Development of a new integration algorithm for parallel implementation of the finite element elasto-plastic analysis

Z. Ding[*] S. Kalyanasundaram[*] L. Grosz[†]
S. Roberts[†] M. Cardew-Hall[*]

(Received 7 August 2000)

Abstract

The accurate integration of stress-strain relations is an important factor in element analysis for elasto-plastic problems. The conven-

[*] Department of Engineering, Faculty of Engineering and Information Technology, The Australian National University, Canberra, ACT 0200, Australia

[†] School of Mathematical Sciences The Australian National University, Canberra, ACT 0200, Australia

[0] See http://anziamj.austms.org.au/V42/CTAC99/Ding for this article and ancillary services, © Austral. Mathematical Soc. 2000. Published 27 Nov 2000.

tional method for this problem is the Euler algorithm which divides the whole integration process into a number of smaller substeps of equal size. It is difficult to control the errors in such integration scheme. In this paper, we will present a new algorithm for integrating strain-stress relations. It is based on the third and the fourth order Runge-Kutta method. This substepping scheme controls the errors in the integration process by adjusting the substep size automatically. In order to implement the substepping scheme on parallel systems, a parallel preconditioned conjugate gradient method is developed. The resulting algorithms have been implemented on a parallel environment defined by a cluster of workstation and their performance will be presented.

Contents

1 Introduction

Computer architectures have been, and continue to be, undergoing progressive development. New parallel computers are having significant influence on finite element analysis. The use of large, complex elasto-plastic models raises a number of questions in relation to accuracy and efficiency. Due to the iterative algorithm used in the finite element analysis, an accurate integration algorithm is a key factor when implementing such simulation on the parallel systems.

For elasto-plastic problem, if the stresses at an integration point cause plastic yielding and an isotropic hardening rule is employed, the stresses are found by solving a system of differential equations of the form

$$\frac{d\sigma}{dT} = D_{ep}(\sigma)\Delta\varepsilon, \quad T \in [0,1] \tag{1}$$

where $\Delta\varepsilon$ is the incremental strain and D_{ep} is the elasto-plastic matrix. In Equation (1), $\sigma|_{T=0}$ defines the stress state which already satisfy the yield criterion, and $\sigma|_{T=1}$ defines the stress at an end of load increment or iteration. Equation (1) defines a classical initial value problem, since $\Delta\varepsilon$ and the stress state at $T = 0$ are known.

The conventional method for solving the system of differential equations defined by Equation (1) is the Euler algorithm. As the Euler scheme is accurate for very small time steps, it is usual to subdivide the whole integration process into smaller substeps and compute the stress-strain response over each substep. Traditionally, the number of substeps is determined from an empirical rule and each substep is assumed to be of the same size[1]. Since the predicted state of stress at the end of a loading increment may not lie on the yield surface, a correction step is usually used to restore the stress back to the yield surface. Although this method has been used widely in the finite element codes, it has following disadvantages:

1. If the correction-step is applied after each substep, the computational time will increase drastically. However, if it is done at the end of integration, it does not significantly affect the accuracy [2].
2. Since the number of substeps is usually determined by an empirical rule which is formulated by trial and error, the inappropriate choice of the number of the substeps usually lead to lose of either accuracy or efficiency.

In this paper, we will introduce a substepping scheme which can be used to integrate the elasto-plastic stress-strain relation with an aim to control the error by adjusting the size of each substep automatically. The resulting algorithm has been applied to a parallel analysis of a three-dimensional cantilever beam and its performance will also be presented.

2 Formulation of Runge-Kutta methods

The Runge-Kutta method is commonly used for numerical integration of ordinary differential equations. To calculate successive values of the dependent variable y of the differential equation

$$\frac{dy}{dx} = y' = f(x, y) \tag{2}$$

We write Equation (2) more compactly as

$$y_{i+1} = y_i + \sum_{j=1}^{n} a_j k_j \tag{3}$$

where

$$k_j = hf\left(x_i + p_{j-1}h, y_i + \sum_{l=1}^{j-1} q_{j-1,l}k_l\right) \quad (j = 1,\ 2, \cdots,\ n) \tag{4}$$

in which, by definition,

$$P_0 = 0 \quad \text{and} \quad \sum_{l=1}^{j-1} q_{j-1,l}k_l = 0 \quad j = 1 \tag{5}$$

The a's, p's, and q's must assume values such that Equation (2) accurately yields successive values of y. These values are determined by making Equation (2) equivalent to a certain specified number of terms of a Taylor-series expansion of y about x_i.

