A highly flexible approach on the steady-state analysis of innovative micro gas turbine cycles

Thomas Krummrein
German Aerospace Center (DLR)
Institute of Combustion Technology
Pfaffenwaldring 38-40
70569 Stuttgart
Germany
Email: Thomas.Krummrein@dlr.de

Martin Henke
German Aerospace Center (DLR)
Institute of Combustion Technology
Pfaffenwaldring 38-40
70569 Stuttgart
Germany
Email: Martin.Henke@dlr.de

Peter Kutne
German Aerospace Center (DLR)
Institute of Combustion Technology
Pfaffenwaldring 38-40
70569 Stuttgart
Germany
Email: Peter.Kutne@dlr.de

Steady state simulations are an important method to investigate thermodynamic processes. This is especially true for innovative micro gas turbine (MGT) based cycles as the complexity of such systems grows. Therefore, steady state simulation tools are required which ensure large flexibility and computation robustness. As the increased system complexity result often in more extensive parameter studies also a fast computation speed is required.

While a number of steady state simulation tools for micro gas turbine based systems are described and applied in literature, the solving process of such tools is rarely explained. However, this solving process is crucial to achieve a robust and fast computation within a physically meaningful range. Therefore, a new solver routine for a steady state simulation tool developed at the DLR Institute of Combustion Technology is presented in detail in this paper.

The solver routine is based on Broyden's method. It considers boundaries during the solving process to maintain a physically and technically meaningful solution process. Supplementary methods are implemented and described which improve the computation robustness and speed.

Furthermore, some features of the resulting steady state simulation tool are presented. Exemplary applications of a hybrid power plant, an inverted Brayton cycle and an aircraft auxiliary power unit show the capabilities of the presented solver routine and the steady state simulation tool. It is shown that the new solver routine is superior to the standard Simulink algebraic solver in terms of system evaluation and robustness for the given applications.

Nomenclature

Symbols

A	Cross-sectional area [m^2]
$\boldsymbol{B}$	Broyden matrix
$\boldsymbol{D}$	Diagonal (Scaling) matrix
D	Diameter [m]
$\boldsymbol{J}$	Jacobian matrix
T	Temperature [K]
$\dot{V}$	Volume flow [m^3/s]
c	Loss factor
c_f	Darcy friction factor
$\boldsymbol{f}$	Residual function
i	Vector element index
k	Iteration index
$\boldsymbol{lb}$	Lower boundaries
$\boldsymbol{ub}$	Upper boundaries
$\dot{m}$	Mass flow [kg/s]
n	Number of solver variables and residuals
$\boldsymbol{x}$	Solver variables
w	Inlet weight factor
Δ	Difference
α	Line search step length
ρ	Density [kg/m^3]

Subscripts

f	Residuals
lim	Limitation
LS	Line Search
Newton	Newton step
QP	Quadratic programming

Acronyms

APU	Auxiliary power unit
IBC	Inverted Brayton cycle
HyPP	Hybrid power plant
MGT	Micro gas turbine
MGTS^3	Micro gas turbine steady state simulation tool
SOFC	Solid oxide fuel cell

1 INTRODUCTION

Micro gas turbines (MGT) are a promising technology for a wide variety of stationary and mobile applications. Today, they are mostly used in the classical Brayton cycle configuration. However, MGT are also the base of many emerging novel cycles. Key examples are the inverted Brayton cycle [1, 2], multi staged cycles and hybrid power plants which combine a MGT with a high temperature fuel cell [3, 4, 5]. The complexity of these cycles is often significantly higher compared to a classical Brayton cycle. Even more so, if the concepts are combined or extended with exhaust gas recirculation, water injection or other modifications.

Experimental investigations of such cycles are limited by time and resources. Therefore, steady states calculation with process simulation tools provides a powerful method to investigate, develop and optimize such systems. However, the simulation tools must overcome the increased complexity to ensure a fast and robust calculation. This is especially true because more complex systems have more degrees of freedom and hence often require extensive parameter studies.

To determine steady state solutions of systems it is possible to simulate the system with a transient simulation tool until a steady state is reached. An example for such a tool is the *TRANSEO* code [6, 7]. However, transient simulations compute more information than needed for steady state analysis in general. Hence, they cannot reach minimal calculation time. This is overcome with the direct determination of a steady state by an iterative solving process. In this case unknown process quantities are initialized with guess values and then changed until deviations to the steady state vanish. Such methods are used in several simulation tools, for example in the *Gas Turbine Simulation Program (GSP)* which is described in detail in [8]. Also more general simulations tools like *Aspen Plus* [9] or *Engineering Equation Solver(EES)* [10] use this method.

The solving process of such tools is the crucial factor which determines how fast and robust a steady state solution can be found and how flexible the simulation can be extended. However, in literature the solving process is typically hardly discussed in detail. Visser describes the method for the GSP solution process [8] which is based on a simplified Newton's method. In the documentation of EES [10] an overview of the solution method is given. It organizes all equations in blocks which are solved using Newtons's method while considering given boundaries of the variables.

At the DLR Institute of Combustion Technology a steady state simulation tool is developed which is suitable to perform fast and robust analysis of complex MGT based cycles. The tool was introduced by Panne et al. [3]. It was further developed, expanded and also validated with experimental data from a Turbec T100 MGT by Henke et al. [1]. The tool was recently revised to the *micro gas turbine steady state simulator (MGTS^3)*. This paper works out the requirements which are advantageous for a cycle simulation tool to support the investigation of innovative MGT based cycles. Based on these requirements the so called *global solver routine* was developed. It greatly enhances the capabilities of earlier versions of the simulator and is described in detail in the paper.

Also a general overview of the MGTS^3 simulation tool and its abilities is given. Finally, application examples show the advantages and capabilities of the presented solver algorithm.

2 STEADY STATE PROCESS SIMULATION TOOLS

With a steady state simulation tool it is possible to create a *system model*, this is a representation of a real world system with equations. The tool assists the describing of the systems, for example with predefined sub-models of system components. Also the tool solves the model equations. In a specific implementation of such a model these equations are solved in a certain order which defines their dependencies among themselves. In addition, this results in a classification of the used variables. Here four types are distinguished: *User given parameters* are all values which are specified by the user. *Interim variables* can be calculated directly from given equations using only variables which are already known in the solving process. However, typically some equations depends on each other or they can be only formulated in an implicit form. Therefore *solver variables* $\boldsymbol{x}$ are needed which are determined somehow by solving procedures of the simulation tool. To do this the system must define the same numbers of *residuals* $\boldsymbol{f}$ which reach zero if the system is in a valid steady state.

2.1 Simulation Tool Requirements

A steady state simulation tool must meet several requirements to be suitable to analyze complex systems: *Modularity* of the system allows to reuse functions and models of system components. It includes also the pos-

sibility to interconnect the component models in any way. This results in higher *flexibility in model creation.* *Flexibility in problem definition* means an easy and fast possibility to adapt the model to other questions. This means that in a certain model it is not fixed which variables are user given parameters and which ones are solver variables. The same applies also for interim variables and residuals. This simplifies the investigation of the system behavior as the same model allows the investigation of different problems.
High *computation speed* allows extensive parameter studies with large systems containing many degrees of freedom.
Computation robustness ensures that a solution is found also with bad starting values.

