\begin{aligned} &p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j} \\ &= \frac{h}{4}\{(\upsilon_{i+1,j} + \upsilon_{i,j})(\zeta_{i+1,j} + \zeta_{i,j}) - (\upsilon_{i,j} + \upsilon_{i-1,j}) \\ &(\zeta_{i,j} + \zeta_{i-1,j}) - (u_{i,j+1} + u_{i,j})(\zeta_{i,j+1} + \zeta_{i,j}) + \\ &(u_{i,j} + u_{i,j-1})(\zeta_{i,j} + \zeta_{i,j-1})\}, \end{aligned} \qquad (20)$$

where $i, j = 1, 2, \ldots, N-1$.

3.7 Finite-difference approximations for equations (8) and (9)

$$u_{i,N} = 1,\ u_{0,j} = u_{N,j} = u_{i,0} = \upsilon_{0,j} = \upsilon_{N,j} = \upsilon_{i,0} = \upsilon_{i,N} = 0, \qquad (21)$$

where $i, j = 1, 2, \ldots, N-1$.

3.8 Finite-difference approximations for equations (10) and (11)

For consistency, the pressure gradients p_x and p_y computed from (10) and (11) should be evaluated at $x = h/2$, $x = 1 - h/2$, $y = h/2$ and $y = 1 - h/2$, respectively, as follows [1]: at the upper and bottom surfaces

$$\begin{aligned} p_{i,0} = p_{i,1} &- \frac{1}{4Re}(\zeta_{i+1,1} + \zeta_{i+1,0} - \zeta_{i-1,1} - \zeta_{i-1,0}) \\ &+ \frac{h}{4}u_{i,1}(\zeta_{i,1} + \zeta_{i,0}), \end{aligned} \qquad (22)$$

$$\begin{aligned} p_{i,N} = p_{i,N-1} &+ \frac{1}{4Re}(\zeta_{i+1,N} + \zeta_{i+1,N-1} - \zeta_{i-1,N} - \zeta_{i-1,N-1}) \\ &- \frac{h}{4}(u_{i,N} + u_{i,N-1})(\zeta_{i,N} + \zeta_{i,N-1}), \end{aligned} \qquad (23)$$

where $i = 1, 2, \ldots, N-1$, and at the left and right surfaces

$$\begin{aligned} p_{0,j} = p_{1,j} &+ \frac{1}{4Re}(\zeta_{1,j+1} + \zeta_{0,j+1} - \zeta_{1,j-1} - \zeta_{0,j-1}) \\ &- \frac{h}{4}\upsilon_{1,j}(\zeta_{1,j} + \zeta_{0,j}), \end{aligned} \qquad (24)$$

$$\begin{aligned} p_{N,j} = p_{N-1,j} &- \frac{1}{4Re}(\zeta_{N,j+1} + \zeta_{N-1,j+1} - \zeta_{N,j-1} - \zeta_{N-1,j-1}) \\ &+ \frac{h}{4}\upsilon_{N-1,j}(\zeta_{N,j} + \zeta_{N-1,j}), \end{aligned} \qquad (25)$$

where $j = 1, 2, \ldots, N-1$.

4. ITERATIVE METHODS FOR NUMERICAL STEADY-STATE SOLUTIONS

In this Section we discuss four iterative methods that we used to obtain the steady-state solutions to equations (14)-(25) [18]. Let $\boldsymbol{x}$ be an n-column vector, let the superscript T refer to the transposition, and let us be

$$\boldsymbol{x}^T = (\zeta_{1,1}, \ldots, \zeta_{N-1,1}; \zeta_{1,2}, \ldots, \zeta_{N-1,2}; \ldots; \zeta_{1,N-1}, \ldots, \zeta_{N-1,N-1}), \qquad (26)$$

or

$$\boldsymbol{x}^T = (\psi_{1,1}, \ldots, \psi_{N-1,1}; \psi_{1,2}, \ldots, \psi_{N-1,2}; \ldots; \psi_{1,N-1}, \ldots, \psi_{N-1,N-1}), \qquad (27)$$

or

$$\boldsymbol{x}^T = (p_{1,1}, \ldots, p_{N-1,1}; p_{1,2}, \ldots, p_{N-1,2}; \ldots; p_{1,N-1}, \ldots, p_{N-1,N-1}), \qquad (28)$$

where the $(N-1)^2$ variables $\zeta_{i,j}$, $\psi_{i,j}$ and $p_{i,j}$ $(i, j = 1, 2,..., N-1)$ at the interior grid points are unknowns in (14), (15) and (20), respectively. Hence we can write (14), (15) and (20) as a linear system of n $(= (N-1)^2)$ equations in the form $A\boldsymbol{x} = \boldsymbol{b}$, where $A = \{a_{i,j}\}$ is an $n*n$ coefficient matrix, and $\boldsymbol{b}$ an n-vector. Let $D = diag\,(a_{11},..., a_{nn})$ be a diagonal matrix containing the diagonal elements of A. The matrix A for (14), (15) or (20) is a block tridiagonal matrix of the form

$$A = \begin{bmatrix} C_{11} & D_{12} & & & \\ D_{21} & \ddots & \ddots & & \\ & \ddots & \ddots & D_{N-2\ N-1} & \\ & & D_{N-1\ N-2} & C_{N-1\ N-1} & \end{bmatrix}, \qquad (29)$$

where all the tridiagonals C_{ii} and diagonals $D_{i,j}$ $(i, j = 1, 2,..., N-1)$ are $N1*N1$ (where $N1 = N-1$) matrices of the forms

$$C_{ii} = \begin{bmatrix} a_{(i-1)N1+1\ (i-1)N1+1} & a_{(i-1)N1+1\ (i-1)N1+2} & & \\ a_{(i-1)N1+2\ (i-1)N1+1} & \ddots & \ddots & \\ & \ddots & \ddots & a_{iN1-1\ iN1} \\ & & a_{iN1\ iN1-1} & a_{iN1\ iN1} \end{bmatrix}, \qquad (30)$$

$$D_{ij} = \begin{bmatrix} a_{(i-1)N1+1\ (j-1)N1+1} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & a_{iN1\ jN1} \end{bmatrix}, \qquad (31)$$

respectively.

4.1 Jacobi's method

Given the linear system $A\boldsymbol{x} = \boldsymbol{b}$, the Jacobi iteration follows from the splitting $A = D - (D - A)$, which leads to the iteration

$$x^{k+1} = Hx^k + d, \; k = 0, 1, 2,..., \qquad (32)$$

where the superscript indicates the iteration number, and

$$H = D^{-1}(D - A), \; d = -D^{-1}b. \qquad (33)$$

4.2 Red-black ordering

The grid points are divided into two classes, red (or odd) and black (or even), and then ordered left to right, bottom to top within each class. The unknowns at the grid points are ordered analogously. Then the given linear system $A\boldsymbol{x} = \boldsymbol{b}$ will become of the form

$$\begin{bmatrix} D_R & C_1 \\ C_2 & D_B \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix}, \qquad (34)$$

where under reorderings

$$D = \begin{bmatrix} D_R & \\ & D_B \end{bmatrix}, \; \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{bmatrix}, \; \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix}. \qquad (35)$$

Then the red-black iteration is

$$\boldsymbol{x}_R^{k+1} = D_R^{-1}(\boldsymbol{b}_1 - C_1\boldsymbol{x}_B^k), \qquad (36)$$

$$\boldsymbol{x}_B^{k+1} = D_B^{-1}(\boldsymbol{b}_1 - C_2\boldsymbol{x}_R^{k+1}), \qquad (37)$$

which show to uncouple into the two separate parts.

4.3 CG method

Given the linear system $A\boldsymbol{x} = \boldsymbol{b}$, let $(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{y}$ be the inner product and $\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k$ the residual at the kth step. The conjugate gradient (CG) iteration is the following scheme:

```
Choose $\boldsymbol{x}^0$, set $\boldsymbol{p}^0 = \tau^0$, compute $(\tau^0, \tau^0)$,
for $k = 0, 1, 2,....,$
$\alpha_k = -(\tau^k, \tau^k)/(\boldsymbol{p}^k, A\boldsymbol{p}^k)$,
$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k\boldsymbol{p}^k$,
$\tau^{k+1} = \tau^k + \alpha_k A\boldsymbol{p}^k$,                    (38)
if not converge, continue,
$\beta_k = (\tau^{k+1}, \tau^{k+1})/(\tau^k, \tau^k)$,
$\boldsymbol{p}^{k+1} = \tau^{k+1} + \beta_k\boldsymbol{p}^k$.
```

4.4 ADI method

Given the linear system $A\boldsymbol{x} = \boldsymbol{b}$, let $A = H + V$ be the following splitting:

$$H = \begin{bmatrix} T_{11} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & T_{N-1\,N-1} \end{bmatrix},$$

$$V = \begin{bmatrix} D_{11} & D_{12} & & \\ D_{21} & \ddots & \ddots & \\ & \ddots & \ddots & D_{N1-1\,N1} \\ & & D_{N1\,N1-1} & D_{N1\,N1} \end{bmatrix}, \quad (39)$$

where H is a block diagonal, V is a block tridiagonal, and all the tridiagonals T_{ii} and diagonals D_{ii} (i = 1, 2,..., N – 1) are $N1*N1$ matrices of the forms

$$T_{ii} = \begin{bmatrix} a_{(i-1)N1+1\,(i-1)\,N1+1/2} & a_{(i-1)N1+1\,(i-1)N1+2} & & \\ a_{(i-1)N1+2\,(i-1)N1+1} & \ddots & \ddots & \\ & \ddots & \ddots & a_{iN1-1\,iN1} \\ & & a_{iN1\,iN1-1} & a_{iN1\,iN1/2} \end{bmatrix}, \quad (40)$$

$$D_{ii} = \begin{bmatrix} a_{(i-1)N1+1\,(i-1)N1+1/2} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & a_{iN1\,iN1/2} \end{bmatrix}, \quad (41)$$

respectively. Then the alternating direction implicit (ADI) iteration is

$$(\alpha_i I + H)\,\boldsymbol{x}^{k+1/2} = (\alpha_i I - V)\,\boldsymbol{x}^{k} + \boldsymbol{b}, \quad (42)$$

$$(\alpha_i I + V)\,\boldsymbol{x}^{k+1} = (\alpha_i I - H)\,\boldsymbol{x}^{k+1/2} + \boldsymbol{b}, \quad (43)$$

where $\alpha_i > 0$ ($i = \zeta$, ψ or p) is a constant.

4.5 Remarks

(1) Initial values

We set in all the iterative solution methods

$$\boldsymbol{x}^0 = \mathbf{o}. \quad (44)$$

(2) Iteration number of ψ

Let M_ψ be the iteration number of ψ per each iteration of ζ. For the red-black, CG or ADI method, we set M_ψ = 1. For the Jacobi method, we set M_ψ = 4, of which case we call J1. Talking from the point of view of overheads, only the Jacobi method incurres overheads due to hold and update the current values of $\psi^k_{i,j}$, but only when we set M_ψ = 1, we can remove these overheads, of which case we call J2.

(3) Relaxation parameters

We used the following relaxation parameters, ω_i ($i = \zeta$, ψ or p):

$$\hat{\boldsymbol{x}}^{k+1} = \boldsymbol{x}^k + \omega_i(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k),\ \boldsymbol{x}^{k+1} = \hat{\boldsymbol{x}}^{k+1}. \quad (45)$$

(4) Convergence test

In order to test for convergence of the iterates, we used the following test:

$$max_{\,1\le i\le n} \left| x_i^{k+1} - x_i^k \right| \le 10^{-6}. \quad (46)$$

(5) Steady flow methods

It is known that some steady-state flow iterative methods are strictly equivalent to time-dependent unsteady flow methods, with under- and over-relaxation adjustments being equivalent to Δt changes, and most steady-state iterative methods are at least analogous to time-dependent unsteady methods [24]. Hence, we can see the transient behavior of the flow by steady flow methods too until iterative solutions converge to the steady-state solution.

5. RESULTS OF PARALLEL COMPUTATIONS OF SQUARE CAVITY FLOWS ON THE NWT COMPUTER SYSTEM

A large amount of numerical information on the Reynolds number range of $Re = 10^{-6} \sim 10^5$ has been collected during this investigation. All the computations were executed in parallel on the NWT computer system. In this Section some selected results are presented with the particular aim of describing the asymptotic behaviour of square cavity flows at the high Reynolds numbers. Furthermore we show comparisons between results of square cavity flows at a fixed Reynolds number in the cases that we vary the selected number, *pe*, of the processing elements, the solution method and the grid size, respectively.

5.1 Effect of the Reynolds number on the behaviour of square cavity flows

First of all, we indicate the process of stabilization of the eddy system in the square cavity (see Figure 1). Square cavity flows at the representative Reynolds numbers are shown in Figures 2-10, where we note that (1) and (2) of Figure 6 agree almost completely with Figure 3 and Figure 4 in [7], respectively. The computational conditions used to obtain these Figures are given in Table 1, and the values of minimum, maximum and each contour for ζ, ψ and p given in Table 2.

The flows on the low Reynolds number range of $Re = 10^{-6} \sim 10^3$ are characterized by three eddies: one, the primary eddy and two other, secondary eddies located at the corners of the bottom wall (see Figures 2-4). As Re increases beyond 10^3, the secondary eddy located at the upper part of the left wall appears within the cavity (see Figure 5). As Re increases furthermore, the tertiary and fourth-order eddies located at the corners of the bottom wall become to appear (see Figures 6-7). As $Re > 2*10^4$, however, the secondary or/and tertiary eddies located at the left or/and right corners of the bottom wall become unsteady (see Figures 8-10). Figure 11 shows effect of Re on the profiles of u, υ, ζ and p through the center of the primary vortex at $Re = 10^{-6}$, 10^2, 10^3, 10^4 and $2*10^4$ corresponding to Figures 2-4 and 6-7, respectively.

(1) Primary vortex

Figure 12 shows the location of the center of the primary vortex as a function of the Reynolds number. Until the Reynolds number comes up to 120, the center of the primary vortex moves upstream. As the Reynolds number increases further, it moves monotonously towards the center of the cavity. In the inviscid limit the total pressure is conserved along the streamlines, in which case the total-pressure contours should coincide with the streamlines. Hence the close similarity between (1) and (3) of Figures 5-10 in the core region is evident. On the other hand, the vorticity changes rapidly at the boundaries and is flat in the middle of the cavity at the high Reynolds numbers.

(2) Secondary vortices

The secondary eddy located at the right corner of the bottom wall grows with Re, attains maximum at $Re = 10^3$, and at $Re > 10^3$ begins to thin out monotonously. It is steady at $Re \leq 2*10^4$, but becomes unsteady at $Re > 2*10^4$. On the other hand, the secondary eddy located at the left corner of the bottom wall grows with Re, attains maximum at $Re = 7*10^3$, and at $Re > 7*10^3$ begins to thin out in monotone. It is steady at $Re \leq 5*10^4$, but becomes unsteady at $Re > 5*10^4$. The third secondary eddy located at the upper part of the left wall begins to appear at $Re > 10^3$, and grows monotonously with Re. It is steady at all the Reynolds numbers.

(3) Tertiary and fourth-order vortices

The tertiary eddies located at the corners of the bottom wall appear slightly at $Re = 10^{-6}$, and grows with Re. The right tertiary eddy is steady until $Re \leq 2*10^4$, but at $Re > 2*10^4$ becomes unsteady. The left tertiary eddy is steady until $Re \leq 5*10^4$, but becomes unsteady at $Re > 5*10^4$. On the other hand, the right fourth-order eddy at the corner of the bottom wall becomes to appear at $Re \geq 2*10^3$, and the left fourth-order eddy at $Re \geq 3*10^4$. These repeat to slightly grow and decay, as the secondary and tertiary eddies at both the corners of the bottom wall become to be unsteady.

5.2 Comparison between results obtained by varying *pe* in the parallel processing

Comparison between $pe = 2^i$ ($i = 0, 1, \ldots, 4$) for each solution method was done for square cavity flows at $Re = 10^3$, where computational conditions are as shown in Table 3. Complete agreement was obtained for the Jacobi, red-black and ADI methods. For the CG method, complete agreement was obtained among u's, υ's and ζ's, but slightly disagreement was indicated among p's as shown in Figure 13, of which reason is due to be necessary for this method to compute a global summation in the parallel processing each iteration, as seen from (38).

5.3 Comparison between results obtained by varying the solution method

Comparison between results obtained for the four solution methods in case of $Re = 10^3$ and $pe = 16$ showed that complete agreement with profiles for $Re = 10^3$ of (1)-(4) of Figure 11 was obtained among u's, υ's and ζ's, but that a little disagreement was indicated among p's as shown in Figure 14, where computational conditions are as shown in Table 3.

5.4 Comparison between results obtained by varying the grid size

Comparison between results obtained for $2^i k*2^i k$ ($i = -2, 0, 1, \ldots, 4$; $1k = 1024$) grids at $Re = 10^3$ by the red-black method is shown in Figure 15, where computational conditions are given in Table 4, including the values of the execution time until convergence and the center of the primary vortex obtained when we executed in parallel under the selected number of the processing elements mentioned in this Table. Sufficient agreement was shown among u's, υ's and ζ's, but considerable disagreement was indicated among p's. Figure 16 shows contours of the total-pressure solution for each grid size, of which values are same as given for Fig. 4 in Table 2. We cannot make its reason clear at present, but there is a possibility of hiding an elementary problem. By the way, we note the followings:

(1) For $8k*8k$ and $16k*16k$ grids, data of 128*128 grid points of the intermediate computational results for ζ and ψ were stored at regular intervals into the secondary memory device, and retrieved to restore full data by means of interpolation before start of the successive iteration. This process had been continued until convergence.

(2) Computations for a $16k*16k$ grid were conducted for another purpose of obtaining data concerned with the mean time between failure (MTBF) of the NWT computer system.

6. PROGRAM PERFORMANCE ON THE NWT COMPUTER SYSTEM

In this Section we indicate results for the actual performance of the parallelised square cavity programs on the NWT computer system. Most of the overheads incurred in the parallel processing arise from the time spent in system software routines supporting user programs. Hence the obtained results in this Section apply only to the system software had been available at the NAL during the period April to June 1993 which we call v1, and during the period April to May 1994 which we call v2. The v2 is an improved version of the v1. By the way all the programs were run with other users on the NWT computer system.

6.1 Measurements of overheads of the used NWT-Fortran-compiler directives

We had used the following NWT-Fortran-compiler directives for MIMD computings of the square cavity programs:

(1) PARALLEL REGION(CD1)/END PARALLEL REGION (CD2); Only the program portion between these statements is executed in parallel by multiple processing elements.

(2) SPREAD DO (CD3); This statement specifies partitioning of repetitions of the DO statement, and execution of each segment on the specified processing element.

(3) SPREAD MOVE (CD4); This statement specifies assignment between local variables and global variables on each processing element.

(4) END SPREAD (CD5); This statement verifies the exit of the SPREAD DO/MOVE statement.

(5) END SPREAD SUM (abc) (CD6); By this statement the global summation operation among the subregions generated by the spread-do-construct is performed at termination of the spread-do-construct.

(6) OVERLAPFIX (CD7); This statement specifies replacement of the overlap range of partitioned local arrays with overlap, with the values of the range corresponding on another processing elements.

(7) MOVEWAIT (CD8); This statement verifies completion of assignment and data transfer that are started by the OVERLAPFIX and SPREAD MOVE statements, respectively.

Figure 17 shows measurements of each overhead for the above-mentioned compiler directives against the selected number of the processing elements, $pe = 2^i$ (i = 1, 2,..., 7). For (5) of this Figure we note the following: There is an optimum number of the processing elements corresponding to the amount of the data transfer. The smaller pe is than the optimum number, the more the data-transfer overhead increases due to bottleneck of the amount of the data transfer. The larger pe is than the optimum number, the more the data-transfer over-head increases due to bottleneck of the packet creation.

