

# Parallel implementation of a monotone domain decomposition algorithm for nonlinear reaction-diffusion problems

M. P. Hardy \*      I. Boglaev†

September 15, 2004

## Abstract

Recently, a monotone iterative domain decomposition algorithm has been proposed for the numerical solution of nonlinear singularly perturbed reaction-diffusion problems. 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.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Discrete Approximation of (1)</b>	<b>3</b>
<b>3</b>	<b>Box-domain decomposition algorithm</b>	<b>3</b>
<b>4</b>	<b>Numerical experiments</b>	<b>6</b>
4.1	Parallel implementation . . . . .	6
4.2	Results . . . . .	7

---

\*Institute of Fundamental Sciences, Massey University, Palmerston North, NEW ZEALAND. <mailto:m.p.hardy@massey.ac.nz>

†<mailto:i.boglaev@massey.ac.nz>

# 1 Introduction

We are interested in the nonlinear singularly perturbed reaction-diffusion problem of elliptic type

$$\begin{aligned} -\mu^2(u_{xx} + u_{yy}) + f(x, y, u) &= 0, & (x, y) \in \omega, \\ \omega = \omega^x \times \omega^y &= (0, 1) \times (0, 1), & u = g \text{ on } \partial\omega, \end{aligned} \tag{1}$$

where  $\mu \ll 1$  is the perturbation parameter. We also assume that  $c^* \geq f_u \geq c_*$ ,  $(x, y, u) \in \bar{\omega} \times \mathbb{R}$ , where  $c^*$  and  $c_*$  are positive constants and  $f_u \equiv \partial f / \partial u$ . The solution is characterised by boundary layers of width  $\mathcal{O}(\mu |\ln \mu|)$ .

On a piecewise equidistant mesh, discrete approximation of (1) leads to an algebraic system of nonlinear difference equations whose solution converges with mesh refinement to 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 [4]. Indeed, as shown in [1], the initial iterate may be constructed using only the boundary conditions and a lower bound on  $f_u$ . No knowledge of the solution is necessary to implement the algorithm.

The advent of the beowulf cluster has brought high-performance computing within reach of academe and thus fostered renewed interest in alternating Schwarz-type domain decomposition iterative algorithms. In [2] the domain is partitioned into nonoverlapping boxes and the monotone iterative method is applied on each subdomain. At each horizontal and vertical boundary, 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. This paper describes a parallel implementation of the algorithm from [2].

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

## 2 Discrete Approximation of (1)

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

$$\begin{aligned}\bar{\omega}^{hx} &= \{x_i, 0 \leq i \leq N_x; x_0 = 0, x_{N_x} = 1; h_{xi} = x_{i+1} - x_i\}, \\ \bar{\omega}^{hy} &= \{y_j, 0 \leq j \leq N_y; y_0 = 0, y_{N_y} = 1; h_{yj} = y_{j+1} - y_j\}.\end{aligned}$$

Given a mesh function  $U$  on  $\bar{\omega}^h$ , let  $\mathcal{D}_x^2 U(P)$  and  $\mathcal{D}_y^2 U(P)$  be the respective central difference approximations to the  $x$ - and  $y$ - second derivatives at point  $P = (x_i, y_j) \in \omega^h$ . Define the linear difference operator  $\mathcal{L}^h$  by

$$\mathcal{L}^h U \equiv -\mu^2 (\mathcal{D}_x^2 + \mathcal{D}_y^2) U.$$

For discrete approximation of the continuous problem (1) we employ the classical central difference scheme

$$\mathcal{L}^h U + f(P, U) = 0, \quad P \in \omega^h, \quad U = g \text{ on } \partial\omega^h. \quad (2)$$

We define our meshes  $\bar{\omega}^{hx}$  and  $\bar{\omega}^{hy}$  in the manner of [3]. Boundary layer thicknesses  $\sigma_x$  and  $\sigma_y$  are chosen as

$$\sigma_x = \min \left\{ 0.25, \mu c_*^{-\frac{1}{2}} \ln N_x \right\}, \quad \sigma_y = \min \left\{ 0.25, \mu c_*^{-\frac{1}{2}} \ln N_y \right\}, \quad (3)$$

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

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

The mesh  $\bar{\omega}^{hx}$  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  $\bar{\omega}^{hy}$  is defined similarly. The difference scheme (2) on the piecewise uniform mesh (3),(4) converges  $\mu$ -uniformly to the solution of the continuous problem (1). That is,

$$\max_{P \in \bar{\omega}^h} |U(P) - u(P)| \leq C (N^{-1} \ln N)^2, \quad N = \min\{N_x, N_y\},$$

where the constant  $C$  is independent of  $\mu$  and  $N$ . The proof of this result can be found in [3].

