Innovative Computational Methods for Structural Optimization

Manolis Papadrakakis, Yiannis Tsompanakis, Nikos D. Lagaros
National Technical University Athens, Greece

Ernest Hinton & Johann Sienz
University of Wales, UK
and
Georg Thierauf & Jianbo Cai
University of Essen, Germany

Abstract. The objective of this paper is to investigate the efficiency of combinatorial optimization methods and in particular algorithms based on evolution strategies, when incorporated into structural optimization problems. Evolution strategies algorithms are used either on a stand-alone basis, or combined with a conventional mathematical programming technique. Furthermore, the structural analysis phase is replaced by a neural network prediction for the computation of the necessary data for the ES optimization procedure. Advanced domain decomposition techniques is also proposed particularly tailored for parallel solution of large-scale sensitivity analysis problems.

1 Introduction

In structural shape optimization problems the aim is to improve a given topology by minimizing an objective function subjected to certain constraints [1,2]. All functions are related to the design variables which are some of the coordinates of the key points in the boundary of the structure. In a gradient-based mathematical programming approach the shape optimization algorithm proceeds with the following steps: (i) a finite element mesh is generated, (ii) displacements and stresses are evaluated, (iii) sensitivities are computed by perturbing each design variable by a small amount, and (iv) the optimization problem is solved and the new shape of the structure is defined. These steps are repeated until convergence has occurred.

The most time-consuming part of this process is devoted to the sensitivity analysis phase which is an important ingredient of all mathematical programming optimization methods [3]. For this reason several techniques have been developed for the efficient calculation of the sensitivities in an optimization problem. The semi-analytical and the finite difference approaches are the two most widely used types of sensitivity analysis techniques. From the algorithmic point of view the semi-analytical technique results in a typical linear solution problem with multiple right-hand sides in which the stiffness matrix remains the same, while the finite difference technique results in a typical reanalysis problem in which the stiffness matrix is modified due to the perturbations of the design variables. In both shape and sizing optimization
problems 60% to 90% of the computations are spent for the solution of equilibrium equations required for the finite element analysis and sensitivity analysis.

On the other hand the application of combinatorial optimization methods based on probabilistic searching, such as evolution strategies (ESs), do not need gradient information and therefore avoid to perform the computationally expensive sensitivity analysis step. During the last three decades there has been a growing interest in problem solving systems based on algorithms which rely on analogies to natural processes. The best known algorithms in this class include evolutionary programming (EP) [4], genetic algorithms (GAs) [5,6], evolution strategies (ESs) [7,8], simulated annealing [9], classifier systems and neural networks [10]. Evolution-based systems maintain a population of potential solutions. These systems have some selection process based on fitness of individuals and some recombination operators. ESs like GAs imitate biological evolution and combine the concept of artificial survival of the fittest with evolutionary operators to form a robust search mechanism.

Mathematical programming methods, such as Sequential Quadratic Programming (SQP), make use of local curvature information derived from linearization of the original functions by using their derivatives with respect to the design variables at points obtained in the process of optimization to construct an approximate model of the initial problem. These methods present a satisfactory local rate of convergence, but they cannot assure that the global optimum can be found. On the other hand, combinatorial optimization techniques, such as ESs, are in general more robust and present a better global behaviour than the mathematical programming methods. They may suffer, however, from a slow rate of convergence towards the global optimum.

Another important technique that follows natural processes, and in particular human brain functions, is artificial neural networks which simulate the structure of the biological neural network of the human brain. The use of artificial intelligence techniques, such as neural networks, to predict analysis outputs has been studied previously in the context of optimal design of structural systems [32,33], and also in some other areas of structural engineering applications. In the review papers of Berrais [34] and Waszczyszyn [35] a number of references can be found on the application of neural networks (NN) in computational mechanics. The principal advantage of a properly trained NN is that it requires a trivial computational effort to produce an acceptable approximate solution. Such approximations appear to be valuable in situations where the actual response computations are intensive in terms of computing time and a quick estimation is required.

In this work the efficiency of ESs in structural shape optimization problems and the combination of ES with NN in sizing structural optimization problems are investigated. In order to benefit from the advantages of mathematical programming and evolution strategies a combination of SQP and ESs is also examined in an effort to increase the robustness as well as the computational efficiency of the optimization procedure. Furthermore, domain decomposition parallel solution algorithms have been implemented in topology and shape optimization problems. The numerical tests presented demonstrate the computational advantages of the proposed approaches which become more pronounced in large-scale and computationally intensive optimization problems as well as in parallel computing environment.
2 Structural optimization

2.1 Shape optimization

The shape optimization method used in the present study is based on a previous work by Hinton and Sienz [1] for treating two-dimensional problems. It consists of the following essential ingredients: (i) shape generation and control, (ii) mesh generation, (iii) adaptive finite element analysis, (iv) sensitivity analysis, when gradient-based optimization methods are applied, and (v) shape optimization.

Structural optimization problems are characterized by various objective and constraint functions which are generally non-linear functions of the design variables. These functions are usually implicit, discontinuous and non-convex. The mathematical formulation of structural optimization problems with respect to the design variables, the objective and constraint functions depends on the type of the application. However, all optimization problems can be expressed in standard mathematical terms as a non-linear programming problem (NLP) which in general form can be stated as follows:

\[
\min \quad F(s) \\
\text{subject to} \quad h_j(s) \leq 0 \quad j = 1, \ldots, m \\
\quad s_i^l \leq s_i \leq s_i^u \quad i = 1, \ldots, n
\]  

where, \( s \) is the vector of design variables, \( F(s) \) is the objective function to be minimized, \( h_j(s) \) are the behavioural constraints, \( s_i^l \) and \( s_i^u \) are the lower and the upper bounds on a typical design variable \( s_i \). Equality constraints are usually rarely imposed. Whenever they are used they are treated for simplicity as a set of two inequality constraints.

The shape optimization methodology proceeds with the following steps: (i) At the outset of the optimization, the geometry of the structure under investigation has to be defined. The boundaries of the structure are modeled using cubic B-splines which, in turn, are defined by a set of key points. Some of the coordinates of these key points will be the design variables which may or may not be independent to each other. (ii) An automatic mesh generator is used to create a valid and complete finite element model. A finite element analysis, is then carried out and the displacements and stresses are evaluated. In order to increase the accuracy of the analysis an h-type adaptivity analysis may be incorporated in this stage. (iii) If a gradient-based optimizer, like the sequential quadratic programming SQP algorithms, is used then the sensitivities of the constraints and the objective function to small changes of the design variables are computed either with the finite difference, or with the semi-analytical method. (iv) The design variables are being optimized. If the convergence criteria for the optimization algorithm are satisfied, then the optimum solution has been found and the process is terminated, else a new geometry is defined and the whole process is repeated from step (ii).

2.2 Sizing optimization

In sizing optimization problems the aim is usually to minimize the weight of the structure under certain behavioural constraints on stresses and displacements. The
design variables are most frequently chosen to be dimensions of the cross-sectional areas of the members of the structure. Due to engineering practice demands the members are divided into groups having the same design variables. This linking of elements results in a trade-off between the use of more material and the need of symmetry and uniformity of structures due to practical considerations. Furthermore, it has to be to taken into account that due to fabrication limitations the design variables are not continuous but discrete since cross-sections belong to a certain set.

A discrete structural optimization problem can be formulated in the following form:

$$\begin{align*}
\text{min} & \quad F(s) \\
\text{subject to} & \quad g_j(s) \leq 0 \quad j = 1, \ldots, m \\
& \quad s_i \in \mathbb{R}^d, \quad i = 1, \ldots, n
\end{align*}$$

where $\mathbb{R}^d$ is a given set of discrete values and design variables $s_i (i=1,\ldots,n)$ can take values only from this set. In the present study the sizing optimization of large-scale 3-D trusses is investigated. These type of structures is very common in engineering practice in order to cover long and/or wide span and column-free spaces such as stadiums, exhibition halls, airplane hangars, etc. The performance of these type of structures has been investigated in terms of economy, structural safety, aesthetic quality and optimum design in a number of papers [37-41].

