Parallel multi-constrained partitioning schemes applied to the problems of deep drawing processes of sheet metal

A. Schmidt[1]

Abstract

Different optimization schemes are used in the conceptual design stage to obtain the solution domain. The results are then used as a starting point for the problem simulation. We address the problem of development of parallel simulation of the solution domain by using the sparse computation schemes. Our contribution is the analysis of the performance of the proposed preprocessor and postprocessor. An example demonstrating the developed method is presented considering a steel metal in the deep drawing process.

Keywords: Parallel partitioning, iterative optimization, greedy algorithm, deep drawing;

1. Introduction

We explore in this paper the sparse computation schemes which are a good approach to increase the performance through parallel simulation. The literature review revealed that the application of parallel schemes in deep drawing processes is rare. The application of Greedy algorithms has been studied (Cansell & Méry, 2007), but no details revealed for drawing processes. Studies on the influence of anisotropy (Padmanabhan et al., 2009) on the optimal blank shape showed dependencies among process parameters without consideration of the potential to break down the problem to the parallel domains. Another approach(Hestenes & Stiefel, 1952) requires significant efforts due to necessity to build a function for adaptive response sampling method and translate it to optimization instructions. There are proposals (Ambrogio et al., 2006), (Schenk & Hillmann, 2004) to improve the obtained results in incremental forming in analytical form, but again their nature is difficult to translate to instructions. Different factors including heating, punch speed, holder force(Palumbo et al., 2007), and mesh pitch (Bellenger & Coorevits, 2005),(Cansell & Méry, 2007), affecting the quality of the optimized form are studied for various metals.

2. Problem formulation

The objective of optimization is for a given domain to determine the form of the blank, force distribution, and boundary conditions by minimizing the system. We have adopted the sparse computation schemes (Ambrogio et al., 2006) for the step-by-step development. For the parallel iterative solver we use the conjugate gradient methods (Dongarra et al., 1991), which proved to be efficient when the number of elements is large and the amount of communication is significant due to the 3-dimentional nature of the problem. Assuming set Ω represents the given domain and is not empty by definition. The greedy method

[1] *Corresponding author.* Department of Mechanical Engineering, Lipetsk State Technical University, Lipetsk, Russia. Tel.: +1 (773) 293-3115; fax: +1 (773) 213-9907. E-mail address: me@alexschmidt.net (A. Schmidt).

is used over this set to compute an optimal solution, which we define as L. We assume that the solution L defines the criteria for optimality, S defines the next possible step of optimization, then the greedy(L,S) is an operator defining pairs with respect to Ω.

2.1. Domain decomposition

Consider a set of candidates C for the possible optimal solution L. Our goal is to optimize the set C by minimizing or maximizing the value of an objective function using Greedy algorithms. We are using the theory proposed by Curtis (Curtis, 2003), which we adopted to the problem using four classes for greedy problems. The development process has the initial greedy model Ω, a greedy refinement model for the best-global principle and, finally, the refinement model for getting an algorithmic expression. We assume that the components of the greedy solution are defined as rep, opt, quotient, domain and lambda and that they satisfy the properties for domain defined by theory. The detailed description of the aforementioned components can be found at (Curtis, 2003). Now, we have the following defined:

S – step of the algorithm;

C – criterion for the global optimality and a pre-order E;

Greedy – step of the greedy algorithm;

Initial_value – initial value of the algorithm.

Preorder $P(O)$, calculated using the following formula:

$$P(O) = O \in E \leftrightarrow E \wedge id(E) \subseteq O \wedge (O;O) \subseteq O. \quad (1)$$

We start with the initialization as the initial event by starting the execution of the system by assigning any value to the solution L. Procedure *compute* computes an optimal solution among the possible ones, where by definition solution ϵ E. We choose criteria and choose to assume that the mathematical structures will satisfy the best-global principle. Now, let us explain the algorithms of the best-global principle, which we are using to evaluate the mathematical model when an invariant is obtained. The related properties are added to introduce the function for modeling the repetition and the next property is derived from the assumptions over the current model. Next, the current model S is refined by modifying the *compute*, which computes an optimal solution among the possible ones. The new refinement C2 defines the next computation step and the next invariant. As for the previous invariant, a solution exists and not empty by definition. A new variable contains the current value and is called *current*. The initial invariant is refined and the next model is introduced until the predefined event is not triggered, which indicates that the final invariant is obtained.

2.2. Shape partitioning procedure

In this section, we review the partitioning procedure which is used to further optimize the process of obtaining the model invariant. We employ parallel algorithms, which are proved to provide better overall performance for the modeling. Consider the system for the modeling:

$$Ax = b \quad (2)$$

This system is solved using the Newton-Raphson method. In scalar form, the system can be rewritten as follows:

$$\tilde{A}x = \tilde{b}, \quad (3)$$

where A is stiffness matrix, x is displacement vector, and b is force vector.

