International Journal of
*Turbomachinery
Propulsion and Power*

IMPACT FACTOR 1.3

CITESCORE 2.3

Article

Turbofan Performance Estimation Using Neural Network Component Maps and Genetic Algorithm-Least Squares Solvers

Giuseppe Lombardo, Pierantonio Lo Greco and Ivano Benedetti

MDPI

https://doi.org/10.3390/ijtpp9030027

International Journal of
Turbomachinery Propulsion and Power

Article

Turbofan Performance Estimation Using Neural Network Component Maps and Genetic Algorithm-Least Squares Solvers

Giuseppe Lombardo *, Pierantonio Lo Greco and Ivano Benedetti

Dipartimento di Ingegneria, Università degli Studi di Palermo, Viale delle Scienze, Edificio 8, 90128 Palermo, Italy; pierantonio.logreco@unipa.it (P.L.G.) ; ivano.benedetti@unipa.it (I.B.)
* Correspondence: giuseppe.lombardo@unipa.it

Abstract: Computational models of turbofans that are oriented to assist the design and testing of innovative components are of fundamental importance in order to reduce their environmental impact. In this paper, we present an effective method for developing numerical turbofan models that allows reliable steady-state turbofan performance calculations. The main difference between the proposed method and those used in various commercial algorithms, such as GasTurb, GSP 12 and NPSS, is the use of neural networks as a multidimensional interpolation method for rotational component maps instead of classical β parameter. An additional aspect of fundamental importance lies in the simplicity of implementing this method in Matlab and the high degree of customization of the turbofan components without performing any manipulation of variables for the purpose of reducing the dimensionality of the problem, which would normally lead to a high condition number of the Jacobian matrix associated with the nonlinear turbofan system (and, thus, to significant error). In the proposed methodology, the component behavior can be modeled by analytical relationships and through the use of neural networks trained from component bench test data or data obtained from CFD simulations. Generalization of rotational component maps by feedforward neural networks leads to an average interpolation error up to around 1%, for all variables. The resulting nonlinear system is solved by a combined genetic algorithm and least squares algorithm approach, instead of the standard Newton's method. The turbofan numerical model turns out to be convergent, and results suggest that the trend in overall turbofan performance, as flight conditions change, is in agreement with the outputs of the GSP 12 software.

Keywords: turbofan; neural network; genetic algorithm; nonlinear modeling; least square method

Citation: Lombardo, G.; Lo Greco, P.; Benedetti, I. Turbofan Performance Estimation Using Neural Network Component Maps and Genetic Algorithm-Least Squares Solvers. *Int. J. Turbomach. Propuls. Power* **2024**, *9*, 27. https://doi.org/10.3390/ijtpp9030027

Academic Editor: Wolfgang Sanz

Received: 21 November 2023
Revised: 15 March 2024
Accepted: 22 March 2024
Published: 23 July 2024

1. Introduction

The need to develop more efficient and environmentally friendly engines is driven, on the one hand, by the trend of more stringent green regulations [1], with the EU as a pioneer in this field [2], and, on the other hand, by increasing fuel cost. Such a class of engines will use new, sophisticated, highly efficient components with operative behaviors that are significantly different from the standard components currently in use. In this scenario, new computational models, oriented to assist the design and test of new components, are of fundamental importance. Turbofan simulations have become more and more reliable and feasible over time due to the continuous increase in computing power and an increasingly multidisciplinary approach to the subjects involved, such as aerodynamics, acoustics, combustion, and materials [3]. In this paper, we present a new effective numerical method that allows reliable steady-state turbofan performance calculations in which components are operated, generalizing the component maps data using artificial neural networks (ANNs). One of the main differences between the proposed method and the one used in various commercial algorithms, such as GSP 12 [4] and NPSS [5], is the absence of the β parameter (GSP 12) or linear piecewise (NPSS) interpolation for the interpolation of fan, compressor, and turbine maps (Table 1). The β parameter is used, in most of the formulations, as an interpolation method in gas turbine simulation programs as it allows the identification of the operating point from only two parameters (β, N) instead of four (fixed-geometry component) or five (variable-geometry components) (N, W or WN, π, η, δ), thus avoiding problems of convergence and accuracy [6]. In the case of a compressor, the linear variant of beta interpolation involves first defining a beam, more or less dense depending on the degree of accuracy required, of parallel lines (each defined by a value of β) to the surge line and then calculating the points of intersection of each of these lines with each curve at constant corrected speed. Such an intersection point identifies, from the $beta/N$ coordinate pair, the corresponding values of corrected flow rate, pressure ratio, and efficiency (W, π, η). Finally, a 2D linear or higher-order interpolation is applied. In the actual implementation of the method, β interpolation was replaced by implicit equations derived by using neural networks, which were trained from four experimental datasets (associated with variables N, W or WN, π, η) divided into two sets of input data and two sets of output data. This technique can also be used to model, by ANN, any component, other than turbines or compressors, for which we have sufficient experimental or numerical data associated with the variables describing their operation. Modeling real components using maps requires a large amount of experimental data, which could have some economic and time impact: for this reason, in the present paper, we open the possibility of performing a limited number of experimental tests (from the GSP 12 map database), which are interpolated using piecewise functions, thus allowing the identification of intermediate operating points between two experimentally detected operating points. A further relevant aspect of such a turbofan model is the possibility of adding/removing components by simply adding/removing the respective equations describing its operation (see Section 2.1): a model being made up of a large number of variables (more than 40) could lead to a high condition number of the Jacobian matrix associated with the system of nonlinear equations (and, thus, a significant error). However, thanks to the scaling mode introduced (in Section 2.8), such errors are significantly contained and results are in good agreement with GSP 12 outputs. The proposed model uses an heuristic method (genetic algorithm, GA) combined with a gradient method (least squares, LSQ) in order to solve the turbofan problem whose components are represented by nonlinear equations or maps (Table 1). Such a combined approach allows the mitigation of the disadvantages of both individual methods: the LSQ needs an initial guess solution close to the actual solution and has a high convergence speed, while, in contrast, the GA does not need any initial guess, but only the definition of upper and lower boundary for the solution, and has a significantly lower convergence speed [7]. The combined approach, thus, makes it possible to obtain a robust iterative algorithm with good convergence and precision. These properties are even more important when the turbofan considered is assembled with unconventional/innovative components, which makes the choice of an appropriate initial guess solution even more uncertain, moving it significantly away from that of a more conventional turbofan. Thus, the turbofan modeling method developed here can serve as a basis to simulate advanced and innovative components, such as ultra-high-bypass fans, advanced burners, variable-geometry compressors, and cooled turbines.

Table 1. Comparison between turbine simulation algorithms.

Code	Solver	Interpolation Meth.
Proposed	GA + LSQ	ANN
GSP 12	Newton Raphson	β
NPSS	Newton Raphson	linear piecewise

2. Methods

In Section 2.1, we first introduce the nonlinear equations system used for modeling the turbofan and propose equations for the rotating components obtained by feedforward neural networks. We then choose the Rolls-Royce Trent 1000 as the reference turbofan to run the off-design performance simulation and give performances in the cruise condition, which are used as input parameters for the present model (Section 2.2). However, unlike the classic two-spool engine architecture, in this article, in order to reduce the complexity a little in the implementation of the code, we opted for a configuration in which the LP turbine drives the fan only (instead of fan + LP compressor). Next, we briefly present the theoretical concepts behind genetic algorithms, least square method (Section 2.3), feedforward neural networks (Section 2.4), and their actual use in Matlab via built-in solvers. In Section 2.5, we highlight the parameters of population size, max generation, and crossover fraction necessary for the operation of the genetic algorithm solver and provide, for both solvers, a common expression for the objective function (fun_{obj}) whose minimization allows the performance of the turbofan to be derived. In Section 2.6, we illustrate the reasons behind the choice of the feedforward neural network as the network for modeling the fan, compressor, and turbine, along with a brief explanation of the training methods, the choice of the number of layers/neurons, and the performance (as measured by the MSE, mean square error) of the networks. In Section 2.7, we introduce an overall algorithm for calculating turbofan performance, which uses the elements illustrated in the previous sections: this algorithm is divided, for ease of understanding, into three functional blocks, which perform well-defined tasks. Finally, in Section 2.8, we illustrate the scaling algorithm, which provides scaling factors for the rotating component maps and a guess solution to initialize the performance calculation algorithm.

2.1. Turbofan Nonlinear Equations System

The turbofan model (Figure 1), adopted as example, consists of a system of 44 nonlinear equations in 44 unknowns. These equations represent the thermodynamic model of each turbofan component in addition to other equations derived from matching relations regarding the power balances of the HP and LP shafts as well as the equality of rotation speeds of the fan and LP turbine, and the compressor and HP turbine.

