35

Comparison of three algorithms for nonlinear metal cutting problems

C.J.Palansuriya[1], C-H.Lai[2], C.S.Ierotheou[3], K.A.Pericleous[4], & D.E.Keyes[5]

Introduction

The properties of a metal can alter and the quality of the cut may degrade considerably due to high temperatures generated in a cutting process. Therefore, the determination of the temperature generated in applying a cutting tool is of industrial interest. The cutting process can be regulated, by actively controlling the speed of the cutter, as well as the application of coolant fluid at the cutter points. This active control may also lengthen the lifetime of the cutter. The active control of the cutting process requires real-time simulation of the temperature distribution. This implies the requirement of fast algorithms.

This paper compares three alternative numerical algorithms applied to a nonlinear metal cutting problem. One algorithm is based on an explicit method [PLIP98] and the other two are implicit. Domain decomposition (DD) is used to break the original domain into subdomains, each containing a properly connected, well-formulated and continuous subproblem. The serial version of the explicit algorithm is implemented in

1 School of Computing & Mathematical Sciences, University of Greenwich, Wellington Street, Woolwich, London SE18 6PF, U.K. email: C.J.Palansuriya@grn.ac.uk
2 Address same as above.
3 Address same as above.
4 Address same as above.
5 Computer Science Department, Old Dominion University, Norfolk, VA 23529-0162 USA and Inst. for Computer Applics. in Science and Engineering, NASA Langley Research Center, MS 403, Hampton, VA 23681-2199 USA

Eleventh International Conference on Domain Decomposition Methods
Editors Choi-Hong Lai, Petter E. Bjørstad, Mark Cross and Olof B. Widlund ©1999 DDM.org

FORTRAN and its parallel version uses MPI (Message Passing Interface) calls. One implicit algorithm is implemented by coupling the state-of-the-art PETSc (Portable, Extensible Toolkit for Scientific Computation) [BGLS] software with in-house software in order to solve the subproblems. The second implicit algorithm is implemented completely within PETSc. PETSc uses MPI as the underlying communication library. Finally, a 2D example is used to test the algorithms and various comparisons are made.

The dimensionless 2d nonlinear metal cutting problem

The metal cutting problem considered here is a 2D thin sheet of metal defined in the domain $D = \{(x, y) : 0 < x < 1 \text{ and } 0 < y < 1\}$. The material properties are assumed to be homogeneous across the domain of interest and the following assumptions are made for idealised cutting :- (a) the application of a cutting tool at the cutter points is equivalent to the application of a heat source at these points, (b) no phase changes occur during cutting and (c) the thickness of the cutter is negligible. The cutting is considered to be applied along the y-axis at $x = x_c$. These assumptions lead to the following dimensionless 2D nonlinear, unsteady, parabolic, heat conduction equation,

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}(k(u)\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(k(u)\frac{\partial u}{\partial y}) + Q_c(y,t)\delta(x - x_c) \in D, \qquad (1)$$

subject to initial condition $u(x, y, 0) = U_i(x, y)$, boundary conditions $u(0, y, t) = B_0(y, t)$, $u(1, y, t) = B_1(y, t)$, $u(x, 0, t) = C_0(x, t)$ and $u(x, 1, t) = C_1(x, t)$. Here $u(x, y, t)$ is a dimensionless temperature distribution, $k(u)$ is the conductivity of the metal, $Q_c(y, t)$ is the unknown source being applied at $x = x_c$, $\delta(x - x_c)$ is the Dirac delta function and U_i, B_0, B_1, C_0 and C_1 are known functions. The unknown source strength may be retrieved by integrating (1) from $x = x_c^-$ to $x = x_c^+$ [PLIP98]. Temperature sensors which are located at $x = x_s$, where $0 < x_c < x_s < 1$, are used to record the temperature variation.

For simulation purposes, the sensor temperatures are modelled by using a periodic function $u(x_s, y, t) = \alpha y(y - 1)^2 sin(\omega t)$. The maximum value is generated by the amplitude α, and the variation with respect to time is generated by the angular frequency ω.