The sizing optimization methodology proceeds with the following steps: (i) At the outset of the optimization the geometry, the boundaries and the loads of the structure under investigation have to be defined. (ii) The design variables, which may or may not be independent to each other, are also properly selected. Furthermore, the constraints are also defined in this stage in order to formulate the optimization problem as in eq. (2). (iii) A finite element analysis, is then carried out and the displacements and stresses are evaluated. (iv) If a gradient-based optimizer, like the SQP algorithm, is used then the sensitivities of the constraints and the objective function to small changes of the design variables are computed either with the finite difference, or with the semi-analytical method. (v) The design variables are being optimized. If the convergence criteria for the optimization algorithm are satisfied, then the optimum solution has been found and the process is terminated, else the optimizer updates the design variable values and the whole process is repeated from step (iii).

3 Gradient-based structural optimization

3.1 Sensitivity analysis

Sensitivity analysis is the most important and time-consuming part of a gradient-based optimization procedure. Although, sensitivity analysis is mostly mentioned in the context of structural optimization, it has evolved into a research topic of its own. The calculation of the sensitivity coefficients follows the application of a relatively small perturbation to each primary design variable. Several techniques have been developed which can be mainly distinguished by their numerical efficiency and their implementation aspects [11].
A classification of the discrete methods for sensitivity analysis is the following. (i) *Global finite difference method*: A full finite element analysis has to be performed for each design variable and the accuracy of the method depends strongly on the value of the perturbation of the design variables. (ii) *Semi-analytical method*: The stiffness matrix of the initial finite element solution is retained during the computation of the sensitivities. This provides an improved efficiency over the finite difference method by a relatively small increase in the algorithmic complexity. The accuracy problem involved with the numerical differentiation can be overcome by using the “exact” semi-analytical method which needs more programming effort than the simple method but it is computationally more efficient. (iii) *Analytical method*: The finite element equations, the objective and constraint functions are differentiated analytically.

### 3.1.1 The global finite difference (GFD) method

In this method the design sensitivities for the displacements $\frac{\partial u}{\partial s_k}$ and the stresses $\frac{\partial \sigma}{\partial s_k}$, which are needed for the gradients of the constraints, are computed using a forward difference scheme:

\[
\frac{\partial u}{\partial s_k} \approx \frac{\Delta u}{\Delta s_k} = \frac{u(s_k + \Delta s_k) - u(s_k)}{\Delta s_k} \quad (3)
\]

\[
\frac{\partial \sigma}{\partial s_k} \approx \frac{\Delta \sigma}{\Delta s_k} = \frac{\sigma(s_k + \Delta s_k) - \sigma(s_k)}{\Delta s_k} \quad (4)
\]

The perturbed displacement vector $u(s_k + \Delta s_k)$ of the finite element equations is evaluated by

\[
K(s_k + \Delta s_k)u(s_k + \Delta s_k) = f(s_k + \Delta s_k) \quad (5)
\]

and the perturbed stresses $\sigma(s_k + \Delta s_k)$ are computed from

\[
\sigma(s_k + \Delta s_k) = DB(s_k + \Delta s_k)u(s_k + \Delta s_k) \quad (6)
\]

where $D$ and $B$ are the elasticity and the deformation matrices, respectively. The GFD scheme is usually sensitive to the accuracy of the computed perturbed displacement vectors which is dependent on the magnitude of the perturbation of the design variables. The magnitude of this perturbation is usually taken between $10^{-3}$ and $10^{-5}$.

### 3.1.2 The semi-analytical (SA) method

The SA method is based on the chain rule differentiation of the finite element equations $Ku=f$:

\[
K \frac{\partial u}{\partial s_k} + \frac{\partial K}{\partial s_k}u = \frac{\partial f}{\partial s_k} \quad (7)
\]

which when rearranged results in

\[
K \frac{\partial u}{\partial s_k} = f^*_k \quad (8)
\]

where

\[
f^*_k = \frac{\partial f}{\partial s_k} - \frac{\partial K}{\partial s_k}u \quad (9)
\]
$f^*_k$ represents a pseudo-load vector. The derivatives of $\partial K/\partial s_k$ and $\partial f/\partial s_k$ are computed for each design variable by recalculating the new values of $K(s_k + \Delta s_k)$ and $f(s_k + \Delta s_k)$ for a small perturbation $\Delta s_k$ of the design variable $s_k$. The derivatives of $\partial f/\partial s_k$ are computed using a forward finite difference scheme.

With respect to the differentiation of $K$ the semi-analytical approach can be divided in two methods: The conventional SA and the "exact" SA. In the conventional sensitivity analysis (CSA), the values of the derivatives in (7) are calculated by applying the forward difference approximation scheme

$$\frac{\partial K}{\partial s_k} \approx \frac{K(s_k + \Delta s_k) - K(s_k)}{\Delta s_k}$$

(10)

In the "exact" semi-analytical method (ESA) [12] the derivatives $\partial K/\partial s_k$ are computed on the element level as follows

$$\frac{\partial k}{\partial s_k} = \sum_{j=1}^{n} \frac{\partial k_j}{\partial s_k}$$

(11)

where $n$ is the number of elemental nodal coordinates affected by the perturbation of the design variable $s_k$ and $\alpha_j$ are the nodal coordinates of the element. The ESA method is more accurate and leads the mathematical optimizer to a faster convergence [13]. This approach is used in the present study.

Stress gradients can be calculated by differentiating $\sigma = DBu$ as follows

$$\frac{\partial \sigma}{\partial s_k} = \frac{\partial D}{\partial s_k} Bu + D \frac{\partial B}{\partial s_k} u + DB \frac{\partial u}{\partial s_k}$$

(12)

Since the elasticity matrix $D$ is not a function of the design variables then (12) reduces to

$$\frac{\partial \sigma}{\partial s_k} = D \frac{\partial B}{\partial s_k} u + DB \frac{\partial u}{\partial s_k}$$

(13)

In (13), $\partial u/\partial s_k$ may be computed as indicated in (2), while the term $\partial B/\partial s_k$ is computed using a forward finite difference scheme. Using the values of $\partial \sigma/\partial s_k$ the sensitivities of different types of stresses (e.g. the principal stresses or the equivalent stresses) can be readily calculated by analytically differentiating their expressions with respect to the shape variables.

3.2 Mathematical programming optimization algorithms

Sequential Quadratic Programming (SQP) methods are the standard general purpose mathematical programming algorithms for solving Non-Linear Programming (NLP) optimization problems [14]. They are also considered to be the most suitable methods for solving structural optimization problems [15-17]. Such methods make use of local curvature information derived from linearization of the original functions, by using their derivatives with respect to the design variables at points obtained in the process of optimization. Thus, a Quadratic Programming (QP) model (or subproblem) is constructed from the initial NLP problem. A local minimizer is found by solving a sequence of these QP subproblems using a quadratic approximation of the objective function. Each subproblem has the form:
minimize \[ \frac{1}{2} p^T H p + g^T p \]
subject to \[ A p + h(s) \leq 0 \]
where \( p \) is the search direction subjected to upper and lower bounds, \( g \) is the gradient of the objective function, \( A \) is the Jacobian of the constraints, usually the active ones only (i.e. those that are either violated, or not far from being violated), \( \bar{s}_i = s_i - s \), \( \bar{s}_u = s_u - s \) and \( H \) is an approximation of the Hessian matrix of the Lagrangian function
\[ L(s, \lambda) = F(s) + \lambda h(s) \]
in which \( \lambda \) are the Lagrange multipliers under the non-negativity restriction (\( \lambda \geq 0 \)) for the inequality constraints. In order to construct the Jacobian and the Hessian matrices of the QP subproblem the derivatives of the objective and constraint functions are required. These derivatives are computed during the sensitivity analysis phase.

There are two ways to solve this QP subproblem, either with a primal [18], or a dual [19] formulation. In the present study a primal algorithm is employed based on an SQP algorithm from the NAG library [20]. The primal algorithm is divided into three phases: (i) the solution of the QP subproblem to obtain the search direction, (ii) the line search along the search direction \( p \), (iii) the update of the Hessian matrix \( H \). Once the direction vector \( p \) is found a line search is performed, involving only the nonlinear constraints, in order to produce a “sufficient decrease” to the merit function \( \varphi \). This merit function is an augmented Lagrangian function of the form [19]
\[ \varphi = F(s) + \sum_i \lambda_i (g_i(s) - \gamma_i) + \rho_i (g_i(s) - \gamma_i)^2 \]
where \( \gamma_i \) are the non-negative slack variables of the inequality constraints derived from the solution of the QP subproblem. These slack variables allow the active inequality constraints to be treated as equalities and avoid possible discontinuities. Finally, \( \rho_i \) are the penalty parameters which are initially set to zero and in subsequent iterations are increased whenever this is necessary in order to control the violation of the constraints and to ensure that merit function follows a descent path.

