Neural, Parallel, and Scientific Computations 18 (2010) 253-268

PARALLEL MONOTONE DOMAIN DECOMPOSITION ALGORITHMS FOR NONLINEAR SINGULARLY PERTURBED REACTION-DIFFUSION PROBLEMS OF PARABOLIC TYPE

MATTHEW HARDY AND IGOR BOGLAEV

Mathematical Sciences Institute, Australian National University
Canberra, Australia
Institute of Fundamental Sciences, Massey University
Palmerston North, New Zealand

ABSTRACT. Recently, a monotone iterative domain decomposition algorithm has been proposed for nonlinear singularly perturbed reaction-diffusion problems of parabolic type. This paper describes a parallel implementation of the algorithm on a distributed memory cluster. Interprocess communication is effected by means of the MPI message passing library. For various domain decompositions, we give the execution time and parallel speedup on up to 16 processors. The parallel scale-up of the algorithm improves as the number of mesh points is increased.

Keywords: parabolic reaction-diffusion problem; boundary layers; implicit scheme; monotone iterative method; domain decomposition; parallel computing

1. INTRODUCTION

We are interested in the nonlinear reaction-diffusion problem of parabolic type

$$-\mu^2 \left(u_{xx} + u_{yy}\right) + u_t = -f(x, y, t, u), \tag{1}$$

$$(x, y, t) \in v = \omega \times (0, t_f], \quad \omega = \{0 < x < 1, 0 < y < 1\},$$

$$c^* \geq f_u \geq 0, \quad (x, y, t, u) \in \overline{v} \times (-\infty, \infty), \quad (f_u \equiv \partial f / \partial u),$$

where μ is a small positive parameter and c^* is constant. The initial-boundary conditions are defined by

$$u(x, y, 0) = u^0(x, y), \ (x, y) \in \overline{\omega}, \quad u = g, \ (x, y, t) \in \partial\omega \times (0, t_f],$$

where $\partial\omega$ is the boundary of ω. For $\mu \ll 1$, problem (1) is singularly perturbed and characterized by boundary layers of width $\mathcal{O}(\mu |\ln \mu|)$ at $\partial\omega$. Note that the assumption $f_u \geq 0$ can always be obtained by a change of variables like $v = \exp(\lambda t)u$, where λ is a constant.

On each time level, discrete approximation of (1) leads to an algebraic system of nonlinear difference equations whose solution converges with mesh refinement to

Received March 15, 2010

that of the continuous problem. The algebraic system is typically solved by Newton's method, or some other iterative technique. One drawback of Newton's method is its sensitivity to the initial guess. In contrast, the method of upper and lower solutions generates a monotonically convergent sequence from any one of a wide class of initial iterates. Indeed, as shown in [1], the initial iterate may be constructed using only the boundary conditions. No knowledge of the solution is necessary to implement the algorithm.

The beowulf cluster has brought high-performance computing within reach of academe and fostered renewed interest in alternating Schwarz-type iterative algorithms. In [2] the spatial domain is partitioned into nonoverlapping boxes on which the monotone iterative method is applied at each time step. At horizontal and vertical subdomain boundaries, interfacial subdomains are introduced and corresponding linear problems generate boundary Dirichlet data for the nonoverlapping subdomains. As shown theoretically and confirmed by serial computations [2], the algorithm retains global monotonicity under such decomposition. In [5], we implemented on a parallel cluster the algorithm from [2] for an elliptic (time-independent) problem. The present paper describes a parallel implementation of the algorithm from [2] for the parabolic reaction-diffusion problem (1).

In Section 2 we define the piecewise uniform mesh on which the central difference implicit scheme converges μ-uniformly to the solution of (1). In Section 3 we describe the domain decomposition from [2] and the associated monotone iterative algorithm. We also state the main results from the convergence analysis. Section 4 discusses our parallel implementation of the algorithm. In Section 5 we give the results of numerical experiments for a model problem. For various domain decompositions we give the convergence iteration counts and execution times on up to 16 processors. Since domain decomposition enhances the algorithm's serial execution, we define parallel speedup in terms of the optimal decomposition for a given number of processors. The parallel scale-up of the algorithm improves as the mesh size is increased.

2. DISCRETE APPROXIMATION

On $\overline{v}$ introduce the mesh $\overline{\omega}^h \times \overline{\omega}^\tau$, $\overline{\omega}^h = \overline{\omega}^{hx} \times \overline{\omega}^{hy}$ defined as

$$\overline{\omega}^{hx} = \{x_i,\ 0 \le i \le N_x;\ x_0 = 0,\ x_{N_x} = 1;\ h_{xi} = x_{i+1} - x_i\},$$

$$\overline{\omega}^{hy} = \{y_j,\ 0 \le j \le N_y;\ y_0 = 0,\ y_{N_y} = 1;\ h_{yj} = y_{j+1} - y_j\},$$

$$\overline{\omega}^{\tau} = \{t_k = k\tau,\ 0 \le k \le N_\tau,\ N_\tau \tau = t_f\}.$$

Define $\mathcal{D}_x^2 U$ and $\mathcal{D}_y^2 U$, the central difference approximations to the second x- and y-derivatives, respectively,

$$\mathcal{D}_x^2 U_{ij}^k = (\hbar_{xi})^{-1} \left[\left(U_{i+1,j}^k - U_{ij}^k\right) (h_{xi})^{-1} - \left(U_{ij}^k - U_{i-1,j}^k\right) (h_{x,i-1})^{-1} \right],$$

$$\mathcal{D}_y^2 U_{ij}^k = (\hbar_{yj})^{-1} \left[\left(U_{i,j+1}^k - U_{ij}^k \right) (h_{yj})^{-1} - \left(U_{ij}^k - U_{i,j-1}^k \right) (h_{y,j-1})^{-1} \right],$$
$$\hbar_{xi} = 2^{-1} (h_{x,i-1} + h_{xi}), \quad \hbar_{yj} = 2^{-1} (h_{y,j-1} + h_{yj}),$$

where $U_{ij}^k \equiv U(x_i, y_j, t_k)$.

We seek a mesh function U on $\overline{\omega}^h \times \overline{\omega}^\tau$ satisfying the implicit difference scheme

$$\mathcal{L}U(P,t) = \frac{1}{\tau} U(P, t-\tau) - f(P,t,U), \quad (P,t) \in \omega^h \times \omega^\tau, \tag{2}$$
$$U(P,0) = u^0(P),\ P \in \overline{\omega}^h, \quad U(P,t) = g(P,t),\ (P,t) \in \partial\omega^h \times \omega^\tau,$$

where $\mathcal{L}U = \left[-\mu^2 \left(\mathcal{D}_x^2 + \mathcal{D}_y^2 \right) + \tau^{-1} \right] U$.