## 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

$\omega$  into  $M \times L$  nonoverlapping main subdomains  $\omega_{m,l}, m = 1, \dots, M, l = 1, \dots, L$ :

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

Then introduce vertical interfacial subdomains  $\theta_m, m = 1, \dots, M - 1$ :

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

and horizontal interfacial subdomains  $\vartheta_l, l = 1, \dots, L - 1$ :

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

With the global mesh  $\bar{\omega}^h = \bar{\omega}^{hx} \times \bar{\omega}^{hy}$ , 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 this box-domain decomposition, the algorithm from [2] may be described as follows.

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

Given a global iterate  $V^{(n)}$ , Steps 1 through 4 below generate the next global iterate  $V^{(n+1)}$ .

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

$$(\mathcal{L}^h + c^*)Z_{m,l}^{(n+1)} = - [\mathcal{L}^h V^{(n)} + f(P, V^{(n)})], \quad P \in \omega_{m,l}^h,$$

with  $Z_{m,l}^{(n+1)}(\partial\omega_{m,l}^h) = 0$ .

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

$$(\mathcal{L}^h + c^*)Z_m^{(n+1)} = - [\mathcal{L}^h V^{(n)} + f(P, V^{(n)})], \quad P \in \theta_m^h,$$

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

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

$$(\mathcal{L}^h + c^*)\tilde{Z}_l^{(n+1)} = - [\mathcal{L}^h V^{(n)} + f(P, V^{(n)})], \quad P \in \vartheta_l^h,$$

with  $\tilde{Z}_l^{(n+1)}(\partial\vartheta_l^h)$  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 through 3 to form the new global iterate  $V^{(n+1)}$ :

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

**Step 5.** If the solution is not converged then increment  $n$  and go to Step 1.

## Convergence of the algorithm

We say that a mesh function  $\bar{V}$  is an upper solution of (2) if it satisfies

$$\mathcal{L}^h \bar{V} + f(P, \bar{V}) \geq 0, \quad P \in \omega^h, \quad \bar{V} \geq g \text{ on } \partial\omega^h.$$

Similarly,  $\underline{V}$  is a lower solution of (2) if it satisfies the reversed inequalities.

**Theorem 1** *Let  $\bar{V}^{(0)}$  and  $\underline{V}^{(0)}$  be upper and lower solutions of (2). From  $\bar{V}^{(0)}$  and  $\underline{V}^{(0)}$ , the algorithm respectively generates sequences  $\{\bar{V}^{(n)}\}$  and  $\{\underline{V}^{(n)}\}$  which converge monotonically from above and below to the unique solution  $U$  of (2):*

$$\underline{V}^{(0)} \leq \underline{V}^{(n)} \leq \underline{V}^{(n+1)} \leq U \leq \bar{V}^{(n+1)} \leq \bar{V}^{(n)} \leq \bar{V}^{(0)} \text{ in } \bar{\omega}^h.$$

The proof of this result may be found in [2]. A method for constructing the initial iterate  $V^{(0)}$  from an arbitrary mesh function is also given therein. Thus, in contrast to Newton's method, the domain decomposition algorithm may be implemented without any prior knowledge of the solution to (2).

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

$$\bar{h}_{x_m}^{b,e} = \frac{1}{2}(h_{x_m}^{b-,e-} + h_{x_m}^{b+,e+}), \quad \bar{h}_{y_l}^{b,e} = \frac{1}{2}(h_{y_l}^{b-,e-} + h_{y_l}^{b+,e+}),$$

and  $x$ - and  $y$ - decomposition parameters  $q^I$  and  $q^{II}$ ,

$$q^I = \max_{1 \leq m < M} \left\{ \frac{\mu^2}{c^* \bar{h}_{x_m}^{b,e} h_{x_m}^{b+,e-}} \right\}, \quad q^{II} = \max_{1 \leq l < L} \left\{ \frac{\mu^2}{c^* \bar{h}_{y_l}^{b,e} h_{y_l}^{b+,e-}} \right\}.$$

**Theorem 2** *For each sequence of Theorem 1 we have, for  $n \geq 1$ ,*

$$\|V^{(n+1)} - V^{(n)}\|_{\bar{\omega}^h} \leq (q + q^I + q^{II}) \|V^{(n)} - V^{(n-1)}\|_{\bar{\omega}^h}. \quad (5)$$

where  $q = 1 - c_*/c^*$  is the convergence rate of the undecomposed monotone iterative algorithm ( $M = L = 1$ ).