Finally a BFGS quasi-Newton update [14] of the approximate Hessian of the Lagrangian function \( L \) is implemented. In order to incorporate the new curvature information obtained through the last optimization step, the updated Hessian \( \tilde{H} \) is defined as a rank-two modification of \( H \):
\[ \tilde{H} = H - \frac{1}{w^T H w} H w w^T H + \frac{1}{y^T y} y y^T \]
where \( w \) and \( y \) denote the change in the design variable vector \( s \) and the gradient vector of the Lagrangian function in (15), respectively. If the quadratic function is convex then the Hessian is positive definite, or positive semi-definite and the solution obtained will be a global optimum, else if the quadratic function is non-convex then the Hessian is indefinite and if a solution exists it is only a local optimum.
4 Solution methods

4.1 Domain decomposition solution methods

In computational mechanics there are basically three domain decomposition formulations combined with the preconditioned conjugate gradient (PCG) method for solving linear finite element problems in parallel computing environments. The first approach is the global subdomain implementation (GSI) in which a subdomain-by-subdomain PCG algorithm is implemented on the global stiffness matrix. In the second approach the PCG algorithm is applied on the interface problem after eliminating the internal degrees of freedom. This Schur complement scheme is called the primal subdomain implementation (PSI) on the interface to distinguish from the third approach which is called the dual subdomain implementation (DSI) on the interface [44].

The DSI approach operates on totally disconnected subdomains, while the governing equilibrium equations are derived by invoking stationarity of the energy functional subject to displacement constraints which enforce the compatibility conditions on the subdomain interface. The augmented equations are solved for the Lagrange multipliers after eliminating the unknown displacements. The resulting interface problem is in general indefinite, due to the presence of floating subdomains which do not have enough prescribed displacements to eliminate the local rigid body modes. The solution of the indefinite problem is performed by a preconditioned conjugate projected gradient (PCPG) algorithm.

4.2 Solution methods in sensitivity analysis problems

The implementation of hybrid solution schemes in structural optimization, which are based on a combination of direct and preconditioned iterative solvers, has not yet received the attention it deserves from the research community, even though in finite element linear solution problems and particularly when dealing with large-scale applications their efficiency is well documented. In a recent study by Papadrakakis et al.[3] a class of efficient hybrid solution methods was applied in the context of shape sensitivity analysis and topology optimization problems in sequential computing environment.

In this work domain decomposition methods are applied for solving the sensitivity analysis part of shape optimization problems, in both sequential and parallel computing environments, after being properly modified to address the special features of the particular type of problems at hand. The most computational intensive part of sensitivity analysis, using either semi-analytical or finite difference sensitivity analysis approaches, is the solution of finite element equilibrium equations (5) or (3), respectively. In the first case the coefficient matrix remains constant and only the right-hand side vector is changing, which is the typical case for solving linear systems with multiple right-hand sides, while in the second case the coefficient matrix is slightly modified resulting in a typical reanalysis problem to be solved.

4.2.1 Solving sensitivity analysis problems with the semi-analytical approach

One of the main shortcomings of iterative solution methods is encountered when a sequence of right-hand sides has to be processed. In such cases direct methods
possess a clear advantage over iterative methods since the most computationally intensive part, associated with the factorization of the stiffness matrices, is not repeated and only a backward and forward substitution is required for each subsequent right-hand side. The Lanczos method has been used in the past for treating a sequence of right-hand sides. An efficient implementation of Lanczos method was proposed by Papadrakakis and Smerou [28] which handles all approximations to the solution vectors simultaneously without the necessity for storing the tridiagonal matrix and the orthonormal basis. Recently a reorthogonalization procedure has been proposed by Farhat et al.[26] for extending the PCG method to problems with multiple and/or repeated right-hand sides based on the K-conjugate property of the search directions \( \mathbf{d}_m = \mathbf{d}_m^T \mathbf{K} \mathbf{d}_i = 0 \) for \( m < i \).

The implementation of the reorthogonalization technique is impractical when applied to the full problem \( \mathbf{K} \mathbf{u}^{(i)} = \mathbf{f}^{(i)} \) due to excessive storage requirements. However, this methodology has been efficiently combined with the domain decomposition FETI method [29] where the size of the interface problem can be order(s) of magnitude less than the size of the global problem. Thus, the cost of reorthogonalization is negligible compared to the cost of the solution of the local problems associated with the matrix-vector products of the FETI method, while the additional memory requirements are not excessive. The modified search direction of the PCG algorithm is given by [29]

\[
d'_{m+1} = d_{m+1} - \sum_{i=1}^{m} \frac{d_i^T \mathbf{F}_i d_{m+1}}{d_i^T \mathbf{F}_i d_i} d_i
\]

(18)

which enforces explicitly the orthogonality condition \( d'_{m+1} \mathbf{F}_i d_i = 0 \), \( i=1,\ldots,m \).

The initial estimate \( \lambda_0^{(i+1)} \) of the solution vector of the subsequent right-hand side of eq. (15) is given by [29]

\[
\lambda_0^{(i+1)} = \mathbf{D}_k^T \mathbf{x} + x'
\]

(19)

where \( \mathbf{D}_k^T \mathbf{F}_k \mathbf{D}_k \mathbf{x} = \mathbf{D}_k^T (\mathbf{f}_k^{(i+1)} - \mathbf{F}_k \mathbf{x}') \) and \( \mathbf{x}' = \mathbf{G}_1 (\mathbf{G}_1^T \mathbf{G}_1)^{-1} \mathbf{f}_1^{(i+1)} \).

4.2.2 Solving reanalysis problems with a two-level domain decomposition method

The hybrid solution schemes proposed in Ref. [3] for treating reanalysis type of problems, based on the global formulation and solution of the problem of eq. (5), proved to be very efficient compared with the standard skyline solver in sequential computing environment. Their parallel implementation, however, is hindered by the inherent scalability difficulties encountered during the preconditioning step which incorporates forward and backward substitutions of a fully factorized stiffness matrix. In order to alleviate this deficiency the GSI subdomain-by-subdomain PCG algorithm (GSI-PCG) is implemented in this study on the global stiffness matrix. The dominant matrix-vector operations of the stiffness and the preconditioning matrices are performed in parallel on the basis of a multi-element group partitioning of the entire domain.
In order to exploit the parallelizable features of the GSI-PCG method and to take advantage of the efficiency of a fully factorized preconditioning matrix, the following two-level methodology is proposed based on the combination of the GSI and the DSI approaches. The GSI-PCG method is employed, using a multi-element group partitioning of the entire finite element domain, in which the solution required during the preconditioning step is performed by the FETI method operating on the same mesh partitioning of the GSI-PCG method. In the proposed methodology the preconditioning step of the GSI-PCG method

\[ z_{m+1} = C_k^{-1}r_{m+1} \]  

(20)

is performed by the FETI solution procedure. For the solution of this problem two methodologies, namely the GSI(PCG)-FETI and the GSI(NCG)-FETI are proposed. The second approach is based on a Neumann series expansion of the preconditioning step.

The GSI(PCG)-FETI method

In the GSI(PCG)-FETI method the iterations are performed on the global level with the GSI-PCG method, using an incomplete Cholesky factorization of the stiffness matrix as preconditioner. Thus, the incomplete factorization of the stiffness matrix \( K_0 + \Delta K \) can be written as \( LDL^T = K_0 + \Delta K - E \), where \( E \) is an error matrix which does not have to be formed. Matrix \( E \) is usually defined by the computed positions of “small” elements in \( L \) which do not satisfy a specified magnitude criterion and therefore are discarded [27]. For the typical reanalysis problem

\[ (K_0 + \Delta K)u = f \]  

(21)

matrix \( E \) is taken as \( \Delta K \), so that the preconditioning matrix becomes the complete factorized initial stiffness matrix \( C_k = K_0 \). Therefore, the solution of the preconditioning step of the GSI-PCG algorithm, which has to be performed at each GSI-PCG iteration, can be effortlessly executed, once \( K_0 \) is factorized, by a forward and backward substitution.

With the parallel implementation of the two-level GSI(PCG)-FETI method the preconditioning step can be solved in parallel by the interface FETI method for treating the repeated solutions required in eq. (20), using the same decomposition of the domain employed by the external GSI-PCG method. The procedure continues this way for every reanalysis problem, while the FETI direction vectors are being reorthogonalized in order to further decrease the number of FETI iterations within the preconditioning step. The solution of eq. (20) is performed \( n_i \times n_r \) times via the FETI method, where \( n_i \) and \( n_r \) correspond to the number of GSI-PCG iterations and the number of reanalysis steps, respectively.