Various Runge-Kutta methods are classified as (m, n) methods. A $(5, 6)$ method, for example, would be a fifth-order method requiring six function evaluations per step. Runge-Kutta methods were among the earliest methods employed in the numerical solution of differential equations, and they are still widely used. The principal advantage of the Runge-Kutta methods is their self-starting feature and consequent ease of programming. However, the function of $f(x, y)$ must be evaluated for several slightly different values of x and y in every step of the solution.

3 Substepping scheme for integration process

It will be shown that methods of high order can be formulated for elastoplastic problems, which are much more efficient than the first order algorithms used up to the present. Since the substepping scheme controls the error by decreasing the step size, it definitely involves a large number of substeps. As such, the cumulative effect of the per-step roundoff errors and their magnification in calculating subsequent substeps must be minimized. In this paper, we will employ Gill's fourth-order Runge-Kutta method which is known for its advantage of minimizing the roundoff error [6].

The strategy is based on the observation that when iterating the Equation (1) for σ_{k+1} we can obtain final approximations σ_{K+1} using order m

$$\sigma^{(m)} = \sigma_k + D_{ep}\Delta\varepsilon \tag{6}$$

We can easily construct a reference solution by using order n

$$\sigma^{(n)} = \sigma_k + D_{ep}\Delta\varepsilon \tag{7}$$

This reference solution $\sigma^{(n)}$ can be considered as an 'embedded' solution [7]. Now, as an estimate for the local error E in the step from T_k to $T_{k+1} = T_k + \Delta T$, we take

$$E = \|{\sigma_{k+1}}^m - \sigma^{(n)}\| \tag{8}$$

for some norm $\|\cdot\|$. Usually, one uses reference solution $\sigma^{(n)}$ such that the orders of $\sigma^{(m)}$ and $\sigma^{(n)}$ differ by 1. Here we follow this approach and choose $n = m - 1$.

Based on the above formulations, the error estimate after a time step ΔT_k is obtained by comparing the estimated stress increments which result from the third and Gill's fourth order Runge-Kutta method. They are given by:

$$\sigma_{k+1} = \sigma_k + (\Delta\sigma_1 + 4\Delta\sigma_2 + \Delta\sigma_3)/6 \tag{9}$$

and

$$\hat{\sigma}_{k+1} = \sigma_k + \frac{1}{6}\left[\Delta\sigma_1 + (2-\sqrt{2})\Delta\sigma_2 + (2+\sqrt{2})\Delta\sigma_3 + \Delta\sigma_4\right] \tag{10}$$

in which

$$\begin{aligned}\sigma_1 &= \sigma_k\\ \sigma_2 &= \sigma_k + 0.5\Delta\sigma_1\\ \sigma_3 &= \sigma_k + 0.5(\sqrt{2}-1)\Delta\sigma_1 + 0.5(2-\sqrt{2})\Delta\sigma_2\\ \sigma_4 &= \sigma_k - 0.5\sqrt{2}\Delta\sigma_2 + (1+0.5\sqrt{2})\Delta\sigma_3 \qquad (11)\end{aligned}$$

Subtracting Equation (9) from Equation (10), we obtain an estimate of the local truncation error in σ_{k+1} according to

$$E_{k+1} = \frac{1}{6}\left[-(2+\sqrt{2})\Delta\sigma_2 + (1+\sqrt{2})\Delta\sigma_3 + \Delta\sigma_4)\right] \qquad (12)$$