2.2 Solver Strategy

The requirements for modularity and flexibility result in the design principle of *local solvers* and a *global solver*: Local solvers are used inside a model of a component or in modular functions. They are used for numerical solving of equations which depends only on quantities which are known inside this subroutine. Therefore, many local solvers can exist in a model. An example is the calculation of an outlet temperature if a certain heat flow is added to a gas flow whose specific heat capacity depends on the temperature. In contrast, only one global solver exists in a system model. It solves quantities which depend on several system components. For example, the global solver determines a gas mass flow through several components such that at the outlet of the system the ambient pressure is achieved. Therefore, the global solver has to be a black box solver which needs no a priori knowledge of the system.

In principle it would be possible to combine all local solvers to the global solver. However, the use of local solvers enables the creation of standalone models of components. This simplifies both, the verification of component models and the reuse of these models outside of the simulation tool, for example in measurement data analyzes. Furthermore, local solvers can be specialized as the structure of the equations is known. This can improve computation robustness and speed.

3 GLOBAL SOLVER

The global solver finds a steady state which is defined by the system model. Therefore, the system can be described as a nonlinear function

$$\boldsymbol{f}: \mathbb{R}^n \mapsto \mathbb{R}^n : \boldsymbol{x} \mapsto \boldsymbol{f}(\boldsymbol{x}) \tag{1}$$

which maps n quantities which are determined by the global solver (*solver variables* $\boldsymbol{x}$) to n residuals ($\boldsymbol{f}(\boldsymbol{x})$). Basically the global solver has to find values for $\boldsymbol{x}$ such that the residual function is solved to zero. However, it must meet the requirements for computation speed and robustness. Furthermore, the use as a steady state solver for process simulations must be considered. This result in four design principles for the global solver:

1. The solver must consider given boundaries for $\boldsymbol{x}$ which are not exceeded during the solving process. This ensures physically meaningful solutions in all iterations. Furthermore this can prevent technically not meaningful solutions. These can occur if multiple physically steady states exist.
2. The main solution process must not use derivations: As the system model cannot provide exact derivations they can only approximated by difference quotients. However, the use of local solvers adds a small noise to the residual function as they solve equations only with a certain precision. Therefore, calculation of difference quotients with too small differences can result in strong deviations to the actual derivation.
3. The solution process must not be rewritten as an optimization problem. This is an often used workaround to consider the boundaries of the first principle. However, optimization routines either internally uses derivations which violates the second principle, or they have to sum up the residual function somehow to a single cost function. Thus, a lot of system information is lost.
4. The computation time of a system evaluation with all of its components and local solvers is typically much longer as the computation time of an iteration of the global solver routine. Therefore, the most important criterion for computation speed is the number of system evaluations which should be as small as possible.

Considering boundaries of $\boldsymbol{x}$ the task of the global solver is:

$$\begin{aligned} &\text{Find a sequence } \left(\boldsymbol{x}^k\right)_{k\in\mathbb{N}} \\ &\quad \text{such that:} \\ &\qquad \boldsymbol{f}(\boldsymbol{x}^\infty) = \boldsymbol{0} \\ &\qquad \boldsymbol{lb}_i^k \le \boldsymbol{x}_i^k \le \boldsymbol{ub}_i^k \quad \forall\, k \in \mathbb{N};\ 1 \le i \le n \end{aligned} \tag{2}$$

Where $\boldsymbol{lb}$ and $\boldsymbol{ub}$ are the lower and upper boundaries, respectively. In practice the sequence is terminated if all elements of $\boldsymbol{f}$ are close to zero within a specified tolerance.

3.1 Broyden's Method

A class of solvers which is very suitable to use in the global solver are *Quasi Newton methods* [11, 12, 13]. These methods are inspired by the Newton-Raphson method. In contrast, they do not use the Jacobian matrix $\boldsymbol{J}^k = \left[\frac{\partial \boldsymbol{f}(\boldsymbol{x}^k)}{\partial x_1}, \ldots, \frac{\partial \boldsymbol{f}(\boldsymbol{x}^k)}{\partial x_n}\right]$ to calculate a new value $\boldsymbol{x}^{k+1}$. Instead an approximation matrix $\boldsymbol{B}^k$ is used which is updated at each iteration using a special update rule. The update rule must satisfy the *Quasi-Newton equation*[11]:

$$\boldsymbol{B}^{k+1} \cdot \left(\boldsymbol{x}^{k+1} - \boldsymbol{x}^{k}\right) = \boldsymbol{f}\left(\boldsymbol{x}^{k+1}\right) - \boldsymbol{f}\left(\boldsymbol{x}^{k}\right) \tag{3}$$

Note that in a one dimensional case the update rule is uniquely determined. In that case all Quasi-Newton methods are equivalent to the secant method.

Here the so called *Broyden's good method* [14, 11] is used which is a prominent member of the Quasi Newton class. It is described in Eq. (4):

Initialization:

$$k := 0 \quad \boldsymbol{x}^0 \in \mathbb{R}^n \quad \boldsymbol{f}^0 := \boldsymbol{f}\left(\boldsymbol{x}^0\right) \quad \boldsymbol{B}^0 \in \mathbb{R}^{n\times n} \tag{4a}$$

Do until $\max\left(\left|\boldsymbol{f}^k\right|\right) \leq$ tolerance:

Newton step:

$$\Delta \boldsymbol{x}^k := -\left(\boldsymbol{B}^k\right)^{-1} \boldsymbol{f}^k \tag{4b}$$

$$\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + \Delta \boldsymbol{x}^k \tag{4c}$$

$$\boldsymbol{f}^{k+1} := \boldsymbol{f}\left(\boldsymbol{x}^{k+1}\right) \tag{4d}$$

Broyden update:

$$\boldsymbol{B}^{k+1} := \boldsymbol{B}^k + \frac{\left(\boldsymbol{f}^{k+1} - \boldsymbol{f}^k - \boldsymbol{B}^k \Delta \boldsymbol{x}^k\right)\left(\Delta \boldsymbol{x}^k\right)^\top}{\left(\Delta \boldsymbol{x}^k\right)^\top \Delta \boldsymbol{x}^k} \tag{4e}$$

$$k := k + 1 \tag{4f}$$

Note that the good Broyden method updates $\boldsymbol{B}^k$ such that for all $\Delta \boldsymbol{x}_\perp$ which are orthogonal to $\Delta \boldsymbol{x}^k$ hold $\boldsymbol{B}^k \Delta \boldsymbol{x}_\perp = \boldsymbol{B}^{k+1} \Delta \boldsymbol{x}_\perp$. Therefore, the Broyden update adds no information on directions orthogonal to $\Delta \boldsymbol{x}^k$.