The finite element mesh is partitioned into N contiguous partitions of n elements. For each partition, a processing node is assigned. The global matrix $\tilde{A}$ is assembled from the partition matrices as:

$$\tilde{A} = \sum_{i=1}^{N} \widetilde{A}_{\iota}, \tag{4}$$

where each partition is assembled in the local matrix as:

$$\widetilde{A}_{\iota} = \sum_{e=1}^{ne_p} \tilde{A}_e \tag{5}$$

The factorization of the sparse partition matrix is expressed:

$$\left(I + \tilde{A}_i - \widetilde{W}_i\right) \equiv L_i^0 U_i^0, \tag{6}$$

where L_i^0 and U_i^0representing the lower and upper incomplete factors respectively.

Based on this, it is possible to write the partitioned pre-conditioner system, developed by Liou and Tezduyar (Liou & Tezduyar, 1991) as follows:

$$\tilde{P} = \prod_{i=1}^{N} L_i^0 U_i^0 \tag{7}$$

$$\tilde{P}^{-1}\tilde{A}x = \tilde{P}^{-1}\tilde{b} \tag{8}$$

According to the formula (8) mesh is decomposed into clusters of elements. The element-level matrices are assembled to cluster-level matrices, and the pre-conditioners are formed as sequential products of these cluster-level matrices. We propose a method based on preconditioning and utilizing this concept. Here, a single cluster resides in each partition as well as all related information required for the formation of the pre-conditioner.

At last, we measure the performance of the proposed modeling based on the pre-conditioners. The other instantiation of the modeling is post-processing of the obtained invariants and measurement of its performance. First, we consider the greedy part of the refinement, followed by the performance validation of the established procedures.

2.3. Shape optimization procedure

We have an abstract model Ω, with some variable x and a corresponding invariant$I(x)$, which is refined using a concrete model Ω_{C}, with variables y, and an invariant $J(x, y)$. Respectively, an abstract $A(x, x')$and a concrete predicate$B(y, y')$, define the following:

$$I(x) \wedge J(x, y) \wedge A(y, y') \Rightarrow \exists x' \cdot \left(A(x, x') \wedge J(x', y')\right) \tag{9}$$

or, in other words, there exists a pair of abstract invariant $I(x)$and a concrete invariant $J(x, y)$such that a transformation $(\exists x')$ from the abstract $A(x, x')$to concrete $B(y, y')$by means of the invariant $J(y, y')$exists. The refinement is identified using a predicate$C(y, y')$, which leads to the transformation as follows:

$$I(x) \wedge J(x, y) \wedge C(y, y') \Rightarrow J(x, y') \tag{10}$$

We assume that the solution will terminate, i.e. the predicate $C(y, y')$ converges, or, otherwise it is infinite. Thus, with each next transformation an invariant $J(y)$is decreasing and the following statement is derived:

$$I(x) \wedge J(x, y) \wedge C(y, y') \Rightarrow J(y') < J(y) \tag{11}$$

Finally, we set up the guards to make sure the derived statement is feasible and the invariants are coherent, namely,

$$I(x) \wedge J(x, y) \wedge G(A) \Rightarrow G(B) \tag{12}$$

where $G(A)$is the guards of the abstract model events and $G(B)$is the guards of the concrete model events.

Next, we perform the functional domain decomposition of the model Ω, which could be divided to improve the computation speed. Explicit decomposition with sub-structuring technique is based on the knowledge derived from the physical description of the problem. The computation is performed with using parallel algorithms for solvers. There are two known types of parallel solvers, which are direct (Gaussian) solver (Curtis, 2003), and iterative (gradient) solver (Atkinson, 1989). Because of the size of the problem, the iterative solver is preferred over another, since it provides better convergence. For parallel computation, the message passing interface (Foster, 1995) is used. Before the optimization is started, the model analysis is performed and its mesh is broken up into independent pieces, then each one is scheduled to parallel computation. The finite element system, boundary conditions, initial conditions, material properties, and global optimization criteria are propagated to every partition, since these are applied to the model globally and do not carry large footprint on the initial problem description. Next, the optimization loop is established assuming that an invariant $J(y)$for the transformation exists. For the first iteration, the initial model must be obtained, which is derived using empirical equations of raw estimation of the final form.