Problem partitioning

Problem partitioning is by a DD method applied at the mathematical/physical problem level [Lai94]. This decomposition provides the primary level of parallelism to the algorithms as discussed in Section 35. In order to solve the inverse problem given in (1) with the additional condition available at $x = x_s$, problem partitioning is carried out to produce three subdomains, such that each subproblem may be solved using a different numerical algorithm. The three subdomains are:

$D_1 = \{(x, y) : 0 < x < x_s \text{ and } 0 < y < 1\}$
$D_2 = \{(x, y) : x_s < x < x_c \text{ and } 0 < y < 1\}$
$D_3 = \{(x, y) : x_c < x < 1 \text{ and } 0 < y < 1\}$

This problem partitioning removes the unknown source term $Q_c(y,t)$ and the Dirac delta function which are associated with the differential equation. The three subproblems (SPs) can be written as follows:

$$SP_1: \quad \frac{\partial u_1}{\partial t} = \frac{\partial}{\partial x}(k(u_1)\frac{\partial u_1}{\partial x}) + \frac{\partial}{\partial y}(k(u_1)\frac{\partial u_1}{\partial y}) \in D_1$$
$$\text{subject to } u_1(x,y,0) = U_i(x,y),\ u_1(0,y,t) = B_0(y,t),$$
$$u_1(x_s,y,t) = u^*(y,t),\ u_1(x,0,t) = C_0(x,t),\ u_1(x,1,t) = C_1(x,t).$$

$$SP_2: \quad \frac{\partial u_2}{\partial t} = \frac{\partial}{\partial x}(k(u_2)\frac{\partial u_2}{\partial x}) + \frac{\partial}{\partial y}(k(u_2)\frac{\partial u_2}{\partial y}) \in D_2$$
$$\text{subject to } u_2(x,y,0) = U_i(x,y),\ u_2(x_s,y,t) = u^*(y,t),$$
$$\frac{\partial u_2(x_s,y,t)}{\partial x} = \frac{\partial u_1(x_s,y,t)}{\partial x},\ u_2(x,0,t) = C_0(x,t),\ u_2(x,1,t) = C_1(x,t).$$

$$SP_3: \quad \frac{\partial u_3}{\partial t} = \frac{\partial}{\partial x}(k(u_3)\frac{\partial u_3}{\partial x}) + \frac{\partial}{\partial y}(k(u_3)\frac{\partial u_3}{\partial y}) \in D_3$$
$$\text{subject to } u_3(x,y,0) = U_i(x,y),\ u_3(x_c,y,t) = u_2(x_c,y,t),$$
$$u_3(1,y,t) = B_1(y,t),\ u_3(x,0,t) = C_0(x,t),\ u_3(x,1,y) = C_1(x,t).$$

Since the temperature values are given at $y = 0$, $y = 1$, $x = 0$ and there are temperature sensors located at $x = x_s$, Dirichlet boundary conditions are defined at the boundary of D_1. The solution of the differential equation provides the required data to calculate the heat flux $\frac{\partial u}{\partial x}(x_s,y,t)$. Therefore, with the knowledge of the temperatures $u(x_s,y,t)$ acquired by the temperature sensors at $x = x_s$, an initial value problem can be formulated in D_2. $u(x_c,y,t)$ values may be obtained by solving this initial value problem. Finally, with the calculated temperatures $u(x_c,y,t)$, another Dirichlet problem can be formulated in D_3. The above three subproblems are well-defined [BBSJ85] [Zwi89]. Hence a unique solution exists for each subproblem and the union of these gives the temperature distribution of the original problem.

Algorithms

Algorithm 1

To solve the probelms in SP_1 and SP_3, a first-order forward difference approximation of the temporal derivative and a second-order Finite Volume (FV) approximation of the spatial derivatives leads to a five-point explicit scheme. Dropping the subscript used in denoting the subdomains, the explicit scheme for the subdomains D_1 and D_3 can be written as,

$$u_{i,j}^{(n+1)} = r_x b_i^{(n)} u_{i-1,j}^{(n)} + r_x a_i^{(n)} u_{i+1,j}^{(n)} + (1 - r_x a_i^{(n)} - r_x b_i^{(n)} - r_y c_j^{(n)} - r_y d_j^{(n)}) u_{i,j}^{(n)} + r_y d_j^{(n)} u_{i,j-1}^{(n)} + r_y c_j^{(n)} u_{i,j+1}^{(n)} \tag{2}$$

