VII International Conference on Computational Plasticity
COMPLAS 2003
E. Oñate and D.R.J. Owen (Eds)
©CIMNE, Barcelona, 2003

EFFICIENT IMPLICIT FEM SIMULATIONS OF SHEET METAL FORMING

A. H. van den Boogaard, T. Meinders and J. Huétink

University of Twente
Faculty of Engineering Technology
P.O. box 217, 7500 AE Enschede, The Netherlands
e-mail: a.h.vandenboogaard@ctw.utwente.nl, web page: http://www.tm.ctw.utwente.nl

Key words: sheet metal forming, finite elements, iterative solvers, implicit time integration

Abstract. *For the simulation of industrial sheet forming processes, the time discretisation is one of the important factors that determine the accuracy and efficiency of the algorithm. For relatively small models, the implicit time integration method is preferred, because of its inherent equilibrium check. For large models, the computation time becomes prohibitively large and, in practice, often explicit methods are used. In this contribution a strategy is presented that enables the application of implicit finite element simulations for large scale sheet forming analysis.*

Iterative linear equation solvers are commonly considered unsuitable for shell element models. The condition number of the stiffness matrix is usually very poor and the extreme reduction of CPU time that is obtained in 3D bulk simulations is not reached in sheet forming simulations. Adding mass in an implicit time integration method has a beneficial effect on the condition number. If mass scaling is used—like in explicit methods—iterative linear equation solvers can lead to very efficient implicit time integration methods, without restriction to a critical time step and with control of the equilibrium error in every increment. Time savings of a factor of 10 and more can easily be reached, compared to the use of conventional direct solvers.

1 INTRODUCTION

The two main solution procedures for the simulation of sheet forming processes are the dynamic explicit and the static implicit algorithms. The dynamic explicit method is frequently used in simulations of sheet forming processes, since it reduces the computation time drastically compared to implicit methods, using a diagonal mass matrix to solve the equations of motion [1–3]. A disadvantage of this method is the conditional stability, necessitating extremely small time steps or artificial adaptations to the model.

The static implicit method is unconditionally stable, but a linear set of equations must be solved repeatedly. For large models, this leads to extremely long computation times if direct solvers are used. The use of iterative solvers in quasi-static elasto-plastic simulations has been reported by a number of researchers [4–10]. These papers concerned mainly the convergence behavior of the iterative solver and the parallel implementation. For shell element models, however, convergence is slow. Current research in this field is mainly focussed on the improvement of preconditioners for this type of problems [11–14].

In this paper some aspects of explicit solvers are used to improve the performance of implicit solvers. In this way the computation time of implicit codes can be decreased by factors. In Sections 2 and 3 the basic finite element formulations are reviewed and the continuum problem is discretised. The discretisation leads to huge linear sets of equations, resulting in large computation times for implicit methods. In Section 4 the implicit and explicit time integration algorithms are compared. In Section 5 it is shown that iterative solvers can reduce the calculation time considerably. Section 6 demonstrates the influence of model parameters and dynamics contributions on the efficiency of iterative solvers.

2 PROBLEM DEFINITION

Although most sheet forming processes can be considered quasi-statically, we start by considering conservation of linear momentum, since this is the basis for implicit as well as explicit time integration methods. This conservation law is commonly written as

$$\rho\dot{\mathbf{v}} = \nabla \cdot \boldsymbol{\sigma} + \rho\mathbf{b} \tag{1}$$

where ρ is the mass density, $\mathbf{v}$ the velocity, $\boldsymbol{\sigma}$ the Cauchy stress and $\mathbf{b}$ the body force per unit mass. A superposed dot denotes a material time derivative, hence $\dot{\mathbf{v}}$ is the acceleration.

The problem to be solved can be phrased as: find a displacement field $\mathbf{u}(\mathbf{x})$ such that the balance equations are fulfilled, subject to the initial conditions

$$\mathbf{u}(\mathbf{x}, t_0) = \mathbf{u}_0(\mathbf{x}), \quad \mathbf{v}(\mathbf{x}, t_0) = \mathbf{v}_0(\mathbf{x}) \quad \forall\, \mathbf{x} \in \Omega \tag{2}$$

and the boundary conditions

$$\mathbf{u} = \bar{\mathbf{u}} \quad \text{on } \Gamma_\mathrm{u} \qquad \text{and} \qquad \boldsymbol{\sigma} \cdot \mathbf{n} = \bar{\mathbf{t}} \quad \text{on } \Gamma_\mathrm{t} \tag{3}$$

where Ω is the deformed domain of the analysed body, Γ_u is that part of the boundary on which the displacements $\bar{\mathbf{u}}$ are prescribed and Γ_t the part where the traction $\bar{\mathbf{t}}$ is prescribed

Using the weighted residual method, integration by parts and the divergence theorem, the weak form of the momentum equation becomes

$$\int_{\Omega} \mathbf{w} \cdot \rho \dot{\mathbf{v}} \, \mathrm{d}\Omega + \int_{\Omega} \nabla \mathbf{w} : \boldsymbol{\sigma} \, \mathrm{d}\Omega = \int_{\Omega} \mathbf{w} \cdot \rho \mathbf{b} \, \mathrm{d}\Omega + \int_{\Gamma_t} \mathbf{w} \cdot \bar{\mathbf{t}} \, \mathrm{d}\Gamma \quad \forall \, \mathbf{w} \in \mathcal{V} \tag{4}$$

where $\mathbf{w}$ is the weighting function and $\mathcal{V}$ contains all C_0 vector functions with a value of 0 on Γ_u.

A finite element discretisation is used to discretise the space dimension, usually by a Galerkin method. The nodal displacements are the degrees of freedom, assembled in a vector $\mathbf{d}$. It finally leads to the semi-discrete equations in matrix format:

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{f}_{\text{int}} - \mathbf{f}_{\text{ext}} = \mathbf{0} \tag{5}$$

with the *mass matrix* $\mathbf{M}$, the internal load vector $\mathbf{f}_{\text{int}}$ and the external load vector $\mathbf{f}_{\text{ext}}$. The internal load vector, and in the case of *e.g.* a pressure load also the external load vector, depend on the displacements. Hence, (5) is nonlinear in the displacements. A time integration algorithm must be selected that can cope with this nonlinearity.

