Software Framework for Application of Automatic Optimization Techniques to Metal Forming

Igor Grešovnik

Abstract

An optimization environment is presented that has been designed for application in forming technology and in other engineering fields. A number of difficulties that can be encountered when dealing with realistic industrial problems call for a multidisciplinary solution approach. An optimization shell has therefore been developed with an open and extensible architecture that enables harmonized operation of a variety of tools specialised for different tasks, which is necessary for efficient solution of optimization problems. The concept is demonstrated on selected examples: optimization of tool shape in sheet forming, optimal pre-stressing of cold forging dies, and optimal design of material with periodic microstructure.
Computational complexity of the numerical models and substantial noise in numerical results are common difficulties that aggravate employment of classical optimization techniques An algorithm based on successive local approximations of the response functions with restricted step approach has been constructed in order to overcome these difficulties. Current work is focused on the design of an optimization library in order to promote systematic development of such algorithms.

1 Introduction

Intensive application of automatic optimisation to the design of forming technologies has not been often reported until recently. The complexity of the forming processes from the physical modelling and numerical point of view often aggravate successful utilization of optimization techniques, which would result in observable economic benefits. In spite of this, there is a strong potential for optimisation in forming industry due to high production volumes and a number of practical problems that have not yet been adequately solved. Earlier work in this area focused predominantly on inverse identification of model parameters [1],[2], making an important contribution to understanding of material response and contact conditions. Other works addressed a limited number of fundamental issues related to optimization [3]-[5]. This enabled automatic optimization of process parameters such as tool or pre-form shape, loading path, lubrication, pre-stressing conditions, tool materials, etc., in order to achieve objectives such as precise tolerances, better performance of the products, targeted structural changes in material, durability of tools, reliability and low cost of the process.

When solving realistic industrial examples, a number of problems are encountered, which can only be overcome by close collaboration between numerical analysts and industrial designers and require application of up-to-date expertise in different engineering fields.

The first prerequisite is a good knowledge of material behaviour. This is seldom satisfactory, therefore preliminary research supported by numerical tools is required to derive adequate material models and to obtain trusted values of model parameters.

Boundary conditions must be determined precisely enough to allow accurate simulation of the process. Heat transfer and friction between the bodies in contact must be estimated. These phenomena are hard to investigate because complex microscopic phenomena play a crucial role and it is not always possible to perform meaningful in-situ measurements. Numerical support to the design of forming technology often requires adaptation of traditional manufacturing procedures, since these do not ensure enough uniform processing conditions.

When simulating forming processes, highly non-linear, path dependent material behaviour with coupled thermal, mechanical and other phenomena is dealt with, large deformations are involved and sometimes multi-scale simulation approach must be adopted. This together with complex geometry leads to large-scale numerical models where parallel solution schemes may be considered.

When the above mentioned issues are adequately solved, numerical simulation can be employed to support the design of the forming process or formed part. This requires knowledge of the process at a technologist's level and eventually of the broader aspects such as the state of the market, production capacities, cost breakdowns, etc. This is necessary to define the objectives and constraints for optimisation procedures in such a way that results will suggest advantageous changes in the design while not conflicting with technological limitations.

Mastering the described requirements is usually achieved in a step by step manner for a certain class of applications. In the scope of this work, a software framework is being designed for supporting the application of automatic optimisation in forming technology. The framework is flexible enough to be applicable to a large variety of problems. It supports multidisciplinary development and systematic updating of solution methodologies for classes of similar problems.

In the subsequent sections, a rough outline of the framework is given first, followed by analysis of some individual features and brief description of selected application areas.

2 Optimization Environment

Rather than building a large, complex homogeneous system that could alone deal with complete problems, the adopted approach was to connect and harmonise smaller specialised units. By having in mind such a modular system, an optimisation shell [6]-[10] has been developed whose primary aim is to control the execution of external analysis programmes within the optimisation loop.

Early experience with practical problems identified the need for high flexibility with respect to problem definition and combination of solution utilities. The shell was therefore built around a powerful interpreter designed for driving numerical applications (Figure 1). An *shell interface library* for accessing the interpreter and other shell functionality enables easy integration of diverse software tools. These are driven in an integrated scheme with centralised data access and synchronised execution flow ([9],[10]). Beside for the solution of practical optimization problems, the system is also intended to serve as development and testing environment.