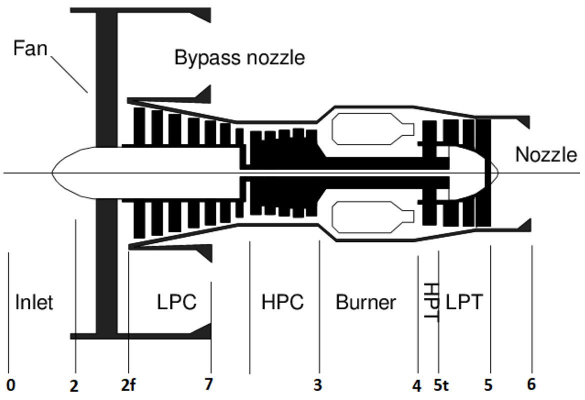

Figure 1. Turbofan main components [8].

The air intake was modeled as an adiabatic component with viscous losses defined by π_{int} (Equation (1)).

$$\begin{cases} T_{t2} = T_{t0} = T_0\left(1 + \dfrac{\gamma - 1}{2} Ma_0^2\right) \\ P_{t2} = \pi_{int} P_{t0} = \pi_{int} P_0 \left(1 + \dfrac{\gamma - 1}{2} Ma_0^2\right)^{\frac{\gamma}{\gamma - 1}} \end{cases} \tag{1}$$

Each rotating component is described by means of two implicit relations appropriately scaled by a factor $fact_{scal}$ (see Section 2.8), obtained from feedforward neural networks training, in particular Equations (2) and (3) for the fan and compressor [9], and Equations (4) and (5) for the turbine (where WN is the product of the correct flow rate and the correct rpm value).

$$\begin{cases} \dfrac{W}{fact_{scal_W}} = f_{1_c}\left(\dfrac{N}{fact_{scal_N}}, \dfrac{\pi}{fact_{scal_\pi}}\right) \\ \dfrac{\eta}{fact_{scal_\eta}} = f_{2_c}\left(\dfrac{N}{fact_{scal_N}}, \dfrac{\pi}{fact_{scal_\pi}}\right) \end{cases} \tag{2}$$

$$\begin{cases} \dfrac{W}{fact_{scal_W}} = f_{1_f}\left(\dfrac{N}{fact_{scal_N}}, \dfrac{\pi}{fact_{scal_\pi}}\right) \\ \dfrac{\eta}{fact_{scal_\eta}} = f_{2_f}\left(\dfrac{N}{fact_{scal_N}}, \dfrac{\pi}{fact_{scal_\pi}}\right) \end{cases} \tag{3}$$

$$\begin{cases} \dfrac{N}{fact_{scal_N}} = f_{1_{tHP}}\left(\dfrac{WN}{fact_{scal_{WN}}}, \dfrac{\pi}{fact_{scal_\pi}}\right) \\ \dfrac{\eta}{fact_{scal_\eta}} = f_{2_{tHP}}\left(\dfrac{WN}{fact_{scal_{WN}}}, \dfrac{\pi}{fact_{scal_\pi}}\right) \end{cases} \tag{4}$$

$$\begin{cases} \dfrac{N}{fact_{scal_N}} = f_{1_{tLP}}\left(\dfrac{WN}{fact_{scal_{WN}}}, \dfrac{\pi}{fact_{scal_\pi}}\right) \\ \dfrac{\eta}{fact_{scal_\eta}} = f_{2_{tLP}}\left(\dfrac{WN}{fact_{scal_{WN}}}, \dfrac{\pi}{fact_{scal_\pi}}\right) \end{cases} \tag{5}$$

The adiabatic modeling and the total pressure relations were also applied to the fan (Equation (6)), compressor (Equation (7)), HP turbine (Equation (8)), and LP turbine (Equation (9)).

$$\begin{cases} T_{t2f} = \left[1 + \dfrac{1}{\eta_f}\left(\pi_f^{\frac{\gamma-1}{\gamma}} - 1\right)\right] T_{t2} \\ P_{t2f} = \pi_f P_{t2} \end{cases} \tag{6}$$

$$\begin{cases} T_{t3} = \left[1 + \dfrac{1}{\eta_c}\left(\pi_c^{\frac{\gamma-1}{\gamma}} - 1\right)\right] T_{t2} \\ P_{t3} = \pi_c P_{t2} \end{cases} \tag{7}$$

$$\begin{cases} P_{t5t} = \dfrac{P_{t4}}{\pi_{t_{HP}}} \\ T_{t5t} = T_{t4}\left[1 - \eta_{t_{HP}}\left(1 - \pi_{t_{HP}}^{\frac{1-\gamma'}{\gamma'}}\right)\right] \end{cases} \tag{8}$$

$$\begin{cases} P_{t5} = \dfrac{P_{t5t}}{\pi_{t_{LP}}} \\ T_{t5} = T_{t5t}\left[1 - \eta_{t_{LP}}\left(1 - \pi_{t_{LP}}^{\frac{1-\gamma'}{\gamma'}}\right)\right] \end{cases} \tag{9}$$

Further equations relate to the compressor–HP turbine (Equation (10)) and fan–LP turbine mass flow balance (Equation (11)), the compressor–HP turbine (Equation (12)) and fan–LP turbine power balance (Equation (13)), the compressor-HP turbine and fan–LP turbine speed equality (Equations (14) and (15), respectively), and the definitions of WN_{tHP} (Equation (16)) and WN_{tLP} (Equation (17)).

$$\pi_c = \frac{(1+f)N_{tHP}}{\pi_b WN_{tHP}} W_c \sqrt{\frac{T_{t4}}{T_{t2f}}} \tag{10}$$

$$\pi_f = \frac{P_r\sqrt{T_{t2f}}}{P_{t2}W_c\sqrt{T_r}}\left[W_f\left(\frac{P_{t2}}{P_r}\right)\sqrt{\frac{T_r}{T_{t2}}} - \dot{m}_{byp}\right] \tag{11}$$

$$\pi_{tHP} = \left[1 - \frac{c_p(T_{t3} - T_{t2f})}{\eta_{mec}c'_p\eta_{tHP}(1+f)T_{t4}}\right]^{\dfrac{-\gamma'}{\gamma'-1}} \tag{12}$$

$$\pi_{tLP} = \left[1 - \frac{c_p\left(1 + \dfrac{\dot{m}_{byp}}{\dot{m}_{core}}\right)(T_{t2f} - T_{t2})}{\eta_{mec}c'_p\eta_{tLP}(1+f)T_{t5t}}\right]^{\dfrac{-\gamma'}{\gamma'-1}} \tag{13}$$

$$N_c = N_{tHP}\sqrt{\frac{T_{t4}}{T_{t2f}}} \tag{14}$$

$$N_f = N_{tLP}\sqrt{\frac{T_{t5t}}{T_{t2}}} \tag{15}$$

$$WN_{tHP} = \frac{\dot{m}_{core}(1+f)N_c\sqrt{\frac{T_{t2f}}{T_r}}}{\frac{P_{t4}}{P_r}} \tag{16}$$

$$WN_{tLP} = \frac{\dot{m}_{core}(1+f)N_{tLP}\sqrt{\frac{T_{t5t}}{T_r}}}{\frac{P_{t5t}}{P_r}} \tag{17}$$

The burner was modeled as a nonlinear component and defined by the quantities π_b and η_b (Equation (18)) (where b_1 and b_2 are characteristic parameters of the burner [10]). In addition to those relations, energy balance across the burner was considered (Equation (19)).

$$\begin{cases} P_{t4} = \pi_b P_{t3} \\ \pi_b = 1 - b_1\left(\dfrac{\dot{m}_{core}P_{ref}}{P_{t3}}\sqrt{\dfrac{T_{t3}}{T_{ref}}}\right)^2\left(f\dfrac{T_{ref}}{T_{t3}}\right)^2 \\ \eta_b = \eta_{b_d} - \dfrac{b_2}{\left(\dfrac{\dot{m}_{core}P_{ref}}{P_{t3}}\sqrt{\dfrac{T_{t3}}{T_{ref}}}\right)^2\left(f\dfrac{T_{ref}}{T_{t3}}\right)^2} \end{cases} \tag{18}$$

$$T_{t4} = \frac{f\eta_b H_i + c_p T_{t3}}{c'_p(1+f)} \tag{19}$$

The nozzle was modeled as an isentropic component where two regimes of operation can be distinguished depending on whether the discharge is sonic ($Ma_6 = 1$) or subsonic ($Ma_6 < 1$). Specifically, defining $r_{crit} = \left(\frac{\frac{\gamma'-1}{2}M_5^2+1}{1+\frac{\gamma'-1}{2}}\right)^{\frac{\gamma'}{\gamma'-1}}$ and $r = \frac{P_6}{P_{t5}}$ for the core nozzle, in the case in which $r > r_{crit}$, the outflow is subsonic and Equation (20) holds.

$$\begin{cases} r = \dfrac{P_6}{P_{t5}} \\ P_6 = P_0 \\ T_6 = T_5\left(\dfrac{P_6}{P_5}\right)^{\frac{\gamma'-1}{\gamma'}} \\ u_6 = \sqrt{\left(\dfrac{2\gamma'}{\gamma'-1}\right)R'T_5\left(1-\dfrac{P_6}{P_5}\right)^{\frac{\gamma'-1}{\gamma'}} + Ma_5^2\gamma' R' T_5} \\ Ma_6 = \dfrac{u_6}{\sqrt{\gamma' R' T_6}} \\ \dot{m}_{core} = \dfrac{\left(\frac{P_5}{R'T_5}\right)\left(\frac{P_6}{P_5}\right)^{\frac{1}{\gamma'}}\sqrt{\left(\frac{2\gamma'}{\gamma'-1}\right)R'T_5\left(1-\frac{P_6}{P_5}\right)^{\frac{\gamma'-1}{\gamma'}} + Ma_5^2\gamma' R' T_5 A_e}}{(1+f)} \end{cases} \tag{20}$$

