Received February 1999
Accepted November 1999

Parallel multilevel solution of Rayleigh-Benard-Marangoni problems

M.B. Davis
Mobile Technology Co, Dallas, Texas, USA
G.F. Carey
The University of Texas at Austin, Austin, Texas, USA

Keywords *Natural convection, Fluids, Buoyancy*

Abstract *The Rayleigh-Benard-Marangoni problem for natural convection in a rectangular cavity with thermocapillary forces on a free surface is investigated using a stream function-vorticity formulation. The nonlinear system is iteratively decoupled and high-degree* p *finite elements are used for the discretization of the physical domain. The linear systems arising from the discretization at each iteration are solved using a spectral multilevel scheme, which is a natural preconditioner for high-*p *(spectral) elements. The spectral multilevel solver lends itself to parallelization in an element-by-element (EBE) framework. Simulation results are presented and compared to previously published results. The multilevel efficiency is compared to previous results for the driven cavity problem. Parallel performance studies are presented for the Cray T3E distributed memory architecture.*

1. Introduction

Natural convection of an incompressible fluid can be driven by two different mechanisms: buoyancy forces due to temperature gradients and thermocapillary forces caused by gradients in the surface tension (Bénard, 1990; Carpenter and Homsy, 1989; Davis, 1968; Zebib *et al.,* 1985). We are particularly interested in the interaction of buoyancy and thermocapillary forces, and their effects in a microgravity environment where buoyancy is small, but the work is equally important for thin fluid layers in a normal gravity environment. The associated coupled flow and transport problem with both buoyancy and thermocapillary effects is termed the Rayleigh-Benard-Marangoni problem. The present formulation of this problem is based on the stream function-vorticity equations for incompressible fluid flow with an energy equation for the temperature field, the Boussinesq approximation to include buoyancy effects, and thermocapillary stresses due to thermal gradients on a free surface.

Under appropriate smoothness assumptions on the solution, p finite element methods in which the degree of the polynomial basis p can be increased to arbitrarily high degree provide superior accuracy and error convergence rates compared to low-degree elements (Babuska *et al.,* 1981). However, the condition number of the linear systems arising from the high-p (or spectral element) discretization deteriorates dramatically with increasing p, so standard iterative

This investigation has been supported by NASA ESSCAN Grant # NCCS5-154.

methods perform poorly (Carey and Barragy, 1989). A multigrid method in which the degree of the polynomial basis is the grid level is a logical preconditioner for such systems (Davis, 1996; Ronquist and Patera, 1987).

Spectral elements also form a natural basis for a processor decomposition for distributed memory parallel architectures. In the element-by-element (EBE) method the elements can be distributed individually to the processors and computations parallelized across the elements (Barragy and Carey, 1988; Davis, 1996). Some nodes will lie on element boundaries, and hence the information corresponding to these nodes will be shared by possibly several different elements. If the element interface corresponds to a processor boundary, then the information is duplicated on different processors, and information which must be shared or updated will then involve message passing for communication. The information to be communicated in this way can be bundled into send lists to minimize the effect of communication latency. This approach is used in the present study for the Rayleigh-Benard-Marangoni problem and the parallel performance is analyzed and presented for the Cray T3E architecture.

The outline of the presentation is as follows: In the next section the 2D steady R-B-M problem is formulated and the Galerkin finite element discretization leads to a coupled algebraic system. The iteratively decoupled scheme and p-multilevel method used to solve this system is then described. In the following section, simulations using the p-multilevel solver are compared to previously published results, and further phenomenological results are presented. Finally, the element-by-element parallelization is developed, and results presented for the selected distributed memory architecture.

2. Formulation and approximation

The classic Rayleigh-Benard problem corresponds to flow between two horizontal plates where the top plate is held at a constant (cool) temperature and the bottom plate is held at a higher constant temperature. At a critical Rayleigh number the heated fluid near the bottom plate becomes less dense and begins to rise while the cooler fluid near the top is more dense and descends. This leads to circular convection cells in two dimensions. If the plate is removed from the upper surface, then a thermocapillary surface traction due to temperature gradients on the free surface (Marangoni effect) also enters (Carpenter and Homsy, 1989; Zebib *et al.,* 1985). This is a direct consequence of the dependence of surface tension on temperature. It is clear that both buoyancy and thermocapillary effects may be important in driving the flow for this classical Rayleigh-Benard-Marangoni (R-B-M) problem. In shuttle and space station microgravity fluid applications buoyancy effects are small and the Marangoni effect is of primary interest. This is also the case in terrestrial applications where the fluid layer is thin (McLay and Carey, 1989; Hook, 1996).

The equations describing Rayleigh-Benard-Marangoni flows are the coupled incompressible Navier-Stokes and heat transfer equations. Here we will confine the study to two-dimensional steady-state flows and assume a non-deforming free surface (in some situations surface deformation for thin films is not negligible and should be included (Hook *et al.,* 1997). We use the stream function-vorticity formulation

$$-\nu\Delta\zeta + \mathbf{u}\cdot\nabla\zeta = f \tag{1}$$

$$-\Delta\psi = \zeta \tag{2}$$

where ζ is the vorticity, ψ is the stream function, $\mathbf{u}$ is the velocity, and f is obtained by taking the curl of the body force vector. The velocities are related to the stream function by $u = \frac{\partial\psi}{\partial y}, v = -\frac{\partial\psi}{\partial x}$. The body force $\mathbf{F}$ for this problem is the gravitional force associated with density differences in the fluid. Introducing the Boussinesq approximation for buoyancy (Carey and Oden, 1986).

$$\mathbf{F} = -\beta(T - T_0)g\mathbf{j} \tag{3}$$

where β is the coefficient of thermal expansion, g is the gravitional acceleration, T and T_0 are the fluid temperature and reference temperature respectively, and $\mathbf{j}$ is the unit vector in the positive y direction (opposite the direction of the gravitational force). Taking the curl of (3) and substituting into (1), the vorticity transport equation can be written

$$-\nu\Delta\zeta + \mathbf{u}\cdot\nabla\zeta = \beta g\mathbf{i}\cdot\nabla T \tag{4}$$

where $\mathbf{i}$ is the unit vector in the positive x direction.

The temperature of the fluid is governed by the energy transport equation

$$\rho c_p\mathbf{u} \cdot \nabla T - k\Delta T = 0 \tag{5}$$

where ρ is the fluid density, c_p is the specific heat, and k is the thermal conductivity.