3. Implementation

The parallel multi-constrained partitioning schema was implemented using an existing finite element package MSC.Marc 2005. The implementation of the method is sufficiently enough that it should be possible to reproduce in any finite element code. Some technical choices described here are dependent on the software package that was used. Namely, the parallel portion of the computation engine was executed using NT-MPICH library originally developed by Karsten Scholtyssik and now maintained by Silke Schuch from RWTH Aachen University in Germany, which is a free implementation of MPICH from Argonne National Laboratory (Forum, 1994)(Foster, 1995). The optimization procedure was implemented as a shell pre-conditioner in MSC.Marc developed by the author. For executing the optimization procedures was used an Intel-based machine with 2.66 GHz 8 processors, 4 GB RAM running Windows Server operating system.

4. Example geometries

In order to evaluate the efficiency of the algorithm, two benchmark problems have been studied. Box with flange is presented in Fig. 1. The geometry in Fig. 2 of circular cup can be also used to analyze the influence of inner-outer profile on the blank shape. In all of the problems, the mesh of different sizes has been studied, 20x20, and 40x40, in which the first number refers to number of elements in the x-direction

and the second refers to that along the y-direction. Depending on the step of the problem, 1 through 8 processors was used for parallel computation with the analysis performed on the commodity hardware.

4.1. Example 1: Geometry of box with flange

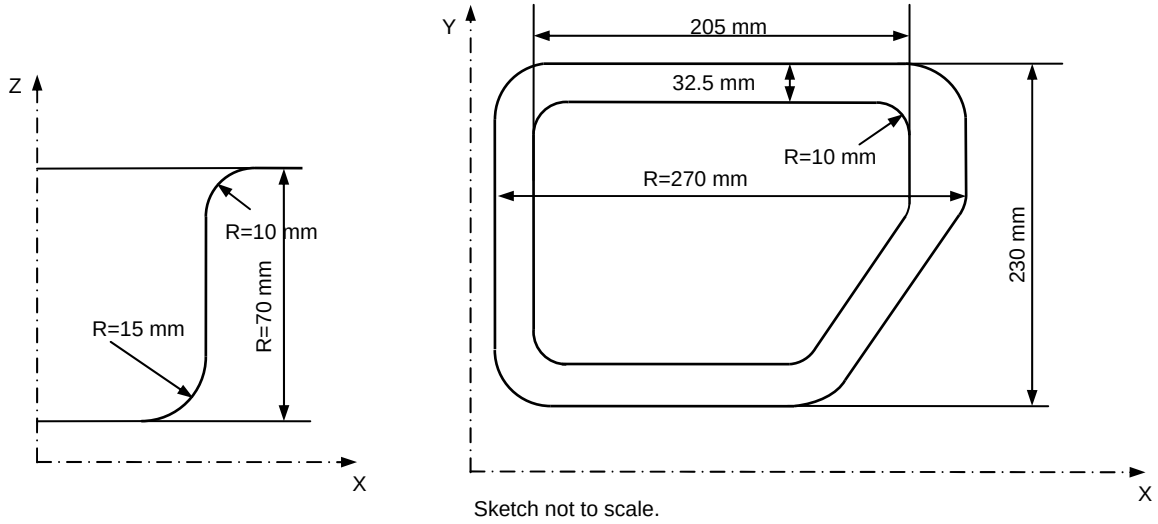


Fig. 1. Box with flange geometry.

4.2. Example 2: Geometry of circular box

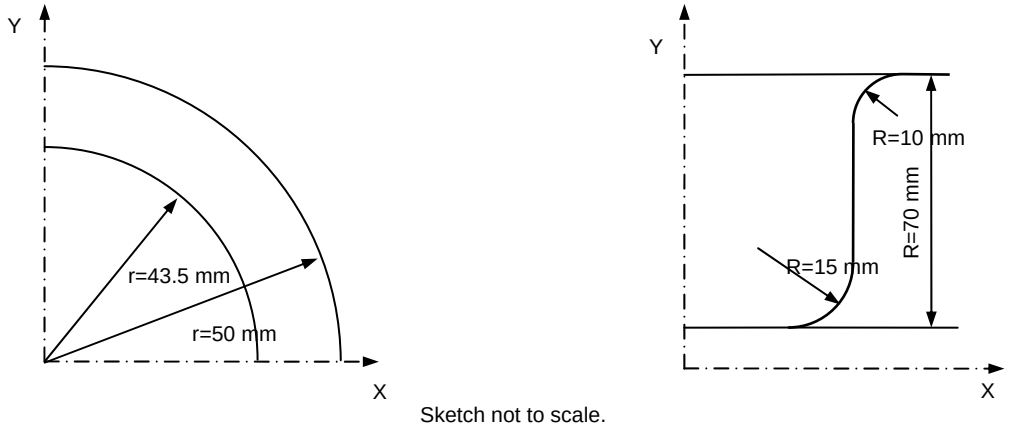


Fig. 2. Round cap geometry.