The optimization shell connects optimization algorithms with the evaluation of the objective and constraint functions at a given set of design parameters, which typically involves a finite element numerical analysis [12] of the process. Interpretation of a specific *analysis* block of code in the command file is performed each time the evaluation is required by an algorithm. The user defines the evaluation of the objective and constraint functions within this code block. Execution of the numerical analysis is controlled by interface functions attached to the interpreter, as well as data exchange between the shell and the numerical analysis environment. Exchange of the current values of design parameters and simulation results between the optimisation algorithm and the numerical analysis is enabled through pre-defined shell variables. Intermediate functions take care of this, which enables straightforward linking of optimization algorithms from their original libraries [9].

Current development of modules incorporated in the optimization shell is focused on general utilities that are closely related to optimization. The simulation and sensitivity analysis tools have been mainly developed separately from the optimization shell and integrated in the pre-application stage. An interfacing methodology has been elaborated for the finite element simulation systems, which enables the shell to have a full control over execution of the finite element analysis and access to its data. Such approach proved suitable for setting up optimization schemes, while possible applications exist beyond that. The scheme can be applied to add flexibility to the simulation environment itself, even when used as stand-alone, e.g. to enable user interference during the analysis or to enable easy switching between available solvers, to facilitate evaluation of user-defined quantities expressed as integral over the time or spatial domains, etc.

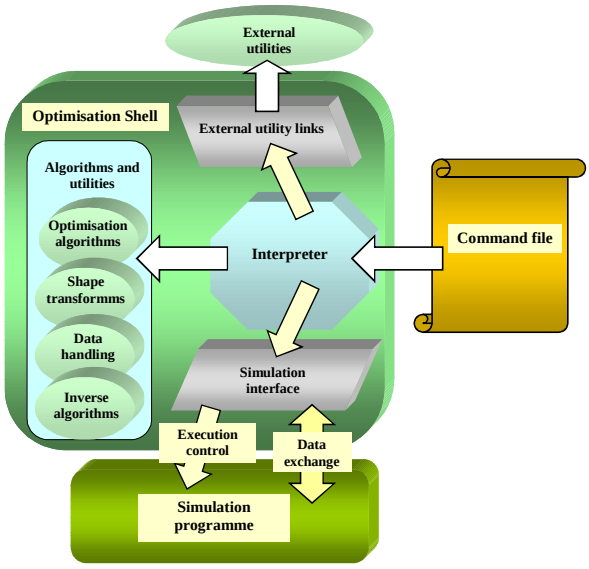


Figure 1: Functional scheme of the optimization system.

2.1 Incorporation and Usage of Algorithms

One of the crucial demands for the optimisation system is the possibility of providing a number of different optimisation algorithms and other utilities. It is also desirable that algorithms are implemented and accessed in a unified way. It should be easy to switch between different algorithms even without any particular knowledge of their function.

Incorporation of new algorithms into the shell is straightforward. Each algorithm is represented in the shell by the corresponding interpreter function, which the user calls to perform the algorithm. In order to define such a function, a corresponding C function must be implemented, linked with the shell code and installed to the interpreter system under a given name. The type of such a function is prescribed. Its only argument is a pointer to the structure that represents the shell interpreter, through which all data regarding the state of the environment can is accessible. The algorithm must be called within the body of this C function.

The user of the shell will define input and output data of the algorithms through arguments of the corresponding *interpreter function*. These data must be extracted within the corresponding C function that calls the algorithm by using the appropriate shell interface library functions. All of these functions conveniently take as argument the pointer to the structure that represents the interpreter. During the interpretation of the command file, this structure contains all the necessary data for extracting arguments as have been passed by the user, and the corresponding shell interface library functions take care of extraction and necessary conversions. Incorporation of new algorithms into the shell does therefore not require any particular knowledge of the shell structure.

Algorithm output is treated in a similar manner than its input, except that the corresponding arguments that the user passes to the relevant interpreter function usually specify names of interpreter variables that will store the output data. Again, the shell interface library functions are called to obtain the storage locations corresponding to these variable names, and to copy the output data to these locations. Since algorithm implementations usually come with their own format of input and output parameters, some simple data conversions between the data types known to the algorithm and the shell supported data types are necessary.