Furthermore we had used the following NWT-Fortran-compiler directives for MIMD computings without any overhead:

(8) PROCESSOR (CD9); This statement is used to declare the number of the processing elements required to execute a program in parallel.

(9) INDEX PARTITION (CD10); This statement is used to indicate the index range of the association of DO loops and arrays with processing elements and the partition type, and to specify the overlap if necessary.

(10) GLOBAL (CD11)/LOCAL (CD12); These statements specify the allocation of variables to the

global and local memory spaces, respectively.

(11) EQUIVALENCE (CD13); This statement specifies the sharing of the storage by two or more variables in a program. When a global variable and a local variable are combined, a global and local memory spaces are shared.

6.2 Some remarks on the parallelised square cavity programs

The flow of the square cavity programs is as follows:

(A) Declare the CD9-CD13 statements, etc.,
(B) insert the CD1 statement,
(C) set the initial values of (16), (21), (44), etc.,
(D) for $k = 1, 2, 3,....$(iterations for ζ and ψ),
(D.1) hold the current values of $\zeta^k_{i,j}$ and $\psi^k_{i,j}$
(D.2) update the boundary values of $\zeta^k_{i,j}$ by (17) and (18)
(D.3) compute the internal values of $\zeta^{k+1}_{i,j}$ and $\psi^{k+1}_{i,j}$ by (14), (15) and (45),
(D.4) test for convergence of the iterates by (46), and if converge, continue,
(E) compute the internal values of $u_{i,j}$ and $\upsilon_{i,j}$ by (19),
(F) compute the values of the right hand side of (20),
(G) for $k = 1, 2, 3,....$ (iterations for p),
(G.1) hold the current values of $p^k_{i,j}$,
(G.2) update the boundary values of $p^k_{i,j}$ by (22)-(25),
(G.3) compute the internal values of $p^{k+1}_{i,j}$ by (20) and (45),
(G.4) test or convergence of the iterates by (46), and if converge, continue,
(H) compute the particular solution of p by (12),
(I) insert the CD2 statement,
(J) store the computational results, (47)

where only the program portions for (D.3), (F) and (G.3) differ with each other every solution method.

Let N_C be the number used the pairs of CD3 and CD5 in the program portion (C), and so on provided that the numbers for (D) and (G) are the values per each iteration. These numbers for the portions common to all the parallelised square cavity programs are as follows: $N_C = N_E = 1$ (pair of CD3 and CD5), $N_{D.1} = N_{G.1} = 1$ (pair of CD3 and CD5) and 1 (pair of CD7 and CD8), $N_{D.2} = N_{G.2} = 2$ (pair of CD3 and CD5) and 1 (pair of CD7 and CD8), $N_{D.4} = N_{G.4} = 1$ (pair of CD3 and CD6), and $N_H = 1$ (pair of CD3 and CD6) and 1 (pair of CD3 and CD5). On the other hand, the numbers of $N_{D.3}$, N_F and $N_{G.3}$ are different every solution method used as follows: Let HU used below mean to hold or update the current values of $\zeta^k_{i\,j}$ or $\psi^k_{i\,j}$.

(1) Jacobi method

The scheme of this method is given by (32). Hence for J1, $N_{D.3} = 10$ of which 5 are used for HU (pair of CD3 and CD5) and 4 (pair of CD7 and CD8), $N_F = 1$ (pair of CD3 and CD5), an $N_{G.3} = 2$ (pair of CD3 and CD5). For J2, $N_{D.3} = N_{G.3} = 2$ (pair of CD3 and CD5), and $N_F = 1$ (pair of CD3 and CD5).

(2) Red-black ordering

The scheme of this method is given by (36)-(37). Hence $N_{D.3} = N_{G.3} = 8$ (pair of CD3 and CD5) and 1 (pair of CD7 and CD8), and $N_F = 1$ (pair of CD3 and CD5).

(3) CG method

The scheme of this method is given by (38). Hence $N_{D.3} = 10$ of which 2 are used for HU (pair of CD3 and CD5), 2 (pair of CD7 and CD8) and 4 (pair of CD3 and CD6), $N_F = 1$ (pair of CD3 and CD5), and $N_{G.3} = 5$ of which 1 is used for HU (pair of CD3 and CD5), 1 (pair of CD7 and CD8) and 2 (pair of CD3 and CD6).

(4) ADI method

The scheme of this method is given by (42)-(43). Hence $N_{D.3} = 26$ of which 4 are used for HU (pair of CD3 and CD5), 4 (set of CD4, CD5 and CD8) and 4 (pair of CD7 and CD8), $N_F = 1$ (pair of CD3 and CD5) and 1 (set of CD4, CD5 and CD8), and $N_{G.3} = 15$ of which 2 are used for HU (pair of CD3 and CD5), 2 (set of CD4, CD5 and CD8) and 2 (pair of CD7 and CD8).

Finally we note the followings with respect to the partition of two-dimensional arrays adopted in this study: All the computations of (12), (14)-(25), (32), (36)-(38) and (42)-(46) can be executed in parallel. For the Jacobi, red-black or CG method, we had adopted the band partition of the index range in the y-direction.

Hence in FORTRAN we can take advantage of continuous loading and continuous storage on each register-to-register processing element except when (18) and (24)-(25) are executed, which time becomes to periodic loading and periodic storage. On the other hand, as is obvious from (42) and (43), for the ADI method, we must adopt the band partition of the index range in the y-direction for computation of (42) and of the index range in the *x*-direction for computation of (43), where the case of the former is continuous loading and continuous storage and the case of the latter becomes both periodic. If we would adopt the band partition of the index range in only the *y*-direction for computations of (42) and (43), the executing time should increase about three hundred times. Furthermore we must suppress the occurrence of the memory-bank conflict in case of periodic loading and periodic storage. When the bank conflict would occur, the performance of the execution should deteriorate about half.

6.3 Definition of several actual rates for the square cavity programs

We use below the following actual rates defined for the square cavity programs:

τ_1 = actual rate of the part (D.3) in (47)
= actual rate of the kernel part for iterations of ζ and ψ,
τ_2 = actual rate of the part (D) in (47)
= actual rate of the overall part for iterations of ζ and ψ,
τ_3 = actual rate of the part (D) excepted (D.4) in (47)
= actual rate of the overall part for iterations of ζ and ψ, when the convergence test is excepted,
τ_4 = actual rate of the part (D) excepted (D.1), a pair of CD7 and CD8 in (D.2), and pairs of CD3 and CD5 used for HU and pairs of CD7 and CD8 in (D.3) as well as (D.4) in (47),
τ_5 = actual rate of the part (D) excepted (D.4) in (47) in case of using the loop coalescing technique,

where τ_4 indicates the actual rate when execution of the local memory access statements with no arithmetic operation, etc. is intentionally suppressed, and hence we define the following:

$d_{ma} = \tau_4/\tau_3$ = degree of degradation of the actual rate of the part (D) excepted (D.4) in (47) mainly due to the local memory access bottleneck.

By the way the loop coalescing is the technique that restructures certain types of multiply nested loops into single parallel loops. When we adopt this technique to the square cavity programs, each code segment must be executed conditionally in the transformed loop to assure correct execution of the original code segment corresponding. Hence we can almost double the amount of arithmetic operations every parallel loop by the loop coalescing, but a new source of degradation of the actual rate is introduced.

6.4 The characteristic parameters of the square cavity programs on the NWT computer system

In practise we find for many parallelised programs that as *pe* becomes larger, synchronisation and other overheads usually increase rapidly with *pe*, with the result that there is often a maximum in the program performance and with a subsequent reduction in performance as *pe* further increases. This measured behaviour can be fitted to the function [13]

$$\tau(pe)=\tau_\infty\left[1+\frac{pe_{1/2}}{pe}\left\{1+\frac{1}{n-1}\left(\frac{pe}{\tilde{pe}}\right)^n\right\}\right]^{-1}, \quad (48)$$

where

$$\tau_\infty=\frac{1}{t_{ser}},\ pe_{1/2}=\frac{t_{par}}{t_{ser}}, \quad (49)$$

and

τ = actual performance of the parallelised program,
τ_∞ = asymptotic performance of the parallelised program
= maximum possible performance of the parallelised program when overheads are negligible
= Amdahl ceiling,
$pe_{1/2}$ = number of the processing elements required to achieve half of the asymptotic performance,

$\tilde{pe}$ = value of pe when the maximum performance given below occurrs
= value of pe more than which the parallelised program cannot usefully utilise,
n = index of synchronisation and other overheads
= rapidity with which overheads increase pe,
t_{ser} = time of execution for the essentially serial part of the parallelised code,
t_{par} = time when executed sequentially for the part of the parallelised code that can be executed in parallel.

The maximum performance, τ_{max}, occurs when $pe = \tilde{pe}$, and has the value

$$\tau_{max} = \tau_\infty \left[1 + \frac{n}{n-1} \left(\frac{pe_{1/2}}{\tilde{pe}} \right) \right]^{-1}. \quad (50)$$

The overhead of synchronisation and others is represented by the factor in braces { } of (48). Hence it is desirable for n to be as small as possible, a larger value of pe means a smalle roverhead, and as $\tilde{pe} \to \infty$, $\tau_{max} \to \tau_\infty$. The values of the characteristic parameters mentioned above are likely to vary considerably between different computer software systems.

Figures 18-21 show measurements of the actual performance τ_2 of the parallelised square cavity program used each solution method against the selected number of the processing elements, $pe = 2^i$ (i = 0, 1,..., 7). As shown in these Figures, the actual performance τ_2 of each parallelised program is fitted very well by equation (48) in the range of $1 \le pe \le 128$, with the estimated values of the characteristic parameters given in Table 5, and the degree of improvement on the actual rate τ_2 of v2 to v1 is at its maximum each solution method as follows: For the Jacobi method (J1), it is 2.85-fold (2^8*2^8 grid) and 2.50-fold ($2^{10}*2^{10}$ grid). For the red-black method, it is 2.84-fold (2^8*2^8 grid), 2.24-fold ($2^{10}*2^{10}$ grid), 1.65-fold ($2^{11}*2^{11}$ grid), 1.21-fold ($2^{12}*2^{12}$ grid), 1.06-fold ($2^{13}*2^{13}$ grid) and 1.07-fold ($2^{14}*2^{14}$ grid). For the CG method, it is 3.73-fold (2^8*2^8 grid) and 1.58-fold ($2^{10}*2^{10}$ grid). For the ADI method, it is 1.28-fold (255*255 grid).

6.5 Effect of the grid size on the actual performance of the square cavity programs on the 128-processing element system

Figure 22 shows effect of the grid sizes of 2^i*2^i (i = 8, 9,..., 14) provided that the maximum grid size for only the CG method is 12,000*12,000, on the actual rate τ_3 of the parallelised square cavity program used each solution method for v2 when pe = 128. As shown in this Figure, the actual rate τ_3 of each parallelised program is fitted very well by the equation

$$\tau_3 = \tau_\infty \left[1 + \left(\frac{s_{1/2}}{s} \right)^2 \right]^{-1}, \quad (51)$$

with the following estimated values of the characteristic parameters:

Jacobi (*J*1) for v2 :
$\tilde{\tau}_\infty$ = 1.037376*10^5 (*Mflop/sec*),
$s_{1/2}$ = 2.116524*10^3 (*flop*), (52)

Jacobi (*J*2) for v2 :
$\tilde{\tau}_\infty$ = 1.346469*10^5 (*Mflop/sec*),
$s_{1/2}$ = 2.286338*10^3 (*flop*), (53)

red-black for v2 :
$\tilde{\tau}_\infty$ = 6.835579*10^4 (*Mflop/sec*),
$s^{1/2}$ = 2.574064*10^3 (*flop*), (54)

CG for v2 :
$\tilde{\tau}_\infty$ = 1.099030*10^5 (*Mflop/sec*),
$s_{1/2}$ = 3.223610*10^3 (flop), (55)

where
s = N +1 where the grid size is s * s,
$\tilde{\tau}_\infty$ = maximum performance of the parallelised program,
$s_{1/2}$ = square root of the half-performance grain size of the parallelised program.

We note here the followings compared results of [11] with results of this Subsection: The former has considered the timing of the single work segment which can be distributed amongst multiple processing elements, and the latter the timing of the complete program comprising many work segments. Both these actual performances, however, can be fitted very well by the pipeline function of the same form as (51), because the programs of the latter except for the ADI method have been parallelised so as to become almost all the local memory access. Furthermore, for example, τ_∞ for the Jacobi method (J2) of (53) is 2.74 times (dyads) and 1.93 times (triads) as large as the values for *pe* =128 in Table 2 of [11], and $s_{1/2}^2$ is 3.21 times (dyads) and 2.59 times (triads). This means that for the complete program increases frequency of the chaining of two vector instructions and the simultaneous use of both the floating-point multiply and add pipelines with all vectors, so that the maximum program performance raises considerably and hence the half-performance grain size increases as well.

6.6 Measurements of the maximum actual performance of the square cavity programs on the NWT computer system

Let τ_{3mm} be the measured maximum of the actual rate τ_3. Table 6 and Figure 23 show measurements of τ_{3mm} of the parallelised square cavity program used each solution method against the selected number of the processing elements, $pe = 2^i$ (i = 0, 1,..., 7), where the grid sizes used to obtain these results are given in Table 7. As shown in Figure 23, τ_{3mm} of the parallelised programs except for the ADI method are fitted very well by the following equations:

Jacobi (*J*1) for v1 :

$$\tau_{3mm} = 7.984252*10^2\, pe\ (Mflop/sec), \qquad (56)$$

Jacobi (*J*1) for v2 :

$$\tau_{3mm} = 8.336185*10^2\, pe\ (Mflop/sec), \qquad (57)$$

Jacobi (*J*2) for v2 :

$$\tau_{3mm} = 1.058900*10^3\, pe\ (Mflop/sec), \qquad (58)$$

red-black for v1 :

$$\tau_{3mm} = 5.419430*10^2\, pe\ (Mflop/sec), \qquad (59)$$

red-black for v2 :

$$\tau_{3mm} = 5.362091*10^2\, pe\ (Mflop/sec), \qquad (60)$$

CG for v1 :

$$\tau_{3mm} = 7.814897*10^2\, pe\ (Mflop/sec), \qquad (61)$$

CG for v2 :

$$\tau_{3mm} = 8.135356*10^2\, pe\ (Mflop/sec). \qquad (62)$$

By the way there is no appropriate equation fitted to measurements of τ_{3mm} for the ADI method.

We here remark the followings for the obtained results:

(1) The reason that actual rates of the red-black method are inferior to ones of the other methods is because the amount of works in each parallel section becomes half the others, as seen from (36) and (37).

(2) As seen from (1) and (2) of Table 6, the degree of degradation of τ_3 mainly due to the local memory access bottleneck is 1.10- to 1.42-fold. The reason that the values of d_{ma} for the Jacobi method (J1) are larger than ones for the other methods is because dimensional computer variables are stored to and retrieved from the local memory each minor iteration for *p*.

(3) For example, τ_{3mm} for the Jacobi method (J2) of (58) is 2.75 times (dyads) and 1.94 times (triads) as large as the values of (14) and (15) of [11], respectively. These values are same as ones mentioned in Subsection 6.5.

(4) To apply the ADI method to the square cavity program is not advisable, because of incurring a large amount of communication overheads due to the global memory access. For three-dimensional cavity programs, however, the scheme of the ADI method as well as the other methods can be parallelised so as to become almost all the local memory access, and hence the ADI method becomes useful as well [12]. Furthermore the clear improvement that τ_{3mm} for v2 are about 1.17 times as large as ones for v1 is based on the fact that communication overheads for v2 have decreased about 5.0- to 5.7-fold for v1 [11].

6.7 Speedup and efficiency of the parallelised programs on the NWT computer system

The speedup, S_{pe}, of a parallelised program and the efficiency or effect of the number of the processing elements, E_{pe}, of a parallelised program with respect to itself are defined as follows [18]:

$$S_{pe} = \frac{T_1}{T_{pe}},\ E_{pe} = \frac{S_{pe}}{pe}, \qquad (63)$$

where

T_1 = execution time of a program using a single processing element by SIMD computing,

T_{pe} = execution time of a parallelised program using *pe* processing elements.

The square cavity program of such a large-scale that the grid size is greater than 1,500*1,500, however, cannot be executed by SIMD computing on the NWT computer system. Hence we must devise the method to estimate the value of T_1 for such the large-scale programs.

We can use two time subroutines measured in microseconds on the NWT computer system: One, the GETTOD which measures the wall clock time and the other, the CLOCKV which measures the operating time of the central processing unit and the operating time of the pipeline unit or vector processing unit for each processing element. The elapsed time, ELP_{pe}, and the operating time of the central processing unit, CPU_{pe}, and the operating time of the vector processing unit, VPU_{pe}, between two points in a program using pe processing elements can be measured by using a pair of the GETTOD and CLOCKV, respectively, and generally $ELP_{pe} \geq CPU_{pe} \geq VPU_{pe}$, where these values mean the time for SIMD computing only when $pe = 1$ and the time for MIMD computing when $pe > 1$. For MIMD computing on the NWT computer system, the compiler directives of CD3-CD8 are treated so as to wait for completion of these processes by means of the spin-loop on purpose to be decreased delays due to insert these directives, and hence overheads due to these directives are added in CPU_{pe}.

Let I be the number of iterations, VTC_{pe} the ratio of VPU_{pe} to CPU_{pe}, DRW_{pe} the division ratio of works into each processing element, TWK_{pe} the total work, TOH_{pe} the total overhead, $WK1_{pe}$ the work per one iteration, $OH1_{pe}$ the overhead per one iteration, and DDS_{pe} the degree of degradation of the speedup. Then we have the following relations:

$$VTC_{pe} = \frac{VPU_{pe}}{CPU_{pe}},\ DRW_{pe} = \frac{VPU_{pe}}{VPU_1},$$
$$TWK_{pe} = \frac{VPU_{pe}}{VTC_1},\ TOH_{pe} = CPU_{pe} - TWK_{pe}, \quad (64)$$

$$WK1_{pe} = \frac{TWK_{pe}}{I},\ OH1_{pe} = \frac{TOH_{pe}}{I},$$
$$DDS_{pe} = \frac{WK1_{pe}}{WK1_{pe} + OH1_{pe}}, \quad (65)$$

from which we can obtain

$$S_{pe} = \frac{DDS_{pe}}{DRW_{pe}} = \frac{CPU_1}{CPU_{pe}}. \quad (66)$$

We now consider the case that the grid size is considerably large. Then, from the reason mentioned above, we have the following relations:

$$CPU_1 \approx ELP_1 = T_1,\ CPU_{pe} \approx ELP_{pe} = T_{pe}, \quad (67)$$

from which and (66) we can obtain the definition of S_{pe} of (63). As seen from Table 7, the grid sizes used to obtain Table 6 satisfy the condition necessary to derive (67). Therefore we can estimate the value of T_1 from the following relation:

$$T_1 \approx CPU_1 = \frac{VPU_1}{VTC_1} = \frac{VPU_{pe}}{DRW_{pe}*VTC_1}$$
$$\approx \frac{VPU_{pe}*pe}{VTC_1}, \quad (68)$$

from which and (66)-(67) we can obtain

$$S_{pe} = \frac{VPU_{pe}}{ELP_{pe}} * \frac{pe}{VTC_1}. \quad (69)$$

Various experiments have shown that the larger the values of VTC_{pe} for the parallelised program are, i.e., the larger the grid size is, the better the value of S_{pe} computed from (69) becomes to coincide with one computed from (63).