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

**Remark 1** *Consider an  $M \times L$  decomposition in which the mesh points are distributed equally among main subdomains. This is called a balanced decomposition. If a balanced domain decomposition has  $M \geq 4$  then  $\theta_1^h$  and  $\theta_{M-1}^h$  overlap the boundary layers and  $q^I$  is maximal. Similarly, if a balanced decomposition has  $L \geq 4$  then  $q^{II}$  is maximal. By contrast, if the interfacial subdomains are located wholly outside the boundary layers, ensuring minimality of  $q^I$  and  $q^{II}$ , then the decomposition is said to be unbalanced.*

## 4 Numerical experiments

We now apply the algorithm to the reaction-diffusion problem

$$\begin{aligned}
 -\mu^2(u_{xx} + u_{yy}) + \frac{u - 4}{5 - u} &= 0, & (x, y) \in \omega, \\
 \omega = \omega^x \times \omega^y = (0, 1) \times (0, 1), & u(x, y) = 1, & (x, y) \in \partial\omega.
 \end{aligned} \tag{6}$$

The solution to the reduced problem ( $\mu = 0$ ) is  $u_r = 4$ . For  $\mu \ll 1$  the problem is singularly perturbed and the solution increases sharply from  $u = 1$  on  $\partial\omega$  to  $u = 4$  on the interior. The function  $f(u) = (u - 4)/(5 - u)$  has its derivative  $f_u$  bounded above and below by  $c^* = 1$  and  $c_* = 1/25$ , respectively.

For the continuous problem (6), we solve the nonlinear difference scheme (2) by the domain decomposition algorithm. With the mesh (3),(4), we take  $N_x = N_y = N$ . 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 [5], suitable for nonsymmetric systems. We fix the perturbation parameter  $\mu = 10^{-3}$  and take  $N = 256, 512$  or  $1024$ . Given  $P$  processors, where  $P \in \{1, 2, 4, 8, 16\}$ , 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 maximal or all minimal. In all experiments, we take as our initial iterate the lower solution  $\underline{V}^{(0)}(P) = 0$ ,  $P \in \omega^h$ ,  $\underline{V}^{(0)}(P) = 1$ ,  $P \in \partial\omega^h$ . Our convergence criterion is  $\|V^{(n)} - V^{(n-1)}\|_{\omega^h} \leq 10^{-5}$ .

### 4.1 Parallel implementation

The present work concerns only the first level of parallelisation. That is, each subdomain is wholly assigned to a processor and no attempt is made to parallelise GMRES. Therefore, we only consider balanced domain decompositions. As suggested by the estimate (5) and confirmed by previous serial experiments of both balanced and unbalanced domain decompositions [2], the latter require fewer iterations for convergence. Nevertheless, the improved convergence rate of an unbalanced domain decomposition would be at least partially offset by extra inter-process communication.

In a balanced domain decomposition, the main subdomains share the mesh points equally. On the other hand, the respective workloads of the corresponding linear problems vary with mesh spacing. In order to balance the computational cost of Step 1 of the algorithm, we first classify the main subdomains by mesh spacing, and then divide each class equally among the processors. 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  $h_{y\mu}$ . Because we take  $N_x=N_y$ , 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). Within a given class, each subdomain incurs the same computational cost. Comparing across classes, the cost per subdomain is greatest for the class F-F, and least for the class C-C. We assign the main subdomains to the  $P$  processors accordingly. 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. Thus, at the first level of parallelisation, the load of Step 1 is distributed equally among the processors. On the other hand, if  $P$  does not divide  $ML/4$ , load balancing would require second level parallelisation 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  $\{0, 1, \dots, P - 1\}$ :  $\bar{\theta}_m^h \mapsto \text{mod}(m - 1, P)$ . The assignment of horizontal interfacial subdomains is similar:  $\bar{\vartheta}_l^h \mapsto \text{mod}(l - 1, P)$ .

