Particle Partitioning Strategies for the Parallel Computation of Solid-Liquid Flows[*]

L. Little[†] Z. Li [†] H. G. Choi[‡] Y. Saad [†]

October 26, 2000

Abstract

The numerical investigation of the interaction of large, solid particles with fluids is an important area of research for many manufacturing processes. Such studies frequently lead to models that are very large and require the use of parallel solution techniques. This paper presents the results of a parallel implementation of a serial code for the direct numerical simulation of solid-liquid flows. The base code is a serial, Arbitrary Lagrangian-Eulerian (ALE) formulation of the equations of motion, which views that particles as solid bodies embedded into the flow domain. This particular model poses some interesting difficulties for domain decomposition type approaches for parallel solution. In particuler, it is not fully understood how the partitioning of the particles among the subdomains influences the performance of parallel solvers. We present several strategies for the partitioning of the solid particles, focusing on the effectiveness of these techniques in terms of parallel speedup and efficiency.

Key words: solid-liquid flows, domain decomposition, parallel methods, preconditioning
AMS subject classifications: 65F10, 65F50, 65N22

1 Introduction

Direct numerical simulations of fluid-particle flows are important in several industrial applications, such as oil refining and drilling, catalyst cracking, coating enhancement and hydraulic fracturing. Medical research areas such as rheology have also benefited from the study of such flows. While researchers have long used particle tracking methods (*e.g.*, in the study of free surface flows), the effect of large particles on the fluid itself has largely been ignored until recently [6].

[*]Work supported by NSF under grant NSF/CTS 9873236 and by the Minnesota Supercomputer Institute

Often, the numerical simulations to be performed contain hundreds or even thousands of particles, resulting in models which require large amounts of computer memory. In addition, as the number of particles increases the allowable time step decreases due to the CFL condition. Hence, the amount of computer time required to perform such simulations can be prohibitive. In light of these difficulties, the use of parallel computers has become necessary. One aspect of parallel computing techniques that has not been systematically investigated is how the partitioning of the particles affects the performance of iterative solvers. This paper presents a few 'overlapping' strategies used when partitioning the particles among processors, in the context of domain decomposition methods.

Most discrete models used to simulate the interaction of solids and liquids fall into one of three types. These are the deforming-spatial-domain/space-time (DST/DT), the distributed Lagrangian multiplier / fictitious domain (DLM), and the arbitrary Lagrangian-Eulerian (ALE) models.

The deforming-spatial-domain/space-time method is described in [17, 18] and is based on using finite element spaces that span both the spatial and temporal domains. With this technique, the deformation of the spatial domain is automatically accounted for. The approach was initially tested on the two-dimensional flow of drafting cylinders. This approach is used in [13] to model the three-dimensional flow of 5 spheres in a vertical channel at a Reynolds number of 100.

The DLM model as it is most frequently used was first employed in [21] and is characterized by structured, fixed computational grids. The particles flow through the grid as time advances, creating a moving fictitious domain for each particle. The Lagrange multiplier referred to is a constraint on the mesh nodes internal to the particles and is included to enforce a condition of rigid body rotation at the internal mesh nodes. The appeal of this model is that fast solution techniques such as multigrid or fast Poisson solvers can be used, because of the structured grids. A primary disadvantage is that local mesh refinement is not possible. This can result in models that are larger than may be necessary. In addition, a smaller time step is required (relative to the ALE method to be described) to obtain accurate representations of the particle physics. This approach was used in [10] to simulate the two-dimensional flow of 540 circular particles and the three-dimensional flow of 2 spheres in a sedimentation column. In [22], the DLM model is used to simulate the three-dimensional fluidization of 1204 spheres.

The ALE model, originally proposed in [20] employs an unstructured, moving grid in which particles are embedded and treated as moving, solid boundaries. The fluid equations are derived relative to the local mesh velocity. Because the mesh velocity varies through the grid, the frame of reference is, in some sense, arbitrary. The unstructured grid generally precludes the efficient use of multigrid solution techniques, but larger time steps can be taken. In addition, local mesh refinement is possible in regions where pressure and velocity gradients are large. The ALE model was employed in [4] to investigate the two-dimensional Poiseuille flow of 100 circles in a vertical channel for Reynolds numbers near 100. In [2], it was used to model the fluidized lift-off of

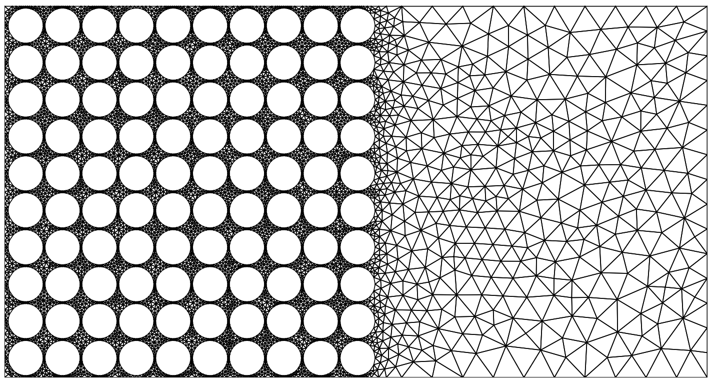


Figure 1: A solid-liquid flow domain consisting of 100 particles in a closed box. Gravity is acting towards the right.

300 particles in a periodic channel for Reynolds numbers as large as 3500.

The above brief overview of research and techniques for modeling solid-liquid flows is by no means comprehensive and readers are encouraged to consult the references given, particularly [4, 10], for more information.

The purpose of this paper is to study the effect of several techniques for partitioning the particles in the parallel implementation of a particular serial ALE code [4] for fluid-particle simulations. The emphasis will be on investigating the parallel performance and efficiency of these techniques.

2 Problem Description and Formulation

The mathematical description of the ALE method used in the solid-liquid simulations to be investigated has been presented in [1, 4] and is summarized here for completeness. Consider a two-dimensional region $\bar{\Gamma}$ containing an incompressible fluid and an arbitrary number of circular particles. Such a domain is illustrated in Figure 1. The region of $\bar{\Gamma}$ occupied by the fluid is given by

$$\Gamma = \bar{\Gamma} \setminus (\Gamma_d \cup \Gamma_n \cup \Gamma_p)$$

where Γ_d, Γ_n and Γ_p are the sections corresponding to Dirichlet boundary conditions, Neuman boundary conditions and particle surfaces respectively.

The motion of the fluid is described by the conservation of mass and momentum equations:

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 & (1)\\ \rho_f \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) &= -\nabla p + \nabla \cdot \mathbf{T} & (2) \end{aligned}$$

where $\mathbf{u}$ is the fluid velocity vector, p is the pressure and ρ_f is the fluid density. If the fluid is taken to be Newtonian then the extra stress tensor $\mathbf{T} = \mu_f \mathbf{A}(\mathbf{u})$ where μ_f is the fluid viscosity and $\mathbf{A}(\mathbf{u}) = \nabla \mathbf{u} + (\nabla \mathbf{u})^T$ is the strain rate tensor. All quantities, aside from μ_f and ρ_f, are assumed to be of the form $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$.

The particle motion is governed by Newton's second law

$$\left. \begin{aligned} \mathbf{M} \frac{\mathrm{d}\mathbf{U}_p}{\mathrm{d}t} &= \mathbf{F}_p + \mathbf{G}_p \\ \frac{\mathrm{d}\mathbf{X}_p}{\mathrm{d}t} &= \mathbf{U}_p \end{aligned} \right\} \quad p = 1, 2, \ldots, N_p \qquad (3)$$

where N_p is the number of particles. The translational and rotational equations have been combined. $\mathbf{M}$ is the generalized mass matrix of the particle, $\mathbf{X}_p, \mathbf{U}_p$ and $\mathbf{F}_p$ are, respectively, the generalized position vector, velocity vector and force vectors of the particle. $\mathbf{G}_p$ represents external forces (*i.e.*, gravity) acting on the particles. The mass matrix can is described by

$$\mathbf{M} = \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{pmatrix}$$

where m is the particle mass per unit length and I is the moment of inertia about the center of mass. The remaining quantities can be written as

$$\mathbf{X}_p = \begin{pmatrix} X \\ Y \\ \Theta \end{pmatrix}; \quad \mathbf{U}_p = \begin{pmatrix} U \\ V \\ \Omega \end{pmatrix}; \quad \mathbf{F}_p = \begin{pmatrix} F_x \\ F_y \\ F_m \end{pmatrix}; \quad \mathbf{G}_p = (\rho_p - \rho_f) g A_p \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Here, X, Y and Θ represent the position and angular orientation of the particle, U, V and Ω are the particle vector and angular velocities, A_p is the particle area, ρ_p is the particle density, and g is the force of gravity which has been assumed to act in the x-direction.

Initial conditions for the fluid and particles are imposed as:

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0 \; ; \qquad \mathbf{X}_p(\mathbf{x}, 0) = \mathbf{X}_0 \; ; \qquad \mathbf{U}_p(\mathbf{x}, 0) = \mathbf{U}_0.$$

The boundary conditions on the fluid are

$$\begin{aligned} \mathbf{u} &= \mathbf{u}_d \quad \text{on } \Gamma_d \\ -p \cdot \mathbf{n} + \mathbf{T} \cdot \mathbf{n} &= \mathbf{u}_n \quad \text{on } \Gamma_n \\ \mathbf{u} &= \mathbf{U}_p + \boldsymbol{\Omega}_p \times (\mathbf{x} - \mathbf{X}_p) \quad \text{on } \Gamma_p. \end{aligned}$$

The last of these conditions reflects the no-slip condition on the particle surfaces. With this notation, it is possible to define fluid and particle Reynolds numbers as

$$Re_f = \frac{\|\mathbf{u}\| D \rho_f}{\mu_f} \ ; \qquad Re_p = \frac{|\ \|\mathbf{U}_p\| - \|\mathbf{u}\|\ | D_p \rho_p}{\mu_f}$$

respectively, where D is a domain dependent length scale, D_p is the particle diameter and $\| * \|$ represents the L^2-norm.

2.1 Discretization

Before describing the discretized equations, the momentum equation must be altered slightly. As the motion of the particles evolves, the portion of $\bar{\Gamma}$ occupied by the fluid changes. The ALE scheme makes use of moving grids to compensate for this. This introduces an arbitrary reference frame indicated by the local grid velocity $\mathbf{u}_g$. The fluid is assumed to be convected relative to $\mathbf{u}_g$, so the modified momentum equation becomes

$$\rho_f \left(\frac{\delta \mathbf{u}}{\delta t} + (\mathbf{u} - \mathbf{u}_g) \cdot \nabla \mathbf{u} \right) = -\nabla p + \nabla \cdot \mathbf{T} \tag{4}$$

where the time derivative is taken in the new reference frame. If $\mathbf{u}_g = 0$, the standard Eulerian frame is recovered while if $\mathbf{u}_g = \mathbf{u}$, the Lagrangian frame is observed.