2.1 Time integration methods

An incremental-iterative strategy is followed as solution method for the nonlinear semi-discrete equations and a Newton–Raphson method is used for the iterative part. For the time integration procedure at the level of the system equations a Newmark algorithm is chosen. The increment in the displacement vector is updated iteratively by

$$\Delta\mathbf{d}^{k+1} = \Delta\mathbf{d}^{k} + \delta\mathbf{d}^{k} \tag{6}$$

where the superscript k denotes the iteration number. Writing (5) for the end of an increment in a residual form gives

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{f}_{\text{int}} - \mathbf{f}_{\text{ext}} = \mathbf{r}(\Delta\mathbf{d}) \tag{7}$$

in which the residual $\mathbf{r}$ must iteratively be brought to zero to conform to equation (5). This leads to the Newton–Raphson iteration relation for $\delta\mathbf{d}^k$:

$$\left(\frac{\partial \mathbf{r}}{\partial \Delta\mathbf{d}}\right)^{k} \delta\mathbf{d}^{k} = -\mathbf{r}^{k} \tag{8}$$

The matrix with derivatives is called the Jacobian matrix and can be written as

$$\frac{\partial \mathbf{r}}{\partial \Delta\mathbf{d}} = \mathbf{M}\frac{\partial \ddot{\mathbf{d}}}{\partial \Delta\mathbf{d}} + \left(\frac{\partial \mathbf{f}_{\text{int}}}{\partial \Delta\mathbf{d}} - \frac{\partial \mathbf{f}_{\text{ext}}}{\partial \Delta\mathbf{d}}\right) \tag{9}$$

The derivatives $\partial\ddot{\mathbf{d}}/\partial\Delta\mathbf{d}$ depend on the selected time integration algorithm. The term between parentheses is known as the stiffness matrix $\mathbf{K}$.

The Newmark integration method is a one-step integration method, using only the current state and the state to be calculated. The basic assumptions in the Newmark method are

$$\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta t \dot{\mathbf{d}}_n + \Delta t^2 \left[(\tfrac{1}{2} - \beta) \ddot{\mathbf{d}}_n + \beta \ddot{\mathbf{d}}_{n+1} \right] \tag{10a}$$

$$\dot{\mathbf{d}}_{n+1} = \dot{\mathbf{d}}_n + \Delta t \left[(1 - \gamma) \ddot{\mathbf{d}}_n + \gamma \ddot{\mathbf{d}}_{n+1} \right] \tag{10b}$$

All values at time n are fixed and the derivatives $\partial \ddot{\mathbf{d}} / \partial \Delta \mathbf{d}$ can now be determined. Equation (9) yields an equivalent stiffness matrix

$$\mathbf{K}^* = \frac{\partial \mathbf{r}}{\partial \Delta \mathbf{d}} = \frac{1}{\beta \Delta t^2} \mathbf{M} + \mathbf{K} \tag{11}$$

where the parameters β and γ can be chosen freely. In the linear case, the Newmark integration method is unconditionally stable for $2\beta \geq \gamma \geq \frac{1}{2}$ and it is second order accurate for $\gamma = \frac{1}{2}$.

The most widely used scheme is the $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$ scheme. For $\gamma > \frac{1}{2}$ the algorithm is dissipative and numerical damping is introduced. An Euler backward integration scheme is achieved by choosing $\beta = \gamma = 1$. This scheme is consistent with the usual integration of constitutive equations at the material point level.

When we choose $\beta = 0$ and $\gamma = \frac{1}{2}$ we obtain the explicit central difference scheme. A more direct way to describe this scheme for constant time increments Δt is

$$\mathbf{M} \ddot{\mathbf{d}}_n = \mathbf{f}_{\text{ext},n} - \mathbf{f}_{\text{int},n} \tag{12}$$

$$\mathbf{d}_{n+1} = 2\mathbf{d}_n - \mathbf{d}_{n-1} + \Delta t^2 \ddot{\mathbf{d}}_n \tag{13}$$

Clearly, $\mathbf{d}_{n+1}$ can be calculated explicitly, and no iterations are needed. The central difference scheme is conditionally stable and the time increment Δt must be less than the critical time increment $\Delta t_{\text{crit}} = 2/\omega_{\max}$, where $\omega_{\max}$ is the maximum eigenfrequency of the linearised system [15]. The big advantage of the central difference scheme is that the mass matrix can be diagonalised and in that case the solution of $\ddot{\mathbf{d}}$ from (12) is trivial.

3 ELEMENTS

The weak form of conservation of momentum (4) forms the basis of the finite element discretisation. Element mass matrices and internal and external load vectors can be derived from the element contributions to the integrals in (4), leading to

$$\mathbf{M}_\text{e} = \int_{\Omega_\text{e}} \rho \mathbf{N}^\text{T} \mathbf{N} \, \mathrm{d}\Omega \tag{14a}$$

$$\mathbf{f}_{\text{int,e}} = \int_{\Omega_\text{e}} \mathbf{B}^\text{T} \boldsymbol{\sigma} \, \mathrm{d}\Omega \tag{14b}$$

$$\mathbf{f}_{\text{ext,e}} = \int_{\Omega_\text{e}} \mathbf{N}^\text{T} \rho \mathbf{b} \, \mathrm{d}\Omega + \int_{\Gamma_\text{te}} \mathbf{N}^\text{T} \bar{\mathbf{t}} \, \mathrm{d}\Gamma \tag{14c}$$

where **N** and **B** interpolate nodal displacements to local displacements and strains respectively.

In this paper discrete Kirchhoff shell elements are used [16]. They have 3 translational and 3 rotational d.o.f. in each node. The out-of-plane displacement is only defined along the element edges, on which a cubic polynomial is assumed. The parameters are determined by the nodal displacements and rotations and constraint conditions at discrete points. The problem arises that the displacements are not explicitly defined over the complete element. This means that a consistent interpolation matrix **N** cannot be defined.

