A Simple 3D Formulation for Modeling Forging Using the Upper Bound Method

J.M. Pitt-Francis, A. Bowyer, A.N. Bramley (1), University of Bath, U.K.
Received on January 9, 1996

Abstract

A method for modelling perfectly plastic metal flow within the forging process, using the well-known upper-bound energy formulation, is presented. The domain of the metal is described by a tetrahedral mesh, in which the velocity field is piecewise linear. It is shown that this formulation is significantly faster than existing finite element models, and hence can be used for early validation.

The formulation can be reversed readily and an algorithm is discussed which allows nodes to detach from the die surface in an order which optimizes the similarity of the reverse process to user-specified shapes.

Keywords : Forging, Modelling, Upper-bound

1 Introduction

TEUBA stands for "Tetrahedral elemental upper-bound analysis". It is intended that the TEUBA system will predict metal flow in 3-D forgings, in a way which is fast, approximate and easy to use. It will complement the existing finite-element modelling tools which are significantly slow, and are difficult to learn to use. The TEUBA analysis is based loosely on that of the existing University of Bath 2-D axisymmetric forging modeller "UBET" (1), but uses a different method for representing the forging as discrete elements. Alongside the development of the 3-D TEUBA modeller, there is also a 2-D plane-strain form of TEUBA which is being written. It is also intended that TEUBA will have a means of generating the reverse steps in a forging sequence such that the preform shape can be optimized to be similar to a specified shape.

2 Philosophy

The TEUBA approach involves modelling the flow of a plastic material, when it is being driven by a rigid moving boundary. In accordance with the upper-bound formulation the required velocity field $\mathbf{u}(\mathbf{x})$ is the one that minimizes the power integral (subject to the given boundary conditions). This integral is

$$I[\mathbf{u}(\mathbf{x})] = \int_D [\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}]^{\frac{1}{2}}\, dv, \qquad (1)$$

where D is the initial domain occupied by the material, the summation notation is adopted so that the indices i and j are allowed to sum from 1 to 3 and

$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right),$$

is the strain-rate tensor.

This must be done subject to the vector velocity field u satisfying the incompressibility equation, that is,

$$\dot{\epsilon}_{ii} = \sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i} = 0. \qquad (2)$$

Many people have used the upper-bound method to investigate the flow of material in the forging process, for example (1) where the axisymmetric workpiece is divided into elements using a simple bouncing algorithm, and these are investigated using a mass conservation rule. More recently a finite element formulation of the upper-bound method was used by (3) to investigate limit analysis of defective cylindrical shells under internal pressure. Their basic formulation is followed in the current paper, with some important simplifications.

The system is driven by fixed boundary conditions where the component of the velocity which is normal to the die surface must have the same velocity as the die. The components of the velocity at the boundary that are tangential to the die surface may be subject to frictional forces.

The 3-D version of TEUBA uses a tetrahedral mesh to describe the domain occupied by the metal, and triangular shell elements to describe the geometries of the die surfaces. The meshes are imported from a commercial tetrahedral mesher, which is also able to take care of re-meshing the metal domain at a later stage in the forging process.

Both versions of TEUBA are being developed in C on a Unix workstation. There are many similarities between the 2- and 3-D versions, so the code is written for an arbitrary number of dimensions, and the differences are resolved during compilation.

3 Mathematics

In order to find the velocity field which minimizes the power integral (subject to the incompressibility constraint), the workpiece domain is split into a number of tetrahedral elements. The velocity function is assumed to take a linear form in each of the elements. That is, in element l,

$$u_i = \sum_{j=1}^{4} \phi^j(\mathbf{x}) V_i^j,$$

for $1 \leq i \leq 3$, where V_i^j is the component of the velocity in the direction i at the j-th tetrahedral vertex node. The function ϕ^j is linear in element l, vanishing at all nodes except the node j where it takes the value 1.

In order to approximate the minimum value of the power functional given in equation (1) it is necessary to find the minimum value of the discrete approximation to

$$I[\mathbf{u}(\mathbf{x})] = \sum_{l=1}^{E} I_l,$$

where E is the number of elements and I_l is the approximation to I in the domain D_l of element l.

In the discretized system, the functional I_l may be considered as a function of the nodal velocities V_i^j, and thus a stationary value may be found on solving the simultaneous equations

$$\frac{\partial}{\partial V_a^b}\left[I(\{V_i^j\})\right] = 0,$$

for $1 \leq a \leq 3$ and $b = 1, 2, \ldots N$ (referring to a global numbering of the N nodes). The functional I is convex and so the stationary point is assured to be a minimum, and if a search is started at a sensible place then system will always converge to the global minimum.

