MULTILEVEL DISTRIBUTED STRUCTURE OPTIMIZATION

J.O. Entzinger*, R. Spallino , W. Ruijter***
*** University of Twente, Mechanical Engineering Department, ** AIRBUS Deutschland GmbH**

Keywords: *Genetic Algorithms, Neural Networks, Optimisation, Composite Structures, Aerospace Structures*

Abstract

An iterative optimisation routine for aircraft structures using Genetic Algorithms (GAs) and Neural Networks (NNs) is presented. In this setup the NNs form a response surface, approximating the key mechanical properties of substructures. NNs are updated every iteration. The GA uses these NNs in the optimisation to quickly determine the feasibility of different variants. All found optimal substructures are checked using a Finite Element (FE) calculation. When the FE outputs differ too much from the NN approximations the solution is added to the NN training set, thus improving the NN's performance.

Main advantages of the algorithm are firstly the possibility to take into account many topologically distinct designs and secondly the flexibility to quickly evaluate the influence of updated loads or different design restrictions (e.g. materials, access holes) on the optimum. The benefit of the feedback of inaccurately estimated panel properties (according to the FE verification) is the improvement of accuracy and convergence. Also this principle drastically reduces the number of datasets (i.e. FE calculations) needed to train the NNs initially.

Two levels are implemented: a global level containing the structure as a whole, and a local level to describe the composite panels the structure is made of more accurately. On the global level a coarse mesh can be used, for it is only needed to derive the loading of the panels. On the local level more detail is needed, for linear static buckling and local strains must be analysed accurately.

On the local level all panel optimisations (including the time-consuming FE checks) are mutually independent and can therefore be parallelised. This speeds up the process significantly when using multiple computers to distribute the workload. In our setup 7-27 machines were used, enabling overnight optimisation of a large structure.

Full scale optimisations have been successfully performed on a major substructure of a representative aircraft design under static loadcases.

1 Introduction

For this research project a combination of Genetic Algorithms (GAs) and Neural Networks (NNs) has been applied to iteratively optimise an aircraft structure by optimising the (more or less similar) panels it consists of.

The GAs are chosen for their robustness and flexibility. Because they don't need any gradient information, GAs allow a mixture of continuous and discrete parameters so different topologies and even different design principles (plate or trusswork) can be handled. The drawback of this algorithm is the number of evaluations needed, which urges for a response surface such as a NN to approximate the outputs of Finite Element (FE) analyses in a fast and accurate way.

When a NN with fixed architecture and training sets is trained with a grid of FE solutions, the errors of the NN estimates appeared quite large. Therefore, a setup with feedback of optima found by the GA has been implemented, so accurate FE values are added to the NN's training set when NN's the accuracy is insufficient. This means less

initial datasets can be used and the NN is only trained to a higher accuracy in the places where the GA expects an optimum. The iterative character of the routine, which optimises all panels and then re-derives the panel loadings from the structure based on the 'optimised' design, clearly shows the benefi t of a (locally) accurate response surface, for the panels to be evaluated are very alike.

The proposed setup is more broadly usable, in fact, it is well usable to handle problems which have the following properties:

- The structure can be divided into smaller, approximately independent components.
- Parametric models are available for both the structure and its components.
- Load paths (or component loads) are to some extent dependent of component designs.
- Optimisation constraints are applied at the component level.
- Multiple topologies may be allowed for a component.

2 Structural Mechanics Problem

Two mechanics problems have to be analysed in the optimisation process:

1. A global level structure analysis with relatively coarse meshes, used to obtain loadings for components
2. A local level component analysis with fi ner meshes, used to obtain the buckling multiplier and the maximum local strain level

All analyses are done automatically using parametric models, for wich linear static FE calculations are performed.

The assembly of an aircraft's vertical tail plane (VTP) structure out of skins ribs and spars is shown in fi gure 1.

