A Distributed Algorithm for 1-D Nonlinear Heat Conduction with an Unknown Point Source

C.-H. Lai

1 Introduction

The mean tool face temperature involved in intermittent cutting operations such as metal cutting and face milling, has a very important influence on the rate of tool wear and tool life. High temperatures may cause the material to fatigue or deform under face milling or other cutting operations [Sha84]. Therefore accurate simulation of temperature distributions of the work piece subject to milling or cutting is vital in order to lengthen the life time of the tool and to guarantee the quality of the cutting. In particular, real-time simulation of such temperature distributions is of industrial interests.

A major barrier in industry is that cutting temperatures are required experimentally which are then used as empirical data in suitably chosen thermal models. The measurement of physically meaningful temperatures is extremely difficult. It is particularly true for the measurement of deformation or shear zone temperatures. On the other hand, thermal models do not provide direct numerical simulation of the cutting process based on the governing differential equation [Bec85]. In order to provide a simulation software for metal cutting, a numerical algorithm is required. It is natural to assume that the application of a cutting tool at the cutting point is equivalent to the application of a source at the same point. Therefore if one can simulate the equivalent source at the cutting point, then one would be able to simulate the temperature distribution. Such approach is often referred to as an inverse problem approach. The approach is used in this paper for the modelling of a simplified cutting process. The aim of this paper is to study a domain decomposition algorithm for the simulation of temperature distributions along a work piece under cutting operations.

The layout of the paper is as follows. First, a description is given of the model for an idealised cutting problem. Second, the partitioning of the physical problem into a number of subproblems is discussed. Third, a distributed numerical algorithm is introduced. Different numerical schemes are employed in different subdomains in

Ninth International Conference on Domain Decomposition Methods
Editor Petter E. Bjørstad, Magne S. Espedal and David E. Keyes

order to solve different subproblems. Numerical tests are provided for three different types of material. An efficiency analysis is also included. Finally, some conclusions are drawn.

2 An Idealised Cutting Model

Pioneering work in remote sensor methods for the retrieval of temperature distributions can be found in [LNK67]. Recently, such methods have been developed into more mature inverse methods [CW88][Ste91][YW86] for various cutting situations. However, the use of inverse methods becomes pragmatic since analytical temperature distributions are difficult to derive. In this paper, sensors and numerical methods are combined to provide solutions to metal cutting problems.

To simplify the cutting problem, a piece of metal of infinite length and of homogeneous material property along the longitudinal direction, is considered. If the cutting tool is applied at a position along the width direction, then it is possible to assume a one-dimensional analogy of the physical problem. Therefore the domain of interest is along the width only, i.e. $x_0 < x < x_1$. Assuming the cutter is applied at $x = x_c$, then the above cutting problem can be described by the one-dimensional nonlinear unsteady parabolic heat conduction equation,

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x}(k(\theta)\frac{\partial \theta}{\partial x}) + Q_c(t)\delta(x - x_c) , \tag{2.1}$$

subject to initial condition $\theta(x, 0) = \Theta(x)$ and boundary conditions $\theta(x_0, t) = \Theta_0$ and $\theta(x_1, t) = \Theta_1$ where $\theta(x, t)$ is the temperature distribution, $k(\theta)$ is the conductivity of the metal, $Q_c(t)$ is the equivalent source being applied at $x = x_c$, and $\delta(x - x_c)$ is the Dirac delta function.

The continuity of the function $\frac{\partial \theta}{\partial t}$ at $x = x_c$ suggests that $\int_{x_c^-}^{x_c^+} \frac{\partial \theta}{\partial t} dx = 0$. Hence the equivalent source strength can be obtained by integrating (2.1) from $x = x_c^-$ to $x = x_c^+$ to give

$$k(\theta)\frac{\partial \theta}{\partial x}|_{x_c^+} - k(\theta)\frac{\partial \theta}{\partial x}|_{x_c^-} + Q_c(t) = 0 . \tag{2.2}$$

The above equation is used to retrieve the source strength. Hence the temperature gradients just to the left ($x = x^-$) and to the right ($x = x^+$) of the cutter must be known. A temperature sensor is attached at $x = x_s$, such that $x_s < x_c < x_1$, and let the temperature measured by means of the temperature sensor be $\theta(x_s, t) = T(t)$. However, it is not necessary to have x_s being less than x_c. The purpose of the temperature sensor is to complete the problem described in (2.1) and to allow the computation of the temperature gradients. The knowledge of the measured temperature at the sensor is then used to back-calculate average or effective measures at the cutting point. Such inverse methods avoid the basic difficulties of the direct method since remote temperatures can be measured more easily and accurately.