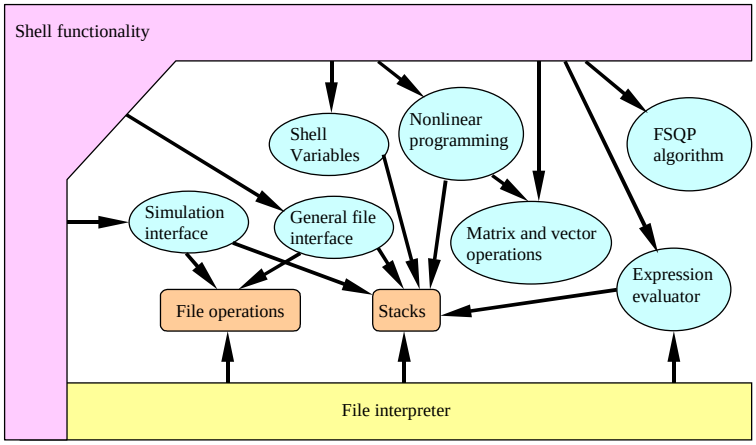


Figure 2: Organization of shell modules. Arrows show dependencies.

Shell functionality with the incorporated algorithms and other utilities is arranged in a hierarchical structure. A cutout of this structure is shown in Figure 2, with indicated relations between different modules. A more elaborate discussion of the shell structure and function can be found in chapter 4 of [9].

3 Direct Shape Parameterization Approach

Many problems in forming are related to the shape of the tools or work piece. Include geometric design in automatic process, one must be able to represent these shapes continuously in terms of a finite number of design parameters, in a form suitable for numerical simulation. A number of practical approaches to this task employ concepts of parametric design developed for computer graphics and CAD [13], a practical example of which can be found in [14].

A different approach has been developed in order to meet the requirements imposed by practical problems in metal forming. These are related to the efficiency when used with large scale problems, ability to adopt the existent preliminary design produced by technologists as a starting guess in optimization, sufficient generality with respect to geometrical layouts for which the approach is applicable. Furthermore, the approach should be well suited for combination with the sensitivity calculation in order to enable efficient optimization techniques, and should work both with fixed topology of the finite element mesh and automatically generated mesh.

In light of the above stated goals, a direct parameterization technique has been constructed where the geometry of objects considered in numerical simulation is defined by parameter dependent maps [16]. A framework for the parameterization is based on two-stage transforms from the reference to physical domain and is sketched in Figure 3.

The part of the object that is parameterized is mapped to the reference domain where the co-ordinates of material points are deemed independent of shape parameters. In order to obtain different initial boundary shapes of the object, a parameter dependent transform is applied to the reference co-ordinates of material points consisting the object, such that

$$\mathbf{x}^{(0)} = \left(x^{(0)}, y^{(0)}\right) = \mathbf{F}(\xi, \eta, \mathbf{p}),$$

where

$$\mathbf{F}(\xi, \eta, \mathbf{p}) = \mathbf{f}(\mathbf{g}(\xi, \eta, \mathbf{p})).$$

$\mathbf{x}^{(0)}$ denotes initial co-ordinates of a material pint, ξ and η are its reference co-ordinates, $\mathbf{p}$ are shape parameters and $\mathbf{F}$ is the composed map applied to obtain physical co-ordinates from the reference ones.

The reference geometrical layout is obtained by mapping a chosen initial design from the physical to reference system by applying the inverse map, $\mathbf{F}^{-1}\left(\mathbf{x}_i^{(0)}, \mathbf{p}\right) = \mathbf{g}^{-1}\left(\mathbf{f}^{-1}\left(\mathbf{x}_i^{(0)}\right), \mathbf{p}\right)$, at some initial parameters that are chosen to produce that design. This can comprise the finite element mesh of the parameterized domain in the

case of fixed mesh topology, or just the boundary definition in the case that automatic mesh generation is applied. Derivatives of the initial co-ordinates of material points are obtained by differentiating **F** with respect to the shape parameters, where the chain rule is applied:

$$\frac{d\,\mathbf{F}(\xi_i,\eta_i,\mathbf{p})}{d\,p_j} = \frac{d\,\mathbf{f}(\xi,\eta)}{d\,(\xi,\eta)}\frac{d\,\mathbf{g}(\xi,\eta,\mathbf{p})}{d\,p_j}.$$