## 4.2 Results

Convergence iteration counts of the algorithm are shown in Table 1 for various mesh sizes  $N$  and  $M \times L$  decompositions. Results corresponding to minimal and maximal interfacial subdomains are given above and below the line respectively. For each mesh size, the convergence iteration count of the undecomposed algorithm is 21. This increases under decomposition. For each  $M \times L$  decomposition, the iteration count is minimised when maximal interfacial subdomains are chosen.

In Table 2 we list the execution time of the algorithm on an  $M \times L$  decomposition with  $P$  processors. Again, 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 Step 1 of the

$N$	256					512					1024				
$L \setminus M$	1	4	8	16	32	1	4	8	16	32	1	4	8	16	32
1	21	$\frac{26}{21}$	$\frac{40}{21}$	$\frac{41}{22}$	$\frac{49}{24}$	21	$\frac{36}{21}$	$\frac{62}{21}$	$\frac{62}{21}$	$\frac{72}{23}$	21	$\frac{59}{21}$	$\frac{102}{21}$	$\frac{102}{21}$	$\frac{113}{23}$
4	$\frac{26}{21}$	$\frac{26}{21}$	$\frac{40}{21}$	$\frac{41}{22}$	$\frac{49}{24}$	$\frac{36}{21}$	$\frac{36}{21}$	$\frac{62}{21}$	$\frac{62}{21}$	$\frac{72}{23}$	$\frac{59}{21}$	$\frac{59}{21}$	$\frac{102}{21}$	$\frac{102}{21}$	$\frac{113}{23}$
8	$\frac{40}{21}$	$\frac{40}{21}$	$\frac{41}{21}$	$\frac{43}{22}$	$\frac{55}{24}$	$\frac{62}{21}$	$\frac{62}{21}$	$\frac{72}{21}$	$\frac{72}{21}$	$\frac{86}{23}$	$\frac{102}{21}$	$\frac{102}{21}$	$\frac{126}{21}$	$\frac{126}{21}$	$\frac{143}{23}$
16	$\frac{41}{22}$	$\frac{41}{22}$	$\frac{43}{22}$	$\frac{44}{22}$	$\frac{57}{24}$	$\frac{62}{21}$	$\frac{62}{21}$	$\frac{72}{21}$	$\frac{73}{21}$	$\frac{88}{23}$	$\frac{102}{21}$	$\frac{102}{21}$	$\frac{126}{21}$	$\frac{127}{21}$	$\frac{144}{22}$
32	$\frac{49}{24}$	$\frac{49}{24}$	$\frac{55}{24}$	$\frac{57}{24}$	$\frac{67}{24}$	$\frac{72}{23}$	$\frac{72}{23}$	$\frac{86}{23}$	$\frac{88}{23}$	$\frac{103}{23}$	$\frac{113}{23}$	$\frac{113}{23}$	$\frac{143}{23}$	$\frac{144}{22}$	$\frac{163}{23}$

Table 1: The convergence iteration count of the algorithm for various  $M \times L$  decompositions. Results corresponding to minimal and maximal interfacial subdomains are given above and below the line, respectively.

algorithm at the first level of parallelisation. Considering first the case of  $P = 1$ , corresponding to the serial code, it is interesting to note that for each mesh size  $N$ , the execution time is minimised by the  $32 \times 32$  decomposition with minimal interfacial subdomains. On the other hand, for  $P \geq 2$  and  $N = 256$  the communication overhead of the  $32 \times 32$  decomposition becomes significant and the optimal decomposition is  $16 \times 4$  with minimal interfacial subdomains. As we increase  $N$  to 512 and 1024, this overhead is increasingly offset by the extra computation between data transfers.

The subdomains  $\bar{\theta}_1^h$  and  $\bar{\theta}_{M-1}^h$  overlap the boundary layer and thus entail a greater computational workload than each of  $\bar{\theta}_2^h, \dots, \bar{\theta}_{M-2}^h$ . In addition, the requirement that the  $ML$  main subdomains of Step 1 be evenly distributed among the processors leads to an uneven distribution of the  $M-1$  subdomains in Step 2. Similar considerations apply to the  $L-1$  horizontal interfacial subdomain problems of Step 3. Fortunately for our first level parallel implementation, the sequential execution of the algorithm is fastest when the interfacial subdomains are minimal. Thus, in the parallel implementation, the idle time during Steps 2 and 3 is minimised.