3.1 Stiffness matrix

The rotational d.o.f. along the normal of the element does not contribute to any deformation in the element. Hence, this so-called drilling d.o.f. gives no contribution to the mass or stiffness matrices. Without further measures, the equivalent stiffness matrix $\mathbf{K}^*$ will be singular and thus will break down any solution procedure. One way to avoid this is to add a small stiffness to the diagonal element that represents this d.o.f. If a direct solver is used, this approach leads to good results. However, the very small diagonal terms lead to ill-conditioned matrices that are detrimental for iterative solvers. The approach described by Zienkiewicz and Taylor [17] may therefore be beneficial in combination with iterative solvers. Here, a coupling between all drilling d.o.f. ϕ_{zi} is achieved by a stiffness relation between drilling moments M_{zi} and rotations ϕ_{zi}:

$$\begin{Bmatrix} M_{z1} \\ M_{z2} \\ M_{z3} \end{Bmatrix} = \alpha \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{Bmatrix} \phi_{z1} \\ \phi_{z2} \\ \phi_{z3} \end{Bmatrix} \tag{15}$$

where α is a factor that scales with the stiffness of the element. This relation contributes to the stiffness matrix and the effect on iterative solvers is demonstrated in Section 6.1.

3.2 Mass matrix

For efficient use in an explicit central difference algorithm the mass matrix must be diagonalised. This can be done in several ways *e.g.* row-sum techniques, proportional scaling or nodal integration [18, 19]. For higher order elements—and that includes shell elements—row-sum and nodal integration can lead to negative values on the diagonal. With proportional scaling the contributions from translational and rotational d.o.f. are mixed up and a physical justification fails.

For Kirchhoff elements a lumped mass matrix was developed based on equivalent acceleration moments or ‘mass moments of inertia’ [20]. The total virtual work due to rotational acceleration for an element along one of its edges is approximated by

$$\delta W = \delta\omega I_{\mathrm{m}}\ddot{\omega} \approx \delta\omega\rho h I\ddot{\omega} \approx \delta\omega\rho h \tfrac{1}{12} b h^3 \ddot{\omega} \approx \delta\omega\rho h \tfrac{1}{3} A^2 \ddot{\omega} = \delta\omega \tfrac{1}{3} m A \ddot{\omega} \tag{16}$$

with m the element mass and A the element area. The resulting diagonal mass matrix becomes

$$\mathbf{M} = \tfrac{1}{3} m \,\mathrm{diag}\left[1 \quad 1 \quad 1 \quad \tfrac{1}{3}A \quad \tfrac{1}{3}A \quad \tfrac{1}{3}A \quad ...(\times 3)...\right] \tag{17}$$

The rotational part of this matrix is not correct, but at least it has correct dimensions and scales properly on mesh refinement. In Section 6.1 the influence of the drilling stiffness and mass matrix on the performance of iterative solvers is investigated.

4 COMPARISON BETWEEN IMPLICIT AND EXPLICIT METHODS

4.1 Implicit methods

The static implicit method is unconditionally stable. Equilibrium is satisfied at the end of each time step. The main drawback of this method, however, is that a linear set of equations must be solved repeatedly. For 2D or planar 3D finite element models and uniform refinement, the computation time increases quadratically with the size of the model when using a direct solver. Another drawback is that sometimes convergence is hard to reach.

In large scale implicit finite element simulations the computation time for a direct solution of linear sets of equations is the bottle-neck. A way to alleviate this problem is to use iterative solvers instead of direct solvers. For shell element models, however, the condition of the matrix is poor and iterative solvers tend to converge very slowly.

Most sheet forming processes can be considered quasi-statically. In this case it is customary to ignore the mass term in (5) altogether and iterate on equilibrium ($\mathbf{f}_{\text{int}} = \mathbf{f}_{\text{ext}}$). The iterations in (8) are performed with the stiffness matrix $\mathbf{K}$ only. The implicit methods of practical relevance are unconditionally stable. This means that the time increment can be based on the required accuracy or convergence behaviour. The determination of iterative increments $\delta\mathbf{d}^k$ requires the solution of a linear set of equations.

In [4, 21] it was demonstrated that, in a 2D analysis, the time needed by a direct solver was negligible for models with less than 10000 d.o.f. and that it dominated the CPU time for models with more than 20000 d.o.f. A conjugate gradient (CG) solver with incomplete Cholesky preconditioning was more efficient in the latter models.

Two serious disadvantages of implicit methods are the (lack of) stability of the Newton–Raphson iteration process and the rapid increase in computation time when direct solvers are used. Iterative solvers can decrease the computation time for large systems, but this depends heavily on the condition of the matrix. Typically for shell structures the condition number of the matrix is very high, and as a consequence iterative solvers converge very slowly or not at all.

4.2 Explicit methods

Compared to the implicit methods, the explicit methods are very easy and straightforward. For reasons of accuracy, only the central difference scheme is used in practice.

In an explicit method no stiffness matrix needs to be generated. The memory requirement for the (diagonal) mass matrix is marginal and since no iterations are required on the material level, no separate start- and end-of-increment data need to be stored and it will never suffer from convergence problems within a time increment. However, equilibrium is not fulfilled after each time step, which can result in erroneous stress distributions and unrealistic product shapes [22].

The most significant drawback of the central difference method is its conditional stability.

With a typical element length of 5 mm, the critical time step is about 1 µs. To reduce the computation time for processes that are not completed within milliseconds, mass scaling is often applied. The mass density of the material is then artificially increased, thus increasing the critical time step. To reach acceptable computation times for processes like deep-drawing, mass scaling with a factor of 100 to 10000 is not uncommon.