The meshes $\overline{\omega}^{hx}$ and $\overline{\omega}^{hy}$ are defined in the manner of [6] and are referred to as Shishkin meshes. For an arbitrary positive constant m, boundary layer thicknesses σ_x and σ_y are chosen as

$$\sigma_x = \min \{0.25, m\mu \ln N_x\}, \qquad \sigma_y = \min \{0.25, m\mu \ln N_y\}, \tag{3}$$

and mesh spacings $h_{x\mu}$, h_x, $h_{y\mu}$ and h_y are defined by

$$h_{x\mu} = \frac{4\sigma_x}{N_x}, \quad h_x = \frac{2(1-2\sigma_x)}{N_x}, \quad h_{y\mu} = \frac{4\sigma_y}{N_y}, \quad h_y = \frac{2(1-2\sigma_y)}{N_y}. \tag{4}$$

The mesh $\overline{\omega}^{hx}$ is constructed thus: in each of the subintervals $[0, \sigma_x]$ and $[1-\sigma_x, 1]$ the fine mesh spacing is $h_{x\mu}$ while in the interval $[\sigma_x, 1-\sigma_x]$ the coarse mesh spacing is h_x. The mesh $\overline{\omega}^{hy}$ is defined similarly. The difference scheme (2) on the piecewise uniform mesh (3), (4) converges μ-uniformly to the solution of the continuous problem (1);

$$\max_{t \in \overline{\omega}^\tau} \|U(t) - u(t)\|_{\overline{\omega}^h} \le C \left(d\left(N^{-1}\right) + \tau \right), \quad N = \min \{N_x, N_y\},$$
$$\|U(t) - u(t)\|_{\overline{\omega}^h} \equiv \max_{P \in \overline{\omega}^h} |U(P,t) - u(P,t)|,$$

where d is near order 2 in N^{-1}, and constant C is independent of μ, N and τ (see [3] for details).

3. BOX-DOMAIN DECOMPOSITION ALGORITHM

For solving the nonlinear difference scheme (2), we now describe the box-domain decomposition algorithm from [2]. Consider a decomposition of ω into $M \times L$ nonoverlapping main subdomains $\omega_{m,l}$, $m = 1, \ldots, M$, $l = 1, \ldots, L$:

$$\omega_{m,l} = (x_{m-1}, x_m) \times (y_{l-1}, y_l), \quad x_0 = 0, x_M = 1,\ y_0 = 0, y_L = 1.$$

Then introduce vertical interfacial subdomains θ_m, $m = 1, \ldots, M-1$:

$$\theta_m = (x_m^b, x_m^e) \times \omega^y, \quad x_m^b < x_m < x_m^e, \quad \theta_{m-1} \cap \theta_m = \emptyset,$$

and horizontal interfacial subdomains ϑ_l, $l = 1, \ldots, L-1$:

$$\vartheta_l = \omega^x \times (y_l^b, y_l^e), \quad y_l^b < y_l < y_l^e, \quad \vartheta_{l-1} \cap \vartheta_l = \emptyset.$$

With the global mesh $\overline{\omega}^h = \overline{\omega}^{hx} \times \overline{\omega}^{hy}$, we also require that

$$\{x_m^{b,e}, x_m\} \subset \omega^{hx},\ m = 1, \dots, M-1, \quad \{y_l^{b,e}, y_l\} \subset \omega^{hy},\ l = 1, \dots, L-1.$$

On each time level $t \in \omega^\tau$, the algorithm from [2] generates n_* iterates $V^{(n)}(P,t),\ (P,t) \in \overline{\omega}^h \times \omega^\tau$ as follows.

Step 0. On the whole mesh $\overline{\omega}^h$ choose an initial mesh function $V^{(0)}(P,t)$ satisfying the boundary conditions, $V^{(0)}(P,t) = g(P,t),\ P \in \partial\omega^h$.

For $n = 1$ to n_* do Steps 1 to 4:

Step 1. For each main subdomain $\overline{\omega}_{m,l}^h$, solve the linear difference problem

$$(\mathcal{L} + c^*) Z_{m,l}^{(n)}(P,t) = -G^{(n-1)}(P,t), \quad P \in \omega_{m,l}^h = \omega_{m,l} \cap \omega^h,$$

$$G^{(n-1)}(P,t) = \mathcal{L}V^{(n-1)}(P,t) + f\left(P,t,V^{(n-1)}\right) - \tau^{-1}V(P,t-\tau),$$

with $Z_{m,l}^{(n)}(\partial\omega_{m,l}^h, t) = 0$.

Step 2. For each vertical interfacial subdomain $\overline{\theta}_m^h$, solve the linear difference problem

$$(\mathcal{L} + c^*) Z_m^{(n)}(P,t) = -G^{(n-1)}(P,t), \quad P \in \theta_m^h = \theta_m \cap \omega^h,$$

with $Z_m^{(n)}(\partial\theta_m^h, t)$ defined by the mesh functions computed in Step 1.

Step 3. For each horizontal interfacial subdomain $\overline{\vartheta}_l^h$, solve the linear difference problem

$$(\mathcal{L} + c^*) \widetilde{Z}_l^{(n)}(P,t) = -G^{(n-1)}(P,t), \quad P \in \vartheta_l^h = \vartheta_l \cap \omega^h,$$

with $\widetilde{Z}_l^{(n)}(\partial\vartheta_l^h, t)$ defined by the mesh functions computed in Step 2 where possible, or the mesh functions from Step 1 otherwise.

Step 4. Piece together the mesh functions from Steps 1-3 to form the new global iterate $V^{(n)}$:

$$V^{(n)}(P,t) = \begin{cases} V^{(n-1)}(P,t) + \widetilde{Z}_l^{(n)}(P,t), & P \in \overline{\vartheta}_l^h, \\ V^{(n-1)}(P,t) + Z_m^{(n)}(P,t), & P \in \overline{\theta}_m^h \setminus \left\{\bigcup_{l=1}^{L-1} \overline{\vartheta}_l^h\right\}, \\ V^{(n-1)}(P,t) + Z_{m,l}^{(n)}(P,t), & P \in \overline{\omega}_{m,l}^h \setminus \left\{\bigcup_{m=1}^{M-1} \overline{\theta}_m^h \bigcup_{l=1}^{L-1} \overline{\vartheta}_l^h\right\}. \end{cases}$$

Step 5. Set up $V(P,t) = V^{(n_*)}(P,t), \quad P \in \overline{\omega}^h$.

Convergence of the algorithm. We say that on a time level $t \in \omega^\tau$, $\overline{V}(P,t)$ is an upper solution with respect to a given function $V(P,t-\tau)$ if it satisfies

$$\mathcal{L}\overline{V}(P,t) + f\left(P,t,\overline{V}\right) - \tau^{-1}V(P,t-\tau) \geq 0, \quad P \in \omega^h,$$

$$\overline{V}(P,t) = g(P,t),\ P \in \partial\omega^h.$$

Similarly, $\underline{V}(P,t)$ is called a lower solution with respect to $V(P,t-\tau)$ if it satisfies the reversed inequality and the boundary condition.

On each time level $t \in \omega^\tau$, we have the following convergence property of the algorithm.