Differently, in the case in which $r < r_{crit}$, the core efflux is sonic and Equation (20) holds with the only exception that $P_6 = P_{t5}r_{crit}$. Similar equations hold for the bypass nozzle. The remaining equations define the static properties upstream of the nozzles from their total properties (Equation (21) where $i = f, c$).

$$\begin{cases} P_i = \dfrac{P_{ti}}{\left[\left(1-\dfrac{\gamma-1}{2}Ma_i^2\right)\right]^{\left(\dfrac{\gamma}{\gamma-1}\right)}} \\ T_i = \dfrac{T_{ti}}{\left(1-\dfrac{\gamma-1}{2}Ma_i^2\right)} \end{cases} \tag{21}$$

Each of the 44 equations is first brought into its homogeneous form ($f(\vec{x}) = 0$) and then normalized by dividing the homogeneous function itself by the physical quantity it represents: for this purpose, the normalization of Equation (19) is given in Equation (22).

$$f_i(\vec{x}) = T_{t4} - \frac{f\eta_b H_i + c_p T_{t3}}{c'_p(1+f)} = 0 \quad \rightarrow \quad f_{norm_i}(\vec{x}) = \frac{\left(T_{t4} - \frac{f\eta_b H_i + c_p T_{t3}}{c'_p(1+f)}\right)}{T_{t4}} = 0 \tag{22}$$

The 44 normalized equations ($f_{norm_i}(\vec{x}) = 0$) are then brought together in a non-linear system (Equation (19)), associated with a vector of the unknowns $\vec{x}$ (which are essentially a collection of the physical quantities contained in the left-hand member of the non-normalized equations shown above).

$$\vec{F}_{norm}(\vec{x}) = \vec{0} \tag{23}$$

2.2. Reference Model

The engine model used as reference is the Rolls-Royce Trent 1000 and its design parameters, derived from the type certificate [11] and ICAO databank [12], served as the basis for the selected cruise quantities in Table 2a.

The knowledge of π_c, π_f, α, $\dot{m}_{tot}$, and F in cruising conditions allowed us to estimate, through an inverse preliminary performance study [13], the value of A_e and A_{byp}, which were 0.41 m^2 and 3.20 m^2, respectively; this was essentially performed by gradually varying the value of A_e until the value of thrust under cruise conditions (F_{cruise}) shown in Table 2a was obtained. Consequently, A_{byp} was also derived. In the absence of data provided by the manufacturer, component efficiencies in Table 2b and $f = \frac{1}{42}$ were assumed. In the following sections, the parameters b_1 and b_2 are assumed equal to 0, and η_{bd} is set equal to 0.96 in order to facilitate the comparison of the outputs with those of GSP 12; finally, $H_i = 43.26\frac{KJ}{Kg}$.

Table 2. (**a**) Trent 1000 performance in cruise. (**b**) Component efficiencies in cruise.

(**a**)	
Characteristic	**Value**
π_c	32.85
π_f	1.54
π_{int}	0.98
π_b	1.00
α	9.12
$\dot{m}_{tot}$	467.00 $\frac{\text{Kg}}{\text{s}}$
F_{cruise}	63.82 KN
N'_{tLP}	2683 rpm
N'_{tIP}	11,164 rpm
N'_{tHP}	11,164 rpm
(**b**)	
$\boldsymbol{\eta}$	**Value**
η_f	0.91
η_c	0.90
η_{tHP}	0.93
η_{tLP}	0.93
η_b	0.96

2.3. Genetic Algorithm and LSQ Solver

The need for the use of the genetic algorithm (GA) arises from the possibility that, for a given flight condition, the initial guess solution, provided by the preliminary performance analysis (Section 2.7), may be relatively far from the actual nonlinear system solution. A GA is a heuristic approach based on the theory of natural evolution, which has been the subject, for decades, of numerous studies concerning its use in the resolution of nonlinear systems of different complexity and size [14]. The application of the GA in the resolution of such systems passes through the definition of a fitness function (fun_{obj}) that allows the transformation of the problem itself into one of optimization. Many forms of fun_{obj} have been proposed. They range from the sum of the absolute values of the single functions [15] up to quadratic expressions [16]; however, there is no clear superiority of one over the other. In this method, the fitness function was chosen equal to $fun_{obj}(\vec{x}) = \sum_{i=1}^{n} f_{norm_i}(\vec{x})^2$, where $n = 44$ is the dimension of the solution vector $\vec{x}$ and f_{norm_i} is defined in Section 2.1. The genetic algorithm process starts by defining a fitness function and a genetic representation of the solution domain: usually, solutions are represented as an array of bits (0 and 1) in order to allow typical genetic operation, such as crossover and mutation, to be conducted more easily. Then, an initial population of solution (consisting of hundreds or thousands of candidate solutions) is generated, usually randomly, within given lower ($\vec{l_b}$) and upper boundaries ($\vec{u_b}$). Individual solutions are then selected through a fitness-based process, which essentially consists of choosing the solutions that best minimize fun_{obj}. The next step is to generate a pool of solutions (children) from those previously selected (parents): for this purpose, the crossover technique is used, which consists of breeding two parent solutions in order to produce one child solution. This operation is repeated several times in order to generate a desired number of child solutions. In order to make the genetic patrimony more heterogeneous, a portion of the population is subjected to the mutation operation, which consists of changing part of the genetic patrimony of the single solution (i.e., replacing 1 with 0 and vice versa). This operation reduces the risk of premature convergence, a situation in which the solution of fun_{obj} becomes stuck in a local minimum. Similarly, a fitness function was also defined for the LSQ solver (the same one chosen for the GA algorithm). Both the LSQ and GA solvers aim to find $\vec{x}$ that minimizes $fun_{obj}(\vec{x})$, which coincides with the solution of the system of nonlinear equations. The combined use of the two techniques allows us to mitigate the disadvantages associated with the single algorithms; in fact, unlike the methods based on the local gradient of a function (LSQ), the GA has the advantage of reducing the probability of incurring a local minimum, which is very common in the case of functions with multiple local minima, at the expense of a higher computational cost.

2.4. Feedforward Neural Networks

In the turbofan under consideration, rotating component maps were considered, whose behaviors were generalized through ANN. In this regard, a careful review of the literature suggested, for the compressor ANN modeling, the use of feedforward neural networks consisting of a small number of layers [17] and an empirically determined number of neurons such that the MSE is minimized: in particular, a study conducted on a dataset of 54 experimental points led to an average error on the π and W_c of a compressor of about 1% [18]. In the following implementation of this method, a feedforward architecture with 5 hidden layers is adopted, in which the connections between the various nodes do not form loops, or information flows unidirectionally from input x_i to output y_i. First, the input x_i is normalized, i.e., its mean is subtracted and is, in turn, divided by the standard deviation. Then, the normalized input (x_{norm_i}) reaches a specific neuron within the first hidden layer, which multiplies it by its weight (w_{ij}) and adds it to its bias (b_{ij}); this result is filtered by an activation function, chosen here as the hyperbolic tangent sigmoid function (tansig) (Equation (24)). This process is repeated for all the inner layers until the output layer is reached. The last layer output is a normalized value (y_{norm_i}) which needs to be denormalized in order to obtain the ANN output value (y_i).

$$\begin{cases} l_1 = tansig(w_1 x_{norm_i} + b_1) \\ l_i = tansig(w_i l_{i-1} + b_i) \\ y_{norm_i} = tansig(w_6 l_5 + b_6) \\ x_{norm_i} = \dfrac{x_i - x_{mean}}{\delta_{std_x}} \\ y_i = y_{norm_i}\delta_{std_y} + y_{mean} \end{cases} \tag{24}$$

Training was performed using Levenberg–Marquardt algorithm. Networks of all the rotating components were built in Matlab with the feedforwardnet command using 5 hidden layers of different size of neurons, each listed in Table 3: the choice of this parameter stems from the need to achieve good network performance without data overfitting phenomena. All networks have 2 inputs (N e π for compressor and fan, WN e π for turbines) and 2 outputs (W e η for compressor and fan, N e η for turbines).

Table 3. Neuron count per layer.

Component Map	Neurons
Fan	6
Compressor	6
HP/LP turbine	8

2.5. GA and LSQ Matlab Implementation

A GA algorithm was implemented in Matlab [19] and called by the "ga" function which required the definition of several parameters such as population size, maximum generation, and crossover fraction, as shown in Table 4: in this regard, as specified in [20], we decided to maintain a high crossrate value (0.9) in order to be able to reduce the population size.

Table 4. GA options.

Feature	Value
Population size	300
Max generation	30
Crossover frac.	0.9

In the same way, the LSQ solver was invoked in Matlab by the "lsqnonlin" command [21] and required the definition of additional parameters such as the starting point ($\vec{x}_0$), $\vec{u}_b$, and $\vec{l}_b$ and the max number of iterations (set to 500).