It is interesting to note the similarity between the explicit method and one of the simplest iterative linear equation solvers *viz.* the Jacobi iteration. Decomposing the matrix $\mathbf{A}$ in a diagonal part $\mathbf{D}$ and an off-diagonal part $\mathbf{F}$, the Jacobi iteration method to solve $\mathbf{Ax} = \mathbf{b}$ can be presented as [23]

$$\mathbf{Dx}^{k+1} = -\mathbf{Fx}^{k} + \mathbf{b} \tag{18}$$

If we use an implicit method with an initial continuation prediction, the initial estimate for the displacement is $\mathbf{d}^0_{n+1} = 2\mathbf{d}_n - \mathbf{d}_{n-1}$. In the next Newton–Raphson iteration, the displacement becomes $\mathbf{d}^1_{n+1} = 2\mathbf{d}_n - \mathbf{d}_{n-1} - \mathbf{K}^{*-1}\mathbf{r}$. If this set of equations is solved by a Jacobi iteration method, setting $\mathbf{A} = \mathbf{K}^*$ and $\mathbf{b} = -\mathbf{r}$, the first iteration of the Jacobi solver (starting with $\mathbf{x}^0 = \mathbf{0}$) yields $\mathbf{d}^1_{n+1} = 2\mathbf{d}_n - \mathbf{d}_{n-1} + \mathbf{D}^{-1}\mathbf{b}$. If the diagonal of the iteration matrix is dominated by $\mathbf{M}/\Delta t^2$ (*e.g.* by choosing $\beta = 1$ in the Newmark scheme and using a lumped mass matrix) then, by comparison with (12) and (13) it shows that one increment in the explicit method performs like one Jacobi iteration.

4.3 An efficient strategy for implicit methods

Regarding the previous discussion on implicit and explicit methods the analysis strategy based on implicit methods can be optimised. It is clear that the use of direct solvers is not a feasible option for large systems. Iterative solvers do not perform very well for shell models, but at the other hand it was noted that the explicit method can be regarded as one of the simplest iterative solvers. Clearly the dominance of the mass terms in an explicit method is essential. This implies that the strategy should be to retain inertial effects in the implicit finite element code in combination with the use of iterative solvers. When the mass matrix is diagonalised and added to the stiffness matrix, the diagonal terms of the system matrix will increase and the condition of the matrix improves, which makes an effective use of iterative solvers feasible.

5 LINEAR EQUATION SOLVERS

For large models, the computation time needed for the solution of a linear set of equations dominates the total computation time of implicit methods. The chosen analysis strategy advocates the use of iterative solvers for this part. In this section some properties of direct and iterative solvers are discussed.

5.1 Direct solvers

Direct solvers are generally based on some form of Gaussian elimination process. Bandwidth optimisation algorithms are used to reduce the bandwidth and thus the required computation time. More recent direct sparse matrix solvers even optimise on a reduced fill-in during the elimination process. The computational complexity of direct solvers is proportional to $n_{\mathrm{eq}}n_{\mathrm{bw}}^2$, where n_{eq} is the number of equations and n_{bw} the (mean) bandwidth. In sheet forming simulations (a curved plane in 3D) a uniform decrease in element size results in a quadratic increase in the number of equations and a linear increase in the bandwidth, so finally a fourth order increase in the computation time (or quadratic in the number of d.o.f.).

5.2 Iterative solvers

The solution of a linear set of equations can be approximated in an iterative way. Several examples can be found in [23–25]. These solvers become interesting if they produce an acceptable iterative displacement vector $\delta\mathbf{d}^k$ in a shorter time and using less memory than a direct solver.

The time requirements for iterative solvers depend on many parameters. For every local iteration, the time depends linearly on the number of d.o.f. but the number of iterations is not fixed. The number of iterations depends on the local convergence criterion and the condition of the matrix. Typically the convergence deteriorates due to high stiffness ratios (stiff contact elements, multi-physics 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.

Several iterative solvers have been compared in [4]. The iterative solvers used here are: Conjugate Gradient (CG) and BiConjugate Gradient Stabilised (Bi-CGSTAB). All solvers are used with Jacobi, SSOR and Incomplete Cholesky (IC) preconditioning.

5.3 Convergence criterion

In the nonlinear solution process the linear set of equations (8) must be solved repeatedly to obtain the iterative displacement increments $\delta\mathbf{d}^k$. The Newton–Raphson iterations are stopped if the Euclidean norm of $\mathbf{r}$ is less than ε times the Euclidean norm of $\mathbf{f}_{\mathrm{int}}$. The actual ratio in iteration k is designated as ε^k:

$$\frac{\|\mathbf{r}^k\|}{\|\mathbf{f}_{\mathrm{int}}\|} = \varepsilon^k \leq \varepsilon \tag{19}$$

We now consider an iterative solution process for the linear system of equations, described by (8). With an iterative solver $\delta\mathbf{d}^k$ is not calculated exactly, but it is approximated by $\delta\hat{\mathbf{d}}^k$ so that an error $\boldsymbol{\epsilon}$ remains according to

$$\mathbf{K}^{*k}\delta\hat{\mathbf{d}}^k = -\mathbf{r}^k + \boldsymbol{\epsilon} \tag{20}$$

The iterative process will be terminated if the Euclidean norm of the error vector $\boldsymbol{\epsilon}$ is less than δ_{it} times the Euclidean norm of $\mathbf{r}^k$:

$$\|\boldsymbol{\epsilon}\| \leq \delta_{\mathrm{it}}\|\mathbf{r}^k\| \tag{21}$$

Relating the convergence to $\mathbf{r}^k$ instead of directly to $\mathbf{f}_{\mathrm{int}}$ facilitates the use of ‘of the shelf’ solvers. It is argued in [4] that the value of δ_{it} should be chosen such that the additional error $\|\boldsymbol{\epsilon}\|$ does not add much to the error ε that is made in the global Newton–Raphson process anyhow.