where (i,j) denotes the (i,j)-th grid point, $r_x = \frac{\Delta t}{(\Delta x)^2}$, $r_y = \frac{\Delta t}{(\Delta y)^2}$, $a_i^{(n)} = \frac{k_{i+1,j}^{(n)} + k_{i,j}^{(n)}}{2}$, $b_i^{(n)} = \frac{k_{i-1,j}^{(n)} + k_{i,j}^{(n)}}{2}$, $c_j^{(n)} = \frac{k_{i,j+1}^{(n)} + k_{i,j}^{(n)}}{2}$, $d_j^{(n)} = \frac{k_{i,j-1}^{(n)} + K_{i,j}^{(n)}}{2}$, (n) denotes the time-step, Δt is the step size along the temporal axis and Δx, Δy are the grid spacings along the spatial axis x, y, respectively. The initial value problem in SP_2 is solved by employing a second-order Euler Predictor-Corrector (P-C) method along the x-

axis for each time-step. Again, the spatial derivatives are discretised using second-order FV approximations and the time derivative with a first-order finite difference approximation. The two step P-C method can be written as:

$$\begin{pmatrix} u \\ v \end{pmatrix}^* = \begin{pmatrix} u \\ v \end{pmatrix} + \triangle x \underline{f}\ , \quad \begin{pmatrix} u \\ v \end{pmatrix}^{\text{new}} = \begin{pmatrix} u \\ v \end{pmatrix} + \frac{\triangle x}{2}\{\underline{f} + \underline{f}^*\}\ , \tag{3}$$

where $v = \frac{\partial u}{\partial x}$, $\underline{f} = \underline{f}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} v \\ \frac{1}{k(u)}(\frac{\partial u}{\partial t} - \frac{\partial}{\partial y}(k(u)\frac{\partial u}{\partial y}) - k'(u)v^2) \end{pmatrix}$ and $\underline{f}^* = \underline{f}\begin{pmatrix} u \\ v \end{pmatrix}^*$. A second order spatially accurate solution may be obtained for each of the three subproblems. Therefore, it is expected to have a second order spatially accurate global solution for the inverse problem (1). The effect of the local truncation error for SP_2 is minimised because of the small size of the subdomain which usually consists of only a few Euler P-C steps. All experiments showed stable results as long as the CFL condition r_x, $r_y \leq 0.25$ was satisfied.

Algorithm 2

Newton's method is applied to D_1 and D_3 and leads to an implicit method. Using Newton's method, the iterative scheme for solving an equation of the type $F(U) = 0$ may be implemented as,

```
{ Solve
      $F'(U_m)V_m = -F(U_m)$
  Update
      $U_{m+1} = U_m + V_m$
} Repeat until $||V|| < Tol$
```

For the metal cutting problem, $F(U) = \frac{\partial}{\partial x}(k(U)\frac{\partial U}{\partial x}) + \frac{\partial}{\partial y}(k(U)\frac{\partial U}{\partial y}) - \frac{\partial U}{\partial t}$ and $F'(U)$ is,

$$\begin{aligned} F'(U) = {} & k''(U)(\tfrac{\partial U}{\partial x})^2 + 2k'(U)\tfrac{\partial U}{\partial x}\tfrac{\partial}{\partial x} + k'(U)\tfrac{\partial^2 U}{\partial x^2} + k(U)\tfrac{\partial^2}{\partial x^2} \\ & + k''(U)(\tfrac{\partial U}{\partial y})^2 + 2k'(U)\tfrac{\partial U}{\partial y}\tfrac{\partial}{\partial y} + k'(U)\tfrac{\partial^2 U}{\partial y^2} + k(U)\tfrac{\partial^2}{\partial y^2} - \tfrac{\partial}{\partial t} \end{aligned}$$

Subdomain D_2 uses the same Euler P-C method as in Algorithm 1. This algorithm is fully implicit and PETSc is used to solve the linearised systems in D_1 and D_3.

Algorithm 3

All three subdomains are solved by using the Newton's method as described in algorithm 2. This algorithm leads to a global linearised system for all three subdomains and they are solved using PETSc.

The implementation of the above three algorithms highlights the fact that the generation of homogeneous and continuous subproblems due to problem partitioning facilitates the use of general purpose libraries, as well as the coupling of subdomain-specialized software solutions.

Exploiting parallelism

The problem partitioning is referred to as "domain parallelism" [ILPP98]; using this concept, each subdomain generated due to the problem partitioning can be mapped directly to a processor and these subproblems may be solved concurrently. However, domain parallelism has an obvious limitation in that it is limited by the number of subdomains and therefore it does not scale with an increasing number of processors. Data partitioning may then be carried out within each subdomain and this is referred to as "domain-data parallelism" [ILPP98].