Broyden's method is very suitable for the global solver in MGTS^3 because:

- The basic algorithm in Eq. (4) needs only one system evaluation per iteration. Even though the rate of convergence is smaller than for the Newton-Raphson method with a Jacobian determined by difference quotients, the total number of system evaluations is generally much lower and the total computation speed faster.
- Broyden's method doesn't use derivations. Therefore, it is robust against noise from local solvers.
- For typical MGT-based systems the variations of the Jacobian matrix are moderate. Therefore, an approximation of the Jacobian is often sufficient.

3.1.1 Scaling

The Newton-Raphson method is invariant to both, scaling of the residual function $\boldsymbol{f}$ and the solver variables $\boldsymbol{x}$. It is easy to show that, in contrast, Broyden's good method in Eq. (4) is only invariant to scaling of the residual function. Therefore, it is useful to scale the unknown variables $\boldsymbol{x}$ to achieve a good convergence behavior. A diagonal and invertible scale matrix $\boldsymbol{D}_x \in \mathbb{R}^{n\times n}$ is used to scale the physical system variables $\boldsymbol{x}_{\text{System}}$ to the variables used in the solver $\boldsymbol{x}_{\text{Solver}}$:

$$\boldsymbol{x}_{\text{Solver}} = \boldsymbol{D}_x \, \boldsymbol{x}_{\text{System}} \tag{5}$$

Here $\boldsymbol{D}_x$ should be chosen such that all elements of $\boldsymbol{x}_{\text{Solver}}$ are in the same order of magnitude.

Even though Broyden's method is invariant to scaling of the residual function such a scaling with a scaling matrix $\boldsymbol{D}_f \in \mathbb{R}^{n\times n}$ can be defined:

$$\boldsymbol{f}_{\text{Solver}} = \boldsymbol{D}_f \, \boldsymbol{f}_{\text{System}} \tag{6}$$

Such a scaling is useful to reduce rounding errors. However, in contrast to the scaling of the unknown variables, it is difficult to define a scale matrix $\boldsymbol{D}_f$ at the beginning of the solver routine without any a priori knowledge of the system: Depending on the start guess of $\boldsymbol{x}$ some values of the residuals can be very large at the beginning and decrease strongly during the iteration process. Likewise, some residual values can be very small at the beginning but reach very large values during the calculation. Therefore, in MGTS^3 a scaling of the residuals by the user is supposed and no additional scaling in the global solver is used.

3.2 Global Solver Routine

Broyden's method is the most important part of the global solver routine. However, additional elements are necessary to fulfill the design principles. This concerns in particular the consideration of boundaries of the solver variables during the iterations. The MGTS^3 global solver routine uses initial lower boundaries $\boldsymbol{lb}^0$ and upper boundaries $\boldsymbol{ub}^0$. These can be further limited for the following iteration by values defined in the model during a system evaluation:

$$\begin{aligned} \boldsymbol{lb}_i^{k+1} &= \max\left(\boldsymbol{lb}_i\left(\boldsymbol{x}^k\right), \boldsymbol{lb}_i^0\right) \\ \boldsymbol{ub}_i^{k+1} &= \min\left(\boldsymbol{ub}_i\left(\boldsymbol{x}^k\right), \boldsymbol{ub}_i^0\right) \\ &\text{for } 1 \leq i \leq n \end{aligned} \tag{7}$$

This method allows the use of boundaries which depend on other solver variables as long as these dependencies are not circular and thus prevent convergence. Typically, the initial boundaries $\boldsymbol{lb}^0$ and $\boldsymbol{ub}^0$ represent physical limits. Adapting these limits can improve the speed of convergence or prevent the solver from converging to a technically not meaningful solution.

During the solving process the Jacobian matrix sometimes needs to be calculated by finite differences. As the boundaries often represent physical limits this can result in a singular Jacobian if the calculation is done with some elements of $\boldsymbol{x}$ on a boundary. In this

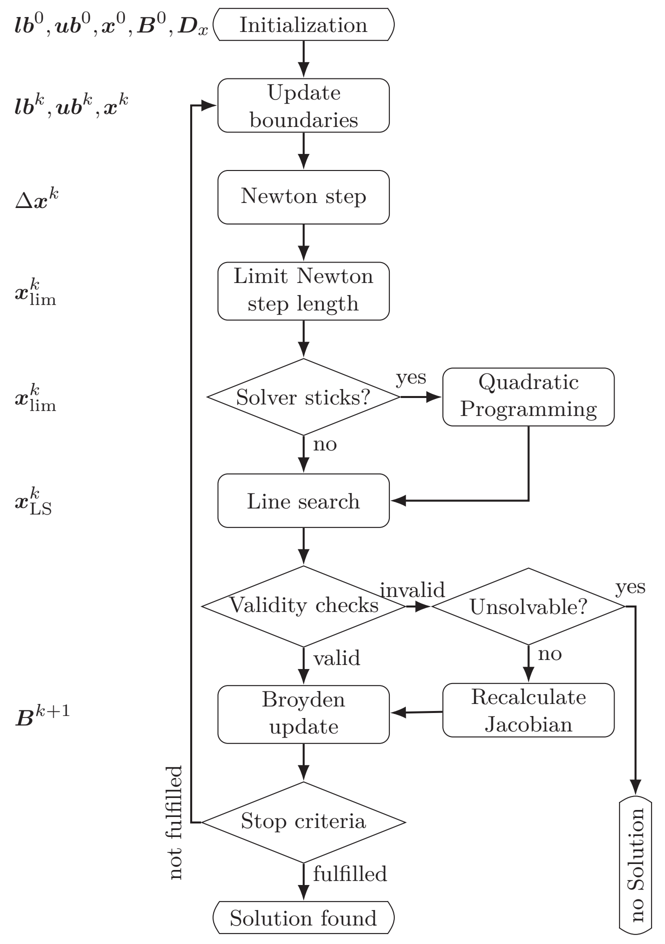

Figure 1. GLOBAL SOLVER PROCEDURE AND CALCULATED QUANTITIES FOR EACH STEP.

case the appropriate elements of $\boldsymbol{x}$ are slightly set besides their boundaries during the calculation of the Jacobian.

Figure 1 shows the main steps of the global solver routine which will be described in this section.

Initialization The initialization process determines the first approximation of the Jacobian $\boldsymbol{B}^0$. This is done by approximating the Jacobian with finite differences. In parameter studies subsequent studies are initialized with the last Broyden approximation of the previous steady state as differences between the boundary conditions are typically small. In this case also the values of the last steady state are taken as start guesses for $\boldsymbol{x}$. Furthermore, the scale matrix $\boldsymbol{D}_x$ is determined using the inverse of the differences of initial upper and lower boundaries. If these boundaries are infinite the absolute values of the start guesses $\boldsymbol{x}^0$ are used instead.

Update Boundaries At the beginning of each iteration k the updated boundaries of the previous iteration (Eq. (7)) are checked. If a boundary is violated the related element of $\boldsymbol{x}^k$ is set to the appropriate boundary and the system is evaluated again.

Newton Step New values $\boldsymbol{x}^k_{\text{Newton}} = \boldsymbol{x}^k + \Delta\boldsymbol{x}^k_{\text{Newton}}$ of the solver variables are calculated for a full Newton step. This is done with the approximation of the Jacobian $\boldsymbol{B}^k$ and Eq. (4b).

