An automated process to create start values for gas turbine performance simulations using neural networks and evolutionary algorithms

Richard-Gregor Becker[1], Martin Bolemant[2], Daniel Krause[2] and Dieter Peitsch[2]

[1] German Aerospace Center (DLR), Institute of Propulsion Technology, Linder Hoehe, 51147 Cologne, Germany
[2] Technical University of Berlin, Institute of Aeronautics and Astronautics, Marchstrasse 12-14, 10587 Berlin, Germany

ABSTRACT

This paper presents a fully automated methodology to create start values for gas turbine performance computer programs. The methodology employs the application of evolutionary algorithms for a more robust convergence of the iterative process of a performance program as well as neural networks for a self-learning start value generation procedure.

The achieved results showed that a connection of both methods for the creation of performance model start values is a feasible option. Different types of neural networks as well as several training methods have been evaluated in order to find the best equivalent model. Having found a practical approach for the structure of the neural networks, several performance models of different levels of complexity have been tested. The combination of both of the above presented steps achieved very good convergence rates in combination with a minimum effort for the creation of start values.

NOMENCLATURE

Abbreviations

ANN	Artificial Neural Network
DLR	German Aerospace Center
ILR	Department of Aeronautics and Astronautics
MSE	Mean Square Error
TUB	Technical University of Berlin

Symbols

A	Area
C	Numerical Error of an Iteration Constraint
F	Thrust
FAR	Fuel to Air Ratio
J	Jacobian Matrix
N	Rotational Shaft Speed
P	Pressure
T	Temperature
V	Value of Iteration Variable
W	Mass Flow

Greek Symbols

∂	Differentiation Operator
β	Auxiliary Compressor Characteristic Coordinate

Subscripts

amb	Ambient
corr	Corrected
G	Gross
i	Index for Variables
j	Index for Constraints
n	Number of Variables / Constraints
N	Net
RED	Reduced
s	Static

Station Numbers

0	Free Stream Air Conditions
2	First Compressor Front Face
24	Intermediate Pressure Compressor Entry
26	High Pressure Compressor Entry
4	High Pressure Turbine Entry
42	Intermediate Pressure Turbine Entry
44	Low Pressure Turbine Entry
5	Last Turbine Discharge
9	Exhaust Nozzle Discharge

INTRODUCTION

Gas turbine computer programs are using component specific performance maps to determine gas turbine Off-Design behavior. Since the working points of the components are not known beforehand for a particular gas turbine power level, those working points are guessed at the beginning of a performance calculation and iteratively adapted to meet finally the laws of conservation of mass and energy. This problem solving process is represented by a non-linear equation system which is generally solved by employing a numerical gradient method such as the Newton-Raphson algorithm. The application of the Newton-Raphson method to gas turbine performance simulation has been rarely published in the past. Therefore this paper starts with giving a short overview of the numerical performance program solving process.

To achieve an acceptable rate of convergence and to maintain numerical stability the gradient procedures require well defined sets of start or guess values. These start values are typically provided as tables of non-dimensional parameters roughly describing the model behavior. In addition the influence of second order effects such as bleed offtake may be regarded depending on their influence. Generally, the procedure of start value generation is time consuming and requires tedious updates with each modification of the baseline performance model.

In order to alleviate the start value generation process this paper describes an alternative methodology for the creation of initial guesses, which is conducted in two steps: In a first step the gas turbine performance model conducts a number of calculations to create a set of in- and output values that represent the specific gas turbine model behavior. Since this initial performance model has no start values defined the convergence rate may expected to be poor. To avoid those convergence problems this paper introduces the employment of an evolutionary algorithm as a support to the conventionally used gradient method. This approach is supposed to be more robust, though it is from nature more time consuming compared to the Newton-Raphson algorithm.

Once the model has produced a representative set of output values, neural networks are applied in a second step as an analogous model which serves as a substitute to the conventional start value generation process. The neural network will be trained with the gas turbine model in- and output values compiled during the

first step.

The combination of both steps – the increase of the solver robustness and the self-learning guess model - provides an automated tool for creation of performance model start values.

THE DEFINITION OF THE NUMERICAL PERFORMANCE PROGRAM PROBLEM

The numerical problem of gas turbine performance simulations originates from the requirement for satisfaction of the laws of conservation. Due to the fact that for a requested power level the Off-Design operating points of the components (and hence their position in the corresponding component characteristic) are not known beforehand, those operating points have to be found iteratively.

Figure 1 shows the minimum set of variables (independents) and constraints (dependents) required for an Off-Design performance model of a simple single spool turbojet configuration. The total amount of variables and constraints is called the matching scheme.

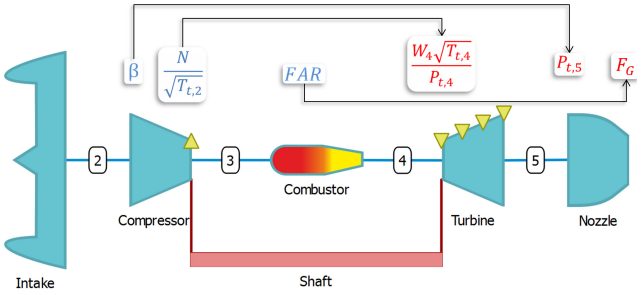


Fig. 1 Structure and matching of a single spool turbojet engine performance model

The first constraint is defined by the power level requirement, which is the engines thrust in the given example. Besides the power requirement, which is setup to fulfil the laws of conservation of mass, the compressor working point is - analogical to real engine operation - determined by the turbine and nozzle capacities at a given combustor temperature rise. Thus, the mass flow of the current turbine working point needs to correspond with the mass flow provided by the compressor, what represents the second constraint. The third constraint is defined by the nozzle entry pressure. It must be sufficiently high to squeeze the mass flow through the nozzle throat area.

To overcome the numerical errors revealed by the constraints the process variables are used. The fuel to air ratio FAR is taken as the first variable and is used to match the required power level, which is thrust for the given example. This is equivalent to real engine operation. Beyond, to find the compressor operating point, that is not known in advance, the non-dimensional speed and the auxiliary coordinate β are introduced as variables. The meaning as well as the reason for incorporation of the non-physics based parameter β is described in [1] and [2]. The non-dimensional speed and β are used to meet the constraints of the turbine entry non-dimensional mass flow and the nozzle entry pressure.

Besides the above described constraints, speed and power equality for turbo-components located on the same shaft needs to be satisfied for steady state operation. This is usually realised within the thermodynamic performance program directly.