The parallel implementation of Algorithm 1 is carried out as follows. D_1 is assigned to one group of processors and D_2 and D_3 are assigned to another group of processors, see [PLIP98, ILPP98] for further explanation. Each processor in a group exploits the data-parallelism within the subdomain or subdomains it owns. For Algorithm 2, three groups of processors are created, one subdomain for each group. In order to re-use the in-house software for initial value problem, D_2 is isolated. Again, each processor within a group exploits the data-paralellism in the subdomain it owns, see Figure 1 (dotted lines represent data partitioning). In Algorithms 1 and 2, there is a homogeneous data dependency within each subdomain. Therefore, data partition along x- and y- axises can be carried out arbitrarily within each subdomain.

In Algorithm 3, all three subdomains are solved using one global linearised system. Therefore, data parallelism is used to partition the global linearised system into blocks. However, data partitioning along the x-axis may produce blocks with different sparsity structures, depending upon how D_2 is allocated between the blocks, and therefore with potentially different load balances and interprocessor message patterns. Hence, we only partition along the y-axis.

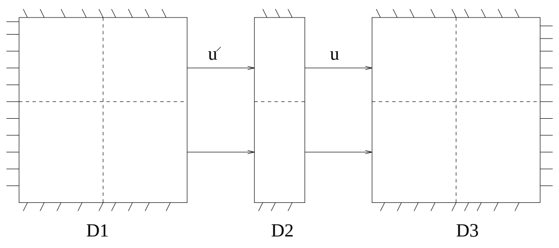

Figure 1 An example of partition for Algorithm 2

Numerical results

Thermal results are obtained for equation (1) with $x_s = 0.5$, $x_c = 0.6$, $U_i(x, y) = 0$, $B_0(y, t) = 0$, $B_1(y, t) = 0$, $C_0(x, t) = 0$ and $C_1(x, t) = 0$. The sensor points are modelled as $u^*(y, t) = \alpha y(y - 1)^2 \sin(\omega t)$, with $\alpha = 0.1$ and $\omega = 2\pi$. The nonlinear heat conductivity is given by $k(u) = \frac{1}{1+u^2}$. The temperature fields and the retrival of source/sink strength may be found in [PLIP98]. Figure 2(a) shows the numerical comparison between the three Algorithms, which shows virtually the same numerical results. The sequential implementation of the algorithms was tested for performance in a Sun Sparc 5 workstation. The runtimes are shown in Figure 2(b). Algorithm 1 does not perform very well since it is using a very small $\triangle t$, necessary in order to satisfy the CFL condition. There is no significant difference between the discretizations of Algorithm 2 and Algorithm 3. The latter is more efficient since it solves the subproblems using one system matrix, with a better load balance.

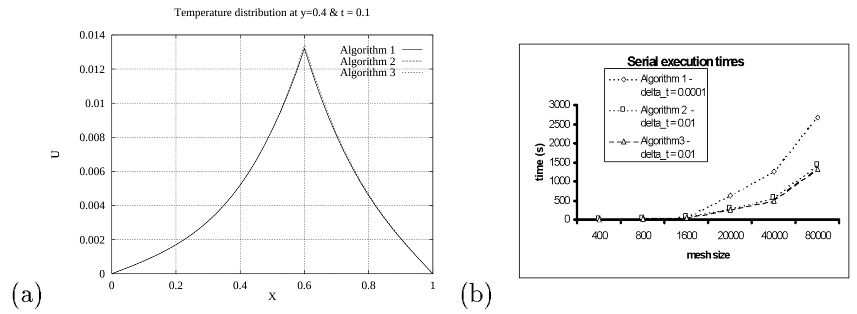

Figure 2 Horizontal solution profile and serial execution times for all three algorithms.

The parallel implementations of the Algorithms were tested on a network of Sun Sparc 5 workstations connected together by an ethernet network. The execution times and the speedups are given in Figures 3, 4 and 5. Algorithm 1 shows very good scalability with increasing number of processors. This is due to the explicit methods in the Algorithm which avoid global synchronisations. Only halo (ghost) points need to be communicated between processors. The other two Algorithms require additional communications in each step of the iterative scheme that lead to more moderate speedups.