Table 8, as one instance, shows comparison between the values of S_{pe} computed from (63) and (69). As seen from this Table, two values of S_{pe} coincide comparatively well despite for such a rather smaller grid size as 1,024*1,024. Hence we can expect that the larger the grid size is, the more the accuracy of S_{pe} estimated from (69) will be raised. Table 9 shows the speedup and efficiency computed from (69) and (63) for the case of τ_{3mm} and v2 of Table 6 provided that $VTC_1 = 1.0$ which results in the most underestimation, respectively.

7. BRIEF DISCUSSIONS ON THE SOLUTION OF THE SQUARE CAVITY PROBLEM, THE SOLUTION METHOD AND THE CONVERGENCE TIME

Firstly we remark the followings for the numerical solution of the square cavity problem:

(1) We think that the non-staggered grid adopted in this study is superior to the staggered grid when we

use the consistent finite-difference approximations for the square cavity problem, and that numerical solutions obtained by treating the steady-state equations are superior to ones obtained by marching the unsteady equations in time.

(2) The solution of ζ and ψ is scarcely influenced by varying the number of the processing elements in the parallel processing, the solution method and the grid size.

(3) The solution of p, however, is slightly influenced by varying the number of the processing elements in the parallel processing only when the CG method is adopted, a little influenced by varying the solution method, and considerably influenced by varying the grid size, as shown in Figure 16. We can't make its reason clear yet. This indicates difficulty in treating the pressure Poisson equation with Neumann boundary conditions that is yet in question.

Secondly we remark the followings for the solution method:

(1) The solution of ζ and ψ on the Reynolds number range of $Re = 0 \sim 5*10^3$ can be easily obtained whichever solution method is adopted. At $Re = 5*10^3 \sim 10^4$, however, it gradually becomes difficult to obtain this solution, and at $Re > 10^4$, at last, it seems hardly possible to obtain this solution by any single iterative soluton method. Hence it is necessary to alter the solution method adopted for computations until development of the boundary layer and after development of the boundary layer. The Jacobi and CG methods are very promising for computation of the former, and for computation of the latter, only the red-black method is hopeful.

(2) The Jacobi preconditioning, preconditioned CG and incomplete Choleski CG methods are useful to obtain the solution of ζ and ψ at small Reynolds numbers, but become useless as Re is larger.

(3) The solution of p at $Re \geq 10$ can be easily obtained whichever solution method is adopted. At $Re < 1$, however, we had better avoid adoption of the Jacobi method to obtain this solution.

Thirdly we remark the followings for the convergence time:

(1) At $Re \leq 600$, the time required until $\zeta^k_{i,j}$ and $\psi^k_{i,j}$ converge becomes longer in order of red-black, Jacobi (J1), Jacobi (J2), CG and ADI for SIMD computings provided that ω_i ($i = \zeta, \psi$) are optimum, and at $600 < Re \leq 5*10^3$, longer in order of Jacobi (J1), Jacobi (J2), red-black, CG and ADI. This time becomes much longer as Re increases. At $Re \geq 10^4$, we find difficulty in being converged to a solution if we did not use the Jacobi-red-black or CG-red-black combination as a solution method. For example, this time by the red-black method becomes more than double as large as one by the CG-red-black combination.

(2) At $Re < 100$, the time required until $p^k_{i,j}$ converge becomes longer in order of red-black, CG, ADI and Jacobi for SIMD computing, at $100 \leq Re \leq 600$ longer in order of red-black, Jacobi, ADI and CG, and at $Re > 600$ longer in order of Jacobi, red-black, ADI and CG. This time becomes shorter as Re increases, and at $Re > 10^3$ nearly constant for a fixed grid size.

Finally we mention that the solution method presented a larger actual rate is not always advantageous to be converged to a numerical solution of ζ, ψ and p, as seen from discussions mentioned above.

8. CONCLUSIONS

In this paper we reported parallel computations of incompressible viscous flow in a lid-driven square cavity on the NWT computer system, described the asymptotic behaviour of this flow at the high Reynolds numbers, examined effects of Re, pe, the solution method and the grid size on numerical solutions of ζ, ψ and p by MIMD computings, and measured the actual performance of the parallelised square cavity program adopted each solution method on the NWT computer system.

In Section 5 we firstly indicated the process of stabilization of the eddy system and the location of the center of the primary vortex in the square cavity on the Reynolds number range of $Re = 0 \sim 10^5$, secondly comparison between results obtained by varying pe in the parallel processing, thirdly comparison between results obtained by varying the solution method, and finally comparison between results obtained by varying the grid size. Its results showed that the degree of difference of ζ, ψ and p by variation of pe is negligible, but that the degree of difference of ζ, ψ and *p* by variation of the solution method is significant. Also its results showed that the degree of difference of ζ and ψ by variation of the grid size is negligible, but that the degree of difference of *pe* by variation of the grid size is very significant when the grid size is huge. Since this phenomenon does not happen in the ordinary grid size, the new problem in treating the pressure Poisson equation with the Neumann boundary condition is possible to happen. We cannot make its reason clear yet. This is our answer for the problem (1) mentioned in Introduction.

In Section 6 we firstly mentioned some remarks on the parallelised square cavity programs, and measured overheads of the NWT-Fortran-compiler directives used in these parallelised programs. This is our answer for the problem (3) mentioned in Introduction. Secondly we estimated the characteristic parameters, (τ_∞, $pe_{1/2}$, $\tilde{pe}$, n), of each parallelised program on the NWT computer system in case that the actual rate is considered as a function of *pe* for a fixed grid size. Thirdly we estimated the characteristic parameters, (τ_∞, $s_{1/2}$), of each parallelised program on the NWT computer system in case that the actual rate is considered as a function of the grid size for a fixed *pe*. This is our answer for the problem (4) mentioned in Introduction. Finally we measured the maximum actual performance of the parallelised square cavity program every solution method against *pe* on the NWT computer system, and estimated the speedup and efficiency of these programs. This is our answer for the problem (2) mentioned in Introduction.

We mention at the end that the system software of the NWT computer system had been available at the NAL during the period April to May 1994 (v2) has been drastically improved in comparison with the system software had been available during the period April to June 1993 (v1), as shown in Section 6. The degree of improvement on the actual rate τ_2 of v2 to v1 is at its maximum for a 256∗256 grid, 2.85-fold (Jacobi (J1)), 2.84-fold (red-black), 3.73-fold (CG), and for a 255∗255 grid, 1.28-fold (ADI).

9. REFERENCES

[1] S. Abdallah, Numerical solution for the pressure Poisson equation with Neumann boundary conditions using a non-staggered grid I, *Journal of Computational Physics* 70 (1987) 182-192.

[2] R.K. Agarwal and M. Douglas, A three-order-accurate upwind scheme for Navier-Stokes solutions at high Reynolds numbers, AIAA-81-0112 (1981).

[3] A.S. Benjamin and V.E. Denny, On the convergence of numerical solutions for 2-D flows in a cavity at large Re, *Journal of Computational Physics 33* (1979) 340-358.

[4] J.D. Bozeman and C. Dalton, Numerical study of viscous flow in a cavity, *Journal of Computational Physics 12* (1973) 348-363.

[5] O.R. Burggraf, Analytical and numerical studies of the structure of steady separated flow, *Journal of Fluid Mech. 24, part 1* (1966) 113-151.

[6] L.F .Donovan, A numerical solution of unsteady flow in a two-dimensional square cavity, *AIAA Journal 8, 3* (1970) 524-529.

[7] U. Ghia, K.N. Ghia and C.T. Shin, High-Resolutions for incompressible flow using the Navier-Stokes equations and a multigrid method, *Journal of Computational Physics 48* (1982) 387-411.

[8] D. Greenspan, Numerical studies of prototype cavity flow problems, *Computer Journal 12* (1969) 88-93.

[9] S. Hatayama, The characteristic parameters of the NWT computer system in the local memory access, *NAL TR* (submitted for publication).

[10] S. Hatayama, The characteristic parameters of the NWT computer system in the global memory access, *NAL TR* (submitted for publication).

[11] S. Hatayama, On an improvement on the NWT system software, *NAL TR* (to be submitted for publication).

[12] S. Hatayama, Elementary benchmarks of the NWT computer system and program performance, *Proc. of the 12th NAL Symp. on Aircraft Computational Aerodynamics, NAL SP-27* (1994).

[13] R.W. Hockney and C.R. Jesshope, *Parallel Computers 2* (Adam Hilger, Bristol, 1988).

[14] M. Kawaguti, Numerical solution of the Navier-Stokes equations for the flow in a two-dimensional cavity, *Journal of The Physical Society of Japan 16, 12* (1961) 2307-2315.

[15] R.D. Mills, On the closed motion of a fluid in a square cavity, *Journal of The Royal Aeronautical Society 69* (1965) 116-120.

[16] R.D.Mills, Numerical solutions of the viscous flow equations for a class of closed flows, *Journal of The Royal Aeronautical Society 69* (1965) 714-718.

[17] M. Nallasamy and K.K. Prasad, On cavity flow at high Reynolds numbers, *Journal of Fluid Mech. 79, part 2* (1977) 391-414.

[18] J.M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems* (Plenum Press, New York, 1988).

[19] F. Pan and A. Acrivos, Steady flows in rectangular cavities, *Journal of Fluid Mech. 28, part 4* (1967) 643-655.

[20] S.G. Rubin and P.K. Khosla, Polynomial interpolation methods for viscous flow calculations, *Journal of Computational Physics 24* (1977) 217-244.

[21] A.K. Runchal and M. Wolfshtein, Numerical integration procedure for the steady state Navier-Stokes equations, *Journal Mechanical Engineering Science 11, 5* (1969) 445-453.

[22] Staff of LRC, Numerical studies of incompressible viscous flow in a driven cavity, *NASA SP-378* (1975).

[23] R.F. Weiss and B.H. Florsheim, Flow in a cavity at low Reynolds number, *The Physics of Fluid 8, 9* (1965) 1631-1635.

[24] P.J. Roache, *Computational Fluid Dynamics* (Hermosa Publishers, Albuquerque, New Mexico, 1972).

Table 1 Computational conditions used to obtain Figures 2-10

Fig.	*Re*	*grid*	*solution method*	*relaxation parameters*		
				ω_ζ	ω_ψ	ω_p
2	$1 * 10^{-6}$	256 * 256	ζ, ψ, p : *red − black*	1.4	1.4	1.5
3	$1 * 10^{2}$	256 * 256	ζ, ψ, p : *red − black*	1.4	1.4	1.5
4	$1 * 10^{3}$	256 * 256	ζ, ψ, p : *Jacobi(J1)*	0.1	0.1	1.0
5	$2 * 10^{3}$	256 * 256	ζ, ψ, p : *Jacobi(J1)*	0.1	0.1	1.0
6	$1 * 10^{4}$	1024 * 1024	ζ, ψ : *CG − rb* p : *CG*	*CG* : 0.0093 *rb* : 0.45	*CG* : 0.8 *rb* : 0.45	0.8
7	$2 * 10^{4}$	1024 * 1024	ζ, ψ : *CG − rb* p : *CG*	*CG* : 0.0006 *rb* : 0.35	*CG* : 0.8 *rb* : 0.35	0.8
8	$3 * 10^{4}$	1024 * 1024	ζ, ψ : *CG − rb* p : *CG*	*CG* : 0.0002 *rb* : 0.25	*CG* : 0.8 *rb* : 0.25	0.8
9	$5 * 10^{4}$	1024 * 1024	ζ, ψ : *CG − rb* p : *CG*	*CG* : 0.0002 *rb* : 0.10	*CG* : 0.8 *rb* : 0.10	0.8
10	$6 * 10^{4}$	1024 * 1024	ζ, ψ : *CG − rb* p : *CG*	*CG* : 0.0002 *rb* : 0.09	*CG* : 0.8 *rb* : 0.09	0.8

note: CG-rb indicates that we used the CG method until development of the boundary layer and that we used the red-black method for computation after development of the boundary layer.

Table 2 Values of minimum, maximum and each contour for ζ, ψ and p in Figures 2-10

Fig.	*stream − function* ψ		*vorticity* ζ		*total pressure p*				
							contours		
	minimum	*maximum*	*minimum*	*maximum*	*minimum*	*maximum*	*initial*	*terminal*	*increment*
2	$-1.00 * 10^{-1}$	$2.00 * 10^{-6}$	$-3.69 * 10^{2}$	$1.41 * 10^{2}$	$-5.12 * 10^{6}$	$5.12 * 10^{6}$	$-5.1 * 10^{6}$	$5.1 * 10^{6}$	$8.0 * 10^{5}$
3	$-1.03 * 10^{-1}$	$1.30 * 10^{-3}$	$-3.75 * 10^{2}$	$1.47 * 10^{2}$	−4.700	6.292	−4.70	6.30	0.015
4	$-1.18 * 10^{-1}$	$1.72 * 10^{-3}$	$-4.15 * 10^{2}$	$2.23 * 10^{2}$	−0.361	1.814	−0.20	0.40	0.02
5	$-1.19 * 10^{-1}$	$2.45 * 10^{-3}$	$-4.42 * 10^{2}$	$2.90 * 10^{2}$	−0.172	1.567	−0.16	0.40	0.02
6	$-1.22 * 10^{-1}$	$3.17 * 10^{-3}$	$-1.80 * 10^{3}$	$1.26 * 10^{3}$	−0.142	1.655	−0.12	0.40	0.02
7	$-1.21 * 10^{-1}$	$3.70 * 10^{-3}$	$-1.89 * 10^{3}$	$1.65 * 10^{3}$	−0.088	1.540	−0.06	0.40	0.02
8	$-1.20 * 10^{-1}$	$5.30 * 10^{-3}$	$-1.93 * 10^{3}$	$1.91 * 10^{3}$	−0.097	1.564	−0.08	0.40	0.02
9	$-1.18 * 10^{-1}$	$4.90 * 10^{-3}$	$-1.97 * 10^{3}$	$2.42 * 10^{3}$	−0.086	1.610	−0.06	0.40	0.02
10	$-1.17 * 10^{-1}$	$5.25 * 10^{-3}$	$-1.98 * 10^{3}$	$2.64 * 10^{3}$	−0.089	1.654	−0.06	0.40	0.02

note1: The value of each contour for ψ in Figures 2-10 is as follows:
initial = −60.0, terminal = 60.0, increment = 1.0.

note2: The value of each contour for ζ in Figures 2-10 is as follows:

eddy	*comment*	*initial*	*terminal*	*increment*
primary	*main*	$-1.5 * 10^{-1}$	0.0	$1.0 * 10^{-2}$
	additional	$-1.5 * 10^{-4}$	0.0	$3.0 * 10^{-5}$
secondary	*left*	0.0	$1.0 * 10^{-4}$	$1.0 * 10^{-5}$
	right	0.0	$5.2 * 10^{-3}$	$4.0 * 10^{-4}$
tertiary	*main*	$-1.5 * 10^{-4}$	0.0	$3.0 * 10^{-5}$
	additional	$-8.0 * 10^{-4}$	0.0	$4.0 * 10^{-4}$

Table 3 Computational conditions used to compare an effect on the number of the processing elements and the solution methods

solution method	*Re*	*grid*	*relaxation parameters, etc.*		
			$\omega_\zeta ; \alpha_\zeta$	$\omega_\psi ; \alpha_\psi$	$\omega_p ; \alpha_p$
Jacobi(J1)	10^{3}	256 * 256	0.9; −	0.9; −	1.0; −
red − black	10^{3}	256 * 256	1.0; −	1.0; −	1.5; −
CG	10^{3}	256 * 256	0.8; −	0.8; −	0.8; −
ADI	10^{3}	256 * 256	1.4; 5.0	1.6; 5.0	1.4; 2.0

Table 4 Computational conditions used to compare an effect on the grid size for the red-black method at *Re* = 10^3, and computational results

computational conditions				*computational results*				
	relaxation parameters				*execution time(sec)*		*center of primary vortex*	
grid	ω_ζ	ω_ψ	ω_p	*pe*	ζ *and* ψ	*p*	*x*	*y*
$256 * 256$	1.00	1.00	1.5	8	223	22.8	0.5294	0.5647
$1k * 1k$	1.60	1.60	1.5	8	2,458	278.3	0.5308	0.5650
$2k * 2k$	1.75	1.75	1.6	16	8,681	694.7	0.5305	0.5647
$4k * 4k$	1.86	1.86	1.6	64	17,105	1,015.7	0.5304	0.5643
$8k * 8k$	1.90	1.90	1.8	64	166,595	4,234.8	0.5285	0.5648
$16k * 16k$	1.93	1.93	1.8	128	773,548	8,608.0	0.5291	0.5645

note: $1k = 1024$.

Table 5 Characteristic parameters of square cavity programs on the NWT computer system

(1) Jacobi method(J1)

	grid size			
	$256 * 256$		$1024 * 1024$	
parameter	*v*1	*v*2	*v*1	*v*2
r_∞	$1.034730 * 10^3$	$4.116968 * 10^3$	$1.288553 * 10^4$	$4.071686 * 10^4$
$pe_{1/2}$	$1.256238 * 10^0$	$5.773551 * 10^0$	$1.687149 * 10^1$	$5.094211 * 10^1$
$\bar{pe}$	$2.751874 * 10^1$	$2.341240 * 10^1$	$8.657361 * 10^1$	$1.112798 * 10^2$
n	$2.941345 * 10^0$	$2.049465 * 10^0$	$2.790058 * 10^0$	$2.945650 * 10^0$

(2) red-black method

	grid size					
	$256 * 256$		$1024 * 1024$		$2048 * 2048$	
parameter	*v*1	*v*2	*v*1	*v*2	*v*1	*v*2
r_∞	$6.520323 * 10^2$	$2.372509 * 10^3$	$1.071891 * 10^4$	$3.490736 * 10^4$	$3.596278 * 10^4$	$1.107416 * 10^5$
$pe_{1/2}$	$1.423487 * 10^0$	$5.878899 * 10^0$	$2.175465 * 10^1$	$6.855245 * 10^1$	$6.982452 * 10^1$	$2.174536 * 10^2$
$\bar{pe}$	$2.351380 * 10^1$	$2.819143 * 10^1$	$1.076090 * 10^2$	$1.072268 * 10^2$	$1.723708 * 10^2$	$2.150169 * 10^2$
n	$1.950822 * 10^0$	$2.041643 * 10^0$	$1.671894 * 10^0$	$2.194413 * 10^0$	$2.340966 * 10^0$	$2.032085 * 10^0$

(3) CG method

	grid size			
	$256 * 256$		$1024 * 1024$	
parameter	*v*1	*v*2	*v*1	*v*2
r_∞	$5.841961 * 10^2$	$2.931970 * 10^3$	$1.113820 * 10^4$	$2.652857 * 10^4$
$pe_{1/2}$	$1.335510 * 10^0$	$5.133247 * 10^0$	$2.956315 * 10^1$	$5.820522 * 10^1$
$\bar{pe}$	$2.704300 * 10^1$	$1.879162 * 10^1$	$1.165913 * 10^2$	$7.816255 * 10^1$
n	$2.795999 * 10^0$	$2.311582 * 10^0$	$3.788892 * 10^0$	$2.692125 * 10^0$

(4) ADI method

	grid size $= 255 * 255$	
parameter	*v*1	*v*2
r_∞	$1.039469 * 10^2$	$1.059596 * 10^2$
$pe_{1/2}$	$4.705021 * 10^{-1}$	$3.370313 * 10^{-1}$
$\bar{pe}$	$6.416783 * 10^0$	$9.729777 * 10^0$
n	$1.824020 * 10^0$	$2.061557 * 10^0$

Table 6 Measurements of maximum actual performance of the square cavity programs on the NWT computer system (note: The values for *pe* = 1 are ones for SIMD computing)