2.6. Modeling Rotating Components with ANN

The database containing the rotational component maps was imported from GSP 12. Data for the individual component were then subjected to piecewise linear interpolation, via the Matlab function "interp1", which allowed the size of the individual database to be increased, maintaining a faithful representation. This last operation was necessary because neural networks require a large amount of data for their proper training. Next, data normalization was performed after estimating the mean value and standard deviation of the dataset obtained by interpolation: normalized data were then used to train a feedforward neural network. The specific configuration of the neural network was defined using the "feedforwardnet" function, while the training was carried out using the "train" function and the network parameters in Table 3. During the network testing phase, mean squared error values for each network were estimated and are reported in Table 5a; in addition to these, in order to better characterize the error on the outputs, the mean percentage errors ($\bar{x}\,\%$), the maximum percentage errors ($err_{max}\,\%$), and their standard deviations ($\sigma\,\%$) were calculated (Table 5b).

Table 5. (**a**) Mean squared errors for turbofan rotating components (values refer to the test dataset). (**b**) Mean percentage errors $\bar{x}\,\%$, max percentage errors $err_{max}\,\%$, and standard deviations $\sigma\,\%$ for the compressor, fan, and turbine output variables (values refer to the test dataset).

(**a**)

MSE	Value
Fan	$2.03 \cdot 10^{-4}$
Compressor	$5.14 \cdot 10^{-4}$
Turbine	$2.87 \cdot 10^{-6}$

(**b**)

Error Measures	W_c	η_c	W_f	η_f	N_{tHP}	η_{tHP}
$\bar{x}\,\%$	0.82	0.08	0.59	0.05	0.04	0.03
$\sigma\,\%$	1.63	0.11	1.71	0.06	0.04	0.04
$err_{max}\,\%$	16.47	1.13	19.63	0.58	1.19	2.56

A careful analysis reveals that the highest percentage error points tend to be located, for both the fan and the compressor, along the stall regions and more at a low to medium N (Figure 2).

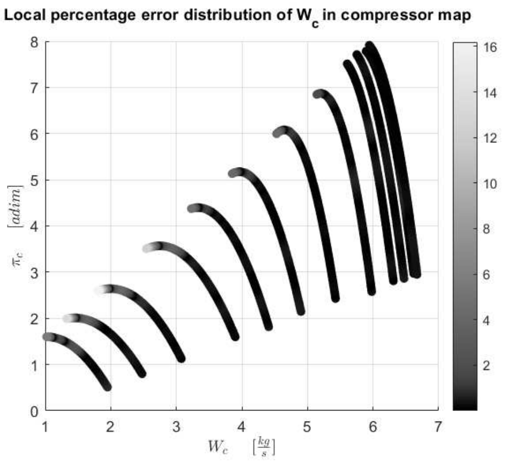

Figure 2. Compressor local error map for different N constant curves.

2.7. Hybrid LSQ—GA Solver

Using the elements described in the previous sections, an algorithm for calculating the performance of a turbofan was formulated: such performances are presented in the form of operating lines on component maps and overall performance as function of f (see Section 3.1). This algorithm is based on the interaction of 3 functional blocks (Figure 3), which perform the specific tasks listed below:

- Preliminary design calculation (Figure 4);
- Nonlinear solution calculation (Figure 5);
- Nozzles condition check (Figure 6).

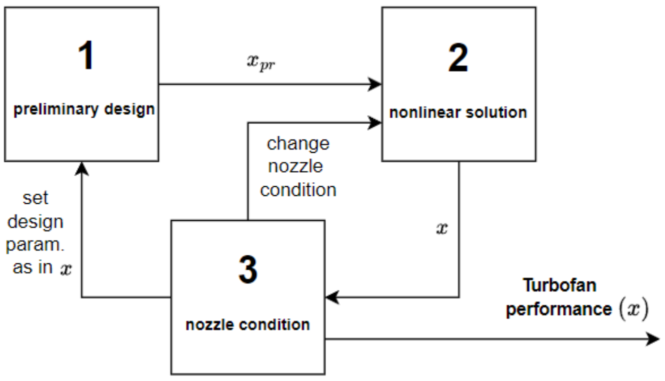

Figure 3. Interactions between the three parts of the algorithm.

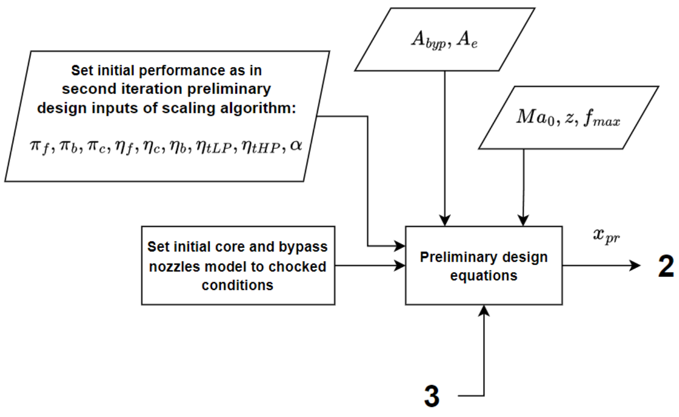
Set initial performance as in second iteration preliminary design inputs of scaling algorithm:
$\pi_f, \pi_b, \pi_c, \eta_f, \eta_c, \eta_b, \eta_{tLP}, \eta_{tHP}, \alpha$
A_{byp}, A_e
Ma_0, z, f_{max}
Set initial core and bypass nozzles model to chocked conditions
Preliminary design equations
x_{pr}
2
3

Figure 4. Preliminary design calculation in Figure 3.

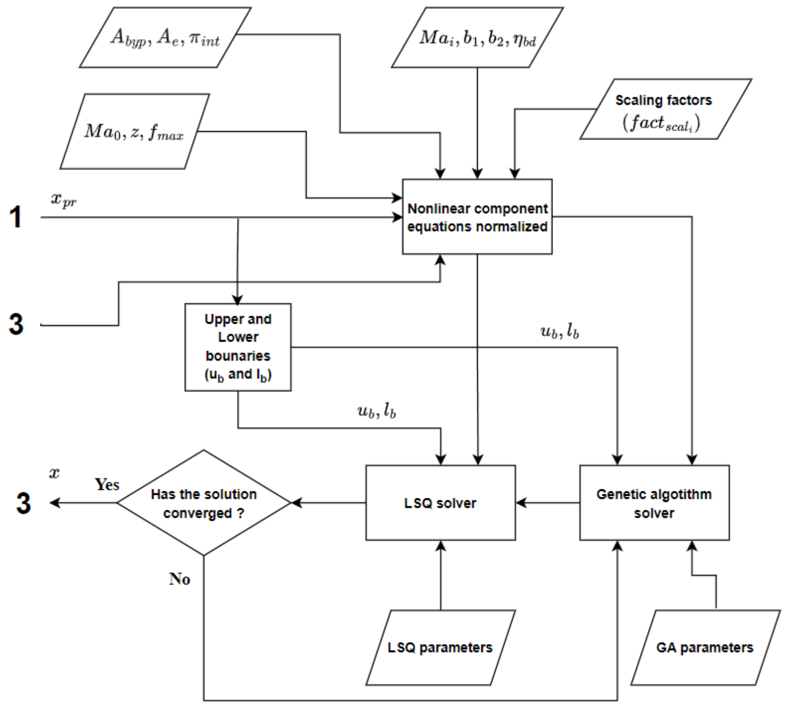
A_{byp}, A_e, π_{int}
$Ma_i, b_1, b_2, \eta_{bd}$
Scaling factors ($fact_{scal_i}$)
Ma_0, z, f_{max}
x_{pr}
1
Nonlinear component equations normalized
3
Upper and Lower bounaries (u_b and l_b)
u_b, l_b
u_b, l_b
x
Yes
3
Has the solution converged ?
LSQ solver
Genetic algotithm solver
No
LSQ parameters
GA parameters

Figure 5. Nonlinear solution calculation in Figure 3.

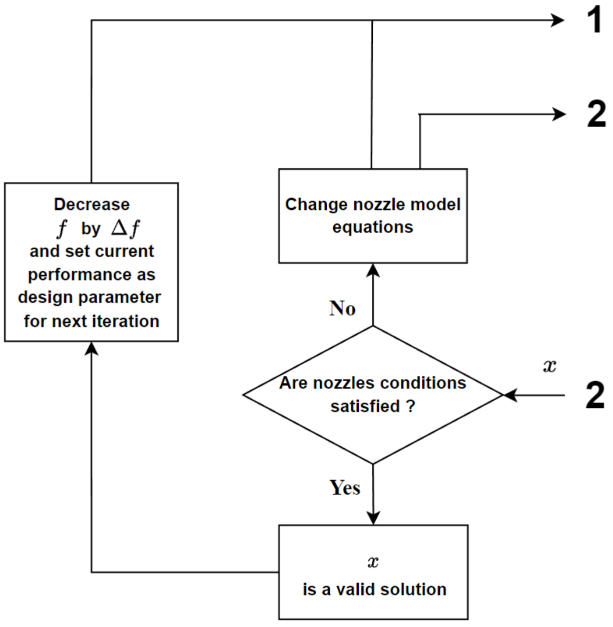
1
2
Decrease f by Δf and set current performance as design parameter for next iteration
Change nozzle model equations
No
Are nozzles conditions satisfied ?
x
2
Yes
x is a valid solution

Figure 6. Nozzles condition check in Figure 3.