The Marangoni problem involves a shear stress boundary condition. For a non-deformable surface $\frac{\partial v}{\partial x} = 0$, and hence the vorticity at the free boundary is $\zeta_{fb} = -\frac{\partial u}{\partial y}$. The surface stress tangent to the free boundary is $\tau_{fb} = \mu\frac{\partial u}{\partial y}$. Therefore, the vorticity is related to the shear stress component by

$$\zeta_f b = -\frac{\tau_f b}{\mu} \tag{6}$$

where μ is the dynamic viscosity of the fluid. The shear stress on the surface is equal to the gradient in the surface tension σ

$$\tau_f b = \frac{\partial\sigma}{\partial x} \tag{7}$$

Using the chain rule $\frac{\partial\sigma}{\partial x} = \frac{\partial\sigma}{\partial T}\frac{\partial T}{\partial x}$ and substituting into (6), the boundary condition for vorticity on the free boundary becomes

$$\zeta_f b = \frac{\sigma_T \frac{\partial T}{\partial x}}{\mu}. \tag{8}$$

where $\sigma_T = -\frac{\partial \sigma}{\partial T}$ is determined empirically for a given fluid. In numerical studies given later, we assume that σ varies linearly with T so σ_T is constant for a given fluid.

The equations are scaled as follows: $\mathbf{x}^* = \frac{x}{L}, \mathbf{u}^* = \frac{\mathbf{u}L}{\nu}, \psi^* = \frac{\psi}{\nu}, \zeta^* = \frac{\zeta L^2}{\nu}, T^* = \frac{T - T_0}{\Delta T}$, where ΔT is the mean temperature difference between the lower plate and the upper surface. Substituting these relations into (2), (4) and (5), we get the scaled, steady state form of the equations.

$$-\Delta\zeta^* + \mathbf{u}^* \cdot \nabla\zeta^* = Gr\mathbf{i} \cdot \nabla T^* \tag{9}$$

$$-\Delta\psi^* = \zeta^* \tag{10}$$

$$\Delta T^* + Pr\mathbf{u}^* \cdot \nabla T^* = 0 \tag{11}$$

and the boundary conditions transform similarly. For convenience, we drop the superscript * henceforth. The boundary conditions are as follows: $\psi = 0$ on $\partial\Omega$ (the boundary is a streamline), $\zeta = g(\psi)$ on $\partial\Omega_1$ (see Davis, 1996, for a discussion of vorticity boundary conditions), $T = T_1(x, y)$ (isothermal boundary) or $\frac{\partial T}{\partial n} = 0$ (adiabatic boundary) on $\partial\Omega_1$, where $\partial\Omega_1$ is that part of the boundary which is *not* a free surface, and $\partial\Omega$ is the entire boundary. On the free surface $\partial\Omega_2$, the boundary conditions are $\zeta = M\frac{\partial T}{\partial x}$ and $\frac{\partial T}{\partial n} + Bi(T - T_{out}) = 0$, where T_{out} is the temperature of the surrounding fluid. In all numerical experiments presented later, the Biot number Bi is set to zero, and then $\frac{\partial T}{\partial n} = 0$. The non-dimensional constants are: the Marangoni number $M = \frac{\sigma_T L \Delta T}{\rho\nu\alpha}$, the Prandtl number $Pr = \frac{\nu}{\alpha}$, the Grashof number $Gr = RaPr = \frac{\beta g L^3 \Delta T}{\nu^2}$, and the Rayleigh number $Ra = \frac{\beta g L^3 \Delta T}{\nu\alpha}$.

Equations (9), (10) and (11) constitute a second order, elliptic, nonlinear coupled system of equations in three scalar variables. In the present work, the equations are discretized using a Galerkin finite element method with a polynomial tensor product basis of arbitrary degree p defined over two-dimensional quadrilateral elements. Use of high-p finite elements can give higher-order accuracy than low-degree elements on meshes with a comparable number of grid points. This is clear from the standard L^2 finite element error estimate for elliptic PDE's (Babuska *et al.,* 1981).

$$\|e\| = C(p)h^\mu, \quad \mu = min(r, p+1) \tag{12}$$

where r is the regularity of the solution ($u \in H^r$) and p is the degree of the basis. For smooth solutions the convergence rate with respect to h is then $O(h^{p+1})$. If we increase the polynomial degree instead of decreasing the grid spacing we get exponential reduction of the error. This rate is not achieved in the vicinity of a singularity due to the local lack of regularity. In such regions the error is $O(h^r)$ and, therefore, increasing p will not increase the rate beyond r. The optimal refinement strategy should be to decrease the element size h near a singularity, and to increase the degree p in smooth regions. Of course, we can grade the mesh (redistribute the grid) to cluster near a singularity and then increase p uniformly.

One disadvantage of the p-type finite element method is that the conditioning of the matrix deteriorates with increasing p. One way to counter this is to apply a preconditioner to the system. A p-type multilevel method may be defined by using the degree of the polynomial basis as the grid level. The intergrid transfers can then be naturally defined in terms of expansions in the appropriate bases. A more detailed discussion of the p-multilevel scheme used here can be found in Davis (1996).

Introducing a weighted residual approach for equations (9), (10), and (11) and integrating by parts we obtain the weak statement for the R-B-M problem: find (ψ, ζ, T) satisfying the essential boundary conditions and such that

$$\int_\Omega (\nabla\zeta \cdot \nabla w + \mathbf{u} \cdot \nabla\zeta w - Gr\mathbf{i} \cdot \nabla T w) dx = 0 \tag{13}$$

$$\int_\Omega (\nabla\psi \cdot \nabla v - \zeta v) dx = 0 \tag{14}$$

$$\int_\Omega (\nabla T \cdot \nabla q + Pr\mathbf{u} \cdot \nabla T q) dx = 0 \tag{15}$$

Introducing a finite element discretization and p finite element basis to define the expansions ζ_h, ψ_h, T_h and substituting into (13), (14), (15), we get a fully coupled system which we can solve using a nonlinear solution scheme. Here we decouple the nonlinear system into blocks corresponding to the individual equations and solve the system iteratively using successive approximation (see Davis (1996)).

Symbolically, the iteratively decoupled form is

$$\mathbf{K}\psi^n = \mathbf{G}^n - 1 \tag{16}$$

$$\mathbf{A}^n\zeta^n = \mathbf{F}^{n-1} \tag{17}$$

$$\mathbf{C}^n\mathbf{T}^n = \mathbf{H} \tag{18}$$

for finite element basis p_k, where the vectors and matrices are defined by

$$K_{ij} = \int_{\Omega_h} (\nabla p_i \cdot \nabla p_j) dx \tag{19}$$

$$G_i^{n-1} = \int_{\Omega h} \zeta_h^{n-1} p_i dx \tag{20}$$

$$A_{ij}^n = \int_{\Omega h} [\nabla p_i \cdot \nabla p_j + ((\psi_h^n)_y (p_j)_x - (\psi_h^n)_x (p_j)_y) p_i] dx \tag{21}$$

$$F_i^{n-1} = \int_{\Omega h} Gr(T_h^{n-1})_x p_i dx \tag{22}$$

$$C_{ij}^n = \int_{\Omega_h} [\nabla p_i \cdot \nabla p_j + Pr((\psi_h^n)_y (p_j)_x - (\psi_h^n)_x (p_j)_y) p_i] dx \tag{23}$$

The vorticity boundary conditions on the free boundary are given by the thermocapillary condition (8) and can be computed on the remainder of the boundary by using the current iterate of ψ_h. Note that not only are $\mathbf{A}^n$, $\mathbf{C}^n$ functions of ψ^{n-1}, they are also in general nonsymmetric. This can cause problems in the solution phase depending on the degree of asymmetry, which increases with the flow velocity.