As an estimate for the local error in the substep from T_k to $T_{k+1} = T_k + \Delta T_k$, we define the relative error for this substep as

$$R_{k+1} = \| E_{k+1} \| / \| \hat{\sigma}_{k+1} \| \qquad (13)$$

Then R_{k+1} is compared with some prescribed tolerance TOL_{sub} and the step is accepted if $R_{k+1} \leq \text{TOL}_{\text{sub}}$, and rejected otherwise. Furthermore, the value of R_{k+1} allows us to make an estimate for the asymptotically optimal stepsize:

$$\Delta T_{k+1} = \Delta T_k \sqrt[4]{\text{TOL}_{\text{sub}}/R_{k+1}} \qquad (14)$$

In case of rejection ΔT_{k+1} is used instead of ΔT_k; in case of acceptance we use ΔT_{k+1} to continue the integration. In order to reduce the substeps rejected, we actually used

$$\Delta T_{k+1} = \Delta T_k \cdot \min\left\{2, \max\left\{0.1, 0.9\sqrt[4]{\text{TOL}_{\text{sub}}/R_{k+1}}\right\}\right\} \qquad (15)$$

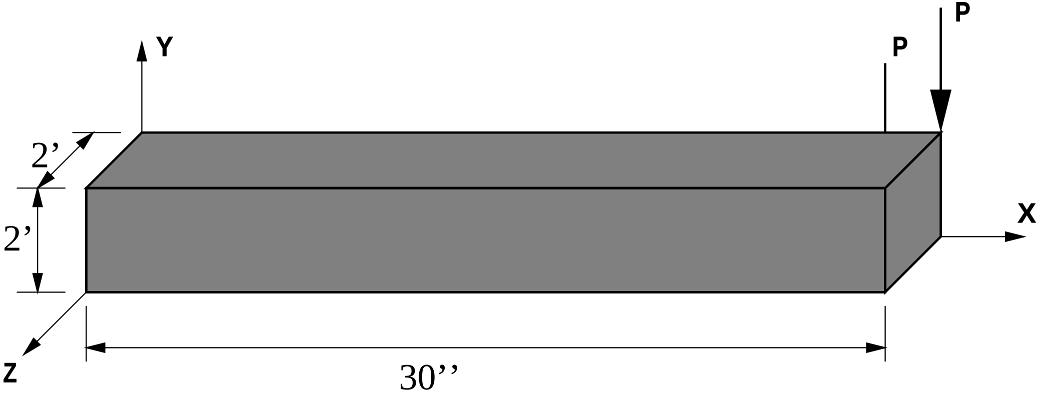

Figure 1: A typical cantilever beam

The constants 2 and 0.1 in this expression serve to prevent an abrupt change in the substep size, and the safety factor 0.9 is added to increase the probability that next substep will be accepted. As suggested by some researcher, for example, in [8], some forms of stress correction must be used when the analysis involves strain (work) hardening. In this paper, a proportional scaling of stresses is used.

4 Performance analysis of substepping schemes

In order to illustrate the performance of the substepping scheme in the finite element analysis, a computer programme was developed and has been applied to a typical three dimensional cantilever beam (Figure 1). The analyses were carried out in 10 equal load increments. To test the influence of different tolerance on the accuracy, an "ideal" run for each analysis is implemented with the same mesh, the same load increments and the same global solution technique, but Sloan's fifth-order Runge-Kutta-England substepping scheme is used to integrate the constitutive law [3]. The TOL_{sub} is set to be very small (10^{-10}) to ensure that the number of the substeps is sufficiently high so that the drift from the yield surface can be ignored. These 'ideal' results are used to compare with the Runge-Kutta (RK) method developed in this research work and Sloan's Modified Euler (ME) scheme [3] with different tolerances. The errors in the elasto-plastic stresses are computed using:

$$\text{Error} = \frac{\| \sum_{i=1}^{N} (\sigma_i - {\sigma_i}^{\text{ideal}}) \|}{\| \sum_{i=1}^{N} ({\sigma_i}^{\text{ideal}}) \|}$$

To test the accuracy and efficiency of the substepping scheme, two analyses for different problem sizes were carried out, one for the problem with 288 degrees of freedom (d.o.f) and 1620 integration points, another for the problem with 432 d.o.f and 3240 integration points. For each analysis, four different models are studied, that is

1. Elastic-perfectly plastic, i.e. no strain hardening model. No stress correction is employed;
2. Elastic-perfectly plastic model with stress correction;
3. Model with the linear strain hardening, but no stress correction;
4. Model with the linear strain hardening, and stress correction is employed.