The blank shape optimization governed by the material flow characteristics, anisotropy, and through a yield criterion. The extensive analysis of the problems makes it feasible to capture the influence of geometry and process parameters on the final result of the deformation.

Table 1 – Material properties of aluminum allow (A 6070-O) and mild steel (DC06).

Property	**A 6070-O**	**DC06**
E	70 GPa	210 GPa

N	0.33	0.3
Y_0	95 MPa	120 MPa
K	145 MPa	530 MPa
N	0.0979	0.268
t	1.5 mm	1.5 mm

Table 2 – Process parameters for box (example 1) and cup (example 2).

Property	**Example 1**	**Example 2**
Punch pitch	70 mm	70 mm
Stress	70 GPa	210 GPa
Number of steps	100	100
Contact	Static friction coefficient = 0.1	Static friction coefficient = 0.1

Table 2 shows the parameters of the processes for two investigated examples of the box and cup drawing processes, correspondingly. Because of the small anisotropy, the sheet is considered to be isotropic. Blank is considered to be deformable, while die, punch, and blank holder are assumed to be rigid.

Move over, the performance of the developed parallel iterative solver is analyzed.

5. Performance and evaluation procedures

5.1. Performance evaluation

The efficient implementation of the parallel iterative solver was developed, which allows to perform the post-processing of the initial model and to obtain the modified and final optimized blanks that show the measured distribution of transformation timings per domain and computation unit.

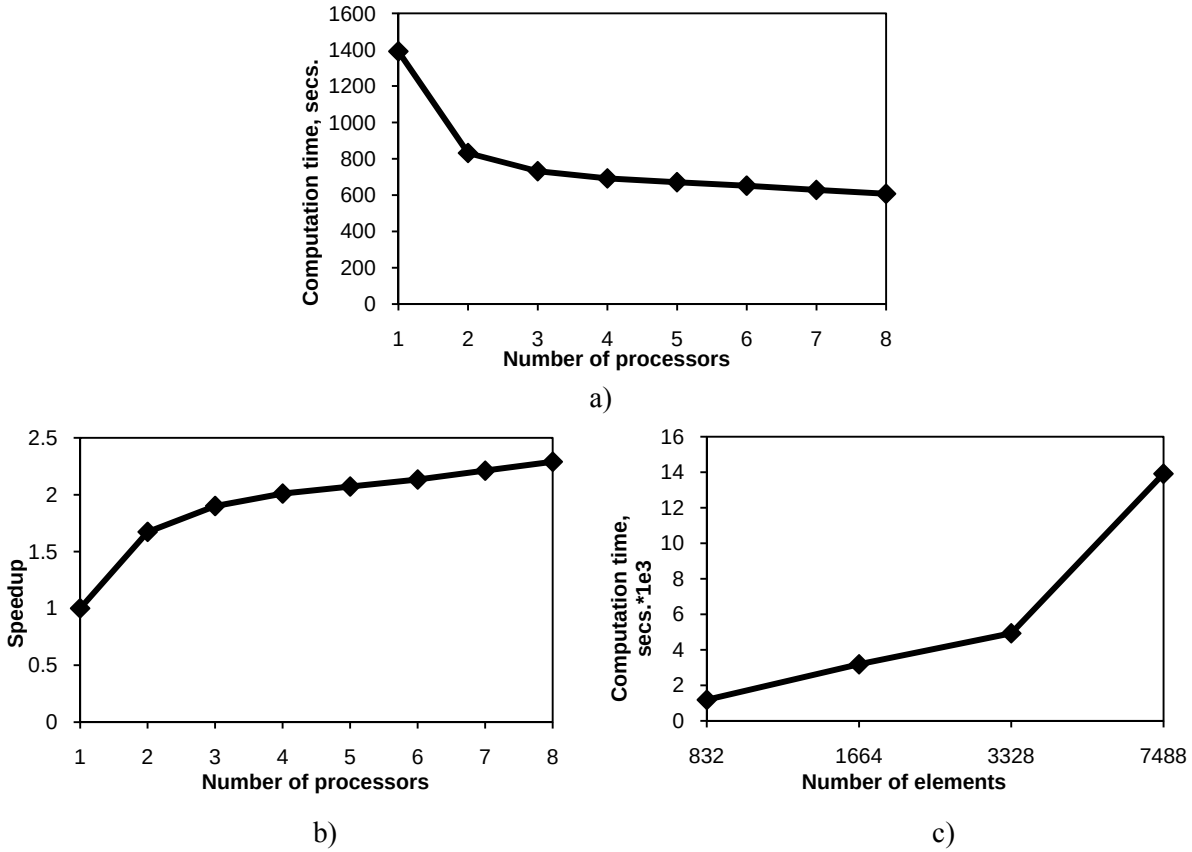


Fig. 3. Performance charts for sample problems.