The linear systems at each solution step are solved using a multilevel scheme. A typical V-cycle on degree p involves smoothing iterations at each degree level followed by residual projections to lower degree levels and a coarse level solve then interpolation and smoothing corrections on successively higher levels to complete the cycle. Note that the algorithm involves an outer iteration for the successive approximation scheme and block decoupling together with inner multilevel interations on each of the linear subsystems. This suggests that a more efficient algorithm can be devised by reducing the convergence tolerances of the respective "inner" and "outer" iterations dynamically as solution proceeds. As the outer, or block, iterations proceed toward convergence, the stopping tolerance on the inner multilevel scheme can be reduced, or the number of V-cycles increased.

3. Numerical results

3.1 Validation studies/Rayleigh-Benard

The scheme and software were first verified on the following benchmark natural convection problem (Davis and Jones, 1983; Davis, 1983). Consider flow in a unit square domain with adiabatic top and bottom walls (no free surface), temperatures $T = 1$, $T = 0$ on the left and right walls respectively, with $Pr = 0.71$, and at several different Rayleigh numbers. No-slip conditions apply on all walls. The computed mean Nusselt number ($Nu = \frac{1}{H}\int_0^L q\, dy$, where q is the heat flux) at the left wall (Nu_0) and at the midplane $x = 0.5$ ($Nu_{1/2}$), and the stream function at the midpoint (ψ_{mid}) are compared to the results from Davis (1983) in Table I. The benchmark case only reports the quantities to four significant figures, and the reported accuracy of the calculations is within 1 per cent. The agreement for all Rayleigh numbers is good, with less than 1 per cent difference in all quantities. The differences do increase as *Ra* increases due to the increased difficulty of the problem. Note that the relative difference between Nu_0 and $Nu_{1/2}$, which should be a general indicator of the convergence of the solution with the grid, is less for the multilevel calculations than for the benchmark case. This is due to the superior accuracy of high-*p* elements. (The benchmark simulations in Davis (1983) used a relatively low-order finite volume grid.)

Contours of the stream function, vorticity and temperature at $Ra = 10^6$ are shown in Figure 1 and provide the essential features of this type of flow. The concentric stream function contours show the expected flow rotation, which is clockwise. In this case and all subsequent cases, the boundary corresponds to the $\psi = 0$ streamline. The vorticity gradients are concentrated near the

Table I.
Comparison of specific results to benchmark case

	p-multilevel			Benchmark		
Ra	Nu_0	$Nu_{1/2}$	ψ_{mid}	Nu_0	$Nu_{1/2}$	ψ_{mid}
10^3	1.1178	1.1178	1.1746	1.117	1.118	1.174
10^4	2.2450	2.2448	5.0737	2.238	2.243	5.071
10^5	4.5189	4.5216	9.1155	4.509	4.519	9.111
10^6	8.8179	8.8252	16.386	8.817	8.799	16.32

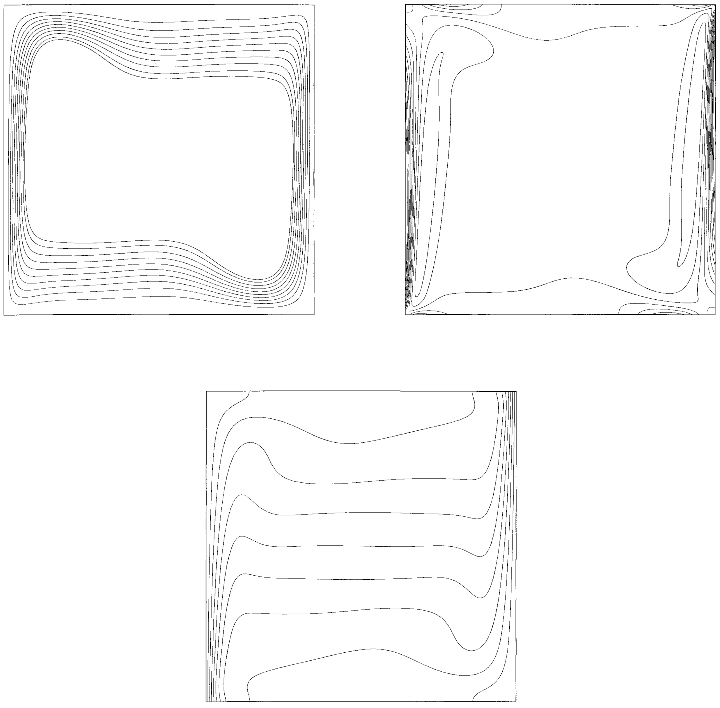


Figure 1.
Stream function contours (upper left, equally spaced between –1.67 and –16.3), vorticity contours (upper right, equally spaced between 1.529×10^4 and $-3{,}178 \times 10^3$, and temperature contours (lower, equally spaced between 0.1 and 0.9), respectively, $Ra = 10^6$, benchmark problem

boundaries, corresponding to a boundary layer type of behavior at this value of *Ra*. The clockwise convection of the vorticity is also evident. The temperature is convected in a clockwise manner from the pure conduction solution. The contour values are the same as in Davis (1983) and show excellent agreement.

The second flow geometry tested was a long rectangular box of length 12 times the height with Pr = 13.3 and Ra = 18668.The temperatures on the bottom surface and top surface are $T = 334K$ and $T = 333K$ respectively. The flow structure is more complex than the previous case and in Argyris *et al.* (1986), this case was used to investigate the transient effects of a disturbance at the lower left corner. Here, we consider the steady-state problem and Figure 2 shows the computed stream function and temperature contours. There are nine equal recirculation cells, with the flow rotating in opposite directions in adjacent cell (counter-rotating pairs). The results agree with those in Argyris *et al.* (1986) at steady state.