Theorem 1. *Let $V(P,t-\tau)$ be given and $\overline{V}^{(0)}(P,t)$ and $\underline{V}^{(0)}(P,t)$ be upper and lower solutions with respect to $V(P,t-\tau)$. From $\overline{V}^{(0)}$ and $\underline{V}^{(0)}$, the algorithm respectively generates sequences $\{\overline{V}^{(n)}\}$ and $\{\underline{V}^{(n)}\}$ which converge monotonically from above and below to the unique solution $V^*(P,t)$ of the problem*

$$\mathcal{L}V(P,t) + f\,(P,t,V) - \tau^{-1}V(P,t-\tau) = 0, \quad P \in \omega^h,$$

$$V(P,t) = g(P,t),\ P \in \partial\omega^h.$$

That is, at each mesh point $P \in \omega^h$, we have for $n = 0,1,2,\ldots$

$$\underline{V}^{(n)}(P,t) \le \underline{V}^{(n+1)}(P,t) \le V^*(P,t) \le \overline{V}^{(n+1)}(P,t) \le \overline{V}^{(n)}(P,t).$$

The proof of this result may be found in [2].

Remark 1. Consider the following approach for constructing initial upper and lower solutions $\overline{V}^{(0)}(P,t)$ and $\underline{V}^{(0)}(P,t)$. Let a mesh function $R(P,r)$ be defined on $\overline{\omega}^h$ and satisfy the boundary condition $R(P,t) = g(P,t)$ on $\partial\omega^h$. Introduce the difference problems

$$\mathcal{L}Z_\nu^{(0)} = \nu\left|\mathcal{L}R(P,t) + f(P,t,R) - \tau^{-1}V(P,t-\tau)\right|, \quad P \in \omega^h,$$

$$Z_\nu^{(0)}(P,t) = 0, \quad P \in \partial\omega^h, \quad \nu = 1,-1.$$

Then the functions $\overline{V}^{(0)}(P,t) = R(P,t)+Z_1^{(0)}(P,t)$ and $\underline{V}^{(0)}(P,t) = R(P,t)+Z_{-1}^{(0)}(P,t)$ are upper and lower solutions, respectively. The proof of this result can be found in [2]. We mention that much discussions on this subject can be found in [7].

A method for constructing the initial iterate $V^{(0)}$ from an arbitrary mesh function is also given therein.

We now consider the convergence rate of the algorithm. Let h_{xm}^{b+} and h_{xm}^{b-} be the respective mesh step sizes to the right and left of x_m^b. Define h_{xm}^{e+} and h_{xm}^{e-} similarly with respect to x_m^e. Let h_{yl}^{b+} and h_{yl}^{b-} be the respective mesh step sizes above and below y_l^b. Define h_{yl}^{e+} and h_{yl}^{e-} similarly with respect to y_l^e. With averages

$$\hbar_{xm}^{b,e} = \frac{1}{2}(h_{xm}^{b-,e-} + h_{xm}^{b+,e+}), \qquad \hbar_{yl}^{b,e} = \frac{1}{2}(h_{yl}^{b-,e-} + h_{yl}^{b+,e+}),$$

define parameters r^I and r^{II}, respectively associated with the x- and y-decompositions,

$$r^I = \max_{1\le m<M}\left\{\frac{\mu^2}{(c^*+\tau^{-1})\,\hbar_{xm}^b h_{xm}^{b+}}; \frac{\mu^2}{(c^*+\tau^{-1})\,\hbar_{xm}^e h_{xm}^{e-}}\right\},$$

$$r^{II} = \max_{1\le l<L}\left\{\frac{\mu^2}{(c^*+\tau^{-1})\,\hbar_{yl}^b h_{yl}^{b+}}; \frac{\mu^2}{(c^*+\tau^{-1})\,\hbar_{yl}^e h_{yl}^{e-}}\right\}.$$

Theorem 2. *Let $U(P,t)$ be the exact solution of the nonlinear scheme (2), and at each $t \in \omega^\tau$ let $V(P,t) = V^{(n_*)}(P,t)$, $P \in \overline{\omega}^h$, where $V^{(0)}(P,t)$ is an upper or lower solution with respect to $V(P,t-\tau)$. Then we can find constant C independent of τ such that*

$$\max_{t\in\omega^\tau} \|V(t) - U(t)\|_{\overline{\omega}^h} \le C\left(c^* + \tau^{-1}\right)\widetilde{r}^{n_*}, \quad \widetilde{r} = r + r^I + r^{II}, \tag{5}$$

here $r = c^*/(c^* + \tau^{-1})$ is the convergence rate of the undecomposed monotone iterative algorithm ($M = L = 1$).

The proof of this result is given in [2].

Remark 2. The implicit two-level difference scheme (2) is of first order with respect to τ. Since $\widetilde{r} = \mathcal{O}(\tau)$, one may choose $n_* = 2$ to keep the global error of the box-domain decomposition algorithm consistent with the global error of the difference scheme (2).

Remark 3. Consider an $M \times L$ decomposition in which the mesh points are distributed equally among main subdomains. This is called a *balanced* domain decomposition. If a balanced domain decomposition has $M \ge 4$ then θ_1^h and θ_{M-1}^h overlap the boundary layers and r^I is maximal. Similarly, if a balanced decomposition has $L \ge 4$ then r^{II} is maximal. By contrast, if the interfacial subdomains are located wholly outside the boundary layers, ensuring minimality of r^I and r^{II}, then the decomposition is said to be *unbalanced*.

4. PARALLEL IMPLEMENTATION

We have implemented the box-domain decomposition algorithm on *Helix*; the distributed memory Linux cluster at Massey University, New Zealand. This comprises 65 nodes, each equipped with two Athlon MP-2100 processors, 1 GB of RAM and 2 gigabit network interface cards. The nodes are arranged on 7 24-port switches such that any two nodes are connected via at most 3 switches. Inter-processor communication is via the MPI library specification.

Because the mesh is only piecewise uniform, the linear system arising from the difference problem on a given subdomain may be nonsymmetric. Therefore, we solve all linear systems with the restarted GMRES(m) algorithm from [8]. Convergence is accelerated by the point Jacobi preconditioner. As suggested by the convergence estimate (5) and confirmed by the serial computations of balanced and unbalanced domain decompositions in [2], the latter require fewer iterations for convergence. However, in order to balance the computational load of the main subdomains, one would need to divide the larger boundary layer subdomains across several processors. Thus, the algorithmic advantage would be at least partially offset by the need for inter-processor communication during the solution of the main subdomain linear problems.

Therefore, our implementation is at the first level. That is, each subdomain is wholly assigned to a processor and we do not parallelize GMRES. We henceforth consider only balanced domain decompositions.