A parameterised panel is shown in fi gure 2. The panels are made of composite material and

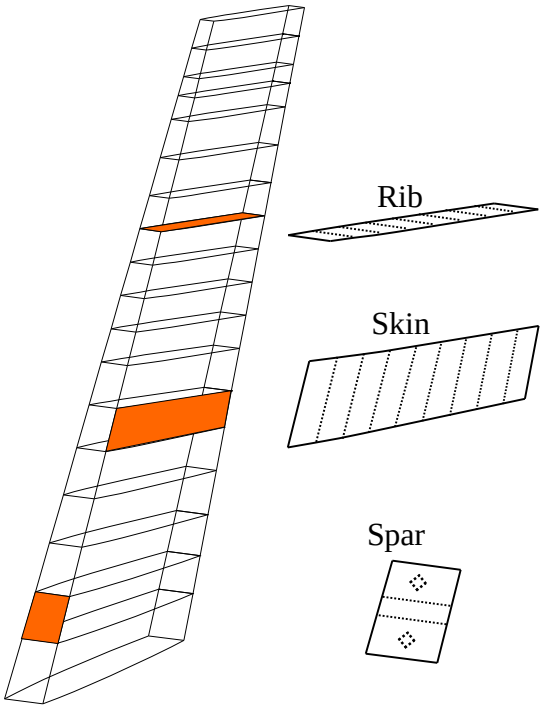


Fig. 1 VTP structure assembly of ribs skins and spars

analysed using layered shell elements. Typical design parameters are length, width, material properties, topology, stiffener/hole positions, stiffener heights and panel thickness. Topologies are distinct in the number of transverse stiffeners, the number of longitudinal stiffeners, the number of holes and their ordering on the panel. Length and width become height and width respectively in the global structure.

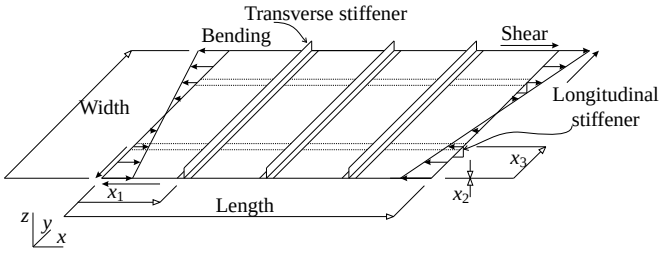


Fig. 2 Parameterised panel

3 Neural Networks

A (feedforward) neural network can be seen as a curve-fi tting technique based on an analogy

with the (human) brain. A set of known inputs and outputs (the training set) is used to set the coefficients in the NN. This process, called training, is a form of non-linear least squares fitting. A reason to choose NNs as a response surface is the fact that they can map virtually any function without any a priori knowledge of the relationship between inputs and outputs. When sigmoid functions are used as transfer functions, the NN's ability to generalize (i.e. accurately map function samples outside the training dataset) from a small amount of inputs is much larger than conventional techniques like fitting with polynomials or fourier-series. Another difference is that sigmoid functions are capable of handling discontinuities in the output, but it is suspected that the quality of the fit is better with 'sleek' functions.

Most function mapping artificial NNs are of the Feed-Forward (FF) type and have neurons grouped in layers. Each neuron has a transfer function (e.g. a sigmoid function) operating on the neuron's input. In FF-NNs only neurons in successive layers are interconnected. This results in the typical shape shown in figure 3 with one input layer (receiving an input vector **x** typically representing design variables), one output layer (producing output vector **y** which is an approximation of the mechanical response of a component, the exact value of which are targets vector **t**) and an arbitrary number of hidden layers. As the lines in the figure point out, the response of a neuron transfer function depends on weighted inputs (by weights matrices **w**) and a bias value (by bias matrix **b**) which acts as an offset to the neuron's input. The NNs employed have so-called back-propagation training algorithms, this means that the contribution of each connection weight is determined by passing errors in a backward manner through the network. The errors are then minimised by employing an iterative scheme such as Levenberg-Marquardt iteration.