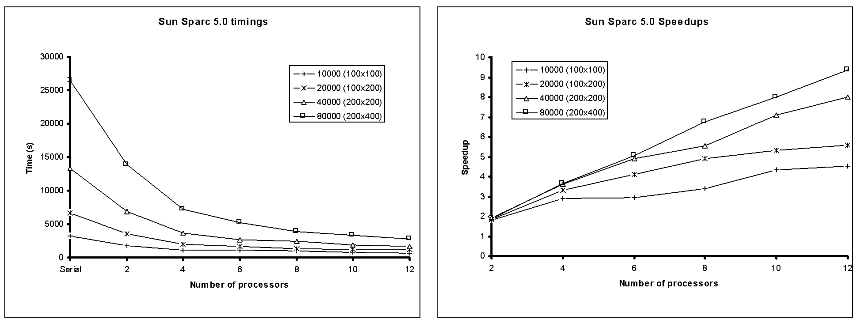

Figure 3 Parallel performance results for Algorithm 1.

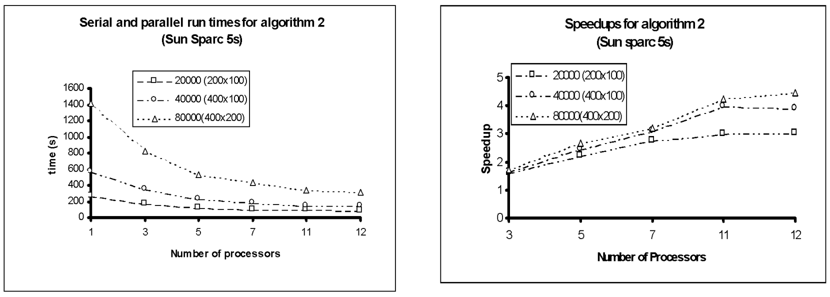

Figure 4 Parallel performance results for Algorithm 2.

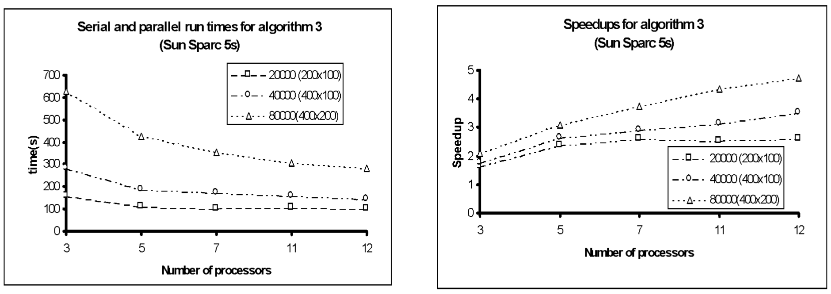

Figure 5 Parallel performance results for Algorithm 3.

Conclusions

The use of alternative numerical algorithms developed by applying DD to the problem domain, in order to calculate the temperature field and to retrieve the unknown source term at the cutter is presented. The three algorithms perform the computation to the same effective accuracy. Algorithms 2 and 3 give better sequential execution times than Algorithm 1. It is shown that good parallelism can be exploited from the DD-based algorithms by using domain-data parallelism and pure data parallelism. An MPI implementation is used to investigate the parallel performance of domain-data parallel versions of the algorithms in a distributed computing environment, that is in a network of Sun Sparc workstations. The parallel performance results show that the two parallelisation strategies can be used effectively in a distributed computing environment.

REFERENCES

[BBSJ85] Beck J. V., Blackwell B., and St.Clair Jr C. R. (1985) *Inverse Heat Conduction: Ill-Posed Problems.* Wiley-Interscience, London.

[BGLS] Balay S., Gropp W., L.C.McInnes, and Smith B.*PETSc 2.0 User Manual.* Argonne National Laboratory. http://www.mcs.anl.gov/petsc/.

[ILPP98] Ierotheou C., Lai C.-H., Palansuriya C., and Pericleous K. (1998) Simulation of 2-d metal cutting by means of a distributed algorithm. *The Computer Journal* 41: 57–63.

[Lai94] Lai C.-H. (1994) Diakoptics, domain decomposition and parallel computing. *The Computer Journal* 37: 840–846.

[PLIP98] Palansuriya C. J., Lai C.-H., Ierotheou C. S., and Pericleous K. A. (1998) A domain decomposition based algorithm for non-linear 2d inverse heat condition problems. In Mandel J., Farhat C., and Cai X.-C. (eds) *Domain Decomposition Methods 10*, volume 218, pages 515–522. American Mathematical Society.

[Zwi89] Zwillinger D. (1989) *Handbook of Differential Equations.* Academic Press Inc., San Diego.