Preliminary design calculation was realized according to [9,13]. Preliminary design calculation (Figure 4) begins by setting both core and bypass nozzle state to chocked on the first iteration, then the preliminary solution is computed starting from the second iteration preliminary design inputs (π_{int} ,π_f, π_b, π_c, η_f, η_c, η_b, $\eta_{t_{LP}}$, $\eta_{t_{HP}}$, α) of scaling algorithm (see Section 2.8 for more details), with the desired value of Ma_0, z, and an initial value of $f = f_{max}$; other input data are A_e and A_{byp}. Specifically, the f_{max} is chosen close to the f_{cruise} but greater than it, and, defining $\epsilon = f_{max} - f_{cruise}$, we have that ϵ increases as the altitude decreases (see Table 6 as an example). Preliminary design calculation output ($\vec{x}_{pr}$) is a 44-element vector which is essentially a rough estimation of vector $\vec{x}$ (turbofan performance): $\vec{x}_{pr}$ constitutes the input for the nonlinear solution calculation.

The nonlinear solution of Figure 5 starts by first calculating the lower ($\vec{l}_b$) and the upper ($\vec{u}_b$) boundary necessary to run the GA solver: this is simply conducted by setting $\vec{l}_b = 0.1 \cdot \vec{x}_{pr}$ and $\vec{u}_b = 10 \cdot \vec{x}_{pr}$. The nonlinear solution block calls the turbofan normalized equations system, described in Section 2.1, and requires as input, in addition to the selected flight condition (f, Ma_0 and z), $\vec{x}_{pr}$, Ma_i (Mach value of the flow for each engine section given in Table 7), A_{byp}, A_e, π_{int}, b_1, b_2, η_{bd}, and the scaling factors ($fact_{scal_i}$). In actuality, the value of Mach in the engine sections varies in off-design conditions; nevertheless, the variation is very small and has a negligible impact on performance. Scaling factors are determined using procedures explained in Section 2.8.

Table 6. Trent 1000 flight conditions for operating lines.

Flight Phase	z [m]	Ma	f_{max}	f_{min}	Δf
cruise	11,000	0.85	0.0263	0.0168	$1.6354 \cdot 10^{-4}$
climb 1	7000	0.6	0.0263	0.0168	$1.6354 \cdot 10^{-4}$
climb 2	5000	0.45	0.0270	0.0168	$1.7560 \cdot 10^{-4}$
take off	0	0.20	0.0312	0.0181	$2.4718 \cdot 10^{-4}$
static	0	0	0.0312	0.0181	$2.4718 \cdot 10^{-4}$

Table 7. Mach hypothesis at the component inlet section.

Fan	Compressor	Burner	LP Turbine	HP Turbine	Byp Nozzle	Core Inlet
0.40	0.40	0.40	0.30	0.40	0.40	0.40

The nonlinear equation system is solved through a combined approach that involves the use of the GA solver first and then the LSQ solver: now, if the LSQ solver fails to compute the solution ($\vec{x}$) with an error, measured by the squared norm of the residual (with i-th residual defined as $fun_i(\vec{x})$), higher than 10^{-20}, this solution is discarded and the GA solver recalculates again starting from $\vec{x}_{pr}$, generating a new population of solutions.

The nonlinear solution is then passed to the nozzle condition check block (Figure 6) in order to check whether the assumptions previously made on the nozzles were corrected. If positive (or $P_6 > P_0$ and $P_7 > P_0$), the algorithm decreases the value of f by a fixed Δf, sets the value of the design parameters (π_f, π_b, π_c, η_f, η_c, η_b, $\eta_{t_{LP}}$, $\eta_{t_{HP}}$, α) equal to their value found in the vector $\vec{x}$, and sends them to the preliminary design block. Thus, a new iteration can start from preliminary design calculation (Figure 4). If negative, the nozzle conditions are changed, imposing different core/bypass nozzle conditions (see Equation (20)), and the entire process is repeated with the same f, in particular:

- If $P_6 > P_0$ and $P_7 <= P_0$, chocked core/unchocked bypass nozzle condition are set;
- If $P_6 <= P_0$ and $P_7 > P_0$, unchocked core/chocked bypass nozzle condition are set;
- If $P_6 <= P_0$ and $P_7 <= P_0$, unchocked core/unchocked bypass nozzle condition are set.

Because of what was stated above, following each iteration from the successful outcome, the value of f is decremented by Δf. In the case where the objective is the determination of the operating line, the algorithm continues until a value of f corresponding to low engine speeds is reached ($f = f_{min}$), while in the case where performance for a specific flight condition (Ma_0, z, f) is of interest, it will be necessary to choose very small values of Δf such that, after a certain number of iterations, the desired value of f is intercepted. The procedure shown in this section is used in Section 3.1 to generate the operating lines on the rotating component maps as well as the performance values at certain flight conditions. Table 8 summarizes the fixed parameters and variables illustrated in the above algorithm.

Table 8. Summary of fixed and variable parameters.

Type	Parameter/Variable List
Fixed parameter	$A_{byp}, A_e, M_0, z, f_{max}, f_{min}, \Delta f, \pi_{int}, Ma_i, b_1, b_2, \eta_{bd}, fact_{scal_i}$
Variable	All other quantities

2.8. Map Scaling

The database for constructing the maps used in this paper was exported from GSP 12 software. However, the use of these maps was subject to a scaling operation since generic maps of compressors and turbines are used in the present paper, which must be appropriately scaled to ensure a given design operating point position on maps. Map scaling consists of determining the scaling factors ($fact_{scal}$ (see Equation (2)) for Equation (5)) to be applied to each of the 4 variables of the fan/compressor (N, W, π, and η) or turbines (N, WN, π, and η). A comprehensive description of the standard scaling procedure of rotary component maps performed by GasTurb 13 software can be found in [22]; the same modalities apply to GSP 12. The scaling operation proposed in this software consists of multiplying each map by a correction factor, calculated as the ratio between the value calculated in the preliminary design phase (var_{pr}) and that provided by a reference point on the map (var_{map}) (Equation (25) where var are map variables).

$$fact_{scal} = \frac{var_{pr}}{var_{map}} \tag{25}$$

However, a slightly different scaling method is used in this work. Such use is made necessary by the need to partially compensate for numerical inaccuracies caused by moderate conditioning of the nonlinear system. The present scaling algorithm basically consists of the algorithm proposed in Section 2.7 plus some modifications, which are detailed below: we will explain the scaling operation of π_f, and similar steps are performed simultaneously for all the remaining design parameters. The scaling algorithm is iterated 2 times. The first iteration ($i = 1$) begins by setting inputs as in Equation (26), where $\pi_{f\,guess}$ is first assumed as in Table 2a,b, and k_i is initially set to 1.

$$\pi_{f\,pr_{(i)}} = \frac{\pi_{f\,guess}}{k_{(i)}} \tag{26}$$

The preliminary design solution ($\pi_{f\,pr_{(i)}}$) is then computed and $fact_{scal_{(i)}}$ is derived from Equation (27) as a function of $k_{(i)}$, $\pi_{f\,pr_{(i)}}$ and the position on the original map ($\pi_{f\,map}$).

$$fact_{scal_{(i)}} = \frac{\pi_{f\,pr_{(i)}}}{\pi_{f\,map}} \cdot k_{(i)} \tag{27}$$

Using $\pi_{f\,pr_{(i)}}$ and $fact_{scal_{(i)}}$, the nonlinear solution ($\pi_{f_{(i)}}$) is then calculated, which is used to determine the parameter $k_{(i+1)}$ (Equation (28)) and a new iteration begins.

$$k_{(i+1)} = k_{(i)} \cdot \frac{\pi_{f_{(i)}}}{\frac{\pi_{f\,guess}}{k_{(i)}}} \tag{28}$$

This process ends at $i = 2$ and the same scaling procedure is performed simultaneously for all other design parameters. The scaling process is considered to be successful only if the results of the second iteration ($i = 2$) of the scaling process provide performance very close to that shown in Table 2a,b; if not, it would be necessary to slightly vary the guess design parameters ($\pi_{f\,guess}$, etc.) by increasing or decreasing that value depending on whether one obtained, in the second iteration ($i = 2$), a value smaller or larger than the value shown in Table 2a,b. This variation procedure is performed manually, but the authors reserve the possibility to create, in a future article, a routine that automates this process as well. The scaling factor obtained in the second iteration ($fact_{scal_2}$) is used as definitive value of the scaling factor, while the second preliminary design input (x_{pr_2}) is used to initialize the solution of the turbofan simulation in Section 2.7, as shown in the diagram in Figure 7. This scaling procedure makes it possible to compensate for most of the errors introduced in the computational phase of the solution of the nonlinear system, which arise from a moderate condition number of the Jacobian matrix associated with the system of equations.