The GSI(NCG)-FETI method

The combination of Neumann series expansion and PCG method on the global level for the solution of reanalysis problems in shape and topology optimization was investigated in a previous study [3]. In this work the Neumann series expansion is used to improve the quality of the preconditioning step of the two-level method by computing a better approximation to the inverse preconditioning matrix. The
preconditioning matrix is now defined as the complete stiffness matrix \((K_0 + \Delta K)\), but the solution for \(z_{m+1}\) of eq. (20) is performed approximately using a truncated Neumann series expansion.

Thus, the preconditioned vector \(z_{m+1}\) of eq. (20) is obtained at each iteration by

\[
z_{m+1} = (I + K_0^{-1}K)z_m + r_m = (I + K_0^{-1}K)z_m + r_m = (I + K_0^{-1}K)z_m + r_m
\]

(22)

where the term in parenthesis can be expressed in a Neumann expansion giving

\[
z_{m+1} = (I - P + P^2 - P^3 + \ldots)K_0^{-1}r_m
\]

(23)

with \(P=K_0^{-1}\Delta K\). The preconditioned residual vector of eq. (23) can now be represented by the following series

\[
z_{m+1} = z'_0 - z'_1 + z'_2 - z'_3 + \cdots
\]

(24)

with

\[
z'_0 = K_0^{-1}r_m
\]

(25)

\[
z'_i = K_0^{-1}(\Delta Kz'_{i-1}), \quad i = 1, 2, \ldots
\]

(26)

The incorporation of the Neumann series expansion in the preconditioned step of the PCG algorithm can be seen from two different perspectives. From the PCG point of view, an improvement of the quality of the preconditioning matrix is achieved by computing a better approximation to the solution of \(u = (K_0 + \Delta K)^{-1}f\) during the preconditioning step, than the one provided by the preconditioning matrix \(K_0^{-1}\). From the Neumann series expansion point of view, the inaccuracy entailed by the truncated series is alleviated by the conjugate gradient iterative procedure.

5 Evolution Strategies (ES)

Evolution strategies were proposed for parameter optimization problems in the seventies by Rechenberg [7] and Schwefel [8]. ES imitate biological evolution in nature and have three characteristics that make them differ from other conventional optimization algorithms: (i) in place of the usual deterministic operators, they use randomized operators: mutation, selection as well as recombination; (ii) instead of a single design point, they work simultaneously with a population of design points in the space of variables; (iii) they can handle continuous, discrete or mixed optimization problems. The second characteristic allows for a natural implementation of ES on parallel computing environments. The ES, however, achieve a high rate of convergence than GA due to their self-adaptation search mechanism and are considered more efficient for solving real world problems [21]. The ES algorithms used in the present study are based on the work of Thierauf and Cai who applied the ES methodologies in sizing structural optimization problems having discrete and/or continuous design variables [22,23].

5.1 ESs Algorithms

The ESs were initially applied for continuous optimization problems, but recently they have also been implemented in discrete and mixed optimization problems
The ESs can be divided into a two-membered evolution strategy (2-ESs) or a multi-membered evolution strategy (M-ESs).

### 5.1.1 The two-member ESs

The earliest evolution strategies were based on a population consisting of one individual only. The two membered scheme is the minimal concept for an imitation of organic evolution. The two principles of mutation and selection, which Darwin in 1859 recognized to be most important, are taken as rules for variation of the parameters and for recursion of the iteration sequence respectively.

The two-membered ESs for the solution of the optimization problem works in two steps:

**Step 1 (mutation).** The parent \( s_p^{(g)} \) of the generation \( g \) produces an offspring \( s_o^{(g)} \), whose genotype is slightly different from that of the parent:

\[
\begin{align*}
  s_o^{(g)} &= s_p^{(g)} + z^{(g)} \\
  \text{where } z^{(g)} &= [z_1^{(g)}, z_2^{(g)}, \ldots, z_n^{(g)}]^T
\end{align*}
\]

**Step 2 (selection).** The selection chooses the best individual between the parent and the offspring to survive:

\[
\begin{align*}
  s_p^{(g+1)} &= \begin{cases} 
    s_o^{(g)}, & \text{if } g_i(s_o^{(g)}) \leq 0 \quad \text{for } i = 1,2,\ldots,l \quad \text{and } f(s_o^{(g)}) \leq f(s_p^{(g)}) \\
    s_p^{(g)}, & \text{otherwise}
  \end{cases}
\end{align*}
\]

The question how to choose the random vector \( z^{(g)} \) in **Step 1** is very important. This choice has the role of mutation. Mutation is understood to be random, purposeless events, which occur very rarely. If one interprets them, as is done here, as a sum of many individual events, it is natural choice to use a probability distribution according to which small changes occur frequently, but large ones only rarely. Two requirements arise together by analogy with natural evolution: (i) the expected mean value \( \xi_i \) for a component \( z_i^{(g)} \) to be zero; (ii) the variance \( \sigma_i^2 \), the average squared deviation from mean value, is small.

The probability density function for normally distributed random events is given by

\[
p(z_i^{(g)}) = \frac{1}{\sqrt{2\pi\sigma_i}} \exp\left(-\frac{(z_i^{(g)} - \xi_i)^2}{2\sigma_i^2}\right)
\]

When \( \xi_i=0 \) the so-called \((0, \sigma_i)\) normal distribution is obtained. By analogy with other deterministic search strategies, \( \sigma_i \) can be called step length, in the sense that it represents average values of the length of the random steps. If the step length is too small the search takes an unnecessarily large number of iterations. On the other hand, if the step length is too large the optimum can only be crudely approached and the search can even get stuck far away from the global optimum. Thus, as in all optimization strategies, the step length control is the most important part of the algorithm after the recursion formula, and it is further more closely linked to the convergence behaviour.
The standard deviation $\sigma_i$ can be adjusted during the search as follows (Rechenberg's 1/5 success rule [7]): "The ratio of successful mutations to all mutations should be 1/5. If it is greater, increase; if it is less, decrease the standard deviations $\sigma_i$." According to Schwefel [8], the check should take place every $n$ mutations over the preceding $10n$ mutations, while the increase and decrease factors of the step length should be $(1/0.85)$ and $0.85$, respectively. During the search, not only the design variables $s_i$, but also the parameters, such as the deviations $\sigma_i$, will be modified by the random operator mutation which replaces the 1/5 success rule.

5.1.2 Multi-membered ESs

The multi-membered evolution strategies differ from the previous two-membered strategies in the size of the population. In this case a population of $\mu$ parents will produce $\lambda$ offsprings. Thus the two steps are defined as follows:

Step 1 (recombination and mutation). The population of $\mu$ parents at $g$-th generation produces $\lambda$ offsprings. The genotype of any descendant differs only slightly from that of its parents.

Step 2 (selection). There are two different types of the multi-membered ESs:

- $(\mu+\lambda)$-ESs: The best $\mu$ individuals are selected from a temporary population of $(\mu+\lambda)$ individuals to form the parents of the next generation.
- $(\mu,\lambda)$-ESs: The $\mu$ individuals produces $\lambda$ offsprings ($\mu<\lambda$) and the selection process defines a new population of $\mu$ individuals from the set of $\lambda$ offsprings only.

In the second type, the existence of each individual is limited to one generation. This allows the $(\mu,\lambda)$-ESs selection to perform better on problems with an optimum moving over time, or on problems where the objective function is noisy.

In Step 1, for every offspring vector a temporary parent vector

$$\tilde{s} = [\tilde{s}_1, \tilde{s}_2, ..., \tilde{s}_n]^T$$

is first built by means of recombination. For continuous problem the following recombination cases can be used:

$$\tilde{s}_i = \begin{cases} 
    s_{\alpha,i} \text{ or } s_{b,i} \text{ randomly} & \text{(A)} \\
    1/2 (s_{\alpha,i} + s_{b,i}) & \text{(B)} \\
    s_{bj,i} & \text{(C)} \\
    s_{\alpha,i} \text{ or } s_{bj,i} \text{ randomly} & \text{(D)} \\
    1/2 (s_{\alpha,i} + s_{bj,i}) & \text{(E)} 
\end{cases}$$

(30)

where $s_1$ is the i-th component of the temporary parent vector $\tilde{s}$, $s_{\alpha,i}$ and $s_{b,i}$ are the i-th components of the vectors $s_{\alpha}$ and $s_{b}$ which are two parent vectors randomly chosen from the population. In case C of (20), $s_{bj,i} = s_{bj,i}$ means that the i-th component of $s$ is chosen randomly from the i-th components of all $\mu$ parent vectors. From the temporary parent $\tilde{s}$ an offspring can be created in the same way as in two-membered ESs (24).