The topic of neurocomputing (i.e. the computer simulation of neural networks) has been

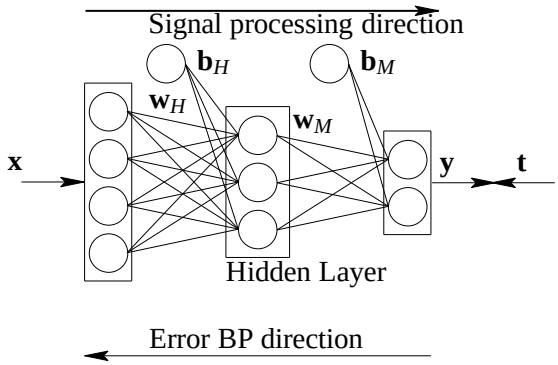


Fig. 3 Typical shape of an artificial neural network

extensively analysed in literature, the reader is referred to [1] and [2] for more specific information on the background and operation of Neural Networks (NN) in engineering applications.

4 Genetic Algorithms

Genetic algorithms are guided random search techniques that work analogous to biological evolution. Much of the GA terminology is derived from the biological counterpart, e.g. there are individuals, populations, genes and fitnesses. An individual represents an element in the solution space. The individual's genes determine its location (i.e. specify all parameter values) and its fitness indicates the performance of the individual with respect to the objective and the constraints. Individuals are grouped in populations.

Assuming that a recombination of genetic material from well performing individuals creates favourable offspring, a GA uses selection and variation to reach a solution to an optimisation problem. Selection is applied by assigning a higher probability to the genetic material of favourable individuals of getting passed on to the next generation. Variation is accomplished by means of crossover and mutation, i.e. recombination of genetic material from selected individuals, respectively random changes in genes.

The algorithm starts by creating an initial population consisting of individuals with randomly assigned genes. The fitness of each individual is determined using an appropriate evaluation function, which is a value depend-

ing on the objective (weight in this project) and penalties (which are given depending on constraint violation). Selection, crossover and mutation are applied consecutively to generate a new population (the next generation). This process is repeated until convergence is reached, that is, until a certain solution dominates the population.

A GA can handle more general classes of functions than most traditional mathematical programming search techniques. Whereas the latter use characteristics of the problem (e.g. gradients and continuity), GAs don't require such assumptions. They can handle also non-differentiable and discontinuous functions. Because of the use of mutation and the fact that GAs operate on a population instead of a single starting point, GAs are less likely to get stuck in a local optimum. However, increased randomness comes with decreased convergence speed, therefore GAs typically require more function evaluations than gradient based optimisers. This is the reason for using a response surface such as NNs instead of direct FE calculations.

A genetic algorithm has been proven to be an effective way of solving a wide range of of non-differentiable problems. The reader is referred to [3] for an application in composite structures, and [4] for a civil engineering application. The GA operators used for this paper are derived from a Matlab implemented GA by Houck et. al. [5].

5 Optimisation Strategy

5.1 Local level

Every panel in the structure is optimised for minimal structural weight using the routine as displayed in fi gure 4. The NNs are used in the evaluation function for estimating the buckling load and maximum local strain for a panel proposed by the GA. Each topology is represented by one NN.

When the buckling load is lower than the applied load, or when the local strain exceeds its maximum allowed value, the solution is penalised. This way the panel's geometry is less likely to persist in the population, so the algorithm tends to panels that don't violate the constraints. On termination of the GA loop the best panel found is checked with a FE analysis, to make sure the NN's estimate is accurate. If this is not the case, the NN is retrained with the information just obtained from the FE check. The NN aims to achieve a certain error level (a user defi ned threshold), if this value is not reached within a given number of training cycles the number of neurons in the NN hidden layer is increased. After all panels have been checked, the optimisation is started again with improved NNs, or returns to the structure analysis when all found optimal panels appeared accurately estimated by the NNs.