In the discretized system the first derivatives of the velocity field can thus be represented as

$$\frac{\partial u_i}{\partial x_j} = \sum_{k=1}^{4} V_i^k \frac{\partial \phi^k}{\partial x_j}.$$

which implies that the discrete version of the strain-rate tensor in an element is

$$\dot{\epsilon}_{ij} = \frac{1}{2}\sum_{k=1}^{4}\left(V_i^k \frac{\partial \phi^k}{\partial x_j} + V_j^k \frac{\partial \phi^k}{\partial x_i}\right).$$

The functional I_l in each element l can therefore be represented in the discrete form as,

$$I_l = \frac{1}{2}\int_{D_l}\left[\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{4}\left(V_i^k \frac{\partial \phi^k}{\partial x_j} + V_j^k \frac{\partial \phi^k}{\partial x_i}\right)^2\right]^{\frac{1}{2}}$$

$$= \frac{1}{2}A_l\left[\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{4}\left(V_i^k \frac{\partial \phi^k}{\partial x_j} + V_j^k \frac{\partial \phi^k}{\partial x_i}\right)^2\right]^{\frac{1}{2}}.$$

where k ranges over the four nodes of element l and the domain D_l has volume A_l. The advantage of using linear basis functions can be seen in the equation above, because each of the first derivatives of the basis functions are constants which can be removed from the integration operation.

Consider the expression for $\frac{\partial u_i}{\partial x_j}$. When this is differentiated with respect to a nodal velocity, the result is

$$\frac{\partial}{\partial V_a^b}\left(\frac{\partial u_i}{\partial x_j}\right) = \frac{\partial \phi^b}{\partial x_j}\delta_{ia}.$$

Thus the variation of the functional I with respect to any of the nodal velocity components may be calculated to be

$$\frac{\partial I}{\partial V_a^b} = \frac{\partial}{\partial V_a^b}\sum_{l=1}^{E}\int_{D_l}\sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\,dv$$

$$= \frac{1}{2}\sum_{l=1}^{E}\int_{D_l}\left[\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}\right]^{-\frac{1}{2}}\frac{\partial}{\partial V_a^b}\left(\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}\right)dv.$$

The expression $[\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}]^{-\frac{1}{2}}$ obviously depends on all the values $\{V_i^j\}$, but since the dependence is highly nonlinear, values are guessed, and the solution is obtained iteratively. For an element l, the calculated energy is defined by

$$W_l = \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}$$

$$= \frac{1}{2}\left[\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{4}\left(V_i^k \frac{\partial \phi^k}{\partial x_j} + V_j^k \frac{\partial \phi^k}{\partial x_i}\right)^2\right]^{\frac{1}{2}}.$$

where k varies over the four nodes of the element, and all the values $\{V_a^b\}$ are guessed or known values from a previous iteration. The variation of element l with respect to V_a^b is thus

$$\frac{\partial I_l}{\partial V_a^b} = \frac{A_l}{2W_l}\left[\sum_{i=1}^{3}\sum_{k=1}^{4}\left(V_a^k \frac{\partial \phi^k}{\partial x_i} + V_i^k \frac{\partial \phi^k}{\partial x_a}\right)\frac{\partial \phi^b}{\partial x_i}\right].$$

and the variation of the entire power integral I is a simple sum of these variations, that is,

$$\frac{\partial I}{\partial V_a^b} = \sum_{l=1}^{E}\frac{\partial I_l}{\partial V_a^b},$$

although only those elements which are adjacent to node b will have a non-zero term in this sum.

Thus an equation can be written down for the variation of each of the components of each of the nodal velocities. These equations are linear (except for the functions W_l) and can therefore be solved iteratively using a sparse matrix solver. The nodes in the mesh

can be preprocessed in such a way that the bandwidth and profile are minimized, and so a banded symmetric Cholesky matrix solver is used to solve each set of simultaneous linear equations.