(1) Jacobi method (J1)

pe	$v1/2$	$\tau_1(Mflop/s)$	$\tau_2(Mflop/s)$	$\tau_3(Mflop/s)$	$\tau_4(Mflop/s)$	$\tau_5(Mflop/s)$	d_{ma}
1	$v1$	896.34	780.48	830.94	– – –	– – –	– –
	$v2$	893.57	809.20	904.00	1439.81	151.13	1.593
2	$v1$	1771.97	1572.66	1592.52	– – –	– – –	– –
	$v2$	1813.18	1614.22	1818.99	2242.01	301.95	1.233
4	$v1$	3232.06	2871.19	3153.80	– – –	– – –	– –
	$v2$	3315.69	2975.14	3324.60	4338.44	602.88	1.305
8	$v1$	6635.02	5963.08	6836.21	– – –	– – –	– –
	$v2$	6948.61	6166.67	6979.90	8867.02	1203.61	1.270
16	$v1$	13145.14	11691.82	13149.84	– – –	– – –	– –
	$v2$	13726.69	12167.42	13720.87	17670.03	2402.97	1.288
32	$v1$	26729.56	23817.70	26838.66	– – –	– – –	– –
	$v2$	27385.34	24630.76	27201.76	35607.49	4791.39	1.309
64	$v1$	51103.70	45733.47	52934.53	– – –	– – –	– –
	$v2$	53630.43	47769.35	53689.25	70617.29	9555.78	1.315
128	$v1$	104003.26	91248.90	100884.89	133872.70	– – –	1.327
	$v2$	107277.71	94212.27	106333.47	142364.71	19072.32	1.340

(2) Jacobi method (J2)

pe	$v1/2$	$\tau_1(Mflop/s)$	$\tau_2(Mflop/s)$	$\tau_3(Mflop/s)$	$\tau_4(Mflop/s)$	$\tau_5(Mflop/s)$	d_{ma}
1	$v2$	1125.67	891.02	1123.79	1257.17	326.71	1.119
2	$v2$	2288.16	1788.17	2266.42	2526.72	651.63	1.115
4	$v2$	4432.07	3408.93	4307.02	5001.47	1298.71	1.161
8	$v2$	9047.68	7050.29	8734.22	10119.67	2594.06	1.159
16	$v2$	17987.69	12474.39	17508.94	20136.27	5173.42	1.150
32	$v2$	36099.57	27893.81	35153.31	40406.65	10299.00	1.149
64	$v2$	72443.40	54959.25	68064.06	80396.20	20585.85	1.181
128	$v2$	145065.44	108455.77	134983.03	157598.09	40829.21	1.168

(3) Red-black ordering

pe	$v1/2$	$\tau_1(Mflop/s)$	$\tau_2(Mflop/s)$	$\tau_3(Mflop/s)$	$\tau_4(Mflop/s)$	$\tau_5(Mflop/s)$	d_{ma}
1	$v1$	602.23	503.22	576.03	– – –	– – –	– –
	$v2$	507.32	508.22	581.88	617.45	41.32	1.061
2	$v1$	1205.67	999.34	1149.04	– – –	– – –	– –
	$v2$	1208.04	1013.47	1162.15	1243.81	82.49	1.070
4	$v1$	2390.78	1962.36	2263.02	– – –	– – –	– –
	$v2$	2387.37	1979.46	2279.09	2472.13	164.87	1.085
8	$v1$	4786.64	3990.67	4554.39	– – –	– – –	– –
	$v2$	4793.20	3989.34	4571.29	4953.45	329.49	1.084
16	$v1$	9479.17	7863.72	9050.39	– – –	– – –	– –
	$v2$	9421.13	7840.44	8967.29	9911.65	658.48	1.105
32	$v1$	19010.27	15747.52	17844.49	– – –	– – –	– –
	$v2$	18727.89	15625.85	17869.22	19801.27	1307.84	1.108
64	$v1$	37518.93	30776.77	35333.05	– – –	– – –	– –
	$v2$	36920.69	30596.73	34799.87	39311.40	2630.01	1.130
128	$v1$	74108.23	59702.51	68853.69	76721.15	– – –	1.114
	$v2$	72604.31	59932.71	68144.19	79179.64	5234.37	1.162

Table 6 Continued

(4) CG method

pe	$v1/2$	$\tau_1(Mflop/s)$	$\tau_2(Mflop/s)$	$\tau_3(Mflop/s)$	$\tau_4(Mflop/s)$	$\tau_5(Mflop/s)$	d_{ma}
1	$v1$	913.22	823.35	885.91	− − −	− − −	− −
	$v2$	918.92	824.03	908.78	969.86	150.05	1.067
2	$v1$	1643.34	1510.85	1683.49	− − −	− − −	− −
	$v2$	1732.86	1567.75	1744.92	1841.04	297.44	1.055
4	$v1$	3437.48	3100.03	3435.19	− − −	− − −	− −
	$v2$	3556.33	3187.61	3563.72	3739.03	592.99	1.049
8	$v1$	6388.46	5722.53	6361.37	− − −	− − −	− −
	$v2$	6689.05	6041.97	6645.19	7051.08	1183.99	1.061
16	$v1$	13193.54	11825.71	13114.86	− − −	− − −	− −
	$v2$	13827.01	12415.07	13751.11	14547.61	2360.57	1.058
32	$v1$	25836.20	23132.94	25571.54	− − −	− − −	− −
	$v2$	27065.42	24219.60	26804.95	28511.95	4701.43	1.064
64	$v1$	51665.29	46009.02	49594.98	− − −	− − −	− −
	$v2$	53728.95	48068.44	53500.98	56370.13	9352.25	1.054
128	$v1$	100915.97	88901.34	100005.14	105547.36	− − −	1.055
	$v2$	103640.27	92122.34	103110.37	110265.77	18591.26	1.069

(5) ADI method

pe	$v1/2$	$\tau_1(Mflop/s)$	$\tau_2(Mflop/s)$	$\tau_3(Mflop/s)$	$\tau_4(Mflop/s)$	$\tau_5(Mflop/s)$	d_{ma}
1	$v1$	642.44	612.00	649.47	− − −	− − −	− −
	$v2$	643.39	612.92	651.19	702.02	260.47	1.078
2	$v1$	325.71	316.23	299.06	− − −	− − −	− −
	$v2$	344.68	344.09	351.03	423.63	243.69	1.207
4	$v1$	514.46	515.93	469.31	− − −	− − −	− −
	$v2$	520.21	518.79	539.02	654.02	376.36	1.213
8	$v1$	737.19	754.89	689.88	− − −	− − −	− −
	$v2$	802.08	791.01	792.26	906.72	552.00	1.144
16	$v1$	1189.79	1198.45	1131.48	− − −	− − −	− −
	$v2$	1251.69	1236.71	1294.24	1516.69	758.25	1.172
32	$v1$	1591.71	1618.20	1480.48	− − −	− − −	− −
	$v2$	1769.16	1781.88	1722.00	1893.16	972.67	1.099
64	$v1$	2483.39	2567.46	2390.19	− − −	− − −	− −
	$v2$	2645.45	2687.21	2798.39	3034.59	1264.77	1.084
128	$v1$	3323.38	3391.06	3114.13	3253.06	− − −	1.045
	$v2$	3178.33	3202.02	3671.87	3930.80	1599.60	1.071

Table 7 Grid size used to obtain Table 6 and Figure 23

	grid size		
pe	$Jacobi(J1/J2)$ $red-black$	CG	ADI
1	$1,500 * 1,500$	$1,024 * 1,024$	$1,025 * 1,025$
2	$2,048 * 2,048$	$1,500 * 1,500$	$1,501 * 1,501$
4	$3,000 * 3,000$	$2,048 * 2,048$	$2,049 * 2,049$
8	$4,096 * 4,096$	$3,000 * 3,000$	$3,001 * 3,001$
16	$5,800 * 5,800$	$4,096 * 4,096$	$4,097 * 4,097$
32	$8,192 * 8,192$	$5,800 * 5,800$	$5,801 * 5,801$
64	$12,000 * 12,000$	$8,192 * 8,192$	$8,193 * 8,193$
128	$16,384 * 16,384$	$12,000 * 12,000$	$12,001 * 12,001$

Table 8 Comparison between the values of S_{pe} computed from (63) and (69) when the grid size is 1,024∗1,024.

pe	$Jacobi(J1)$		$red-black$		CG	
	(63)	(69)	(63)	(69)	(63)	(69)
2	1.98	1.95	2.00	1.95	1.69	1.65
4	3.57	3.77	3.89	3.80	2.83	2.75
8	7.17	7.05	7.38	7.18	5.02	4.86
16	12.17	12.72	13.28	12.84	8.93	8.54
32	19.68	20.30	21.66	20.68	14.68	13.98
64	27.38	27.57	30.01	27.85	19.07	17.59
128	30.10	28.68	32.23	28.28	17.21	15.19

note: The value of VTC_1 is 0.988(Jacobi(J1)), 0.988(red-black) and 0.800(CG), respectively.

Table 9 Speedup and efficiency for the case of τ_{3mm} and v2 of Table 6 provided that $VTC1$ = 1.0.

pe	$Jacobi(J1)$		$Jacobi(J2)$		$red-black$		CG		ADI	
	S_{pe}	E_{pe}	S_{pe}	E_{pe}	S_{pe}	E_{pe}	S_{pe}	E_{pe}	S_{pe}	E_{pe}
2	1.967	0.984	1.966	0.983	1.959	0.979	1.915	0.957	0.726	0.363
4	3.903	0.976	3.903	0.976	3.886	0.971	3.799	0.950	1.143	0.286
8	7.774	0.972	7.716	0.965	7.713	0.964	7.503	0.938	2.414	0.302
16	15.328	0.958	15.296	0.956	15.153	0.947	14.816	0.926	2.947	0.184
32	30.318	0.947	30.411	0.950	29.999	0.937	29.164	0.911	6.708	0.210
64	59.375	0.928	60.048	0.938	58.540	0.915	57.047	0.891	7.335	0.115
128	116.643	0.911	117.354	0.917	113.980	0.890	110.031	0.860	22.750	0.178

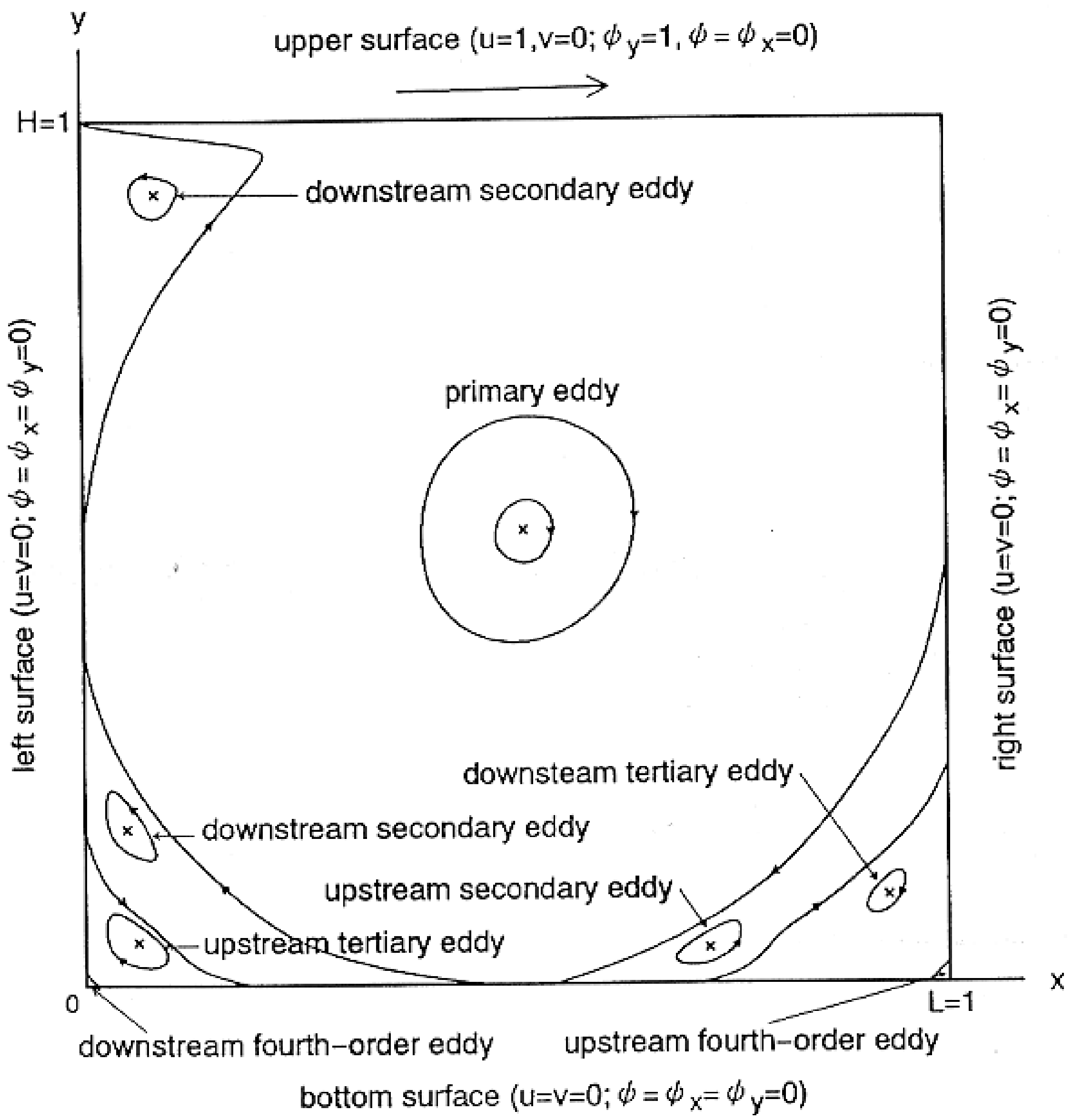
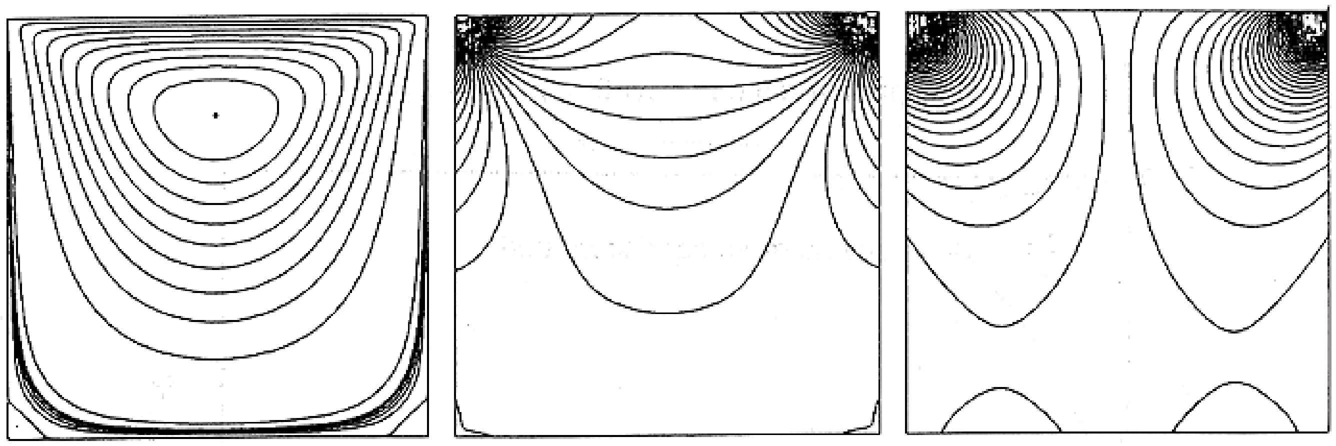
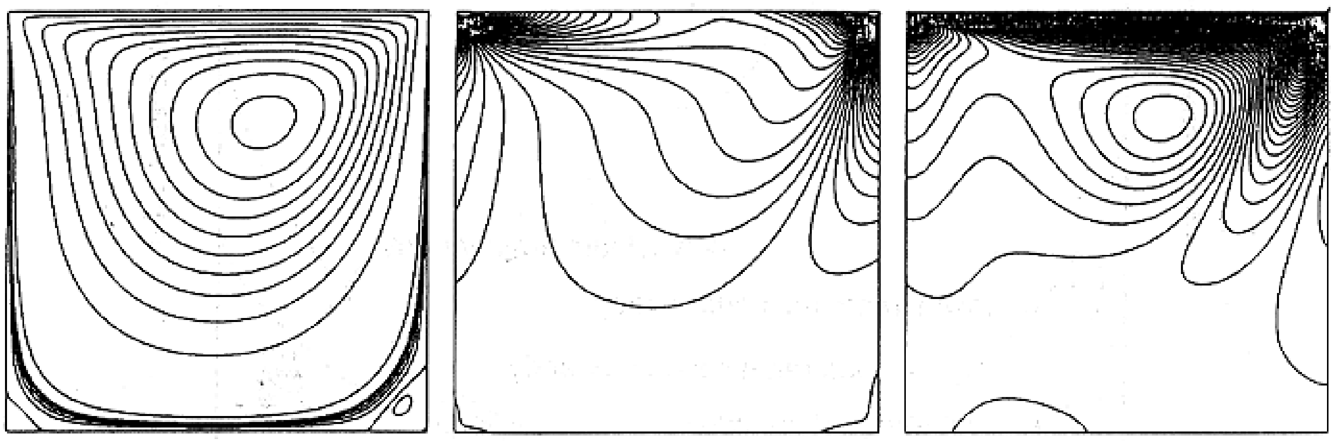

y
upper surface (u=1,v=0; ϕ_y=1, ϕ = ϕ_x=0)
H=1
downstream secondary eddy
left surface (u=v=0; ϕ = ϕ_x= ϕ_y=0)
primary eddy
right surface (u=v=0; ϕ = ϕ_x= ϕ_y=0)
downsteam tertiary eddy
downstream secondary eddy
upstream secondary eddy
upstream tertiary eddy
0
L=1
x
downstream fourth−order eddy
upstream fourth−order eddy
bottom surface (u=v=0; ϕ = ϕ_x= ϕ_y=0)

Fig. 1 Cavity flow configuration, coordinates, nomenclature and boundary conditions



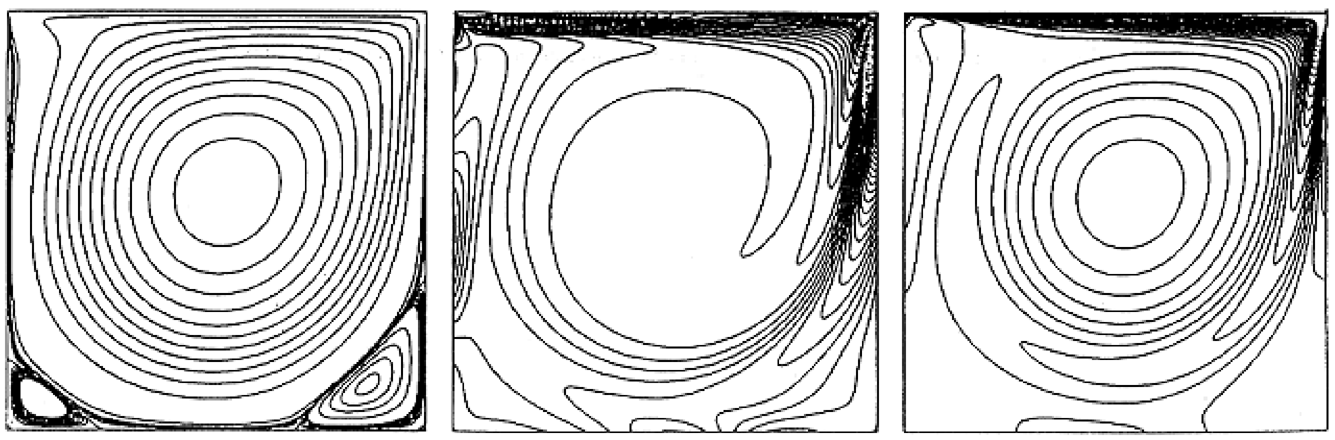
(1) Stream-function contours (2) Vorticity contours (3) Total pressure contours