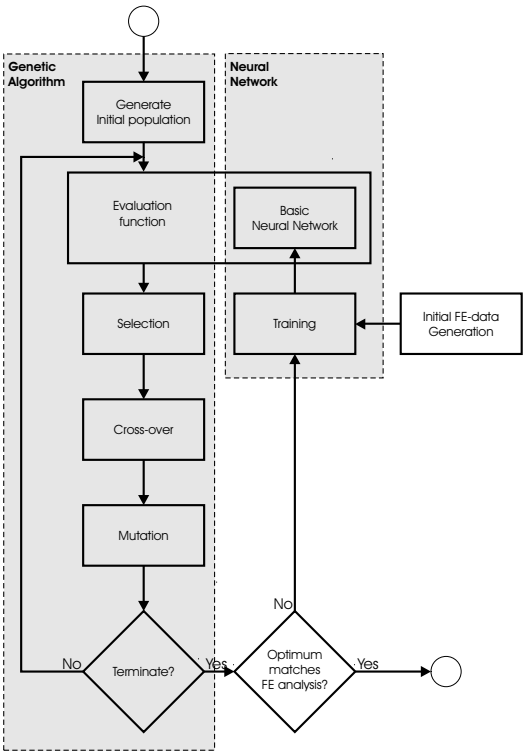


Fig. 4 Component level layout

5.2 Additional rules

To avoid repetition of equal FE analyses in subsequent feedback iterations, the following rules are applied:

- The topology of the optimal panel solution

is excluded in the next feedback iteration, forcing the optimiser to search in different regions of the design space. This improves the accuracy of the set of NNs.

- Once a NN for a certain topology is retrained, all panels for which this topology was excluded re-add it to their search space.

- Once a feasible solution (judging from FE) has been found, the next solution must be lighter. This way, the optimiser can find optima that are approximated too conservative by the NN. Example: Normally a lighter panel which doesn't satisfy the constraints (according to the NN approximation of its properties) gets a penalty, so it is found unfavourable. However, when all solutions which are not lighter than the optimum also get a penalty, the lighter panel might still be found as optimum. When this optimum is checked by a FE analysis, it might appear feasible (i.e. not violating any constraints) and badly approximated by the NN.

5.3 Global level

No optimisation takes place on the global structural level. On this level all components (panels) are assembled and the loading of each component is derived from the global model for a number of different loadcases. A representative load is calculated for each panel. After loads for each component have been obtained from a structure analysis, the components are optimised using the iterative procedure of optimising, checking and retraining the NNs as described in §5.1. The newly found 'optimal' panels are put into the global structural model to obtain the new loading conditions for all panels. The optimisation is done again, but with updated loads as long as the loads obtained from the global structure change.

5.4 Distributed computing implementation

Since all panel optimisations are independent, the local level process (§5.1) can be parallelised to a large extent using multiple computers, thus speeding up the overall optimisation process. This means that the GA panel optimisation, but more importantly, the time-consuming FE checks of optima and NN (re)training cycles are distributed among several computers in a LAN. The aforementioned procedures are implemented in a distributed manner as shown in the following pseudo code routine:

```
main(){
    if(no datasets loaded){
        tasklist(make initial NN datasets);
    }
    while(structure loadpaths keep changing AND
        it1<max structure it){
        task(Structure analysis);
        while(components not accurate
            AND it2<max component it){
            tasklist(Optimize components);
            tasklist(Check optima w. FE);
            tasklist(Retrain NNs);
            it2++;
        }
        it1++;
    }
}
```

In this routine the `task()` and `tasklist()` commands indicate that a queue of tasks is formed of which all the members are sent to the slave machines to be executed.

6 Optimisations

The results of three optimisation runs are presented. Results are obtained using a Java implementation of the proposed distributed computing optimisation algorithm written at the University of Twente.

6.1 Setups

1. A (large) series of components only (spar panels) using the additional rules in §5.2. PATRAN/NASTRAN analyses are used on 7 HP-UX machines @400MHz.

2. A structure run (as introduced in §5.3) using simple skin optimisation (thickness only) for 18 skins. All 9 ribs and 18 spars are fully optimised (parameters are stated in §2). ANSYS analyses are used on 20 windows machines @2.6GHz.

3. As 2. but for 36 skins, 18 ribs and 36 spars, using 27 windows machines @2.6GHz.

6.2 Optimisation constraints

Linear buckling and local strain constraints have to be satisfied for all panels. Extra constraints for the respective setups are:

1. In all spars except the 6 lower ones (front and rear) access holes are required.
2. Topologies 9b and 19-21 as shown in figure 5 are not taken into account.
3. In all spars except the 4 upper ones (front and rear) access holes are required and no longitudinal stiffeners are allowed. Topologies 9b and 19-21 as shown in figure 5 are not taken into account.