Fig. 3 a) presents the graph of the performance evaluation of the sample problem as a function of number of processors to computation time. It can be observed that the computational time speeds up with the number of processors and reaches an asymptotic limit. The limit can be derived according to an argument put forth by Gene Amdahl as formulae (Amdahl, 1967):

$$SPA = \frac{1}{(1-R) + \left(\frac{R}{N}\right)} \tag{13}$$

$$M_{SPA} = lim_{N\to\infty} SPA = \frac{1}{1-R} \tag{14}$$

Where SPA is the Algorithmic speed up, $M_{SPA}$is the theoretical limit of the speed up, N is the number of processors, $R = \frac{P}{S+P}$ is the ratio of the parallel code execution to the total time, S is the serial time, P is the parallel time.

According to the performance chart depicted on Fig. 3, we can conclude that by increasing the mesh size, number of partitions, and calculating units the modeling time is observed to be reduced. There is no benefit to increase the mesh size beyond 40x40 elements, 8 partitions and 8 processors and the computational gain will not worth of cost of adding calculating units. Obviously, the number of partitions is derived through logical analysis of the model, whereas there are eight different profiles can be derived from the given model, and number of partitions is equivalent to number of calculating units. Increasing

the number of units per partition saturates the benefits due to the overhead related to the increase in ratio of communication to computation time for each unit.

5.2. Performance of parallel partitioning

In this section, the validation and performance of the parallel partition is performed and compared with sequential procedure. A number of test problems which parameters are listed in Table 3 were considered. In each case different number of partitions and processors were in use and computation time was obtained for the test problems.

Table 3 – Mesh parameters for validation of parallel partitioning performance.

Mesh	Name	Type	Nodes	Elements	Partitions
1	Punch	Quad	258	105	1, 2, 4, 8
2	Cup with flange	Quad	401	360	1, 2, 4, 8
3	Box	Quad	1681	1600	1, 2, 4, 8

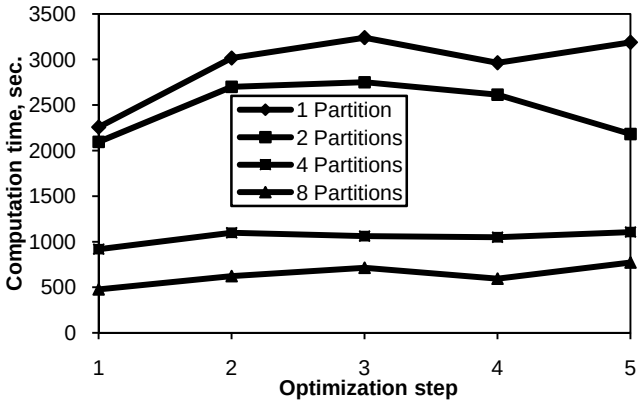


Fig. 4. Computation time over optimization step and number of partitions (box).

Fig. 4 shows the dependency between the number of partitions and distribution of the timings over optimization steps.

It is evident that the initial timing is due to non-positive definiteness before the final shape is reached. As the optimizing procedure performs an optimization of the shape, the computation time diminishes. In the process of analysis in the shape-optimization loop the time is given per iteration and number of partitions. The computations performed with greater number of partitions exhibits that less time consumed per optimization step. It is amply showed that with greater number of partition domains observed an improvement in timing distribution in the process of the part forming.

5.3. Evaluation of optimized solution

Evaluation of the scalability and performance of the sparse schemes is performed for stabilized finite element formulation of the model. The timing for a Newton-Raphson iteration is based on the single step, which includes the element-level computation, assembly of matrices, and obtaining of a solution of the resulting linear system of equations with using Gauss method. Three different meshes with different pitch are used for evaluation of the scalability and performance which are listed in Table 4.

Table 4 – Meshes for evaluation of the scalability and performance.

Mesh	Name	Type	Nodes	Elements	Boundary Conditions
1	Punch	Quad	258	105	1
2	Cup with flange	Quad	401	360	2
3	Box	Quad	880	832	2

The timings of the computation are split into the components as follows:

1. Communication timings (CT).
2. Formation of the linear system of equations timings (FS).
3. Partition level products timings (PR).
4. Computation timings (CM).

All four components are summed into overall time:

$$T_a = \sum_{i=0}^{4} t_i \tag{15}$$

Using the following metrics, we perform evaluation of the formation and partition components. The sample problems listed in Table 4 are evaluated and results are listed in Table 5.

Table 5 – Computations timing for sample problems.