In the formulation so far, no allowance has been made for the incompressibility of the material, as represented by equation (2). This restraint is tackled by making use of the *penalty function method.* In the TEUBA modeller it is necessary to force the metal flow to be divergence free (incompressible), and for this purpose it is necessary to impose the condition $\epsilon_{ii} = 0$, everywhere in D. With a little thought it can be seen that

$$\sum_{i=1}^{3} \epsilon_{ii} = 0 \quad \text{in } D \quad \Longleftrightarrow \quad \int_D \left(\sum_{i=1}^{3} \epsilon_{ii} \right)^2 dv = 0.$$

The proof from left to right is obvious since the integrand on the righthand-side is a vanishing function. The reverse proof can be seen from the fact that the integrand is a non-negative function, and if its integral is zero it must vanish everywhere in the domain D.

For the purposes of the iterative scheme (described in the previous section) the functional

$$J(\mathbf{u}) = \int_D \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\, dv + \lambda \int_D (\epsilon_{ii})^2\, dv,$$

will be considered. Since the minimization of either functional I or J requires an iterative scheme, the value of the parameter λ can be varied at each step of the calculation. If λ is small then it will be impossible to satisfy the incompressibility condition, but as $\lambda \rightarrow \infty$ the numerical errors incurred by the presence of a large numbers in the matrix solver will be expected to swamp the scheme.

In order to find a sensible value of λ to use in the calculation one must be able to measure the success of the penalty function, and vary the value accordingly. At the start of the iterative scheme, values are set for the two parameters λ and θ. These values are arbitrary, and are determined empirically. At the end of each step of the calculation, the *compressibility,*

$$\begin{aligned} K &= \int_D [\epsilon_{ii}]^2\, dv \\ &= \sum_{l=1}^{E} \left\{ A_l \left[\sum_{i=1}^{3} \sum_{k=1}^{4} V_i^k \frac{\partial \phi^k}{\partial x_i} \right]^2 \right\}, \end{aligned}$$

is calculated. If the compressibility is significant then the penalty term λ is multiplied by the arbitrary factor θ and the next estimate of the iteration is calculated. If the compressibility converges to a value (which should be close to zero), then the penalty factor λ is held constant for the rest of the calculation - that is, until the velocity field convergences.

A mesh is read in from the tetrahedral mesher after it has been processed for bandwidth reduction, using the Gibbs, Pool and Stockmeyer algorithm, see (2). In the present stage of development sticking friction boundary conditions are set automatically on the top and bottom surfaces. This restraint means that only simple upsetting can be investigated, but TEUBA is soon to be upgraded to include a full die description, a contact recognition algorithm, and a friction model. This means that industrial components with a high degree of geometric complexity can be investigated with TEUBA, such as can be modelled by commercial finite element forging software.

All free nodal velocity components are set according to their position from the centre of gravity with a small random variation (in order to avoid the situation where $W_l = 0$ for some l, when the denominator of I_l becomes singular). Initial values are chosen for λ and θ. The iteration then proceeds with λ multiplying by a factor of θ on each step until the *compressibility* convergences to zero. The iteration then continues with a constant value of λ until the velocity field converges too.

After the mesh has been moved according to the velocity field at each node, then the iteration procedure is started again. For this new increment the old values of the velocity field are reused (since they must be quite close to the values that are to be calculated next).

4 Performance

Early indications are that the TEUBA modeller is fast, and reasonably accurate. Obviously its disadvantage when compared with full finite element modellers is that the crudeness of the algorithm leads to a lack of accuracy and detail. Several simple examples have already been run using the TEUBA modeller.

One example involved modelling the upsetting of a cylinder to half of its original height and comparing against the UBET code. An axisymmetric UBET calculation was run first and a similar (but three-dimensional) TEUBA calculation proceeded. Both examples used the yield stress of medium carbon steel for load prediction and were performed by a Unix workstation. Both upsetting calculations took about 20 seconds to run, and TEUBA was slightly faster (despite the fact that UBET effectively works in two dimensions). The load predictions (which are upper-bound estimates) from the two programs deviated from each other by a maximum of about 5% during the complete operation.

A second example involved the plane-strain upsetting of a rectangle of material. Here the commercial finite element package was used for the triangular meshing, and the comparison. The mesh had 258 nodes and it was found that the finite element routine took 67 increments to reduce it by a quarter of its height. The 2-D version of TEUBA was accordingly run with the same number of increments. It was found that the finite element code took 1

minute of CPU to perform the first increment and a total CPU time of 20 minutes, while TEUBA took 3 seconds to complete the initial increment and 40 seconds to complete the operation. The TEUBA code was also reversed, and the result can be seen in Figure 1. It is expected that TEUBA will be slightly slower once a contact recognition algorithm is introduced.

