Iterative solvers in forming process simulations

A.H. van den Boogaard[1,2], A.D. Rietman[1] and J. Huétink[1,2]
[1] *Fac. Mechanical Engineering, University of Twente, Enschede, The Netherlands*
[2] *Netherlands Institute for Metals Research, Delft, The Netherlands*

ABSTRACT: The use of iterative solvers in implicit forming process simulations is studied. The time and memory requirements are compared with direct solvers and assessed in relation with the rest of the Newton–Raphson iteration process. It is shown that conjugate gradient–like solvers with a proper preconditioning can significantly improve the overall time performance of a forming process simulation as compared to the use of direct solvers. For the presented examples the time spent in the solver is not dominating the total solution time anymore.

1 INTRODUCTION

In finite element simulations of forming processes the size of the models has reached a point where the direct solution of resulting linear sets of equations is the bottle-neck. The linear sets of equations arise from the implicit solution method by a linearization e.g the Newton–Raphson method. One way to avoid the excessive demand on computer resources (CPU time and memory space) is the use of an explicit solution method (Mercer et al. 1995). A disadvantage of this method is the conditional stability, necessitating extremely small time steps or artificial adaptations to the model. Another way to alleviate the problem is to use iterative solvers instead of direct solvers, to solve the linear sets of equations. In the field of elasto-plastic simulations (Kacou and Parsons 1993), (Mahnken 1995) and (Jefferson and Thomas 1997) have studied the behavior of iterative (multigrid) solvers in a Newton–Raphson iteration process.

In this presentation different iterative solvers are assessed in the simulation of a tensile test, deepdrawing and a 3D thermomechanical compression test. This includes features that are known to deteriorate the convergence of iterative solvers: incompressibility, mixed physics (thermo-mechanical), shell elements and contact conditions. The behavior of iterative solvers at mesh refinement is verified with the tensile test simulation. With the deepdrawing example, the influence of the local convergence criterion on the global Newton–Raphson process is examined. Finally with the compression test simulation the influence of contact stiffness and incompressibility is examined.

2 THE NEWTON–RAPHSON PROCESS

The goal of this study is the speed up of numerical simulation of forming processes. Therefore the time and memory usage of different solvers has to be related to the overall time and memory requirements of a Newton–Raphson process. It would be a waste of time to optimize a part of the code that only takes a fraction of the total resources.

In relation to direct and iterative solvers in a Newton–Raphson process it is relevant to note that for a direct solver, e.g. a Cholesky decomposition, the time complexity is of order $O(n_b^2 \cdot n)$ where n_b is the bandwidth of the matrix and n is the number of degrees of freedom. The following substitution phase has a complexity of only $O(n_b \cdot n)$. In the following comparison, the solution time for the direct solvers is optimized by a bandwidth optimization algorithm according to Sloan (Sloan 1989).

A direct solver can take advantage of the modified Newton–Raphson method, where only in the first iteration a new matrix is set up (and decomposed) and in later iterations only the relatively fast substitution is performed. Iterative solvers must be used to their full extends in every iteration, even if the matrix does not change.

One full Newton–Raphson iteration can broadly be split in 3 parts. The calculation of the tangential stiffness matrix, the calculation of the incremental displacement vector and the calculation

of the internal force vector. Furthermore some initialization and closure of each increment takes time, independent of the number of NR-iterations. In figure 1 the relative CPU-times for tension test simulations (see 5.1) are presented for increasing numbers of degrees of freedom. It can be seen