Two other important elements of the model must also be mentioned briefly. The first is the use of the combined fluid-particle formulation [4, 10]. This formulation greatly simplifies the solution of the discrete equations because it eliminates the need for computing the forces and torques imparted by the fluid on the particles. The principle behind this approach is to incorporate the equations for the particle motion (3) into the momentum equation (4) when writing these equations in their weak form. Appropriate use of integration by parts results in the cancellation of all particle boundary integrals. The second important modification to the model is the use of a particle projection technique described in [9]. In this approach, which is used in conjunction with the combined formulation, the equations associated with the velocity variables on the particle surfaces are projected onto the particle variables $\mathbf{U}_p$. The advantage of this technique is that it yields a saddle point problem with an SPD matrix in the (1,1) block. For simulations involving several thousands of particles, the memory savings associated with this transformation are significant.

The spatial discretization in the current ALE model is achieved through a P2-P1 finite element formulation with isoparametric P2 elements employed to characterize the circular particle boundaries.

The time stepping process is a four step, second-order accurate operator splitting scheme [1, 3]. This scheme is implicit for the fluid variables while the particles are advanced using a simple first-order, explicit relation. Let $\mathbf{u}^n$ and $\mathbf{U}_p^n$ denote the fluid and particle velocity vectors and p^n represent the pressure, which are known at some discrete time $t = n\Delta t$. The splitting method proceeds by first solving a convection diffusion equation, yielding intermediate velocities $\hat{\mathbf{u}}$ and $\hat{\mathbf{U}}_p$

$$\left.\begin{aligned} \rho_f \left(\frac{\hat{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \frac{1}{2}\tilde{\mathbf{u}}^n \cdot \hat{\mathbf{u}} + \frac{1}{2}\tilde{\mathbf{u}}^n \cdot \mathbf{u}^n \right) &= \mathbf{T}^n + \hat{\mathbf{T}} \\ \mathbf{M}\frac{\hat{\mathbf{U}}_p - \mathbf{U}_p^n}{\Delta t} &= \mathbf{F}_p + \mathbf{G}_p. \end{aligned}\right\} \qquad (5)$$

Here, $\mathbf{T}^n$ and $\hat{\mathbf{T}}$ represent the discretized extra stress tensors, given by

$$\mathbf{T}^n = \frac{1}{2}\mu_f \mathbf{A}(\mathbf{u}^n) \ ; \qquad \hat{\mathbf{T}} = \frac{1}{2}\mu_f \mathbf{A}(\hat{\mathbf{u}}).$$

For the case of a Newtonian fluid, these become discrete Laplacian terms. The convection term has been linearized in terms of the relative grid velocity $\tilde{\mathbf{u}}^n = \mathbf{u}^n - \mathbf{u}_g^n$. Because there is no reference to the pressure, the computed $\hat{\mathbf{u}}$ does not necessarily satisfy the incompressibility condition. This is enforced in the second step by solving

$$\left.\begin{aligned} \rho_f \left(\frac{\mathbf{u}^* - \hat{\mathbf{u}}}{\Delta t} \right) &= \nabla p^n \\ \mathbf{M}\frac{\mathbf{U}_p^* - \hat{\mathbf{U}}_p}{\Delta t} &= \mathbf{F}_p \end{aligned}\right\} \qquad (6)$$

for the intermediate velocities $\mathbf{u}^*$ and $\mathbf{U}_p^*$. Another feature of the combined formulation is that it allows the third and fourth steps in [3] to be combined into a single step. In conjunction with the particle projection step, this results in a symmetric saddle point problem which simultaneously computes the pressure field while enforcing the incompressibility condition:

$$\left.\begin{aligned} \rho_f \left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} \right) &= -\nabla p^{n+1} \\ \mathbf{M}\frac{\mathbf{U}_p^{n+1} - \mathbf{U}_p^*}{\Delta t} &= \mathbf{F}_p \\ \nabla \cdot \mathbf{u}^{n+1} &= 0. \end{aligned}\right\} \qquad (7)$$

The mesh velocity $\mathbf{u}_g^n$ is obtained by solving a Laplace equation on the pressure space, then interpolating the result to the velocity grid. Eventually the translational and rotational motion of the particles requires that the computational domain be remeshed. This is determined by evaluating the quality of the elements after moving the mesh according to $\mathbf{u}_g^n$. When remeshing is performed, the existing $\mathbf{u}^n$ is quadratically interpolated to the new mesh.

The code is not limited to Newtonian fluids. It also supports Oldroyd-B fluids, which are used in the modeling of blood flows. Other fluids can be included, provided

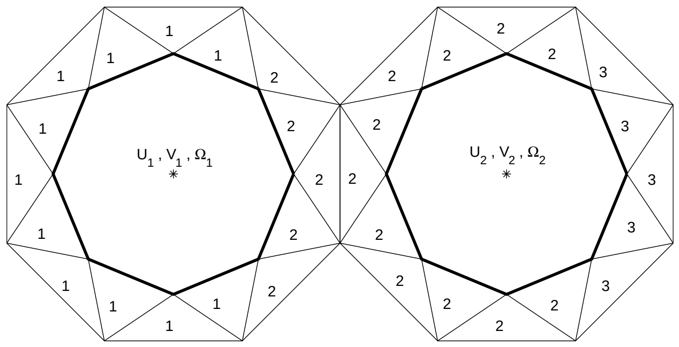


Figure 2: An example of a typical partitioning of the surface elements for two closely spaced particles.

the constitutive equation for **T** is known. More general particles shapes, such as ellipses and rectangles can also be selected. The individual particles can have different radii, however, all particles must be of the same basic shape.

3 Parallel Implementation

The parallel implementation was achieved through the use of PSPARSLIB [12], an MPI-based, portable library of domain decomposition-type solvers for general sparse linear systems. The main preconditioners available are variants of the additive and multiplicative Schwarz procedures described in [15, 16], as well as the more sophisticated Schur complement approach described in [14]. The local systems are typically solved by a simple local ILU-type techniques, possibly accelerated with a local GMRES iteration. A variety of (global) accelerators are available. However, the primary accelerator is FGMRES a flexible variant of GMRES which allows the preconditioner to vary at each step. PSPARSLIB also provides sparse data structures and communication routines for implementing the distributed matrix-vector product operation and the preconditioners mentioned above.

The basic input to the PSPARSLIB data structure routines is the distribution of the local equations among the various subdomains. For the ALE code under consideration this is somewhat complicated due to the particle projection step described in Section 2. In order to correctly generate the discrete particle equations, information from all elements that come in contact with the particle surface is required. As an example, consider Figure 2, where it is supposed that some technique for subdividing the computational domain has been used. Here, two particles are close enough that their surface elements come in contact. To generate the equations for particle P_1, all the surface elements and particle variables associated with particle P_2 will also be required. In the example domain shown in Figure 1, all of the particles exhibit this characteristic.

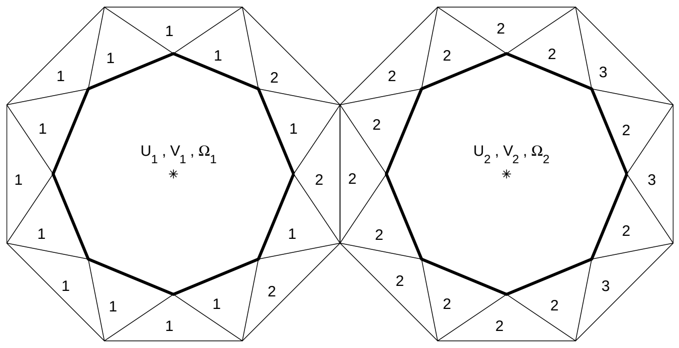


Figure 3: Particle partitioning example from Figure 2 with method **M4** imposed.

Because the density of the finite element grid is greatest near particles, the situation where the particle surface elements are distributed among two or more subdomains is almost unavoidable and some method of partitioning these particles must be used. Four approaches have been considered.

M1 Overlap all particle variables and surface elements on all relevant subdomains. For the example indicated in Figure 2 all the elements and particle variables would be present on subdomains 1, 2 and 3.

M2 Overlap only the particle variables on all relevant subdomains. In Figure 2 the particle variables associated with particles P_1 and P_2 would appear on domains 1, 2 and 3 while the surface elements would be placed in their respective domains. Information from the required particles is available locally to avoid communication during the generation of the local equations.

M3 Overlap nothing. Place particle in domain based on weight of surface elements. For Figure 2 the variables for particle P_1 would be placed in subdomain 1 and those for particle P_2 would be placed in subdomain 2.

M4 Same as M3, but with the additional constraint that all elements having an edge on the particle are reassigned to same domain as particle. This is illustrated in Figure 3.

It is also possible to graft a layer of structured elements onto each particle, as was done in [7, 13]. This effectively isolates each particle from its neighbors and greatly reduces the number of overlapping particles. Unfortunately, for particles in close proximity, the layer of elements must be extremely thin in order to obtain elements that are suitable for computation. For the domain shown in Figure 1 this was observed to increase the number of elements by 100-200%, hence this approach is not considered here.

Experimentation with these methods indicates that method **M1** converges faster if some degree of additional element overlap at subdomain boundaries is included and that methods **M2**–**M4** will not converge at all without additional overlap. One layer of overlap is obtained by examining all elements that share a node with a given element (say E_i) and including them in the same domain as E_i. More layers can be obtained by applying this idea recursively.

Two techniques for dividing $\bar{\Gamma}$ into N_s subdomains have been examined. The first is the program Metis [8]. For the second, a simple one-dimensional partitioner is employed. This partitioner places each element in one of N_b accumulation bins based on the x-coordinate of the centroid. The elements within each bin are sorted in order of increasing y-coordinate. Then, starting with the first bin, elements are placed into the first subdomain until $N_e = \frac{N_b}{N_s}$ elements are accumulated. The next set of N_e elements are placed in the second subdomain and so on until all the elements are exhausted. This partitioning technique will be referred to as the bin-sorting method. The final partitioning procedure is then described by

1) Partition domain on an elemental basis,

2) Adjust partitioning to account for particle overlap using one of methods **M1**–**M4** above,

3) Include additional layers of element overlap.

In the current model, only the saddle point equation (7) has been implemented in parallel. The solution of this equation represents the largest portion of solution time. Equations (5) and (6) are implemented in serial and are solved in each processor. Though this seems somewhat wasteful, these equations are effectively solved using diagonally preconditioned Bicgstab. Further, the current mesh generator is serial and this constitutes the main limitation to increasing the problem size.

4 Numerical Experiments

We consider four test cases for purposes of examining the computational accuracy as well as the performance of the parallel solver in light of the partitioning methods **M1**–**M4**. In all cases, the fluid and particles start from rest and all particles are circular with a radius of 1 cm.