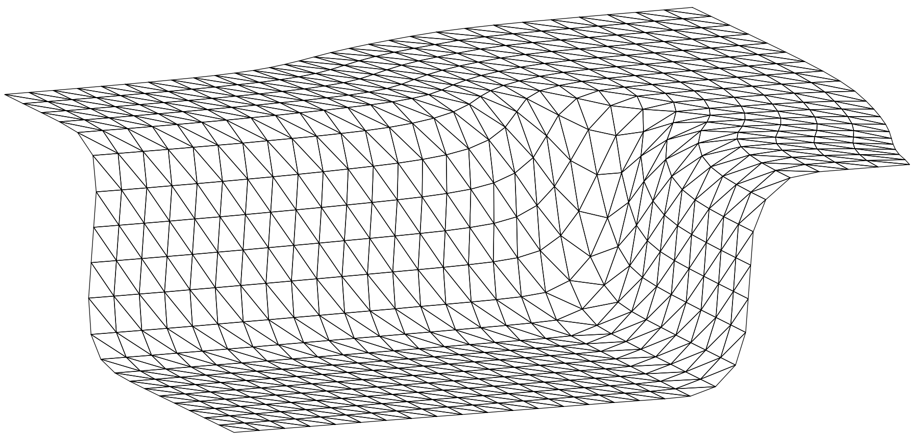


Figure 1: Finite element mesh for a quarter section of a rectangular product.

6 STABILISATION BY DYNAMICS CONTRIBUTIONS

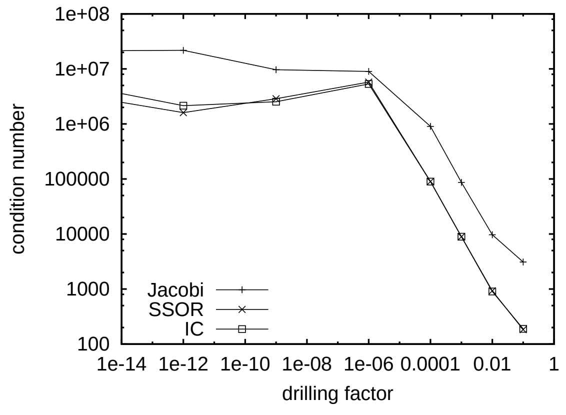
In this section the influence of dynamics contributions on the simulation of deep drawing is investigated. The main goal of the dynamics contributions is to stabilise the calculation and improve the convergence of iterative solvers. First a relatively small model is used to investigate the influence of drilling stiffness, time integration scheme and punch velocity. Then the impact on computation time in a large scale analysis is presented. An extended description is given in [20, 21].

6.1 Deep drawing of a rectangular product

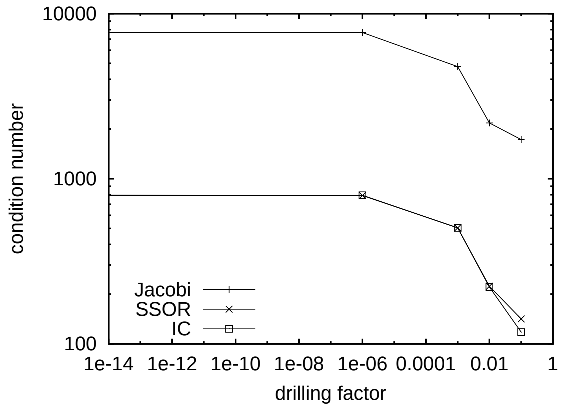
In this section a relatively small model for the deep drawing of a rectangular product is used to determine the influence of several model parameters. The results from the backward difference scheme are used as reference values. The difference with the Newmark scheme and the influence of the drilling stiffness and punch velocity are presented. A lumped mass matrix approach is used.

In Figure 1 the final shape, at a depth of 100 mm, is presented for a quarter section of the product. The initial dimensions of the blank are 313 mm by 221 mm, with a thickness of 1 mm. The unbalance criterion of the global Newton–Raphson iterations according to (19) is set at $\varepsilon = 0.01$

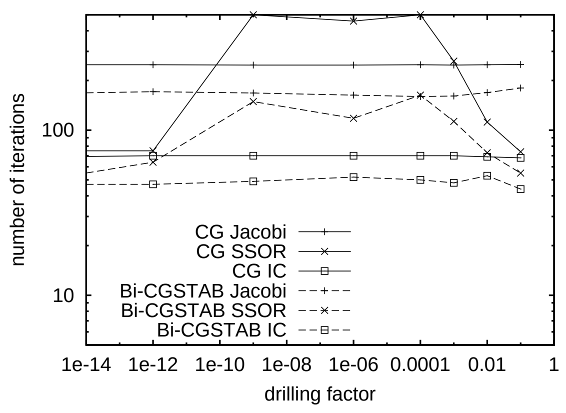
The influence of several parameters on the condition number of the matrix and on the required number of iterations is determined. To this end, for every parameter set 40 increments with 0.5 mm punch displacement per increment were taken, using a direct solver. After 20 mm displacement a large part of the blank is plastic, while some parts in the flange are still elastic. This results in large stiffness ratios that are usually detrimental for iterative solvers. The matrix and right-hand side vector that were assembled in the first iteration of the 40-th step were stored. The condition number for this matrix was derived and the set of equations was solved by a CG and a Bi-CGSTAB method with Jacobi, SSOR and IC preconditioning with a local convergence