The term initial co-ordinates is used here for non-deformed configuration in the physical space, at any values of the shape parameters. In the case of fixed mesh topology, the reference co-ordinates (ξ,η) are calculated in advance for all nodes of the mesh. When automatic mesh generation is applied, they must be obtained for each node of the generated mesh by application of $\mathbf{F}^{-1}$ to its physical co-ordinates.

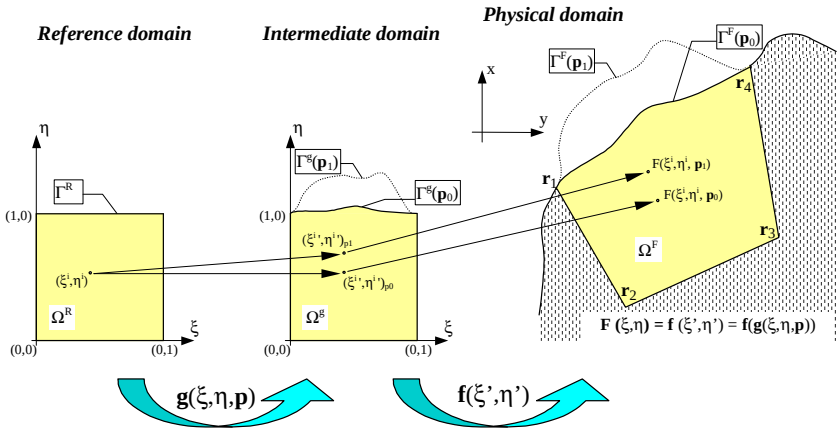


Figure 3: Representation of the shape parameterization approach.

The basic property of the applied shape transform is that its second stage **f** is independent of shape parameters and is designed in such a way that **g** acts on a simple domain. **f** and **g** can therefore be explored independently and the above described prototype parameterization algorithm can be used. The second stage **f** was fixed for 2D and 3D case, for 2D being

$$\mathbf{f}^{2D}(\xi,\eta) = \mathbf{r}_1 + (-\mathbf{r}_1+\mathbf{r}_2)\xi + (-\mathbf{r}_1+\mathbf{r}_4)\eta + (\mathbf{r}_1-\mathbf{r}_2+\mathbf{r}_3-\mathbf{r}_4)\xi\eta.$$

The points $\mathbf{r}_1$ through $\mathbf{r}_4$ define the parameterized region in the physical space. The inverse $\mathbf{f}^{-1}$ can be calculated analytically in 2D, while a modified Newton algorithm is used for calculation of inverse in 3D.

f is considered only on a domain in the reference space above the unit interval in 2D or unit square in 3D (Figure 3); points outside this region are not mapped within the domain affected by the parameterization. In certain cases, **f** is not unique on the considered domain. The problem is treated by restriction of this domain in such a way that **f** is unique on the restricted domain. This is achieved by a-priori restricting the range of values of the first stage transform by composition of an additional function (monotonous by limited range) upon the last co-ordinate of the image of **g**.

Since **f** is considered only on a domain above the unit interval in 2D (or unit square in 3D), the parameter dependent part of **F** is conveniently defined (in 2D) as a stretch according to some parameter dependent function *s*, i.e.

$$\mathbf{g}(\xi,\eta,\mathbf{p}) = (\xi, s(\xi,\mathbf{p})\cdot\eta),$$

where *s* is a parametric family of curves defined over a unit interval (2D case) or surfaces defined over a unit square. This is a convenient feature because inversion of **g** involves merely direct calculation of *s*, and the complexity of calculation is reduced to calculation, differentiation and inversion of **f** and to calculation and differentiation of *s*. Different ways of defining *s* were considered. In 2D case,

cubic splines, Bezier curves, linear combination of gauss-like curves and the diffuse approximation of a set of values were considered. Bezier surfaces and linear combination of Gauss-like surfaces were considered in the 3D case.

