

The AGEB Algorithm for Solving the Heat Equation in Three Space Dimensions and Its Parallelization Using PVM

Mohd Salleh Sahimi¹, Norma Alias², and Elankovan Sundararajan²

¹ Department of Engineering Sciences and Mathematics,
Universiti Tenaga Nasional, 43009 Kajang, Malaysia

Sallehs@uniten.edu.my,

² Department of Industrial Computing,
Universiti Kebangsaan Malaysia, 43600 UKM, Malaysia

norm_ally@hotmail.com

elan@ftsm.ukm.my

Abstract. In this paper, a new algorithm in the class of the AGE method based on the Brian variant (AGEB) of the ADI is developed to solve the heat equation in 3 space dimensions. The method is iterative, convergent, stable and second order accurate with respect to space and time. It is inherently explicit and is therefore well suited for parallel implementation on the PVM where data decomposition is run asynchronously and concurrently at every time level. Its performance is assessed in terms of speed-up, efficiency and effectiveness.

1 Introduction

The ADI method deals with two-dimensional parabolic (and elliptic) problems. Since the method has no analogue for the one-dimensional case, in [1] the alternating group explicit method which offers its users many advantages was developed. It is shown to be extremely powerful and flexible. It employs the fractional splitting strategy of Yanenko [2] which is applied alternately at each intermediate time step on tridiagonal systems of difference schemes. Its implementation was then extended to two space dimensions [3]. In this paper, we present the formulation of AGEB for the solution of the heat equation in three space dimensions and then describe its parallel implementation on the PVM on a model problem.

2 Formulation of the AGEB Method

Consider the following heat equation,

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} + h(x, y, z, t), \quad (x, y, z, t) \in R \times (0, T] \quad (1)$$

subject to the initial condition

$$U(x, y, z, 0) = F(x, y, z), \quad (x, y, z, t) \in R \times \{0\} , \tag{2}$$

and the boundary conditions

$$U(x, y, z, t) = G(x, y, z, t), \quad (x, y, z, t) \in \partial R \times (0, T] , \tag{3}$$

where R is the cube $0 < x, y, z < 1$ and ∂R its boundary. A generalised approximation to (1) at the point $(x_i, y_j, z_k, t_{N+1/2})$ is given by (with $0 \leq \theta \leq 1$),

$$\frac{u_{i,j,k}^{[N+1]} - u_{i,j,k}^{[N]}}{\Delta t} = \frac{1}{(\Delta x)^2} \left\{ \theta(\delta_x^2 + \delta_y^2 + \delta_z^2)u_{i,j,k}^{[N+1]} + (1 - \theta)(\delta_x^2 + \delta_y^2 + \delta_z^2)u_{i,j,k}^{[N]} \right\} + h_{i,j,k}^{[N+1/2]}, \quad i, j, k = 1, 2, \dots, m , \tag{4}$$

leading to the seven-point formula,

$$\begin{aligned} -\lambda\theta u_{i-1,j,k}^{[N+1]} + (1 + 6\lambda\theta)u_{i,j,k}^{[N+1]} - \lambda\theta u_{i+1,j,k}^{[N+1]} - \lambda\theta u_{i,j-1,k}^{[N+1]} - \lambda\theta u_{i,j,k-1}^{[N+1]} \\ - \lambda\theta u_{i,j+1,k}^{[N+1]} - \lambda\theta u_{i,j,k+1}^{[N+1]} = \lambda(1 - \theta)u_{i-1,j,k}^{[N]} + (1 - 6\lambda\theta)(1 - \theta)u_{i,j,k}^{[N]} \\ + \lambda(1 - \theta)u_{i+1,j,k}^{[N]} + \lambda(1 - \theta)u_{i,j-1,k}^{[N]} + \lambda(1 - \theta)u_{i,j,k-1}^{[N]} + \lambda(1 - \theta)u_{i,j+1,k}^{[N]} \\ + \lambda(1 - \theta)u_{i,j,k+1}^{[N]} + \Delta t h_{i,j,k}^{[N+1/2]} . \end{aligned} \tag{5}$$

By considering our approximations as sweeps parallel to the xy -plane of the cube R , (5) can be written in matrix form as,

$$\mathbf{A}\mathbf{u}_{[xy]}^{[N+1]} = \mathbf{f} . \tag{6}$$

By splitting \mathbf{A} into the sum of its constituent symmetric and positive definite matrices $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, \mathbf{G}_4, \mathbf{G}_5$ and \mathbf{G}_6 we have

$$\mathbf{A} = \mathbf{G}_1 + \mathbf{G}_2 + \mathbf{G}_3 + \mathbf{G}_4 + \mathbf{G}_5 + \mathbf{G}_6 , \tag{7}$$

with these matrices taking block banded structures as shown in Fig. 1.