To summarize, a proper equilibrium state is reached when the following statements apply:

1. The mass flow delivered by the compressor equals the turbine capacity at actual turbine working point.
2. The compressor pressure ratio is sufficiently high to squeeze the delivered mass flow through the nozzle throat.
3. The combustor temperature rise is adequate to comply with the power level requirement.

Table 1 gives an overview of the minimum matching schemes for three different engine types used further on in this paper. A broader overview about the required matching schemes for various types of real world gas turbine engines is given in [1].

THE CONVENTIONAL ANSWER TO THE NUMERICAL PERFORMANCE PROGRAM PROBLEM

The numerical performance program problem can be expressed by a system of non-linear equations. Thereby the values of the variables V_i are modified until the sum of numerical errors of the constraints C_j diminishes to zero. To solve the numerical problem either the nested loop or Newton-Raphson technique are usually employed.

Nested loop technique

In the early days of gas turbine performance simulation the nested loop technique was used. Herein the variable and the constraint which correspond (most) to each other are solved individually in nested loops. Examples for the application of nested loops are given in [1] and [3].

The rise of the computer performance allowed a growing complexity of gas turbine performance models. As a consequence the drawbacks of the nested loop technique became more and more significant such as an inflexible and confusing structure, numerical inefficiency and poor convergence rates [4].

Newton-Raphson algorithm

As a substitute to the nested loop technique, the Newton-Raphson algorithm has been employed to gas turbine performance calculation. With its introduction a consequent separation of mathematics and physics could be achieved. A model update or addition of a further constraint did not result in a modification of the solver algorithm anymore as for the nested loop technique.

In the first step of the Newton-Raphson algorithm process the partial derivatives $\frac{\partial C_j}{\partial V_i}$ of the numerical error functions are determined. The total quantity of partial derivatives is combined in the Jacobian matrix J as shown by equation (1).

$$J(\boldsymbol{V}) = \begin{pmatrix} \frac{\partial C_1}{\partial V_1} & \cdots & \frac{\partial C_1}{\partial V_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial C_n}{\partial V_1} & \cdots & \frac{\partial C_n}{\partial V_n} \end{pmatrix} \quad (1)$$

Table 1 Required matching schemes for different model types

	Single Spool Turbojet			Two-Spool Turbofan, unboosted with seperate nozzles		Three-Spool Turbofan with seperate nozzles		
Variables	β_{26}	$\frac{N_H}{\sqrt{T_{t,26}}}$	FAR	β_2	β_{12}	$\frac{N_L}{\sqrt{T_{t,2}}}$	β_{24}	$\frac{N_I}{\sqrt{T_{t,2}}}$
Constraints	$P_{t,5}$	$\frac{W_4\sqrt{T_{t,4}}}{P_{t,4}}$	F_G	$\frac{W_{44}\sqrt{T_{t,44}}}{P_{t,44}}$	$P_{t,15}$	$\frac{W_{26}\sqrt{T_{t,26}}}{P_{t,26}}$	$\frac{W_{42}\sqrt{T_{t,42}}}{P_{t,42}}$	$\frac{W_{24}\sqrt{T_{t,24}}}{P_{t,24}}$

In order to find the zeros of the numerical errors C_j the equation system (2) is solved in a subsequent step using the Gaussian algorithm.

$$\Delta \boldsymbol{V}_i = -\left(J(\boldsymbol{V}_i)\right)^{-1} f(\boldsymbol{V}_i) \tag{2}$$

Naturally, both steps have to be repeated several times until the numerical error is within the predefined tolerance due to the non-linear character of the numerical error functions.

A description of the implementation of the pure Newton-Raphson algorithm as well as specifications about modifications is provided by [5]. Broyden proposes an advancement [6] to reduce the number of required iterations.

The application of the Newton-Raphson algorithm to gas turbine performance programs is for example specified in [3], [4], [7] and [8]. Schutte [9] mentions potential numerical convergence problems and introduces as an answer to this a step size regulation of the variables. This method is picked up later in this paper as a possible means to improve the solver convergence rate.

THE NEED TO DEFINE START VALUES

The convergence rate as well as the number of conducted iterations of the Newton-Raphson solving process is extremely dependent on the quality of the predefined start or guess values. Conventionally, the start values of the variables for Off-Design calculations are set up non-dimensionally as functions (lookup tables) of the non-dimensional power parameter (e.g. thrust for aero- engines).

However, due to the non-ideal non-dimensional gas turbine behaviour, the guess values need to include at least the influence of flight Mach number and preferably bleed setting in order to keep the convergence rate high. Other geometry changes of major impact might also be incorporated. For detailed models, containing a complex control system logic, it might be worth considering altitude as well. Finally, the lookup tables for the start values end up multidimensional, requiring lots of time for creation.

1. STEP: INCREASE OF THE SOLVER ROBUSTNESS

This paper introduces two different methods to increase the solver robustness when not having suitable guess values defined. The first is a rather simple approach which automatically introduces a bunch of intermediate steps when the initial jump from one Off-Design point to another failed to converge. The second makes use of an evolutionary algorithm. Theoretically, both methods can be combined, whereby the evolutionary algorithm would serve as fall back solution when the intermediate step method does not achieve the desired success.

All of the below introduced improvements have been implemented in DLR's C++ in-house performance code GTlab-performance [7].

Automated introduction of intermediate calculation steps

According to [9], three failure modes for the Newton-Raphson solving process and related algorithms exist: The first and also most trivial failure mode describes the situation when there simply is no solution for the given problem. It is evident that no mitigation process for this mode is available. Problems of the second failure category occur because the start value for the iteration process is not close enough to the solution and the iteration algorithm diverges. The third and last category contains problems for which the target function is discontinuous and the successive execution of the gradient steps leads to alternating solutions.

Convergence problems of the second category can be avoided by an automatically executed stepwise adaptation of the target operating point. For the present paper, a stepping algorithm similar to the solver convergence failure repair algorithm described in [9] was implemented. A brief description of the approach is given in the following.

At the beginning of the robust stepping algorithm the current state of the solver variables such as the boundary and initial conditions, the independent variables and the user defined dependent variables are stored as the baseline reference. Furthermore the target conditions, which define the new operating point, are also saved as the target reference. The solving process starts with a standard Newton-Raphson iteration. In case of convergence, no robust stepping is needed. If the Newton-Raphson attempt fails to converge, the target solver variables are adapted to a new operating point which lies halfway between the baseline reference and the target reference. The solver then tries to find the solution to the new operating point. If the solver attempt is successful, the solver variables of the new operating point are saved as the baseline reference and the target solver variables are found by incrementing the new baseline variables with the same step size. In case of no convergence the step size is divided further to get even closer to the initial reference. The process is repeated until a solution to the target reference point was found or specified numbers for maximum steps and maximum bisection depth are reached.