Multi-membered ESs termination criteria are the following: (i) when the absolute or relative difference between the best and the worst objective function values is less than a given value $\varepsilon_1$, or when (ii) the mean value of the objective
values from all parent vectors in the last 2*n generations has not been improved by
less than a given value $\varepsilon_2$.

5.2 ES in structural optimization problems

Usually this type of problems are solved with a mathematical programming algorithm
such as the sequential quadratic programming method SQP [17], the generalized
reduced gradient method (GrG) [24], the method of moving asymptotes (MMA)
[25], which need gradient information. In structural optimization problems, where the
objective function and the constraints are particularly highly non-linear functions of
the design variables, the computational effort spent in gradient calculations is usually
large.

In two recent studies by Papadrakakis et. al. [30,31] it was found that
probabilistic search algorithms are computationally efficient even if greater number of
analyses are needed to reach the optimum. These analyses are computationally less
expensive than in the case of mathematical programming algorithms since they do not
need gradient information. Furthermore, probabilistic methodologies were found, due
to their random search, to be more robust in finding the global optimum, whereas
mathematical programming algorithms may be trapped in local optima. Finally, the
natural parallelism inherent in probabilistic search algorithms makes them very
attractive for application in parallel computer architectures.

The ES optimization procedure starts with a set of parent vectors and if any
of these parent vectors gives an infeasible design then this parent vector is modified
until it becomes feasible. Subsequently, the offsprings are generated and checked if
they are in the feasible region. According to ($\mu$$+$$\lambda$) selection scheme in every
generation the values of the objective function of the parent and the offspring vectors
are compared and the worst vectors are rejected, while the remaining ones are
considered to be the parent vectors of the new generation. On the other hand,
according to ($\mu$,$\lambda$) selection scheme only the offspring vectors of each generation are
used to produce the new generation. This procedure is repeated until the chosen
termination criterion is satisfied.

The computational efficiency of the multi-membered ES discussed in this
work is affected by the number of parents and offsprings involved. It has been
observed that values of $\mu$ and $\lambda$ equal to the number of the design variables produce
best results [30]. The ES algorithm for structural optimization applications can be
stated as follows:

1. Selection step :
   selection of $s_i$ ($i = 1,2,...,\mu$) parent vectors of the design variables
2. Analysis step : solve $K(s_i)u_i = f$ ($i=1,2,...,\mu$)
3. Constraints check : all parent vectors become feasible
4. Offspring generation :
   generate $s_j$, ($j=1,2,...,\lambda$) offspring vectors of the design variables
5. Analysis step : solve $K(s_j)u_j = f$ ($j=1,2,...,\lambda$)
6. Constraints check :
   if satisfied continue, else change $s_j$ and go to step 4
7. **Selection step:**

   selection of the next generation parents according to \((\mu+\lambda)\) or \((\mu,\lambda)\) selection schemes

8. **Convergence check:** If satisfied stop, else go to step 3

So far little effort has been spent in applying probabilistic search methods in shape optimization problems which are usually solved with a mathematical programming algorithms. The use of combinatorial type algorithms appears to be promising even if greater number of analyses are needed to reach the optimum. This is due to the fact that since the number of design variables in shape optimization problems is relatively small the number of analyses is limited to few tens or hundreds. Moreover, these analyses are less computationally expensive than in the case of mathematical programming algorithms as they do not need gradient information. Furthermore, the same advantages stated in the previous section are also valid for the probabilistic methodologies in shape optimization problems.

5.3 **ES for discrete optimization problems**

In engineering practice the design variables are not continuous because usually the structural parts are constructed with certain variation of their dimensions. Thus design variables can only take values from a predefined discrete set. For the solution of discrete optimization problems a modified ES algorithm has been proposed by Thierauf and Cai [22]. The basic differences between discrete and continuous ES are focused on the mutation and the recombination operators. In the discrete version of ES the random vector \(z^{(g)}\) is properly generated in order to force the offspring vector to move to another set of discrete values.

The fact that the difference between any two adjacent values can be relatively large is against the requirement that the variance \(\sigma_i^2\) should be small. For this reason it is suggested that not all the components of a parent vector, but only a few of them (eg. \(\ell\)) should be randomly changed in every generation. This means that \(n-\ell\) components of the randomly changed vector \(z^{(g)}\) will have zero value. In other words, the terms of vector \(z^{(g)}\) are derived from

\[
  z_i^{(g)} = \begin{cases} 
    (\kappa + 1)\delta s_i & \text{for } \ell \text{ randomly chosen components} \\
    0 & \text{for } n - \ell \text{ other components}
  \end{cases} \tag{31}
\]

where \(\delta s_i\) is the difference between two adjacent values in the discrete set and \(\kappa\) is a random integer number which follows the Poisson distribution

\[
p(\kappa) = \frac{\gamma^\kappa}{\kappa!} e^{-\gamma} \tag{32}
\]

\(\gamma\) is the standard deviation as well as the mean value of the random number \(\kappa\). The choice of \(\ell\) depends on the size of the problem and it is usually taken as the \(1/5\) of the total number of design variables. The \(\ell\) components are selected using uniform random distribution in every generation according to eq. (31).

For discrete optimization the procedure terminates when one of the following termination criteria is satisfied: (i) when the best value of the objective function in the last \(4n\mu/\lambda\) generations remains unchanged, (ii) when the mean value of the
objective values from all parent vectors in the last $2*n*\mu/\lambda$ generations has not been improved by less than a given value $\varepsilon_b (=0.0001)$, (iii) when the relative difference between the best objective function value and the mean value of the objective function values from all parent vectors in the current generation is less than a given value $\varepsilon_c (=0.0001)$, (iv) when the ratio $\mu_b/\mu$ has reached a given value $\varepsilon_d (=0.5$ to $0.8)$ where $\mu_b$ is the number of the parent vectors in the current generation with the best objective function value.

5.4 The hybrid approach

The main advantage of the SQP optimizer is that it captures very fast the right path to the nearest optimum, irrespective if it is a local or a global optimum. After locating the area of this optimum it might oscillate until all constraints are satisfied. Even small constraint violations often slow down the convergence rate of the method. On the other hand ESs are not so sensitive as SQP to small constraint violations but proceed with slower rate, due to their random search, and usually need a greater number of analyses. However, these analyses are very fast since they don’t require expensive gradient calculations. Furthermore, the absence of strict mathematical rules, which govern the convergence rate of the mathematical programming methods, make ESs less vulnerable to local optima and therefore much more reliable to obtain the global optimum in non-convex optimization methods.

In order to benefit from the advantages of both methodologies a hybrid approach is proposed, which combines the two methods in an effort to increase the robustness and the computational efficiency of the optimization procedure. Two combinations of SQP and ESs methodologies are implemented: (i) In the first approach the SQP method is used first, giving a design very close to the optimum, followed by ESs in order to accelerate convergence and avoid the oscillations of SQP due to small constraint violations around optimum. The transition from one algorithm to the other is performed when

$$\frac{|f_{j+1} - f_j|}{f_j} \leq \varepsilon$$

(33)

where $\varepsilon$ is taken $0.01$. This approach appears to be more suitable when the design space is convex, i.e. there is a unique optimum irrespective of the starting design. (ii) In the second approach the sequence of the methods in reversed. An ESs procedure is used first in order to locate the region where the global optimum lies, then the SQP is activated in order to exploit its higher order of accuracy in the neighborhood of the optimum. In this case the switch is performed when there is a small difference ($\varepsilon=0.1$) between the best designs of two consecutive generations. This approach appears to be more rational in the general case when more complex and non-convex design problems are to be solved with many local optima.

6 Basic principles of artificial neural networks theory

The aim of the present study is to train a neural network to provide computationally inexpensive estimates of analysis outputs required during the optimization process. A trained network presents some distinct advantages over the numerical computing
paradigm. It provides a rapid mapping of a given input into the desired output quantities, thereby enhancing the efficiency of the redesign process. This major advantage of a trained NN over the conventional procedure, under the provision that the predicted results fall within acceptable tolerances, leads to results that can be produced in a few clock cycles, representing orders of magnitude less computational effort than the conventional computational process. The learning algorithm which was employed for the training is the well known Back Propagation (BP) algorithm [36,42].