T1 A single particle in a gravity driven sedimentation column. The domain is a closed, square box with side length 20 cm. The particle and fluid densities are 1.01 and 1.00 $\frac{\text{g}}{\text{cm}^3}$ respectively and the fluid viscosity is 0.01 poise. The particle is initially in the lateral center of the column as shown in Figure 4.

T2 1000 particles in a sedimentation column. The domain is a rectangular box of width 21.05 cm and length 104.5 cm. The particle and fluid densities are 1.1 and 1.0 $\frac{\text{g}}{\text{cm}^3}$ respectively. The particles are initially configured in a 20×50 array (see Figure 10).

T3 Lift-off of 300 particles in a periodic channel. The domain is 12 cm wide, 63 cm long and periodic in the x-direction. The fluid and particle properties are the same as in case **T1**. The flow is driven by a pressure gradient of 1.0 $\frac{\text{dyn}}{\text{cm}}$. The particles are initially in a 5 $\times$ 60 crystal array on the bottom of the channel (Figure 15).

T4 Lift-off of 7008 particles in a periodic domain. The domain is 102.4 cm wide and 248 cm long. The fluid and particle densities are 1.14 and 1.00 $\frac{\text{g}}{\text{cm}^3}$ respectively and the fluid viscosity is 0.01 poise. Initially, the particles are configured in a 96 $\times$ 73 crystal array.

The first two cases are important in the modeling of industrial coating processes and increasing the efficiency of chemical reactions through catalyst cracking. In this case, the fluid represents a coating that is to be applied to the particles. The latter cases are used to model the processes of hydraulic fracturing and lubricated transport. The flows are driven by a pressure gradient. For sufficiently high gradients, the particles are lifted off the upper layer of the initial crystal layer into the free stream. Eventually all particles are lifted off and settle into an aggregate height in the mid stream. They cannot completely rise to the top of the channel because the shear stress at the upper wall is acting to drive the particles back down into the free stream.

All the tests described here were performed on an IBM-SP2 supercomputer located at the Minnesota Supercomputing Institute. This machine has several subconfigurations, but the portion used for these experiments consists of 17 nodes, each node having four 222 MHz Power3+ processors sharing 16 GB of memory. Additionally, due to the expense and difficulty of obtaining dedicated computer time, only the parallel performance tests were performed in dedicated mode. The timing of the performance studies was done using a wall-clock timer provided in the MPI implementation (MPI_WTime), however, by the results to be presented in Sections 4.2 and 4.3, it is important to note that these times might be in error by as much as one second even though the runs were performed in dedicated mode at times when the overall system use was low.

In all cases, the parallel solver for the saddle point problem is an additive Schwarz algorithm using local ILU preconditioning and GMRES for the accelerator. The convergence criteria is a reduction of the initial preconditioned residual be a factor of 10^{-8}. The original serial code employed a conjugate gradient algorithm with ILU(0) preconditioning. The parameters for GMRES and ILU for each test case are given in Table 1. In each case, both Metis and the Bin-sorting partitioning are tested. The number of bins for the Bin-sorting technique was taken to be $N_b = 20$. Experimentation showed that this value gave reasonable performance results for all test cases under consideration.

Before proceeding to the test case results, it is instructive to make two comments about the behavior observed when running simulations with the parallel code. First, with the exception of Case **T1**, all of the cases eventually require remeshing of the domain. However, the initial mesh size is the largest one. Remeshing of the domain

	T1	**T2**	**T3**	**T4**
m	50	100	80	100
k	2	10	4	10

Table 1: Krylov subspace dimensions m for GMRES(m) and fill-level k in ILU(k), for Cases **T1**–**T4**.

Procs (n)	E_n	Period (s)	$U_{\text{term}}\left(\frac{\text{cm}}{\text{s}}\right)$
1	–	1.34	2.47
2	6.20(-4)	1.34	2.47
4	6.47(-4)	1.34	2.47
8	7.81(-4)	1.34	2.47
16	7.12(-4)	1.34	2.47

Table 2: Results of accuracy tests for Case **T1**. The error in column 2 represents the absolute error in the total solution vector relative to that obtained on 1 processor.

typically results in somewhere between 2-5% fewer elements than the initial size. Second, in order for the parallel solver to converge, it is necessary to employ an averaging matrix-vector product operation which averages all the data in the overlapping regions.

4.1 Case T1

The first test is a simple accuracy test. This is done to ensure that an acceptably accurate solution is being computed independent of the number of processors. The simulation is run for 3000 time steps, corresponding to $t = 15$ seconds.

The physical behavior of the particle in this simulation has been studied extensively. The particle rapidly accelerates to a terminal velocity, then enters a periodic steady state, as shown in Figure 5. The results of these tests are in Table 2. The steady state period of oscillation and the average terminal velocity for the particle are in good agreement with the values given in [1]. Further, there is little variation in the errors in the total solution vector at $t = 15$. Because the angular displacement of the particle is small, remeshing never needs to be done in this case.

The result of these tests is in Table 2. The steady state period of oscillation and the average terminal velocity for the particle agree with the value given in [1]. Further, there is a consistent degree of accuracy when the number of processors is varied.

The parallel speedup performance is shown in Figures 6 and 7. In addition, iteration counts, parallel efficiencies and total times are given in Table 3. Despite the relatively small size of the saddle point problem, reasonable speedup can be obtained for large numbers of processors. The best approach for this case appears to be either methods **M1** or **M2**, though all the methods give nearly identical results for a fixed partitioner.

	M1							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
1	32	3.45	1.00	1.00	32	3.45	1.00	1.00
2	38	2.42	1.46	0.73	36	2.24	1.58	0.79
4	38	1.30	2.72	0.68	41	1.84	1.92	0.48
8	43	0.98	3.61	0.45	48	1.48	2.39	0.29
16	45	0.66	5.36	0.33	47	1.08	3.27	0.20

	M2							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
1	32	3.45	1.00	1.00	32	3.45	1.00	1.00
2	38	2.38	1.46	0.73	36	2.20	1.61	0.80
4	38	1.30	2.72	0.68	40	1.89	1.87	0.46
8	43	0.98	3.61	0.45	48	1.37	2.58	0.32
16	45	0.66	5.36	0.33	47	1.08	3.27	0.20

	M3							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
1	32	3.45	1.00	1.00	32	3.45	1.00	1.00
2	38	2.42	1.46	0.73	36	2.23	1.58	0.79
4	38	1.30	2.72	0.68	42	1.78	1.99	0.49
8	43	0.98	3.61	0.45	49	1.42	2.49	0.31
16	45	0.68	5.20	0.32	54	1.22	2.90	0.18

	M4							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
1	32	3.45	1.00	1.00	32	3.45	1.00	1.00
2	38	2.42	1.46	0.73	36	2.20	1.61	0.80
4	38	1.30	2.72	0.68	42	1.78	1.99	0.49
8	43	0.98	3.61	0.45	49	1.41	2.51	0.31
16	46	0.68	5.20	0.32	54	1.21	2.92	0.18

Table 3: Performance results for Case **T1** using Metis and Bin-sorting partitioners. Absence of data indicates that the method did not converge in 300 iterations.

This is mainly due to the fact that the single particle does not create a substantive amount of overlap, hence all four methods are essentially the same. Note that there is a large discrepancy between Metis and the Bin-sorting partitioning. This is due to the fact that Metis partitions by attempting to minimize edge connections while the Bin-sorting partitioner is based only on element balance. Also, since the times may be in error by as much as one second (see Section 4.2), it is difficult to say which partitioner performs better.

Figures 8 and 9 give the amount of time spent in the major components of the parallel solver: Total time, GMRES, matrix vector and preconditioning operations. These figures reveal that both the GMRES solver and the matrix-vector operation do not scale well with increasing numbers of processors. This is due to the small size of the saddle point problem.

4.2 Case T2

The parallel performance results for this case are given in Table 4 and Figures 11 and 12 and indicate that Method **M2** with Metis partitioning performs the best in terms of overall parallel speedup and efficiency, though all methods exhibit poor performance. Again, possible perturbations in the accuracy of the timings lead one to conclude that Methods **M1** and **M2** are very similar. Figures 13 and 14 show the times of the individual components in the parallel solver for Method **M2** and Metis partitioning. In spite of the poor overall performance, all of the component operations appear to be scaling at the same rate.

The parallel performance in this case is best explained by examining Table 5 which contains information on how the data is partitioned among the subdomains for Methods **M1-M4** and both Metis and Bin-sort partitioning. The first rows give information on how many particles are shared among the various subdomains. The next rows give the average number of particles per subdomain as well as the maximum and minimum number of particles on any subdomain. The third set of rows give an indication of the load balancing expressed as a percentage of the total number of equations in the saddle point problem. Again, average, maximum and minimum values are given. The last set of rows is perhaps the most revealing. These indicate how many of the local variables need to be communicated to adjacent processors expressed as a percentage of the total number of variables local to each processor.

In the initial particle configuration, every particle is affected by adjacent particles through the particle projection step described in Section 3. This can be observed in the first line of Table 5 (Split Particles). Eventually, the number of particles that must communicate with adjacent subdomains reaches approximately 25%.

This table also reveals the possibility of large errors in the timing of the routines. If the local equations generated by Methods **M3** and **M4** are examined, it is seen that these two methods are identical, yet the execution times differ by 0.7 seconds for Bin-sorting partitioning and 1.5 seconds for Metis partitioning (for 16 processors).

The equivalence of these 2 methods implies that all the domain boundaries lie between particles. This can easily be seen in Figure 10, which includes the Metis and Bin-sorting partitionings of the domain for 8 processors. Also, although not explicitly shown, the same remarks regarding Methods **M3** and **M4** be made for Case **T1**.

The data communication portions of Table 5 reveal that the amount of data to be communicated becomes significant as the number of processors increases. For Methods **M1** and **M2** and 16 processors, the local variables to be communicated with adjacent processors is 50% and 30% of all variables, on average, respectively. The use of an Additive Schwarz algorithm implies that, aside from a few scalar product operations in the GMRES accelerator, substantial communication occurs only in the matrix-vector operation. However, the large quantities of data to be averaged result in a bottleneck in the solution process. As one might expect, the amount of data communication is less for Method **M2**, but still substantial. Method **M3** requires only about half as much data communication as Method **M2**, but the performance is more erratic, particularly for 2 processors and Metis partitioning.

4.3 Case T3

The parallel performance results for this test are given in Table 6 and Figures 17 and 18. These figures show that method **M2** with Bin-sorting partitioning performs the best in terms of overall parallel speedup and efficiency. Figures 19 and 20 show the times of the individual components in the parallel solver. These figures also show that the GMRES and matrix-vector operations are not scaling well with increasing processors. This is due to the use of the averaging matrix-vector operation necessary to obtain convergence and also the fact that GMRES is not a constant work per iteration solver.