Using the well-known fact of the parabolic-elliptic correspondence and employing the fractional splitting of Brian [4], the AGEB scheme takes the form,

$$\begin{aligned} (r\mathbf{I} + \mathbf{G}_1)\mathbf{u}_{[xy]}^{(n+1/7)} &= (r\mathbf{I} - (\mathbf{G}_1 + \mathbf{G}_2 + \mathbf{G}_3 + \mathbf{G}_4 + \mathbf{G}_5 + \mathbf{G}_6))\mathbf{u}_{[xy]}^{(n)} + \mathbf{f} \\ &= ((r\mathbf{I} + \mathbf{G}_1) - \mathbf{A})\mathbf{u}_{[xy]}^{(n)} + \mathbf{f} \\ (r\mathbf{I} + \mathbf{G}_2)\mathbf{u}_{[xy]}^{(n+2/7)} &= r\mathbf{u}_{[xy]}^{(n+1/7)} + \mathbf{G}_2\mathbf{u}_{[xy]}^{(n)} \\ (r\mathbf{I} + \mathbf{G}_3)\mathbf{u}_{[xy]}^{(n+3/7)} &= r\mathbf{u}_{[xy]}^{(n+2/7)} + \mathbf{G}_3\mathbf{u}_{[xy]}^{(n)} \\ (r\mathbf{I} + \mathbf{G}_4)\mathbf{u}_{[xy]}^{(n+4/7)} &= r\mathbf{u}_{[xy]}^{(n+3/7)} + \mathbf{G}_4\mathbf{u}_{[xy]}^{(n)} \\ (r\mathbf{I} + \mathbf{G}_5)\mathbf{u}_{[xy]}^{(n+5/7)} &= r\mathbf{u}_{[xy]}^{(n+4/7)} + \mathbf{G}_5\mathbf{u}_{[xy]}^{(n)} \\ (r\mathbf{I} + \mathbf{G}_6)\mathbf{u}_{[xy]}^{(n+6/7)} &= r\mathbf{u}_{[xy]}^{(n+5/7)} + \mathbf{G}_6\mathbf{u}_{[xy]}^{(n)} \\ \mathbf{u}^{(n+1)} &= \mathbf{u}^{(n)} + 2(\mathbf{u}^{(n+6/7)} - \mathbf{u}^{(n)}) . \end{aligned} \tag{8}$$

