INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2005; **47**:1301–1306
Published online 24 January 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.930

Parallel ILU preconditioning and parallel mesh adaptation with load balancing for general domain decompositions for the Navier–Stokes equations

Ørnulf Staff*,† and S. Ø. Wille

Faculty of Engineering, Oslo University College, Norway

SUMMARY

A complete framework for solving the incompressible Navier–Stokes equations in parallel is presented. An unstructured mesh is decomposed into non-overlapping subdomains corresponding to the number of processors. Each subdomain is adaptively refined independently based on local Reynolds number estimates. Computational load is balanced by transferring element-octrees between subdomains.

A parallel conjugate gradient solver with ILU preconditioning is achieved by resolving node dependencies based on mesh structure. Each node is sorted by category giving an *a priori* pivoting suited for parallel solution. The parallel solver has convergence rates comparable to serial solvers with a similar ILU strategy. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: variable segregation; ILU preconditioning; Stokes equations; Navier–Stokes equations; domain decomposition

1. INTRODUCTION

Simulations on single processors are often limited by CPU speed and available central memory. Even fairly modest three-dimensional problems can surpass what can be solved on a single processor in a reasonable amount of time. Parallel multiprocessing is a frequently used strategy for overcoming such limits [1–3]. In this work a parallel finite element solver for a stationary and incompressible formulation of the Navier–Stokes equations is presented. The solver utilizes ILU preconditioning, *a priori* pivoting and segregation of variables. An unstructured mesh is split into a number of subdomains (see Figure 1). Earlier work [4] required the subdomains to be slices of the global domain with unconnected interfaces. A more general algorithm has been devised to allow arbitrary domain partitioning.

*Correspondence to: Ørnulf Staff, Faculty of Engineering, Oslo University College, Cort Adelersgate 30, N-0254 Oslo, Norway.
†E-mail: ornulfs@iu.hio.no

Received 27 April 2004
Revised 15 December 2004

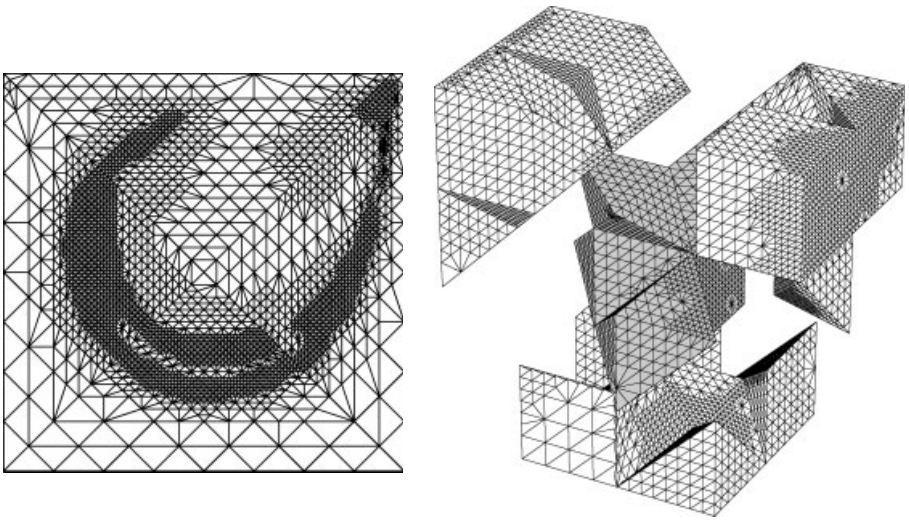

Figure 1. Unstructured adaptive mesh for a lid-driven cavity at Reynolds number 800. The left figure shows the adaptive structure of a 2D mesh. The right figure is a corresponding 3D mesh split into 4 subdomains for parallel solution on 4 processors. Synchronization of interface nodes is done by message passing.

The model problem is the stationary incompressible Navier–Stokes equations on a lid-driven cavity. A constant velocity with zero y component is specified at the lid. All other boundaries have no-slip zero velocities imposed.

2. NUMERICAL METHODS

The structure of a finite element equation matrix is determined by mesh structure and node numbering. Node numbers are arbitrary, and any node pivot will produce a valid equation system. The matrix structure for one possible pivot of the nodes of a submesh of Figure 1 is shown in Figure 2. Nodes internal to the submesh are sorted spatially with regard to the centre of the submesh. Nodes on interfaces to other submeshes are sorted by the highest numbered submesh they belong to.