3.2 Marangoni and R-B-M flow

The next numerical experiment compares pure buoyancy-driven flow to thermo-capillary-driven flow. The flow domain and boundary conditions correspond to those in the first validation study ($T = 1$ and $T = 0$ on left and right walls, respectively) except that the top is now a flat free surface. The Rayleigh number is 10^3 and the problem is solved at several different Marangoni numbers. Figure 3 shows the solution at M = 1, 100 and 1,000, respectively. At M = 1, surface tension effects are small and the solution is similar in structure to a classic buoyancy driven flow. The streamlines are roughly circular and are not particularly concentrated near any boundary. At M = 100, the effect of the thermocapillary force at the free surface is more pronounced, and at M = 1,000, the stream function solution looks roughly like the classic driven cavity problem in that the flow is being strongly driven by surface tension at the top boundary. It is interesting to consider the case where the surface tension is oppositely directed. This is the case for certain fluids when impurities are present (see McLay and Carey (1989)). In Figure 4 we show the stream function contours for M = –10 and M = –100, respectively. The contours at M = –10 look similar to the solution at M = 1, due to the small

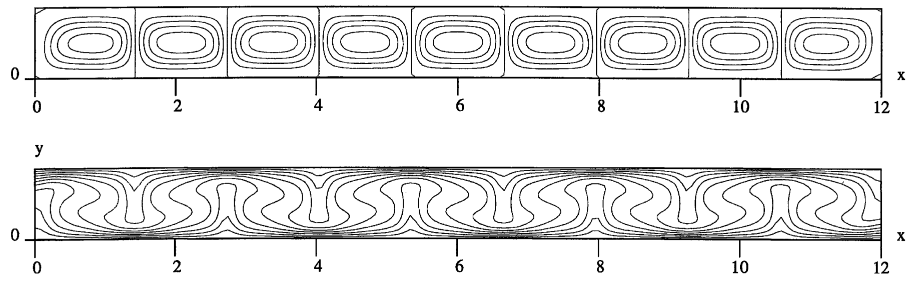


Figure 2. Stream function contours (top, equally spaced between –11.8 and 11.8) and temperature contours (bottom, equally spaced between 0.1 and 0.9, long rectangular box

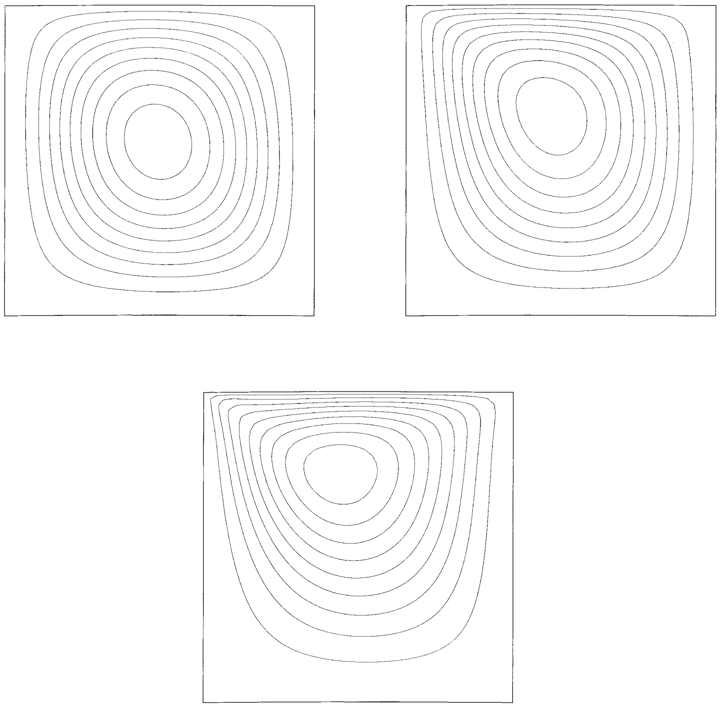


Figure 3.
Stream function contours, free surface, $Ra = 10^3$, $M = 1$ (left, equally spaced between 0.115 and 1.32), $M = 100$ (center, equally spaced between 0.157 and 1.81), and $M = 1{,}000$ (right, equally spaced between 0.417 and 4.8)

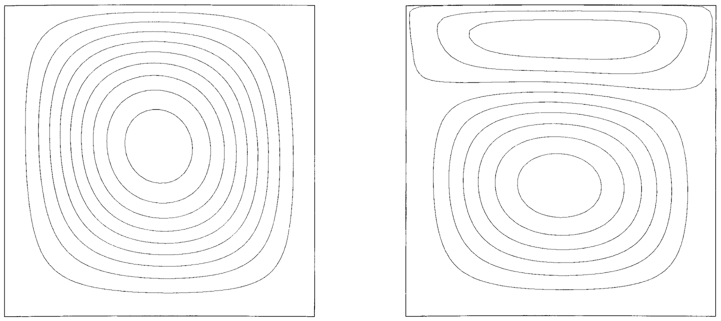


Figure 4.
Stream function contours, free surface, $Ra = 10^3$, $M = -10$ (left, equally spaced between 0.109 and 1.26) and $M = -100$ (right, equally spaced between –0.354 and 0.844)

thermocapillary effect. At $M = -100$, though, the surface tension effect is strong enough to reverse the flow on the top surface and two counter-rotating cells are formed.

The relative effect of buoyancy in a free surface flow can be reduced by increasing the length of a two-dimensional cavity relative to the height. Figures 5 and 6 show the stream function contours for a free surface flow with $=Ra = 10^3$, $M = 10$, and $Pr = 1$. The left and right side walls are again held at $T = 1$ and $T = 0$, respectively. The length of the box is held constant at $L = 1$, while the height takes the successive values $H = 1, 0.5, 0.1, 0.05$. The horizontal dimension is scaled so that all plots appear on the same square, for clarity in comparing the contours. One can see that as the height is reduced, all convection between the top and bottom surfaces occurs at the ends of the cavity, and the streamlines concentrate near the free surface, indicating that the flow is progressively driven more strongly by the surface tension. Note that, in contrast to the case in which a slender rectangular cavity is heated at the bottom plate and multiple cells occur (Figure 2), only one recirculation cell is now formed. The average Nusselt number at the bottom surface is plotted against *L/H* in Figure 7. One can see that the heat transfer at the bottom plate quickly asymptotes to the pure conduction case, represented by $Nu = 1$.

3.3 Multilevel convergence

The convergence of the multilevel method is measured by calculating the reduction in the discrete L^2 norm of the residual on the fine grid p ($\mathbf{r}_p^n = \mathbf{b}_p - \mathbf{A}_p\mathbf{u}_p^n$,where $\mathbf{u}_p^n$ is the solution approximation vector after V-cycle n) over one complete V-cycle. The residual reduction factor is defined as $\frac{\|\mathbf{r}^n\|}{\|\mathbf{r}^{n-1}\|}$, and the lower the reduction factor, the more efficient the multilevel scheme. Table II shows the residual reduction factor for the first problem in Section 3.2 with $Ra = 10^4$, $M = 100$, and $Pr = 1$ at $p = 1,\cdots,6$ on a 6×6 element grid and a 10×10 element grid. For each value of p the coarsest level is $p = 1$,

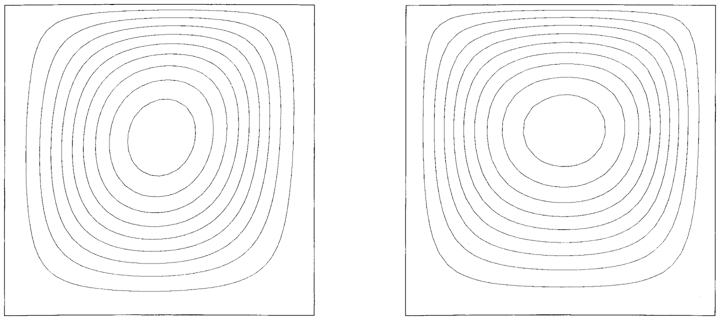


Figure 5. Stream function contours, $Ra = 10^3$, $M = 10$, $H = 1$ (left, equally spaced between –0.258 and 0.748) and $H = 0.5$ (right, equally spaced between –1.38 and –0.12). Vertical dimension scaled for visualization