Note that in all the following figures and tables, '****' denotes the diverged solution due to the large error tolerances. 'C' means that stress correction is applied. NH denotes the elastic-perfect plastic model and WH denotes the model which involves strain hardening.

4.1 Accuracy

The errors in the computed stresses from different algorithms with different tolerance are listed in Table 1 and 2. With reference to the Table 1 and 2, it can be seen that, for the case with no strain hardening, we can get the accuracy without stress correction which is adequate for the engineering computation if $\mathrm{TOL}_{\mathrm{sub}}$ is set to be smaller than 10^{-3}. This accuracy can be improved when the stress correction is employed. However for the analyses with strain hardening, both the Runge-Kutta scheme and the Modified Euler scheme can not give an improved accuracy as $\mathrm{TOL}_{\mathrm{sub}}$ decreases. This is

Table 1: Results of errors for problem with 288 d.o.f with different tolerance

		$\mathrm{TOL}_{\mathrm{sub}}$					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	***	***	1.07	1.6×10^{-2}	4.3×10^{-3}	3.9×10^{-4}
	RK	***	***	***	1.9×10^{-2}	1.6×10^{-3}	7.2×10^{-4}
NH	MEC	0.26	0.24	1.8×10^{-2}	4.9×10^{-4}	4.1×10^{-5}	4.1×10^{-6}
	RKC	***	0.28	7.9×10^{-2}	7.6×10^{-3}	1.5×10^{-4}	6.9×10^{-6}
WH	ME	***	***	0.89	0.88	0.87	0.87
	RK	***	***	***	0.17	0.17	0.17
WH	MEC	0.13	9.3×10^{-2}	5.9×10^{-3}	2.5×10^{-4}	1.5×10^{-5}	1.69×10^{-6}
	RKC	***	***	1.5×10^{-2}	4.5×10^{-3}	4.9×10^{-5}	3.4×10^{-5}

Table 2: Results of errors for problem with 432 d.o.f with different tolerance

		$\mathrm{TOL}_{\mathrm{sub}}$					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	***	***	7.4×10^{-2}	3.6×10^{-2}	2.8×10^{-3}	3.2×10^{-4}
	RK	***	***	****	2.1×10^{-2}	1.1×10^{-3}	7.6×10^{-4}
NH	MEC	***	***	3.8×10^{-2}	6.3×10^{-4}	3.8×10^{-5}	4.0×10^{-6}
	RKC	****	***	0.18	1.1×10^{-2}	1.6×10^{-4}	9.2×10^{-6}
WH	ME	***	***	***	***	***	***
	RK	***	***	***	***	***	***
WH	MEC	0.13	9.3×10^{-2}	5.9×10^{-3}	2.5×10^{-4}	1.5×10^{-5}	1.7×10^{-6}
	RKC	0.19	0.14	2.6×10^{-2}	2.0×10^{-3}	1.8×10^{-4}	7.3×10^{-6}

illustrated in Table (1) that for the problem with 288 d.o.f, the errors are large no matter how the error tolerance is reduced. For the problem with 432 d.o.f which involves more integration points, large magnitude of errors cause each analysis to diverge. It is important to note that, when stress correction is used, the accuracy can be improved drastically. This implies that, due to the approximation nature of the finite element method, yield surface drift may occur with the stresses moving away from the yield surface. This deviation is practically independent of the integration scheme adopted. When the model involves strain (work) hardening where the yield surface is moving with loading increment, the drift is more significant. Since such discrepancies are usually cumulative, it is important to ensure the the stresses are corrected back to the current yield surface at each step of the calculation. Since we apply the stress correction at the end of each substep, it does not affect the accuracy significantly. However, it can make the computed stress fulfil the plasticity criterion at the end of load increment (iteration) and avoid error accumulation and therefore instabilities in the following load steps.