A reasonable ILU fill-in strategy is for the decomposition to maintain the same structure as the assembled finite element matrix [5]. To perform parallel ILU, nodes and their corresponding matrix rows are split in three categories:

1. *Internal*: Nodes internal to the submesh.
2. *Receive*: Nodes where the submesh is the highest numbered submesh containing the node.
3. *Transmit*: Nodes where some other higher numbered submesh also contains the node.

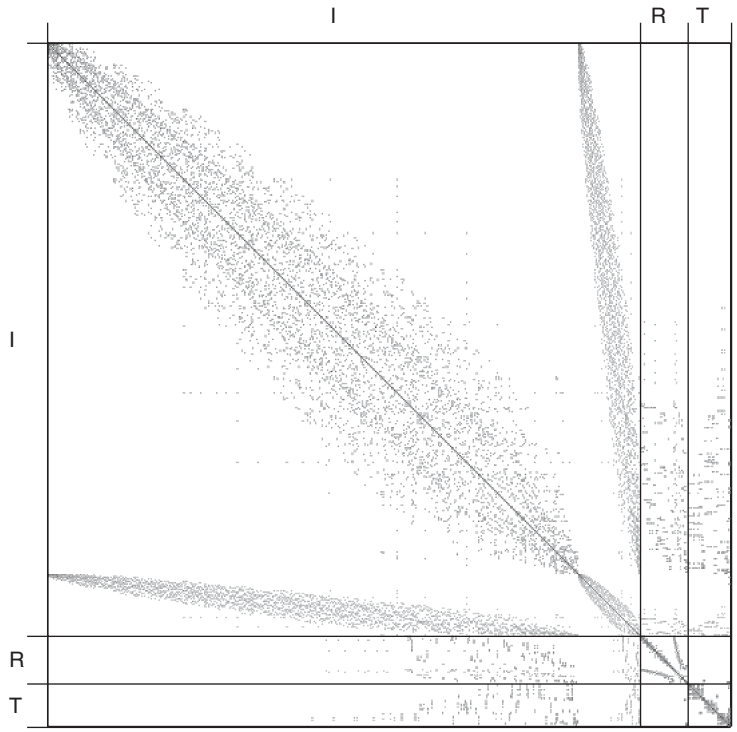

Figure 2. The matrix structure for mesh 2 in Figure 1. Each non-zero value in the matrix is marked with a dot. Nodes internal to the submesh are labelled I, those on receive interfaces are labelled R and transmit interfaces are labelled T. The submatrices connecting receive and transmit nodes are almost entirely zero. This allows much of the synchronization between subdomains to happen in parallel.

Internal nodes are independent and are handled in parallel on all processors. Receive nodes are handled after data has been received from all other submeshes containing the nodes. Data for transmit nodes is sent when all dependencies to internal and receive nodes have been resolved. Good parallel performance stems from the fact that the receive and transmit interfaces are largely independent of one another. Most data can be sent *before* waiting for data for the receive nodes.

The parallel preconditioner and the subsequent conjugate gradient solver perform operations on an implicitly formed matrix of the entire domain. For any given mesh, subdomain partitioning affects only the pivot of this implicit matrix.

The information required to produce a pivot vector and determine all transmit dependencies are contained locally in each submesh. Parallel performance is affected by the number of interface nodes. However, dependencies are completely resolved and the solver will produce a valid ILU preconditioner for any subdomain partitioning.

Good performance depends on an approximately equal number of nodes in each submesh. Because the solver does not impose any restriction on valid partitioning, any dynamic load balancer can be used [6].

3. EFFICIENCY AND SCALABILITY

The solution for a 3D driven cavity at Reynolds number 800 is shown in Figure 3. The mesh used is akin to those in Figure 1. They are adaptively refined with regard to element Reynolds number estimates. Refinement thus occur in the upper corners and along the main vortex inside the cavity. The 3D cavity in Figure 1 is divided in four subdomains suitable for parallel computation on four processors. Each subdomain is balanced with its neighbours to contain a similar number of elements while minimizing the number of interface nodes. This will tend to produce continuous subdomains with a high ratio of internal nodes to interface nodes. However, the solver will function for any partitioning of the domain and does not depend on continuous subdomains.