This rather simple approach effectuates a substantial gain in solver robustness. During the execution of test cases throughout the whole flight envelopes at various power settings for a 2-shaft and a 3-shaft turbofan engine, it has been observed that the implementation of the stepping algorithm as a fall back solution to the standard Newton-Raphson based approach cured almost every convergence problem of the present study. However, the robustness gain is bought dearly by computational efficiency deficits. Furthermore, the stepping algorithm fails on convergence problems of failure category three, which lead to the introduction of a more complex solving strategy described in the following subsection.

Incorporation of an evolutionary algorithm

An alternative means to increase the robustness of the performance program solving process has been provided by adding an evolutionary algorithm. Evolutionary algorithms are stochastic search methods used for optimisation with a high likelihood to find the global optimum. Those types of algorithms mimic natural evolutionary processes by using the operations derived by Darwins theory of evolution, namely selection, crossing and mutation. The resulting solutions, which are called individuals, are evaluated by a so called fitness function. The fitness value of each individual indicates its probability to survive. The higher the fitness value of an individual (it equals the quality of the solution), the more likely will be its survival. Due to their stochastic mode of operation evolutionary algorithms are very time consuming.

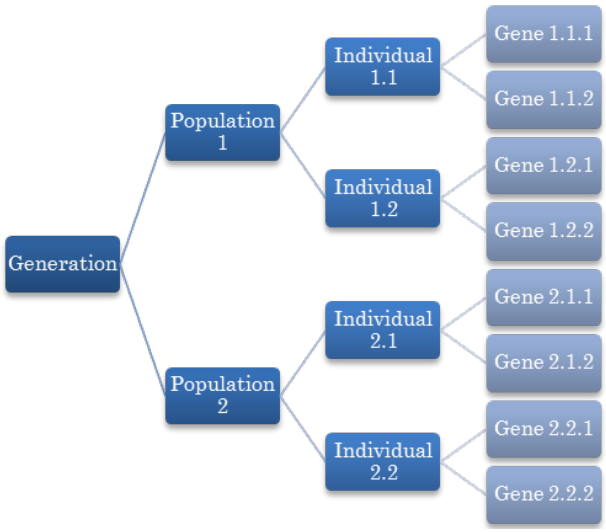


Fig. 2 Program Structure of Evolutionary Algorithm

The employed evolutionary algorithm has been developed at the Department of Aeronautics and Astronautics (ILR) of Technical University of Berlin (TUB). Its basic data structure is shown in figure 2. Each generation is split into several populations in order to provide the possibility to conduct parallel computing. A population comprises a number of individuals, each of them representing a possible solution for the respective problem. Individuals contain a set of genes. Any gene is representing a variable of the process wherefore the optimum is searched. Thereby the genes are characterised by floating point numbers as it is usual for evolutionary strategies. This is in contrast to the representation of the gene values within genetic algorithms (GAs). Within GAs the genes are implemented in binary coded notation.

Between two subsequent generations the operations selection, crossing and mutation are applied to the individuals. The introduced evolutionary algorithm contains multiple methods for each of operation. For the current problem, which is the increase of performance model solver robustness, adequate methods have been selected as default from the available:

- Selection is realised by a tournament selection. Herein, for a number of randomly selected individuals out of the current generation the strongest individual (with highest fitness value) will be added to the next generation. This procedure is repeated until the new generation contains the predefined number.
- Crossing is realised by uniform cross over. Each gene of a child individual has a user defined probability to be inherited from either parent individual 1 or 2.
- The option non-uniform is applied for the operation mutation. The particular genes are varied with a Gaussian distribution around its previous value.

Beales function:

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2 \quad (3)$$

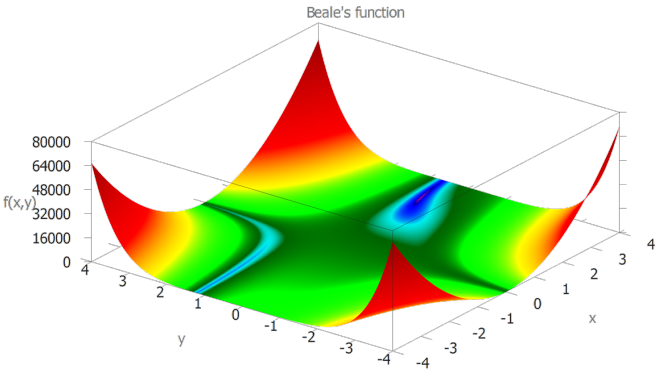
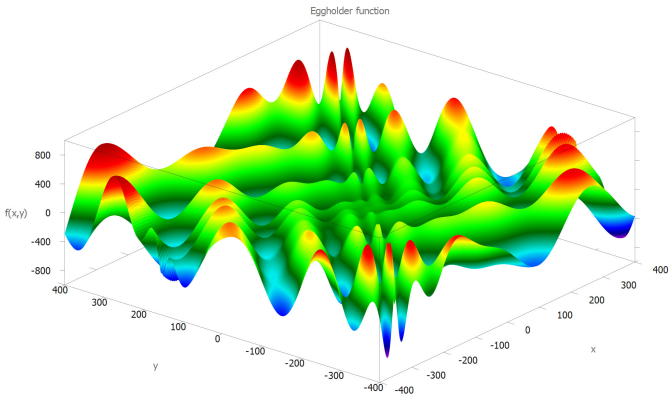


Fig. 3 Beale's test function for optimization algorithms

The proper function of the developed evolutionary algorithm has been proven by conducting a large number of tests with dedicated optimisation test functions as defined in [10]. The algorithm has successfully passed those tests. Two examples of the test functions are Beale's (3) and Eggholder (4) function. Their function traces are shown in figures 3 and 4. Especially, for functions as the highly multimodal Eggholder function, the success of gradient based optimization algorithms is very limited. For this purpose evolutionary algorithms are much more suitable.