From the results for  $P = 1$ , there is clearly some algorithmic speedup deriving purely from the domain decomposition. On the other hand, the parallel implementation of a given domain decomposition allows us to realise the natural parallelism. Given the algorithm's serial performance, we quantify the parallel speedup of our implementation. For a given mesh size  $N$  and number of processors  $P$ , let  $T(N, P)$  be the minimum execution time over all domain decompositions. In Table 2,  $T(N, P)$  is the smallest time in the major cell corresponding to  $N$  and  $P$ . We define the parallel speedup of the algorithm by the ratio  $T(N, 1)/T(N, P)$ . This is plotted in Figure 1 for each

		256					512					1024				
$P$	$L \setminus M$	1	4	8	16	32	1	4	8	16	32	1	4	8	16	32
1	1	34	<u>32</u>	29	25	15	249	356	497	400	252	2037	4014	5665	5394	4798
	4	<u>46</u>	<u>15</u>	<u>14</u>	7	7	356	233	239	196	120	4014	3069	4895	4174	2484
	8	29	14	9	8	9	497	240	272	179	99	5642	4856	5482	3674	3188
	16	<u>33</u>	<u>42</u>	34	33	28	394	432	434	387	324	3272	4058	3944	3565	3493
	32	24	7	8	7	8	400	196	179	91	91	5348	4148	3674	3330	2255
2	1	28	38	32	30	26	332	393	385	343	299	3245	3849	3580	3439	3193
	4	<u>15</u>	<u>7</u>	<u>8</u>	<u>8</u>	<u>7</u>	251	119	97	90	83	4789	2469	3182	2254	1144
	8	17	35	28	26	18	212	326	325	300	223	2960	3490	3509	3218	2949
	16	*	<u>22</u>	<u>17</u>	<u>14</u>	<u>9</u>	*	220	307	237	137	*	2553	3597	3388	3029
	32	34	20	16	10	10	268	259	207	117	117	2241	2137	2043	1876	1876
4	1	<u>22</u>	<u>9</u>	<u>8</u>	<u>4</u>	<u>5</u>	220	139	129	102	63	2554	1951	3091	2516	1354
	4	35	37	29	26	25	268	346	304	272	227	2234	3037	2893	2682	2440
	8	<u>17</u>	<u>8</u>	<u>5</u>	<u>5</u>	<u>6</u>	307	129	141	94	52	3603	3092	3324	2003	1659
	16	20	29	21	20	18	259	304	286	251	205	2118	2902	2619	2302	2233
	32	<u>14</u>	<u>4</u>	<u>5</u>	<u>5</u>	<u>6</u>	234	102	93	48	48	3384	2514	1991	1729	1181
8	1	16	26	20	18	16	206	271	250	216	180	2042	2689	2320	2177	2023
	4	8	4	6	6	8	135	64	51	48	46	3026	1340	1663	1177	593
	8	<u>10</u>	<u>25</u>	<u>18</u>	<u>16</u>	<u>14</u>	118	227	206	183	128	1881	2434	2243	2040	1869
	16	*	*	<u>9</u>	<u>7</u>	<u>5</u>	*	*	187	141	76	*	*	1965	1797	1580
	32	11	11	9	5	5	148	116	64	64	64	1176	1066	962	962	962
16	1	*	5	3	3	3	*	80	71	55	34	*	988	1570	1293	698
	4	*	<u>22</u>	<u>17</u>	<u>16</u>	<u>15</u>	*	186	169	145	121	*	1621	1605	1416	1293
	8	<u>9</u>	<u>5</u>	<u>3</u>	<u>3</u>	<u>3</u>	187	71	75	49	27	1959	1565	1688	1017	847
	16	11	17	13	12	11	148	169	164	138	117	1175	1591	1495	1278	1257
	32	8	3	3	3	4	141	58	51	26	26	1790	1289	1011	879	598
32	1	9	15	12	10	9	114	146	139	114	95	1059	1417	1280	1147	1058
	4	5	3	3	4	5	76	34	26	25	24	1578	713	843	597	305
	8	5	15	11	9	8	62	122	118	95	66	956	1293	1251	1062	967
	16	*	*	*	<u>4</u>	<u>3</u>	*	*	*	77	42	*	*	*	1016	874
	32	5	5	3	3	3	61	33	33	33	33	542	478	478	478	478
64	1	*	*	<u>3</u>	<u>2</u>	<u>2</u>	*	*	<u>36</u>	<u>31</u>	<u>19</u>	*	*	<u>779</u>	<u>658</u>	<u>371</u>
	4	*	*	14	13	13	*	*	127	112	102	*	*	1140	1020	970
	8	*	<u>3</u>	<u>2</u>	<u>2</u>	<u>2</u>	*	<u>37</u>	<u>38</u>	<u>26</u>	<u>15</u>	*	<u>781</u>	<u>847</u>	<u>524</u>	<u>431</u>
	16	4	2	2	2	3	127	93	77	64	64	1131	833	693	658	658
	32	<u>5</u>	<u>2</u>	<u>2</u>	<u>2</u>	<u>3</u>	76	31	27	14	14	1017	687	525	449	306
128	1	60	112	77	61	51	60	112	77	61	51	543	1017	694	607	545
	4	3	2	2	2	3	42	20	15	14	14	875	388	442	316	165
	8	3	14	6	5	4	33	102	64	51	35	478	975	663	548	484
	16	*	*	*	*	<u>3</u>	*	*	*	*	<u>25</u>	*	*	*	*	<u>559</u>
	32	3	3	3	3	3	18	100	61	32	20	287	843	530	344	288