For computer simulation purpose, the sensor temperature is modelled by the sinusoidal function $T(t) = \alpha \sin \omega t$. Its maximum value is governed by the amplitude α and its variation with respect to time is governed by the angular frequency ω.

3 Problem Partitioning

In order to solve the inverse problem given in (2.1) with the extra condition given at $x = x_s$, the problem is partitioned into three subproblems defined in the subdomains namely, $S_1 = \{x : x_0 < x < x_s\}$, $S_2 = \{x : x_s < x < x_c\}$, and $S_3 = \{x : x_c < x < x_1\}$. The partition is basically driven by the problem at the physical level [Lai94] and the effect is to remove the unknown source term $Q_c(t)$. In other words, the differential equations in these three subdomains do not involve the Dirac delta function. Since the temperature is given at $x = x_0$ and there is a temperature sensor located at $x = x_s$, therefore Dirichlet boundary conditions are defined at the boundary of S_1. One can then solve the differential equation to obtain the derivative $\frac{\partial\theta}{\partial x}(x_s, t)$. Hence with the knowledge of the temperature $\theta(x_s, t)$ acquired by the temperature sensor at $x = x_s$, an initial value problem can be formulated in S_2. Therefore $\theta(x_c, t)$ can be obtained by solving the initial value problem. Note that k drops out because $k(u_1) = k(u_2)$ at $x = x_s$. Finally another Dirichlet problem can be formulated in S_3. Thus, we have the three subproblems as follow:

SP_1: $\frac{\partial u_1}{\partial t} = \frac{\partial}{\partial x}(k(u_1)\frac{\partial u_1}{\partial x})$ in S_1
subject to $u_1(x, 0) = \Theta(x)$, $u_1(x_0, t) = \Theta_0$, $u_1(x_s, t) = T(t)$.

SP_2: $\frac{\partial u_2}{\partial t} = \frac{\partial}{\partial x}(k(u_2)\frac{\partial u_2}{\partial x})$ in S_2
subject to $u_2(x, 0) = \Theta(x)$, $u_2(x_s, t) = T(t)$, $\frac{\partial u_2(x_s,t)}{\partial x} = \frac{\partial u_1(x_s,t)}{\partial x}$.

SP_3: $\frac{\partial u_3}{\partial t} = \frac{\partial}{\partial x}(k(u_3)\frac{\partial u_3}{\partial x})$ in S_3
subject to $u_3(x, 0) = \Theta(x)$, $u_3(x_c, t) = u_2(x_c, t)$, $u_3(x_1, t) = \Theta_1$.

The above three subproblems are well-defined [Bec85][Zwi89], and a unique solution exists for each of them. The direct sum of these subproblem solutions gives the temperature distribution of the original problem, i.e.

$$\theta(x,t) = \begin{cases} u_1(x,t), & x \in S_1 \\ u_2(x,t), & x \in S_2 \\ u_3(x,t), & x \in S_3 \end{cases} . \tag{3.3}$$

Note that the above algorithm is intrinsically sequential. However, a careful load balancing would make a distributed algorithm with minimal communication possible.