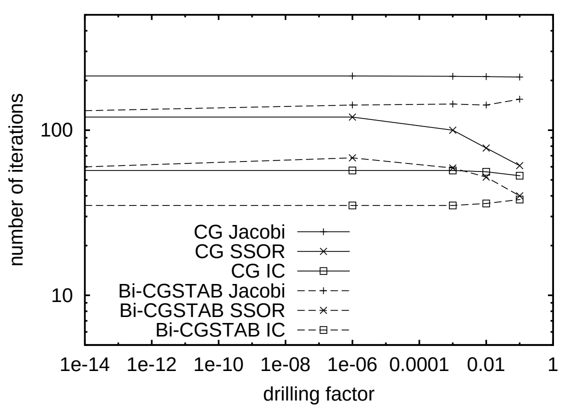
(a) Condition numbers, static



(b) Condition numbers, $v = 10\,\mathrm{m/s}$



(c) Number of iterations, static

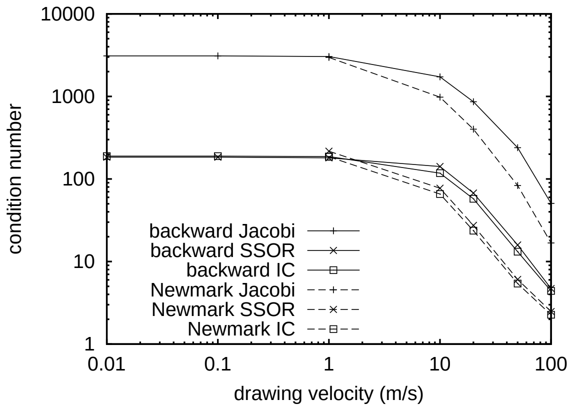


(d) Number of iterations, $v = 10\,\mathrm{m/s}$

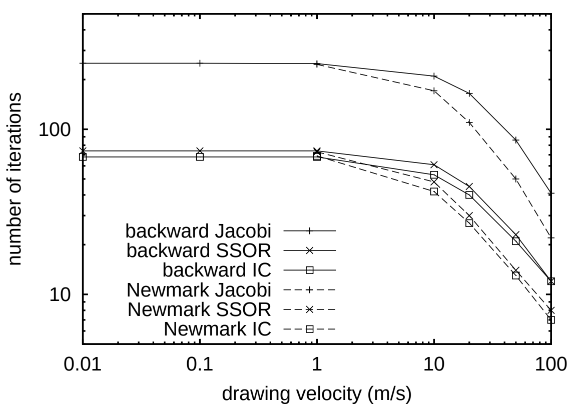
Figure 2: Influence of the drilling stiffness factor α.

criterion $\delta_{\mathrm{it}} = 10^{-5}$ according to (21).

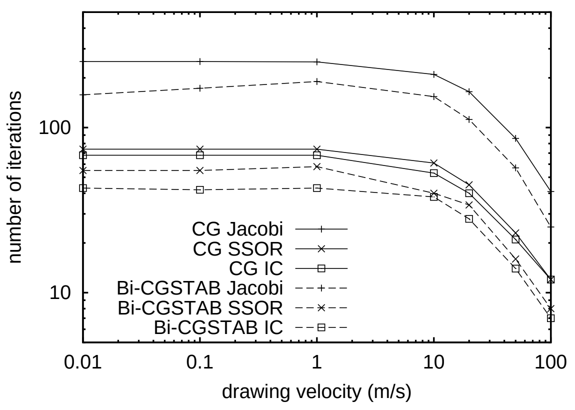
The influence of the artificial drilling mode stiffness (α in (15)) is presented in Figure 2. It can be seen in Figure 2(a) that, in a static analysis, the condition numbers are very high for a low α and start to decrease rapidly for $\alpha > 10^{-6}$. At a punch speed of $10\,\mathrm{m/s}$, Figure 2(b), the mass terms regularise the matrix, but still an improvement of the condition number is achieved for $\alpha > 10^{-6}$. Looking at Figure 2(c) and Figure 2(d) (note the logarithmic scale) it is evident that the condition number of the matrix is not the only relevant parameter. For Jacobi and IC preconditioned methods, the decay in condition number for $\alpha > 10^{-6}$ does not influence the number of required iterations much. However, for the SSOR preconditioned methods in the static analysis, an anomalous increase of required iterations is seen for $10^{-12} < \alpha < 0.1$. This may be due to the particular spectrum of eigenvalues, which is known to influence the convergence



(a) Condition numbers; backward vs. Newmark



(b) Convergence; backward vs. Newmark integration with CG solver



(c) Convergence; CG vs. Bi-CGSTAB solver

Figure 3: Influence of the velocity on condition number and convergence.

behaviour [24]. This detrimental effect does not appear in the dynamic analysis, and some beneficial effect can be observed for the SSOR preconditioner for $\alpha > 10^{-3}$.

In this particular example, the factor $\alpha = 0.1$ did not influence the convergence of global Newton–Raphson iterations and, given the results as presented in Figure 2, this value was used for all subsequent analyses.

In Figure 3, the influence of the punch velocity on the condition numbers and the required number of iterations for the iterative solvers is presented. Since no rate dependent material is used, an increase in punch velocity is equivalent to a quadratic increase in mass density (mass scaling). It can be seen in Figure 3(a) that the condition number decreases for velocities higher than $1\,\mathrm{m/s}$. The number of iterations for the CG solver needed to achieve a residue of 10^{-5} are presented in Figure 3(b). The condition number and number of iterations scale very well. It can also be seen from these graphs that the Newmark method results in a lower condition number and number of iterations, compared to the backward difference method. This can be explained by the stronger weighting of the mass matrix in (11). In this case $\beta = 0.3$ and $\gamma = 0.6$ was used, in order to add some numerical damping. In Figure 3(c) the difference between the CG and Bi-CGSTAB iterative solvers are presented. In all circumstances, the Bi-CGSTAB solver converged within less iterations than the CG solver. But, because one CG iteration is faster than one Bi-CGSTAB iteration, the CG solver was always faster. Because the IC preconditioning is more time-consuming than SSOR, the CG solver with SSOR preconditioning was in almost all circumstances the fastest of the applied iterative solvers. With only 3754 d.o.f. in the matrix, this is a relatively small model and the direct symmetric skyline solver was faster than the iterative solvers for punch velocities below 50 m/s.

The inertial effect also influences the convergence behaviour of the Newton-Raphson procedure. The simulation without dynamics contributions needed 586 iterations for the entire simulation of 200 increments. The simulation of a dynamic process with a punch velocity of 50 m/s, using the backward difference method, needed 502 iterations for the entire simulation. Hence, a reduction of 14 % in the number of iterations is achieved by the stabilising effect of inertia. Both simulations were performed using a direct skyline matrix solver in order to separate this effect from possible effects of iterative solvers.

The thickness strain distribution in the product for a static and a dynamic simulation was presented in [20]. It was observed that the inertial terms *do* influence the thickness strain distribution for a punch velocity of 50 m/s. A simulation performed with a drawing speed of 20 m/s showed hardly any difference between the results for the static and dynamic simulations. Already at this speed, the iterative solvers benefit a lot from the added inertia.