Fig. 2 Square cavity flows at *Re* = 10^{-6}



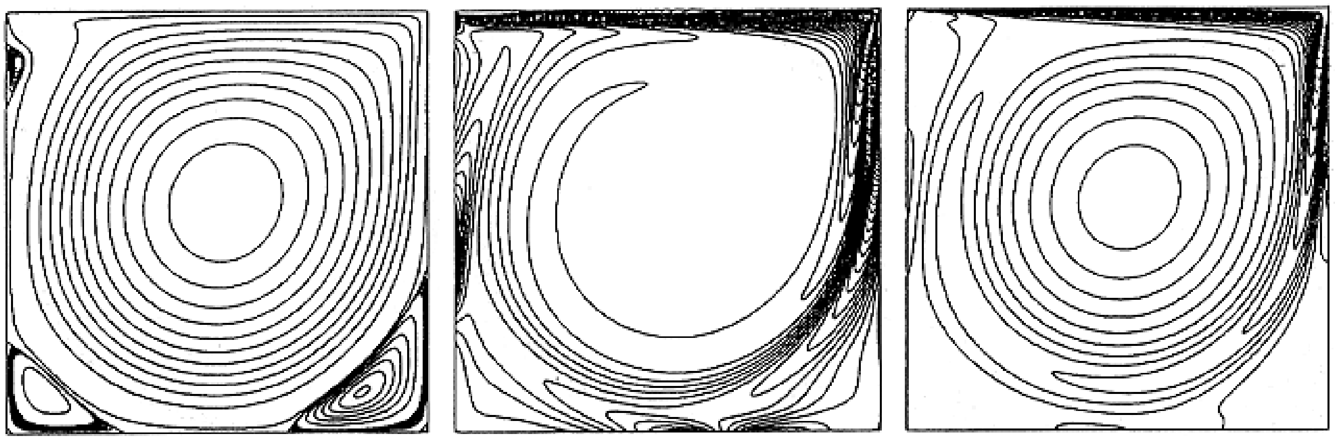
(1) Stream-function contours (2) Vorticity contours (3) Total pressure contours

Fig. 3 Square cavity flows at *Re* = 10^{2}



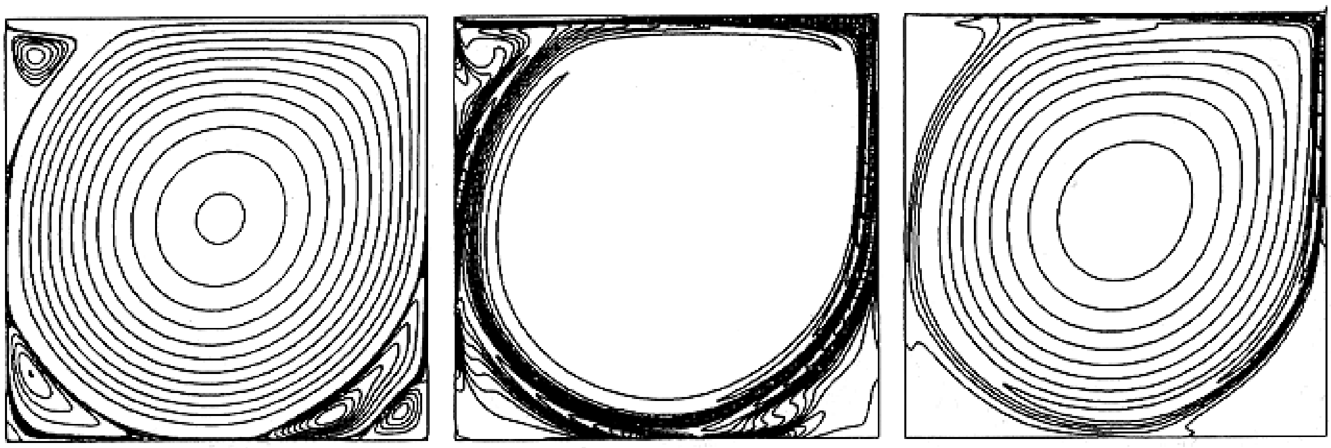
(1) Stream-function contours (2) Vorticity contours (3) Total pressure contours

Fig. 4 Square cavity flows at *Re* = 10^{3}



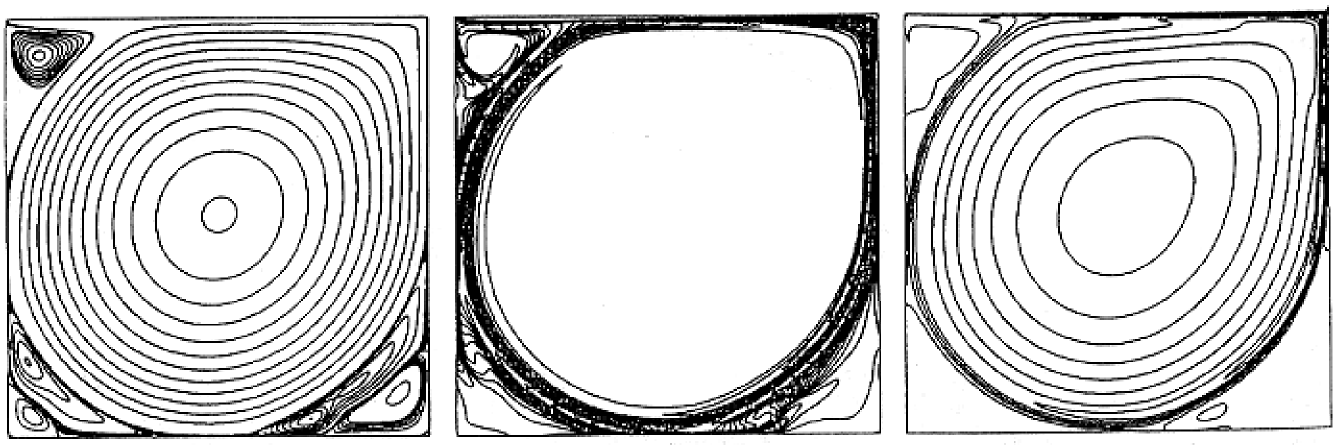
(1) Stream-function contours (2) Vorticity contours (3) Total pressure contours

Fig. 5 Square cavity flows at *Re* = $2*10^3$



(1) Stream-function contours (2) Vorticity contours (3) Total pressure contours

Fig. 6 Square cavity flows at *Re* = 10^4



(1) Stream-function contours (2) Vorticity contours (3) Total pressure contours

Fig. 7 Square cavity flows at *Re* = $2*10^4$

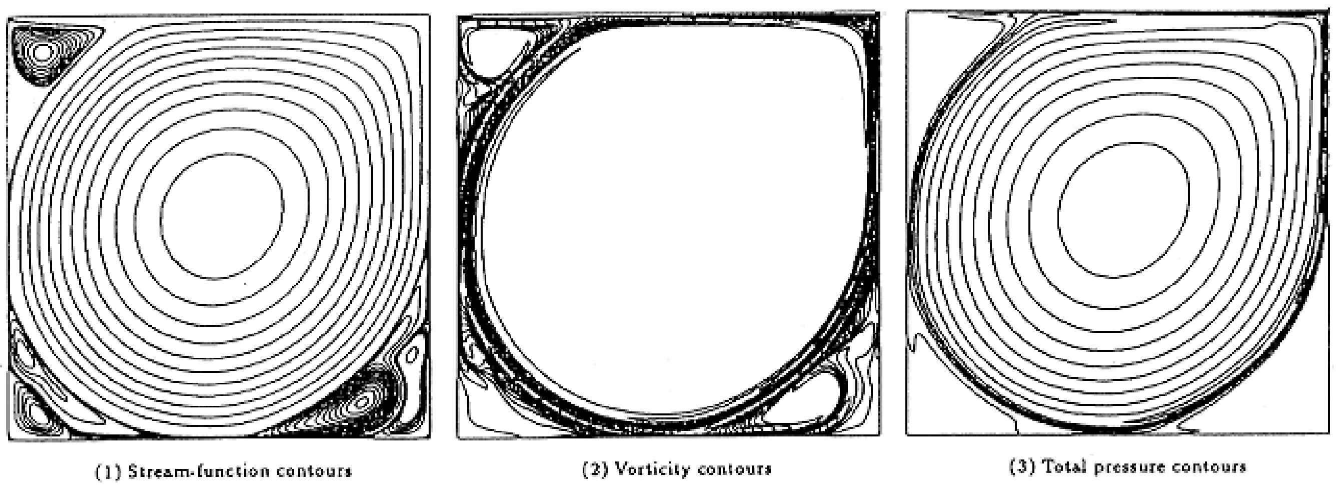


Fig. 8 Square cavity flows at *Re* = $3*10^4$ (a snapshot)

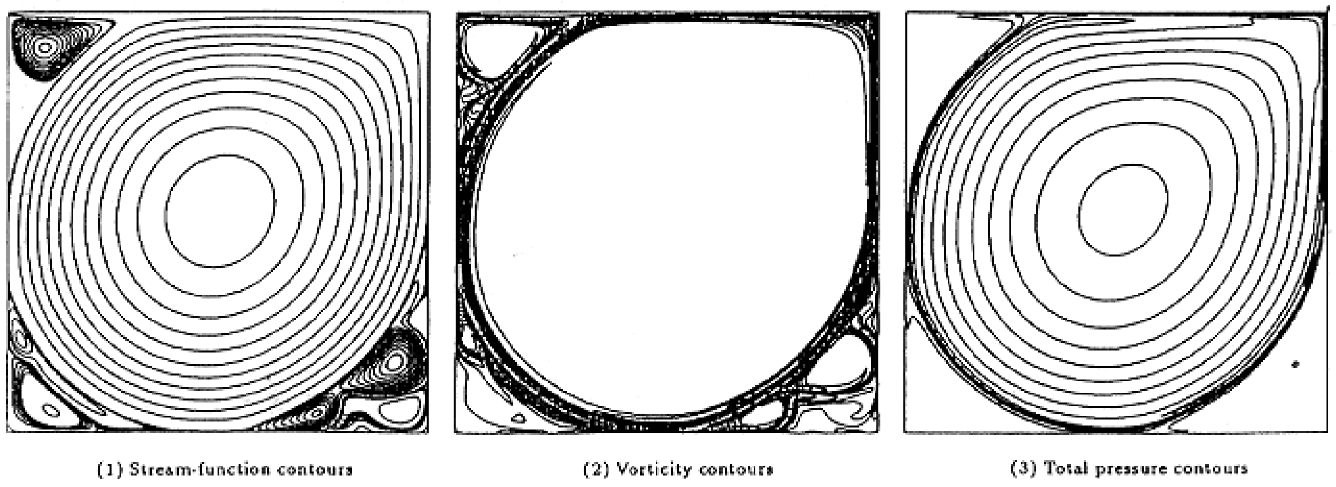


Fig. 9 Square cavity flows at *Re* = $5*10^4$ (a snapshot)

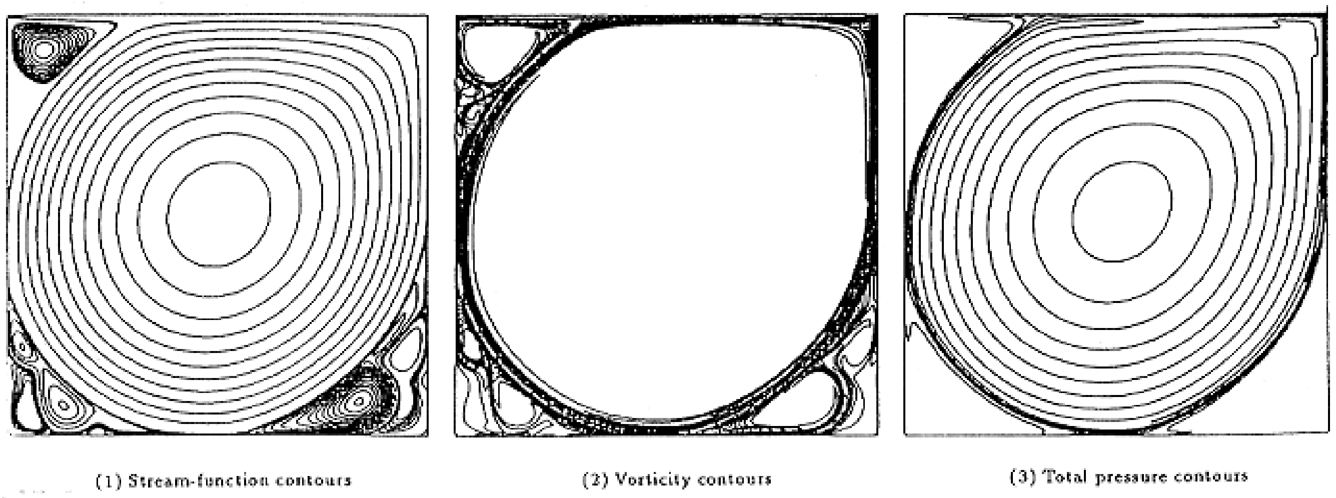


Fig. 10 Square cavity flows at *Re* = $6*10^4$ (a snapshot)

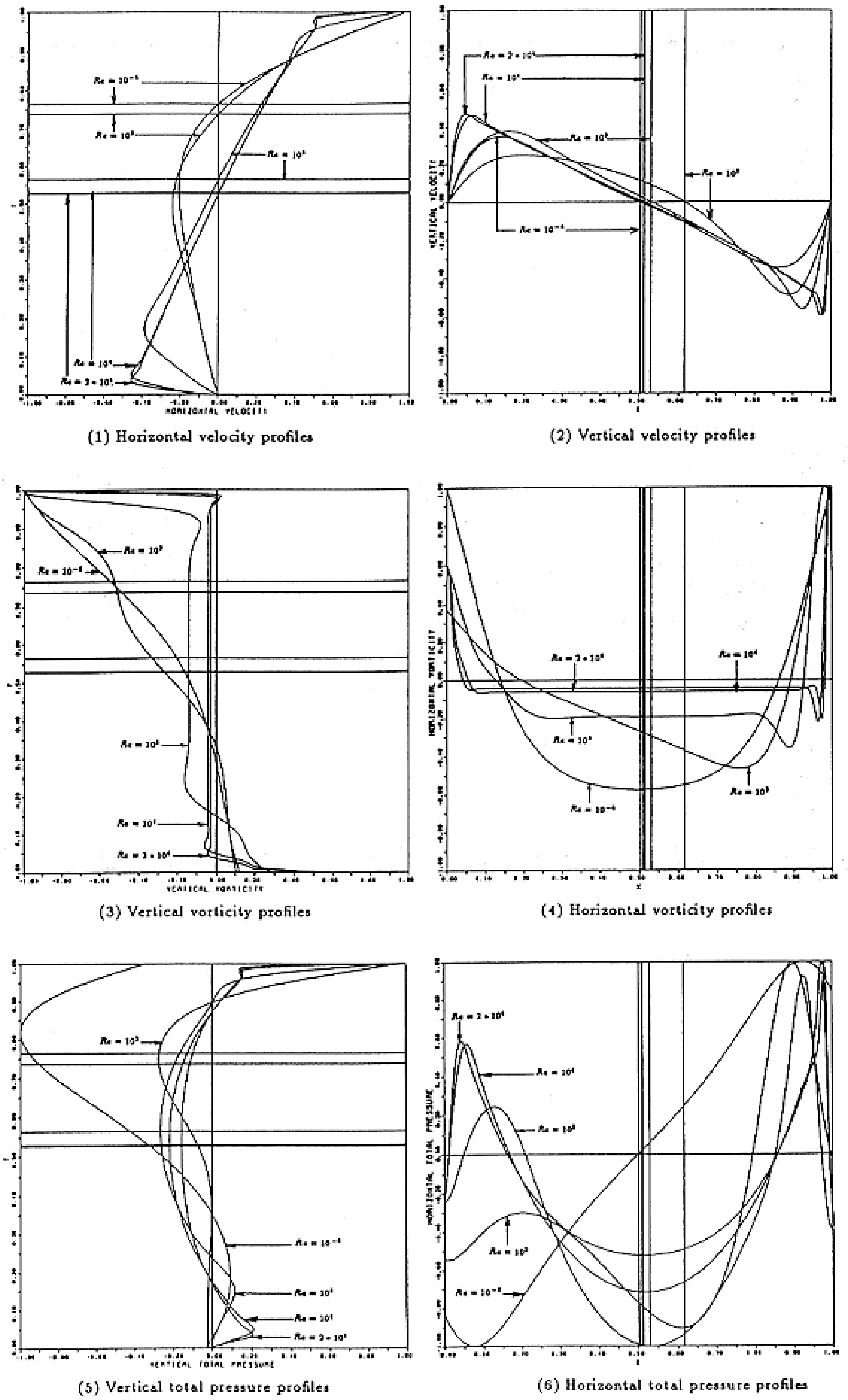


Fig. 11 Effect of Re on profiles of u, υ, ζ and p through center of primary vortex (note: The vertical and horizontal beelines indicate the values of x and y of the center of the primary vortex for each *Re*)