Table 2: The execution time, rounded up to the nearest second, of the algorithm on an  $M \times L$  decomposition with  $P$  processors. Results corresponding to minimal and maximal interfacial subdomains are given above and below the line, respectively. A \* indicates that the computational load of Step 1 cannot be balanced at the first level of parallelisation.

mesh size  $N$ . Also shown is the ideal speedup. It can be seen that for  $N = 256$ , the parallel speedup is greatest for  $P = 8$ . As  $N$  increases the parallel scale-up improves.

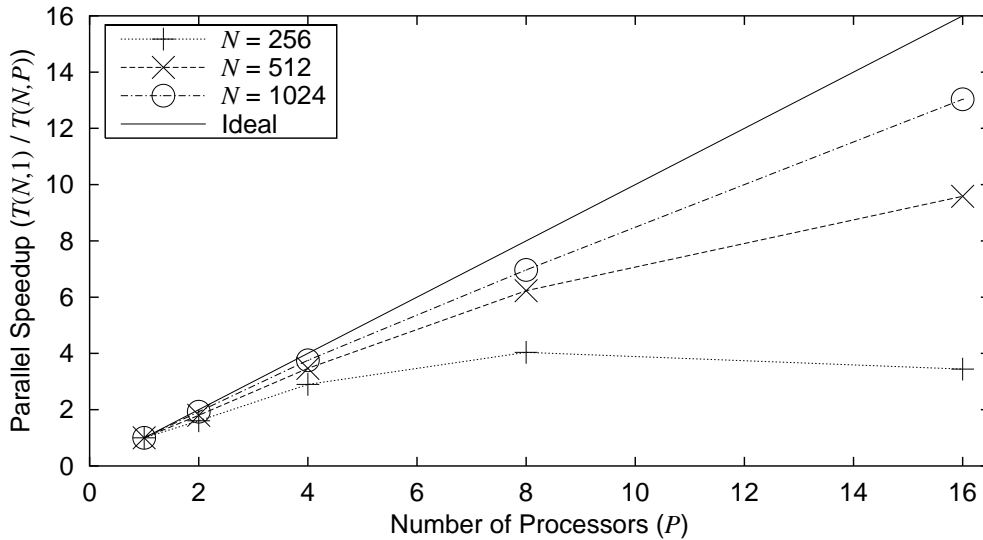


Figure 1: The parallel speedup for three mesh sizes  $N$ .

## References

- [1] I. Boglaev. On monotone iterative methods for a nonlinear singularly perturbed reaction-diffusion problem. *J. Comput. Appl. Math.*, 162:445–466, 2004.
- [2] I. Boglaev and M. P. Hardy. Monotone box-domain decomposition algorithms for nonlinear singularly perturbed reaction-diffusion problems. Reports in Mathematics 7, Massey University, 2004. (partly submitted to *Advances in Difference Equations*).
- [3] J. J. H. Miller, E. O’Riordan, and G. I. Shishkin. *Fitted numerical methods for singular perturbation problems*. World Scientific, Singapore, 1996.
- [4] C. Pao. Monotone iterative methods for finite difference system of reaction-diffusion equations. *Numer. Math.*, 46(4):571–586, 1985.
- [5] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual method for solving nonsymmetric linear systems. 7:856–869, 1986.