4 The Distributed Numerical Algorithm

One obvious way of distributing subproblems is to employ as many loosely coupled workstations as the number of subproblems. In the present case there should be three workstations. It is possible to treat the algorithm as a pipe line process, in which case the solution of SP_1 at a new time step will be computed first and then SP_2, etc. Therefore there is a time-lag for SP_3 compare with SP_2 and SP_1, i.e. all of the three workstations are occupied with work at the beginning of the third time step and before the last two time steps. Let W_{SPi} denotes the computational work involved in solving SP_i. The situation $W_{SP1} \neq W_{SP2} \neq W_{SP3}$ yields a distributed algorithm with computing time depending on $\max\{W_{SP1}, W_{SP2}, W_{SP3}\}$. Since the sensor is located in the neighbourhood of the cutter, therefore the subdomain S_2 is usually very small. On the other hand, $u_1(x_s, t) = T(t)$ is known, the subproblem SP_1 can be operated completely independent of SP_2 and SP_3. It is therefore possible to reduce the communication time by putting SP_2 and SP_3 into a workstation for sequential process. In such situation, the computation of the source strength does not involve inter-processor communication. To maintain load balancing in the two workstations, we require $W_{SP1} \simeq W_{SP2} + W_{SP3}$. The equality means that the two processes are synchronised. If $W_{SP1} \neq W_{SP2} + W_{SP3}$, then it is important that $\frac{\partial u_1(x_s,t)}{\partial x}$ is sent from the first processor to the second processor and is kept in the local memory of the second processor for the use in subsequent computations. For synchronised processes, such storage is not necessary. The distributed algorithm is given as:

Distributed Algorithm for Metal Cutting Process

Processor 1:
  for i = 1, number_of_steps
    $t := i * \triangle t$;
    Compute the solution of SP_1 at time t; Compute $\frac{\partial u_1(x_s,t)}{\partial x}$;
    Non-blocking Send $\frac{\partial u_1(x_s,t)}{\partial x}$ to Processor 2;
  end-for
Processor 2:
  for i = 1, number_of_steps
    $t := i * \triangle t$; Blocking Receive $\frac{\partial u_1(x_s,t)}{\partial x}$ from Processor 1;
    Compute the solution of SP_2 at time t;
    Compute the solution of SP_3 at time t;
    Compute $\frac{\partial u_2(x_s,t)}{\partial x}$ and $\frac{\partial u_3(x_s,t)}{\partial x}$;Retrieve $Q_c(t)$ using (2.2);
  end-for

The meaning of non-blocking send in the above algorithm is that computation in the sending processor resumes as soon as the message is safely on its way to the receiving processor. The meaning of blocking receive in the above algorithm is that the receiving processor has to wait until the correct message from the sending processor has arrived.

Numerical Schemes

A first order forward difference approximation of the temporal derivative and a second order central difference approximation of the spatial derivatives are used in the subproblems SP_1 and SP_3. An explicit scheme is resulted from the difference approximation. Dropping the subscripts used in denoting the subdomains, the explicit scheme for SP_1 and SP_3 can be written as

$$u_i^{(n+1)} = rb_i^{(n)} u_{i-1}^{(n)} + (1 - r(a_i^{(n)} + b_i^{(n)}))u_i^{(n)} + ra_i^{(n)} u_{i+1}^{(n)} , \tag{4.4}$$

where i denotes the i-th grid point, $r = \frac{\triangle t}{(\triangle x)^2}$, $a_i^{(n)} = \frac{k_{i+1}^{(n)} + k_i^{(n)}}{2}$, $b_i^{(n)} = \frac{k_i^{(n)} + k_{i-1}^{(n)}}{2}$, (n) denotes the time-step, $\triangle t$ is the step size along the temporal axis and $\triangle x$ is the mesh size along the spatial axis x.

The subproblem SP_2 becomes a second order initial value problem along the spatial dimension when the first order backward difference approximation, denoted as m, of

Table 1 Conductivity parameters.

Material	a	b	c	d
A	0.5	-1.1	1.0	-1.0
B	0.8	-0.5	0.01	-0.01
C	1.0	-0.3	0.0001	-0.0001

the temporal derivative at $x = x_s$ is substituted into $\frac{\partial u_2}{\partial t}$ of SP_2. A one-step modified Euler integration scheme is applied to solve the initial value problem and is written as in the following pair of calculations,

$$\begin{pmatrix} u \\ v \end{pmatrix}^* = \begin{pmatrix} u \\ v \end{pmatrix} + \triangle x \underline{f} \ , \ \begin{pmatrix} u \\ v \end{pmatrix}^{\text{new}} = \begin{pmatrix} u \\ v \end{pmatrix} + \frac{\triangle x}{2} \{\underline{f} + \underline{f}^*\} \ , \tag{4.5}$$