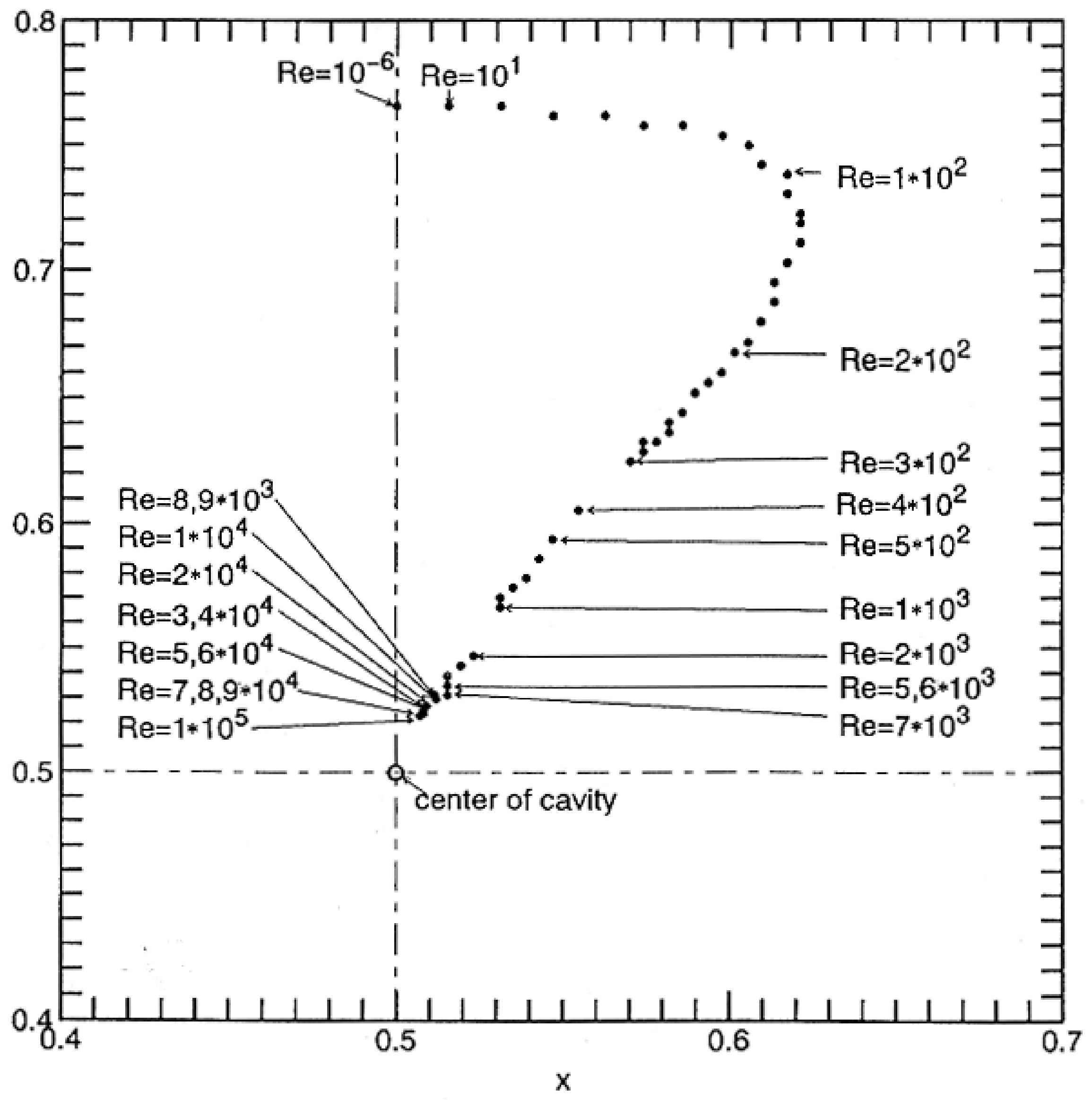
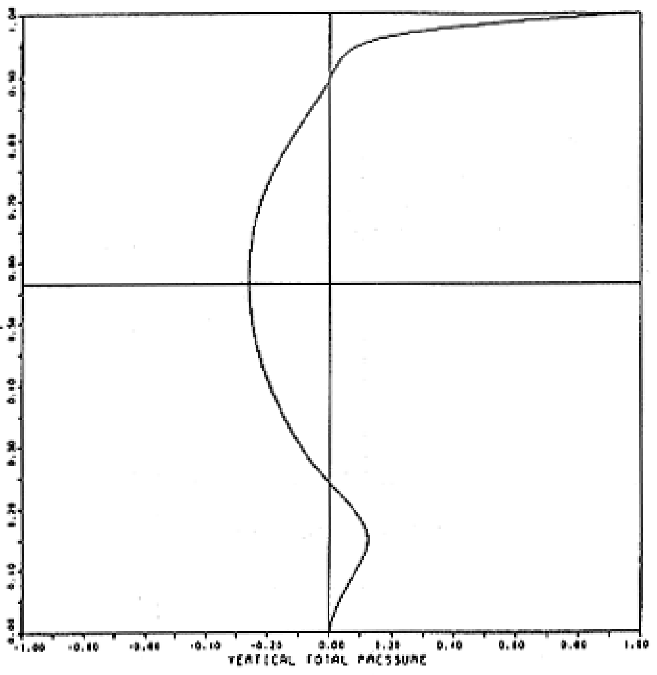
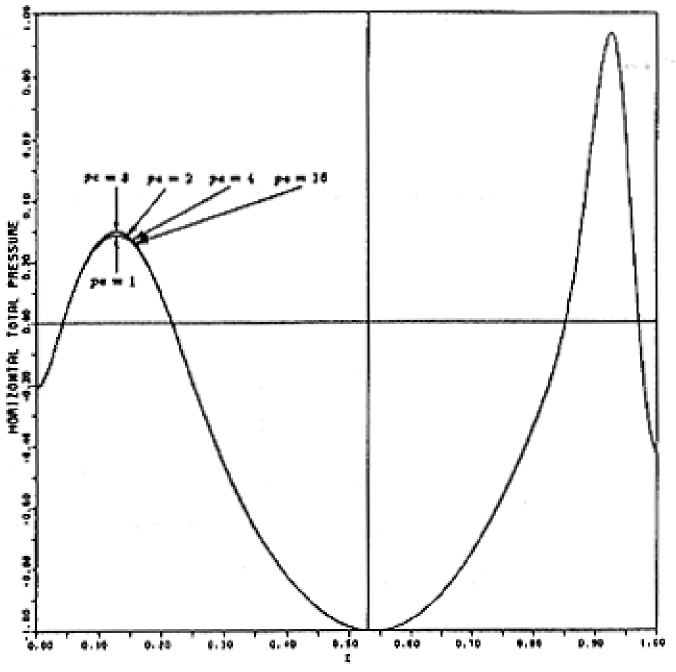


Fig. 12 Location of center of primary vortex as a function of Reynolds number

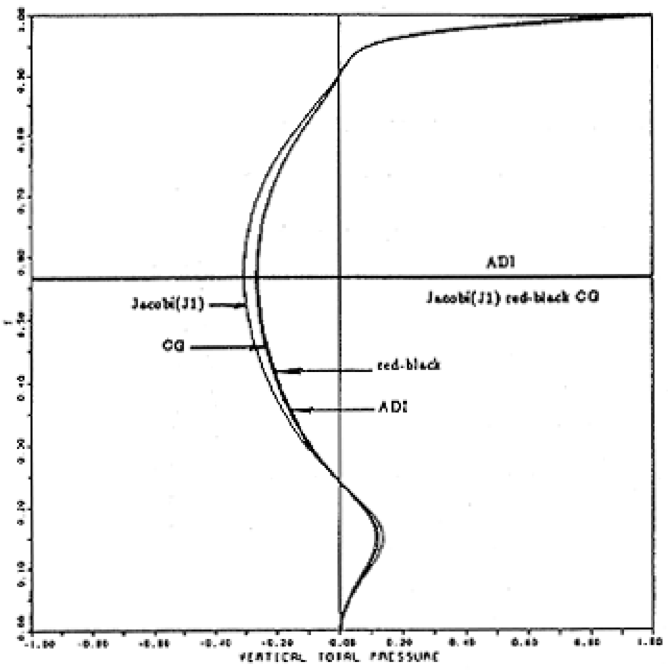


(1) Vertical total pressure profiles

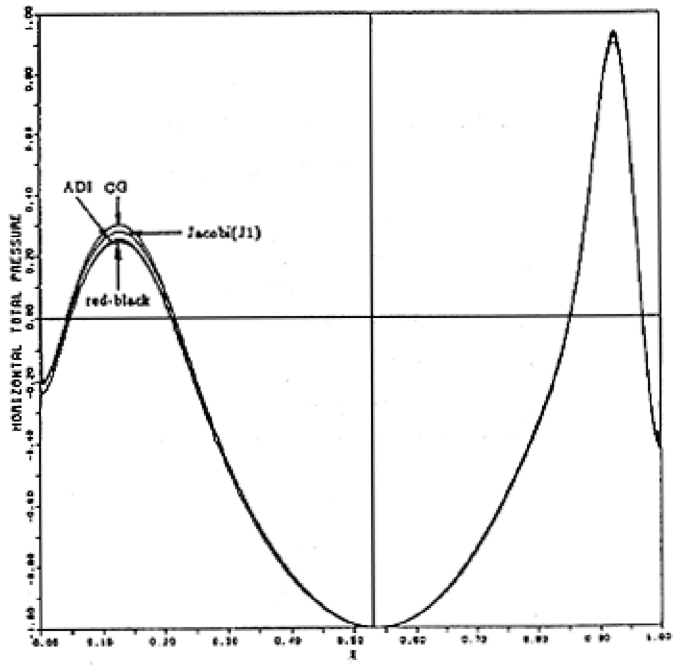


(2) Horizontal total pressure profiles

Fig. 13 Effect of *pe* on profiles of *p* through center of primary vortex in case of *Re* = 10^3 and the CG method

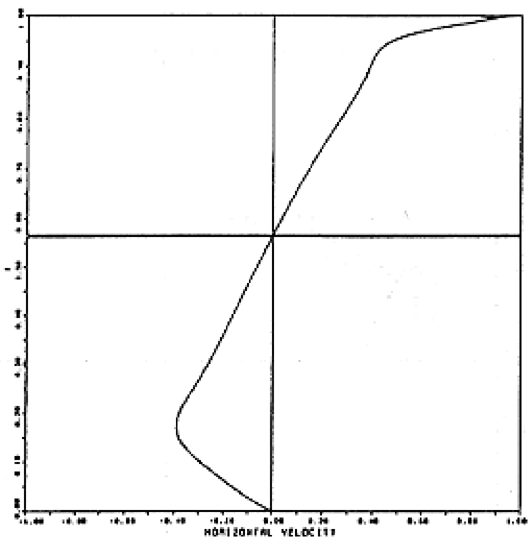


(1) Vertical total pressure profiles

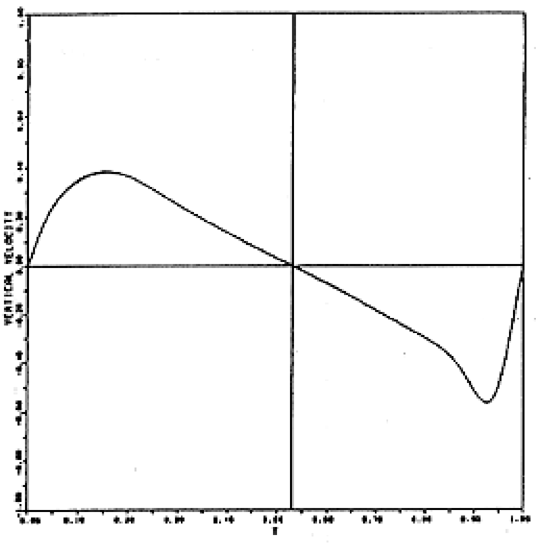


(2) Horizontal total pressure profiles

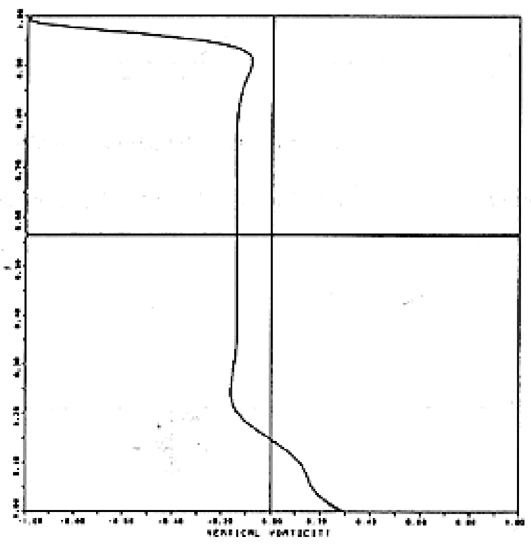
Fig. 14 Effect of the solution method on profiles of *p* through center of primary vortex in case of *Re* = 10^3 and *pe* = 16



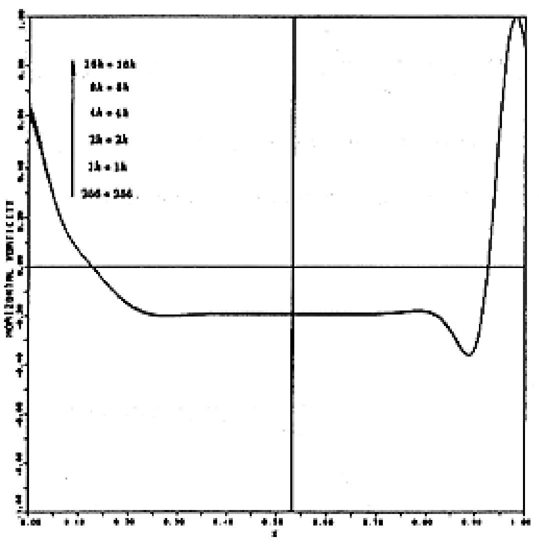
(1) Horizontal velocity profiles



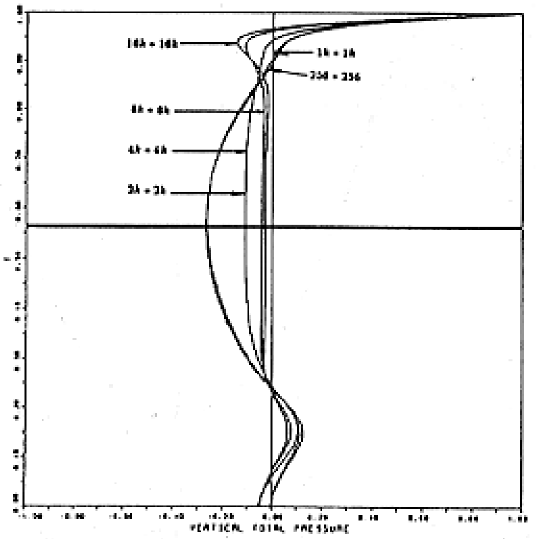
(2) Vertical velocity profiles



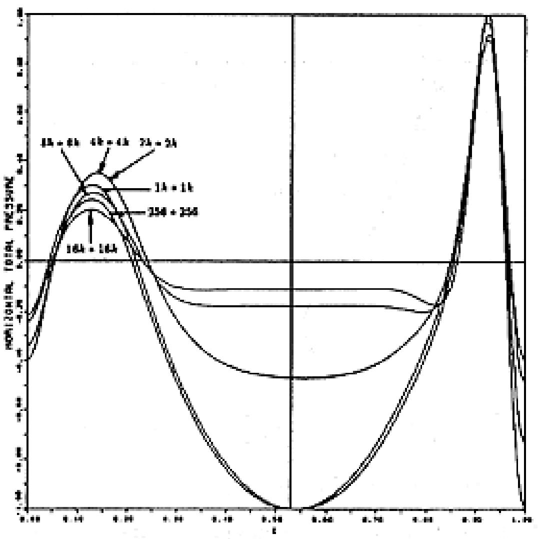
(3) Vertical vorticity profiles



(4) Horizontal vorticity profiles



(5) Vertical total pressure profiles



(6) Horizontal total pressure profiles

Fig. 15 Effect of grid size on profile of *u*, *v*, ζ and *p* through center of primary vortex in case of $Re = 10^3$ and the red-black method

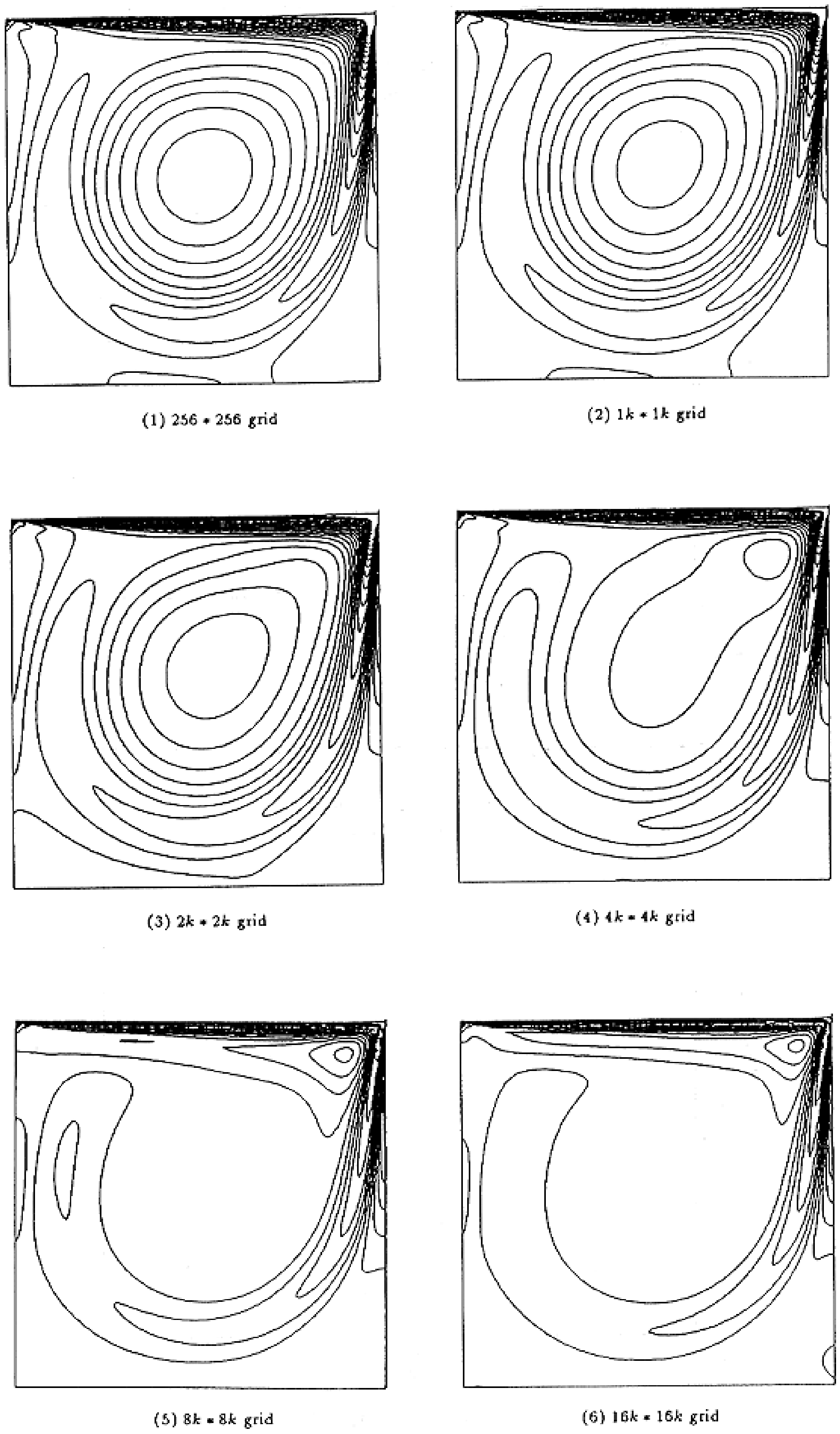
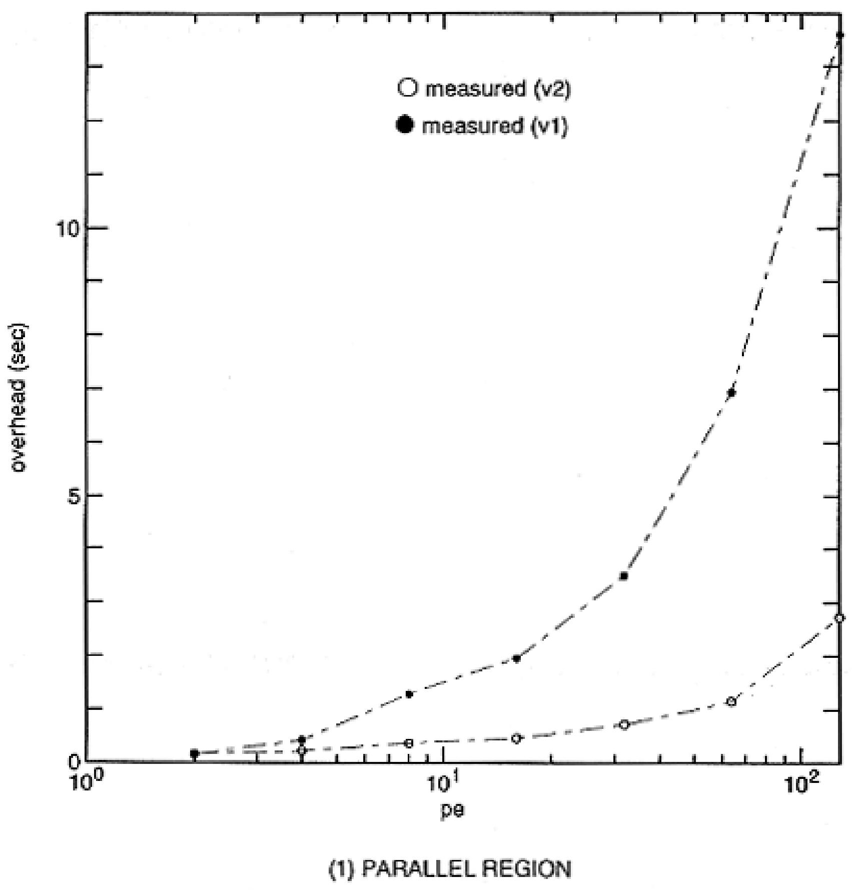