Eggholder function:

$$f(x,y) = -(y+47)\sin\left(\sqrt{\left|y+\frac{x}{2}+47\right|}\right) - x\sin\left(\sqrt{|x-(y+47)|}\right) \quad (4)$$



Besides the search for a single optimum, the implemented evolutionary algorithm is capable of doing a multi objective optimization. Though, for the current application the single objective approach is the method of choice. The utilised objective function f_O has been defined as the sum of squares of the relative numerical constraint errors as shown by equation (5). Due to the fact that the actual problem needs to be minimised but the evolutionary algorithm searches for a maximum the sign of the objective function is inverted.

$$f_O(\boldsymbol{C}) = (-1) \cdot \sum_j (C_j)^2 \quad (5)$$

The above described evolutionary algorithm has been coupled to the performance program GTlab-performance [7]. The interface between both programs is realised by an appropriate identification of the performance program variables with genes on the evolutionary algorithm side. The combination of genes to an individual represents a single possible solution to the performance model.

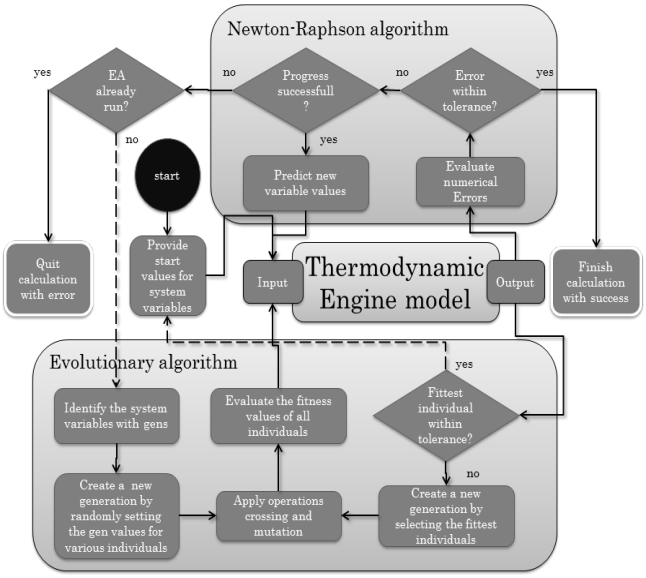


Fig. 5 Sequence of performance program coupled with Evolutionary Algorithm

Figure 5 displays the process procedure. In case the conventional solving process fails, the evolutionary algorithm is activated with the aim to find the set of variable values that maximises the objective function (5). The absolute maximum is for the current problem always zero.

The first generation is created by setting random values for the genes of all individuals. Following, the objective function of each individual is evaluated by running the performance model, where for the variables are set to the values provided by the genes. The numerical constraint errors of the performance model determine the individual fitness with equation (5). For the subsequent generation the fittest individuals are selected followed by an application of the operations crossing and mutation. Again the objective functions are determined by running the performance model and the whole process is repeated until the fittest individual fulfils the desired fitness requirement.

Due to the very time consuming nature of the evolutionary algorithm it is a suitable option to choose a broad tolerance, what means that the stop criterion can be well below zero. This is okay since the evolutionary algorithm should not solve the problem completely. It serves thereby as a case specific start value generator. If an individual is found that exceeds the desired fitness value, the corresponding variable setting is used to run the Newton-Raphson algorithm again. This procedure shortens the solving time compared to a pure application of the evolutionary algorithm. On the basis of the performed investigations the criterion for quitting the evolutionary algorithm has been set to -1e-01. When reaching fitness above that value, the Newton-Raphson solver always managed to find the solution properly.

The proposed methodology has been tested on a performance model of a 2-shaft turbofan engine that includes variables as shown in table 1. By the employment of an evolutionary algorithm as a support to the Newton-Raphson solver all of the test cases have shown convergence.

Investigations have been conducted regarding an optimal size of the populations. The termination criterion for the tests has been set to a value of -1e-01. The achieved results (figure 6) show that the number of required generations to reach the termination criterion reduces with rising number of individuals per population. This was expected since the probability to find a good individual in a generation is higher if the population size and with it the generation size is higher. Nevertheless, the calculation time between the different results was nearly equal due to the similar number of conducted model evaluations.

As a good compromise finally the value of 600 individual per population has been chosen.

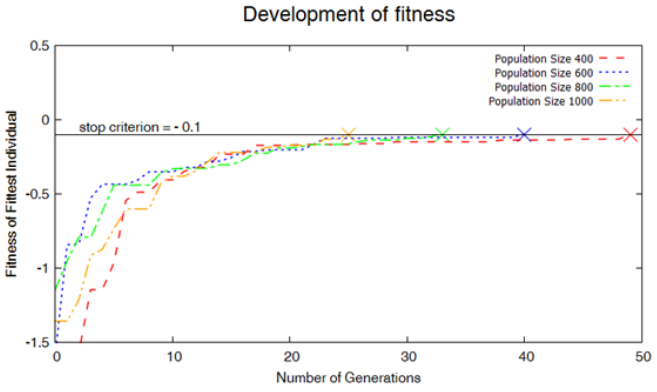


Fig. 6 Results from the employment of an Evolutionary Algorithm for different population sizes

2. STEP: INCORPORATION OF AN ALTERNATIVE GUESS MODEL

For the purpose of start value generation any model can be used which serves as an equivalent to the baseline performance and has a comparatively low complexity. As far as the authors know, all currently used guess generators utilize non-dimensional or quasi non-dimensional performance parameters in order to take advantage of the non-dimensional behaviour of the gas turbine and thus to reduce parameter variations over the flight envelope. Table 2 lists the most commonly used non-dimensional parameters and their representation for guess generation.

Table. 2 Quasi non-dimensional parameter groups used in guess models

Performance Parameter	Quasi Non-Dimensional Group
Shaft Speed	$\frac{N}{\sqrt{T}}$
Air Mass Flow	$\frac{W \cdot \sqrt{T}}{P}$
Fuel Mass Flow	$\frac{WF}{P \cdot \sqrt{T}}$
Thrust	$\frac{F}{A9 \cdot Ps0}$

In the following subsections two guess models are presented. The first model represents the current standard for guess generation and is based on interpolation tables. This approach is presented as the reference methodology and serves as baseline only. The second model is an alternative guess model based on artificial neural networks that was implemented to overcome the drawbacks of the reference model. The principles of operation of both models are described and details of the development process of the alternative model are given.

Classical interpolating guess models

In classical interpolating guess models, the generation of start values is performed by means of interpolation tables. These interpolation tables store excerpts of the engines performance behaviour in form of quasi non-dimensionals at discrete operating points. The excerpts contain data of the following categories:

- Environmental boundary conditions such as e.g. the flight altitude, the ambient temperature and flight Mach number
- Aircraft/Engine boundary conditions such as the amount of customer bleed air or shaft power extracted from the engine
- Power parameters e.g. the engine pressure ratio (EPR), shaft speed (NL), power lever angle or fuel flow
- All parameters that can be used as independent variables to the solving process.