In the present implementation the objective is to investigate the ability of the NN to predict accurate structural analysis outputs that are necessary for the ES optimizer. This is achieved with a proper training of the NN. The NN training comprises the following tasks: (i) select the proper training set, (ii) find a suitable network architecture and (iii) determine the appropriate values of characteristic parameters such as the learning rate and momentum term.

An important factor governing the success of the learning procedure of a NN architecture is the selection of the training set. A sufficient number of input data properly distributed in the design space together with the output data resulting from complete structural analyses are needed for the BP algorithm in order to provide satisfactory results. Overloading the network with unnecessary similar information results to overtraining without increasing the accuracy of the predictions. A few tens of structural analyses have been found sufficient for the examples considered to produce a satisfactory training of the NN. Ninety percent of those runs is used for training and the rest is used to test the results of the NN.

Most researchers split the design space into subregions and try to combine randomly the values within each subregion in order to obtain a training set which is representative of the whole design space. This procedure leads frequently to a huge number of training patterns in order to ensure that the whole design space is properly represented. In an effort to increase the robustness as well as the computational efficiency of the NN procedure various types of training set selection were investigated in a previous study [31]. In this study two types of training set selection are used: (i) the training set is chosen automatically based on a Gaussian distribution of the design variables around the midpoints of the design space, (ii) the training set is chosen using data from the structural analyses carried out in a number of ES optimization steps until the design reaches stationarity near the optimum. This happens when the value of objective function remains unchanged for a number of ES generations.

The first type of the training set selection was motivated from the fact that usually the searching for the optimum and its location lies in the region near the midpoints of the design space. A Gaussian distribution was therefore used for the random selection of input data in order to cover the whole design space and enforce the selection of most input patterns around the midpoints of the design space. This approach proved to be more efficient than choosing randomly combinations of input data from the whole range of the design space using a uniform distribution of the design variables [31]. The second type of the training set selection is based on the fact that in most cases the ES optimizer very fast tracks the path to the optimum and then it may oscillate around it until convergence is achieved at a slower rate. Therefore it is more efficient to produce the training sets in the vicinity of the design
point where the optimizer has reached a stationary point. This way a smaller number of training sets is required and the NN training is performed much faster and accurately.

7 Structural optimization based on ES and NN

After the selection of the suitable NN architecture the training procedure is performed using a number (M) of data sets, selected as described previously, in order to obtain the I/O pairs needed for the NN training. Since the NN based structural analysis can only provide approximate results it is recommended that a correction on the output values should be performed in order to alleviate any inaccuracies entailed, especially when the constraint value is near the limit which separates the feasible and the infeasible region. This is achieved with a relaxation of this limit during the NN testing phase before entering the optimization procedure. A “correction” of the allowable constraint values was therefore performed proportional to the maximum testing error of the NN configuration. The maximum testing error is the largest average error of the output values among testing patterns. Whenever the predicted values were found smaller than those derived from a conventional structural analysis the allowable values of the constraints were decreased according to the maximum testing error of the NN configuration and vice versa.

The proposed ES-NN methodology can be described with the following algorithms according to the two types of training set selection schemes that were previously described:

7.1 Algorithm 1

The combined ES-NN optimization procedure is performed in two phases. The first phase includes the training set selection, the structural analyses required to obtain the necessary I/O data for the NN training, and finally the selection, training and testing of a suitable NN configuration. The second phase is the ES optimization stage where the trained NN is used to predict the response of the structure in terms of objective and constraints function values, due to different sets of design variables, instead of the standard structural analysis computations. The proposed methodology ES-NN can be described with the following algorithm:

- NN training phase:
  1. Training set selection step: select \( s_i \) (i = 1,2,...,M) input patterns.
  2. Structural analysis step: solve \( K(s_i)u_i = f \) (i=1,...,M).
  3. Training step: selection and training of a suitable NN architecture.

- ES-NN optimization phase:
  1. Selection step: selection of \( s_i \) (i = 1,2,...,\( \mu \)) parent vectors of the design variables.
  2. Prediction step: using NN to compute optimization function values for the \( \mu \) parent vectors.
3. **Constraints check**: all parent vectors become feasible.
4. **Offspring generation**:
   
   generate $s_j$, ($j=1,2,...,\lambda$) offspring vectors of the design variables.
5. **Prediction step**: using NN to compute optimization function values for the $\lambda$ offspring vectors.
6. **Constraints check**:
   
   if satisfied continue, else change $s_j$ and go to step 4.
7. **Selection step**:
   
   selection of the next generation parents according to ($\mu+\lambda$) or ($\mu,\lambda$) selection schemes.
8. **Convergence check**: If satisfied stop, else go to step 3.

### 7.2 Algorithm 2

According to the second type of training set selection the proposed ES-NN methodology can be described with the following algorithm: The combined ES-NN optimization procedure is performed in three phases. The first phase is the ES optimization stage until a stationary point is obtained. This is the case when the mean value of the objective values from all parent vectors in the last $n*\mu/\lambda$ generations has not been improved by less than a given value $\varepsilon_d$ (=0.05). The second phase includes the training set selection in the vicinity of the stationary point from the previous structural analyses during previous ES steps. This way the necessary I/O data required for the NN training are obtained, and finally the selection, training and testing of a suitable NN configuration. The third phase is identical to the second phase of algorithm 1. The second algorithm is described as follows:

- **ES optimization phase**:
  1. **Selection step**:
     
     selection of $s_i$ ($i = 1,2,...,\mu$) parent vectors of the design variables
  2. **Analysis step**: solve $K(s_i)u_i = f$ ($i=1,2,...,\mu$)
  3. **Constraints check**: all parent vectors become feasible
  4. **Offspring generation**:
     
     generate $s_j$, ($j=1,2,...,\lambda$) offspring vectors of the design variables
  5. **Analysis step**: solve $K(s_j)u_j = f$ ($j=1,2,...,\lambda$)
  6. **Constraints check**:
     
     if satisfied continue, else change $s_j$ and go to step 4
  7. **Selection step**:
     
     selection of the next generation parents according to ($\mu+\lambda$) or ($\mu,\lambda$) selection schemes
  8. **Stationarity check**: If satisfied continue, else go to step 3

- **NN training phase**:
  1. **Training set selection step**: choose $s_i$ ($i = 1,2,...,M$) I/O data.
  2. **Training step**: selection and training of a suitable NN architecture.
  3. **Testing step**: test NN and “correct” allowable constraint values.
8 Numerical examples

8.1 Shape optimization test problems
The performance of the optimization methods discussed is investigated in one characteristic plane stress test example with isotropic material properties (elastic modulus \( E=210,000 \text{ N/mm}^2 \) and Poisson's ratio \( \nu=0.3 \)). The SQP method used for the mathematical programming based optimization is taken from the NAG library [20]. The problem definition of this example is given in Figure 1 where, due to symmetry, only a quarter of the plate is modeled. The plate is under biaxial tension with one side loaded with a distributed loading \( p=0.65 \text{ N/mm}^2 \) and the other side loaded only with half of this value, as shown in Figure 1. The objective is to minimize the volume of the structure subject to an equivalent stress limit of \( \sigma_{\text{max}}=7.0 \text{ N/mm}^2 \).

The design model consists of 8 key points and 5 primary design variables (2, 3, 4, 5, 6) which can move along radial lines. The movement directions are indicated by the dashed arrows. The stress constraints are imposed as a global constraint for all the Gauss points and as key point constraints for the key points 2, 3, 4, 5, 6 and 8. The problem is analyzed with a fine mesh of 38,800 d.o.f. giving a sparse global stiffness matrix with relatively large bandwidth. The characteristic d.o.f. for 4 and 8 subdomains, as depicted in Figure 2a and 2b, are given in Table 1. The ESA and the GFD methods are used to compute the sensitivities with \( \Delta s=10^{-5} \). All tests were performed on a Silicon Graphics Power Challenge computer with R4000 processors.