Limitation If a boundary is violated the step size $\Delta\boldsymbol{x}^k_{\text{Newton}}$ is reduced to the largest possible step length $s^k_{\max}$ without altering the direction of the step:

$$\begin{aligned} s^k_{\max} &= \underset{0\le s\le 1}{\arg\max}\,\{s\} \\ &\qquad s.t.: \boldsymbol{lb}^k_i \le \boldsymbol{x}^k_i + s\cdot\Delta\boldsymbol{x}^k_{\text{Newton},i} \le \boldsymbol{ub}^k_i \\ &\qquad \text{for } 1\le i\le n \\ \boldsymbol{x}^k_{\text{lim}} &= \boldsymbol{x}^k + s^k_{\max}\cdot\Delta\boldsymbol{x}^k_{\text{Newton}} \end{aligned} \tag{8}$$

Quadratic Programming The maximal Newton step length $s^k_{\max}$ can become zero if elements of $\boldsymbol{x}^k$ are already on a boundary and the Newton step is in direction of this boundary. Then $\boldsymbol{x}^k_{\text{lim}}$ is identical with $\boldsymbol{x}^k$ and the solver sticks to this point. Therefore, if $s^k_{\max}$ is smaller than a critical value, the Newton equation (4b) is replaced with a constrained optimization problem:

$$\begin{aligned} &\text{if } s^k_{\max} < s_{\text{critical}}: \\ &\boldsymbol{x}^k_{\text{lim}} = \underset{\boldsymbol{x}_{\text{QP}}}{\arg\min}\left\{\left\|\boldsymbol{D}_{\text{QP}}\left(\boldsymbol{f}^k + \boldsymbol{B}^k(\boldsymbol{x}_{\text{QP}} - \boldsymbol{x}^k)\right)\right\|^2\right\} \\ &\qquad\qquad \text{s.t.:}\quad \boldsymbol{lb}^k_i \le \boldsymbol{x}_{\text{QP,i}} \le \boldsymbol{ub}^k_i \quad \text{for } 1\le i\le n \\ &\qquad\qquad \text{with:}\quad \boldsymbol{D}_{\text{QP}} \in \mathbb{R}^{n\times n} \end{aligned} \tag{9}$$

Here $\boldsymbol{D}_{\text{QP}}$ is a scaling matrix which is diagonal and invertible. It is build using the inverse of the absolute values of $\boldsymbol{f}^k$. Note that Eq. (9) is equivalent to Newton's method for an unconstrained case. Furthermore, note that the use of an optimization routine is not contradictory to the third design principle of the global solver as the optimizing is not for the system itself. The use of quadratic programming is in particularly helpful if the solution for one element of $\boldsymbol{x}$ is close to or on a boundary. In MGTS^3 the quadratic program is solved with an *active set method* [15] as this method guarantees the exact compliance of the boundaries.

Line Search For some problems or bad values of $\boldsymbol{x}^0$ the full limited newton step is not optimal. Therefore, it is sometimes useful to maintain the direction of the newton step but to reduce the step length [14, 15]. The line search finds an optimum for a scaled norm of $\boldsymbol{f}(\boldsymbol{x})$:

$$\begin{aligned} \boldsymbol{x}^k_{\text{LS}} &= \underset{\boldsymbol{x}}{\arg\min}\left\{\|\boldsymbol{D}_{\text{LS}}\boldsymbol{f}(\boldsymbol{x})\|^2\right\} \\ &\qquad \text{s.t.:}\quad \boldsymbol{x} = \boldsymbol{x}^k + \alpha\cdot\left(\boldsymbol{x}^k_{\text{lim}} - \boldsymbol{x}^k\right) \\ &\qquad\qquad \alpha_{\min} \le \alpha \le 1 \end{aligned} \tag{10}$$

The scale matrix $\boldsymbol{D}_{\mathrm{LS}}$ is built with the absolute inverses of the elements of $\boldsymbol{f}\left(\boldsymbol{x}_{\mathrm{lim}}^{k}\right)$. As minimizing algorithm a variant of a derivative free optimizing routine of Brent is used [16]. Some additional stop criteria and minor changes were added to the original algorithm to solve Eq. (10) with fewer function calls and without an unnecessarily high precision.

The line search can improve the convergence and for difficult problems it can increase the computation speed. However, at least one additional system evaluation is needed to perform the line search. Furthermore, often $\boldsymbol{x}_{LS}^{k}$ is close to or exact $\boldsymbol{x}_{\mathrm{lim}}^{k}$. Therefore, the line search is only used if at least a certain number of iterations are needed or the newton step seems to worsen the result. This is if $\left\|\boldsymbol{D}_{\mathrm{LS}}\boldsymbol{f}\left(\boldsymbol{x}_{\mathrm{lim}}^{k}\right)\right\| > \left\|\boldsymbol{D}_{\mathrm{LS}}\boldsymbol{f}^{k}\right\|$.

Validity Checks The provided system can be infeasible. To detect this, the quotient of the maximum absolute value of $\boldsymbol{f}(\boldsymbol{x}_{\mathrm{LS}}^{k})$ and the maximum absolute value of $\boldsymbol{f}^{k}$ is used. If this quotient is several times in a row close to 1 the global solver sticks either on a limit or on a local minimum of $\|\boldsymbol{f}(\boldsymbol{x})\|$ and the system is infeasible. If the quotient is significant larger then 1 the solver process continues although the system behavior is contradictory to the expected one. If this happens several times in a row the Broyden matrix is replaced by a new calculated Jacobian. This is also done if a boundary is hit several times in a row.

Broyden Update The Jacobian approximation is updated according to Eq. (4e) with $\Delta\boldsymbol{x}^{k} = \boldsymbol{x}_{LS}^{k} - \boldsymbol{x}^{k}$. The routine terminates if all absolute values of the residual function $\boldsymbol{f}\left(\boldsymbol{x}_{LS}^{k}\right)$ are smaller than a desired tolerance. Otherwise the iteration counter k is increased and a new iteration is started.

4 MGTS^3 SIMULATION TOOL

The global solver is only one aspect of the MGTS^3 simulation tool. Other parts of the simulation tool were already described by Panne et al. [3] and Henke et al. [17]. In this section a short overview of the simulation tool is given with special emphasis on further developments not mentioned in previous publications.

4.1 Model Creation in Simulink

MGTS^3 is implemented in Matlab and Simulink: All component models are written in Matlab code to allow reusability for other applications independent of Simulink. Simulink is used as a graphical user interface to create and simulate system models and to organize the component library. It allows a graphical interconnection between the component blocks using specified interfaces like gas flows or shafts. Simulink autonomously determines the calculation order of all blocks. A special initial method is implemented which collects all residuals for the global solver and distributes the calculated solver variables. Hence, this allows a flexible method to define the positions and numbers of residuals and solver variables in the model. Although Simulink is designed as transient simulation platform, it is used here to perform steady state studies. For this, the simulation time is interpreted as index for parameter studies.

4.2 Components Overview

All gas flows are assumed as ideal gas mixtures. Temperature depending polynomials are used to calculate heat capacity [18], viscosity and heat conductivity [19] of pure substances.