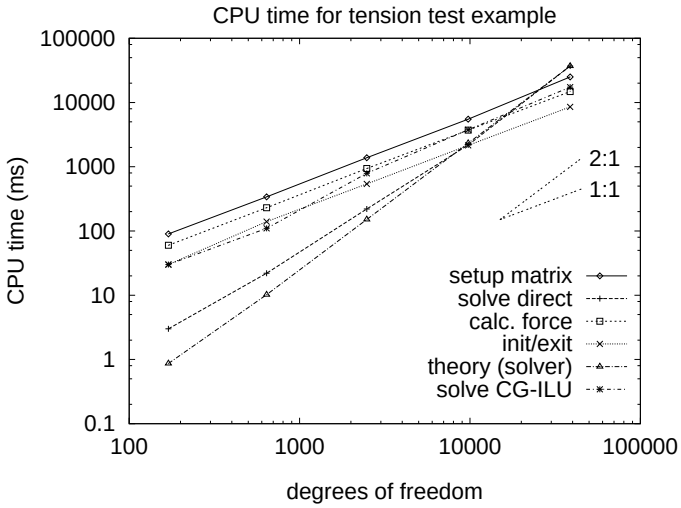

Figure 1: CPU usage for tension test simulation

that the setup of the matrix, calculation of internal force vector and the initialization and closure of an increment are linear in time (mind the logarithmic axis). As could be expected from theory, the time complexity for a direct solver tends to be quadratic in the number of dof's for a 2D analysis. Up to about 10000 dof's the time in the solver is negligible compared to the setup of the matrix and internal force vector. Above approximately 40000 dof's (in this example) the solver takes more than half of the solution time. In figure 1 also a preconditioned Conjugate Gradient solver is presented. Although initially this solver takes much more time than a direct solver, the time complexity is only slightly more than linear in the number of dof's. Within the presented range the CG solver takes less time than the setup of the matrix. In this example systems with more than 20000 dof's the CG solver can be advantageously used over the direct solver. From the directions in the logarithmic graph it can be seen that the advantage is increasing rapidly if the number of dof's increase.

In a 3D simulations, the bandwidth increases faster at mesh refinement than in 2D simulations, therefore the time required by a direct solver can easily be more than 90% of an iteration.

3 ITERATIVE SOLVERS

The solution of a linear set of equations can be approximated in an iterative way. Several examples can be found in (Golub and Van Loan 1989), (Hackbush 1994) and (Barret et al.). These solvers become interesting if they produce an acceptable incremental displacement vector in a shorter time and using less memory than a direct solver. The memory requirements are very clear. Usually only the non-zero components of a matrix have to be stored, while with direct solvers, due to fill-in the mean bandwidth times the number of rows has to be stored. With mesh refinement in 2 directions, the bandwidth increases linearly and the number of degrees of freedom increase quadratically. In 3 dimensions the bandwidth increases quadratically and the number of degrees of freedom increase cubically. In both cases the typical connectivity remains the same and therefore the number of non-zero's per row remains the same. Scaled to the number of dof's, direct solvers need a total amount of memory, proportional to $n^{3/2}$ 2D and $n^{5/3}$ in 3D. Iterative solvers need memory, proportional to n.

The time requirements for iterative solvers are not so clear. For every local iteration, the time is linearly dependent on the number of degrees of freedom but the number of iterations is not fixed. The number of iterations depends on the local convergence criterion (see section 4) and the condition of the matrix. For conjugate-gradient-like solvers the ratio between the largest and smallest eigenvalues determines the convergence rate. Typically the convergence deteriorates due to high stiffness ratios (stiff contact elements, multiphysics or shell elements), due to decreasing material stiffness (plasticity) and due to singularities (constant volume constraints). Preconditioning of the matrix can significantly improve the convergence characteristics.

In the examples presented in 5 several iterative solvers each with several preconditioners are compared. The iterative solvers are: Symmetric Succesive Over Relaxation (SSOR), Conjugate Gradient (CG), Generalized Minimum Residual (GMRES), Quasi-Minimal Residual (QMR) and BiConjugate Gradient Stabilized (Bi-CGSTAB). Apart from the first all iterative solvers where used without preconditioner and with Jacobi, SSOR and Incomplete Lower Upper (ILU) preconditioning. Pure SSOR iterations are a kind of generalization of Gauß–Seidel iterations and are presented for comparison. The others are all CG-like solvers, from which pure CG is derived for symmetric matrices and the others for non-symmetric matrices. Experience show however that CG is also very efficient for some non-symmetric systems and especially Bi-CGSTAB is sometimes faster and more stable then CG, even for symmetric matrices. In the experiments also the CG on Normal Equations (CGN), Conjugate Gradient Squared (CGS) and BiConjugate Gradient (BiCG) methods were applied, but

these performed consistently worse then the earlier mentioned methods and are not referenced in the sequel.