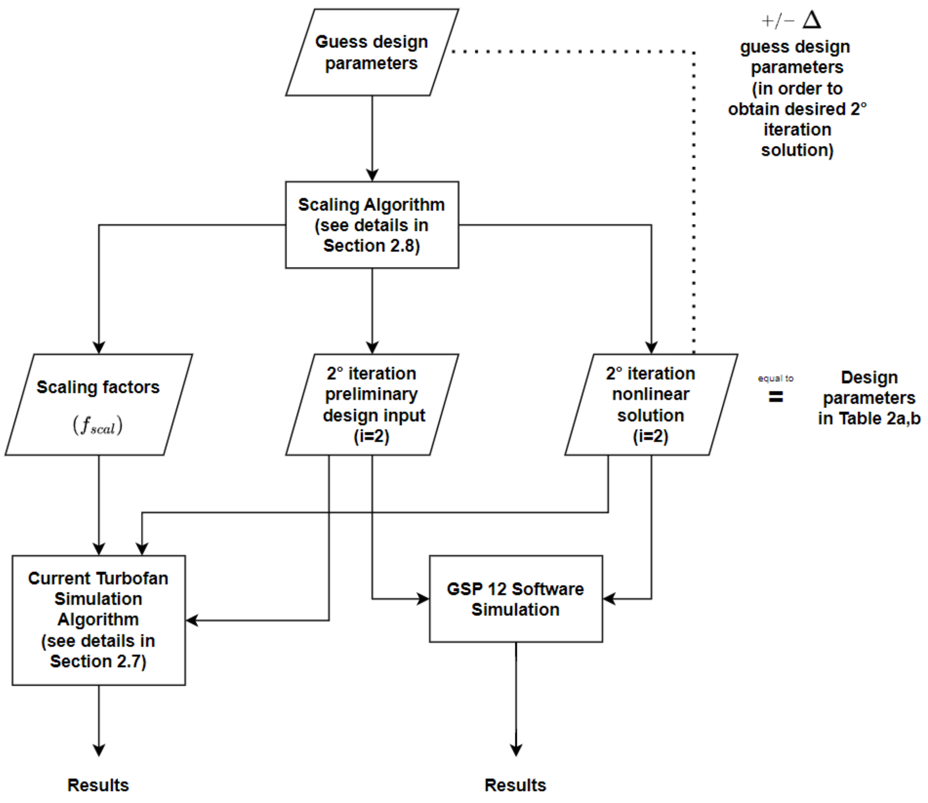
Guess design parameters
+/− Δ guess design parameters (in order to obtain desired 2° iteration solution)
Scaling Algorithm (see details in Section 2.8)
Scaling factors (f_{scal})
2° iteration preliminary design input (i=2)
2° iteration nonlinear solution (i=2)
equal to
=
Design parameters in Table 2a,b
Current Turbofan Simulation Algorithm (see details in Section 2.7)
GSP 12 Software Simulation
Results
Results

Figure 7. Overall turbofan simulation algorithm including scaling block.

3. Results

In this section, results of the numerical simulations are reported in the form of operating points on the rotating component maps and overall turbofan performance (F and $\dot{m}_{fuel}$) as function of f.

3.1. Performance Evaluation for Relevant Flight Phases and Comparison with GSP 12

Simulations are performed for the different flight phases: they vary from cruising condition to static operation at sea level (Table 6).

These results are reported both in the form of steady-state operating points on the various maps (Figures 8–11) and in the form of graphs for the overall performance (Figures 12 and 13) as f varies. In particular, current algorithm validation is obtained through comparison with simulations carried out with GSP 12 software [4], which is provided with the same data as in Table 2a,b. Validation simulations are carried out for values of z and Ma_0 reported in Table 6 in correspondence with four distinct values of f in the interval between f_{max} and f_{min} (Table 9): the current algorithm operating points are identified in Figures 8–11 using circle markers, while GSP 12 points are identified using square markers.

Table 9. GSP points flight conditions.

Flight Phase	f_1	f_2	f_3	f_4
Cruise	0.0263	0.0232	0.0199	0.0168
Climb 1	0.0263	0.0232	0.0199	0.0168
Climb 2	0.0270	0.0237	0.0202	0.0168
Take-off	0.0312	0.0266	0.0216	0.0183
Static	0.0312	0.0266	0.0216	0.0182

More extensive simulations aimed at identifying the operating line are performed for the same conditions in Table 6 using around 48 to 54 values (depending on the flight phases) of f between f_{max} and f_{min} spaced apart from each other by Δf (Table 6). Figures 8–11 show the variation of π_f, π_c, π_{tLP}, and π_{tHP} as a function of the respective corrected mass flow rates W or WN (in the case of turbines), as f varies, starting from the right (where $f = f_{max}$) to the left ($f = f_{min}$), forming the operating line. It can be seen how the resulting operating line is in agreement, with a fair margin of error, with the trend resulting from the data provided by GSP 12.

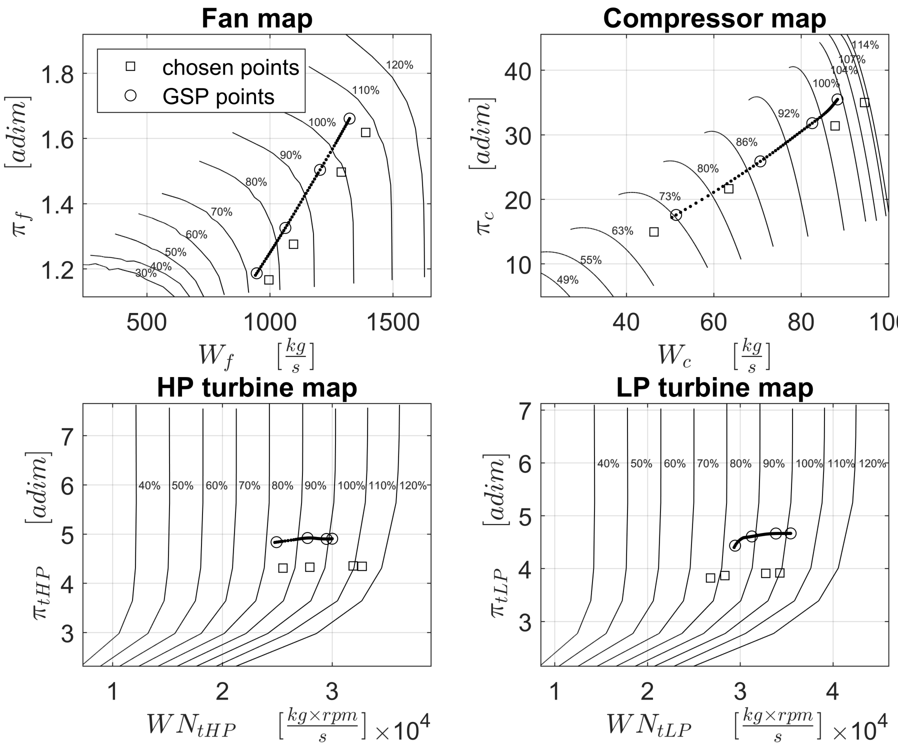

Figure 8. Operating lines on maps for z = 11,000 m and $Ma_0 = 0.85$ as f decreases from 0.0263 to 0.0168.

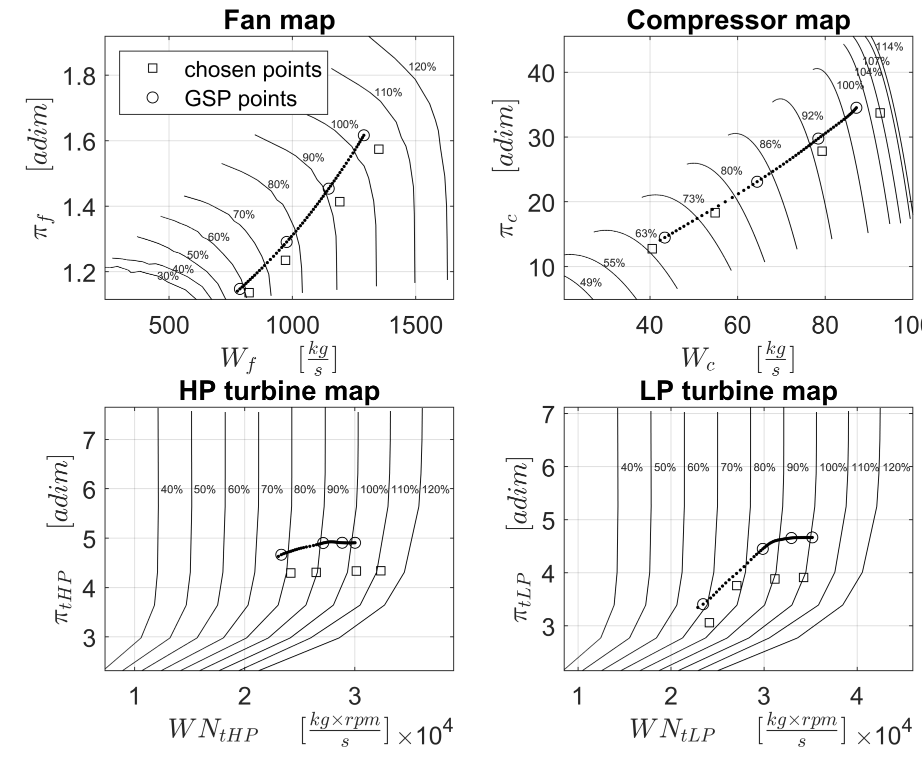

Figure 9. Operating lines on maps for $z = 5000$ m and $Ma_0 = 0.45$ as f decreases from 0.0270 to 0.0168.