Turbomachinery behavior is calculated using the pressure ratio and isentropic efficiency [1, 3]. These values can be interpolated from a map according to the operation point or directly specified. The second method is for example useful during the design of new systems. Recuperators and water heat exchangers are calculated assuming a constant efficiency [1, 17]. Other components implemented in MGTS^3 are combustion chambers [3], piping, flow branchings and mixers with or without recirculation, fuel cell models [3, 5] and generators [17].

4.3 Heat and Pressure Losses

Nearly all components consider heat and pressure losses. For both losses a selection of loss mechanism is available. Pressure loss calculation can be selected as absolute, relative, proportional to the squared volume flow or proportional to the squared mass flow divided by the density. The first three methods are further described by Henke et al. [17]. The last one is equivalent to the Darcy-Weisbach equation with a constant friction factor c_{f}:

$$\Delta p = \frac{c_{\mathrm{f}} \cdot L}{2 \cdot D \cdot A^2} \cdot \rho \cdot \dot{V}^2 = \underbrace{\frac{c_{\mathrm{f}} \cdot L}{2 \cdot D \cdot A^2}}_{c_{\Delta p}} \cdot \frac{\dot{m}^2}{\rho} \tag{11}$$

Heat loss calculation can be selected as a constant temperature difference, a change of temperature proportional to the temperature difference of working gas and ambient conditions (Eq. (12a)) or as a heat flow proportional to this difference (Eq. (12b)). The last two cases uses a mean component temperature T_{mean} which is calculated by a weighted mean value of the inlet and the outlet temperatures of the component (Eq. (12c)):

$$\Delta T_{\mathrm{loss}} = c_{\mathrm{loss},T} \cdot (T_{\mathrm{mean}} - T_{\mathrm{amb}}) \tag{12a}$$

$$\dot{Q}_{\mathrm{loss}} = c_{\mathrm{loss},\dot{Q}} \cdot (T_{\mathrm{mean}} - T_{\mathrm{amb}}) \tag{12b}$$

$$T_{\mathrm{mean}} = w \cdot T_{\mathrm{in}} + (1 - w) \cdot T_{\mathrm{out}} \tag{12c}$$

The weight factor w can be given or it is calculated using a mean logarithmic temperature difference of the

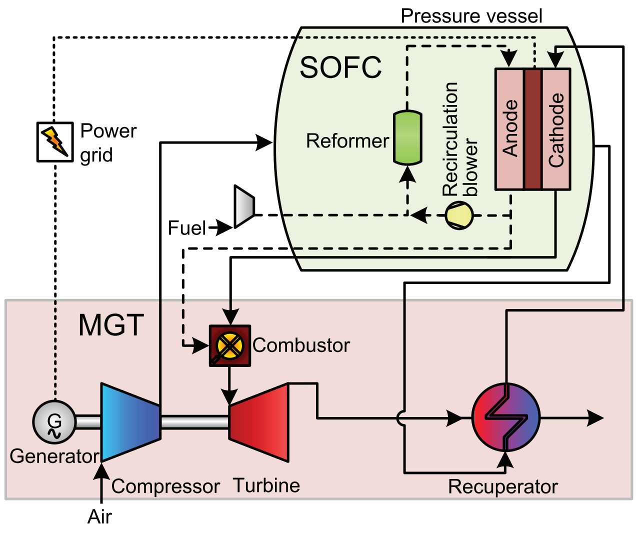

Figure 2. SCHEMATIC STRUCTURE OF HYBRID POWER PLANT DEMONSTRATOR.

component to the ambient [20]. The second method results in a larger computation effort but guarantees physical meaningful heat losses even for very small mass flows.

5 EXEMPLARY CASE STUDIES

Some capabilities of the MGTS^3 system and its global solver are shown with three exemplary applications. They demonstrate the flexibility, robustness and computation speed of the simulation tool. Also, the different elements of the solver routine are analyzed separately to visualize their impact on the performance. The examples include a comparison between the MGTS^3 global solver, the previously used solver and Simulink's standard algebraic loop solver, which is widely used for academic and industrial applications.

5.1 Hybrid Power Plant

The here investigated hybrid power plant (HyPP) is a combination of a solid oxide fuel cell (SOFC) and a micro gas turbine. A HyPP demonstrator is currently build at DLR with a designated electric power of about 30 kW and a predicted electric efficiency above 60 %. Figure 2 shows the structure and most important components of the HyPP steady state model. The fuel cell is placed inside a pressure vessel which is purged by pressurized air [5]. The MGTS^3 model of the HyPP describes the compressor and the turbine with turbomachinery maps derived from measurement data. The fuel cell is represented by a 0D-model of the SOFC stack which is developed by the DLR Institute of Engineering Thermodynamics and parametrized by measurement data. Heat and pressure losses are considered for all components and the piping.

From the modeling point of view, the recirculation of the anode gas is interesting as the gas properties of the recirculated flow cannot be calculated directly. One intuitive solution is given by iteratively calculating the mixed gas flow properties based on recirculated gas flow properties of the last iteration (e.g. Panne et al. [3]). However, this simple solution must not be used in combination with a global Newton-like solver method. On the one hand, it can lead to a large amount of interim system evaluations between global solver iterations, caused by the iterations of the intuitive solution. On the other hand, the solution is not as robust against bad starting values. Therefore, in MGTS^3 the recirculation is solved by the global solver: The temperature and the species concentrations of the mixed flow are calculated by the global solver. The corresponding residuals are determined later in the system calculation process using the incoming flow, the recirculated flow and the mentioned values of mixed flow. Note that as long as the numbers of species is not extremely large this method results in less system evaluations than the intuitive method mentioned above as the convergence speed is much faster.

5.1.1 Comparison to Simulink Algebraic Solver

The HyPP steady state model is used to investigate the number of system evaluations for different global solver methods which is also an indication of the computation time. In this analysis the global solver method is compared with the Simulink standard method to solve algebraic equations using an "algebraic constraint" block. The system models for the MGTS^3 global solver and the reference case are identical. According to the Simulink documentation its algebraic solver uses either a trust-region method or a line-search method [21]. As with the HyPP model the behavior of both Simulink algorithms is very similar only the trust-region-algorithm is mentioned here. It is based on the HYBRD1 program of the FORTRAN library MINPACK [22] which is itself based on a method of Powell [23, 24]. It uses also the Broyden method to approximate the Jacobian matrix. Simulink uses no boundaries of the solver variables which causes very often nonphysical values for the HyPP model. Therefore, the solver variables are bounded using an additional sigmoid function. However, this solution is not suitable for problems where the solution of some solver variables is close or equal to their boundary. Furthermore, the computation precision of the Simulink solver cannot be defined by the user. Therefore, in the following the Simulink solver is forced to achieve approximately the used precision of the MGTS^3 global solver which is 10^{-6} for the normalized residuals. This is achieved by a multiplication of the residuals with an empirical value at the port of the algebraic constraint block.