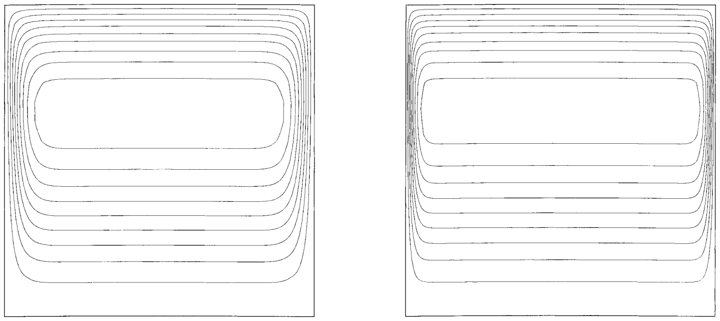


Figure 6. Stream function contours, $Ra = 10^3$, $M = 10$, $H = 0.1$ (left, equally spaced between –0.331 and -2.882×10^{-2}) and $H = 0.05$ (right, equally spaced between -8.815×10^{-4} and -7.66×10^{-5}). Vertical dimension scaled for visualization

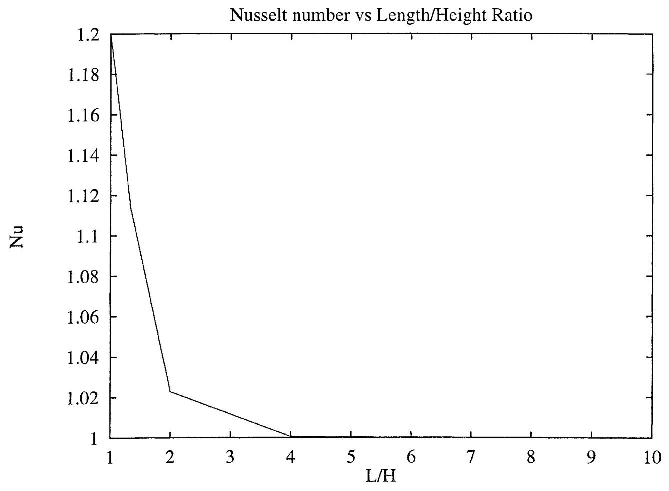


Figure 7.
Nusselt numbers vs $\frac{L}{H}$, $Ra = 10^3$, $M = 10$

Table II.
Multilevel convergence factors, $Ra = 10^4$, $M = 100$

	6 × 6 grid			10 × 10 grid		
p	ψ	ζ	T	ψ	ζ	T
2	0.12	0.08	0.62	0.11	0.09	0.63
3	0.18	0.16	0.80	0.18	0.15	0.80
4	0.31	0.32	0.90	0.26	0.32	0.91
5	0.40	0.52	0.94	0.31	0.52	0.97
6	0.54	0.70	0.99	0.44	0.70	0.997

that is a complete multilevel cycle. The tabulated values represent an average of the reduction factors over the nonlinear solution iterations. Notice that the convergence factors are essentially independent of the grid size, which is the desired result. They are not, however, independent of the degree p. The higher p, the worse the convergence factor. This is due to the fact that the exact solution on the coarse grid becomes further and further from the fine grid solution as p increases. The linear systems on intermediate grids are not solved exactly, but only get a prespecified number of pre- and post-smoothing operations. Therefore, the frequencies operated on by the intermediate grids are only damped by a certain amount, not eliminated as on the coarse grid. Also notice that the factors for the vorticity and temperature equations are worse than for the stream function equation, due to the additional difficulty in solving the nonsymmetric transport equation.

4. Element-by-element parallelization

In the finite element method a given problem domain is partitioned into a union of elements for discrete solution. Hence schemes in which individual elements or blocks of elements are operated on by a processor and the processor decomposition follows element boundaries provide a natural way to parallelize finite element methods (Barragy and Carey, 1988; 1992; Davis, 1996). Adjacent elements share nodes on the element interface, so the information associated with these nodes may be stored on different processors. This information is updated during, for example, global matrix-vector product or inner product operations, and this means that messages must be passed between processors in order to update these values. The ratio of communication to computation is important because it can limit efficiency. In parallel element-by-element schemes, the use of high-*p* elements, which have more internal degrees of freedom, results in a higher computation to communication ratio than for the same number of low-degree elements.

For a message passing paradigm, the time to send a message can be represented conveniently by

$$t_m = \alpha + \beta L_m \tag{24}$$

where α is the startup time, β is the time per byte for message transfer, and L_m is the length of the message in bytes. Rather than send a large number of short messages, it is obviously better to send a few longer messages so that the startup "overhead" is minimized. Otherwise the startup time may dominate the communication time. The optimum situation would be to send one long message rather than several small messages and to overlap this communication with computation.

The previous argument motivates the need for message bundling using sendlists. A data structure is developed for the present work in which each processor has a pointer array which contains the element and node numbers that are shared with another processor. The order in which this information is to be placed into a message is also stored. Thus, when a vector is to be updated, a message vector is filled in order and sent to the appropriate processor. In turn, a message is received from that processor. A pointer array indicates which element and local node corresponds to a given position in the array, in the same way as for the message which was sent. In this fashion all of the communication between adjacent processors can be accomplished using one message each way, and message latency is minimized. There is, however, some overhead in the packing and unpacking phases.

The decomposition of the domain is accomplished by counting out elements in order to successive processors. That is, the first $\frac{N_e}{N_p}$ elements are placed on processor 0, the second $\frac{N_e}{N_p}$ on processor 1 and so on. The elements are numbered naturally in a structured grid. A representative decomposition for a 4×4 grid on 8 processors is shown in Figure 8. It should be noted that this scheme can result in non-contiguous subdomains, but for moderate numbers of processors ($\frac{N_e}{N_p} \gg 1$) the subdomains tend to be contiguous.