The appropriate algorithms for evaluation of *s* and its parametric derivatives were developed in a separate module with the application interface that enables seamless integration in the routines for manipulating the complete transform. This is demonstrated e.g. in the technical specification of application interface for spline calculation [17].

3.1 Example Applications of Shape parameterization

The first example of application of shape parameterization is in can forming. Market demands encourage can manufacturers to produce cans in variety of attractive shapes. For this purpose, additional forming stages are introduced in the production process. During the shaping stage, a cylindrical metal sheet is formed by a tool placed inside the sheet and expanded outwards in radial direction (Figure 4).

Non-uniform expansion of the sheet causes its thinning and can eventually lead to tearing of the material. The spring-back effect is responsible to considerable changes in can shape after unloading where tools are drawn back, because of which the final can shape is not directly related to tool shape.

The tool shape was optimized with the intention of compensating for the spring-back effect in view of achieving the desired shape of the can within given tolerance and at the same time minimizing the peak effective stress within the can after forming. Other objectives such as maximizing the volume of the can were also considered.

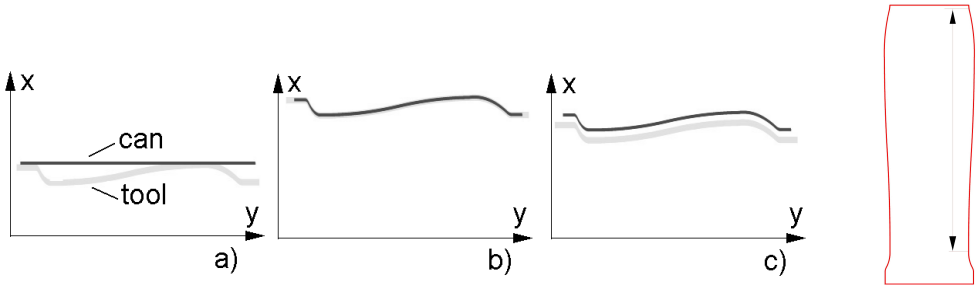


Figure 4: Stages of the can shaping. Contour of the tool and the blank is shown a) before, b) during and c) after the tool expansion. The produced can shape is shown on the right hand side.

Figure 5 shows the resulting shape obtained by maximization of volume, with prescribed constraint on admissible shapes and maximal effective stress attained during forming. The sequential quadratic method was used for solving the problem. Derivatives of the objective and constraint functions were obtained by direct differentiation of the finite element model [19]. In further work, improvement of the shaping process by optimization of kinematics of tool segments and tribological conditions has been considered [18].

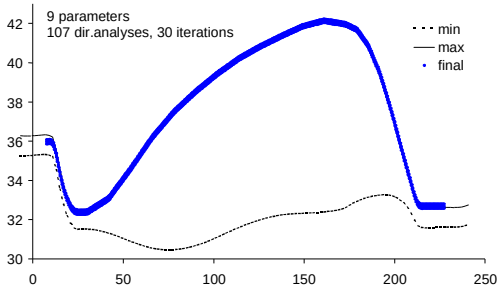


Figure 5: Final can shape with indicated upper and lower admissible shape constraints.

Another typical application is optimal pre-stressing of cold forging tools (Figure 6) [9].

Cold forging dies are subjected to high operational loads which often lead to fatigue failure. To reduce growth of fatigue cracks the

dies are used in a pre-stressed condition. This is achieved by putting the die in a stress ring. The stress field must be such that plastic cycling and tensile stress concentrations during loading are minimized at critical locations. This can be achieved by optimizing the geometry of the interface between the stress ring and die insert in such a way that the resulting pressure field at the interface induces the desirable stress field inside the die (Figure 6 b).