The here investigated model configuration specifies among others the electric power output, the turbine outlet temperature, the stack outlet temperature and the recirculation rate as user given parameters. The most

Table 1. NUMBERS OF SYSTEM EVALUATIONS FOR DIFFERENT START VALUES OF THE AIR MASS FLOW. (NC=NO CONVERGENCE)

Start value (rel. to solution)	Simulink solver	MGTS^3 solver
1.00	37	26
1.05	133	28
1.10	155	33
1.40	127	41
1.60	229	40
1.80	275	59
1.90	NC	63
2.00	NC	55
2.50	NC	106
3.00	NC	570
4.00	NC	323
4.50	NC	409

important quantities solved by the global solver are the fuel and air mass flows, the shaft speed and the SOFC fuel utilization. In total, there are 15 global solver variables and residuals.

5.1.2 Solver Comparison: Start Value Variation

First the behavior of the global solver methods is investigated for an increasing difference of the start values to the solution. To do this, a HyPP reference point is chosen with an electrical load of 30 kW and a stack temperature of 1123 K. This point is simulated several times, each with different start values for one solver variable while the start values for all other solver variables are set to the steady state solution of the reference point.

Table 1 shows exemplary the number of system evaluations for varying deviations of the start values of air mass flow. It can be seen that the MGTS^3 global solver needs significantly less system evaluation than the Simulink solver. At least with small variations of the start values this is surprising because the base of both solvers is the Broyden method. The possibilities to investigate the Simulink solver are limited but further investigation indicates that this is caused by the scaling of the solver variables in the MGTS^3 solver which is probably not done in the Simulink solver. For larger variations of the start value the additional strategies of the MGTS^3 global solver become more important, namely the consideration of solver value boundaries in the solver routine itself and the additional line search. Beside needing less system evaluations, this results also in a more robust convergence behavior for a much larger variation of the start values.

Table 2. NUMBERS OF ITERATIONS AND SYSTEM EVALUATIONS FOR DIFFERENT VARIANTS OF GLOBAL SOLVER FOR A SMALL PARAMETER STUDY.
(NA=NOT AVAILABLE)

Solver method	First steady state		Following steady states (average)	
	Iter.	Sys. eval.	Iter.	Sys. eval.
MGTS^3	22	44	10.9	14.6
MGTS^3 Newton instead Broyden	11	167	7.9	120.3
MGTS^3 always line search	22	84	10.8	34.3
Simulink trust-region	NA	254	NA	128.3

5.1.3 Solver Comparison: Consecutive Simulations

With the same configuration of the HyPP steady state model a small exemplary parameter study is performed by reducing the specified electric power output from 30 kW to 20 kW in ten steps. The study is performed using different global solver configurations: With the global solver as described above, with the global solver but with Newton's method instead the Broyden update and with Broyden update combined with a line search performed every iteration. Furthermore the parameter study is performed with the Simulink algebraic solver. Start values of solver variables are set such that they significantly differ from the solution: The fuel mass flow differs by about 50 %, the air mass flow by about 16 %. The mean absolute deviation of the other values is about 55 %.

Table 2 shows the numbers of iterations and the numbers of system evaluations for the first steady state and a mean value for the the following nine steady states. Here the MGTS^3 global solver takes the Broyden matrix and start values from the last steady state. It is assumed that the Simulink solver has the same behavior even tough this is not described in the documentation. As expected, Newton's method need usually less iterations than the Broyden update but the numbers of system evaluations is significant larger as the computation of the Jacobian matrix needs a lot of system evaluations. In addition, there are examples where Newton's method needs more iterations than Broyden's method. This can happen if some crucial Jacobian matrices result in a significant deterioration of the solver variables, for example if local solvers produce large errors in the derivations. Also the result shows an example where the line search does not improve the performance as the number of iterations is not affected while the number of system evaluations is much larger. However, the line search can be very useful for bad start values. In summary, the parameter study confirms important aspects of the global solver routine, namely the derivative free method and the use of a line search only in certain cases.

With the Simulink solver it is only possible to count the overall system evaluations but not the iteration steps of the solver itself as the solver routine is hidden from the user. The average number of system evaluations is more than eight times larger as with the MGTS^3 solver. Using manual scaling of solver variables the number of system evaluations per steady state can be reduced to about 70.

The overall processing time[1] without model initialization in a typical parameter study is less than 1.5 s per steady state with the MGTS^3 system and the HyPP model. This is four to six times faster than the Simulink solver. The increase in computation time is smaller than the increase in system evaluations. This is mainly caused by process overhead such as saving of the system results.

5.2 Inverted Brayton Cycle

The next demonstration case is based on an inverted Brayton cycle (IBC). Utilizing such a cycle, the power of a micro gas turbine is reduced compared to the classical Brayton cycle with same turbomachinery without a strong impact on electrical efficiency [1, 26]. Therefore, this cycle is particularly suitable for combined heat and power generation in small residential houses. At the DLR Institute of Combustion Technology an IBC application is currently investigated using MGTS^3 [2]. A schematic overview of the cycle is given in Fig. 3. The model includes an exhaust gas recirculation, a recuperator and two water heat exchangers. Heat and pressure losses are considered for all components. The turbomachinery models either rely on component maps or on user-given pressure ratios and efficiencies. The latter is for example useful to identify optimal turbomachinery characteristics for a given cycle.

The model was parametrized and validated with experimental data by Agelidou et al. and a performance study was performed [2]. The model contains each 13 solver variables and residuals and was able to reproduce the experimental measurement data with good agreement. The MGTS^3 global solver converged for all operation points.

Henke et al. implemented the same model in the previous steady state simulation tool of the DLR Institute of Combustion Technology [1]. While the component models are essentially identical with the MGTS^3,

[1]Computed with Matlab/Simulink R2017a on a virtual Win7 machine (2.3 GHz). Some results of Matlab's `bench` command [25]: Relative speed: ca. 0.45; LU: ca. 0.30 s; FFT: ca. 0.28 s; ODE: ca. 0.07 s

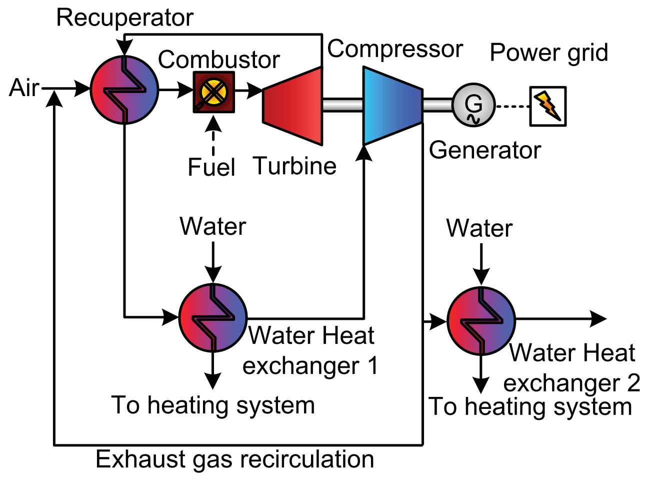

Figure 3. STRUCTURE OF INVERTED BRAYTON CYCLE MODEL.

the global solver differs completely: Instead of a black box solver which solves all solver variables simultaneous, cascaded secant solvers are used, one for each solver variable. Therefore, the user has to specify manually the connection between a solver variable and a residual. Furthermore, some special computation rules are implemented to achieve a stable convergence. Therefore, this global solver is specialized only on the IBC model and no flexibility in problem definition as described above is achieved. However, this method allows to use the "intuitive solution" to implement the recirculation as described in the hybrid power plant section. This reduces the number of solver variables to 4.