One of the conclusions of Sloan's work on the substepping scheme [3] is that stress correction is not required to improve the accuracy. However, the results of the present study clearly indicate the importance of the role of stress correction in the substepping scheme. The accuracy of the results can be improved by at least an order of magnitude by using stress correction for elasto-plastic problem involving strain hardening.

Table 3: Total substeps needed for analysis of problem with 288 d.o.f with different tolerance

		TOL$_{\text{sub}}$					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	****	****	16006	33790	85734	246458
	RK	****	****	****	13154	34310	131360
NH	MEC	15266	14356	17384	33822	85792	247236
	RKC	****	14506	11156	13394	34310	131460
WH	ME	****	****	7422	15918	42166	123714
	RK	****	****	****	6360	17762	65992
WH	MEC	6258	6390	9498	20872	53640	157198
	RKC	****	****	6090	7782	21670	83616

4.2 Efficiency

In Tables 3 and 4, we list the total number of substeps for overall solution of different analysis. We can see that, the Runge-Kutta scheme generally requires less substeps than the Modified Euler scheme for the fixed value of TOL$_{\text{sub}}$. For TOL$_{\text{sub}}$ which are equal to 10^{-3} and 10^{-4}, the Runge-Kutta scheme uses less than half of number of substeps consumed by the Modified Euler scheme. This confirms that the high-order Runge-Kutta scheme does not need more substeps to obtain high level of accuracy. Due to this, the Runge-Kutta scheme usually uses less CPU time than the Modified Euler scheme for the cases which TOL$_{\text{sub}}$ are greater than 10^{-5}. Application of stress correction does not increase the computational time significantly, but

Table 4: Total substeps needed in for analysis of problem with 432 d.o.f with different tolerance

		$\mathrm{TOL}_{\mathrm{sub}}$					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	****	****	40388	82688	205652	585420
	RK	****	****	****	32748	83384	309676
NH	MEC	****	****	41804	82900	206028	586384
	RKC	****	****	31296	34224	83600	310268
WH	ME	****	****	****	****	****	****
	RK	****	****	****	****	****	****
WH	MEC	14988	18512	23100	49284	128380	372718
	RKC	13968	14424	14724	19536	50824	194368

it does improve the accuracy considerably. Based on above analyses, we will use the Gill's fourth-order Runge-Kutta method in the following parallel implementation.

5 Parallel preconditioned conjugate gradient method

To implement the substepping scheme on the parallel systems, an efficient algorithm for equation solution is necessary. Recently, Law [4] developed a parallel conjugate gradient algorithm by using transformation relationships between the displacement (as well as force) vectors local to each processor and the corresponding global vectors. In the present implementation, Law's element-by-element algorithm has been modified to a substructure-by-substructure algorithm and a diagonal preconditioner is used to accelerate the convergence rate. In order to increase the convergence rate of CG method further, a parallel minimal residual (MR) smoothing method is also employed [5]. Throughout the process, the formation of global system of equations is not performed. The storage space required for each processor includes an substructure matrix and vectors. A parallel substructure preconditioned conjugate gradient (PPCG) algorithm is now described.