Typically, the stored performance data has been pre-calculated by the gas turbine performance model and covers a grid like structure of operating points over the complete flight envelope with variations to the aircraft/engine boundary conditions and power parameters. In order to reduce the complexity of the interpolation tables, it is common practice to provide two sets of tables: The first table set maps the available power parameters to a single primary power parameter. For the present paper the primary power parameter was chosen to be NHRT2, which is the speed of the high pressure shaft corrected by the total temperature at the first compressor front face. Figure 7 shows plots of the mapping tables provided for the mixed flow two-shaft turbofan model used throughout the paper. The graphs chart the primary power parameter NHRT2 exemplarily over the non-dimensional secondary power

parameters EPR and NL. Table 3 gives their definitions.

Table. 3 Power parameter definitions

Parameter	Brief	Definition
FNQPSAMB	Quasi Non-Dimensional Net Thrust	$\frac{F_N}{Ps, amb}$
EPR	Engine Pressure Ratio	$\frac{P5}{P2}$
NLRED	Reduced Low Pressure Shaft Speed	$\frac{NL}{\sqrt{T2}}$

A grey-scale colour map is used to show the parameter variations with given flight Mach number. It can be observed that the Mach number has a strong influence on the variations of the non-dimensional thrust and the engine pressure ratio. Thus, the Mach number needs to be reflected as an additional dimension for the interpolation of both parameters at minimum.

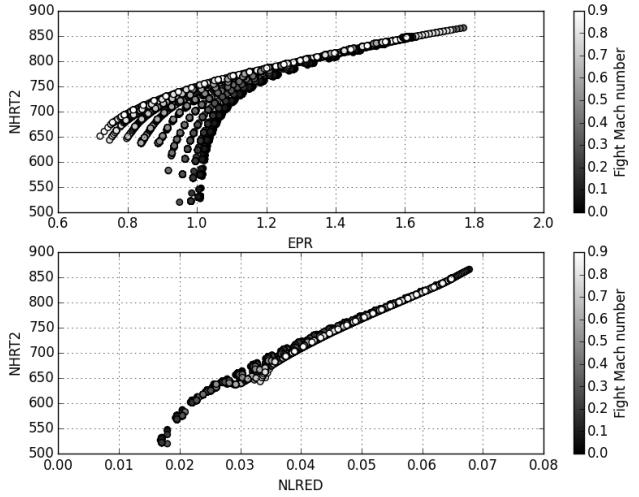


Fig. 7 Power parameter conversion tables to the primary parameter NHRT2: Engine pressure ratio (top) and corrected low pressure shaft speed (bottom)

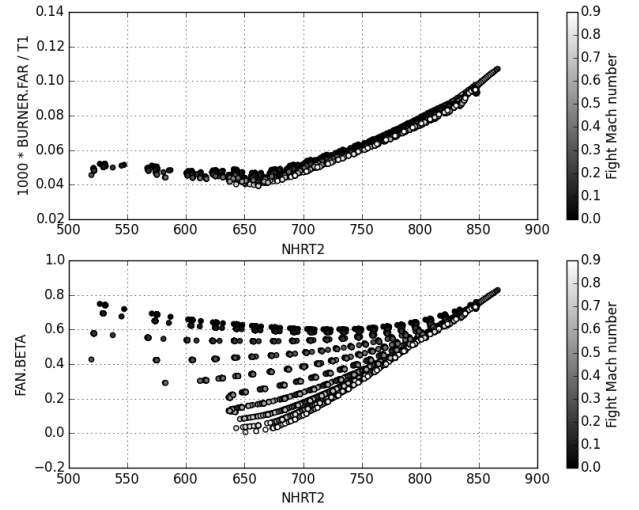


Fig. 8 NHRT2 to independent conversion data tables. Non dimensional fuel flow (top) and auxiliary fan characteristics coordinate β (bottom)

After the determination of the primary power parameter by means of the first table set it can be used as an input to the second interpolation series, which maps the power parameter to specified independent variables. In total six additional independent parameters as described in table 1 are provided for the two shaft turbofan example of the present paper. Figure 8 shows two representative parameter maps for the quasi non-dimensional fuel to air ratio and the auxiliary fan characteristics coordinate β. Again, the greyscale indicates the Mach number influence on the parameters.

Once the described procedure of table based guess model generation is set up, it is a robust and fairly accurate tool to predict start values. However, the generation of the necessary grid points especially when using additional dimensions for secondary parameters such as bleed and/or power extractions is time consuming and error prone. To evade the drawbacks of the classical model a new alternative approach was implemented, which is described in the next subsection.

Guess models based on artificial neural networks

To alleviate the generation of the start values self-learning artificial neural networks (ANN) have been implemented as an alternative to the classical guess table approach. Artificial neural networks are particularly suitable for the task of guess modelling, because

- ANN can be used to deduce functional behaviour from observations and thus can be trained on arbitrary operating point data
- ANN have the ability to implicitly detect complex non-linear relationships between independent and dependent variables [11]
- Neural networks have the ability to detect all possible interaction between predictor variables [11]

The idea was to automatically train a single neural network on a preferably small set of operating point data that is able to predict starting values for all needed independent variables on the basis of given boundary conditions and power parameters. The training of the ANN has to be carried out for each engine model that is meant to be augmented by guess models. Similar to the generation of sampling nodes for the classical tables method, the ANN training data is obtained by operating point simulations. However, ANN training algorithms do not depend on spatially structured or ordered data, which makes the sampling process easier compared to classical guess model generation.

To use the ANN approach in real world applications the neural network functionality needs to be incorporated into the gas turbine performance program. For the present paper the FANN software library, which is described in [12], was linked to the GTlab performance synthesis module. The decision to use FANN was based on the fact that it is published under a liberal software license. It is also well documented and features a graphical user interface besides the C++-library module. For the generation of an alternative guess model by means of artificial neural networks studies have been conducted to find an ANN topology that is appropriate to model a wide range of engine models. The investigated neural network topologies share certain common properties:

- Standard back-propagation is used as the training method for all networks.
- The sigmoid function was chosen as a common activation function.
- The training was stopped, when a mean squared error of 1.0e-04 was reached.