Table 3 shows the number of system evaluations for solving the IBC model with a certain residual precision. The listed values are generated with a single exemplary steady state, the computation time is without initialization of the model. Again, the MGTS^3 global solver shows a large reduction of the number of system evaluations to less than a sixth. Also it can be seen that the computation time is reduced to less than 5 % and thus its reduction is stronger than the reduction in the number of system evaluations. This is the result of several optimizations regarding implementation details in MGTS^3, mainly a reduction of Matlab/Simulink overhead that was unavoidable for the previous solver.

In typical parameter studies the solver converge for all operation points with an average computation time of 0.5 s without initialization.

5.3 Auxiliary Power Unit

The last mentioned application case is the simulation of the aircraft auxiliary power unit (APU) GTCP36-28. This is a MGT without recuperation which provides electric energy and compressed air. The APU and its simulation model are described in detail by Zanger et al. [27] including measurement and simulation data. Figure 4 shows the structure of the APU model

Table 3. NUMBERS OF SYSTEM EVALUATIONS AND COMPUTATION TIME FOR THE IBC MODEL.

Precision	Previous steady state simulation tool		MGTS[3]	
	Sys. eval.	Time [s]	Sys. eval.	Time [s]
$1 \cdot 10^{-3}$	152	40	21	1.5
$5 \cdot 10^{-4}$	155	44	26	1.5
$3 \cdot 10^{-4}$	185	57	26	1.4
$1 \cdot 10^{-4}$	620	256	35	1.5
$1 \cdot 10^{-5}$	NA	NA	50	1.9
$1 \cdot 10^{-6}$	NA	NA	41	1.9

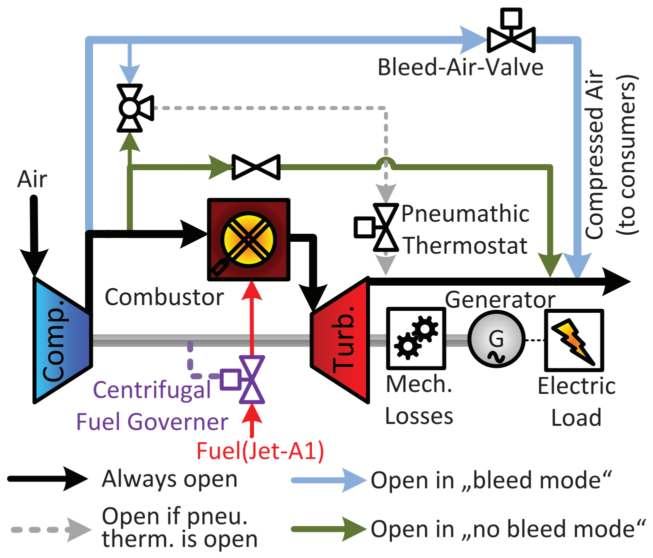

Figure 4. STRUCTURE OF AUXILIARY POWER UNIT MODEL.

used here. The line width of the flows indicates qualitatively the mass flow rate. The challenging aspect of the simulation is given by the many flow branches that are considered in the model. As Fig. 4 shows, the APU has a *bleed mode* and a *no bleed mode*. In each mode some of the flow paths are closed. This can be very challenging for a simulation system: The global solver has to determine the mass flow split ratio at each branch which must not exceed the range between 0 and 1. The constraints are typically pressure differences at the flow mixers or pressure ratios of a nozzle or a valve in the branch. However, for a closed branch the pressure difference at the flow mixer is useless. Furthermore, systems with nested branches can generate singular Jacobian matrices if the main branch is closed. Figure 4 shows that this happens in the no bleed mode: Then the branching from the bleed air to the pneumatic thermostat has no effect on any constraints.

MGTS[3] overcomes these challenges: As MGTS[3] considers limits, the physically meaningful range of flow ratios cannot be exceeded. Furthermore, nested branches result not in an initial singular Jacobian as during its calculation the exact upper or lower boundaries are avoided. The Broyden update usually does not result in a singular matrix during the calculation. If so, then this is handled well during the validity checks. Furthermore, each flow of the model contains a flag which is set during the solving procedure and marks whether the branch is open or not. Therefore, the flow mixer can decide whether the pressure difference is a useful constraint.

The MGTS[3] tool can reproduce the measurement data very well [27] and is able to perform a robust calculation of the system. The average calculation time is about 1.1 s per steady state[1]. The calculation is quite robust although for some parameter studies the nested branches can still be challenging. A very important fact for the robustness is the consideration of boundaries for the solver variables and the ability to find solutions with some values close to or on their boundaries. Therefore, no comparison with the standard Simulink solver was possible, as no configuration or initial conditions could be found that enabled convergence.

6 CONCLUSIONS AND OUTLOOK

Simulation tools for steady states of MGT-based systems are required to be highly flexible, utilizing fast and robust computation routines. The presented MGTS[3] tool fulfills these demands. Therefore, it provides a valuable tool to model and investigate innovative cycles. This is mainly achieved by the global solver routine which is introduced in this paper. The routine is based on Broyden's method which results in a fast computation without the need to compute the Jacobian in each iteration. Furthermore, the derivative free method is robust against noise in the residuals caused by local solvers. While Broyden's method is well known and often used the here presented global solver routine expands it using two main principles: A limitation of the Newton step in each iteration, to consider boundaries, ensures physically and technically meaningful solutions throughout the solution procedure. A line search along the direction of the Newton step often improves the convergence if start values of the problem are poorly chosen or the system is highly nonlinear.

MGTS[3] and its global solver routine solver can handle complex innovative cycles. This includes recirculations as well as nested and closed branches. Its capabilities are demonstrated by three application examples which are presented in this paper.

The analysis of the hybrid power plant shows that the new global solver routine has a much higher performance as the Simulink built in trust-region algebraic solver: In typical parameter studies only about 10 % of the system evaluations are needed. Furthermore, a significantly higher robustness against variations in the start values is observed.

Compared with the solver routine of the previous steady state tool a substantial improvement is reached which is shown with the example of the Inverted Brayton Cycle: Firstly, the flexibility of the model is enhanced as the solver must not be adapted manually for different problems. Secondly, the computation time and number of system evaluations is also reduced drastically.

The new solver routine can also handle very complex problems, for example with solutions directly on boundaries or in the instance of singular Jacobians. This is demonstrated in the third application case of an aircraft auxiliary unit.

Nevertheless, there is potential for improvements: The global solver can run into problems if the residual function is not continuous. This can happen if e.g. the residuum is composed by several conditions. An example is the implementation of a fuel control which is limited by the maximum turbine inlet temperature, turbine outlet temperature and electric output at the same time. A possible solution for that is currently being developed. For this approach each residuum is divided in several sub-residuals. The system is solved such that all sub-residuals are lower or equal to zero and at least one sub-residuum is equal to zero. It is expected that this method further expands the application possibilities of the global solver and the MGTS^3 tool.