Fig. 16 Effect of the grid size on total pressure contours in case of $Re = 10^3$ and the red-black method ($1k = 1024$)



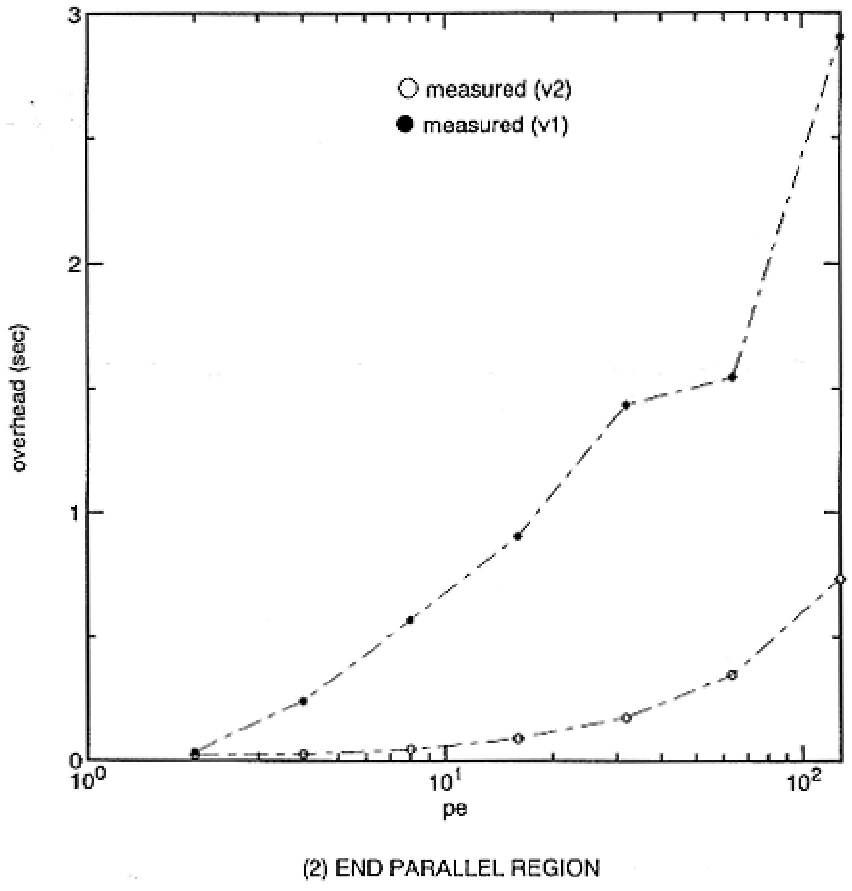
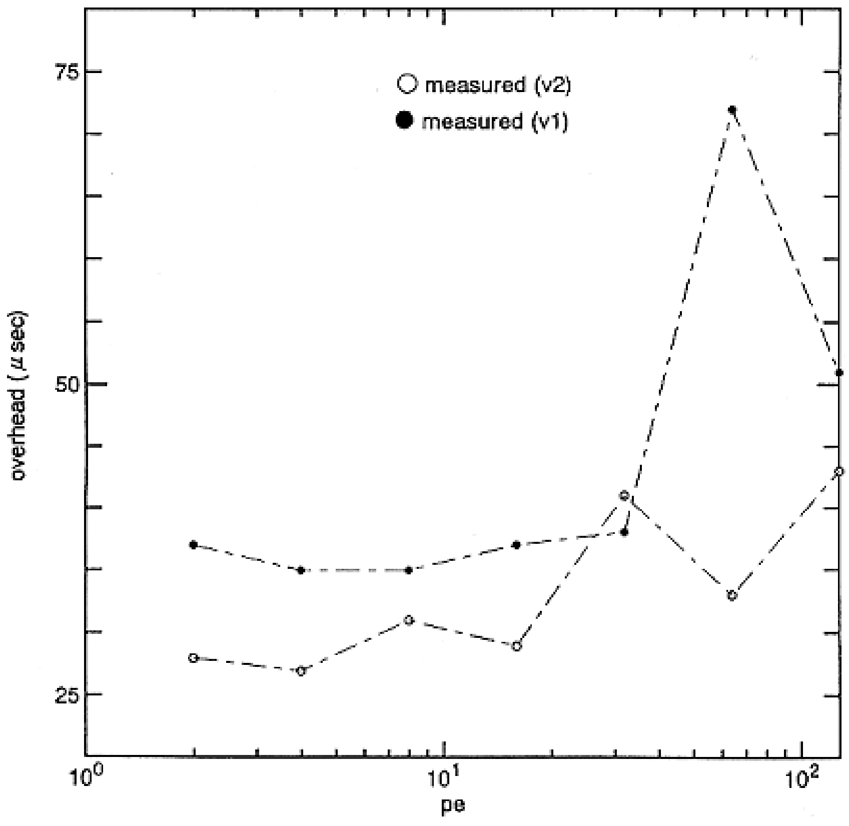
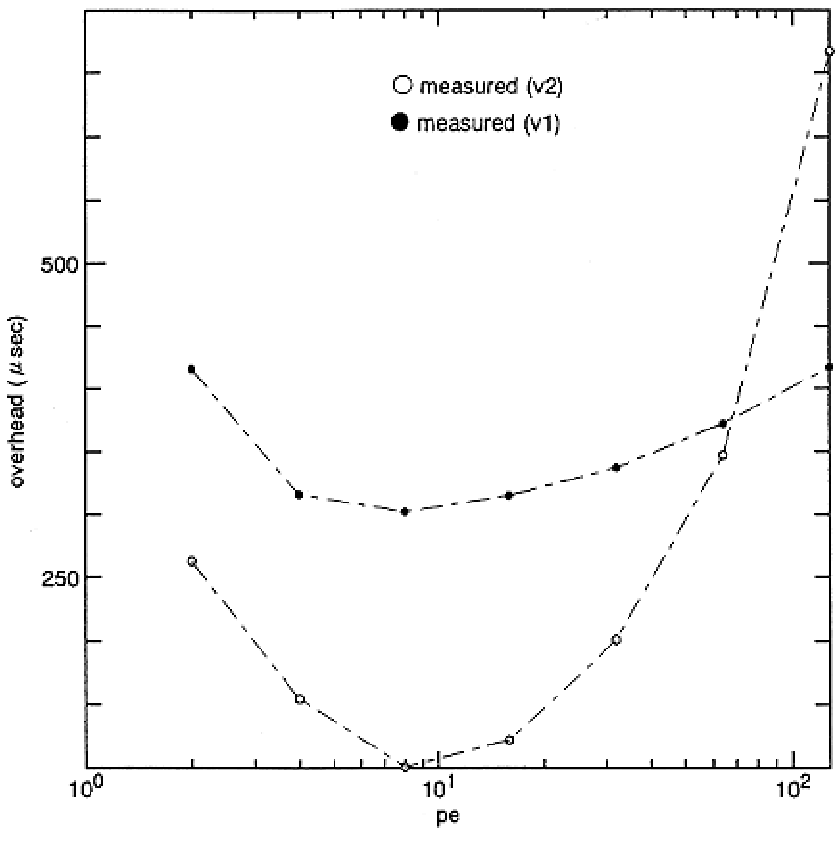


Fig. 17 Overheads for the NWT-Fortran-compiler directives against the number of processing elements

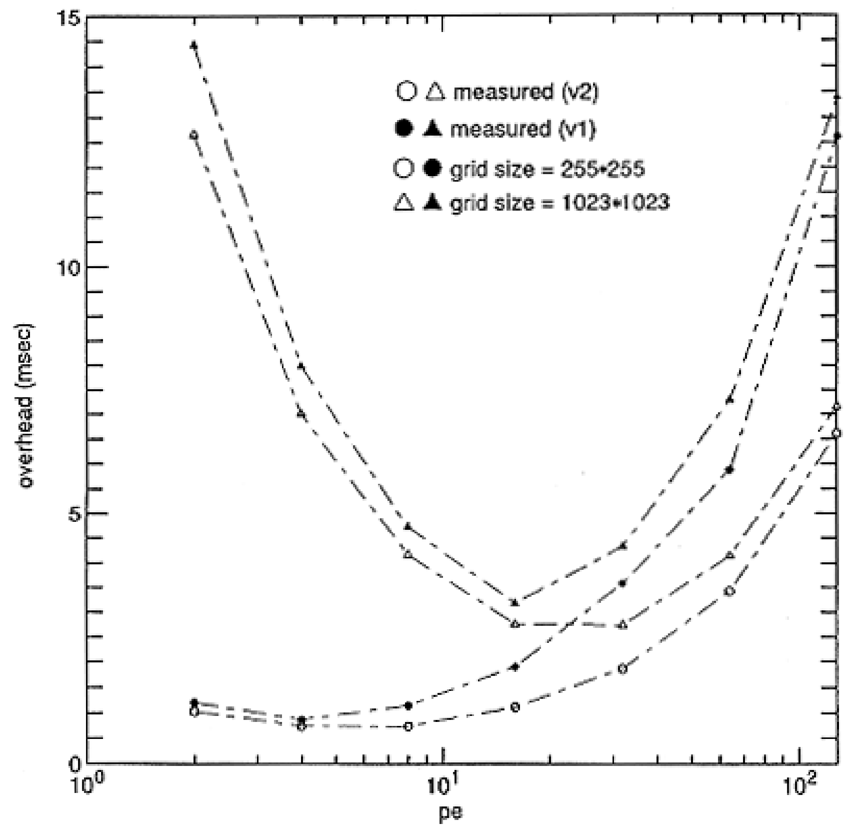


(3) SPREAD DO plus END SPREAD
(note : This result is not very precise due to the indirect measurement.)

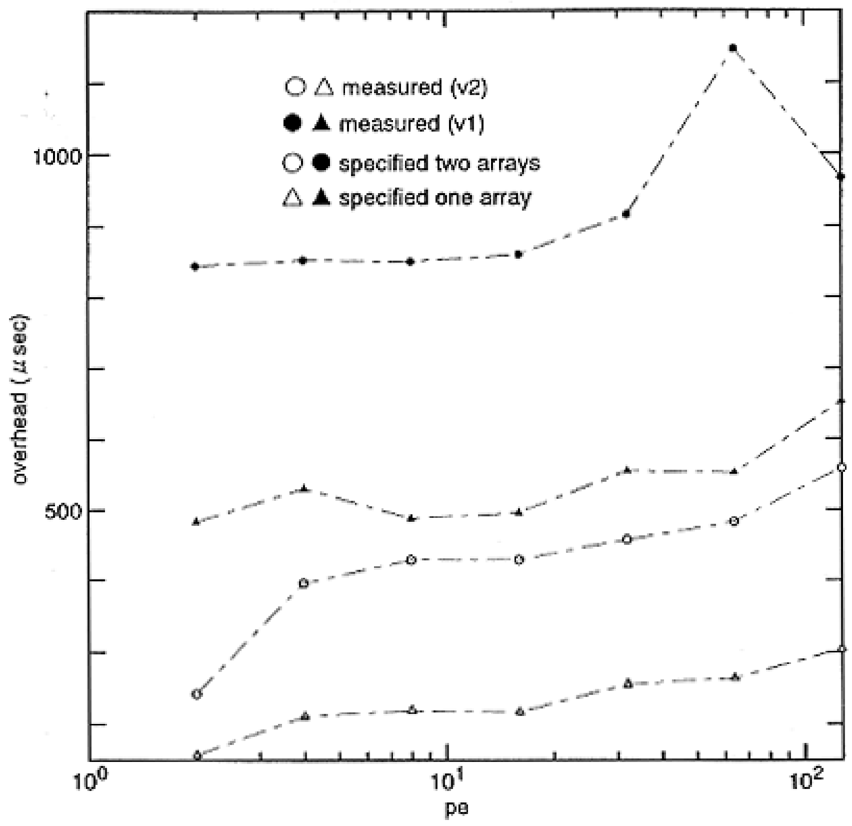


(4) SPREAD DO plus END SPREAD SUM(abc)
(note : This result is not very precise due to the indirect measurement.)

Fig. 17 Continued

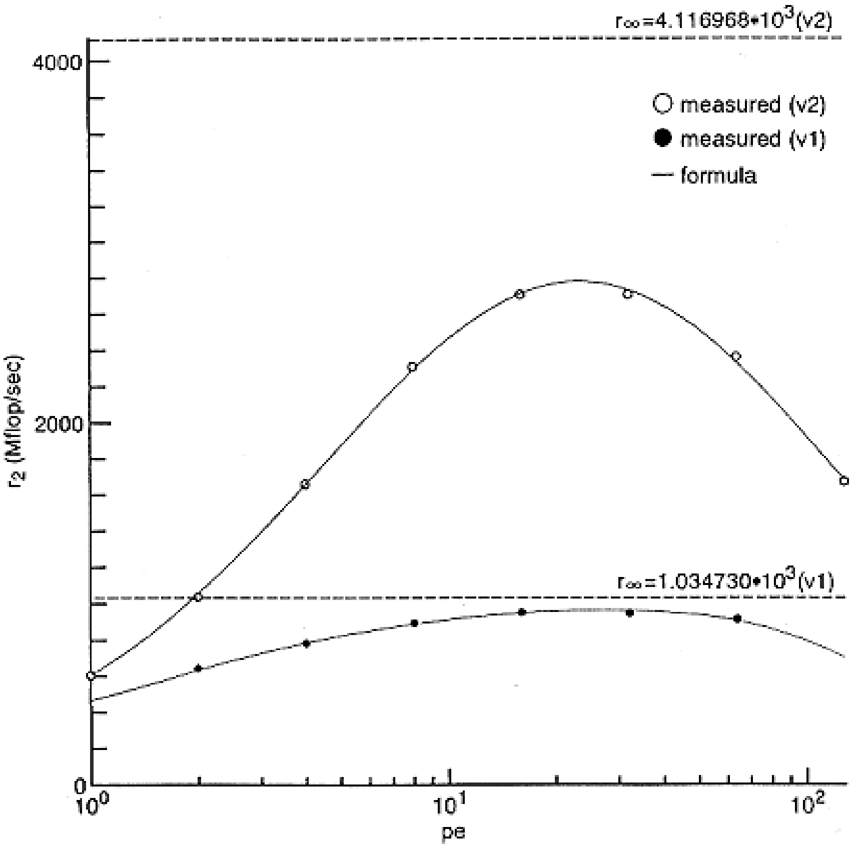


(5) SPREAD MOVE plus END SPREAD plus MOVEWAIT (note : This result indicates the mean values of six data obtained when an array data of the dimension shown above is transfered between the global and local memory spaces.)

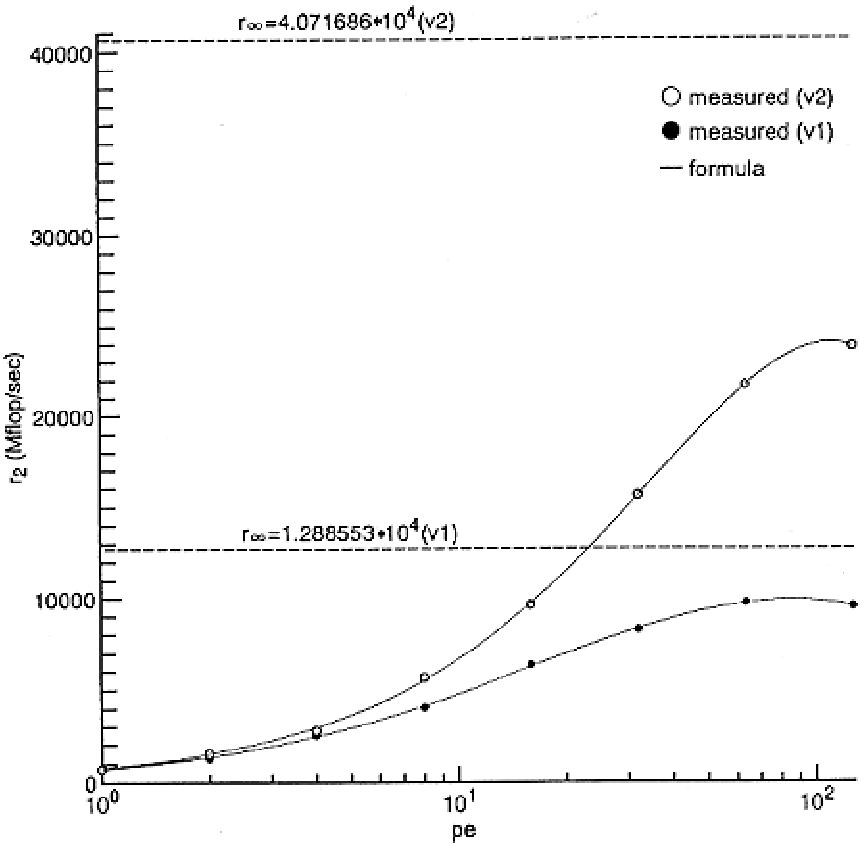


(6) OVERLAPFIX plus MOVEWAIT (note : This indicates the results when one or two arrays of the overlap range = (1,1) are specified in the OVERLAPFIX statement.)

Fig. 17 Continued

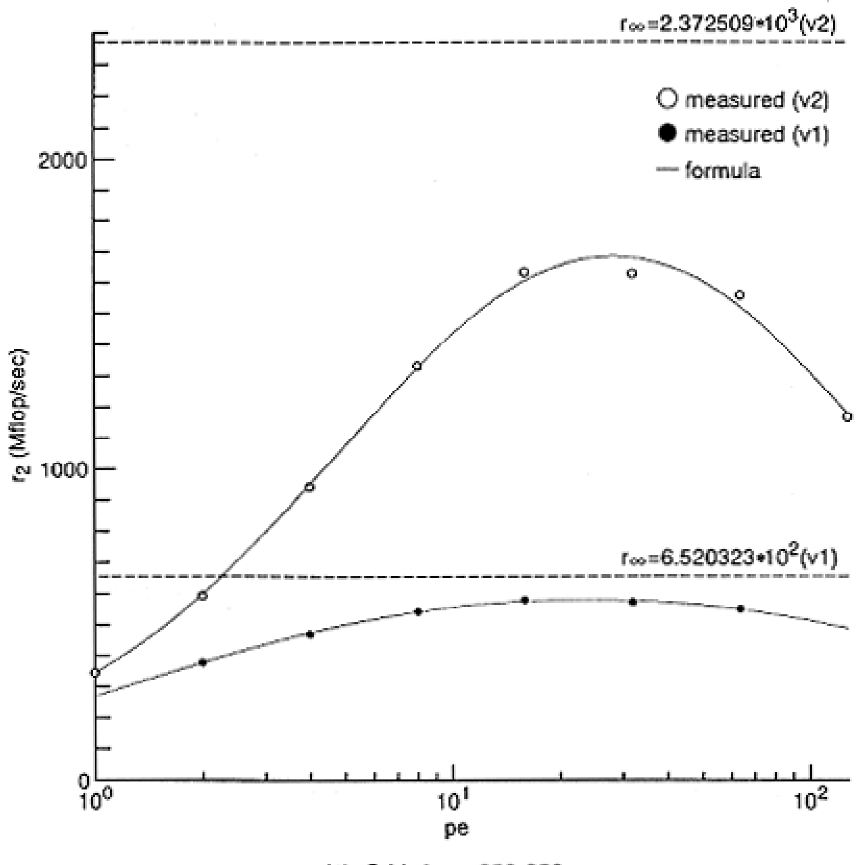


(1) Grid size = 256•256.