Suppose we have an $M \times L$ decomposition and P processors. Assuming that $M \geq 4$ and $L \geq 4$, there are $ML/16$ main subdomains in each of the four corners $[0, \sigma_x] \times [0, \sigma_y]$, $[1-\sigma_x, 1] \times [0, \sigma_y]$, $[0, \sigma_x] \times [1-\sigma_y, 1]$ and $[1-\sigma_x, 1] \times [1-\sigma_y, 1]$. Thus, there are $ML/4$ main subdomains in which the respective x- and y-mesh spacings are $h_{x\mu}$ and $h_{y\mu}$. Let us denote this class by F-F (fine-fine). Next, in each of the regions $[\sigma_x, 1-\sigma_x] \times [0, \sigma_y]$, $[\sigma_x, 1-\sigma_x] \times [1-\sigma_y, 1]$, $[0, \sigma_x] \times [\sigma_y, 1-\sigma_y]$ and $[1-\sigma_x, 1] \times [\sigma_y, 1-\sigma_y]$, there are $ML/8$ main subdomains in which the respective x- and y- mesh spacings are either $h_{x\mu}$ and h_y or h_x and $h_{y\mu}$. In the numerical experiments of the next section we take $N_x{=}N_y$ and hence these $ML/2$ subdomains are equivalent. We label this class F-C (fine-coarse). Finally, the region $[\sigma_x, 1-\sigma_x] \times [\sigma_y, 1-\sigma_y]$ is covered by $ML/4$ main subdomains in which the respective x- and y-mesh spacings are h_x and h_y. We denote this class C-C (coarse-coarse).

The linear difference problem on $\omega^h_{m,l}$ may be written in the form

$$\left[-\mu^2(\mathcal{D}_x^2 + \mathcal{D}_y^2) + \tau^{-1} + c^*\right] Z^{(n)}_{m,l} = -G^{(n-1)}(P,t), P \in \omega^h_{m,l}.$$

If the mesh $\omega^h_{m,l}$ is uniform in each direction, then the rows of the coefficient matrix $-\mu^2(\mathcal{D}_x^2+\mathcal{D}_y^2)$ have entries which sum to zero (excepting those rows which correspond to mesh points adjacent to $\partial\omega^h_{m,l}$). Strict diagonal dominance of the coefficient matrix in the linear difference problem is assured by the positive parameters τ^{-1} and c^* and the relative contribution of these parameters increases with the subdomain mesh spacing. Therefore, the GMRES workload is least for the C-C subdomains and greatest for the F-F subdomains.

In order to balance the load of Step 1 of the algorithm, we divide each class equally among the processors. If P divides $ML/4$, each processor is assigned $ML/(4P)$ main subdomains of class F-F, $ML/(2P)$ main subdomains of class F-C and $ML/(4P)$ main subdomains of class C-C. If P does not divide $ML/4$ then load balancing requires second level parallelization and we do not implement the $M \times L$ decomposition on P processors. Similar considerations apply to decompositions in which M or L is equal to one, except that there are at most two classes of main subdomain. The vertical interfacial subdomains are shared as equally as possible among the P processors $\{0, 1, \ldots, P-1\}$: $\overline{\theta}^h_m \mapsto \mod (m-1, P)$. The assignment of horizontal interfacial subdomains is similar: $\overline{\vartheta}^h_l \mapsto \mod (l-1, P)$.

Consider the distribution of an 8×4 decomposition on four processors, shown schematically in Figure 1. During Step 1 of the algorithm, each processor solves over its assigned main subdomains and this workload is balanced. During Step 2, each processor solves over its assigned vertical interfacial subdomains. In contrast to the

other processors which each have two, Processor 3 has only one vertical interfacial subdomain and so is idle for approximately half of Step 2. In Step 3, Processors 0,1,2 each solve over their assigned horizontal interfacial subdomain while Processor 3 remains idle. This idle time results in a loss of computational efficiency.

Another overhead of any parallel implementation is that of inter-processor communication. Before each of the algorithm's three steps, data must be transferred between subdomains. In Figure 1 we have indicated the data which must be transferred from Processor 0 to Processor 1 before the latter can solve over its main subdomains. From the left-most vertical interfacial subdomain on Processor 0, we must transfer the three blocks of data `a`, `b` and `c` to main subdomains on Processor 1. From the horizontal interfacial subdomain on Processor 0, we must transfer the three blocks of data `d`, `e` and `f` to Processor 1. In order to minimize the inter-processor communication, the data blocks `a`–`f` are buffered and sent as one message. This saving in communication comes at the relatively small (local) cost of composing the message on Processor 0 and decomposing the received message on Processor 1.

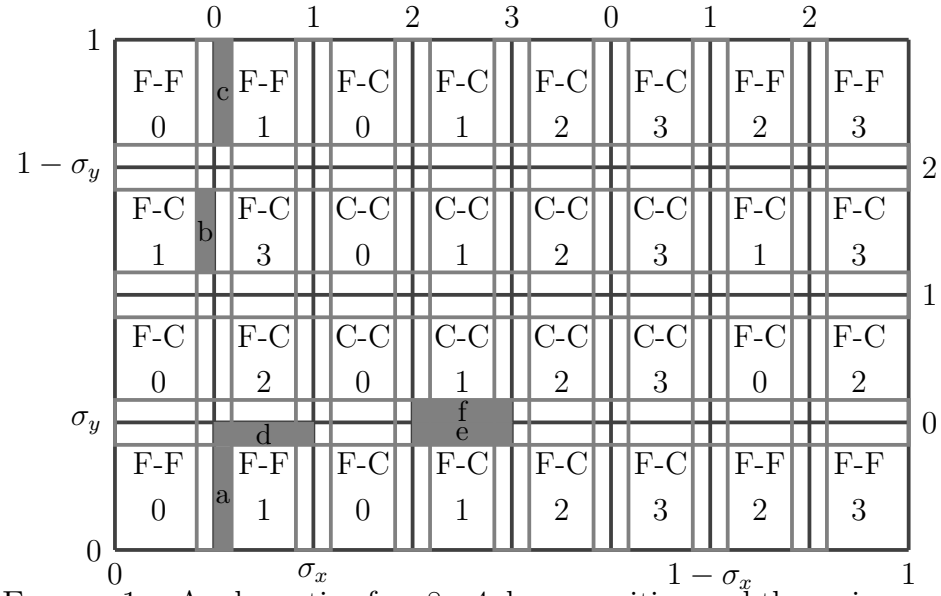

FIGURE 1. A schematic of an 8×4 decomposition and the assignment of the subdomains to four processors $\{0, 1, 2, 3\}$. The mesh spacing classes of the main subdomains are also indicated. The six blocks of data `a`–`f` must be transferred from Processor 0 to Processor 1 before Step 1 of the algorithm.

5. NUMERICAL EXPERIMENTS

We now apply the algorithm to the reaction-diffusion problem

$$-\mu^2(u_{xx} + u_{yy}) + u_t = -\frac{u-4}{5-u},$$
$$(x, y, t) \in \omega \times (0, 1], \quad \omega = \{0 < x < 1, 0 < y < 1\},$$

$$u(\omega,0)=0,\ u(\partial\omega,0)=1,\quad u(\partial\omega,t)=1,\ t\in(0,1],$$

which models the biological Michaelis-Menten process without inhibition [4]. The steady state solution to the reduced problem ($\mu=0$) is $u_r=4$. For $\mu\ll 1$ the problem is singularly perturbed and the steady state solution increases sharply from $u=1$ on $\partial\omega$ to $u=4$ on the interior. The solution to the parabolic problem approaches this steady state with time.