6.2 Front door panel

In this section, the forming of an AUDI front door panel is discussed. This product served as a benchmark for the NumiSheet ’99 conference [26]. The blank holder is doubly curved which gives rise to numerical instabilities while closing the blank holder if dynamics contributions are not taken into account. This automotive product will be used to investigate the performance of the improved implicit code with respect to the conventional implicit code.

Two simulations were performed. For both simulations, the dynamics contributions were switched on, gravity loads were applied and automatic refinement used. The deep drawing velocity was set to 10 m/s. One simulation was performed in which the Bi-CGSTAB iterative solver with SSOR preconditioning was used, and one simulation was performed with a direct solver (Cholesky decomposition).

The results of the simulation using the iterative solver are discussed first. During the first 90 steps, the blank holder is closed. Then the punch moves downwards, until the desired deep drawing depth is reached in step 191. The final deformed mesh is presented in Figure 4.

During deep drawing, the mesh is refined in areas with high curvatures [27]. The initial mesh contains 17244 d.o.f., whereas the mesh ends up with 79344 d.o.f. The generation of new

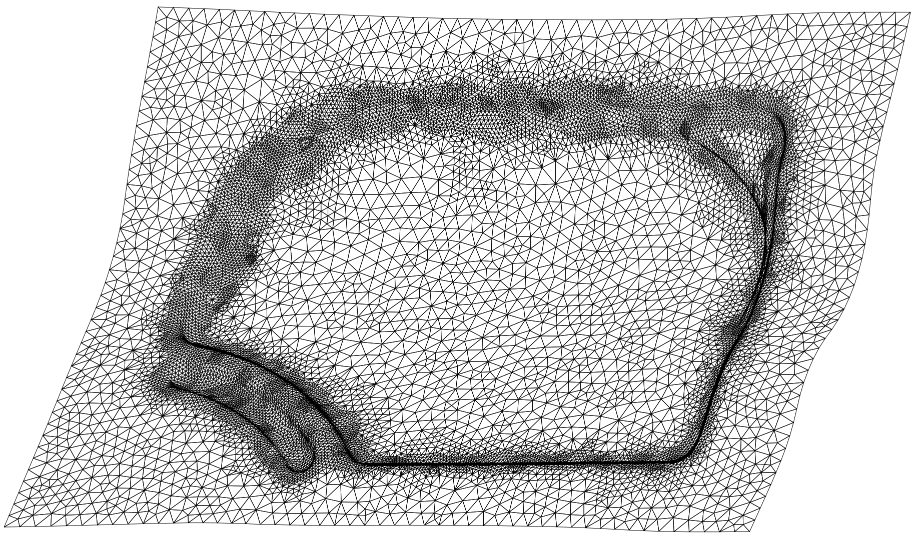
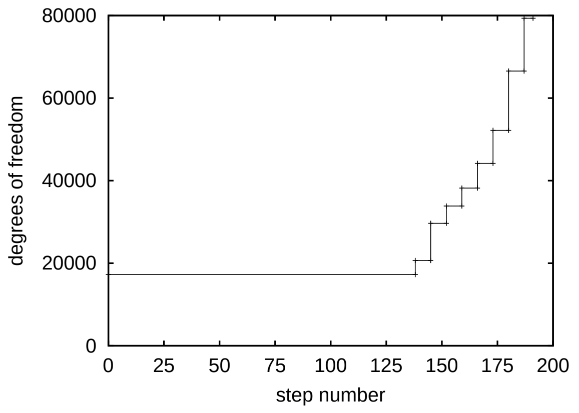
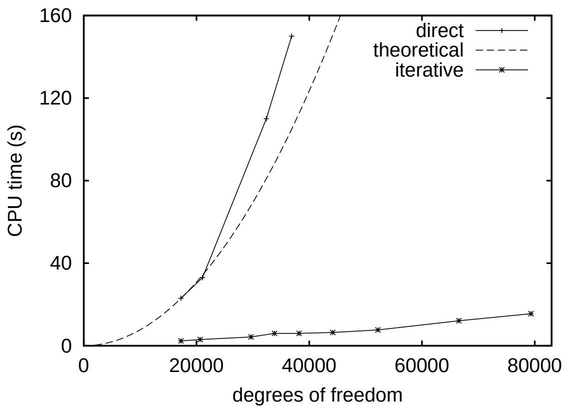


Figure 4: Final shape of front door panel.



(a) Increase of d.o.f. during the simulation



(b) Computation time for direct and iterative solver

Figure 5: Evolution of d.o.f. and solver time in front door panel simulation.

elements during the simulation is graphically represented in Figure 5(a). The computation time for the entire simulation was 5.5 hours on a HP8000 workstation.

Next, a simulation was started using a direct solver. At increment 166 the mesh contained 36876 d.o.f. and another mesh refinement was required. At this moment the simulation could not proceed due to insufficient internal memory. Figure 5(b) shows the computation time for one linear solution with the iterative and the direct solvers. The theoretical increase in computation time for the direct solver, based on uniform refinement proportional to n_{eq}^2, is also included. The figure clearly shows that the iterative solver is significantly faster than the direct solver. For

17244 d.o.f. the iterative solver is 10 times faster than the direct solver while this factor increases up to 25 for 36876 d.o.f. The time for the direct solver increases more than quadratically, which is due to the non-uniformity of the refined mesh. The computation time of the iterative solver increases slightly more than linearly. If the theoretical approach is taken as a lower bound for the prediction of the computation time for the direct solver in the case of 79344 d.o.f., the computation time will increase by at least a factor 40 with respect to the iterative solver. With the iterative solver the total calculation time is no longer dominated by the linear solver.

7 CONCLUSION

The convergence of iterative solvers for sets of equations derived from elasto-plastic shell element models is usually slow. To improve the convergence of iterative solvers, inertial effects can be included in the algorithm. In this way the matrix becomes more diagonally dominant and the condition number of the matrix improves, which is advantageous for iterative solvers. Special attention should be given to the drilling d.o.f. in shell elements. Without (rotational) mass contribution, the presented artificial stiffness largely influences the performance of the iterative solver with SSOR preconditioning. If (artificial) rotational mass is added to this d.o.f., the influence of the drilling stiffness is reduced.