![Figure 1 Square plate: (a) initial shape; (b) final shape](image)

<table>
<thead>
<tr>
<th>Subdomains</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total d.o.f.</td>
<td>38,800</td>
<td>38,800</td>
</tr>
<tr>
<td>Internal d.o.f.*</td>
<td>9,738</td>
<td>5,122</td>
</tr>
<tr>
<td>Interface d.o.f.</td>
<td>998</td>
<td>2,290</td>
</tr>
</tbody>
</table>

*of the larger subdomain

Table 1. Square plate: Characteristic d.o.f. for 4 and 8 subdomains
The performance of the solution methods presented in section 4 is investigated first in serial computing mode with the conventional direct skyline and PCG, NCG and Lanczos solvers proposed in Ref. [3]. NCG is a PCG algorithm in which the preconditioning step is performed via a Neumann series expansion of the stiffness matrix. Furthermore, the parallel performance of FETI and its variants FETI1 and FETI2 is investigated in both types of sensitivity analysis problems, while the two-level PCG method is applied for the GFD sensitivity analysis test cases. In FETI1 the rigid body modes are computed explicitly and are not obtained as a by-product of the factorization procedure as in standard FETI, while in FETI2 the rigid body modes are also computed explicitly and the local problem is solved via the PCG algorithm with preconditioner the complete factorized stiffness matrix stored in single precision arithmetic. The convergence tolerance for all solution methods was taken as $10^{-3}$.

The following abbreviations are used: Direct is the conventional skyline direct solver; $PCG(\psi)$ and $Lanczos(\psi)$ are the PCG and Lanczos solvers respectively, with the preconditioning matrix produced via a complete, or an incomplete Cholesky factorization controlled by the rejection parameter $\psi$ [27]. A value of $\psi$ between 0 and 1 correspond to an incomplete Cholesky preconditioner, while $\psi=0$ gives the complete factorized matrix. $NCG-i$ is the NCG solver with $i$ terms of the Neumann series expansion. The same abbreviations previously used for the one-level dual domain decompositions method are adopted for the shape optimization examples.

<table>
<thead>
<tr>
<th>method</th>
<th>time (s)</th>
<th>storage (Mbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct skyline</td>
<td>502</td>
<td>95</td>
</tr>
<tr>
<td>Lanczos (0)</td>
<td>524</td>
<td>63</td>
</tr>
<tr>
<td>Lanczos (1E-9)</td>
<td>417</td>
<td>29</td>
</tr>
<tr>
<td>PCG (0)</td>
<td>514</td>
<td>61</td>
</tr>
<tr>
<td>PCG (1E-9)</td>
<td>425</td>
<td>27</td>
</tr>
<tr>
<td>FETI-reorth</td>
<td>486</td>
<td>43</td>
</tr>
<tr>
<td>FETI1-reorth</td>
<td>466</td>
<td>43</td>
</tr>
<tr>
<td>FETI2-reorth</td>
<td>414</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 2. Square plate: Performance of the methods in sequential mode in ESA sensitivity analysis
Finally, the variants of the two-level methodology, combined with the FETI method, namely the GSI(PCG)-FETI and GSI(NCG)-FETI methods, are compared with the other solvers, both in serial and parallel computing modes, in the case of GFD sensitivity analysis problems.

Table 2 demonstrates the performance of the methods operated in a sequential mode for the case of ESA sensitivity analysis. In the standard FETI method and its variants the operations are carried out in 4 subdomains. Table 3 demonstrates the performance of FETI and FETI2, for the case of ESA sensitivity analysis, operated on parallel computing mode in 4 and 8 processors using 4 and 8 subdomains, respectively. The benefits from the use of the reorthogonalization are also evident both in terms of FETI iterations and computing time. Tables 4 and 5 depict the performance of the methods, for the case of GFD sensitivity analysis, operated on sequential and parallel computing modes, respectively. In Tables 3 and 5 the iteration history is also depicted for six right-hand sides, which correspond to the initial finite element solution and the sensitivity analysis for the five design variables of the problem.
<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
<th>Storage (Mbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct skyline</td>
<td>2,790</td>
<td>95</td>
</tr>
<tr>
<td>Lanczos (0)</td>
<td>732</td>
<td>65</td>
</tr>
<tr>
<td>Lanczos (1E-9)</td>
<td>745</td>
<td>61</td>
</tr>
<tr>
<td>PCG (0)</td>
<td>714</td>
<td>27</td>
</tr>
<tr>
<td>PCG (1E-9)</td>
<td>2,108</td>
<td>43</td>
</tr>
<tr>
<td>FETI-reorth</td>
<td>1,782</td>
<td>26</td>
</tr>
<tr>
<td>FETI1-reorth</td>
<td>795</td>
<td>27</td>
</tr>
<tr>
<td>FETI2-reorth</td>
<td>779</td>
<td>29</td>
</tr>
</tbody>
</table>

Table 4. Square plate: Performance of the methods in parallel mode in ESA sensitivity analysis

<table>
<thead>
<tr>
<th>Method</th>
<th>Iterations</th>
<th>Time (s)</th>
<th>Storage (Mbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FETI-reorth</td>
<td>33</td>
<td>667</td>
<td>43</td>
</tr>
<tr>
<td>FETI2-reorth</td>
<td>33</td>
<td>574</td>
<td>26</td>
</tr>
<tr>
<td>GSI(PCG)-FETI2-r</td>
<td>33</td>
<td>256</td>
<td>27</td>
</tr>
<tr>
<td>GSI(NCG)-FETI2</td>
<td>33</td>
<td>249</td>
<td>29</td>
</tr>
</tbody>
</table>

Table 5. Square plate: Performance of the methods in parallel mode in GFD sensitivity analysis

Table 6 depicts the performance of mathematical programming (MP) and evolution strategies (ES) optimization methods in sequential and parallel computing modes for two test cases of this example corresponding to two initial designs, one close and the other away from the optimum. In Table 6 the following abbreviations are used: MP corresponds to the Mathematical Programming-SQP method. ESA and GFD refer to exact semi-analytical and global finite difference methods of sensitivity analysis, respectively.

<table>
<thead>
<tr>
<th>Optimization Method</th>
<th>Number of Opt. Steps</th>
<th>Optimum Volume (mm³)</th>
<th>Sequential time (s)</th>
<th>Parallel time in p processors (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOD INITIAL DESIGN (V₀=307.3 mm³) - 1528 d.o.f.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MP-ESA</td>
<td>33</td>
<td>280</td>
<td>965.8</td>
<td>702.9 (p=5)</td>
</tr>
<tr>
<td>MP-GFD</td>
<td>33</td>
<td>280</td>
<td>1785.6</td>
<td>873.5 (p=5)</td>
</tr>
<tr>
<td>ES (3+3)</td>
<td>134</td>
<td>279</td>
<td>2336.8</td>
<td>1217.2 (p=3)</td>
</tr>
<tr>
<td>ES (5+5)</td>
<td>85</td>
<td>279</td>
<td>1271.5</td>
<td>354.2 (p=5)</td>
</tr>
<tr>
<td>ES (10+10)</td>
<td>150</td>
<td>279</td>
<td>2780.7</td>
<td>337.4 (p=10)</td>
</tr>
<tr>
<td>BAD INITIAL DESIGN (V₀=373.4 mm³) - 1546 d.o.f.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MP-ESA</td>
<td>72</td>
<td>280</td>
<td>1894.7</td>
<td>1198.6 (p=5)</td>
</tr>
<tr>
<td>MP-GFD</td>
<td>75</td>
<td>279</td>
<td>4189.1</td>
<td>1673.4 (p=5)</td>
</tr>
<tr>
<td>ES (3+3)</td>
<td>134</td>
<td>279</td>
<td>2406.8</td>
<td>1219.7 (p=3)</td>
</tr>
<tr>
<td>ES (5+5)</td>
<td>127</td>
<td>279</td>
<td>2141.3</td>
<td>586.8 (p=5)</td>
</tr>
<tr>
<td>ES (10+10)</td>
<td>191</td>
<td>279</td>
<td>2998.5</td>
<td>497.2 (p=10)</td>
</tr>
<tr>
<td>MP-ES</td>
<td>24 (4+20)</td>
<td>281</td>
<td>386.1</td>
<td>196.1 (p=5)</td>
</tr>
<tr>
<td>ES-MP</td>
<td>28 (10+8)</td>
<td>280</td>
<td>480.1</td>
<td>275.9 (p=5)</td>
</tr>
</tbody>
</table>