For the test problem we solve the nonlinear difference scheme (2) by the domain decomposition algorithm. With the mesh (3), (4), we take $N_x=N_y=N$. We fix the perturbation parameter $\mu=10^{-3}$, take $N=256$, 512 or 1024 and $\tau=0.1$ or 0.05. Given up to 16 processors, we are interested in the execution time of the algorithm under various domain decompositions. We suppose that $\{M,L\}\subset\{1,4,8,16,32\}$ and that the interfacial subdomains are chosen to be either all minimal or all maximal. For the minimal interfacial subdomains the number of mesh points in the x- and y-directions is three, and for the maximal interfacial subdomains the numbers of mesh points in the x- and y-directions are $N/M-2$ and $N/L-2$, respectively, (it means that the maximal interfacial subdomains do not overlap each other).

The numerical solution at $t_0=0$ is simply given by the initial-boundary conditions $V(\omega^h,t_0)=0,\ V(\partial\omega^h,t_0)=1$. The mesh function $\underline{V}^{(0)}(P,t_1)$ defined by $\underline{V}^{(0)}(P,t_1)=V(P,t_0),\ P\in\overline{\omega}^h$ is clearly a lower solution with respect to $V(P,t_0)$. We initiate the algorithm with $\underline{V}^{(0)}(P,t_1)$ and thus generate a sequence of lower solutions. At each time level t_k, we define a converged solution $V(P,t_k)=\underline{V}^{(n_*)}(P,t_k)$ with $n_*=n_*(t_k)$ minimal subject to $\|\underline{V}^{(n_*)}(t_k)-\underline{V}^{(n_*-1)}(t_k)\|_{\overline{\omega}^h}<\delta$, where δ is a specified tolerance. At the next time level, t_{k+1}, we require an initial iterate that is a lower solution with respect to $V(P,t_k)$. Since the boundary condition and function $f(u)=(u-4)/(5-u)$ are independent of time, we may choose $\underline{V}^{(0)}(P,t_{k+1})=V(P,t_k),\ P\in\overline{\omega}^h$. Indeed, since $\underline{V}^{(0)}(P,t_1)=V(P,t_0)\le\underline{V}(P,t_1)$, on time level t_2 with $\underline{V}^{(0)}(P,t_2)=\underline{V}(P,t_1)$, we have

$$\begin{aligned}&\mathcal{L}\underline{V}^{(0)}(P,t_2)+f(P,\underline{V}^{(0)}(P,t_2))-\tau^{-1}\underline{V}(P,t_1)\le\\&\mathcal{L}\underline{V}(P,t_1)+f(P,\underline{V}(P,t_1))-\tau^{-1}V(P,t_0)\le 0,\end{aligned}$$

where $P\in\overline{\omega}^h$ and $V^{(0)}(P,t_2)=g(P),\ P\in\partial\omega^h$. Thus, $\underline{V}^{(0)}(P,t_2)=\underline{V}(P,t_1)$ is a lower solution, and by induction on k, we prove the required result. Now, from Theorem 1, it follows by induction on k that the mesh function $\overline{V}(P,t_{k+1})$ defined by $\overline{V}(\omega^h,t_{k+1})=4,\ \overline{V}(\partial\omega^h,t_{k+1})=1$ is an upper solution with respect to $V(P,t_k)$ and thus our computed mesh functions satisfy

$$0\le\underline{V}^{(n)}(P,t_k)\le 4,\ P\in\overline{\omega}^h,\ 0\le n\le n_*,\ 0\le k\le N_\tau,$$

and hence we may suppose that $f_u=1/(5-u)^2$ is bounded below and above by $c_*=1/25$ and $c^*=1$, respectively. We take as our convergence tolerance $\delta=10^{-5}$.

We list in Table 1 average convergence iteration counts for the $M \times L$ domain decomposition algorithm. Each major cell of the table corresponds to a choice of N and τ and within each major cell we give the results corresponding to twenty-five $M \times L$ decompositions with minimal interfacial subdomains. We mention that domain decomposition does not increase the iteration count if the interfacial subdomains are chosen maximally. Because the final simulation time is $t = 1$, the average is over 10 time steps for $\tau = 0.1$ and 20 time steps for $\tau = 0.05$. For a given decomposition, the average iteration count is smaller for $\tau = 0.05$ than for $\tau = 0.1$. This simply reflects the smaller change in solution between time steps.

	τ	0.10					0.05				
N	$L\backslash M$	1	4	8	16	32	1	4	8	16	32
256	1	5.00	6.00	7.60	7.60	7.70	4.00	5.00	6.00	6.00	6.00
	4	6.00	6.00	7.60	7.60	8.00	5.00	5.00	6.00	6.00	6.00
	8	7.60	7.60	9.00	9.00	9.00	6.00	6.00	6.00	6.00	6.00
	16	7.60	7.60	9.00	9.00	9.00	6.00	6.00	6.00	6.00	6.00
	32	7.70	7.70	9.00	9.00	9.00	6.00	6.00	6.00	6.00	6.00
512	1	5.00	7.00	11.00	11.00	11.00	4.05	5.65	8.00	8.00	8.00
	4	7.00	7.00	11.00	11.00	11.00	5.65	6.00	8.00	8.00	8.00
	8	11.00	11.00	13.00	13.00	13.00	8.00	8.00	9.00	9.00	9.00
	16	11.00	11.00	13.00	13.00	13.00	8.00	8.00	9.00	9.00	9.00
	32	11.00	11.00	13.00	13.00	13.00	8.00	8.00	9.00	9.00	9.00
1024	1	5.00	10.00	18.10	18.10	18.10	4.05	7.00	12.20	12.20	12.20
	4	10.00	10.00	18.10	18.10	18.10	7.00	7.00	12.20	12.20	12.20
	8	18.10	18.10	23.00	23.00	23.00	12.20	12.20	15.00	15.00	15.00
	16	18.10	18.10	23.00	23.00	23.00	12.20	12.20	15.00	15.00	15.00
	32	18.10	18.10	23.00	23.00	23.00	12.20	12.20	15.00	15.00	15.00

TABLE 1. The average convergence iteration count for various τ, N and $M \times L$ decompositions with minimal interfacial subdomains.

In Tables 2 and 3 we give the execution times of the algorithm for $\tau = 0.1$ and $\tau = 0.05$, respectively. Where there is some choice for the interfacial subdomains, results corresponding to minimal and maximal interfacial subdomains are given above and below the line, respectively.