Mesh	N*	CT	FS	PR	CM (%)
1	1	3.56	37.39	8.60	6.95
1	5	4.00	18.73	4.46	14.26
2	1	27.84	1.08	2030.15	19.84
2	5	26.91	0.38	1145.05	10.47
3	1	55.93	824.13	1377.27	17.64
3	5	82.02	1234.08	1872.14	16.22

*N – optimization step.

It is observed that the formation and partition components exhibit the growth depending on the convergence and the size of the problem. Communication costs increase with the size of the problem and complexity of the elements in the mesh.

6. Discussion on results

6.1. Influence of thickness variation

Different factors, causing variations in the thickness in sheet stamping, discussed in the work (Huang et al., 2006), were expanded and included in consideration with parallel optimizations. The objective of this study is for a given domain to minimize the thickness variations in the formed part. Depending on the material properties, different earring profiles are formed or disruption of metal continuity occurs. Fig. 5 shows thickness variations along the x, y and 45 degree predicted by simulation. It also shows the dependency between distance from the center of the blank and distribution of thickness. It is observed that the thickness variation towards the outer side of blank is due to high deformation and suboptimal shape; as the number of optimization increases, the thickness variation diminishes. As shown for the case of

initial and optimized final blank, the mentioned variations are reduced considerably. This leads to more uniform distribution of strains that may be vital for a drawing process.

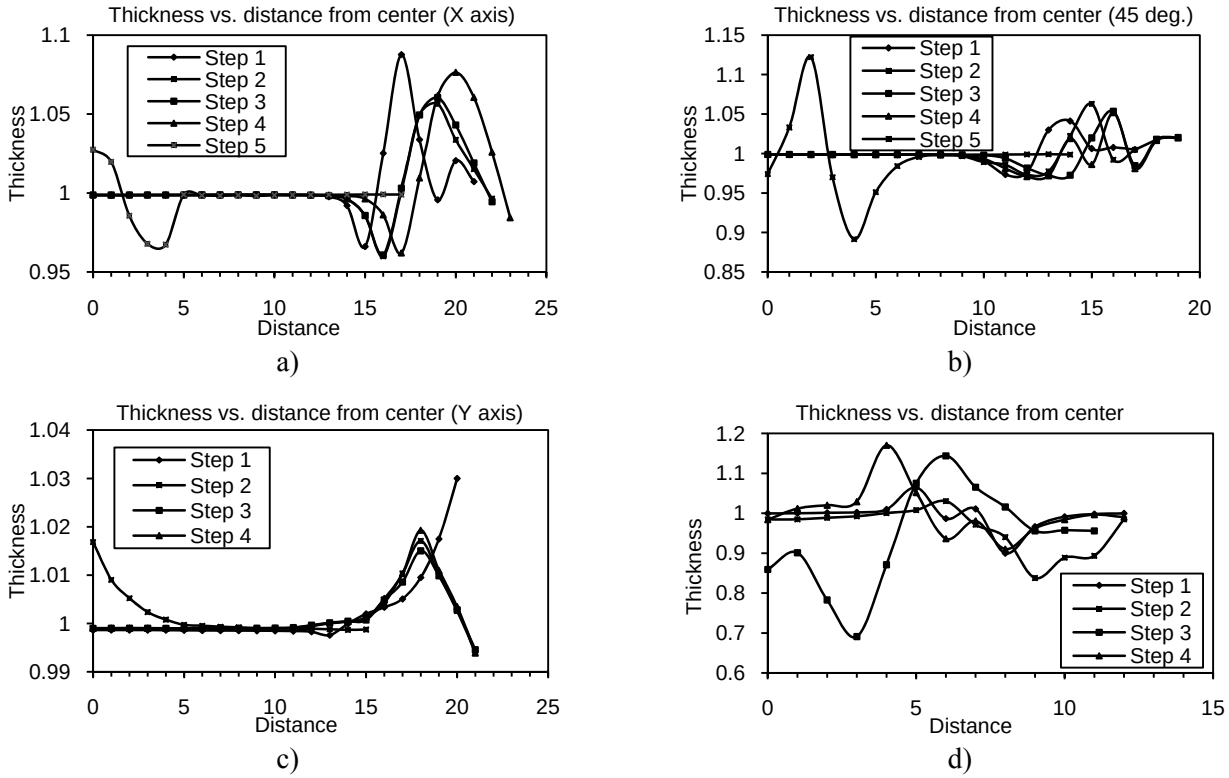
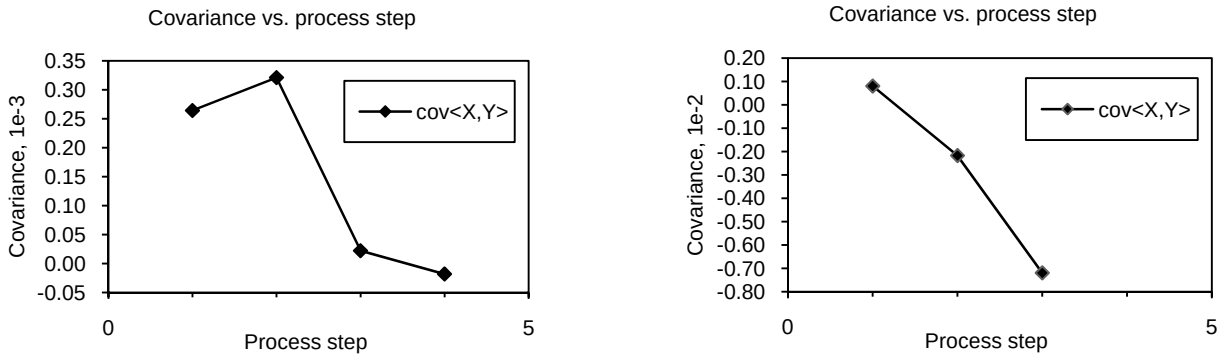