4 CONVERGENCE CRITERION

The nonlinear finite element equations are solved by an incremental iterative procedure. The loads are prescribed incrementally and in every increment the increment in displacements $\Delta\mathbf{u}$ is calculated.

The iterative solution of the incremental displacements can be performed by e.g. a Newton–Raphson procedure. During these iterations the changes in the incremental displacements are denoted by $\Delta(\Delta\mathbf{u})$.

Inside the Newton–Raphson procedure a linear system of equations must be solved. This can be done by an iterative solver e.g. Gauß–Seidel, Conjugate Gradient or GMRES. In the iterative linear solver once again a variation of the change in incremental displacement appears leading to $\Delta(\Delta(\Delta\mathbf{u}))$.

This many *deltas* makes the description of the procedure very complex and therefore some new symbols are introduced.

We can consider increment j of an incremental procedure. The internal force vector (based on $\int \mathbf{B}^{\mathrm{T}}\boldsymbol{\sigma}\,\mathrm{d}V$) must equilibrate the external force vector (based on the applied loading).

$$\mathbf{f}_{int}(\mathbf{u}^{j-1} + \Delta\mathbf{u}^{j}) = \mathbf{f}_{ext}^{j} \tag{1}$$

Here $\mathbf{f}_{int}$ is the internal force vector that is a function of the displacements calculated in the previous increment $(j-1)$ and the displacement increment $\Delta\mathbf{u}^j$.

In the sequel of this text we do not consider the incremental procedure anymore and the index j is skipped. The displacements at the start of the increment are fixed values and can be disregarded. To avoid the many *deltas* we write the incremental displacements as $\mathbf{d} = \Delta\mathbf{u}$ leading to:

$$\mathbf{f}_{int}(\mathbf{d}) = \mathbf{f}_{ext} \tag{2}$$

or equivalently

$$\mathbf{f}_{ext} - \mathbf{f}_{int}(\mathbf{d}) = 0 \tag{3}$$

Since $\mathbf{f}_{int}$ is a nonlinear function of $\mathbf{d}$ this equation is solved iteratively. At iteration i (3) is not fulfilled completely, but a residual $\mathbf{r}^i$ remains:

$$\mathbf{f}_{ext} - \mathbf{f}_{int}(\mathbf{d}^i) = \mathbf{r}^i \tag{4}$$

We now write the internal force vector as the last known value at iteration i, a linearized increment $\mathbf{K}^i\Delta\mathbf{d}^{i+1}$ and an error $\mathbf{e}^{i+1}$.

$$\mathbf{f}_{int}(\mathbf{d}^{i+1}) = \mathbf{f}_{int}(\mathbf{d}^i) + \mathbf{K}^i\Delta\mathbf{d}^{i+1} + \mathbf{e}^{i+1} \tag{5}$$

In the Newton–Raphson process the increment $\Delta\mathbf{d}^{i+1}$ is calculated by:

$$\Delta\mathbf{d}^i = (\mathbf{K}^i)^{-1}\mathbf{r}^i \tag{6}$$

Substituting (6) into (5) and the result into (4) it is clear that $\mathbf{r}^{i+1}$ equals $\mathbf{e}^{i+1}$.

The iterations are stopped if the Euclidean norm of $\mathbf{r}$ is less than ε times the Euclidean norm of $\mathbf{f}_{int}$. The actual ratio in iteration i is designated as ε^i.

$$\frac{\|\mathbf{r}^i\|}{\|\mathbf{f}_{int}\|} = \varepsilon^i \leq \varepsilon \tag{7}$$

We now consider an iterative solution process for the linear system of equations, described by (6). To simplify the derivations $\Delta\mathbf{d}^i$ is written as $\mathbf{x}$ and $\mathbf{r}^i$ as $\mathbf{y}$.

$$\mathbf{K}\mathbf{x} = \mathbf{y} \tag{8}$$

With an iterative solver $\mathbf{x}$ is not exactly calculated, but it is approximated by $\mathbf{x}^*$ so that

$$\mathbf{K}\mathbf{x}^* - \mathbf{y} = \boldsymbol{\epsilon} \tag{9}$$

The process will terminate if the Euclidean norm of the error vector $\boldsymbol{\epsilon}$ is less than δ times the Euclidean norm of $\mathbf{y}$.

$$\|\boldsymbol{\epsilon}\| \leq \delta\|\mathbf{y}\| \tag{10}$$

Relating the convergence to $\mathbf{y}$ instead of directly to $\mathbf{f}_{int}$ facilitates the use of 'of the shell' solvers as is done in this research.