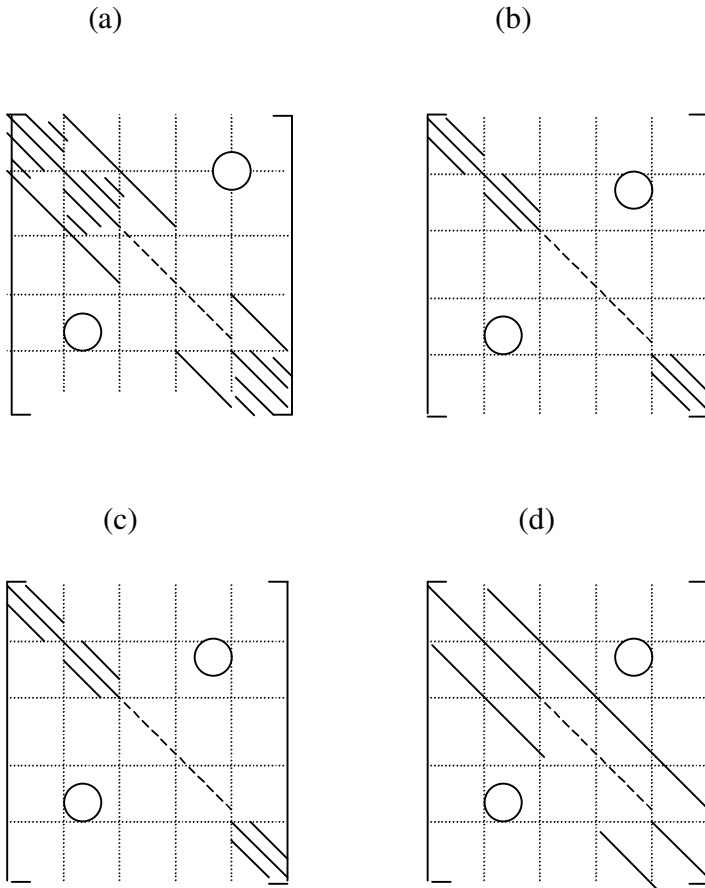


Fig. 1. (a) \mathbf{A} , (b) $\mathbf{G}_1 + \mathbf{G}_2$, (c) $\mathbf{G}_3 + \mathbf{G}_4$, and (d) $\mathbf{G}_5 + \mathbf{G}_6$. Note that $\text{diag}(\mathbf{G}_1 + \mathbf{G}_2) = \text{diag}(\mathbf{A})/3$, $\text{diag}(\mathbf{G}_3 + \mathbf{G}_4) = \text{diag}(\mathbf{A})/3$, $\text{diag}(\mathbf{G}_5 + \mathbf{G}_6) = \text{diag}(\mathbf{A})/3$. All are of order $(m^3 \times m^3)$.

The approximations at the first and the second intermediate levels are computed directly by inverting $(r\mathbf{I}+\mathbf{G}_1)$ and $(r\mathbf{I}+\mathbf{G}_2)$. The computational formulae for the third and fourth intermediate levels are derived by taking our approximations as sweeps parallel to the yz -plane. Here, the u values are evaluated at points lying on planes which are parallel to the yz -plane and on each of these planes, the points are reordered row-wise (parallel to the y -axis). Finally, by considering our approximations as sweeps parallel to the xz -plane followed by a reordering of the points column-wise (parallel to the z -axis) enable us to determine the AGEB equations at the fifth and sixth intermediate levels. Note that all solutions at each iterate are generated rather than stored. Hence the actual inverse is not used by the algorithm. The AGEB sweeps involve tridiagonal systems which in turn entails at each stage the solution of (2×2) block systems. The iterative procedure is continued until convergence is reached.

3 Parallel Implementation of the AGEB Algorithm

One must ensure that an effective parallel implementation of the algorithm leads to a substantial increase in the computational count per data exchange, major reduction in synchronisation frequency and subsequent decrease in communication sessions.

A typical parallel implementation involves the assignment of a block of grids to each task to a surface so that each task only communicates with its limited nearest neighbours. Only the top, bottom, left and right surfaces of the block need to be exchanged between neighbouring tasks. As an example, Fig. 2 illustrates the pattern of communication with 4 tasks ($p = 4$) on a linear system of order $m = 32$.

It is also important to maintain load balancing in the distribution of m grids to tasks P_1, P_2, \dots, P_p . The data decomposition of the AGEB algorithm is run asynchronously and simultaneously at every time level, where each task is allocated m/p grids. It proceeds for every task at each time level until the local error and the approximate solution are computed at the last time level. These tasks then send the local errors to the master, which in turn processes the global error.

On the PVM, parallel implementation of AGEB is based on one master and many tasks. The master program is responsible in constructing the m grid sizes, computing the initial values, portioning the grid into blocks of surface, assigning these blocks to the p task modules, distributing the task to different processors and receiving local errors from the tasks. Each block that is assigned to a task module is composed of m/p blocks.