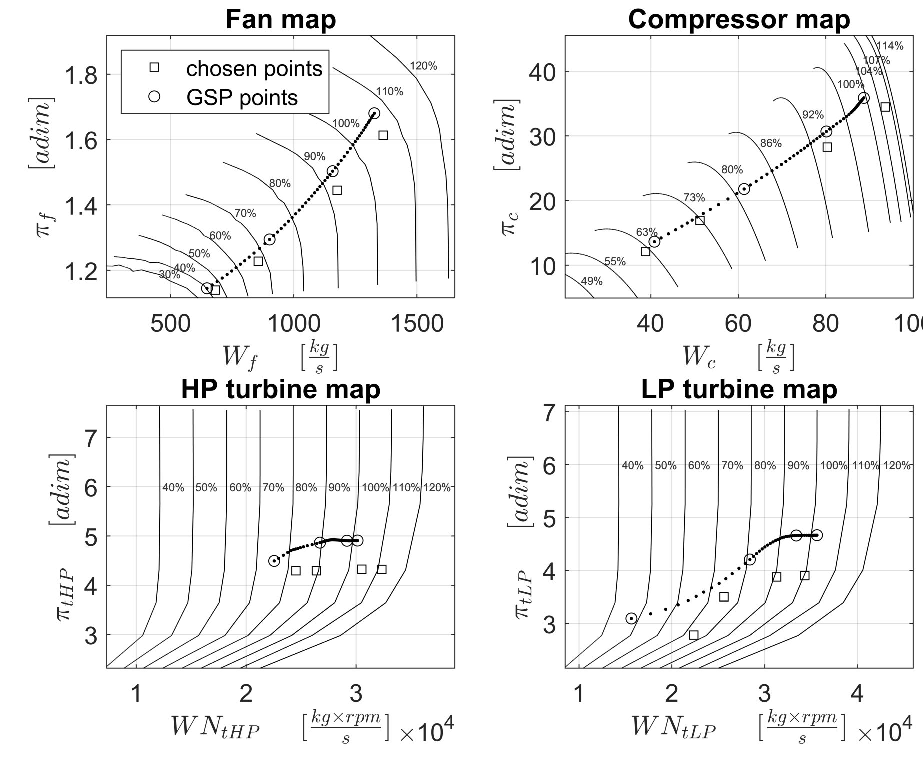

Figure 10. Operating lines on maps for $z = 0$ m and $Ma_0 = 0.20$ as f decreases from 0.0312 to 0.0183.

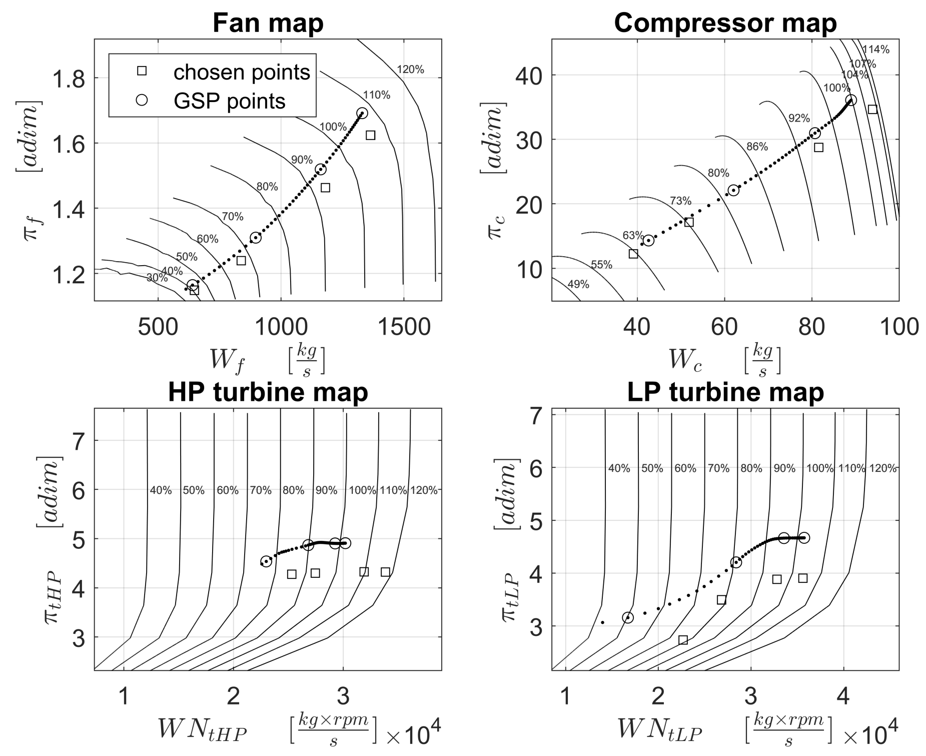

Figure 11. Operating lines on maps for $z = 0$ m and $Ma_0 = 0$ as f decreases from 0.0312 to 0.0182.

Similar considerations also apply to Figures 12 and 13 (where lines are used for simulation results while GSP 12 points are depicted by symbols), which show a nearly linear trend of $\dot{m}_{fuel}$ and F as function of f, in very good agreement with the output of GSP 12.

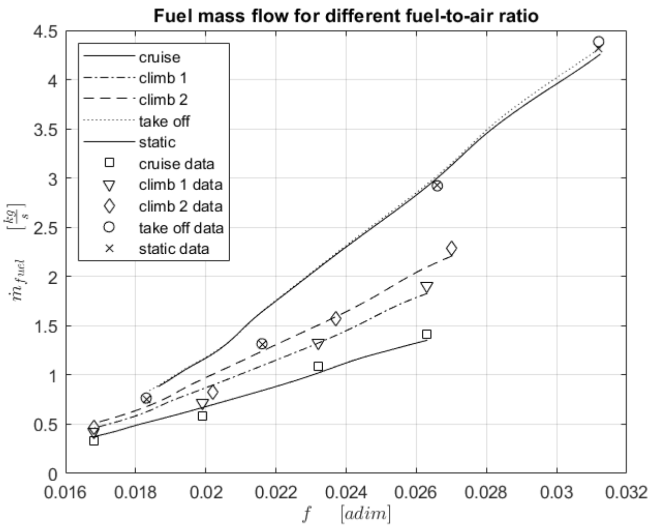

Figure 12. Trend of $\dot{m}_{fuel}$ as f decreases for flight conditions shown in Table 6.

Finally, a comparison of computational times between the present algorithm and GSP 12 was performed at three distinct flight phases and for a fixed number of points (20) between f_{max} and f_{min} (Table 10). The results show that there is limited difference in computational time; nevertheless, it is necessary to take into account that GSP 12 also computes solutions of a fair amount of intermediate points.

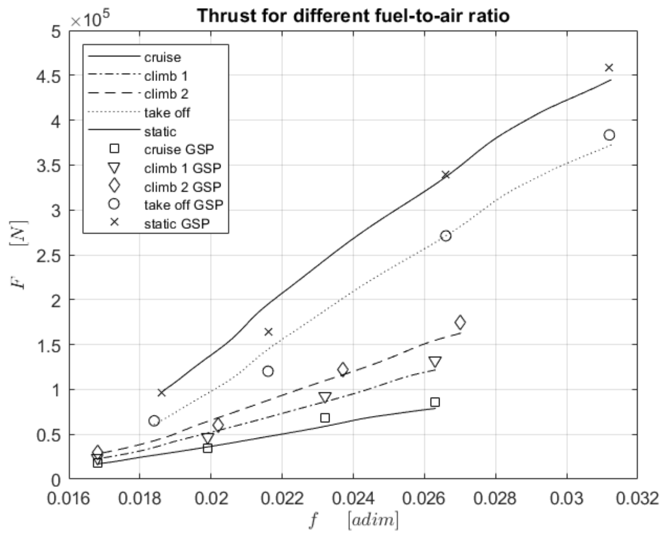

Figure 13. Trend of F as f decreases for flight conditions shown in Table 6.

Table 10. Comparison of computational time.