Fig. 5. Thickness variations predicted by simulation in x, y and 45 degree directions.

Fig. 5 shows that excessive flange was observed for both geometries using initial blanks. It is also observed that the thickness variation in the center is probably due to the low mesh pitch. In the first iteration, shown in Fig. 5 a) along X axis, the thickness reduction is less due to the smaller initial length along this direction. Thereafter, during the consecutive iterations, the thickness reduces due to the larger geometry along this direction. Fig. 5 b) shows the thickness variation along the Y axis. Marginal difference in thickness occurred between iterations, due to similar dimensions of the blanks along this direction. Fig. 5 c) shows the thickness variation long the diagonal direction.

Due to thinning in sheet metal formed parts the defects occur and hence distribution of the thickness contributes directly to the quality of part.



a) b)

Fig. 6. a) Thickness covariance over number of process steps in box. b) Thickness covariance over number of process steps in cup.

To analyze the distribution of thickness in the formed part, the covariance of initial invariant$I(x)$ and formed geometry $J(x,y')$ was calculated to obtain the measure of the strength of the correlation between the two invariants (see Sec. 2.3 Shape optimization procedure).

By definition, the covariance is calculated as follows:

$$cov(X,Y) = \langle (X-\mu_X)(Y-\mu_Y) \rangle \tag{16}$$

where $\mu_X = \langle X \rangle$and$\mu_Y = \langle Y \rangle$are the respective means, replacing X with $I(x)$ and Y with $J(x,y')$, the equation can be written explicitly as

$$cov(X,Y) = \langle (I(x)-\mu)\big(J(x,y')\big) \rangle = \sum_{i=1}^{N} \frac{(x_{i-}\bar{x})(y_i-\bar{y})}{N} \tag{17}$$

Taking into account a consideration of the theoretical equality of invariants $I(x) = J(x,y')$, it then can be derived as follows:

$$cov(X,Y) = \langle X^2 \rangle - \langle X^2 \rangle = \sigma_X^2 \tag{18}$$

The thickness covariance shown on Fig. 6 in the formed part should be minimized to provide a good solution by improving the flow of the material in the instrument cavities by foregoing dynamic analysis with partition domains. Fig. 6 a) shows the thickness covariance in the box which is growing in the first iteration due to the suboptimal blank profile determined using empirical equations. Thereafter, from the second iteration, the covariance reduces to a larger optimization. Fig. 6 a) shows the thickness covariance in the cup which is reduces from the first iteration due to a simpler geometry and high prediction through empirical equations for this shape.

6.2. Optimal blank shape

Next, the forming results in different strength depending on the directions of the blank. This induces considerable difference in the flow of material and hence the results in the blank profile in the formed part.

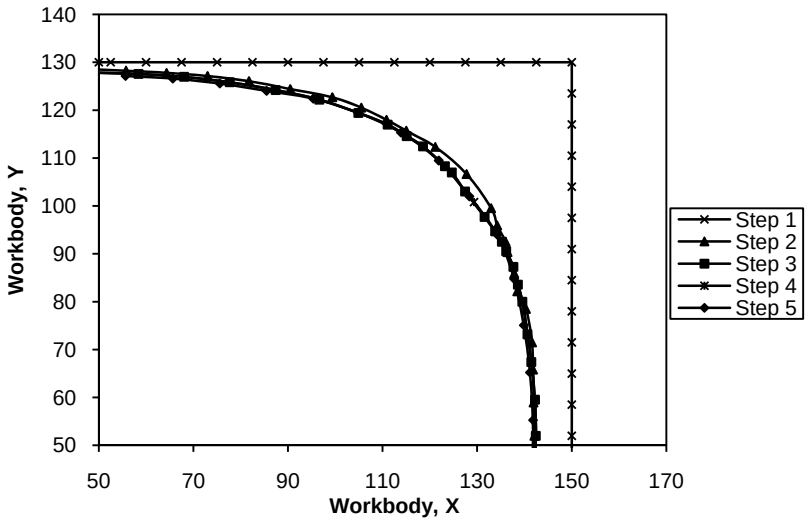


Fig. 7. Blank shape profile predicted by optimization algorithm.