Computational overhead is a critical component in any parallel solver. Scalability is greatly affected if the parallel iterative solver requires more iterations than its serial counterpart in order to reach an acceptable solution quality. Figure 4 shows the proposed ILU strategy to be largely independent of the number of subdomains. Within reasonable ratios between mesh sizes and number of processors, preconditioner quality is not reduced by parallelization.

Communication overhead is a function of per-message network latency and the number of interface nodes. Message latency overhead depends on the number of messages sent, which is fairly constant. The number of interface nodes grows slower than the number of internal nodes. Because computational load is a function of the total number of nodes, good parallel performance can be achieved on almost any hardware with a sufficiently large number of elements.

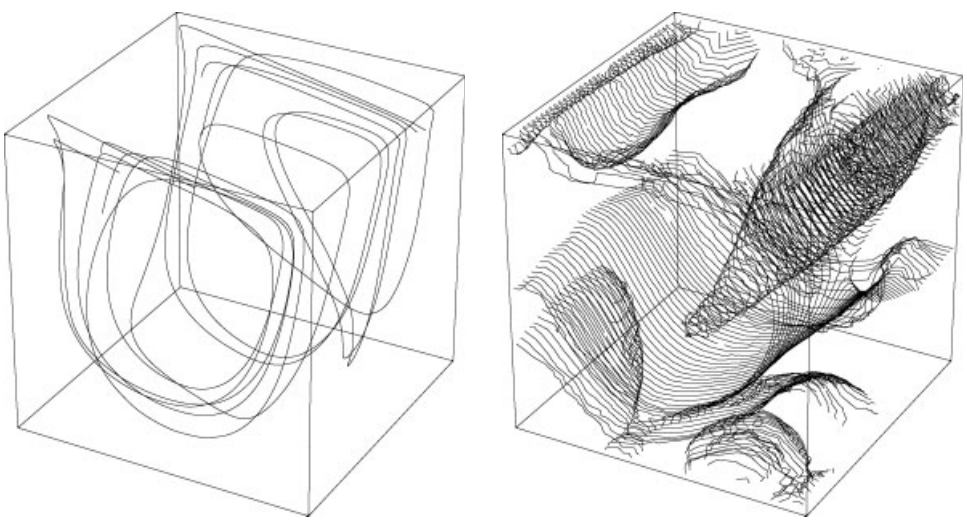

Figure 3. Velocity streamlines and pressure isobars for a 3D driven cavity at Reynolds number 800. The lid is driven by a velocity with at 35° angle to the x-axis. Solving is performed by an unstructured domain decomposition as illustrated in Figure 1. The streamlines profile clearly show the main vortex inside the cavity. The upper left and upper right corners are regions of low and high pressure, respectively.

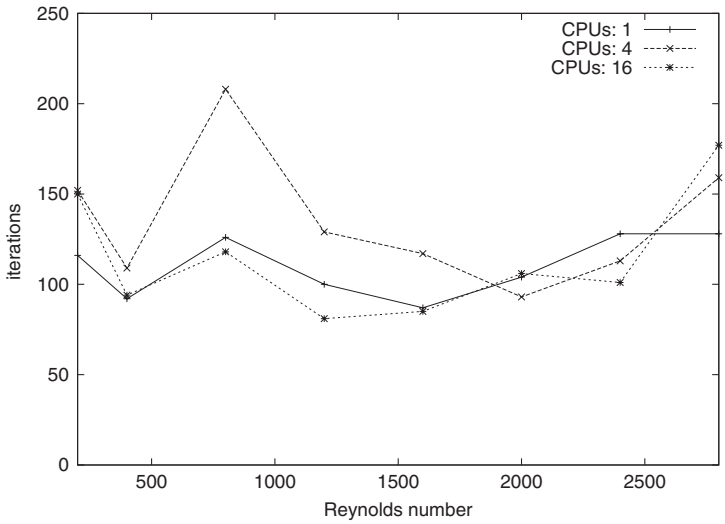

Figure 4. Required number of iterations to achieve $\|Ax-b\|<5.0\times10^{-6}$. Continuation is performed by scaling the previous solution when the velocity is increased. Each Reynolds number is calculated with three adaptive grid refinements and five Newton iterations per refinement.