It must be mentioned that the structure optimisations (setup 2 and 3) a non-realistic global loadcases and constraints and are used for testing and illustrative purposes only. The components only run is an early actual optimisation run.

Loading:

- Spars: combined shear and bending
- Ribs: shear only
- Skins: combined compression and shear
- Structure: global torsion and bending loadcases

Topologies used for spar panels are depicted in figure 5. Panels are rotated 90 degrees compared to the global structure. For ribs only topologies with transverse stiffeners are taken into account (rib topologies differ in no. of stiffeners only).

New topologies can easily be added. Also topologies using trusswork instead of composite panels could be implemented, however this is beyond the scope of the current research.

7 Results

Some typical features of the optimisation results are highlighted.

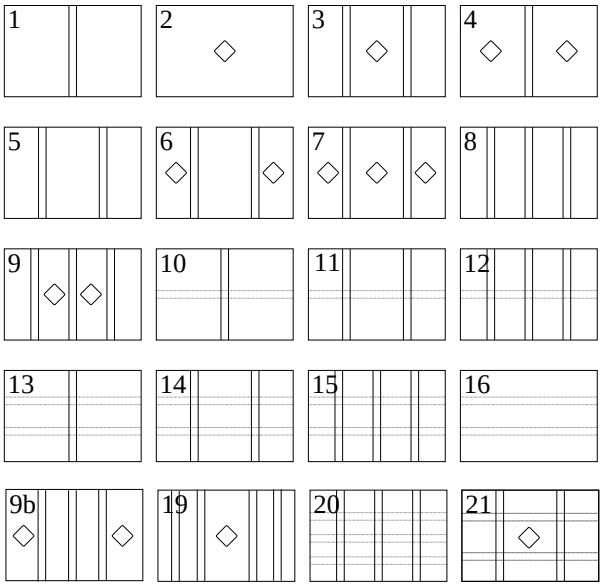


Fig. 5 Panel configurations that can be chosen in optimisation

7.1 Convergence

A components only optimisation (setup 1) with application of the extra rules mentioned in §5.2 typically converges in 50-80 feedback iterations (depending on the problem complexity) and takes about 18 hours to complete on 7 HP-UX workstations.

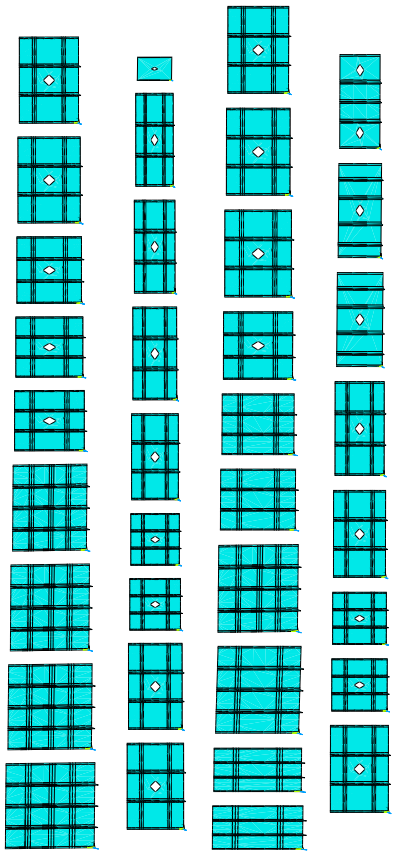
The structure optimisations (setups 2 and 3) needed 2-3 structure iterations to converge, using a fixed number of 35 component (NN feedback) iterations. With an average FE solving time of ca. 1 minute per panel the optimisations completed in 8-9 hours.

Runtimes are indications. During the process some machines were not available all the time. Also the speed is strongly dependent on the mesh sizes used. Nevertheless, it is clear that overnight optimisation of a complete structure is possible when enough computers are available.