It is felt that the value δ should be related to the required accuracy ε of the global process. If we use the approximated vector $\Delta\mathbf{d}^*$ instead of $\Delta\mathbf{d}$ in (5) we get:

$$\mathbf{f}_{int}(\mathbf{d}^{i+1}) = \mathbf{f}_{int}(\mathbf{d}^i) + \mathbf{K}^i\Delta\mathbf{d}^{i+1*} + \mathbf{e}^{i+1} + \boldsymbol{\epsilon} \tag{11}$$

The new residue is now formed by $\mathbf{e}^{i+1}$ and $\boldsymbol{\epsilon}$.

$$\mathbf{r}^{i+1} = \mathbf{e}^{i+1} + \boldsymbol{\epsilon} \tag{12}$$

The norm of $\mathbf{r}^{i+1}$ is bounded by:

$$\|\mathbf{e}^{i+1} + \boldsymbol{\epsilon}\| \leq \|\mathbf{e}^{i+1}\| + \|\boldsymbol{\epsilon}\| \tag{13}$$

We require that $\|\boldsymbol{\epsilon}\| \leq \delta\|\mathbf{r}^i\|$. The global convergence norm is now:

$$\frac{\|\mathbf{r}^{i+1}\|}{\|\mathbf{f}^{int}\|} = \varepsilon^{i+1} \leq \frac{\|\mathbf{e}^{i+1}\|}{\|\mathbf{f}^{int}\|} + \delta \cdot \varepsilon^i \tag{14}$$

In an exact solution of (8) the new unbalance ratio would be the first part of the right hand side and if this would be less than ε the Newton–Raphson process is considered to be converged. If we achieve

that $\delta \cdot \varepsilon^i$ is much smaller than ε the global process is only marginally influenced. We can conclude that if δ is set to the value of:

$$\delta = \eta \cdot \frac{\varepsilon}{\varepsilon^i} \tag{15}$$

with a small η, the global iteration process is only marginally influenced by the linear iteration process. Several authors have advocated a value of $\eta = 0.1$ (Jefferson and Thomas 1997), (Blaheta and Axelsson 1997) and (Kacou and Parsons 1993), based however on different definitions of the convergence norms. The criterion could be less strict in the first iteration of every increment if convergence is not expected in the first iteration (Mahnken 1995). In section 5.2 the influence of η on the global convergence will be demonstrated.

5 EXAMPLES

The solvers that were used in the following examples were taken from the public domain (Skalický 1995), they are also described in (Barret et al.). The cited direct solvers did not use pivoting and the GMRES solver was restarted after every 20 iterations. All presented examples were run on a HP 9000-735 workstation with a PA 7100 processor and 128 Mb RAM (spec fp92 = 170).

5.1 *Tensile test*

First a simple tension test problem is used to assess the rate of convergence in case of large plastic deformations and the influence of mesh refinement. All types of non-stationary iterative solvers performed best with an SSOR preconditioner. The different solvers are compared at three stages. In the first step the system represents the fully elastic behavior. In the second stage (at increment 10) the tension bar is fully plastic. In the third stage (at increment 100) the strain is localized in a necking zone and the tension bar is partly relaxing elastically. In figure 2 the results of the last stage (localization) are presented for different mesh densities. The coarsest mesh has 5 elements in radial direction and 15 in axial direction. The finer meshes are derived by halving the element size in both directions. This leads to a finest mesh of 80×240 elements in the finest mesh. A Nadai hardening curve with exponent 0.13 was taken in an elastoplastic material model.