These results are qualitatively similar to those obtained in Case **T2**. Again, all of the particles in the initial configuration are affected by the particle projection step. Table 7 gives the data distribution as explained in the Section 4.2, although the data communication is even more severe. For 16 processors, Methods **M1** and **M2** require approximately 50% to 90% respectively of the local variables to be communicated (depending on the partitioning). It is interesting to note that the parallel performance is much improved despite the increase in data communication percentages. This confirms that the quantity of data to be communicated, which is much less than Case **T2**, also plays a significant role in performance.

Table 6 shows that Metis partitioning fails for Methods **M3** and **M4** with large numbers of processors. This can be explained by looking at the partitioning of the domain shown in Figure 15 and also the particle distribution information in Table 7. Metis returns a partitioning that is highly fragmented and gives unequal particle distributions, while Bin-sorting results in a more evenly balanced distribution.

Figures 22 and 21 show the performance of the parallel solver at time $t = 0.31$ seconds. The maximum particle Reynolds number observed at this time is about 3300. Table 8 gives the data distribution among the processors. The results here

	M1							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
2	56	85.6	1.000	1.000	58	87.2	1.000	1.000
4	62	58.9	1.453	0.727	64	60.6	1.439	0.796
8	67	41.9	2.043	0.511	69	44.1	1.977	0.494
16	72	32.2	2.658	0.332	76	36.7	2.376	0.297

	M2							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
2	61	91.3	1.000	1.000	62	92.3	1.000	1.000
4	73	65.6	1.392	0.696	71	64.7	1.427	0.713
8	80	44.8	2.038	0.510	83	47.5	1.943	0.489
16	88	33.1	2.758	0.345	95	43.9	2.103	0.263

	M3							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
2	74	114.9	1.000	1.000	71	106.2	1.000	1.000
4	147	124.9	0.920	0.460	80	72.9	1.457	0.728
8	89	48.9	2.350	0.587	88	50.1	2.120	0.530
16	112	38.8	2.961	0.370	100	46.1	2.304	0.288

	M4							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
2	74	115.1	1.000	1.000	71	106.2	1.000	1.000
4	147	124.6	0.924	0.462	80	73.5	1.445	0.722
8	89	48.8	2.359	0.590	88	49.2	2.159	0.540
16	112	40.3	2.856	0.357	100	46.8	2.263	0.284

Table 4: Performance results for Case **T2** using Metis and Bin-sorting partitioners. Absence of data indicates that the method did not converge in 300 iterations.

	Metis Procs				Bin-sorting Procs			
	2	4	8	16	2	4	8	16
	M1							
Split Part	24	92	170	244	25	75	177	258
Part per								
Ave	512	273	147	79	513	269	147	80
Max	516	287	159	86	515	280	155	85
Min	508	256	133	66	510	255	127	65
Eqns per								
Ave	0.5117	0.2726	0.1467	0.0785	0.5125	0.2687	0.1473	0.0806
Max	0.5135	0.2825	0.1554	0.0831	0.5149	0.2773	0.1530	0.0840
Min	0.5100	0.2649	0.1377	0.0724	0.5101	0.2581	0.1354	0.0703
Data Comm per								
Ave	0.0558	0.1995	0.3475	0.4689	0.0593	0.1671	0.3622	0.4979
Max	0.0560	0.2530	0.4265	0.5950	0.0594	0.2190	0.4160	0.5786
Min	0.0557	0.1482	0.2533	0.2984	0.0592	0.1130	0.2296	0.3211
	M2							
Eqns per								
Ave	0.5010	0.2519	0.1267	0.0638	0.5015	0.2524	0.1287	0.0656
Max	0.5025	0.2569	0.1308	0.0656	0.5024	0.2530	0.1308	0.0685
Min	0.4994	0.2497	0.1229	0.0621	0.5006	0.2521	0.1270	0.0639
Data Comm per								
Ave	0.0257	0.0982	0.1800	0.2568	0.0292	0.0876	0.2126	0.3063
Max	0.0261	0.1250	0.2295	0.3391	0.0322	0.1131	0.2773	0.4033
Min	0.0253	0.0677	0.1202	0.1519	0.0262	0.0500	0.1247	0.1650
	M3							
Split Part	0	0	0	0	0	0	0	0
Part per								
Ave	500	250	125	63	500	250	125	63
Max	507	258	132	66	510	259	130	66
Min	493	236	116	53	490	235	105	41
Eqns per								
Ave	0.5009	0.2517	0.1265	0.0637	0.5014	0.2523	0.1286	0.0655
Max	0.5024	0.2567	0.1306	0.0654	0.5023	0.2528	0.1307	0.0684
Min	0.4993	0.2496	0.1228	0.0620	0.5006	0.2520	0.1268	0.0637
Data Comm per								
Ave	0.0140	0.0477	0.0882	0.1290	0.0151	0.0464	0.1179	0.1780
Max	0.0163	0.0614	0.1229	0.1671	0.0196	0.0679	0.1478	0.2315
Min	0.0116	0.0334	0.0566	0.0681	0.0106	0.0229	0.0678	0.1067
	M4							
Eqns per								
Ave	0.5009	0.2517	0.1265	0.0637	0.5014	0.2523	0.1286	0.0655
Max	0.5024	0.2567	0.1306	0.0654	0.5023	0.2528	0.1307	0.0684
Min	0.4993	0.2496	0.1228	0.0620	0.5006	0.2520	0.1268	0.0637
Data Comm per								
Ave	0.0140	0.0477	0.0882	0.1290	0.0151	0.0464	0.1179	0.1780
Max	0.0163	0.0614	0.1229	0.1671	0.0196	0.0679	0.1478	0.2315
Min	0.0116	0.0334	0.0566	0.0681	0.0106	0.0229	0.0678	0.1067

Table 5: Distribution of particles, equations and communication for Case **T2** for Metis and Bin-sorting partitionings. The particle splittings are the same for Cases **M1** and **M2** and Cases **M3** and **M4**.

are somewhat different. As can be seen from the domain plots given in Figure 16, the particles have begun to lift-off from the initial positions along the bottom of the channel. This results in more spacing between the particles and hence, the number of particles that need to be overlapped due to the particle projection is reduced, at least for large numbers of processors. The number of overlapped particles is not reduced substantially for Metis partitioning, but Bin-soring results in about 50% less overlapping than in the initial configuration. This fact is reflected in both the reduction of the amount of data to be communicated and in the improvement in parallel performance. Also, Metis returns a more evenly distributed particle partitioning and as a result, the performance for Methods **M3** and **M4** is nearly the same as for Bin-sorting partitioning.

4.4 Discussion of Results

The performance results obtained for these test cases are somewhat inconclusive, but they do point to a number of important facts. It is apparent that an equal distribution of the particles is necessary for both good performance and robustness. There does not seem to be a uniform result with respect to the performance of the basic partitioner. In general, Metis partitioning is easier to implement and slightly faster than Bin-sorting, but this portion of the particle partitioning process represents only about 5% of the total effort. One disadvantage of the Bin-sorting partitioner is that it requires to select a number of bins N_b. This parameter can greatly affect the execution time, which can vary in some cases by as much as 200%.

The particle partitioning methods seem to be gathered into 2 distinct groups. Generally, Methods **M1** and **M2** perform the best and exhibit very similar performance. Method **M3** is slightly less effective but also involves significantly less data communication. This drop in effectiveness is believed to be due to the lack of particle overlap.

The averaging matrix-vector product operation is necessary for the particle partitioning methods to converge, but unfortunately it also becomes extremely costly for large numbers of processors where the percentage of local variables that need to be exchanged becomes significant. Reducing the amount of overlapping particles might improve the performance of this operation. One approach that would be simple to implement would be a compromise between Methods **M2** and **M3**. Instead of overlapping all particles, some subset (*i.e.*, every other one) of particles could be overlapped.

Another performance improvement that could be made is to use a constant work per iteration solver. A Bicgstab algorithm was test to see if this was indeed the case, but it was observed to lead to drastic increases in the iteration counts for large numbers of processors. Additionally, it would be useful to stabilize the number of iterations as the processor number is increased. The Schur complement preconditioner of [14] has been observed to have this property, however, the current version of this preconditioner assumes that there is no overlap between neighboring domains. The preconditioner will work with overlapping variables, but the performance was observed to degrade to the point that the iteration counts were nearly identical to the Additive Schwarz algorithm.

	M1							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
1	55	43.8	1.00	1.00	55	43.8	1.00	1.00
2	64	27.9	1.57	0.78	56	24.2	1.80	0.90
4	67	17.7	2.47	0.62	65	19.7	2.22	0.56
8	69	14.7	2.97	0.37	69	13.4	3.26	0.41
16	71	8.6	5.09	0.32	74	10.1	4.33	0.27

	M2							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
1	55	43.8	1.00	1.00	55	43.8	1.00	1.00
2	65	26.9	1.62	0.81	64	27.1	1.61	0.81
4	70	16.7	2.62	0.66	68	17.6	2.48	0.62
8	79	12.4	3.53	0.44	72	10.5	4.17	0.52
16	83	7.1	6.16	0.39	78	7.00	6.25	0.39

	M3							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
1	55	43.8	1.00	1.00	55	43.8	1.00	1.00
2	126	52.2	0.83	0.42	71	30.6	1.43	0.72
4	–				85	21.2	2.07	0.52
8	–				127	18.2	2.40	0.30
16	–				136	11.4	3.84	0.24

	M4							
	Metis				Bin-sorting			
Procs	Its	Time	Speedup	Efficiency	Its	Time	Speedup	Efficiency
1	55	43.8	1.00	1.00	55	43.8	1.00	1.00
2	126	52.0	0.84	0.42	71	30.8	1.42	0.71
4	–				85	20.6	2.12	0.53
8	–				127	17.1	2.56	0.32
16	–				136	10.3	4.25	0.27

Table 6: Performance results for Case **T3** using Metis and Bin-sorting partitioners. Absence of data indicates that the method did not converge in 300 iterations.