7.2 Optimised design

A picture of the optimised spars is shown in figure 6. Keep in mind that all configurations are rotated 90 degrees compared to the ones depicted in figure 5. It is clear that for larger panels, more stiffeners are chosen to prevent buckling. A re-

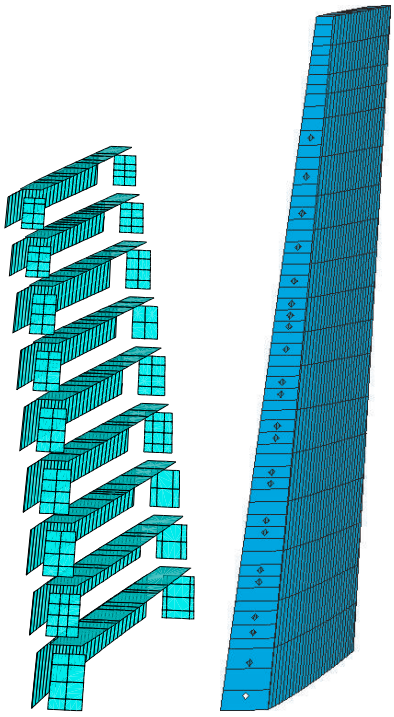
markable fact is the appearance of holes in panels where they were not required, probably due to the weight the hole itself saves. Surrounding a hole by stiffeners seems by far the best option when a hole is required. Addition of new topologies with one or more holes, more than 2 transversal stiffeners and 2 or more longituninal stiffeners might be attractive for upper panels in the rear spar, according to this result.



(a) Front spar (b) Rear spar

Fig. 6 Optimised designs (with realistic loadcases)

In figure 7 the optimised structure designs are depicted. As can be seen more stiffeners are chosen in the lower panels, for they typically have a higher loading. The influence of stiffener placement can be clearly seen in the rear spar of the first structure, which has longitudinal stiffeners placed divergently towards the lower end (towards higher bending).



(a) Setup 2 (b) Setup 3

Fig. 7 Optimised designs (loadcases are not realistic)

8 Conclusions

- The proposed program can be a powerful design tool as it allows overnight evaluation of design decisions such as:
  - Use of different materials
  - Different hole placement throughout the structure
  - Use of different variables (i.e. fixed or variable stiffener heights)
  - Allowance of different topologies
- The effect of load changes on the optimal design can be quickly evaluated
- The feedback of the FE outputs of preliminary optima to the NN is essential to reach an accurate optimum
- The use of additional rules (see §5.2) greatly improves the efficiency and effectiveness of the optimisation

- The robustness of the routine is not proven, though the fact that optima can be reproduced whithin narrow bounds with different initial NN training sets and other GA random seeds indicates convergence to a global optimum
- Integrated structure optimisation is relatively effi cient, for later iterations benefi t of (locally) well trained NNs from previous iterations (due to the fact that only loading changes).
- Neural Network training time can become dominant for NNs with large datasets (typically favourable topologies with many variables)

9 Acknowledgement

The contributions of Jeroen Hol and Jelmer Wind are highly appreciated.

References

[1] W.M. Jenkins. Neural network-based approximations for structural analysis. *Developments in Neural Networks and Evolutionary Computing for Civil and Structural Engineering*, pages 25–35, 1995.

[2] L. Ziemianski Z. Waszczyszyn. Neural networks in mechanics of structures and materials - new results and prospects of applications. *Proceedings of European Conference on Computational Mechanics, München*, 1999.

[3] S. Rizzo R. Spallino. Optimal design of laminated composite plates. Computer Aided Optimum Design of Structures IV, Boston, 1999.

[4] G. Giambanco R. Spallino, S. Rizzo. Evolution strategies for the shakedown optimal design of r.c. framed structures. European Congress on Computational Methods in Applied Science and Engineering, Barcelona, 2000.

[5] M.G. Kay C.R. Houck, J.A. Joines. *A Genetic Algorithm for Function Optimization: A Matlab Implementation*. GAOTv5 Manual (Genetic Algorithm Optimization Toolbox), 1998.