Algorithm: Parallel Substructure Preconditioned Conjugate Gradient combined with MR Smoothing

```
    Initialization:

1:      (a) $\{x^{(s)}\}_0 = \{z^{(s)}\}_0 = 0$
        (b) $\{r^{(s)}\}_0 = \{f^{(s)}\}$
        (c) Compute $[C]^{(s)}$

2:      Exchange $[C^{(s)}]$ with neighbour $j$

3:      (a) $[C_g{}^{(s)}] = \sum_{j\in adj(s)}[C^{(j)}] + [C^{(s)}]$
        (b) $\{t^{(s)}\}_0 = \{s^{(s)}\}_0 = [C_g{}^{(s)}]^{(-1)}\{r^{(s)}\}_0$
```

```
4:    Exchange $\{t^{(s)}\}_0$ with neighbour $j$

5:    (a) $\{t_g{}^{(s)}\}_0 = \sum_{j\in adj(s)} \{t^{(j)}\}_0 + \{t^{(s)}\}_0$
      (b) $\{s_g{}^{(s)}\}_0 = \{p^{(s)}\}_0 = \{t_g{}^{(s)}\}_0$
      (c) $\rho_0{}^{(s)} = \{t_g{}^{(s)}\}_0^T\{t^{(s)}\}_0$

6:    (Merge Sum) $\gamma_0 = \rho_0 = \sum \rho^{(s)}{}_0, s = 1, ..., p$

Iterate $k = 1, 2, \cdots$ If $\gamma_k/\gamma_0 <$ tolerance terminate

1:    (a) $\{h^{(s)}\}_k = [K^{(s)}]\{p^{(s)}\}_{k-1}$
      (b) $\beta_k{}^{(s)} = \{p^{(s)}\}^T{}_{k-1}\{h^{(s)}\}_k$
      (c) $\sigma_k{}^{(s)} = \beta_k{}^{(s)}/\gamma_{k-1}$

2:    (Merge Sum) $1/\alpha_k = \sum \sigma_k{}^{(s)}, s = 1, ..., p$

3:    (a) $\{x^{(s)}\}_k = \{x^{(s)}\}_{k-1} + \alpha_k\{p^{(s)}\}_{k-1}$
      (b) $\{r^{(s)}\}_k = \{r^{(s)}\}_{k-1} - \alpha_k\{h^{(s)}\}_k$
      (c) $\{t^{(s)}\}_k = [C_g{}^{(s)}]^{(-1)}\{r^{(s)}\}_k$

4:    Exchange $\{t^{(s)}\}_k$ with neighbour $j$

5:    (a) $\{t_g{}^{(s)}\}_k = \sum_{j\in adj(s)}\{t^{(j)}\} + \{t^{(s)}\}$
```

```
     (b) $\rho_k{}^{(s)} = \{t^{(s)}\}^T{}_k\{t^{(s)}\}_k$

6:   (Merge Sum) $\rho_k = \sum \rho_k{}^{(s)}, s = 1, ..., p$

7:   (a) $\{p^{(s)}\}_k = \{t_g{}^{(s)}\}_k + (\rho_k/\rho_{k-1})\{p^{(s)}\}_{k-1}$
     (b) $\mu_k{}^{(s)} = \{s_g{}^{(s)}\}^T{}_k(\{t^{(s)}\}_k - \{s^{(s)}\}_k)$
     (c)$\theta_k{}^{(s)} = (\{t_g^{(s)}\}_k - \{s_g^{(s)}\}_{k-1})^T \times (\{t^{(s)}\}_k - \{s^{(s)}\}_{k-1})$

8:   (a) (Merge Sum) $\mu_k = \sum \mu_k{}^{(s)}, \quad \theta_k = \sum \theta_k{}^{(s)}$
     (b) $\delta_k = -\mu_k/\theta_k$

9:   (a) $\{s^{(s)}\}_k = \{s^{(s)}\}_{k-1} + \delta_k(\{t^{(s)}\}_k - \{s^{(s)}\}_{k-1})$
     (b) $\{s_g^{(s)}\}_k = \{s_g^{(s)}\}_{k-1} + \delta_k(\{t_g^{(s)}\}_k - \{s_g^{(s)}\}_{k-1})$
     (c) $\{z^{(s)}\}_k = \{z^{(s)}\}_{k-1} + \delta_k(\{x^{(s)}\}_k - \{z^{(s)}\}_{k-1})$
     (d) $\gamma_k{}^{(s)} = \{s_g{}^{(s)}\}^t{}_k\{s^{(s)}\}_k$

10:  (Merge Sum) $\gamma_k = \sum \gamma_k{}^{(s)}$

11:  Go To Step 1
```

g denotes the assembled vector.