	Metis Procs				Bin-sorting Procs			
	2	4	8	16	2	4	8	16
	M1							
Split Part	16	36	95	130	22	46	95	195
Part per								
Ave	158	84	50	28	161	89	50	32
Max	159	89	56	33	162	97	61	38
Min	157	80	45	24	160	80	44	25
Eqns per								
Ave	0.5254	0.2772	0.1603	0.0890	0.5341	0.2956	0.1671	0.1032
Max	0.5275	0.2825	0.1835	0.1012	0.5353	0.3204	0.1939	0.1212
Min	0.5234	0.2683	0.1467	0.0804	0.5328	0.2734	0.1509	0.0874
Data Comm per								
Ave	0.1192	0.2419	0.5057	0.6518	0.1463	0.3059	0.5356	0.8634
Max	0.1199	0.2941	0.6318	0.7709	0.1474	0.3692	0.6146	0.9161
Min	0.1184	0.1918	0.3915	0.5741	0.1451	0.2393	0.4418	0.7611
	M2							
Eqns per								
Ave	0.5045	0.2543	0.1284	0.0652	0.5117	0.2675	0.1406	0.0770
Max	0.5069	0.2577	0.1365	0.0707	0.5123	0.2742	0.1514	0.0827
Min	0.5021	0.2508	0.1225	0.0614	0.5111	0.2611	0.1372	0.0736
Data Comm per								
Ave	0.0646	0.1335	0.2903	0.4024	0.0946	0.2094	0.3960	0.7165
Max	0.0652	0.1569	0.3744	0.5113	0.0960	0.2358	0.4567	0.7929
Min	0.0640	0.1063	0.2344	0.3220	0.0932	0.1759	0.3374	0.6325
	M3							
Split Part	0	0	0	0	0	0	0	0
Particles per								
Ave	150	75	38	19	150	75	38	19
Max	150	83	46	25	150	75	39	21
Min	150	64	20	8	150	75	36	15
Eqns per								
Ave	0.5043	0.2541	0.1282	0.0650	0.5115	0.2672	0.1403	0.0786
Max	0.5067	0.2576	0.1360	0.0705	0.5121	0.2738	0.1510	0.0823
Min	0.5019	0.2506	0.1223	0.0613	0.5109	0.2610	0.1371	0.0735
Data Comm per								
Ave	0.0389	0.0791	0.1548	0.2190	0.0570	0.1346	0.2481	0.4421
Max	0.0419	0.1112	0.2727	0.3325	0.0601	0.1598	0.3095	0.5467
Min	0.0359	0.0508	0.1058	0.1248	0.0539	0.1074	0.1855	0.3376
	M4							
Eqns per								
Ave	0.5043	0.2541	0.1282	0.0650	0.5115	0.2672	0.1403	0.0786
Max	0.5067	0.2576	0.1360	0.0705	0.5121	0.2738	0.1510	0.0823
Min	0.5019	0.2506	0.1223	0.0613	0.5109	0.2610	0.1371	0.0735
Data Comm per								
Ave	0.0389	0.0791	0.1548	0.2190	0.0570	0.1346	0.2481	0.4421
Max	0.0419	0.1112	0.2727	0.3325	0.0601	0.1598	0.3095	0.5467
Min	0.0359	0.0508	0.1058	0.1248	0.0539	0.1074	0.1855	0.3376

Table 7: Distribution of particles, equations and communication for Case **T3** for Metis and Bin-sorting partitionings at time $t = 0$ seconds. The particle splittings are the same for Cases **M1** and **M2** and Cases **M3** and **M4**.

	Metis Procs				Bin-sorting Procs			
	2	4	8	16	2	4	8	16
	M1							
Split Part	18	36	85	128	11	28	51	108
Part per								
Avg	159	84	49	28	156	84	45	27
Max	164	87	57	33	157	87	53	33
Min	154	81	40	21	154	81	39	20
Eqns per								
Avg	0.5210	0.2718	0.1520	0.0835	0.5175	0.2773	0.1480	0.0851
Max	0.5233	0.2818	0.1613	0.0919	0.5179	0.2849	0.1589	0.0965
Min	0.5187	0.2672	0.1420	0.0767	0.5171	0.2674	0.1401	0.0782
Data Comm per								
Avg	0.1072	0.2101	0.4363	0.6003	0.0879	0.2139	0.3624	0.6379
Max	0.1080	0.2484	0.5986	0.6724	0.0896	0.2590	0.4601	0.7686
Min	0.1064	0.1745	0.3643	0.5057	0.0862	0.1671	0.2946	0.5491
	M2							
Eqns per								
Avg	0.5051	0.2552	0.1307	0.0670	0.5081	0.2612	0.1350	0.0720
Max	0.5072	0.2641	0.1384	0.0725	0.5082	0.2653	0.1385	0.0772
Min	0.5030	0.2470	0.1232	0.0616	0.5080	0.2567	0.1327	0.0698
Data Comm per								
Avg	0.0579	0.1174	0.2604	0.3806	0.0593	0.1389	0.2541	0.4815
Max	0.0588	0.1322	0.3730	0.4258	0.0597	0.1703	0.3158	0.5872
Min	0.0571	0.1031	0.2018	0.3112	0.0588	0.1104	0.2116	0.4097
	M3							
Split Part	0	0	0	0	0	0	0	0
Part per								
Avg	150	75	38	19	150	75	38	19
Max	157	77	49	26	152	77	43	22
Min	143	70	27	12	148	71	34	16
Eqns per								
Avg	0.5049	0.2550	0.1305	0.0668	0.5079	0.2611	0.1348	0.0719
Max	0.5070	0.2639	0.1382	0.0723	0.5081	0.2651	0.1383	0.0770
Min	0.5029	0.2468	0.1230	0.0614	0.5078	0.2565	0.1326	0.0697
Data Comm per								
Avg	0.0405	0.0785	0.1679	0.2544	0.0475	0.1084	0.1995	0.3732
Max	0.0431	0.0847	0.2225	0.3654	0.0477	0.1344	0.2316	0.4318
Min	0.0379	0.0716	0.1042	0.1749	0.0473	0.0848	0.1572	0.3131
	M4							
Eqns per								
Avg	0.5049	0.2550	0.1305	0.0668	0.5079	0.2611	0.1348	0.0719
Max	0.5070	0.2639	0.1382	0.0723	0.5081	0.2651	0.1383	0.0770
Min	0.5029	0.2468	0.1230	0.0614	0.5078	0.2565	0.1326	0.0697
Data Comm per								
Avg	0.0405	0.0785	0.1679	0.2544	0.0475	0.1084	0.1995	0.3732
Max	0.0431	0.0847	0.2225	0.3654	0.0477	0.1344	0.2316	0.4318
Min	0.0379	0.0716	0.1042	0.1749	0.0473	0.0848	0.1572	0.3131

Table 8: Distribution of particles, equations and communication for Case **T3** at time $t = 0.31$ seconds for Metis and Bin-sorting partitionings. The particle splittings are the same for Cases **M1** and **M2** and Cases **M3** and **M4**.

Because the Schur complement preconditioner is much more expensive than simple local ILU factorizations, it is not of much use in the current implementation. It would be beneficial to extend the functionality of this preconditioner to include overlapping.

5 Conclusions

We have presented an investigation on the effectiveness of several techniques for partitioning the solid particles in a solid-liquid flow interaction simulator. While the results are mixed for large numbers of particles, those obtained do indicate some general properties of the partitioner that are essential for good parallel speedup and efficiency. This study points to some of the difficulties encountered when implementing parallel iterative solvers for real life applications, and provides a few strategies to overcome them. One of the main stumbling blocks in getting good parallel efficiency is the thin line between two conflicting requirements: To be efficient the preconditioners require sufficient overlap between the subdomains, and on the other hand excessive overlap leads to excessive overhead and a deterioration of parallel efficiency. A compromise is often hard to read.

Acknowledgments. The authors wish to gratefully acknowledge many helpful discussions with Howard Hu (University of Pennsylvania), Matt Knepley (Purdue University), and Prof. Dan Joseph (University of Minnesota). We received countless hours of expert technical assistance in parallel programming from Shuxia Zhang (Minnesota Supercomputer Institute), throughout the duration of this study. Finally, this work would not have been possible without the resources and the environment provided to us by the Minnesota Supercomputing Institute.

References

[1] H. G. Choi, *Splitting method for the combined formulation of fluid-particle problem,* Submitted to Comp. Meth. Appl. Meth. Engr., 2000.

[2] H. G. Choi, D. D. Joseph, *Fluidization by lift of 300 circular particles in plane Poiseuille flow,* Submitted, 2000.

[3] H. G. Choi, H. Choi, J. Y. Yoo, *A fractional four-step finite element formulation of the unsteady, incompressible Navier-Stokes equations using SUPG and linear equal-order element methods,* Comp. Meth. Appl. Meth. Engr., 143 (1997), pp. 333-348.

[4] H. H. Hu, *Direct simulation of flows of solid-liquid mixtures,* Int. J. Multiphase Flow, 22 (1996), pp. 335-352.

[5] P. Y. Huang, H. H. Hu, D. D. Joseph, *Direct simulation of the sedimentation of elliptic particles in Oldroyd-B fluids,* Int. J. Multiphase Flow, 22 (1996), pp. 335-352.

[6] H. H. Hu, D. D. Joseph, M. J. Crochet, *Direct simulation of flows of fluid-particle motions,* Theor. Comp. Fluid Dyn., 3 (1992), pp. 285-306.

[7] A. Johnson, T. Tezduyar, *Fluid-particle simulations reaching 100 particles,* AHPCRC Preprint 97-010, Army High Performance Computing Research Center, Minneapolis, Minnesota, 1997.

[8] G. Karypis, V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs,* SIAM J. Sci. Comput., 20 (1999), pp. 359-392.

[9] B. Maury, R. Glowinski, *Fluid-particle flow: A symmetric formulation,* C.R. Acad. Sci. Paris, 324 (1997), pp. 1079-1084.

[10] R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, *A distributed Lagrange multiplier/ fictitious domain method for particulate flows,* Int. J. Multiphase Flow, 25 (1999), pp. 755-794.

[11] M. G. Knepley, A. H. Sameh, V. Sarin, *Parallel simulation of particulate flows,* 5th Intl. Symp. on Solving Irregular Structured Problems in Parallel. Springer-Verlag, (to appear), 1998.

[12] Y. Saad, A. Malevsky, *PSPARSLIB: A portable library of distributed memory sparse iterative solvers,* In V. E. Malyshkin, et. al., editor, *Proceedings of Parallel Computings Technologies (PaCT-95),* 3rd International Conference, St. Petersburg, Sept., 1995.

[13] T. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, V. Kalro, M. Litke, *High performance computing techniques for flow simulations,* AHPCRC Preprint 96-010, Army High Performance Computing Research Center, Minneapolis, Minnesota, 1996.

[14] Y. Saad, M. Sosonkina, *Distributed Schur complement techniques for general sparse linear systems,* Technical Report UMSI–97–159, Minnesota Supercomputing Institute, Minneapolis, MN, 1997.

[15] S. Kuznetsov, G.-C. Lo, Y. Saad, *Parallel solution of general sparse linear systems,* Technical Report UMSI–97–98, Minnesota Supercomputing Institute, Minneapolis, MN, 1997.

[16] G.-C. Lo, and Y. Saad, *Iterative solution of general sparse linear systems on clusters of workstations,* Technical Report UMSI–96–117, Minnesota Supercomputing Institute, Minneapolis, MN, 1996.