A 3-D example was also run. A finite element modeller took a total 22 minutes to upset a square prism mesh (with 407 nodes) by a tenth of its height. TEUBA took 6 minutes to duplicate the calculation, in the same number of increments.

5 Reversibility

It can be proved that the upper-bound is reversible in a continuous sense, but it will obviously fail over a large time-step because the velocity components will differ greatly. Various authors, including (4) have successfully implemented a reverse model by iterating back and forth over the time-step.

The main problem with these models is deciding when nodes should become detached from the die surface. Algorithms involving surface pressure have been used, but seem to have little physical reasoning behind them. The method used by (4) in their axisymmetric analysis was to step the forging backwards for each possibility of a node detaching. Using a *shape complexity factor* they then compared the resulting geometries to see which one was most like a cylinder, and took this route as being the ideal reverse increment. Thus the preform shape was always assured to be like a cylinder (which is an ideal stock shape for axisymmetric forging).

There are obvious complications when extending this scheme to three dimensions, but it is possible. The ideal forging stock shape (cylinder, cuboid etc.) can be selected by the forger, and at each reverse increment the possible shapes can be tested against the ideal shape using a *shape comparison function*. Suitable functions for this shape comparison are currently being tested.

6 Conclusion

A formulation has been invented for the kernel of the TEUBA modeller. The basic machinery of integral minimisation (subject to incompressibility) with piecewise linear elements is all in place. A number of features are being added to this kernel so that the modeller can be used for real industrial forging examples by the novice user. Still more work needs to be done in order to have a modeller which works *automatically* in reverse mode.

References

(1) Bramley, A. N., 1987 Computer aided forging design. Annals of the CIRP. **36**, 135-138.

(2) Everstine, G. C., 1979. A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront. Int. J. Num. Meth. Engng, **14** No. 6, 837-853.

(3) Liu, Y. H., Cen Z. Z. & Xu B. Y., 1994 A numerical method for plastic limit analysis of 3-D structures. Int. J. Solids Structures **32**, No. 12, 1645-1658.

(4) Zhao, G., Wright E. & Grandhi R. V., 1995 Forging preform design with shape complexity control in simulating backward deformation. Int. J Mach. Tools Manufact. **35**, No. 9 1225-1239.

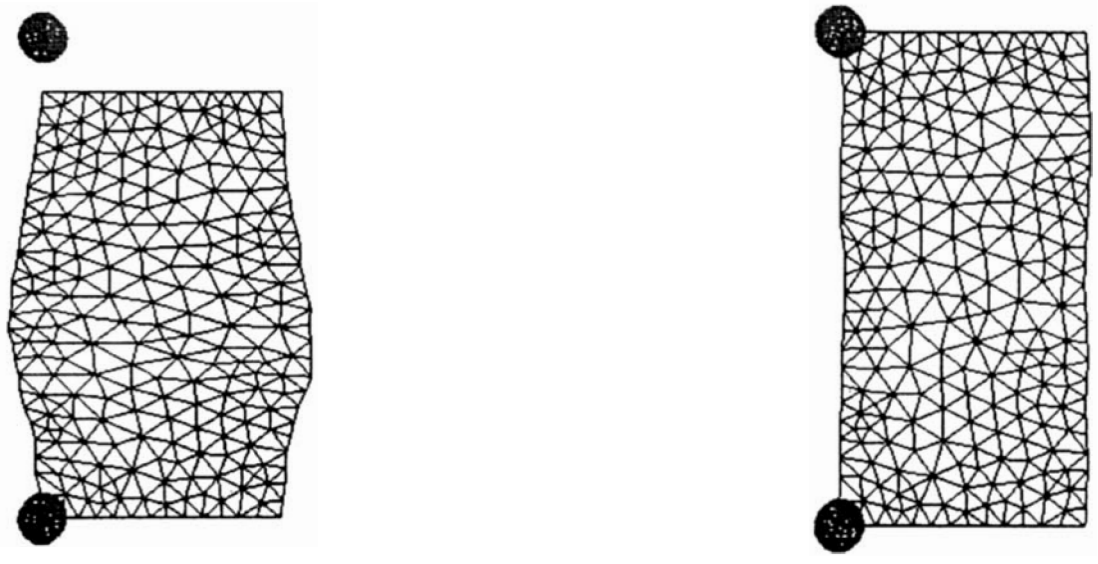

Figure 1: The picture on the left shows a TEUBA plain-strain upsetting during the calculation, while the one on the right is the result of reversing the process. (The circles are for reference to the initial shape).