Consider first the serial implementation corresponding to $P = 1$. For $N = 256$ and $\tau = 0.1$, the execution time of the undecomposed algorithm ($M = 1$, $L = 1$) is 19.59 seconds. If one chooses the interfacial subdomains minimally, then domain decomposition reduces the serial execution time. The execution time is minimized to 4.77 seconds by the 4×32 decomposition with minimal interfacial subdomains. For

	N	256					512					1024				
P	$L\backslash M$	1	4	8	16	32	1	4	8	16	32	1	4	8	16	32
1	1	20	18/25	17/19	14/16	7.1/7.9	160	197/238	289/258	246/228	160/149	997	1968/1681	3061/1863	2938/1950	2765/1869
	4	18/25	9.2/28	6.3/23	5.5/22	5.3/17	198/239	134/286	145/285	131/269	68/206	1966/1679	1525/2128	2623/2303	2263/2309	1454/2030
	8	17/19	6.2/23	6.6/21	6.5/20	6.0/15	289/257	144/285	158/302	108/272	62/219	3069/1863	2630/2303	3002/2393	2072/2275	1741/2164
	16	14/16	5.3/22	6.4/20	6.4/19	6.1/14	247/227	129/268	109/272	68/248	60/208	2944/1958	2259/2288	2071/2265	1877/2299	1226/2140
	32	6.6/7.8	**4.8**/17	5.7/15	6.0/14	5.8/9.9	156/148	71/204	60/219	60/206	**54**/166	2763/1870	1449/2031	1733/2164	1217/2136	**692**/1979
2	1	$*$	12/19	10/12	7.7/9.7	4.0/4.7	$*$	127/174	189/172	152/144	89/86	$*$	1262/1206	1958/1252	1886/1283	1755/1208
	4	12/19	5.3/23	3.5/17	2.9/16	3.0/13	128/174	83/222	81/204	71/188	35/146	1269/1201	973/1603	1685/1654	1379/1607	797/1398
	8	10/12	3.5/18	3.5/14	3.6/13	3.5/10	189/171	81/206	86/201	56/177	31/140	1955/1255	1678/1654	1825/1630	1134/1505	911/1427
	16	7.8/9.5	3.0/16	3.5/13	3.6/11	3.7/8.7	153/144	70/189	55/178	35/156	31/125	1883/1288	1382/1611	1132/1510	987/1488	633/1383
	32	3.8/4.2	**2.8**/13	3.3/9.8	3.5/8.8	3.7/6.2	86/87	36/145	31/138	31/124	**28**/95	1762/1205	794/1400	912/1429	631/1381	**351**/1289
4	1	$*$	$*$	5.4/7.2	4.9/5.3	2.1/2.7	$*$	$*$	111/113	89/84	48/47	$*$	$*$	1135/659	1127/657	1062/617
	4	$*$	2.9/14	2.1/10	**1.8**/9.2	1.8/7.7	$*$	47/136	44/134	37/113	20/86	$*$	560/857	1006/888	811/852	436/739
	8	5.4/7.2	2.1/11	2.1/9.0	2.0/8.0	2.3/6.5	110/113	43/135	45/140	29/117	16/93	1131/661	1005/887	1058/880	623/800	484/753
	16	5.3/5.4	1.9/9.2	2.2/7.6	2.4/6.6	2.7/5.4	89/84	37/113	29/115	18/93	17/73	1137/657	808/846	621/798	522/780	331/717
	32	2.1/2.6	1.9/7.9	1.9/6.4	2.3/5.2	2.4/3.9	48/47	19/86	17/92	17/73	**16**/54	1060/617	439/737	491/750	334/715	**184**/661
8	1	$*$	$*$	$*$	2.3/2.9	1.2/1.3	$*$	$*$	$*$	45/45	25/24	$*$	$*$	$*$	606/398	548/367
	4	$*$	$*$	1.3/8.1	1.2/7.7	**1.1**/6.8	$*$	$*$	23/98	19/86	11/70	$*$	$*$	509/710	408/667	226/587
	8	$*$	1.3/8.1	1.4/4.8	1.4/4.3	1.3/3.5	$*$	22/98	23/79	16/65	**8.9**/52	$*$	506/701	537/552	313/487	245/455
	16	2.3/2.9	1.5/7.5	1.4/4.5	1.4/3.9	1.8/3.4	44/45	19/86	16/65	9.9/52	9.1/41	612/398	406/647	313/491	265/457	168/425
	32	1.2/1.3	1.2/6.8	1.4/3.3	1.6/3.0	2.2/2.7	25/24	11/70	9.4/51	9.5/40	9.3/29	548/367	231/580	255/465	176/427	**97**/395
16	1	$*$	$*$	$*$	$*$	1.0/1.4	$*$	$*$	$*$	$*$	15/14	$*$	$*$	$*$	$*$	290/185
	4	$*$	$*$	$*$	1.2/6.7	**1.1**/6.4	$*$	$*$	$*$	11/74	7.1/64	$*$	$*$	$*$	207/552	120/483
	8	$*$	$*$	1.7/4.8	1.4/3.7	1.2/3.3	$*$	$*$	13/75	9.6/53	7.0/45	$*$	$*$	270/533	162/397	128/358
	16	$*$	1.5/7.4	1.4/4.2	1.4/2.6	1.6/2.3	$*$	11/73	9.7/53	6.3/32	6.3/24	$*$	205/531	160/397	136/287	88/244
	32	1.0/0.9	1.2/6.5	1.5/3.5	1.5/2.8	2.9/3.1	14/14	7.2/63	**6.3**/44	6.5/24	6.9/17	290/186	119/467	133/357	91/246	**51**/201

TABLE 2. The execution time of the algorithm on an $M \times L$ decomposition with P processors. Each simulation comprises ten time steps with $\tau = 0.1$. Results corresponding to minimal and maximal interfacial subdomains are given above and below the line, respectively. A $*$ indicates that it is not possible to balance the computational load of Step 1 at the first level of parallelization. For each choice of N and P, the minimum execution time is given in bold type.

$N = 256$ and $\tau = 0.05$, the serial execution time of the undecomposed algorithm is 20.39 seconds. Again, if one chooses minimal interfacial subdomains, the execution