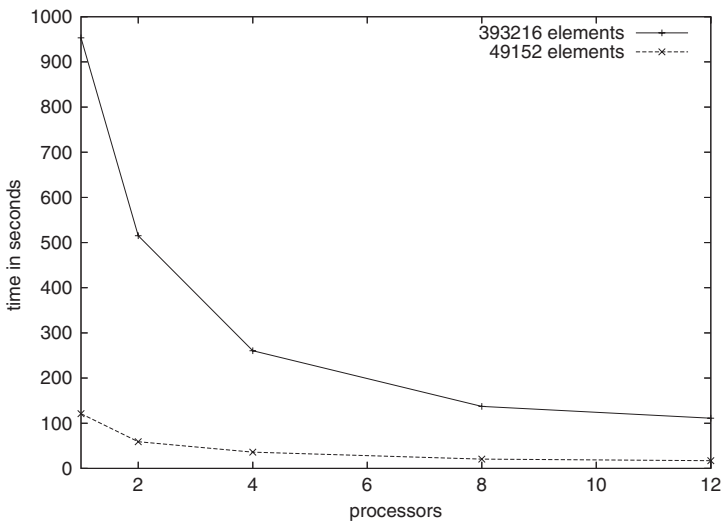

Figure 5. Runtime in seconds for three 3D uniform mesh resolutions as a function of number of processors. Speedup on 12 processors is 7.1 for the small problem and 8.6 for the large.

Experiments show that the algorithms are suitable for implementation on non-specialized parallel hardware. Trials were conducted on a cluster of identical computers running Linux. Each computer was equipped with 2.8 GHz Intel Pentium IV processor and 1 GB of RAM. They were connected through a standard 100 mbit switched Ethernet.

The runtimes for 2D meshes with different mesh resolutions are shown in Figure 5. The measurements were performed on uniform non-adaptive meshes. For an adaptive mesh over a given problem, the element count is largely a function of velocity and thus Reynolds number.

The meshes used for measurement in Figure 5 consisted of 49 152 and 393 216 elements. In comparison, the solution with Reynolds number 800 in Figure 3 consisted of 204 928 elements, using an adaptively refined mesh.

The time measurements show an increased reduction in parallel runtime for larger problem sizes. While the coarsest meshes do not benefit from more than 4 processors, higher mesh resolutions scale well up to 12 processors.

4. DISCUSSION

Through *a priori* node pivoting, serial dependencies between interface nodes can be minimized in order to allow a solver with largely parallel synchronization of subdomains. All decisions to construct the pivot are based on locally available information. There is no need for a central controlling processor.

The parallel solver constructs dependencies for any subdomain partitioning and is therefore well suited for adaptive mesh generators with parallel refinement. Mesh generation and load balancing are not burdened with any additional constraints imposed by the equation solver.

The considered ILU performs fill-in based on mesh structure. While practical for parallelization, this strategy is not a strict requirement. Dependency calculations could be performed *after* an ILU with adaptive fill-in of the internal nodes. The dependencies would then be a function of matrix connectivity rather than mesh structure. Thus, any ILU strategy could theoretically be used with structural limitations imposed only on interface nodes.

REFERENCES

1. Johnson A, Tezduyar T. Methods for 3D computation of fluid-flow interactions in spatially periodic flows. *Computer Methods in Applied Mechanics and Engineering* 2001; **190**:3201–3221.
2. Gropp WD, Kaushik DK, Keyes DE, Smith BF. Analyzing the parallel scalability of an implicit unstructured mesh cfd code. *Lecture Notes in Computer Science* 2001; **1970**:395–404.
3. Bauer AC, Patra AK. Performance of parallel preconditioners for adaptive hp fem discretization of incompressible flows. *Communications in Numerical Methods in Engineering* 2002; **18**:305–313.
4. Wille SØ, Staff Ø, Loula AFD. Block and full matrix ILU preconditioners for parallel finite element solvers. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**(13–14):1381–1394.
5. Wille SØ, Staff Ø, Loula AFD, Carey GF. The influence of the order of fill-in on the convergence rate for ILU preconditioned iterative solvers. *International Journal of Numerical Methods for Heat and Fluid Flow* 2004; **14**(3):325–340.
6. Arulananthan A, Johnson S, McManus K, Walshaw C, Cross M. A generic strategy for dynamic load balancing of distributed memory parallel computational mechanics using unstructured meshes. In *Parallel Computational Fluid Dynamics*: *Recent Developments and Advances Using Parallel Computers*, Emerson DR *et al.* (eds). Elsevier: Amsterdam, 1998; 43–50 (*Proceedings of the Parallel CFD'97*, Manchester, 1997).