It can be seen that for the finer meshes (where iterative solvers are faster than direct solvers) the Bi-CGSTAB solver with SSOR preconditioning is the fastest of the examined solvers in all stages of the analysis.

For most solvers, the results for the finest mesh where almost independent of the analysis stage.

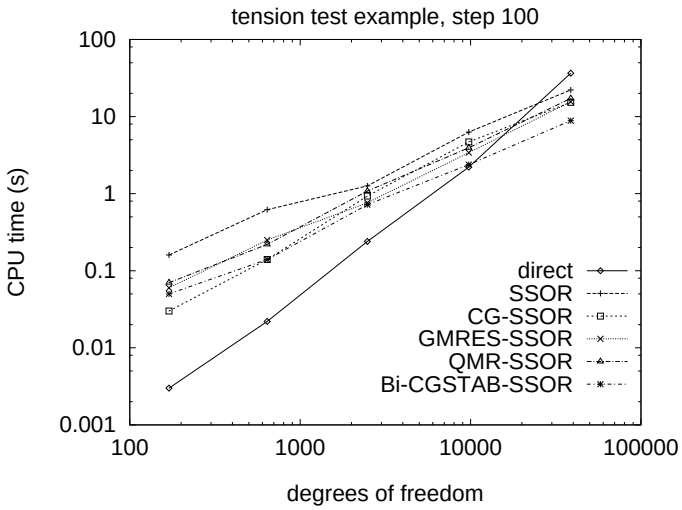

Figure 2: CPU usage for solve step at increment 100

Only the pure SSOR method needed 60% more time in the second stage (uniformly plastic) than in the first and last stage (elastic and localized plasticity). The Bi-CGSTAB solver performed about 15% worse in stage 2 than in stage 1 and 3, but still this method was the best in all stages. The other solvers differed less.

5.2 *Deepdrawing*

In the second example the deepdrawing of an S-shaped rail is simulated. The example originates from the NumiSheet '96 conference (Lee et al. 1996). The presented model contains 12000 triangular shell elements and a total of 36968 degrees of freedom. In the stiffness matrix only the symmetric part of the contact elements was taken. In 92 increments a flat piece of sheet metal was simulated to deform into the end geometry as shown in figure 3. With a direct solver, every iteration took about 400 s of which 200 were used for the direct solver. The same simulation was carried out with a CG solver with ILU preconditioning for different values of η (see equation 15).

In figure 4 the convergence of the global residual norm is presented of step number 5. The first iteration is always performed with a local criterion $\delta = 0.01$ and should be independent of η. In the figure the straight horizontal lines represent the convergence characteristics with the direct solver. For the second and third iteration δ depends on the achieved convergence in the previous iteration and on η. The global criterion was fixed at 0.02. It can be seen that for $\eta \to 1$ the global criterion tends to this norm. For this particular model and this particular step, global convergence is always achieved in the third iteration, even with $\eta = 0.9$ but in general it can be seen from figure 4 that for $\eta > 0.05$ the global convergence can deteriorate.