where $v = \frac{\partial u}{\partial x}$, $\underline{f} = \underline{f}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} v \\ \frac{m}{k} - \frac{k'}{k}v^2 \end{pmatrix}$ and $\underline{f}^* = \underline{f}\begin{pmatrix} u \\ v \end{pmatrix}^*$. A second order accurate solution may be obtained for each of the three subproblems. Therefore it is expected to have a second order accurate global solution for the inverse problem (2.1). The effect of global truncation errors for SP_2 is minimised because of the small size of the subdomain which usually consists of only a few modified Euler steps. Since the numerical schemes in SP_1 and SP_3 are explicit, therefore the CFL condition, $\triangle t \leq \frac{\triangle x^2}{2K}$ where $K = \max\{k_i^{(n)}\}$, must be satisfied.

A sequential Fortran program has been written to perform the above tasks. PVM (Parallel Virtual Machine) [Gei94] is used to provide distributive directives in order that the tasks can be distributed onto a network of Sun workstations. For the present studies, only two Sun workstations are required.

Numerical Tests

A number of tests was performed by taking the conductivity as $k(u) = a + bu + cu^2 + du^3$, where a, b, c, and d are parameters used to describe the material property of a piece of metal. Table 1 shows three sets of different conductivity parameters used in the subsequent tests. The nonlinearity of the conductivity decreases as $|c|$ and $|d|$ decreases. In particular the last set of conductivity parameters represents almost a constant conductivity. For the conductivity as shown in Table 1, α and ω are chosen to be 0.4 and 2π respectively. The boundary points are located at $x = x_0$ and x_1. The initial value $\Theta(x)$ and the boundary values Θ_0 and Θ_1 are chosen to be zeros. The locations of the sensor and the cutter, i.e. x_s and x_c, are chosen to be 0.4 and 0.5 respectively. Numerical results are provided for two mesh sizes $\triangle x = \frac{1}{20}$ and $\triangle x = \frac{1}{40}$ and the corresponding $\triangle t$'s are chosen to be 0.001 and 0.0002. The resulting numbers of modified Euler steps in SP_2 are two and four respectively. Temperature distributions at $t = 1.1$ seconds are shown in Figure 1. The results show that $\triangle x$ has little effect on the temperature distribution. Source strength variations with respect to t for different mesh sizes are also similar.

Efficiency Analysis

In order to study the parallel computational work, the number of elementary operations per grid point per time-step is needed. The number of elementary operations involved in the numerical schemes are listed in Table 2. In this analysis, it is assumed that the workstations are loosely coupled in a local network such as Ethernet. It is easy to work out the number of operations for the explicit scheme as 43. Let t_s be the CPU time required to perform 43 floating point operations, then the CPU time required to march one time-step in SP_1 and SP_3 are $n_1 t_s$ and $n_3 t_s$ respectively where n_1 and n_3 are the number of grid points in SP_1 and SP_3. By counting the operations involved in the modified Euler's method, the CPU time required to march one time-step forward in SP_2 is $\frac{56}{43} n_2 t_s$. Therefore the following parallel computational time for n time steps is estimated,

$$t_p = n_1 t_s + \frac{56}{43} n_2 t_s + n_3 t_s + (n-1)\max\{n_1 t_s, \frac{56}{43} n_2 t_s + n_3 t_s\} + t_c$$

where t_c is the average communication time between any two workstations. The speed-up can then be estimated as

$$S = (n_1 + \frac{56}{43} n_2 + n_3) t_s n / t_p \tag{4.6}$$

It is natural to ignore t_c during evening or weekend runs, and as $n \to \infty$, the ideal speed-up is obtained as

$$S = \frac{n_1 + \frac{56}{43} n_2 + n_3}{\max\{n_1, \frac{56}{43} n_2 + n_3\}} \tag{4.7}$$