In the present work we can use an element-type data structure and recast all matrix-vector or projection operations at the element level. This means that

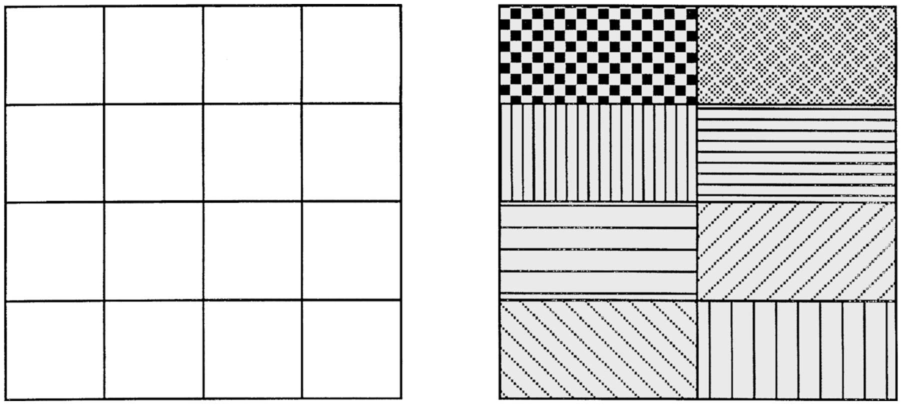


Figure 8.
Procesor decomposition for a 4 $\times$ 4 grid of 16 elements on eight processors

instead of addressing a component in a vector by its global node number, it is addressed by its element and local node number. In addition, each element has a pointer array which stores its neighbor elements and nodes that are shared with this neighbor. A specific processor will store information only for elements local to that processor. Elements are therefore addressed by the number local to that processor rather than a global element number. The pointer array for neighbor information includes the local element number and processor number for neighboring elements. This format facilitates parallel coding.

The formation of the matrix and RHS vector for finite element methods is usually accomplished by computing the local element matrices and vectors and assembling them to get the global matrix and RHS. However, in the present parallel algorithm we no longer form the global matrix and RHS, but leave them in element form. The matrix and RHS calculation phase is therefore completely parallel. If the matrix is to be preconditioned using a global Jacobi preconditioner (diagonal scaling), then the diagonal elements of the matrices may be assembled to find the scaling vector. This accumulation phase will involve communication across processor boundaries.

Iteration by point iterative methods (Jacobi, SOR, etc.) as a smoother or gradient methods (CG, BCG, etc.) for the coarse grid solve involves repeated matrix-vector multiplications or dot products and requires that the information on shared nodes be updated. For instance, to compute a matrix-vector product such as

$$\mathbf{A}_p \mathbf{u}_p{}^* = \mathbf{v}_p \tag{25}$$

in the residual calculation for the multilevel scheme, we write $\mathbf{U}_p^e = \mathbf{B}_e \mathbf{u}_p{}^*$ where $\mathbf{B}_e$ is the Boolean (adjacency or connectivity matrix) for element e and relates local to global variables. Then

$$\mathbf{A}_p = \sum_{e=1}^{N_e} \mathbf{B}_e^T \mathbf{A}_p^e \mathbf{B}_e \tag{26}$$

and

$$\begin{aligned}\mathbf{A}_p\mathbf{u}_p^* &= (\sum_{e=1}^{N_e}\mathbf{B}_e^T\mathbf{A}_p^e\mathbf{B}_e)\mathbf{u}_p^* = \sum_{e=1}^{N_e}\mathbf{B}_e^T\mathbf{A}_p^e\mathbf{u}_p^e \\ &= \sum_{e=1}^{N_e}\mathbf{B}_e^T\mathbf{v}_e = v\end{aligned} \tag{27}$$

Hence the calculation requires element matrix-vector products that can be carried out independently in parallel. Note that the solution vector is stored in summed form, but still in element format. The element accumulation in (27) requires communication if the element boundary corresponds to a processor boundary.

Hence we see that when appropriately designed, multilevel methods can also exploit parallel EBE and subdomain approaches. Obviously the smoothing phase proceeds as before with EBE matrix-vector products updated across processor boundaries. The issues of the residual calculation, restriction, and prolongation also need to be addressed.

For example, consider the residual calculation $\mathbf{r}_p = \mathbf{b}_p - \mathbf{A}_p\mathbf{u}_p*$. In the EBE structure we obtain

$$\sum_{e=1}^{N_e}\mathbf{B}_e^T\mathbf{r}_p^e = \sum_{e=1}^{N_e}\mathbf{B}_e^T\mathbf{b}_p^e - \sum_{e=1}^{N_e}\mathbf{B}_e^T\mathbf{A}_p^e\mathbf{B}_e\mathbf{u}_p^* \tag{28}$$

but $\mathbf{B}_e\mathbf{u}_p* = \mathbf{u}_p^e$ so (28) implies

$$\sum_{e=1}^{Ne}\mathbf{B}_e^T\mathbf{r}_p^e = \sum_{e=1}^{Ne}\mathbf{B}_e^T(\mathbf{b}_p^e - \mathbf{A}_p^e\mathbf{u}_p^e) \tag{29}$$

and we can use directly the element residuals

$$\mathbf{r}_p^e = \mathbf{b}_p^e - \mathbf{A}_p^e\mathbf{u}_p^e \tag{30}$$

Note also that because the element bases are defined locally we can introduce a local change of basis at the element level and an element projection matrix $\mathbf{M}_e^{q,p}$. Then the element residual projection follows as

$$\mathbf{r}_q^e = \mathbf{M}_e^{q,p}\mathbf{r}_p^e \tag{31}$$

Thus residual calculation and restriction take place on the element level, without communication, and are completely parallel operations. The coarse level error correction vector is stored in summed format. Therefore, prolongation to higher levels can also take place locally on an element and is once again completely parallel.

To summarize, the basic steps of the parallel algorithm for a two-level scheme are:

(1) Processor partition. A processor partitioning of the mesh is made (contiguous element blocks are desirable). Sendlists for interprocessor communication are constructed.

(2) High-level smoothing iteration.
   - For each processor subdomain. In parallel, compute element matrix and vector contributions at every level and store element wise$\{\mathbf{A}_p^e\}, \{\mathbf{b}_p^e\}, \{\mathbf{A}_q^e\}, \{\mathbf{b}_q^e\}$.
   - For $k = 1, 2, \ldots, K$ smoothing iterations carry out relaxed Jacobi iteration (or a similar scheme). This involves local element matrix-vector products with element solution vector iterate $\{\mathbf{u}_p^e\}$ and communication between adjacent processors for element nodes on an interprocessor boundary.

(3) Residual computation and projection. For each subdomain in parallel, compute element residuals (level p) and locally project to level q to get residuals $\mathbf{r}_q^e$. For the hierarchic basis this reduces to simply computing the first N_q^e components of the residual for each element e.

(4) Coarse grid solution. The coarse grid system is solved in parallel using an element-by-element generalized conjugate gradient solver.

(5) High-level update. The coarse level correction for each element is projected element wise to the higher level using (23) and these p-level element corrections are added to the current p-level element iterate.

(6) Return to Step 2(b) and repeat the cycle until the fine grid iterate satisfies a specified stopping test.

Remark 1: In the above procedure all matrices and vectors are generated and stored element wise. This permits a more straightforward parallel implementation and simplifies coding.

Remark 2: Additional level projections and smoothing iterations can be included in the usual way.

Figure 9 shows the speedup and efficiency curves for the benchmark case in section 3.1 with $Ra = 10^4$. For each value of p, the maximum problem size which will fit in memory on one processor of the Cray T3E is used. The element grids which represent this maximum memory case are given in the legend. This is an attempt to compare the speedups at each p for the largest case (for that value of p). Notice that the speedup is best for $p = 1$, and falls off for higher p. This is due to the fact that more $p = 1$ elements fit in memory, so for constant problem size at higher numbers of processors the $p = 1$ case still has a reasonable number of elements per processor.