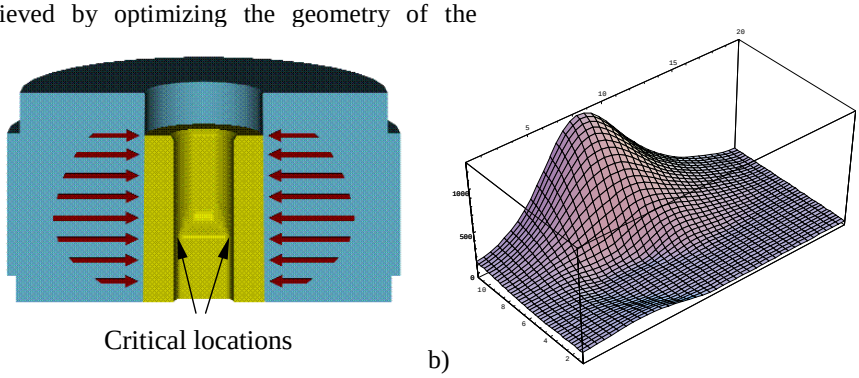


Figure 6: a) Cold forging tools with critical locations where cracks tend to occur. b) Optimal pressure around the interface between the die and stress ring.

4 Optimization of Internal Structure of Inhomogeneous Material

The problem of optimal design of material structure has been addressed. The example includes optimization of shape of inclusions in a periodic microstructure [20],[21] with respect to given criteria concerning the overall response of a specimen under a prescribed loading (Figure 7). A specimen supported at its bottom ends is loaded by a vertical force acting in the middle of its topper surface. The specimen consists of periodic inclusions of a harder material incorporated in a softer matrix. The task is to optimize the shape and orientation of these inclusions by taking into account criteria that incorporate the deflection of the specimen, work of external forces and energy dissipation.

Computation of the response of the structure has been performed by a two scale finite element model with strongly coupled scales [22]. This allows for more accurate computation in the case where the scale effect is expressed, i.e. where the size of the structure would affect the macroscopic response.

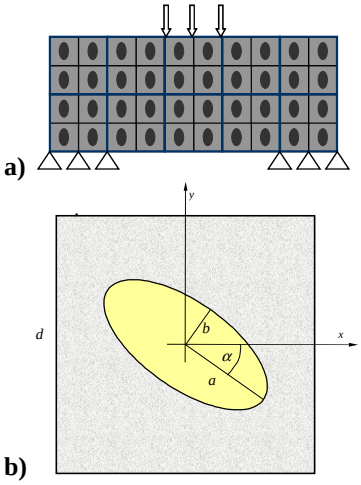
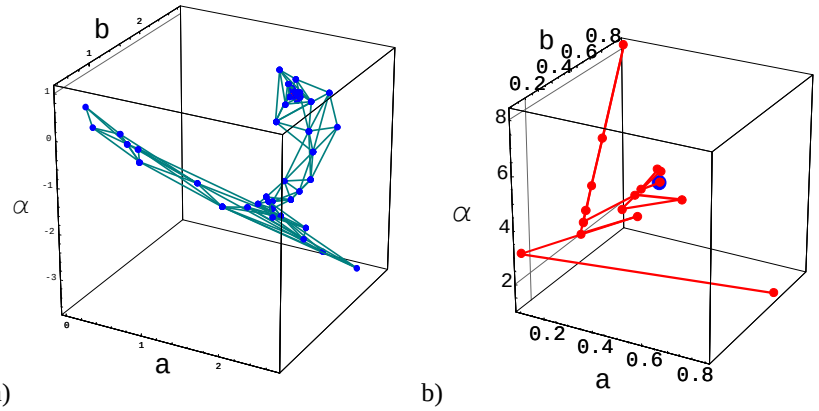


Figure 7: a) Studied structure under loading and b) three parametric description of the shape of inclusions within microscopic periodic cells.

Finite element analysis of the problem has been implemented in FEAP and parallelized on top of the parallel interface adopted in PLATON – a Problem Solving Environment (PSE) for distributed numerical optimisation. The problem was then solved on a cluster of Linux workstations. Optimization was performed by the Nelder-Mead simplex and the sequential quadratic programming method [23] (Figure 8).



a) b)

Figure 8: Convergence of a) simplex and b) SQP algorithm in the parameter space.