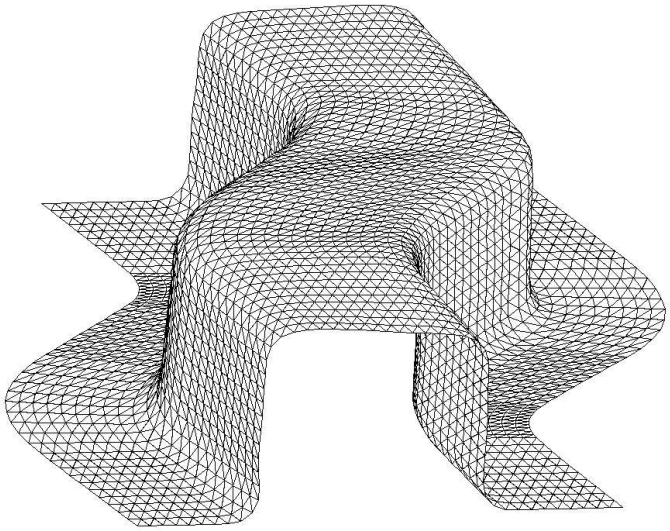

Figure 3: S-Rail final geometry

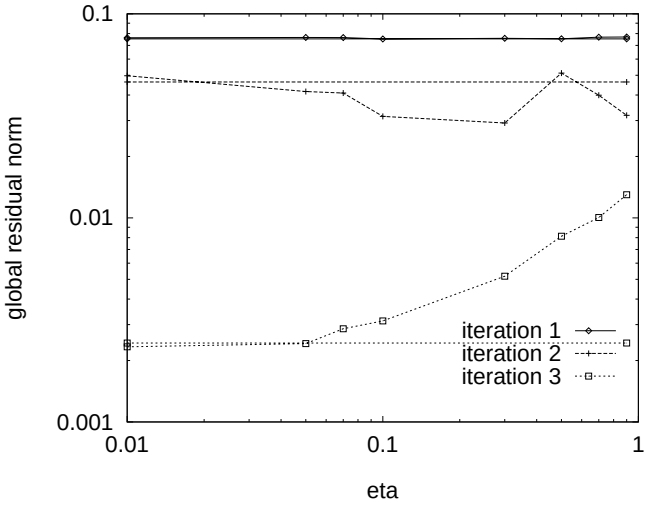

Figure 4: Global convergence at different η

An important difference between the direct solver and the iterative solver is the memory requirement. With the direct solution of the system of equations the maximum size of the analysis was 186 Mb. With the iterative solver this reduced to 120 Mb.

The time necessary to perform the first 5 steps with different values of η and with the direct solver is presented in table 1. For small values of η the CPU time increases because more local iterations must be performed in the iterative solver. For larger values of η at some point more global Newton–Raphson iterations must be performed. The optimum for this particular example is about $\eta = 0.3$. Since not much more time is needed for $\eta = 0.1$ this last value seems to be a robust and efficient choice.

With the direct solver the simulation of all 92 steps took 42 hours. Using $\eta = 0.1$ the simulation time reduced to 26 hours. In total the same number of global iterations were needed. In some steps one iteration less and in some other steps one iteration more than with the direct solver was needed to achieve a global convergence of 0.02 in the residual norm.

Table 1: Influence of η on convergence and time

η	NR	local	CPU time (s)
direct	11	NA	4518
0.01	11	472	2717
0.05	11	285	2495
0.1	11	237	2466
0.3	11	182	2419
0.5	11	177	2448
0.7	12	172	2603
0.9	12	165	2578

5.3 *Compression test*

The third example is the full 3D thermomechanical simulation of a plane strain compression test (see figure 5). Due to the mixed physics and contact conditions the matrix is non-symmetric. The presented model contains 6810 independent degrees of freedom in 1540 linear hexahedron elements. The total height reduction of 76 percent is obtained in 380 increments. It will be shown that iterative solvers can reduce the CPU times in solving considerably. The sensitivity of the iterative solvers for the contact penalty will also be a point of discussion. The simulations are performed with a Bi-

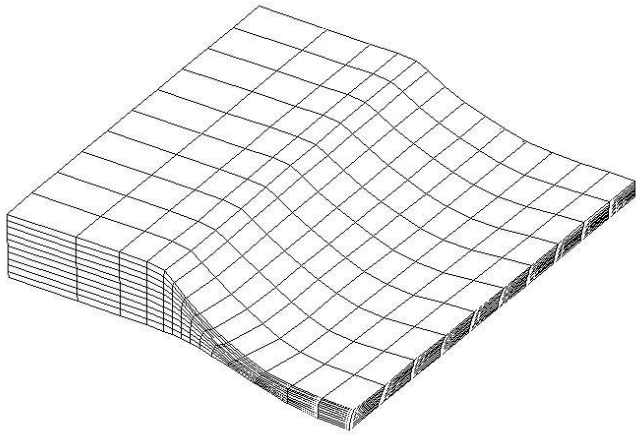

Figure 5: Plane strain compression final geometry

CGSTAB solver with SSOR preconditioner. After starting up a number of 10 increments has been used to determine the appropriate criterion η. For a standard penalty factor $\eta = 0.01$ proved best whereas for a penalty three times as high $\eta = 0.001$

had to be chosen considering the CPU time as well as the solution error. Global Newton–Raphson convergence was reached at $\varepsilon = 0.005$ for both calculations.