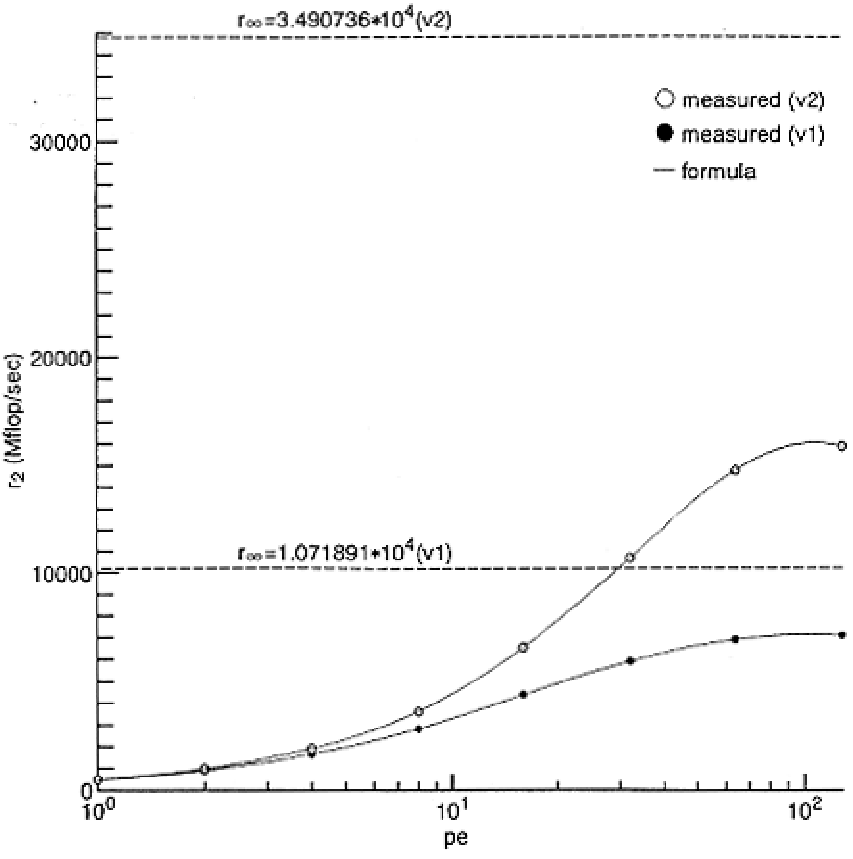


(2) Grid size = 1024•1024.

Fig. 18 Actual performance τ_2 of the square cavity program used the Jacobi method (J1)

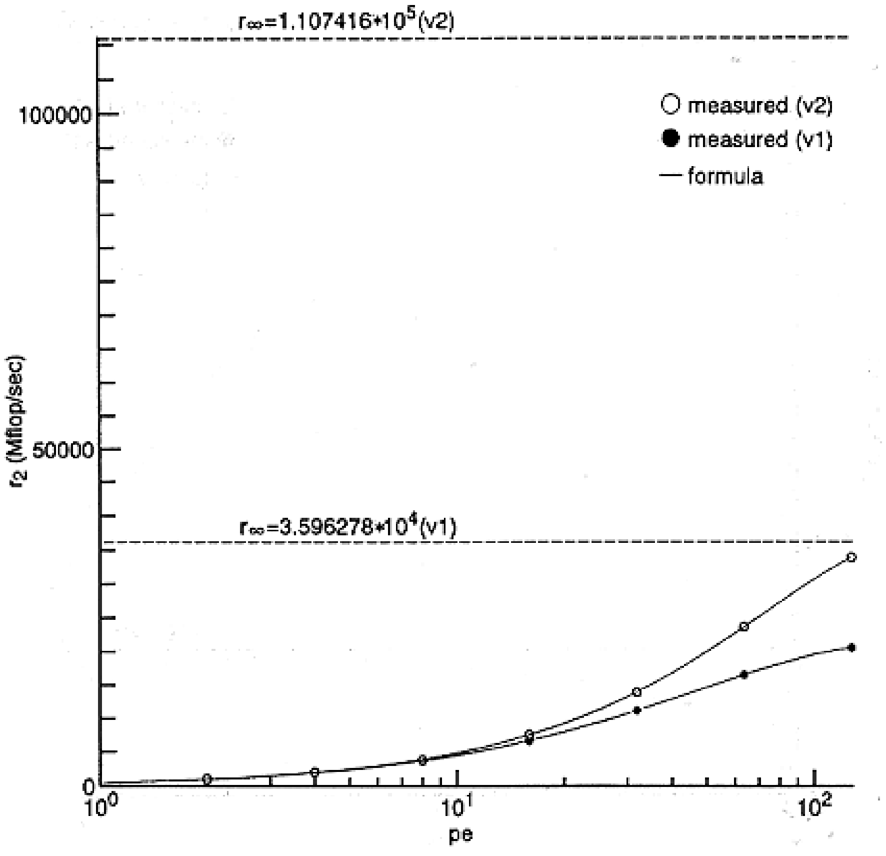


(1) Grid size = 256*256.

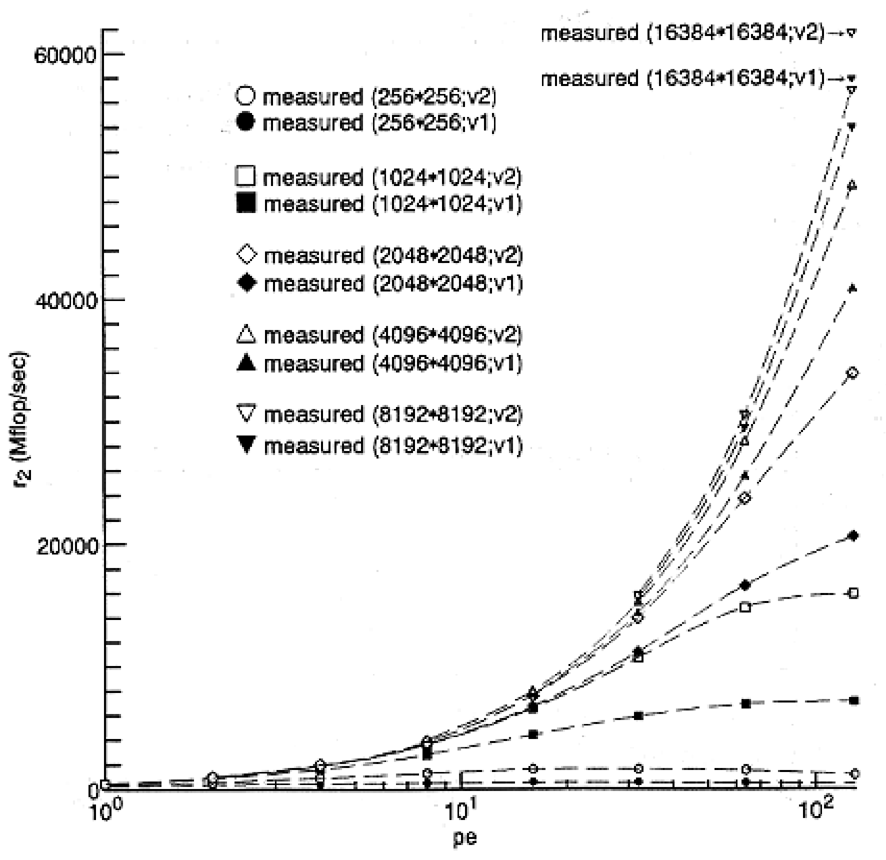


(2) Grid size = 1024*1024.

Fig. 19 Actual performance τ_2 of the square cavity program used the red-black method

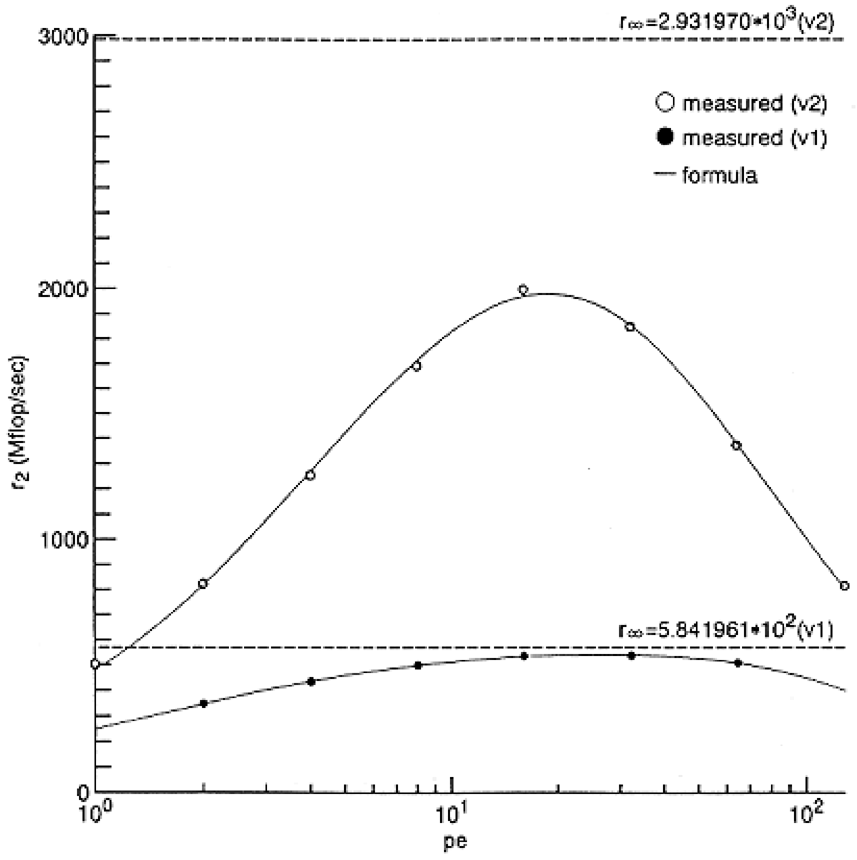


(3) Grid size = 2048*2048.

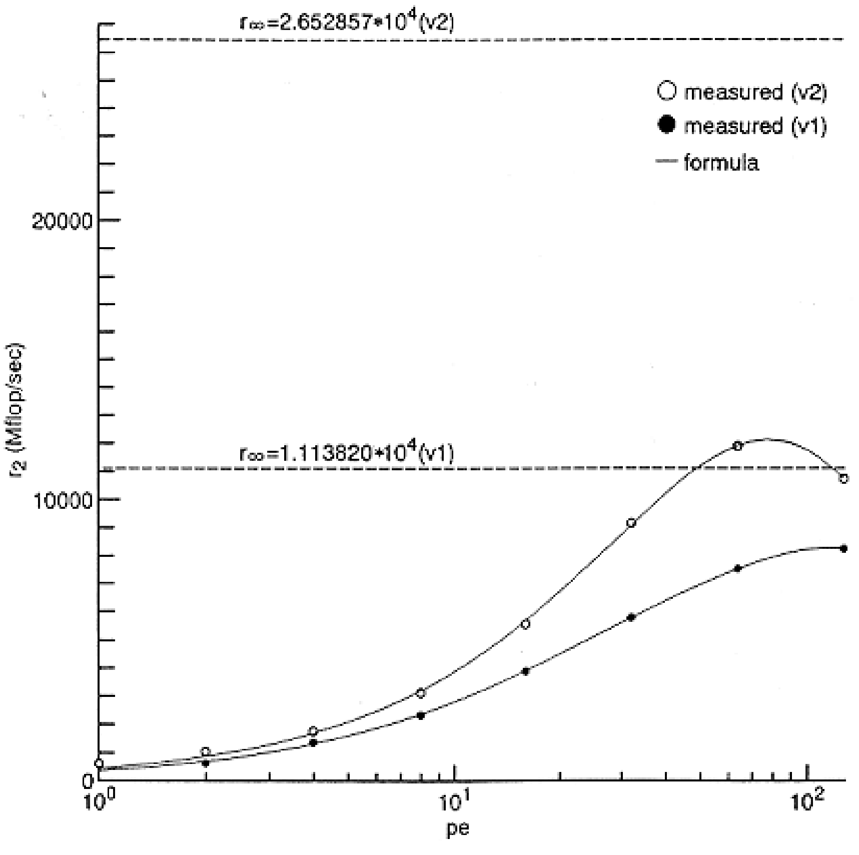


(4) Parameter = grid size.

Fig. 19 Continued



(1) Grid size = 256*256.



(2) Grid size = 1024*1024.

Fig. 20 Actual performance τ_2 of the square cavity program used the CG method

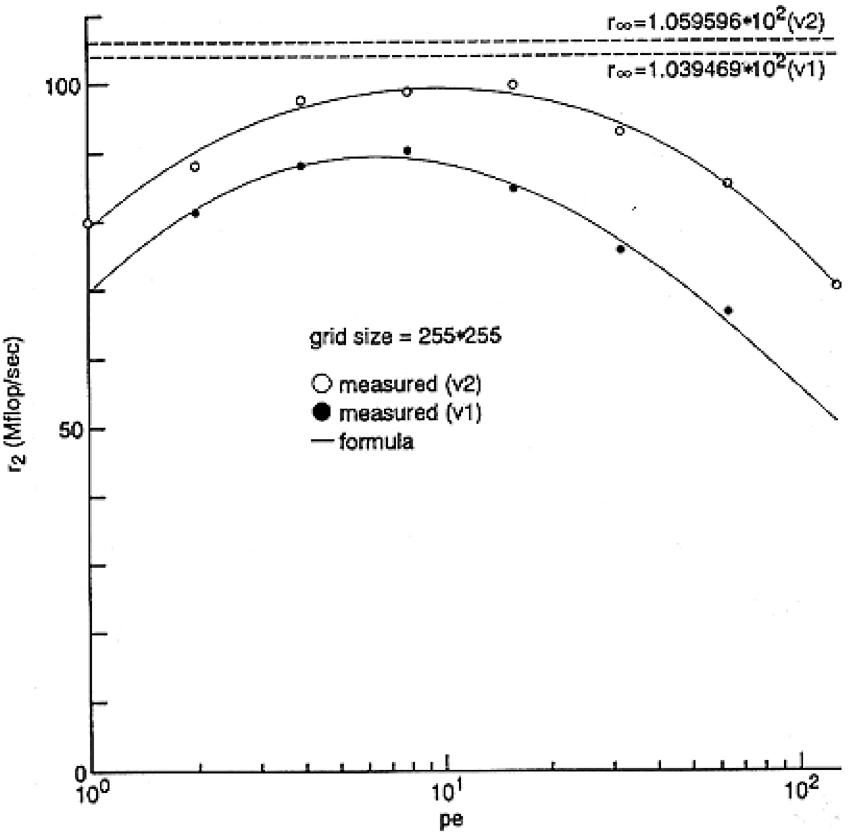


Fig. 21 Actual performance τ_2 of the square cavity program used the ADI method

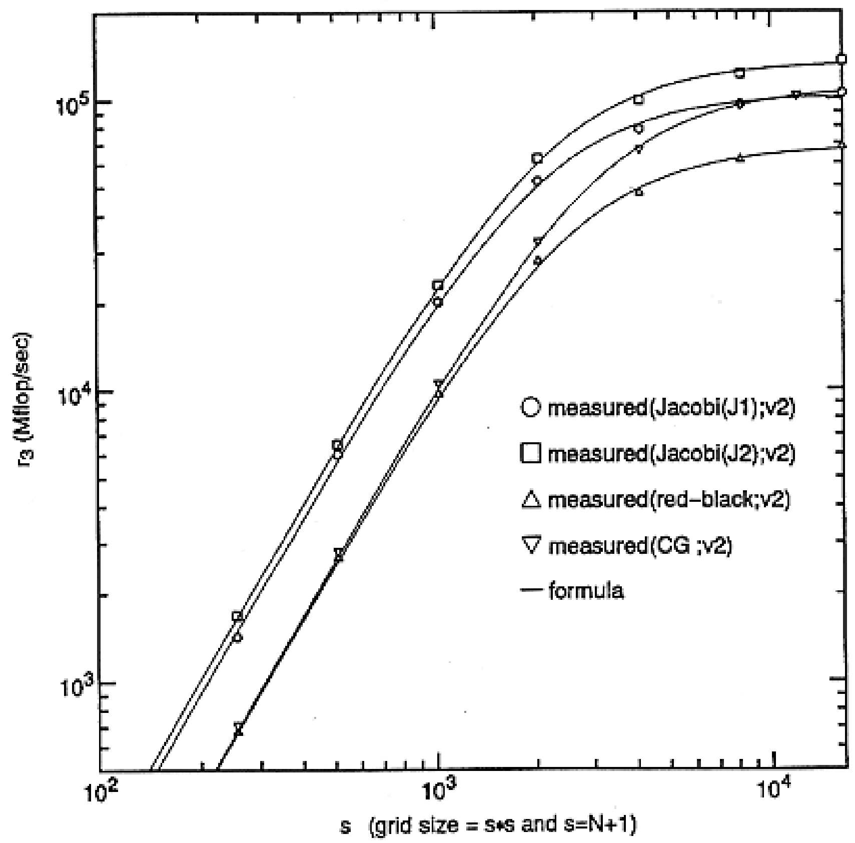


Fig. 22 Actual performance τ_3 of the square cavity program against the grid size (*pe* = 128)

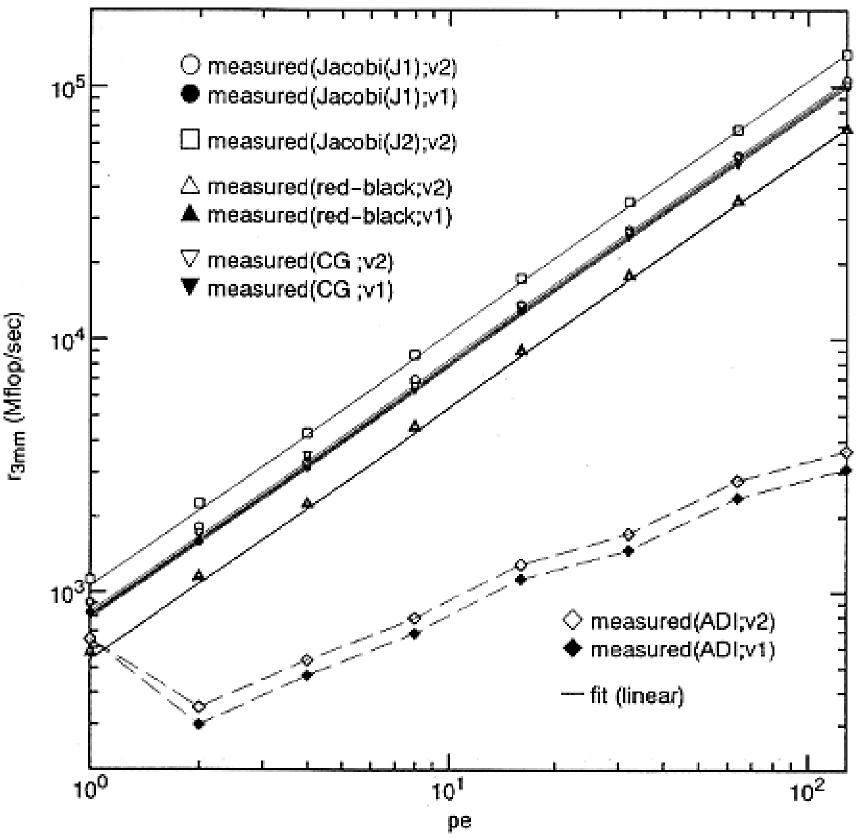


Fig. 23 Measured maximum of actual performance τ_3 of the square cavity programs against *pe* (note: The values for *pe* = 1 are ones for SIMD computing)