6 Numerical results

The parallel computer used in this research is a Linux-Alpha workstation cluster. The Linux-Alpha Cluster consists of twelve 533MHz Alpha LX164s each with 256MB of memory and 5.3GBs of IDE disk connected by a HP fast Ethernet switch. The developed algorithms have been applied to the elasto-plastic finite element analysis of a typical three dimensional deep cantilever beam (Figure 2). As usual, the two important metrics, speedup and efficiency, are tested respectively. They are defined in the following form:

$$\text{Speedup} = \frac{\text{Time for solution on 1 process}}{\text{Time for solution on } p \text{ processes}} \tag{16}$$

and the efficiency by:

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{Number of processes}} \tag{17}$$

A horizontal strip-wise partitioning scheme is considered for this application. This numerical experiment is designed to determine performance of the parallel algorithms when a combination of good load balancing and reduced interprocessor communication is employed.

The characteristics and performances of horizontal partitioning scheme for 3-D deep beam are summarized in Table 5. From the results we can see

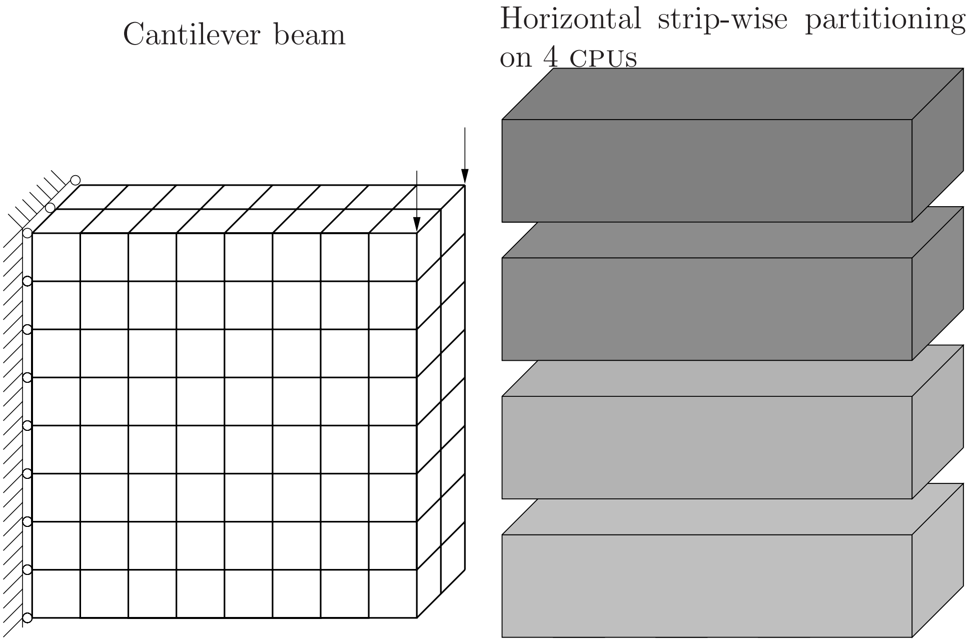

Figure 2: A three dimensional deep cantilever beam and the partitioning scheme

Table 5: Performance of horizontal partitioning on fixed number of processors, showing Efficiency E and Speedup S (continued next page).

Number of processor	Number of elements in each substructure	Number of interface nodes in each substructure	CPU time on 1 processor (sec)	CPU time on n processors (sec)	S	E (%)
2	480	25	87.70	44.52	1.97	98.5
	960	45	360.62	184.31	1.96	97.8
	1920	81	746.47	375.50	1.99	99.4
	3840	153	1790.85	898.20	1.99	99.7
	7680	289	5892.09	3015.00	1.95	97.7
	15360	561	28936.26	14817.03	1.95	97.7
3	320	25	87.70	29.53	2.97	99.0
	640	45	360.62	126.33	2.85	95.2
	1280	81	746.47	276.80	2.70	90.0
	2560	153	1790.85	624.97	2.87	95.5
	5120	289	5892.09	1967.03	2.99	99.9
	10240	561	28936.26	10059.02	2.88	95.9
4	240	25	87.70	22.23	3.95	98.6
	480	45	360.62	92.64	3.89	97.3
	960	81	746.47	191.25	3.90	97.6
	1920	153	1790.85	460.38	3.89	97.3
	3840	289	5892.09	1511.71	3.91	97.4
	7680	561	28936.26	7342.32	3.94	98.5

Table 5: continued.