The matrices of the first iterations of increments 2, 20 and 200 were analyzed with different solvers. In increment 2 the billet was still entirely elastic. Solver times of the fastest iterative solvers are compared with the direct solver in table 2. For this

Table 2: Influence of penalty and step number on CPU time for 1 linear solve

step	method	normal penalty	high penalty
all	direct	158	158
2	CG SSOR	5.84	4.67
2	GMRES SSOR	6.87	6.80
2	BICGSTAB SSOR	6.91	5.80
20	CG SSOR	7.74	6.30
20	GMRES SSOR	10.32	7.12
20	BICGSTAB SSOR	7.15	6.89
200	CG SSOR	9.09	7.45
200	GMRES SSOR	10.28	9.06
200	BICGSTAB SSOR	10.87	8.87

relatively small 3D problem, the (nonsymmetric) iterative solvers where 15 to 30 times as fast as the direct solver. As in the tensile test example the SSOR preconditioner again performs better than the ILU preconditioner.

The penalty factor for the contact elements does not seem to be of much influence on solving of the system, the global amount of N–R iterations however increases considerably with a higher penalty. This is reflected by the total running times of 7h.48m. and 12h.9m. for the normal and high penalty respectively.

For both penalty factors the Bi-CGSTAB and CG solver with SSOR preconditioner perform best. The CG solver however was mathematically derived for symmetric systems only so the Bi-CGSTAB is recommended for the considered forming problem. For further reduction of CPU times one should optimize the assembly of the system and the force calculation while only 25% is needed for the solver.

6 CONCLUSIONS

It is shown that significant time and memory savings can be reached by using iterative solvers with appropriate preconditioners. Even in the large deepdrawing and 3D compression examples, the linear solution stage is not dominating the total solution process. This means that for these particular examples a further significant reduction of the analysis time can not be achieved from the linear solver alone. Optimization should focus also on the setup of the matrix and solution of the internal force vector.

REFERENCES

Barret, R., M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM. Also available as `http://www.netlib.org/templates/templates.ps`.

Blaheta, R. and O. Axelsson (1997). Convergence of inexact Newton-like iterations in incremental finite element analysis of elasto-plastic problems. *Comput. Methods Appl. Mech. Engrg. 141*, 281–295.

Golub, G. H. and C. F. Van Loan (1989). *Matrix Computations* (2nd ed.). The Johns Hopkins University Press.

Hackbush, W. (1994). *Iterative Solution of Large Sparse Systems of Equations*. Number 95 in Applied Mathematical Sciences. Springer-Verlag.

Jefferson, A. D. and H. R. Thomas (1997). Convergence criteria for iterative solvers applied to nonlinear plasticity problems. In *Computational Plasticity, Fundamentals and Applications*, pp. 441–446.

Kacou, S. and I. D. Parsons (1993). A parallel multigrid method for history-dependent elastoplasticity computations. *Comput. Methods Appl. Mech. Engrg. 108*, 1–21.

Lee, J. K., G. L. Kinzel, and R. H. Wagoner (Eds.) (1996). *Proceedings of the 3rd International Conference: Numisheet '96*. The Ohio State University.

Mahnken, R. (1995). A Newton-multigrid algorithm for elasto-plastic/viscoplastic problems. *Comp. Mechs. 15*, 408–425.

Mercer, C. D., J. D. Nagtegaal, and N. Rebelo (1995). Effective application of different solvers to forming simulations. In *Simulation of Materials Processing: Theory, Methods and Applications*, pp. 469–474.

Skalický, T. (1995). *LASPack Reference Manual* (1.12.2 ed.). Dresden University of Technology. Available as `http://www.tu-dresden.de/mwism/skalicky/laspack/laspack.html`.

Sloan, S. W. (1989). A fortran program for profile and wavefront reduction. *Int. J. Numer. Meth. Eng. 28*, 2651–2679.