Flight Phase	f_{max}	f_{min}	n°Points	GSP 12 Time $[s]$	Actual Algo Time $[s]$
Cruise	0.0263	0.0189	20	50	56
Climb 2	0.0270	0.0189	20	56	68
Static	0.0312	0.0189	20	57	86

4. Conclusions

The present work demonstrates how a steady-state nonlinear turbofan model with components represented by nonlinear equations or implicit relations derived from neural network training can be effectively solved using GA and LSQ algorithms. A major advantage over current commercial codes (GSP 12, GasTurb) is the ease of implementation in Matlab and the ability to implement various components by simply modifying the system equations. This simplicity of implementation is made possible by the use of a scaling mode that can compensate for most of the errors caused by the potential ill-conditioning of a large nonlinear system. The results obtained suggest that trends in F and $\dot{m}_{fuel}$, as flight conditions change, are consistent with the outputs obtained from GSP 12 software. A small contribution to the error can also be attributed to differences, difficult to detect, in turbofan modeling between the present algorithm and that of GSP 12. The average interpolation error for all variables of the maps and their standard deviations do not exceed 1% and 1.5%, respectively. A relevant aspect of the reduction in the accuracy of the simulations could be determined by the high local maximum percentage error, which is around 16.50% for the fan and compressor maps (but turns out to be about 1% for the turbine map); however, for the present simulations, this aspect is negligible because the operating points do not fall within these regions. More investigation into the interaction of the various errors is needed in order to solve these problems; however, it emerges unequivocally that the feasibility of such an approach is related to finding the ANN architecture and the structure of the data that minimize both the MSE and the maximum local error. This work also demonstrates the effectiveness and feasibility of using the combined GA/LSQ approach for solving nonlinear systems of even considerable dimensionality. Future studies will focus on improving the accuracy and method reliability, extending it to nonstationary off-design performance.

Author Contributions: Conceptualization, G.L. and P.L.G.; methodology, G.L. and P.L.G.; software, P.L.G.; validation, P.L.G. and G.L.; formal analysis, G.L.; investigation, G.L. and P.L.G.; resources, G.L.; data curation, P.L.G.; writing—original draft preparation, G.L. and P.L.G.; writing—review and

editing, I.B.; visualization, P.L.G.; supervision, G.L. and I.B.; project administration, G.L. and I.B. All authors have read and agreed to the published version of the manuscript.

Funding: G. Lombardo and I. Benedetti acknowledge the support of the Department of Engineering of the University of Palermo through the grant Premio gruppo di ricerca 2020. P.Lo Greco acknowledges the support of the PON Ricerca e Innovazione 2014–2020 Azione IV.5 "Dottorati su Tematiche Green" (XXXVII ciclo). The APC was funded by PON Ricerca e Innovazione 2014–2020 Azione IV.5 "Dottorati su Tematiche Green" (XXXVII ciclo).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Rotating component maps are available within GSP 12 installation software. Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Nomenclature

A	=	nozzle exhaust area $[\mathrm{m}^2]$
c_p	=	air specific heat at constant pressure $\left[\frac{\mathrm{J}}{\mathrm{K \cdot Kg}}\right]$
c'_p	=	combustion gasses specific heat at constant pressure $\left[\frac{\mathrm{J}}{\mathrm{K \cdot Kg}}\right]$
f	=	fuel flow to core airflow ratio [adim]
F	=	thrust [N]
$fact_{scal}$	=	map scaling factor [adim]
H_i	=	fuel lower heating value $\left[\frac{\mathrm{J}}{\mathrm{Kg}}\right]$
$\dot{m}$	=	mass flow rate $\left[\frac{\mathrm{Kg}}{\mathrm{s}}\right]$
Ma_0	=	flight Mach number [adim]
N	=	corrected rpm [rpm]
N'	=	shaft rpm [rpm]
P_{ti}	=	stagnation pressure at section i [Pa]
R'	=	combustion gasses specific constant $\left[\frac{\mathrm{J}}{\mathrm{K \cdot Kg}}\right]$
T_{ti}	=	stagnation temperature at section i [K]
W	=	corrected mass flow $\left[\frac{\mathrm{Kg}}{\mathrm{s}}\right]$
k	=	iteration variable in scaling factor algorithm [adim]
z	=	altitude [m]
α	=	bypass ratio [adim]
γ	=	ratio between c_p and c_v of air [adim]
γ'	=	ratio between c_p and c_v of combustion gasses [adim]
η	=	adiabatic efficiency [adim]
η_{mech}	=	mechanichal efficiency [adim]
π	=	total pressure ratio [adim]
τ	=	total temperature ratio [adim]

Subscript

byp	=	bypass nozzle
c	=	compressor
$core$	=	core flow
e	=	core nozzle
f	=	fan
HP	=	high pressure
LP	=	low pressure
map	=	value referred to unscaled/original map

pr	=	value referred to preliminary design phase
tHP	=	HP turbine
tLP	=	LP turbine
tot	=	intake flow

References

1. Kuklinska, K.; Wolska, L.; Namiesnik, J. Air quality policy in the U.S. and the EU—A review. *Atmos. Pollut. Res.* **2015**, *6*, 129–137. [CrossRef]
2. ACARE. Fly the Green Deal, Europe's Vision for Sustainable Aviation. 2022. Available online: https://www.acare4europe.org/wp-content/uploads/2022/06/20220815_Fly-the-green-deal_LR-1.pdf (accessed on 21 November 2023).
3. Evans, A.; Follen, G.; Naiman, C.; Lopez, I. Numerical Propulsion System Simulation's National Cycle Program. In Proceedings of the 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Cleveland, OH, USA, 13–15 July 1998. [CrossRef]
4. Visser, W. Generic Analysis Methods for Gas Turbine Engine Performance: The Development of the Gas Turbine Simulation Program GSP. Ph.D. Thesis, TU Delft, The Netherlands, 2015. [CrossRef]
5. Ohio Aerospace Institute. NPSS User Guide. 2012. Available online: http://www.wolverine-ventures.com/userdocs/version241_docs/UserGuide.pdf (accessed on 21 November 2023).
6. Mist'e, G.A.; Benini, E. Improvements in Off Design Aeroengine Performance Prediction Using Analytic Compressor Map Interpolation. *Int. J. Turbo Jet Engine* **2012**, *29*, 69–77. [CrossRef]
7. Liu, G.; Han, X.; Lam, K.Y. A combined genetic algorithm and nonlinear least squares method for material characterization using elastic waves. *Comput. Methods Appl. Mech. Eng.* **2002**, *191*, 1909–1921. [CrossRef]
8. McCusker, J.R.; Danai, K. Selection of Outputs for Gas-Turbine Engines by Parameter Signatures. *ASME Dyn. Syst. Control. Conf.* **2010**, *2*, 239–246. [CrossRef]
9. Kerrebrock, J.L. *Aircraft Engines and Gas Turbines*; MIT Press: Cambridge, MA, USA, 1992.
10. Flack, R.D. *Fundamental of Jet Propulsion with Applications*; Cambridge University Press: New York, NY, USA, 2005; Chapter 9.
11. EASA. Type Certificate Data Sheet for Trent 1000 Engines Series. 2022. Available online: https://www.easa.europa.eu/en/downloads/7733/en (accessed on 21 November 2023).
12. ICAO. ICAO Aircraft Engine Emissions Databank. 2021. Available online: https://www.easa.europa.eu/en/domains/environment/icao-aircraft-engine-emissions-databank (accessed on 21 November 2023).
13. Asoliman, I.; Ehab, M.; Mahrous, A.; El-Sayed, A.; Emeara, M. Performance Analysis of High Bypass Turbofan Engine Trent 1000-A*. In Proceedings of the 3rd IUGRC International Undergraduate Research Conference, Cairo, Egypt, 30 July–1 August 2018.
14. Pourrajabian, A.; Ebrahimi, R.; Mirzaei, M.; Shams, M. Applying genetic algorithms for solving nonlinear algebraic equations. *Appl. Math. Comput.* **2013**, *219*, 11483–11494. [CrossRef]
15. Mangla, C.; Ahmad, M.; Uddin, M. Optimization of complex nonlinear systems using genetic algorithm. *Int. J. Inf. Tecnol.* **2021**, *13*, 1913–1925. [CrossRef]
16. Wu, Z.; Kang, L. A fast and elitist parallel evolutionary algorithm for solving systems of non-linear equations. In Proceedings of the 2003 Congress on Evolutionary Computation, Canberra, Australia, 8–12 December 2003; Volume 2, pp. 1026–1028. [CrossRef]
17. Yu, Y.; Chen, L.; Sun, F.; Wu, C. Neural-network based analysis and prediction of a compressor's characteristic performance map. *Appl. Energy* **2007**, *84*, 48–55. [CrossRef]
18. Gholamrezaei, M.; Ghorbanian, K. Compressor map generation using a feed-forward neural network and rig data. *Proc. Inst. Mech. Eng. A—J. Power* **2010**, *224*, 97–108. [CrossRef]
19. Matlab Manual. GA Function. 2022. Available online: https://www.mathworks.com/help/gads/ga.html (accessed on 21 November 2023).
20. Rexhepi, A.; Maxhuni, A.; Dika, A. Analysis of the impact of parameters values on the Genetic Algorithm for TSP. *Int. J. Comput. Sci. Issues* **2013**, *10*, 158–164.
21. Matlab Manual. lsqnonlin Function. 2022. Available online: https://www.mathworks.com/help/optim/ug/lsqnonlin.html (accessed on 21 November 2023).
22. GasTurb 13, Design and Off-Design Performance of Gas Turbines. 2018. Available online: https://www.gasturb.com/Downloads/Manuals/GasTurb13.pdf (accessed on 21 November 2023).