[17] T. Tezduyar, J. Liou, M. Behr, *A new strategy for finite element computations involving moving boundaries and interfaces–the DSD/ST procedure: I. The concept and the preliminary numerical tests,* Comp. Meth. Appl. Meth. Engr., 94 (1992), pp. 339-351.

[18] T. Tezduyar, J. Liou, M. Behr, S. Mittal *A new strategy for finite element computations involving moving boundaries and interfaces–the DSD/ST procedure: I. Computation of free-surface flows, two-liquid flows, and flows with drafting cylinders,* Comp. Meth. Appl. Meth. Engr., 94 (1992), pp. 353-371.

[19] T. Nomura, T. J. R. Hughes, *An arbitrary Lagrangian-Eulerian finite element method for interaction of fluid and a rigid body,* Comp. Meth. Appl. Mech. Engr., 95 (1992), pp. 115-138.

[20] A. Huerta, W. K. Liu, *Viscous flow with large free surface motion,* Comp. Meth. Appl. Mech. Engr., 69 (1988), pp. 277-324.

[21] R. Glowinski, T.-W. Pan, J. Périaux, *A fictitous domain method for Dirichlet problems and applications,* Comp. Meth. Appl. Mech. Engr., 111 (1994), pp. 283-303.

[22] T. W. Pan, D. D. Joseph, R. Bai, R. Glowinski, V. Sarin, *Fluidization of 1204 spheres: simulation and experiment,* Submitted.

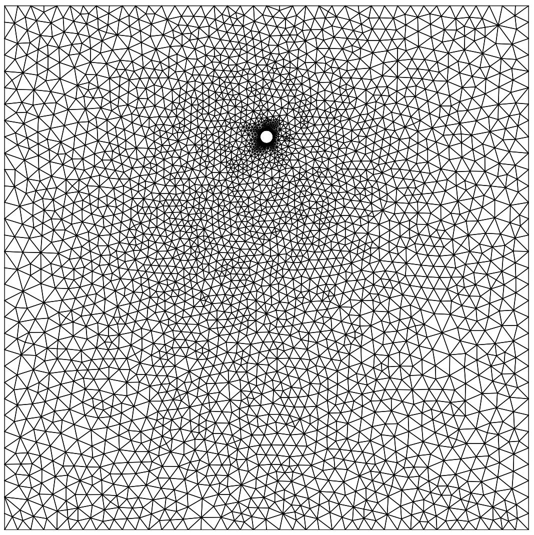


Figure 4: The initial particle configuration for Case **T1**. The mesh has 6392 elements leading to a saddle point problem size of 28634 unknowns.

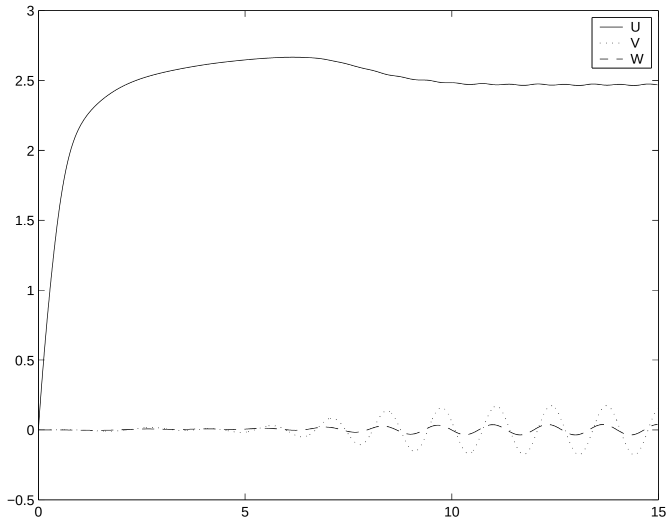


Figure 5: The time history of the velocity vector and angular velocity for the particle in test case **T1**.

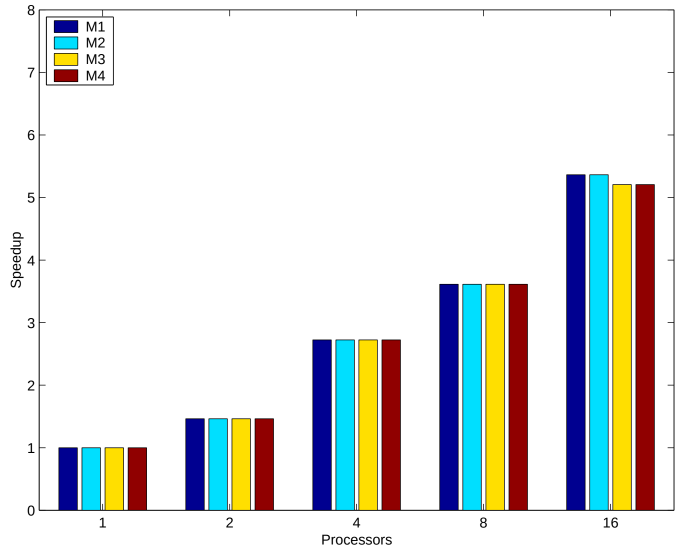


Figure 6: Parallel speedup for Case **T1** and Metis partitioning at time $t = 0$ seconds.

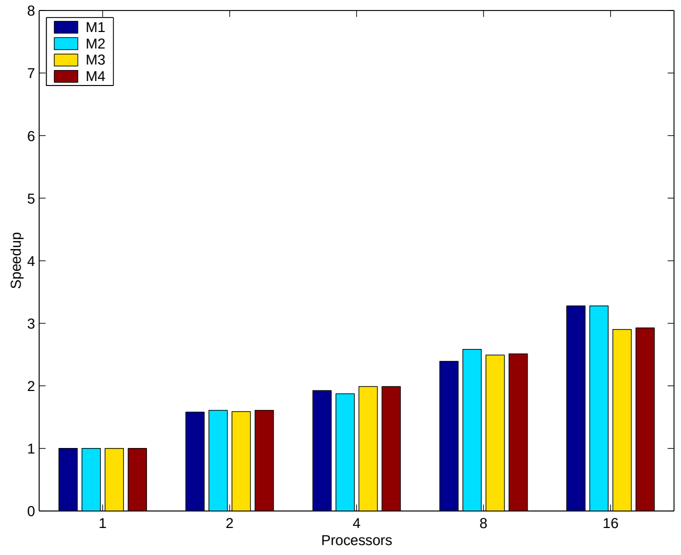


Figure 7: Parallel speedup for Case **T1** and Bin-sorting partitioning at time $t = 0$ seconds.

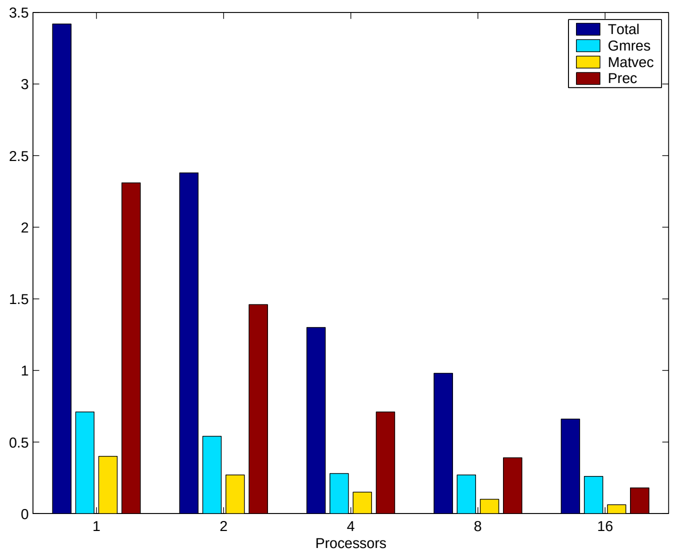


Figure 8: Absolute times for the major components of the parallel solver for Case **T1** using Method **M2** and Metis partitioning at time $t = 0$ seconds.

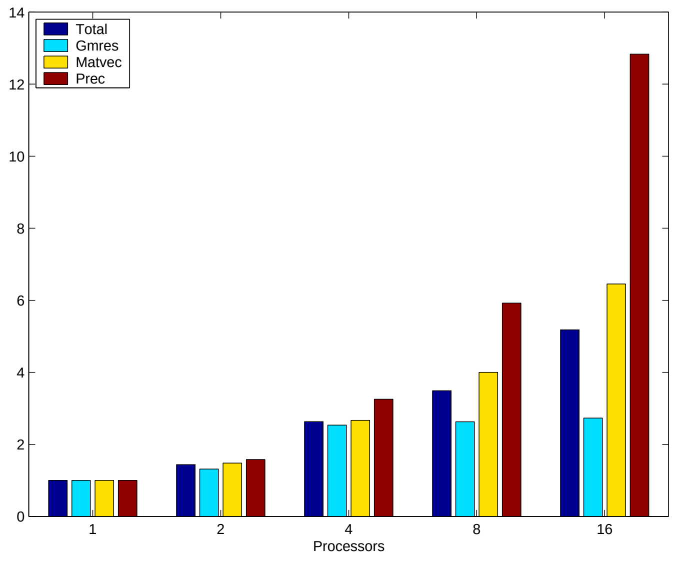
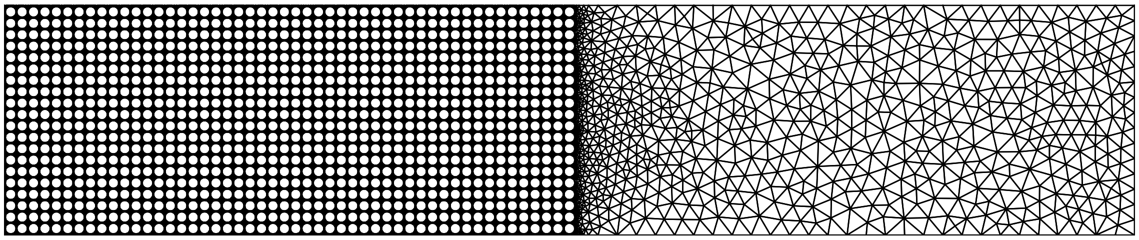
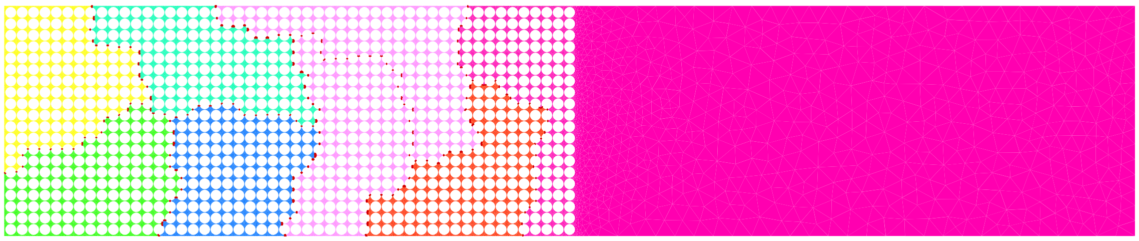


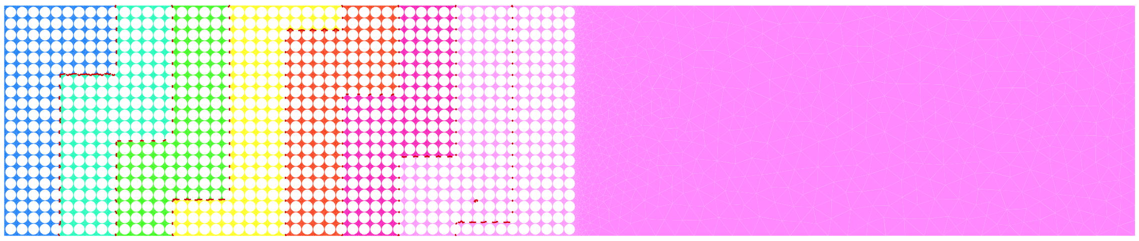
Figure 9: Scaled times for the major components of the parallel solver for Case **T1** using Method **M2** and Metis partitioning at time $t = 0$ seconds.