A task process starts computations after it receives a work assignment. A task module q ($q < P$) performs the AGEB iterations on the grid points of the assigned block which is composed of surfaces with indices between

$$SUR_q(\text{start}) = \frac{m(q-1)}{p} \quad \text{and} \quad SUR_q(\text{end}) = \frac{mq}{p} - 1 \quad ,$$

where SUR refers to the surface. The task q will transmit

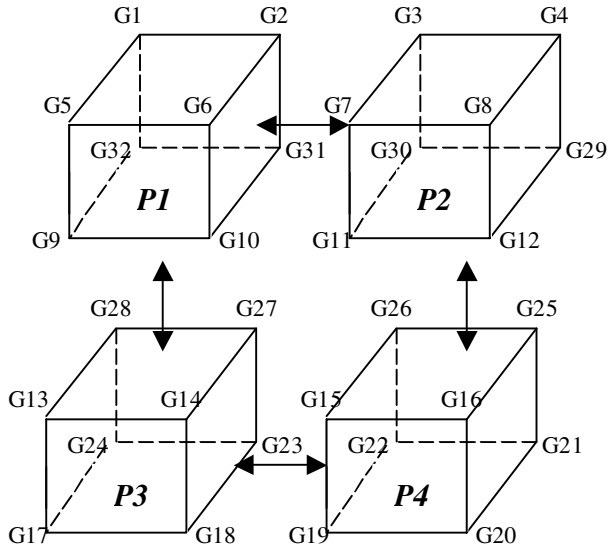


Fig. 2. Communication of data exchange between 4 tasks and 32 grids

- i. to its upper neighbour (task $q - 1$), $SUR_q(\text{start})$ and receive from it

$$SUR_q(\text{start}) - 1 = SUR_{q-1}(\text{end})$$

- ii. to its lower neighbours (task $q + 1$), $SUR_q(\text{end})$ and receive from it

$$SUR_q(\text{end}) + 1 = SUR_{q+1}(\text{start}) .$$

Since multiple copies of the same task code run simultaneously, the tasks will exchange data with their neighbours at different times. At this point, the barrier function is called by the PVM library routine for synchronisation. The tasks will repeat the above procedure, until the local convergence criterion is met. The definition of the residual computed in the task q is as follows,

$$r[i][j][k] = \max \left\{ \left| u_{i,j,k}^{[N+1]} - u_{i,j,k}^{[N]} \right|, \quad (i, j, k) \in q \right\} .$$

The tasks will return all its local errors to the master module. After receiving the locally converged blocks from the tasks, the master module checks whether the global convergence is satisfied,

$$r[i][j][k] \leq \epsilon, \quad \forall i, j, k \in [0, m] ,$$

where $r[i][j][k] = \max \left\{ \left| u_{i,j,k}^{[N+1]} - u_{i,j,k}^{[N]} \right|, \quad (i, j, k) \in q \right\}$.

This procedure is repeated and the system terminates if a global convergence is reached. Otherwise the master repartitions the blocks and reassigns them to the p tasks.

4 Numerical Results and Discussion

The following problem is solved using the AGEB algorithm,

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} + h(x, y, z, t), \quad 0 \leq x, y, z, t \leq 1, \quad t \geq 0 ,$$

with

$$h(x, y, z, t) = (3\pi^2 - 1)e^{-t} \sin \pi x \sin \pi y \sin \pi z ,$$

subject to the initial condition,

$$U(x, y, z, 0) = \sin \pi x \sin \pi y \sin \pi z ,$$

and boundary conditions

$$U(0, y, z, t) = U(1, y, z, t) = U(x, 0, z, t) = U(x, y, 0, t) = U(x, y, 1, t) = 0 ,$$

Our PVM platform consists of a cluster of 9 SUN Sparc Classic II workstations each running at a speed of 70 MHz and configured with 32 Mbytes of system memory. The workstations are connected by a 100/10 base T/3com Hub, Super stack II via a Baseline Dual Speed Hub 12 port network. The PVM software only resides in the master processor.

As measures of performance of our algorithm, the following definitions are used:

$$\text{Speed-up ratio} \quad S_p = T_1/T_p \tag{9}$$

$$\text{Efficiency} \quad E_p = S_p/p \tag{10}$$

$$\text{Effectiveness} \quad F_p = S_p/C_p \tag{11}$$

where $C_p = pT_p$, T_1 the execution time on a serial machine and T_p the computing time on a parallel machine with p processors.