To demonstrate the robustness of the developed greedy algorithm, the initial blanks were considered for the flow process. The initial blank contour shown on Fig. 1, has width of 270 mm, and height of 230 mm, and final profile after each step of deep drawing of box is shown on Fig. 7. The flow of the material differs at various directions due to the intrinsic material properties, anisotropy, yield stress, and strain-stress dependency. Fig. 7 shows the length and blank profile, using A 6070-O material models at the end of the each iteration in blank shape optimization procedure. As the number of iterations increase, the flange contour gets closer to the target contour. The shape error reaches a value of 1.60 mm within the five iterations as shown in Fig. 7. Any further iteration has a negligible impact on the required blank shape profile.

7. Concluding remarks

In this paper we presented the parallel methods for simulation of the solution domain by using sparse computation schemes. The resulting iterative sparse-based solver shows good scalability and enables carrying out simulations in parallel manner in short turnaround times. Tests involving typical forming problems indicate that greedy algorithms are capable of providing better optimized shape forms.

References

Ambrogio, G., Cozza, V., Filice, L. & Micari, F., 2006. An analytical model for improving precision in single point incremental forming. *Journal of Materials Processing Technology*, pp.92-95.

Amdahl, G.M., 1967. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings*. Atlantic City, N.J., Apr. 18-20, 1967. AFIPS Press, Reston, Va.

Atkinson, K.A., 1989. *An Introduction to Numerical Analysis, 2nd edition*. New York: John Wiley & Sons.

Bellenger, E. & Coorevits, P., 2005. Adaptive mesh refinement for the control of cost and quality in finite element analysis. *Finite Elements in Analysis and Design*, pp.1413-506.

Cansell, D. & Méry, D., 2007. Incremental Parametric Development of Greedy Algorithms. *Electronic Notes in Theoretical Computer Science*, (185), pp.47-62.

Cavin, P., Gravouil, A., Lubrecht, A.A. & Combescure, A., 2005. Efficient FEM calculation with predefined precision through automatic grid refinement. *Finite Elements in Analysis and Design*, pp.1043-55.

Curtis, S.A., 2003. The classification of greedy algorithms. *Science of Computer Programming*, (49), p.125–157.

Dongarra, J., Duff, I., Sorensen, D. & van der Vorst, H., 1991. *Solving Linear Systems on Vector and Shared Memory Computers*. Philadelphia, PA: SIAM.

Forum, 1994. MPI: a message-passing interface standard. *International Journal of Supercomputer*, p.159–416.

Foster, I., 1995. Message Passing Interface. In *Designing and Building Parallel Programs (Online)*. Addison-Wesley. Ch. 8.

Hestenes, M.R. & Stiefel, E., 1952. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6).

Huang, Y., Lo, Z.Y. & Du, R., 2006. Minimization of the thickness variation in multi-step sheet metal stamping. *Journal of Materials Processing Technology*, pp.84-86.

Hu, W., Yao, L.G. & Hua, Z.Z., 2008. Optimization of sheet metal forming processes by adaptive response surface based on intelligent sampling method. *Journal of Materials Processing Technology*, pp.77-88.

Liou, J. & Tezduyar, T.E., 1991. A clustered element-by-element iteration method for finite element computations. In R. Glowinski, ed. *Domain Decomposition Methods for Partial Differential Equations*. Ch. 13. pp.140-50.

Padmanabhan, R. et al., 2009. Numerical study on the influence of initial anisotropy on optimal blank shape. *Finite Elements in Analysis and Design* , (45), pp.71-80.

Palumbo, G. et al., 2007. Numerical and experimental investigations on the effect of the heating strategy and the punch speed on the warm deep drawing of magnesium alloy AZ31. *Journal of Materials Processing Technology*, pp.342-46.

Press, W. ., Flannery, B. ., Teukolsky, S. . & Vetterling, W. ., 1992. Sparse Linear Systems. §2.7. In *Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed.* Cambridge, England: Cambridge University Press. pp.63-82.

Schenk, O. & Hillmann, M., 2004. Optimal design of metal forming die surfaces with evolution strategies. *Journal Computers and Structures*, pp.1695-705.