5 Algorithm design

In many practical cases, computational complexity of the numerical model and substantial noise in numerical results are primary difficulties that seriously aggravate employment of classical optimization techniques [25]. An algorithm based on successive local approximations of the response functions with restricted step approach has been constructed in order to overcome these difficulties. A rough outline of the algorithm is as follows:

1. Choose the center point $\mathbf{x}_0$ (initial guess) and the trust region parameter r.
2. Sample response functions in a chosen number of points contained in the sampling region $\Omega_{\mathbf{x}_i,r}$ centered around $\mathbf{x}_i$.
3. Build approximations of the objective and constraint functions.
4. Solve the problem with approximated objective and constraint functions, subjected in addition to step length constraints defined by r. Set a new $\mathbf{x}_i$ to the solution of this problem.
5. Update r according to algorithm progress and with respect to agreement of samples in the last iterate with approximations in previous iterates.
6. If not converged, repeat the procedure from step 2, otherwise post-process the collected data and finish with $\mathbf{x}_i$ as problem solution.

The goal is to spend as few function evaluations as possible for converging to the solution of the optimization problem with the requested accuracy, as well as to ensure stability in presence of noise. It is crucial that sampling is performed economically. Usually less samples are calculated in each iteration than the minimum needed for the approximation, and are therefore combined with samples acquired in previous iterations. Different techniques are employed to choose sampling points in a way that ensures maximal accuracy of approximations.

Updating the trust region parameter r is the a subtle part of the algorithm. This parameter determines the constraint on step size as well as the size of the sampling region. In general, reducing r means smaller potential advance in each iteration (because step restriction is tighter), but also better agreement of approximations with sampled functions provided that the sampled functions are smooth enough. Far from the solution, we attempt to keep r large in order to enable faster progress, but not too large because in this case local approximation becomes too inaccurate over the trust region. Near the solution we should allow quick reduction of r over iterations in order to ensure rapid local convergence.

5.1 Optimization library

Further research is focused on more systematic development of efficient optimization algorithms applicable for larger number of design parameters and complex response with substantial level of noise. An optimization library intended for development, testing, comparison and application of such algorithms is being developed. The library will provide a modular collection of building blocks for algorithm development (sampling, approximation building, regularization techniques, linear algebra routines, algorithms

for solving mathematical programming sub-problems, parallel job management, etc.) and extensible set of test cases with performance monitoring facilities.

At the current stage, internal standards for data structures and common function prototypes have been set up in such a way that linkage with external libraries and software is simplified, while other serviceability aspects are kept in mind such as efficiency, thread safety, extensibility, etc. A basic toolbox is provided, on basis of which a framework of the test suite and a prototype algorithm were built. Future plans include definition of interface standards for parallel algorithms, extension of the toolbox and construction of implementations of some basic classes of optimization algorithms, and elaboration of documentation after which a public release of the library will be prepared. This will hopefully create a basis for long term development of efficient algorithms with active exchange of research results and testing procedures with broader community in the relevant research field. The library will also be included in the optimization shell "Inverse". This will enable prompt application of the designed algorithms to practical problems, with possibility of combining with additional tools such as shape parameterization utilities.

Acknowledgment

This work was performed by financial assistance of the European Commission, in the scope of the COST 526 action, and the Slovenian Ministry of education, science and sport under the contract No. Z2-3200-1533-02

References

[1] Gavrus A, Massoni E and Chenot J L. Computer Aided Rheology for Nonlinear Large strain Thermo-viscoplastic Behaviour Formulated as an Inverse Problem. Proc. of Inverse Problems in Engineering Mechanics 1994, Paris.

[2] Rodic T, Gresovnik I and Owen D R J. Application of Error Minimization concept to Estimation of Hardening Parameters in the Tension Test. Proc. of COMPLAS 95, Barcelona, 779-786.

[3] Maniatty A M. and Chen M F. Shape Optimization for Steady Forming Processes. Proc. of COMPLAS 95, Barcelona, 719-730.

[4] Fourment L, Balan T and Chenot J L. Optimal design for non steady-state metal forming processes- II Application of shape optimization in forging. Int. J. Num. Meth. in Engng, vol. 39, n° 1: 51-66.

[5] Wright E and Grandhi R V. Integrated Process and Shape Design in Metal Forming With Finite Element Sensitivity Analysis. Design Optimisation: Int. Journ. of Product & Process Improvement, 1999, Vol.1, No. 1: 55-78.

[6] Optimisation Shell Inverse, electronic document at http://www.c3m.si/inverse/ , maintained by the Centre for Computational Continuum Mechanics, Ljubljana.