The relation

$$n_1 = \frac{56}{43} n_2 + n_3 \tag{4.8}$$

is used to check the load balancing of the distributed algorithm. For the case $n_1 > \frac{56}{43} n_2 + n_3$, $S = 1 + \frac{56}{43}\frac{n_2}{n_1} + \frac{n_3}{n_1}$, for the case $n_1 = \frac{56}{43} n_2 + n_3$, $S = 2$, and for the case $n_1 < \frac{56}{43} n_2 + n_3$, $S = (\frac{56}{43}\frac{n_2}{n_1} + \frac{n_3}{n_1})^{-1} + 1$. Therefore, the speed-up ratio satisfies $1 < S_p \leq 2$, for any positive integer n_1. If the problem size approaches the limits $\frac{n_2}{n_1} \to 0$ and $\frac{n_3}{n_1} \to 1$, then the ideal speed-up approaches 2. Figures 2 and 3 confirm the above result by plotting CPU times against various problem sizes in both sequential and distributed runs. The results shown in these figures were run on SUN SPARC5 workstations connected locally by Ethernet.

The analysis above shows that the use of multi-step methods for the solutions of the initial value problem in SP_2 is not recommended because the communication time will increase. However, if SP_1 is a small subdomain which requires SP_2 to be solved with SP_1 in a processor in order to achieve load balance, then multi-step methods can be employed which do not increase the communication time.

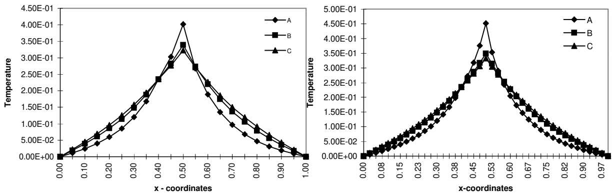
5 Conclusions

The use of the distributed algorithm for the retrieval of heat source at the cutter and the calculation of the temperature distribution is presented. PVM is used to examine

Table 2 Operations count.

Elementary Operations									
$k(u)$	$k'(u)$	r	m	$\frac{k'}{k}$	a_i	b_i	$\underline{f}$	$\begin{pmatrix} u^* \\ v^* \end{pmatrix}$	$\begin{pmatrix} u \\ v \end{pmatrix}$
9	7	2	2	17	20	11	23	27	29

Figure 1 Temperature distributions for $h = \frac{1}{20}$ and $h = \frac{1}{40}$ at $t = 1.1$s.



the efficiency of the algorithm in a distributed environment. Numerical results show that the algorithm is scalable. Fast and parallel numerical schemes may then be used within individual subproblems to reduce the overall computing time.

REFERENCES

[Bec85] Beck J. (1985) *Inverse Heat Conduction*. John Wiley & Son Inc., New York.

[CW88] Chow J. and Wright P. (1988) On-line estimation of tool/chip interface temperature for a turning operation. *ASME Journal of Engineering for Industry* 110: 56–64.

[Gei94] Geist A. (1994) *PVM - a User Guide and Tutorial for Networked Parallel Computing*. MIT Press, London.

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

[LNK67] Lipman M., Nevis B., and Kane G. (1967) A remote sensor method for determining average thermal properties heated by moving heat sources. *ASME Journal of Engineering for Industry* 89: 333–338.

[Sha84] Shaw M. (1984) *Metal Cutting Principles*. Oxford University Press, London.

[Ste91] Stephenson D. (1991) An inverse method for investigating deformation zone temperatures in metal cutting. *ASME Journal of Engineering for Industry* 113:

Figure 2

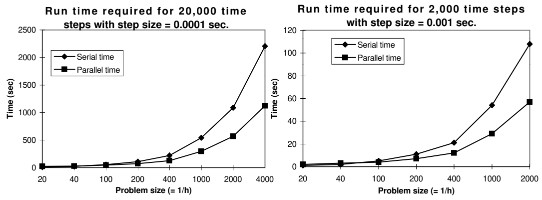
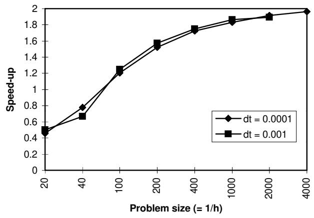


Figure 3 Speedup ratio.



129–136.
[YW86] Yen D. and Wright P. (1986) A remote temperature sensing technique for estimating the cutting interface temperature distribution. *ASME Journal of Engineering for Industry* 108: 252–263.
[Zwi89] Zwillinger D. (1989) *Handbook of Differential Equations.* Academic Press Inc., San Diego.