Figure 3 shows the speedup factor plotted against the number of processors p . It indicates that high speedups are obtained only for large values of m . An impressive gain in the speedups can be expected for even larger problems. Possible reasons for this are: the relatively high communication time for passing data between the master and slaves compared with the computation time of the AGEB method; wasteful idle time at a barrier synchronisation point before proceeding with the next iteration; the contribution from parallel processing which is less than the number of processors and the contribution from the distributed memory hierarchy, which reduces the time consuming access to virtual memory for large linear equations. However, we are constrained by the relatively small

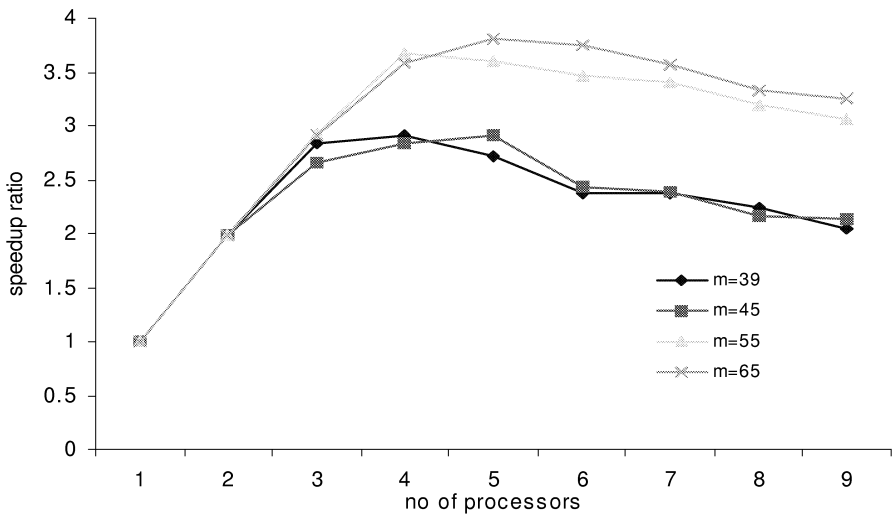


Fig. 3. Speedup ratio vs Number of processors

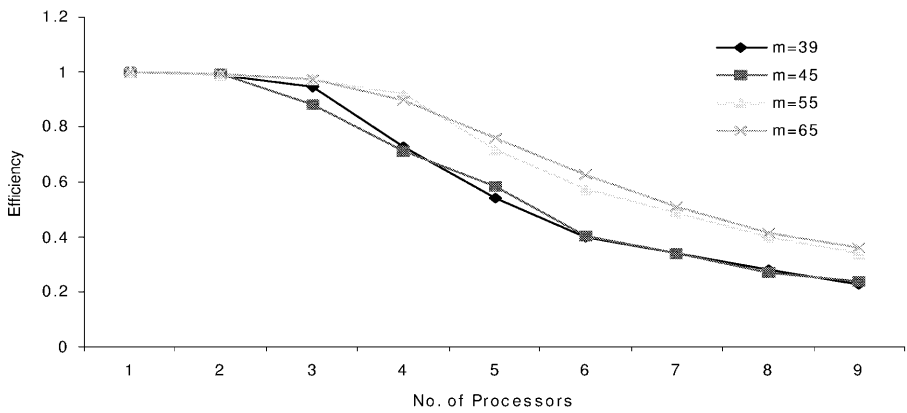


Fig. 4. Efficiency vs Number of processors

memory of the system. The speedup starts to degrade when more than 5 processors are used. It must also be noted that the timing for message passing is relatively slow for inter-processor communication using Ethernet Card 10-based network.

As expected, Fig. 4 depicts that efficiency decreases with increasing p . It deteriorates when more than 3 processors are used. This deterioration is a result of poor load balancing attained when only the small block is spread across more than 3 processors. Hence the high overhead cost due to synchronisation. From (9)–(11),

$$F_p = S_p / (pT_p) = E_p / T_p = E_p S_p / T_1$$

which clearly shows that F_p is a measure both of speedup and efficiency. Therefore, a parallel algorithm is said to be effective if it maximises F_p and hence $F_p T_1 (= S_p E_p)$. From Fig. 5, we see that $F_p T_1$ has a maximum when $p = 3$ for $m = 39$ which indicates that $p = 3$ is the optimal choice of number of processors. Similarly, we can infer from $m = 45$ and 55 that the optimal choice of number of processors is given respectively by $p = 3$ and $p = 4$ allowing for inconsistencies due to load balancing.