Figure 10 also shows speedup and efficiency, but the number of elements (256) is the same for each value of p. This case is an attempt to compare speedup at different p for a constant element grid. As expected, the high p elements have better speedup due to the fact that there is more computational work per element at higher p.

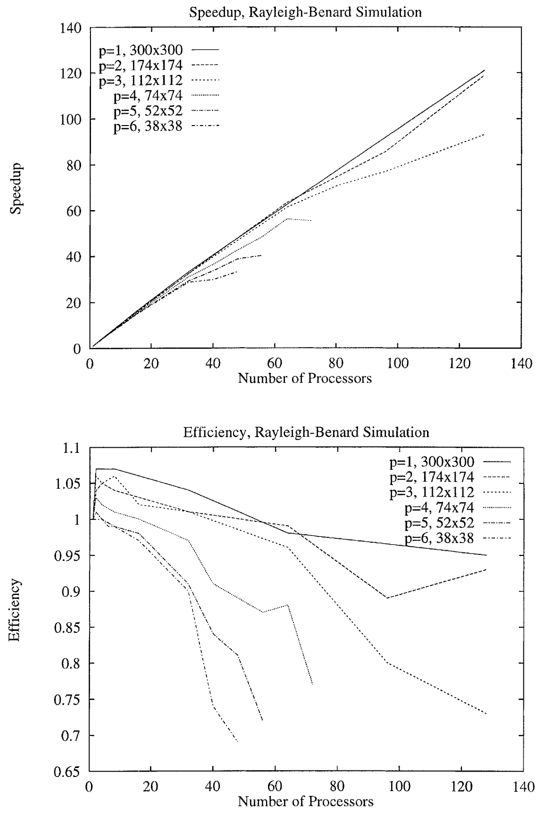


Figure 9.
Speedup, Cray T3E, memory limit for each p

In Figure 11 speedup and efficiency are again presented, but here the number of grid points on the fine grid is the same for each value of p. The element grids which achieve this for each value of p are noted in the legend. Again, $p = 1$ elements perform best in terms of speedup, and performance falls off at higher p.

One reason that lower degree elements perform better in these examples is the effect of dot products on the coarse grids. The coarse grid is solved using a gradient-type iterative solver, which involves dot products at each iteration. The dot product requires a global accumulation of the sum, and the communication time required for this increases with the number of processors. The coarse grid for these studies is the bilinear element. So for high-p elements, the coarse grid contains relatively few bilinear elements, and therefore a smaller amount of computation. Hence, as the number of processors is increased, the communication times of the dot products becomes a significant part of total computational time, and this effect is more pronounced for the high-p elements.

Figures 12-14 show the scaled speedup, which is defined as the MFLOP rate versus the number of processors with the problem size per processor held constant. The comparisons are shown for the same problem and the same grids

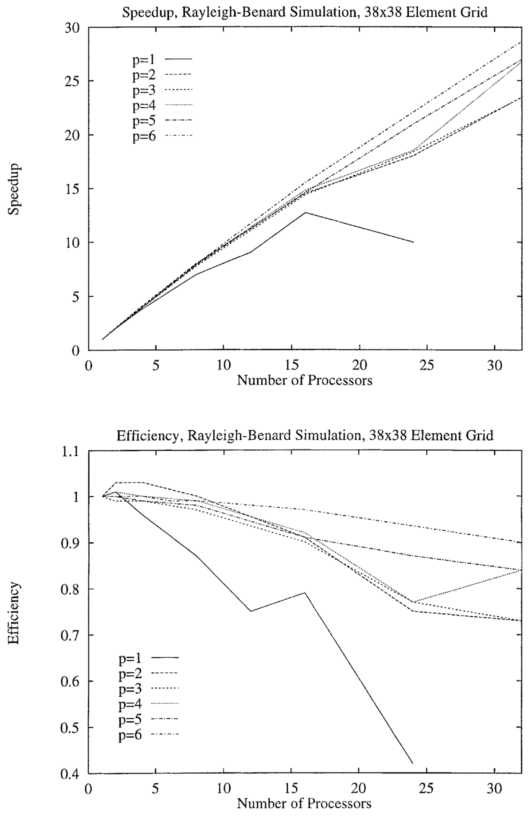
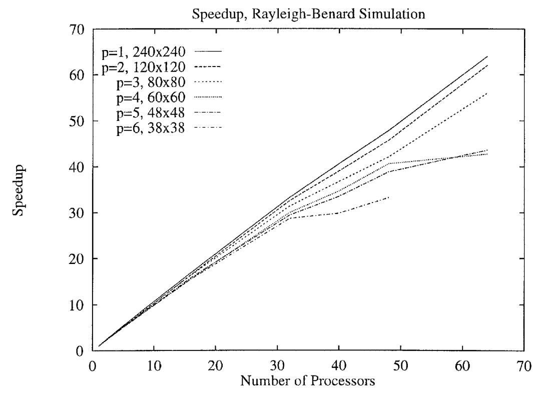


Figure 10.
Speedup, Cray T3E, constant element grid

as mentioned in the speedup comparisons. For each case the performance is best for $p = 6$, and falls off monotonically for lower p elements. However, this is due primarily to the fact that $p = 6$ elements perform better on a per processor basis. The degree of scalability for all element degrees is about the same. The per processor performance is low for this machine, and this is due to the fact that some of the optimization flags were not used in these examples, and further optimizations (like BLAS) were not used. Once again, dot products degraded the performance for higher numbers of processors.

5. Conclusions

In this study we consider a class of coupled viscous flow and heat transfer problems involving thermocapillary surface tension and buoyancy effects using a parallel high-p element-by-element approach. Several phenomenological studies are carried out to explore the role of surface tension (the Marangoni effect) relative to buoyancy. The simulations involve a parallel multilevel strategy in which the element degree defines the level. Fixed problem size speedup and scaled speedup studies as well as efficiency studies