Under realistic deep drawing conditions, the contribution of the mass matrix is not enough to improve the behaviour of iterative solvers. If mass scaling or reduced process times are used, comparable to that used in explicit methods, the reduction in computation time can be tremendous. Another advantage of using dynamics contributions in the implicit finite element code is that it stabilises the Newton–Raphson iterations.

It is shown that significant time and memory savings can be achieved by using iterative solvers with appropriate preconditioners. In the large deep drawing example, the linear solution stage no longer dominates the total solution process.

REFERENCES

[1] K. Mattiasson, L. Bernspång, A. Honecker, E. Schedin, T. Hamman, and A. Melander. On the use of explicit time integration in finite element simulation of industrial sheet metal forming processes. In *Numerical simulations of 3D Sheet Forming Processes*, pages 479–498, Dusseldorf, 1991. VDI-Berichte, VDI Verlag GmbH.

[2] C. D. Mercer, J. D. Nagtegaal, and N. Rebelo. Effective application of different solvers to forming simulations. In S.-F. Shen and P. R. Dawson, editors, *Simulation of Materials Processing: Theory, Methods and Applications*, pages 469–474, Rotterdam, 1995. Balkema.

[3] D. Y. Yang, D. W. Jung, I. S. Song, D. J. Yoo, and J. H. Lee. Comparative investigation into implicit, explicit, and iterative implicit/explicit schemes for the simulation of sheet-metal forming processes. *Journal of Materials Processing Technology*, 50:39–53, 1995.

[4] A. H. van den Boogaard, A. D. Rietman, and J. Huétink. Iterative solvers in forming process simulations. In J. Huétink and F. P. T. Baaijens, editors, *Simulation of Materials Processing: Theory, Methods and Applications*, pages 219–224, Rotterdam, 1998. Balkema.

[5] S. Kacou and I. D. Parsons. A parallel multigrid method for history-dependent elastoplasticity computations. *Computer Methods in Applied Mechanics and Engineering*, 108:1–21, 1993.

[6] R. Mahnken. A Newton-multigrid algorithm for elasto-plastic/viscoplastic problems. *Computational Mechanics*, 15:408–425, 1995.

[7] A. D. Jefferson and H. R. Thomas. Convergence criteria for iterative solvers applied to non-linear plasticity problems. In *Computational Plasticity, Fundamentals and Applications*, pages 441–446, 1997.

[8] R. M. Ferencz and T. J. R. Hughes. Iterative finite element solutions in nonlinear solid mechanics. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of numerical analysis*, volume 6. North-Holland, Amsterdam, 1998.

[9] D. Demarco and E. N. Dvorkin. Modeling of metal forming processes: implementation of an iterative solver in the flow formulation. *Computers and Structures*, 79:1933–1942, 2001.

[10] K. Mocellin, L. Fourment, T. Coupez, and J. L. Chenot. Toward large scale F.E. computation of hot forging process using iterative solvers, parallel computation and multigrid algorithms. *International Journal for Numerical Methods in Engineering*, 52:473–488, 2001.

[11] P. Saint-Georges, G. Warzee, R. Beauwens, and Y. Notay. High-performance PCG solvers for FEM structural analysis. *International Journal for Numerical Methods in Engineering*, 39:1313–1340, 1996.

[12] P. Saint-Georges, G. Warzee, Y. Notay, and R. Beauwens. Problem-dependent preconditioners for iterative solvers in FE elastostatics. *Computers and Structures*, 73:33–43, 1999.

[13] M. Benzi, R. Kouhia, and M. Tůma. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Computer Methods in Applied Mechanics and Engineering*, 190:6533–6554, 2001.

[14] M. Gee, W. A. Wall, and E. Ramm. Parallel iterative solvers in nonlinear shell analysis. *Zeitschrift für angewandte Mathematik und Mechanik*, 81(S2):389–390, 2001.

[15] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. Wiley, Chichester, 2000.

[16] J. L. Batoz, K. J. Bathe, and L. W. Ho. A study of three-node triangular plate bending elements. *International Journal for Numerical Methods in Engineering*, 15:1771–1812, 1980.

[17] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method, volume 2: Solid Mechanics*. Butterworth–Heinemann, Oxford etc., 5th edition, 2000. ISBN 0-7506-5055-9.

[18] T. J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Analysis*. Prentice-Hall, Englewood-Cliffs, 1987.

[19] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method, volume 1: The Basis*. Butterworth–Heinemann, Oxford etc., 5th edition, 2000. ISBN 0-7506-5049-4.

[20] T. Meinders, A. H. van den Boogaard, and J. Huétink. Improvement of implicit finite element code performance in deep drawing simulations by dynamics contributions. *Journal of Materials Processing Technology*, 2003. accepted.

[21] A. H. van den Boogaard, T. Meinders, and J. Huétink. Efficient implicit finite element analysis of sheet forming processes. *International Journal for Numerical Methods in Engineering*, 56:1083–1107, 2003.

[22] J. C. Gelin, B. Liu, and C. Labergere. Stable and accurate algorithms for the simulation of deep drawing and tube hydroforming processes. In E. Oñate, editor, *European Congress on Computational Methods in Applied Sciences and Engineering*, Barcelona, 2000. CIMNE Publishing. on cd-rom.

[23] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 2nd edition, 1989.

[24] W. Hackbush. *Iterative Solution of Large Sparse Systems of Equations*. Number 95 in Applied Mathematical Sciences. Springer, 1994.

[25] R. Barret, M. Berry, T. F. 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, Philadelphia, PA, 1994. URL http://www.netlib.org/templates/Templates.html.

[26] J. C. Gelin and P. Picart, editors. *Numisheet'99, Numerical Simulation of 3D Sheet Forming Processes*, volume 2, Besançon, 1999. Burs Edition.

[27] T. Meinders. *Developments in numerical simulations of the real-life deep drawing process*. PhD thesis, University of Twente, 2000.