Figure 9 shows simplified topology of the artificial neural networks that have been used throughout the investigation. Therein, the leftmost column represents the input layer, which consists of at least four input neurons. Three of the input neurons represent the environmental boundary conditions, namely the flight altitude, the flight Mach number and the ambient static temperature. Of course, any other parameter set describing the static and total thermody-

namic state of the engine environment would be suitable. The fourth input neuron is needed to map the input power parameter. Tripled black dots have been inserted to visualize the possibility of extending the input layer by additional neurons. These would be needed to account for additional parameters such as bleed or power extraction or alternative power parameters. Future work will reveal the practicability. The two middle columns are hidden layer section. In the present study both - the number of hidden layers and the number of hidden layer neurons - have been varied, which is indicated by groups of black dots. The rightmost column shows the output layer. The number of output neurons depends on the number of performance model variables that have to be predicted. The minimum number of variables is coupled to the gas turbine process under investigation as described in Table 1.

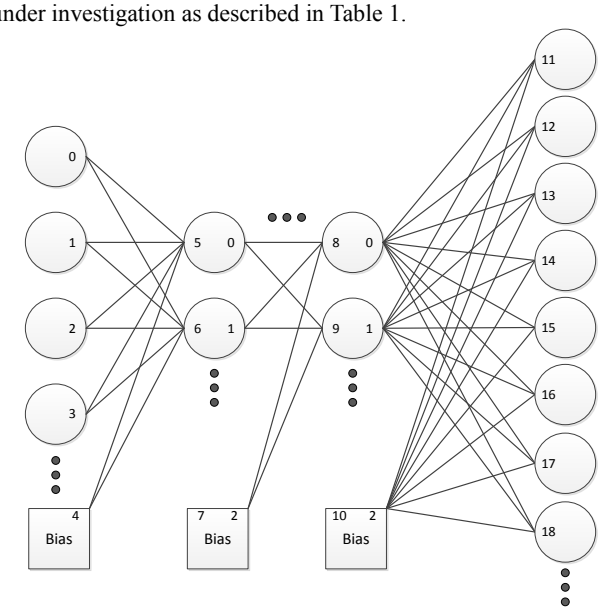


Fig. 9 General topology of the investigated artificial neural networks

Studies have been undertaken to find the network topology parameters which are best suitable for guess modelling of aircraft engines. For that purpose two generic aircraft engine models have been generated: a generic two-shaft turbofan engine and a three-shaft turbofan engine. Both engine models were used to generate ample sets of training data. In order to find optimal ranges for the number of hidden layers and the number of hidden layer neurons with regard to accuracy and computational performance, parametric studies on both training data sets have been conducted.

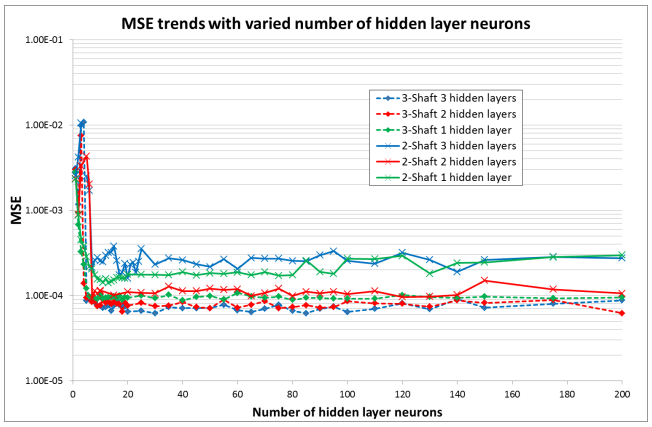


Fig. 10 Trends of the mean square error of the training with varied number of hidden layers and hidden layer neurons

Figure 10 plots trends of the mean square error (MSE) of the ANN approximations over the number of hidden layer neurons. The dashed trend lines represent error data for the 3-shaft model, whereas the full lines cover data of the 2-shaft engine. The colours green, red and blue represent network topologies with one, two and three hidden layers respectively. It can be seen that for all investigated configurations a range from 12-20 hidden layer neurons is sufficient to approximate the engine behaviour in reasonable accuracy. Decreasing the number of hidden layer neurons any further would expose the model to the danger of a drastic decrease in accuracy of one to two orders of magnitude. Further increase of the neuron numbers does not lead to improvements in accuracy and is thus ineffective. In terms of the number of hidden layers two layers seem to be the best compromise between both models. Interestingly, increasing the number of hidden layers to three layers did not come along with an increase in accuracy. This observation is assumed to be caused by over fitting of the neural network.

Another observation can be made about the approximation accuracy for both engine models. The MSE of the 2-shaft model approximations is generally higher than the errors of the 3-shaft predictions despite the fact that the 3-shaft model is slightly more complex. This might be connected to an incompleteness of the 2-shaft performance model.

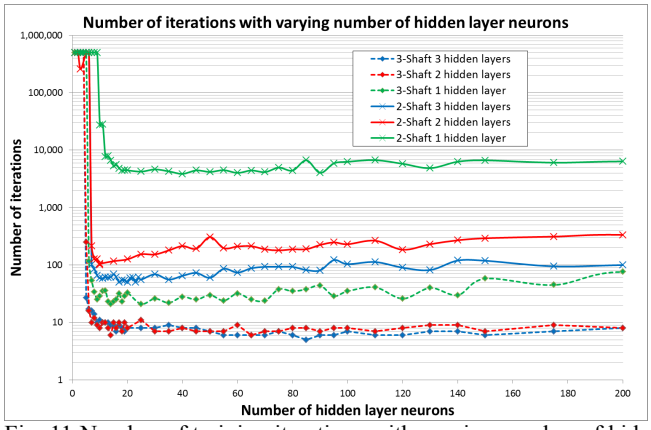


Fig. 11 Number of training iterations with varying number of hidden layers and hidden layer neurons

Figure 11 depicts the number of iterations needed by the different topology and problem configurations to reach a mean square error in the order of 1e-4. Again dashed lines represent the 3-shaft engine and full lines the 2-shaft engine models. The colour scheme is continued. The chart supports the assumption on an optimal range of 12-25 hidden layer neurons, because for this range also the number of iterations seems to be minimal. Consistently, one hidden layer appears to be inferior to two or three layers due to its lack of approximation capabilities. In terms of the number of iterations alone, the three hidden layer topology seems to be the best option.