	N	256					512					1024				
P	$L\backslash M$	1	4	8	16	32	1	4	8	16	32	1	4	8	16	32
1	1	20	20/26	18/22	15/18	8.3/9.4	160	206/247	276/262	240/239	164/163	1297	1964/1959	2982/2130	2914/2200	2790/2127
	4	20/26	11/30	7.5/25	7.1/24	6.5/19	205/246	143/308	138/299	122/281	65/219	1959/1958	1591/2554	2678/2717	2339/2681	1545/2405
	8	18/22	7.3/25	6.9/24	7.0/22	6.7/18	277/262	136/298	144/303	85/273	62/229	2979/2129	2671/2714	2990/2775	2079/2619	1795/2512
	16	15/17	6.9/23	6.8/22	7.0/22	7.0/16	239/236	121/285	86/272	65/260	61/221	2897/2211	2343/2684	2073/2614	1903/2629	1266/2454
	32	7.7/8.6	**6.1**/19	6.4/18	6.8/17	6.7/12	163/160	61/221	60/227	60/221	**57**/184	2789/2121	1537/2422	1784/2522	1261/2456	**724**/2297
2	1	$*$	14/20	12/14	8.9/11	4.5/5.6	$*$	132/181	182/174	151/149	98/95	$*$	1253/1416	1907/1416	1861/1446	1787/1373
	4	14/20	6.4/24	4.2/20	3.8/18	3.8/15	133/180	90/240	82/214	69/196	32/156	1260/1419	1016/1971	1708/1945	1424/1892	845/1683
	8	11/14	4.1/19	3.7/16	3.8/14	3.9/12	182/174	81/216	79/201	48/179	32/145	1909/1419	1707/1953	1831/1861	1133/1708	935/1620
	16	8.9/11	3.8/18	3.8/14	3.9/13	4.2/10	159/149	68/201	43/180	34/166	32/137	1862/1452	1436/1892	1131/1705	996/1689	656/1569
	32	4.4/5.2	**3.4**/14	3.6/12	4.0/11	4.2/7.8	96/93	32/157	31/145	31/134	**30**/106	1785/1375	842/1681	941/1642	654/1583	**368**/1491
4	1	$*$	$*$	6.1/8.0	4.6/5.7	2.4/3.0	$*$	$*$	106/117	86/88	51/53	$*$	$*$	1088/788	1119/736	1065/701
	4	$*$	3.4/14	2.2/13	**2.1**/10	2.2/9.2	$*$	51/149	43/144	36/121	18/95	$*$	585/1057	1020/1091	833/1006	463/894
	8	6.4/8.0	2.4/12	2.3/11	2.3/9.5	2.5/8.3	107/117	43/144	41/144	23/121	17/100	1090/788	1016/1086	1065/1085	621/954	498/909
	16	4.5/5.6	2.2/10	2.2/8.7	2.3/7.5	2.8/6.2	85/88	35/122	23/120	18/101	17/81	1111/734	830/997	620/944	529/878	341/815
	32	2.4/2.9	2.5/8.6	2.4/7.4	2.6/6.3	2.8/4.8	51/52	17/94	17/98	17/80	**16**/62	1061/700	465/885	502/902	349/816	**193**/766
8	1	$*$	$*$	$*$	2.7/3.2	1.5/1.6	$*$	$*$	$*$	43/47	26/28	$*$	$*$	$*$	586/464	548/418
	4	$*$	$*$	1.6/9.3	1.5/8.7	**1.3**/8.0	$*$	$*$	23/104	18/93	9.6/78	$*$	$*$	516/820	420/791	238/696
	8	$*$	1.5/9.5	1.6/5.6	1.5/5.0	1.5/4.3	$*$	23/103	21/79	13/66	**9.1**/54	$*$	513/827	539/639	317/580	252/527
	16	2.5/3.2	1.8/8.3	1.6/5.3	1.6/4.6	2.3/4.0	44/47	19/92	13/65	9.8/56	9.4/45	590/465	420/790	313/584	266/566	174/507
	32	1.4/1.5	1.8/7.6	1.6/4.2	1.9/4.0	2.4/3.3	26/27	10/77	9.3/53	9.8/44	9.7/33	547/416	239/697	261/535	183/511	**100**/452
16	1	$*$	$*$	$*$	$*$	1.4/1.2	$*$	$*$	$*$	$*$	16/16	$*$	$*$	$*$	$*$	285/221
	4	$*$	$*$	$*$	1.4/7.5	**1.3**/7.3	$*$	$*$	$*$	11/79	7.3/73	$*$	$*$	$*$	216/631	125/582
	8	$*$	$*$	1.6/5.6	1.4/4.2	1.5/3.9	$*$	$*$	12/73	8.9/53	7.1/47	$*$	$*$	271/613	169/453	134/425
	16	$*$	1.8/8.3	1.6/4.8	1.5/3.1	1.8/2.9	$*$	10/79	8.7/53	6.5/34	6.5/26	$*$	213/640	162/456	137/324	95/287
	32	1.4/1.3	1.6/7.4	1.7/4.0	2.1/3.4	2.8/3.6	16/16	7.7/70	7.4/45	**6.4**/26	7.3/20	284/219	123/585	133/423	95/290	**53**/252

TABLE 3. The execution time of the algorithm on an $M \times L$ decomposition with P processors. Each simulation comprises twenty time steps with $\tau = 0.05$. Results corresponding to minimal and maximal interfacial subdomains are given above and below the line, respectively. A $*$ indicates that it is not possible to balance the computational load of Step 1 at the first level of parallelization. For each choice of N and P, the minimum execution time is given in bold type.

time is reduced by domain decomposition and is minimized to 6.12 seconds on the 4×32 decomposition with minimal interfacial subdomains. Similarly, for $N \geq 512$ and

each τ, it is evident from Tables 2 and 3 that the algorithm's serial execution is fastest on the 32×32 decomposition with minimal interfacial subdomains. Clearly, we have a notion of *serial speedup*, which derives purely from the domain decomposition. For $P \geq 2$ our parallel implementation exploits the natural parallelism of the algorithm at Steps 1-3. Consider now the problem with $N = 512$ and $\tau = 0.1$. For three $M \times L$ decompositions and each value of P, we give in Table 4 the execution time, number of inter-process subdomain data transfers per global iteration (Steps 1 through 4 of the algorithm) and the total number of MPI messages per global iteration by which this data is transmitted. The serial code ($P = 1$) executes fastest on the 32×32 decomposition. For $P \leq 4$, the number of MPI communications is the same for all three listed decompositions. Therefore, we see that the 32×32 decomposition allows for the fastest execution on $P \leq 4$ processors. For $P \geq 8$ we find that the number of MPI communications necessitated by the 32×8 or 8×32 decomposition is significantly less than that by the 32×32 decomposition. As a result, we observe that the 32×32 decomposition does not minimize the execution time when $P \geq 8$. On the other hand, if N increases to 1024, the linear algebra arising from the 32×32 decomposition becomes increasingly significant and the communication overhead is increasingly justified. Thus, we see from Tables 2 and 3 that the 32×32, minimal interfacial subdomain decomposition minimizes the execution time for any number of processors when $N = 1024$.

P	32×32			32×8			8×32		
1	54.16	0	0	62.16	0	0	59.88	0	0
2	28.08	4928	6	31.41	1160	6	31.08	1160	6
4	15.58	6400	36	16.44	1740	36	17.26	1620	36
8	9.29	7136	168	8.93	1968	161	9.36	1786	161
16	6.88	9008	720	7.01	2212	489	6.26	2118	502

TABLE 4. For three $M \times L$ decompositions and five values of P, a triple giving the execution time, number of inter-process subdomain transfers per global iteration and number of MPI messages per global iteration. The problem has $N = 512$ and $\tau = 0.1$.

For each combination of τ, N, and P, we define the parallel speedup of our implementation with reference to the optimal domain decomposition. Let $T(\tau, N, P)$ be the minimal execution time over all domain decompositions on P processors. Thus, $T(\tau, N, P)$ is the smallest number in the major cell of Tables 2 and 3 corresponding to τ, N and P. We define the parallel speedup on P processors as the ratio

$$S(\tau, N, P) = \frac{T(\tau, N, 1)}{T(\tau, N, P)}.$$

In Figure 2 we show the parallel speedup as a function of P for $\tau = 0.1$ and the three values of N. The corresponding graph for $\tau = 0.05$ is shown in Figure 3. It is evident from these graphs that the parallel scale-up of the algorithm improves as the number of mesh points is increased.