(a)



(b)



(c)

Figure 10: (a) The initial particle configuration for Case **T2**. Gravity is acting towards the right. The mesh has 118483 elements leading to a saddle point problem size of 468819 unknowns. (b) Metis partitioner with 8 subdomains. (c) Bin-sorting partitioner with 8 subdomains.

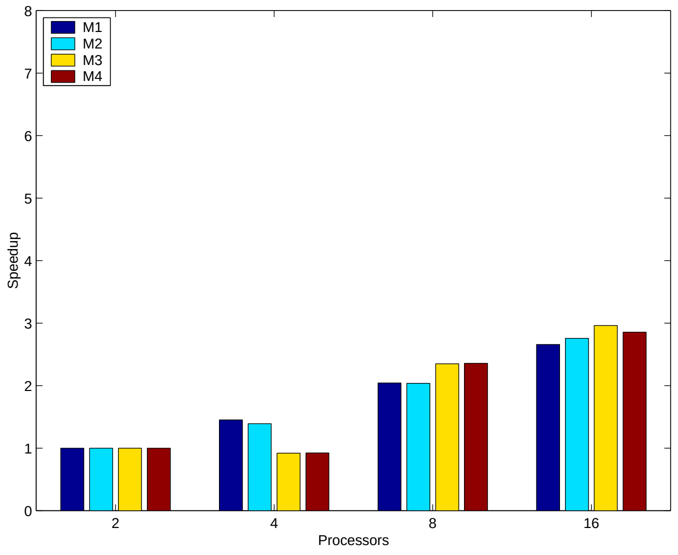


Figure 11: Parallel speedup for Case **T2** and Metis partitioning at time $t = 0$ seconds.

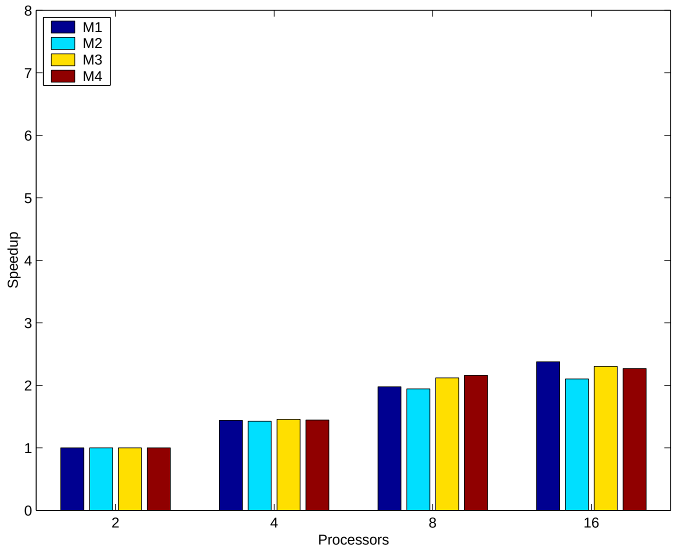


Figure 12: Parallel speedup for Case **T2** and Bin-sorting partitioning at time $t = 0$ seconds.

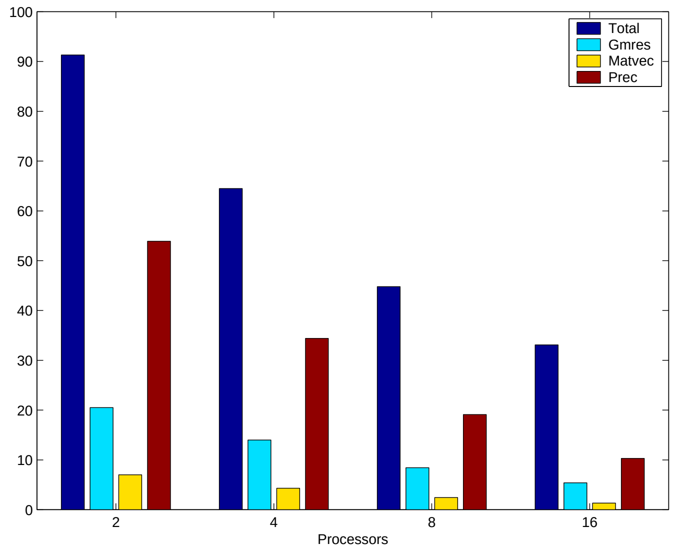


Figure 13: Absolute times for the major components of the parallel solver using Method **M2** and Metis partitioning for Case **T2** at time $t = 0$ seconds.

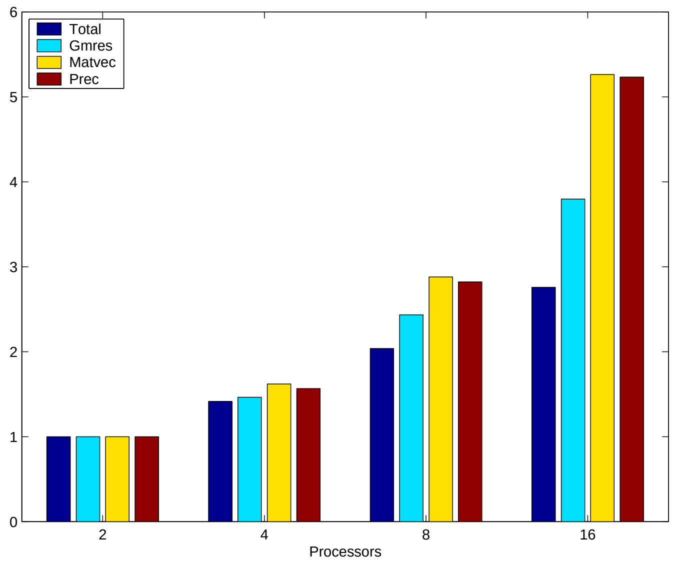
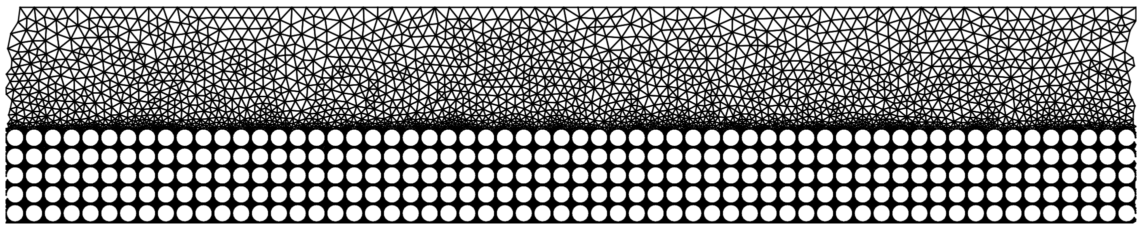
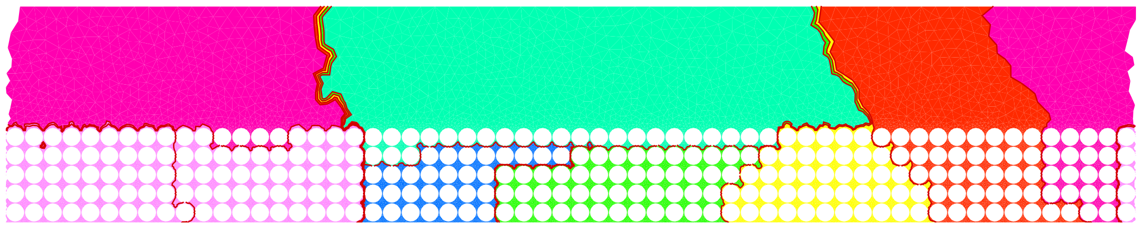


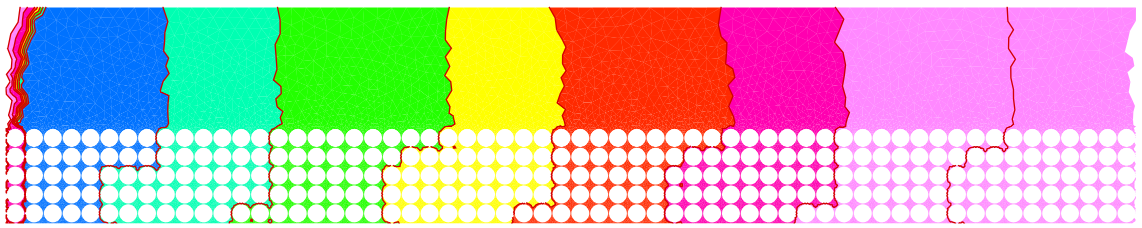
Figure 14: Scaled times for the major components of the parallel solver using Method **M2** and Metis partitioning for Case **T2** at time $t = 0$ seconds.