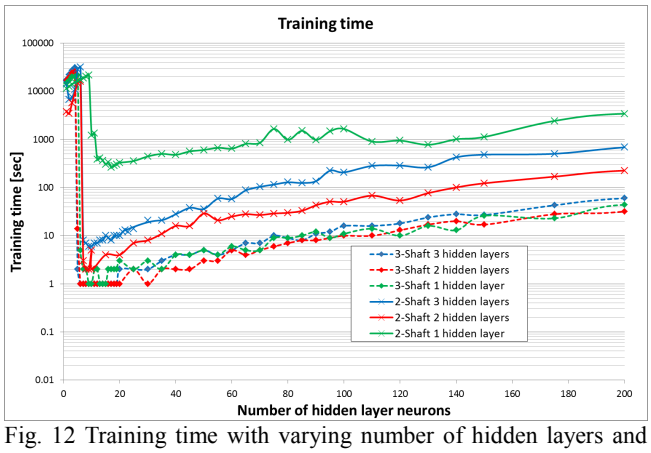


Fig. 12 Training time with varying number of hidden layers and hidden layer neurons

Figure 12 plots the training time consumed to iterate the mean squared error of the investigated configuration approximations below the desired accuracy limit. Line style and colour scheme are continued. The training time results indicate that a range of 10 to 25 hidden layer neurons is the best choice for the guess model setup. In terms of computational time two hidden layers perform best for both engine models.

The results of the topology studies lead to the conclusion that in terms of accuracy and computational time a network topology with 2 hidden layers each equipped with 15-25 hidden layer neurons is a suitable compromise for guess model approximations. In the following test cases, network topologies with these parameters have been employed exclusively.

TEST CASE DEFINITION AND RESULTS

To evaluate the general robustness and computational speedup of the classical and the alternative guess models a test case was implemented. The objective of the implementation was to emulate guess model operation under typical but challenging conditions. To do so, the test case utilizes the generic two-shaft mixed flow turbofan model described in reference [13] and executes sequences of randomly chosen operating point simulations to evaluate the numeric performance and stability of the guess models. Three types of guess models have been evaluated. The first is a standard configuration of the GTlab performance code augmented by the robust solver mode. No additional guess model was used. This configuration is used as the baseline. The second configuration is a robust solver mode augmented code with additional classical interpolation tables for start value generation. The guess tables have been generated on the basis of flight envelope computations with a resolution of 8000 evaluated operating points. The third and last configuration uses the activated robust solver mode and an artificial neural network guess model. The corresponding ANN topology features 4 input neurons for the boundary conditions and power parameter, 2 hidden layers with 22 neurons respectively and 10 neurons in the output layer that represent the independent variables to be guessed. The network was trained on a minimal flight condition resolution which is comprised by the corner points of the generic flight envelope surplus the design cruise condition as depicted in figure 13. Definitions of the corner points may be taken from table 4. For all boundary condition points working lines have been evaluated from minimum to maximum power level to generate the training data.

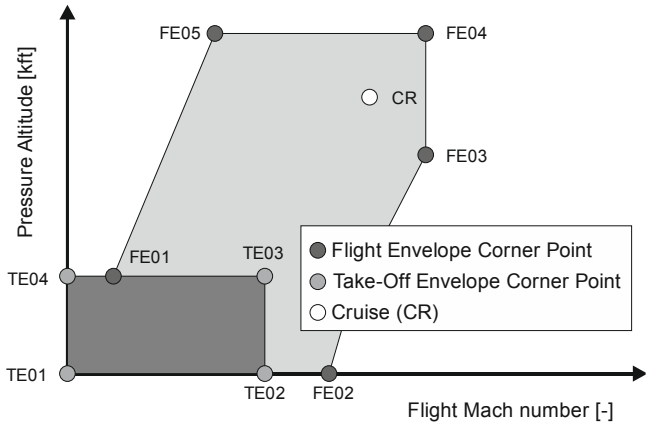


Fig. 13 Generic flight envelope definition with corner point designation

Table. 4 Definition of the flight envelope corner points

Operating Point	Mach number [-]	Altitude [ft]	Altitude [m]
TE01	0	0	0
TE02	0.45	0	0
TE03	0.45	9000	2743.2
TE04	0	9000	2743.2
FE01	0.2	9000	2743.2
FE02	0.5	0	0
FE03	0.85	29000	8839.2
FE04	0.85	39000	11887.2
FE05	0.35	39000	11887.2
CR	0.8	35000	10668

The most frequent failure mode for Newton-Raphson iterations in gas turbine performance codes occurs when the start value for the iteration process is not close enough to the solution and the iteration algorithm diverges. To provoke the occurrence of such failures the test case generates operating points that are randomly distributed in the flight envelope as depicted in figure 14. In addition to the sampled altitude and Mach number parameters, the power level was set by a uniform random distribution of the quasi non-dimensional thrust. Secondary parameters such as customer bleed or power extraction have been kept constant.

A test case execution was comprised of 30 runs. Each run generated 100 random operating points, which are consecutively evaluated by GTlab-performance using the three different calculation modes. The overall numbers of iterations, the overall runtime as well as the number of convergent operating points and the number of robust mode operations have been recorded. It is worth to mention, that although a high total number of evaluations were performed none of the three guess model configurations failed to converge an operating point. For the last two configurations with embedded guess models, no robust solver operations have been recorded whereas the standard configuration failed to converge about 6.3% of the test points in first pass and had to resort to robust solver mode.

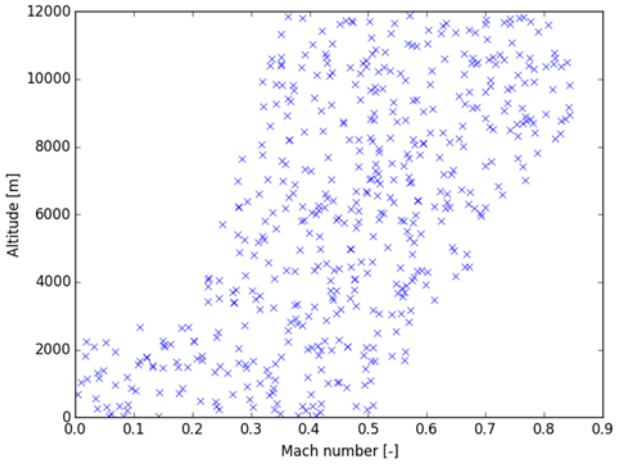


Fig. 14 Randomly generated operating points within generic flight envelope

Figure 15 compares the results for the overall number of iterations needed to converge the 3000 evaluated operating points. It can be seen that both guess models help to reduce the number of iterations. In case of the classical table based guess model about 39% of the iterations needed by the standard configuration can be saved. The neural network based configuration performances even slightly better and achieves a saving of 47%.