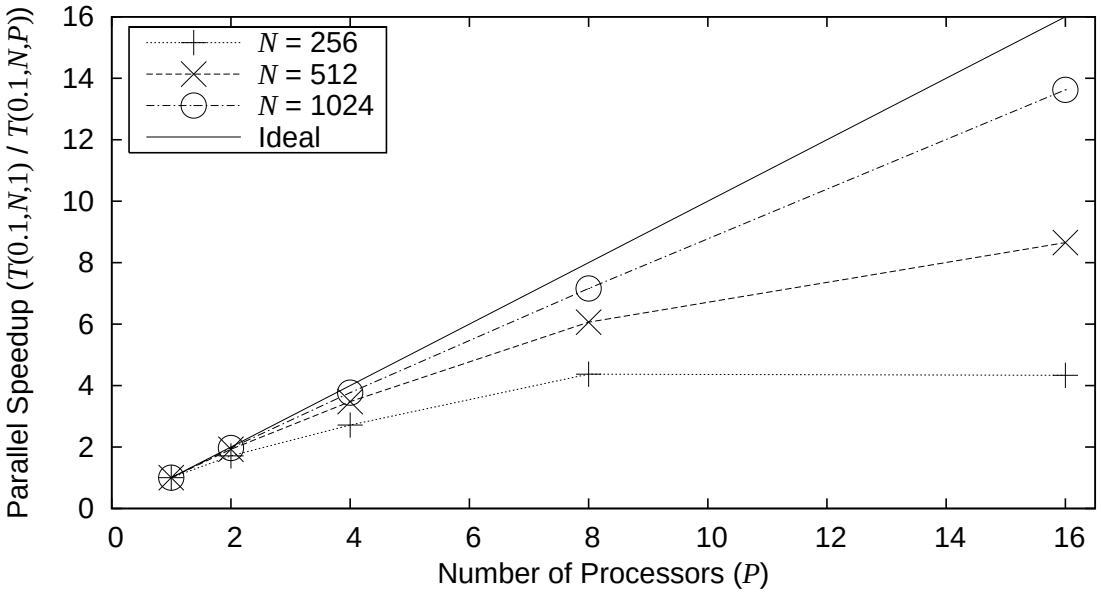

FIGURE 2. The parallel speedup $S(\tau = 0.1, N, P)$ as a function of number of processors P for three values of N and $\tau = 0.1$.

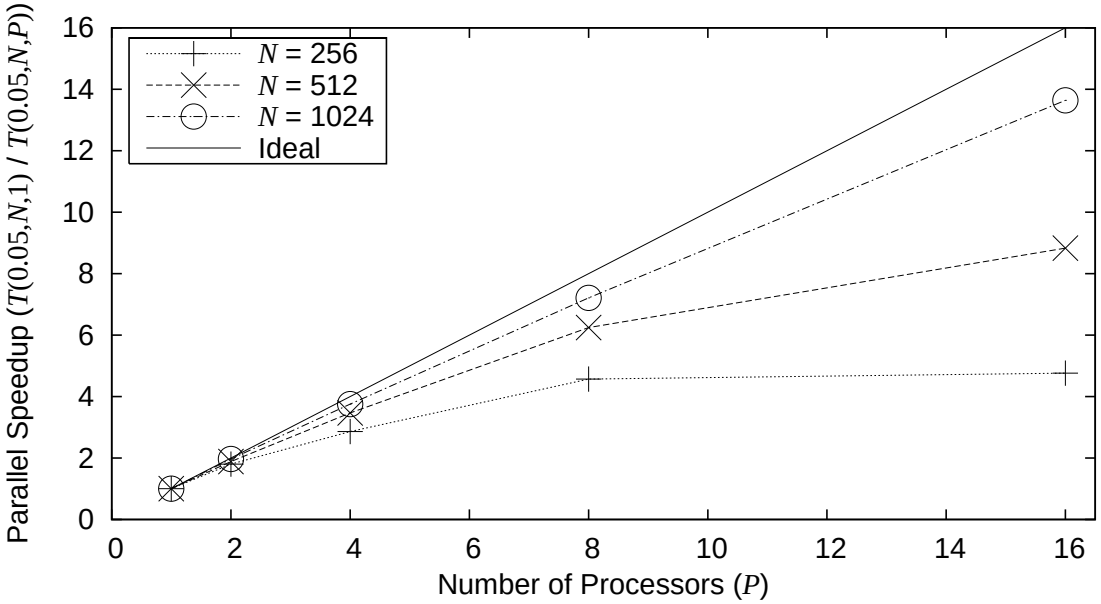

FIGURE 3. The parallel speedup $S(\tau = 0.05, N, P)$ as a function of number of processors P for three values of N and $\tau = 0.05$.

6. CONCLUSION

We have described the domain decomposition algorithm from [2] for singularly perturbed reaction-diffusion problems of parabolic type. A parallel implementation of the algorithm on a distributed memory cluster has been discussed.

From numerical experiments with a model test problem, we have found that if one chooses the interfacial subdomains maximally then the convergence iteration count remains at the undecomposed value. However, our measurements of wall time have shown that the algorithm executes fastest when the interfacial subdomains are chosen minimally. We have observed that domain decomposition can reduce the serial execution time of the algorithm. We have thus measured parallel speedup on P processors with reference to the best decomposition for P processors. The parallel scale-up of the algorithm improves as the number of mesh points is increased. With $N = 1024$ and $\tau = 0.1$ or 0.05, the parallel speedup on 16 processors is 13.6, corresponding to a parallel efficiency of 85.0

If one choose the interfacial subdomains minimally, then the domain decomposition algorithm reduces the serial execution time of the undecomposed algorithm. The parallel domain decomposition algorithm executes fastest with the minimal interfacial subdomains. When the number of mesh points increases, then the 32×32, minimal interfacial subdomain decomposition minimizes the parallel execution time for any number of processors.

REFERENCES

[1] I. Boglaev, Monotone iterative algorithms for a nonlinear singularly perturbed parabolic problem, J. Comput. Appl. Math. 172, 313–335 (2004).

[2] I. Boglaev, M.P. Hardy, Monotone box-domain decomposition algorithms for nonlinear singularly perturbed reaction-diffusion problems, Adv. Difference Equ. Vol. 2006, Article ID 70325, 38 pages (2006)// DOI:10.1155/ADE/2006/70325.

[3] I. Boglaev, M.P. Hardy, Uniform convergence of a weighted average scheme for a nonlinear reaction-diffusion problem, J. Comput. Appl. Math. 200, 705–721 (2007).

[4] E. Bohl, Finite Modelle gewöhnlicher Randwertaufgaben, Teubner, Stuttgart (1981).

[5] M.P. Hardy, I. Boglaev, Parallel implementation of a monotone domain decomposition algorithm for nonlinear reaction-diffusion problems, ANZIAM J. 46, C290–C303 (2005).

[6] J.J.H. Miller, E. O'Riordan, G.I. Shishkin, Fitted Numerical Methods for Singular Perturbation Problems, World Scientific, Singapore (1996).

[7] C.V. Pao, Finite difference reaction diffusion equations with nonlinear boundary conditions, Numer. Meth. Part. Diff. Eqs. 11, 355–374 (1995).

[8] Y. Saad, M.H. Schultz, A generalized minimal residual method for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7, 856–869 (1986).