Table 6. Square plate example - Test case 1 (5 design variables) : Performance of the optimization methods
ES-(µ+λ) refers to the number of parents and offspring vectors, µ, λ respectively, for the evolution strategies approach. MP-ES, ES-MP are the two hybrid approaches defining the sequence of the two optimizers, while the number of optimization steps for each optimizer is depicted in parenthesis. Finally, for the parallel implementation of the optimizers the number of processors used for the case of ES optimizer is equal to the number of design variables p=n, whereas for the case of ESs is p=µ. ESs manage to find the optimum solution in both test cases of the square plate example, whereas the MP approach failed in the second test case. This could be explained by the fact that the MP optimizer was trapped into the infeasible region due to its inability to overcome severe violations mainly on the global stress constraint. The computing time spent by the optimizers is affected by the initial design, particularly in the case of the MP approach. It can also be observed that ESs present a competitive performance to MP optimizer in sequential computing mode and perform much better in parallel computing mode. The use of hybrid approaches, especially the MP-ES, leads to a significant reduction of computing time in both sequential and parallel modes for the first test case. For the second test case, however, since SQP fails both hybrid approaches fail. The mathematical programming optimizer using the ESA sensitivity analysis method is faster than GFD method in sequential mode. In parallel mode, however, GFD becomes competitive to ESA. The natural parallelization scheme implemented has a beneficial effect to all versions of the optimizers used. In particular, this effect is more pronounced in the case of ESs where a higher efficiency is achieved.

**Figure 3 Double-layered space roof truss**

8.2 Sizing optimization test problem

The performance of the sizing optimization methodology discussed in previous sections is investigated for a characteristic three dimensional double-layered space roof truss depicted in Figure 3 with discrete design variables. The NN method used in this study is the back propagation algorithm META-NETS [43]. In the Tables 7-10, containing the results of the test examples, the following abbreviations are used: ES refers to the standard evolution strategies optimization procedure, in which structural analyses are performed in the conventional way. ES-NN refers to the combined NN
and ES optimization procedure, where the structural analysis response is predicted by a trained NN. For the two different types of training set selection that have been compared in this study the following abbreviations are used: (i) GT stands for the random selection of training set based on a Gaussian distribution of the design variables in the design space according to Algorithm 1, (ii) AT stands for the “automatic” training set selection using the results obtained at the initial stages of the ES optimization procedure according to Algorithm 2. The symbol “(c)” is used when the allowable limits of the constraints have been adjusted, as discussed previously, in order to “correct” the NN predictions near the feasible region limits, while symbol “(v)” indicates that the final design is violating the constraints and thus it is infeasible.

In order to investigate the influence of the curvature in the optimum design. Four different topologies were tested corresponding to 0°, 5°, 10°, 15° inclination of the curved surface at the supports. The modulus of elasticity is 200 GPa (29,000 ksi) and the yield stress is $F_y=250$ MPa (36 ksi). Each member is assumed to have a thin-walled tubular cross section. The cross-sectional area is considered to be the design variable of each member. Members are divided to forty eight groups according to their position. For all test cases the finite model consists of 8,000 members, 2,071 nodes and 6,183 degrees of freedom. The loads are taken as uniform vertical forces applied at joints equivalent to uniform load of 0.10 kN and a concentrated vertical load 50 kN at the center of the structure which corresponds to the maximum load of a crane and it is equally distributed to the central nodes of the roof. The objective function in all test cases is the weight of the structure. The constraints are imposed on the maximum nodal displacement and the maximum axial and buckling stresses in each element group. The values of allowable axial stress is $\sigma_a=150$ MPa, whereas the maximum allowable displacement is limited to 3 cm.

### Table 7 - Test case 1 (0° inclination): Performance of the optimization methods

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Number of FE analyses</th>
<th>Number of NN analyses</th>
<th>Computing time (s)</th>
<th>Optimum volume (mm$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Analysis</td>
<td>Training</td>
</tr>
<tr>
<td>ES</td>
<td>4,150</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NN-ES-GT</td>
<td>480</td>
<td>3,827</td>
<td>10,771</td>
<td>1,546</td>
</tr>
<tr>
<td>NN-ES-GTc</td>
<td>480</td>
<td>4,094</td>
<td>10,771</td>
<td>1,546</td>
</tr>
<tr>
<td>NN-ES-A</td>
<td>472</td>
<td>3,174</td>
<td>10,591</td>
<td>345</td>
</tr>
<tr>
<td>NN-ES-Ac</td>
<td>472</td>
<td>3,515</td>
<td>10,591</td>
<td>345</td>
</tr>
</tbody>
</table>

### Table 8 - Test case 2 (5° inclination): Performance of the optimization methods

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Number of FE analyses</th>
<th>Number of NN analyses</th>
<th>Computing time (s)</th>
<th>Optimum volume (mm$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Analysis</td>
<td>Training</td>
</tr>
<tr>
<td>ES</td>
<td>3,940</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NN-ES-GT</td>
<td>480</td>
<td>4,017</td>
<td>10,771</td>
<td>1,493</td>
</tr>
<tr>
<td>NN-ES-GTc</td>
<td>480</td>
<td>3,862</td>
<td>10,771</td>
<td>1,493</td>
</tr>
<tr>
<td>NN-ES-A</td>
<td>512</td>
<td>3,116</td>
<td>11,489</td>
<td>361</td>
</tr>
<tr>
<td>NN-ES-Ac</td>
<td>512</td>
<td>3,406</td>
<td>11,489</td>
<td>361</td>
</tr>
</tbody>
</table>
### Table 9 - Test case 3 (10° inclination): Performance of the optimization methods

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Number of FE analyses</th>
<th>Number of NN analyses</th>
<th>Computing time (s)</th>
<th>Optimum volume (mm³)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Analysis</td>
<td>Training</td>
</tr>
<tr>
<td>ES</td>
<td>4,210</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NN-ES-GT</td>
<td>480</td>
<td>4,132</td>
<td>10,771</td>
<td>1,575</td>
</tr>
<tr>
<td>NN-ES-GTc</td>
<td>480</td>
<td>4,256</td>
<td>10,771</td>
<td>1,575</td>
</tr>
<tr>
<td>NN-ES-A</td>
<td>423</td>
<td>3213</td>
<td>9,492</td>
<td>337</td>
</tr>
<tr>
<td>NN-ES-Ac</td>
<td>423</td>
<td>3147</td>
<td>9,492</td>
<td>337</td>
</tr>
</tbody>
</table>

### Table 10 - Test case 4 (15° inclination): Performance of the optimization methods

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Number of FE analyses</th>
<th>Number of NN analyses</th>
<th>Computing time (s)</th>
<th>Optimum volume (mm³)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Analysis</td>
<td>Training</td>
</tr>
<tr>
<td>ES</td>
<td>4,280</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NN-ES-GT</td>
<td>480</td>
<td>4,021</td>
<td>10,771</td>
<td>1,631</td>
</tr>
<tr>
<td>NN-ES-GTc</td>
<td>480</td>
<td>4,267</td>
<td>10,771</td>
<td>1,631</td>
</tr>
<tr>
<td>NN-ES-A</td>
<td>437</td>
<td>3613</td>
<td>9,806</td>
<td>356</td>
</tr>
<tr>
<td>NN-ES-Ac</td>
<td>437</td>
<td>3221</td>
<td>9,806</td>
<td>356</td>
</tr>
</tbody>
</table>

For all test cases the \((\mu+\lambda)\)-ES approach is used with \(\mu=\lambda=20\). The number of NN input units is equal to the number of design variables, whereas the output units are ninety eight corresponding to the two values of axial and buckling stresses for the forty eight element groups, the value of the maximum nodal displacement and the value of the objective function. The NN configuration has two hidden layers each one consisting of 35 nodes, which results in a 48-35-35-98 NN architecture used for all runs. The performance of the Gaussian and the “automatic” NN training set selection with 480 and 200 training sets, respectively, is shown in Tables 7-10 for the four configurations of the roof. It is obvious from the results that the performance of the proposed ES-NN methodology is superior to the performance of the conventional ES optimization procedure, since a dramatic improvement in total computing time required by ES-NN over ES is observed in all test cases examined. A significant improvement is also observed in the performance, both in terms of computing time and optimum values of the objective function, of the proposed ES-NN methodology when the “automatic” type of NN training is used over the Gaussian type of NN training. As it can be observed from the results obtained the curved type of structure is more economical from the flat roof type even though the surface of the structure is longer. For greater slopes, however, the overall weight grows since the surface of the structure increases significantly.

**Acknowledgments**

This work was supported by HC&M/9203390 project of the EU.
References