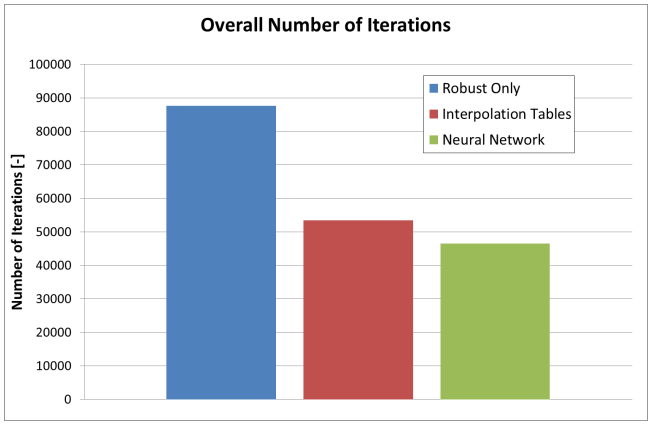


Fig. 15 Overall number of iterations needed by different guess models to fulfil the random envelope experiment

Figure 16 presents a runtime comparison for the guess model configurations. It can be observed that the runtimes of the configurations with implemented guess models are about 36% and 40% lower than the runtime of the standard configuration. However, the improvements that were achieved in terms of iteration count could not be transferred one-to-one to runtime enhancements. This may be attributed to the additional computational costs that are associated with the evaluation of the interpolation tables and artificial neural networks respectively. Additionally, the recorded data suggest that the runtime of the neural network evaluations is slightly higher than the evaluation of the classical table model. However, single comparative evaluation tests did not support this observation. Until further investigations are conducted, these small differences are credited to implementation details within the performance code.

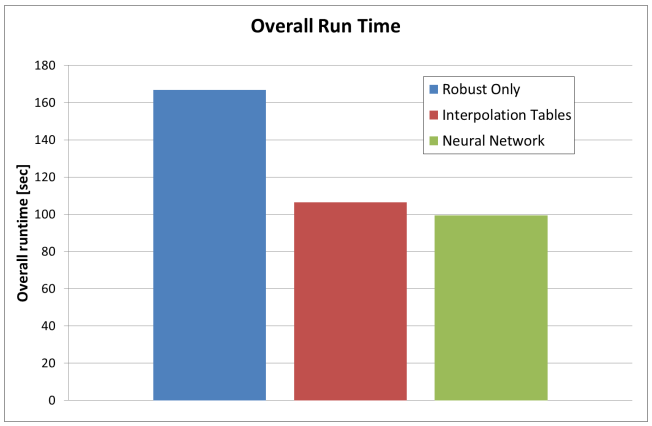


Fig. 16 Overall runtime required by different guess models to fulfil the random envelope experiment

CONCLUSION

A new methodology to increase performance program robustness has been presented. The methodology combines the incorporation of an evolutionary algorithm as a support to the gradient solver as well as an alternative guess model based on artificial neural networks.

On the solver side two alternative methods have been presented to improve the performance model convergence rate when no start values are defined. One method is doing a step wise approximation of the inputs between two calculation points the other makes use of an evolutionary algorithm. Both led to 100% convergence rates for properly set up performance models.

Furthermore, an artificial neural network topology has been identified by means of parametric studies that satisfies the guess modelling requirements with regard to accuracy and computational resources. The application of the found network topology to generic test cases showed promising results in comparison to classical guess model generation with interpolation tables. The artificial neural network approach was superior to the classical methodology in both the number of required iterations and the overall computational time. Yet, the most important advantages of the neural network approach lie in its automation capability, the comparably low effort to generate the training data and the effortless incorporation of secondary input parameters.

In summary the presented methodology of increasing solver robustness and incorporation of a self-learning guess value generator, provides a tool for an almost automated handling of guess values. Thus, the time spend on guess creation can be reduced significantly by the proposed method.

Future investigations will be undertaken to test and quantify the neural network approach with respect to a higher flexibility in power parameter choice as well as the incorporation of additional dimensions like bleed and power off-take. Furthermore other types of surrogate models like Gaussian process regression are planned to be considered in future studies.

REFERENCES

[1] Walsh, P., Fletcher, P., 2004, “Gas Turbine Performance”, Blackwell Publishing

[2] Bolemant, M., Peitsch, D., 2014, “An Alternative Compressor Modeling Method within Gas Turbine Performance Simulations”, DLRK 2014-340047, DGLR Kongress, Augsburg

[3] Applied Vehicle Technology Working Group 036, 2007, “Performance Prediction and Simulation of Gas Turbine Engine Operation for Aircraft, Marine, Vehicular and Power Generation”, NATO Research and Technology Organization, RTO TR-AVT-036

[4] Kurzke, J., 2007, “About Simplifications in Gas Turbine Performance Calculations”, GT2007-27620, ASME Turbo Expo 2007, Montreal

[5] Press, W., Teukolsky, S., Vetteling, W., Flannery, B., 2007, “Numerical Recipes”, 3. Edition, Cambridge University Press

[6] Broyden, C., 1965, “A class of Methods for Solving Nonlinear Simultaneous Equations”, Math. Compt. 19 577-593, Mathematics of Computation

[7] Becker, R., Wolters, F., Nauroz M., Otten, T., 2011, “Development of a Gas Turbine Performance Code and its Application to Preliminary Engine Design”, DLRK2011-24485, Deutscher Luft- und Raumfahrtkongress 2011, Bremen

[8] Xingxing, J., Chunwei, G., Hong, L., Weihong, X., 2013, “Performance Calculation of a Three-Shaft Gas Turbine: Experiment Evaluation and Analysis”, GT2013-94490, ASME Turbo Expo 2013, San Antonio

[9] Schutte, J., 2009, “Simultaneous Multi-Design Point Approach to Gas Turbine On-Design Cycle Analysis for Aircraft Engines”, PhD Thesis, Georgia Institute of Technology

[10] Jamil, M., Yang, X., 2013, “A literature survey of benchmark functions for global optimization problems”, Vol. 4, No. 2, pp.150-194, Int. Journal of Mathematical Modelling and Numerical Optimisation

[11] Tu, J. V., 1996, “Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes”, Vol. 49, No. 11, pp. 1225-1231, Journal of Clinical Epidemiology

[12] Nissen, S., 2005 , “Neural Networks Made Simple”, Vol. 2, pp. 14–19, Software 2.0

[13] Schnell, R., Ebel, P.-B., Becker, R.-G., Schoenweitz, D., 2013, “Performance Analysis of the Integrated V2527-Engine Fan at Ground Operation”, ODAS 2013