Number of processor	Number of elements in each substructure	Number of interface nodes in each substructure	CPU time on 1 processor (sec)	CPU time on n processors (sec)	S	E (%)
5	192	25	87.70	17.96	4.88	97.6
	384	45	360.62	82.34	4.38	87.6
	768	81	746.47	156.99	4.75	95.1
	1356	153	1790.85	400.35	4.47	89.5
	3072	289	5892.09	1189.82	4.95	99.0
	6144	561	28936.26	6012.85	4.81	96.3
6	160	25	87.70	15.20	5.78	96.6
	320	45	360.62	66.09	5.45	90.9
	640	81	746.47	128.92	5.79	96.5
	1280	153	1790.85	321.35	5.57	92.9
	2560	289	5892.09	1023.00	5.76	96.0
	5120	561	28936.26	4823.71	5.98	99.9

that the performance of horizontal partitioning scheme in this application is good. In fact, even for small problem size (i.e. 960 elements), almost perfect results are obtained. This perfect results can also be attributed to, a) substepping scheme can integrate the strain-stress relations accurately, so it uses less iterations, and thus makes the solution more efficient; b) the diagonal storage scheme we used save the time wasted on zero fill-ins; c) the preconditioner and MR smoothing improve the convergence rate of CG algorithm. It clarifies that, to get an optimal performance on the parallel systems like the workstation cluster on which the communication is very expensive, adoption of the partitioning scheme must be based on the fact that such partitioning scheme involves least amount of interprocessor communication.

7 Conclusion

In this paper, a substepping scheme which controls the error in the integration process by permitting the size of each substep to vary in accordance with the behaviour of the constitutive law and a parallel substructure preconditioned conjugate gradient method have been presented. This solution algorithm does not require the formation of the global system equations. Each processor in the parallel system is assigned a substructure and stores only the information relevant to the substructure that the processor represents. The combination of these two algorithms have been applied to a typical three dimensional elasto-plastic stress analysis. The result indicated that the combination of these algorithms shows a good speedup when increasing the number of processors. In summary, the combination of these two algorithms provides a powerful practical strategy for parallel finite element analysis of elasto-plastic problems.

References

[1] D. R. J. Owen and E. Hinton, *Finite Elements in Plasticity: Theory and Practice*, Pineridge Press, West Cross Lane, Swansea, U.K., 1980. C564

[2] J. W. Wissmann and C. Hauck, "Efficient Elastic-Plastic Finite Element Analysis with Higher Order Stress-Point Algorithms", *Comput. Structures*, Vol. 17, No. 1, pp.89–95 (1983) C564

[3] S. W. Sloan,"Substepping Schemes for the Numerical Integration of Elastoplastic Stress-strain Relations", *Int. J. Numer. Methods Eng.*, Vol.24, pp.893–911 (1987) C570, C570, C573

[4] K. H. Law, "A Parallel Finite Element Solution Method", *Computers & Structures*, Vol. 23, No. 6, pp.845–858, 1986 C575

[5] Z. Ding, "Parallel Substepping Scheme for Elasto-Plastic Finite Element Analysis", Master Thesis, The Australian National University, 1999 C576

[6] M. L. James, G. M. Smith and J. C. Wolford, *Applied Numerical Methods for Digital Computation*, Harper & Row, Publisher, New York, 1985 C566

[7] V. K. Arya, "Efficient and Accurate Explicit Integration Algorithm with Application to Viscoplastic Models", *Int. J. Numer. Methods Eng.*, Vol. 39, pp.261–279 (1996) C567

[8] A. Gens and D. M. Potts, "Critical State Models in Computational Geomechanics", *Eng. Comput.*, Vol. 5, pp.178–197 (1988) C569

[9] B. P. Sommeijer, *Parallelism in the numerical integration of initial value problems*, Stichting Mathematisch Centrum, Amsterdam, 1993