[7] Gresovnik I and Rodic T. Optimisation System Utilising a General Purpose Finite Element System. Proc. of WCSMO-2, Zakopane, 1997, Vol. 1, 61-66.

[8] Rodic T and Gresovnik. I. A computer system for solving inverse and optimisation problems. Eng. Comput., 1998, vol. 15, no. 7: 893-907.

[9] Gresovnik I. A General Purpose Computational Shell for Solving Inverse and Optimisation Problems - Applications to Metal Forming Processes. Ph.D. thesis, University of Wales Swansea, 2000. Available on the Internet at http://www.c3m.si/inverse/doc/phd/index.html

[10] I. Gresovnik., T. Sustar, T. Rodic. Parallelization of an optimization shell. In Proc. of The 3rd World congress of structural and multidisciplinary optimization. Buffalo, NY, May 17-21, 1999. pp. 221-223.

[11] Gresovnik I. Quick Introduction to Optimization Shell Inverse. Electronic document at http://www.c3m.si/inverse/doc/other/ .

[12] O. C. Zienkiewicz, R. Taylor. The Finite Element Method, Vol. 1, 2 (fourth edition). McGraw-Hill, London, 1991.

[13] Marsh D. Applied Geometry for Computer Graphics and CAD. Springer, 1999.

[14] Kegl M. Shape optimal design of structures: an efficient shape representation concept. Int. J. Num. Meth Eng., 2000, Vol. 49: 1571-1588.

[15] I. Gresovnik, T. Rodic, D. Jelovsek. Simple two stage transforms designed for optimisation of shape in forming processes. In Proceedings of The 4th ESAFORM Conference on Material Forming, Liege, Belgium, April 23-25, 2001.

[16] I. Gresovnik, T. Rodic, Practical considerations regarding optimisation of shape in forming processes, In Proceedings of The 5th International ESAFORM Conference on Material Forming, Akadeimia Góeniczo-Hutnicza Kraków, 2002.

[17] I. Gresovnik. Spline Interpolation with Sensitivities. Internal report, C3M, 2003.
[18] T. Rodic, D. Cukjati and I. Gresovnik. Optimal design of preform geometry and tribological conditions in can forming. Accepted for publication in Engineering Computations.
[19] S. Stupkiewicz, J. Korelc, M. Dutko, T. Rodic. Shape sensitivity analysis of large deformation frictional contact problems. Computer Methods in Applied Mechanics and Engineering, vol 191/33 pp 3555-3.
[20] A. Ibrahimbegovic, I. Gresovnik, D. Markovic, S. Melnyk and T. Rodic Shape optimization of two-phase material with microstructure. Accepted for publication in Engineering Computations.
[21] I. Gresovnik, D. Markovic, T. Rodic, A. Ibrahimbegovic. Optimization of Inclusion Shape in Inhomogeneous Structural Elements. Proceedings of NATO-ARW advanced workshop on Multi-physics and Multi-scale Computer Models in Non-linear Analysis and Optimal Design of Engineering Structures Under Extreme Conditions, Bled, Slovenia, June 13 - 17, 2004.
[22] A. Ibrahimbegovic and D. Markovic: Strong coupling methods in multi-phase and multi-scale modeling of inelastic behavior of heterogeneous structures. Comput. Methods Appl. Mech. Engrg., Vol. 192, pp. 3089-3170, 2003.
[23] C. T. Lawrence,J. L. Zhou, A. L. Tits. User's Guide for CFSQP Version 2.5. Institute for Systems Research, University of Maryland, Technical Report TR-94-16r1, 1997.
[24] Van Keulen F, Toropov V V. Multipoint Approximations for Structural Optimization Problems with Noisy Response Functions. Electronic document at http://www-tm.wbmt.tudelft.nl/~wbtmavk/issmo/paper/mam_nois2.htm.
[25] I. Gresovnik, S. Hartman and T. Rodic. Development of Optimization Methodology for Reduction of Defect Risk at Blow Forming. Accepted for publication at the 6th World Congress on Structural and Multidisciplinary Optimization, Rio, 2005.
[26] I. Grešovnik, ” Ioptlib home”, electronic document at http://www.c3m.si/igor/ioptlib/