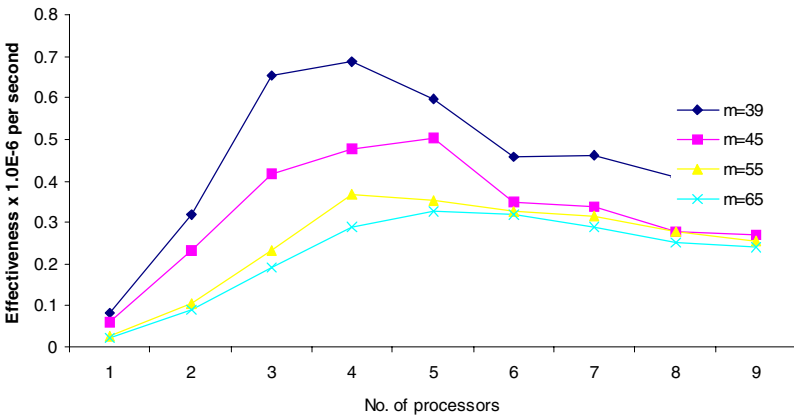


Fig. 5. Effectiveness vs Number of processors

As a general conclusion, the inherently explicit, stable and highly accurate algorithm is found to be well suited for parallel implementation on the PVM where data decomposition is run asynchronously and concurrently at every time level.

5 Conclusions

The stable and highly accurate AGEB algorithm is found to be well suited for parallel implementation on the PVM where data decomposition is run asynchronously and concurrently at every time level. The AGEB sweeps involve tridiagonal systems which require the solution of (2×2) block systems. Existing parallel strategies could not be fully exploited to solve such systems. The AGEB algorithm, however, is inherently explicit and the domain decomposition strategy is efficiently utilised.

The PVM is favoured to MPI as the computing platform because of the flexibility of the former to communicate across architectural boundaries. This is especially relevant since this research project was initially undertaken with the view of utilising readily available heterogeneous cluster of computing resources in our laboratories. Furthermore, we have already noted that higher speedups could be expected for larger problems. Coupled with this is the advantage that the PVM has on its fault tolerant features. Hence, for large real field problems which we hope to solve in the immediate future these features become more important as the cluster gets larger.

Acknowledgments

The authors wish to express their gratitude and indebtedness to the Universiti Kebangsaan Malaysia, Universiti Tenaga Nasional and the Malaysian government for providing the moral and financial support under the IRPA grant for the successful completion of this project.

References

1. Evans, D.J., Sahimi, M.S.: The Alternating Group Explicit Iterative Method (AGE) to Solve Parabolic and Hyperbolic Partial Differential Equations. In: Tien, C.L., Chawla, T.C. (eds.): *Annual Review of Numerical Fluid Mechanics and Heat Transfer*, Vol. 2. Hemisphere Publication Corporation, New York Washington Philadelphia London (1989)
2. Yanenko, N.N.: *The Method of Fractional Steps*. Springer-Verlag, Berlin Heidelberg New York (1971)
3. Evans, D.J., Sahimi, M.S.: The Alternating Group Explicit (AGE) Iterative Method for Solving Parabolic Equations I: 2-Dimensional Problems. *Intern. J. Computer Math.* 24 (1988) 311-341
4. Peaceman, D.W.: *Fundamentals of Numerical Reservoir Simulation*. Elsevier Scientific Publishing Company, Amsterdam Oxford New York (1977)
5. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V., *PVM: Parallel Virtual Machine & User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, Mass (1994)
6. Hwang, K. and Xu, Z., *Scalable Parallel Computing: Technology, Architecture, Programming*. McGraw Hill, (1998)
7. Lewis, T.G. and EL-Rewini, H., *Distributed and Parallel Computing*, Manning Publication, USA, (1998)

8. Quinn, .M.J., Parallel Computing Theory and Practice, McGraw Hill, (1994)
9. Wilkinson,.B. and Allen, M., Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall,Upper Saddle River, New Jersey 07458 (1999)