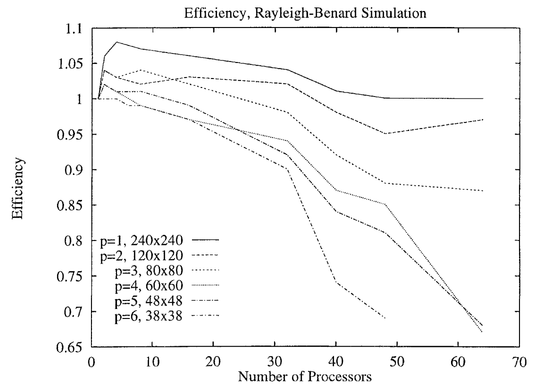


Figure 11. Speedup, Cray T3E, constant problem size

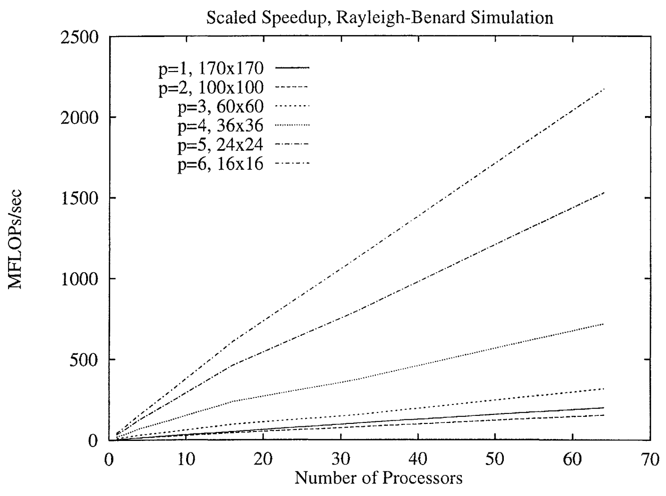


Figure 12. Scaled speedup, Cray T3E, maximum memory

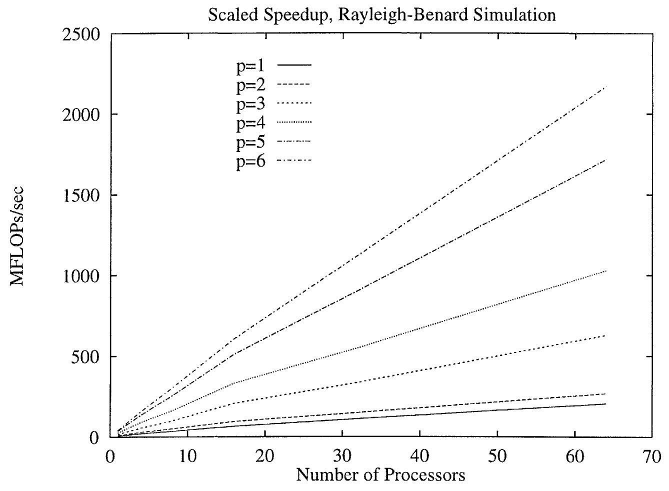


Figure 13.
Scaled speedup, Cray T3E, constant number of elements

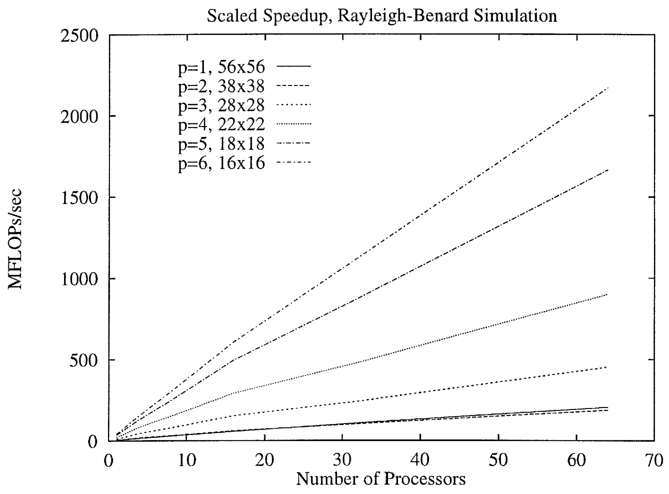


Figure 14.
Scaled speedup, Cray T3E, constant problem size

on the Cray T3E are provided. The parallel performance for different element degree choices are studied and the degradation in parallel performance as a result of the coarse grid dot products is examined.

References

Argyris, J.H., Doltsinis, J. St, Pimenta, P.M. and Wustenburg, H. (1986), "Finite element solution of viscous flow problems", in Gallagher, R.H., Carey, G.F., Oden, J.T. and Zienkiewicz, O.C. (Eds), *Finite Elements in Fluids*, John Wiley & Sons, Vol. 6, p. 107-10.

Babuska, I., Szabo, B.A. and Katz, I.N. (1981), "The *p*-version of the finite element method", *SIAM J. Numer. Anal.*, Vol. 38 No. 3, pp. 515-45.

Barragy, E. and Carey, G.F. (1988), "A parallel element-by-element solution scheme", *Int. Jour. Num. Meth. Eng.*, Vol. 26, pp. 2367-82.

Barragy, E. and Carey, G.F. (1992), "Parallel-vector computations with high-*p* element-by-element methods", *Int. Jour. Comp. Math.*, Vol. 44, pp. 329-39.

Bénard, H. (1900), "Les tourbillons cellulaires dans une nappe liquide", *Rev. Gén. Sci. Pure Appl.*, Vol. 11, pp. 1261-71, 1309-23.

Carey, G.F. and Barragy, E. (1989), "Basis function selection and preconditioning high degree finite element and spectral methods", *BIT*, Vol. 29, p. 794-804.

Carey, G.F. and Oden, J.T. (1986), *Finite Elements: Fluid Mechanics*, Prentice-Hall, Englewood Cliffs, NJ, p. 294.

Carpenter, B.M. and Homsy, G.M. (1989), "Combined buoyant-thermocapillary flow in a cavity", *J. Fluid Mech*, Vol. 207, pp. 121-32.

Davis, M.B. (1996), "Parallel multilevel solution of interatively decoupled transport problems", PhD thesis, University of Texas at Austin, Austin, TX.

De Vahl Davis, G. (1968), "Laminar natural convection in an enclosed rectangular cavity", *Int. J. Heat Mass Transfer*, Vol. 11, pp. 1675-93.

De Vahl Davis, G. (1983), "Natural convection in a square cavity: a benchmark numerical solution", *IJNMF*, Vol. 3, pp. 249-64.

De Vahl Davis, G. and Jones, I.P. (1983), "Natural convection in a square cavity: a comparison exercise", *IJNMF*, Vol. 3, pp. 227-48.

Van Hook, S.J. (1996), "NASA Grant Report: long-wavelength instability in surface-tension-driven Bénard convection", *NASA Grant Report*, Vol. 1, pp. 2679-90.

Van Hook, S.J., Schatz, M.F., Swift, J.B., McCormack, W.D. and Swinney, H.L. (1997), "Long-wavelength surface-tension-driven Bénard convection: experiment and theory", *J. Fluid Mech.*, Vol. 75 No. 45.

McLay, R. and Carey, G.F. (1989), "Coupled heat transfer and viscous flow, and magnetic effects in weld pool analysis", *International Journal for Numerical Methods in Fluids*, Vol. 9, pp. 713-30.

Ronquist, E.M. and Patera, A.T. (1987), "Spectral element multigrid. I. Formulation and numerical results", *J. Sci. Comp.*, Vol. 2 No. 4, pp. 389-406.

Zebib, A., Homsy, G.M. and Meiburg, E. (1985), "High Marangoni number convection in a square cavity", *Phys. Fluids*, Vol. 28 No. 12, pp. 3467-76.