(a)

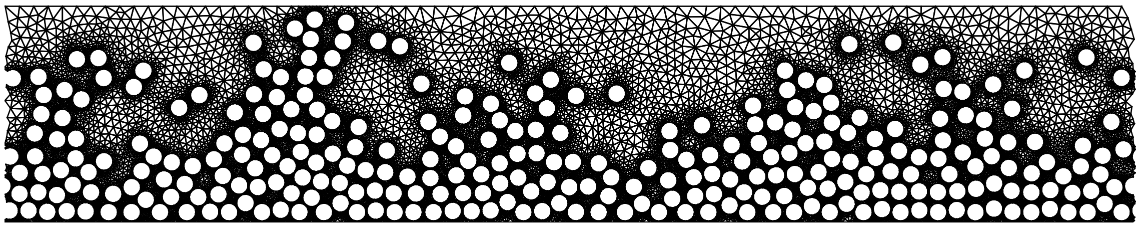


(b)

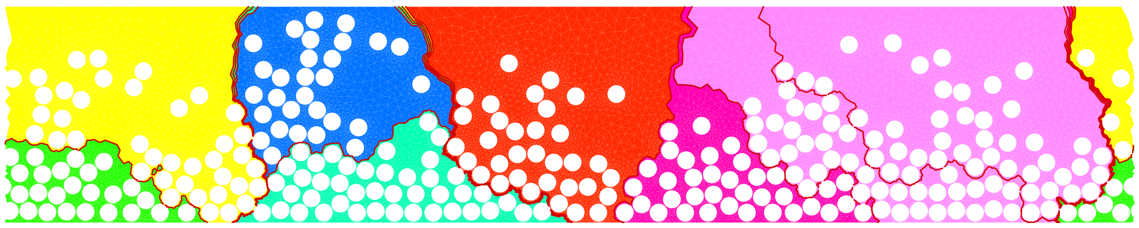


(c)

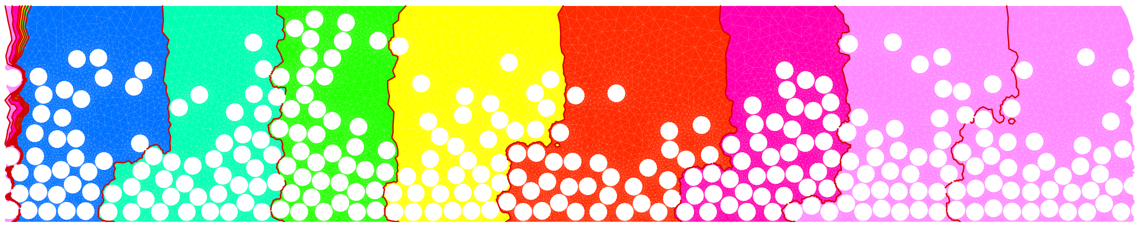
Figure 15: (a) The initial particle configuration for Case **T3**. The pressure gradient is acting towards the right. The mesh has 41509 elements leading to a saddle point problem size of 166362 unknowns. (b) Metis partitioner with 8 subdomains. (c) Bin-sorting partitioner with 8 subdomains.



(a)



(b)



(c)

Figure 16: (a) The particle configuration for Case **T3** at time $t = 0.31$ seconds. The mesh has 35701 elements leading to a saddle point problem size of 148113 unknowns. (b) Metis partitioner with 8 subdomains. (c) Bin-sorting partitioner with 8 subdomains.

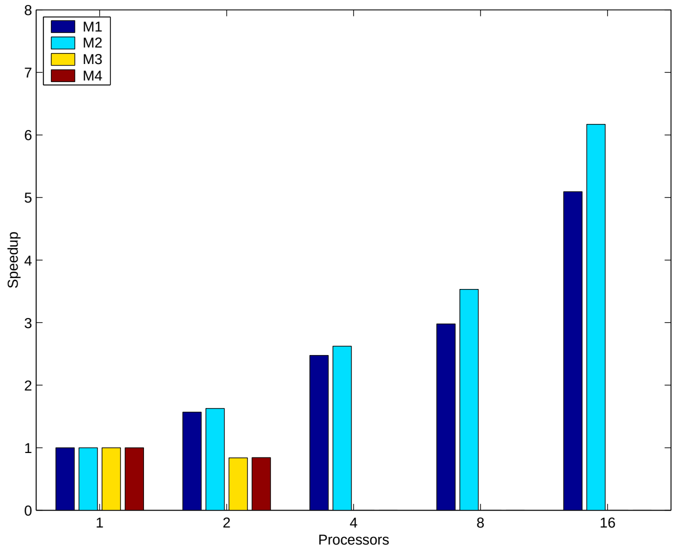


Figure 17: Parallel speedup for Case **T3** and Metis partitioning at time $t = 0$ seconds.

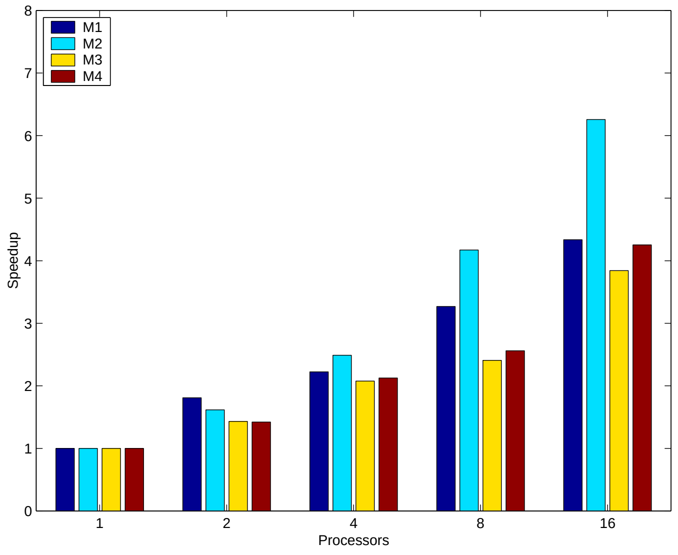


Figure 18: Parallel speedup for Case **T3** and Bin-sorting partitioning at time $t = 0$ seconds.

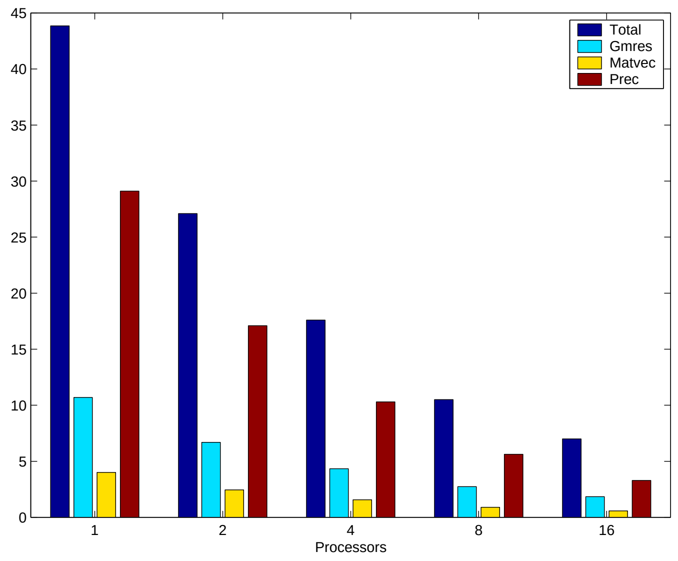


Figure 19: Absolute times for the major components of the parallel solver for Case **T3** using Method **M2** and Bin-sorting partioning at time $t = 0$ seconds.

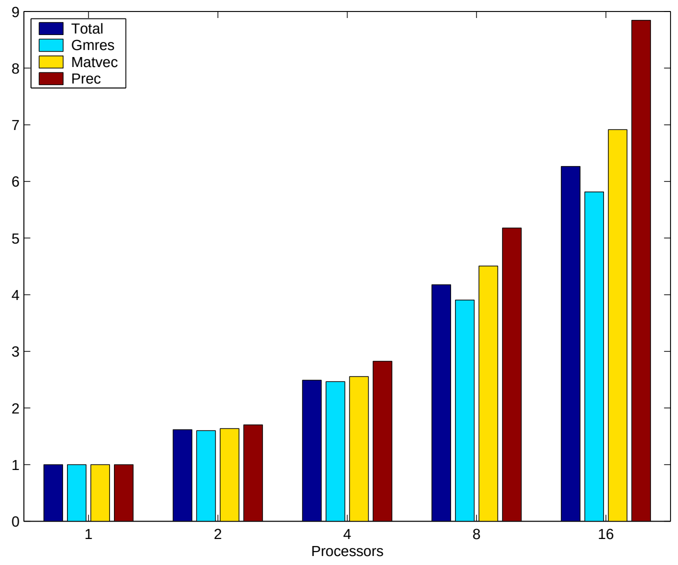


Figure 20: Scaled times for the major components of the parallel solver for Case **T3** using Method **M2** and Bin-sorting partioning at time $t = 0$ seconds.

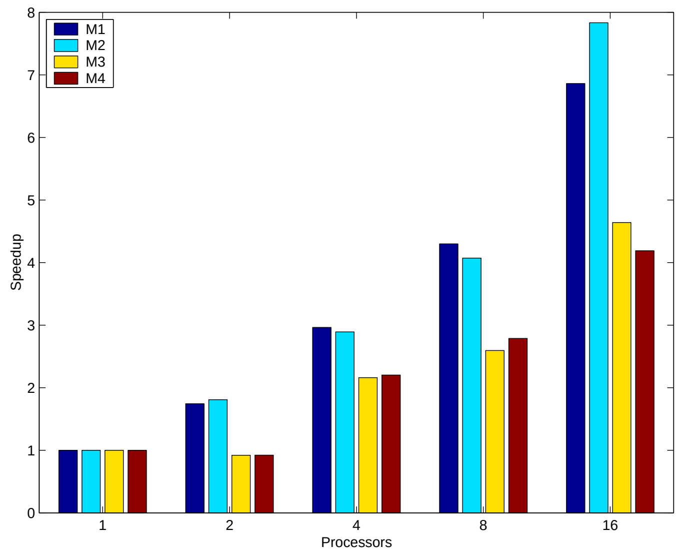


Figure 21: Parallel speedup for Case **T3** and Metis partitioning at time $t = 0.31$ seconds.

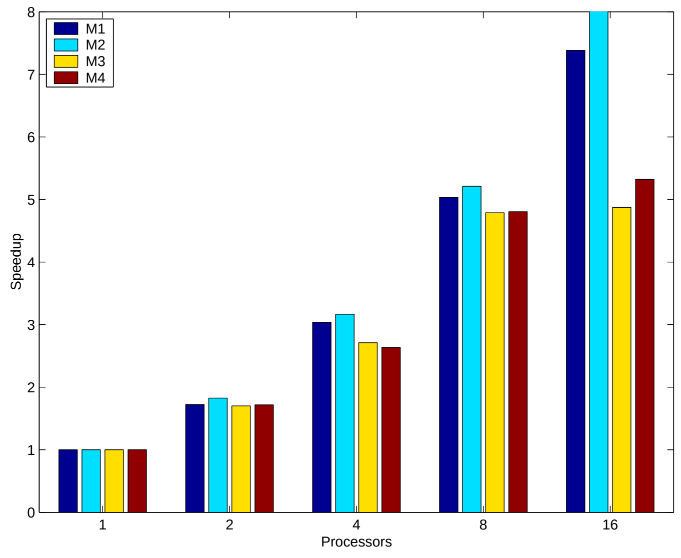


Figure 22: Parallel speedup for Case **T3** and Bin-sorting partitioning at time $t = 0.31$ seconds.