Acknowledgements

The authors thank our colleges of the DLR Institute of Engineering Thermodynamics for the SOFC stack model. We also thank the HyPP team of the DLR Institute of Combustion Technology, especially Melanie Herbst for the turbomachinery map data. Finally, we thank Eleni Agelidou and Jan Zanger for the experimental data which are used to develop the models of IBC and APU.

This work was funded by the Initiative and Networking Fund of the Helmholtz Association in the future topic "Energy Systems Integration" under grant number ZT-0002.

REFERENCES

[1] Henke, M., Monz, T., and Aigner, M. "Inverted Brayton Cycle with exhaust gas recirculation - a numerical investigation." *Journal of Engineering for Gas Turbines and Power* Vol. 135. No. 9 (Aug. 2013): pp. 091203–1 - 091203–7. DOI: `10.1115/1.4024954`.

[2] Agelidou, E., Henke, M., Monz, T., and Aigner, M. "Numerical investigation of an inverted brayton cycle micro gas turbine for CHP application based on experimental data." *Proceedings of ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition (GT 2018)*. GT2018-76377. Oslo, Norway, June 11–15, 2018.

[3] Panne, T., Widenhorn, A., Boyde, J., Matha, D., Abel, V., and Aigner, M. "Thermodynamic Process Analyses of SOFC/GT Hybrid Systems." *5th International Energy Conversion Engineering Conference and Exhibit (IECEC)*. AIAA 2007-4833. St. Louis, Missouri, June 25–27, 2007. DOI: `10.2514/6.2007-4833`.

[4] Calise, F., Palombo, A., and Vanoli, L. "Design and partial load exergy analysis of hybrid SOFC–GT power plant." *Journal of Power Sources* Vol. 158. No. 1 (July 2006): pp. 225–244. DOI: `10.1016/j.jpowsour.2005.07.088`.

[5] Steilen, M., Salettia, C., Heddrich, M. P., and Friedrich, K. A. "Analysis of the in Influence of heat transfer on the stationary operation and performance of a Solid Oxide Fuel Cell/Gas Turbine hybrid power plant." *Applied Energy* Vol. 211 (2018): pp. 479–491. DOI: `10.1016/j.apenergy.2017.11.038`.

[6] Traverso, A. "TRANSEO Code for the Dynamic Performance Simulation of Micro Gas Turbine Cycles." *Proceedings of ASME Turbo Expo 2005: Power for Land, Sea, and Air*. GT2005-68101, pp. 45–54. Reno, Nevada, June 6–9, 2005. DOI: `10.1115/GT2005-68101`.

[7] Traverso, A., Massardo, A. F., and Scarpellini, R. "Externally fired micro-gas turbine: modelling and experimental performance." *Applied Thermal Engineering* Vol. 26. No. 16 (2006): pp. 1935–1941. DOI: `10.1016/j.applthermaleng.2006.01.013`.

[8] Visser, W. "Generic Analysis Methods for Gas Turbine Engine Performance: The development of the gas turbine simulation program GSP." PhD thesis. TU Delft, Jan. 6, 2015. DOI: `10.4233 / uuid : f95da308 - e7ef - 47de - abf2 - aedbfa30cf63`.

[9] Schefflan, R. *Teach Yourself the Basics of Aspen Plus*. Wiley, Hoboken, New Jersey (2011).

[10] Klein, S. *EES - Engineering Equation Solver for Microsoft Windows Operating Systems*. Manual. F-Chart Software, (2009).

[11] Martínez, J. M. "Practical quasi-Newton methods for solving nonlinear systems." *Journal of Computational and Applied Mathematics* Vol. 124. No. 1–2 (2000). Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations: pp. 97–121. DOI: `10.1016/S0377-0427(00)00434-9`.

[12] Barnes, J. G. P. "An Algorithm for Solving Non-Linear Equations Based on the Secant Method." *The Computer Journal* Vol. 8. No. 1 (Apr. 1965): pp. 66–72. DOI: `10.1093/comjnl/8.1.66`.

[13] Kosmol, P. *Methoden zur numerischen Behandlung nichtlinearer Gleichungen und Optimierungsaufgaben*. German. B.G. Teubner, Stuttgart, Germany (1993).

[14] Broyden, C. G. "A class of methods for solving nonlinear simultaneous equations." *Mathematics of Computation* Vol. 19. No. 92 (Apr. 1965):

pp. 577–593. DOI: `10.1090/s0025-5718-1965-0198670-6`.

[15] Nocedal, J. and Wright, S. J. *Numerical optimization.* 2. ed. Springer, New York (2006).

[16] Brent, R. P. *Algorithms for minimization without derivatives.* Prentice-Hall, Englewood Cliffs, New Jersey (1973).

[17] Henke, M., Klempp, N., Hohloch, M., Monz, T., and Aigner, M. "Validation of a T100 Micro Gas Turbine Steady-State Simulation Tool." *Proceedings of ASME Turbo Expo 2015: Turbine Technical Conference and Exposition,GT2015-42090, June 15-19, 2015, Montreal, Canada.* 2015.

[18] Goos, E., Burcat, A., and Ruscic, B. *Extended Third Millennium Ideal Gas and Condensed Phase Thermochemical Database for Combustion With Updates From Active Thermochemical Tables.* 2010. URL: `http://burcat.technion.ac.il/dir/BURCAT.THR`.

[19] Kleiber, M., Joh, R., and Span, R. "Properties of Pure Fluid Substances." In: [28], chap. D3, pp. 301–418.

[20] Gnielinski, V. "Heat Transfer in Pipe Flow." In: [28], chap. G1, pp. 691–700.

[21] The MathWorks Inc. *Algebraic Loops - How the Algebraic Loop Solver Works.* 2017. URL: `https://mathworks.com/help/simulink/ug/algebraic-loops.html#bsn46y8-2` (visited on 01/26/2018).

[22] Moré, J. J., Garbow, B. S., and E., H. K. *User Guide for Minpack-1.* Ed. by A. N. Laboratory. Rept. ANL-80-74. (1980).

[23] Powell, M. J. D. "A hybrid method for nonlinear equations." In: [29], pp. 87–114.

[24] Powell, M. J. D. "A fortran subroutine solving systems of nonlinear algebraic equations." In: [29], pp. 115–161.

[25] The MathWorks Inc. *bench - Matlab Benchmark.* 2017. URL: `https://mathworks.com/help/matlab/ref/bench.html` (visited on 01/26/2018).

[26] Agelidou, E., Monz, T., Huber, A., and Aigner, M. "Experimental investigation of an inverted brayton cycle micro gas turbine for CHP application." *Proceedings of ASME Turbo Expo 2017: Turbomachinery Technical Conference and Exposition (GT 2017).* GT2017-64490, pp. V008T26A023. Charlotte, North Carolina, June 26–30, 2017. DOI: `10.1115/gt2017-64490`.

[27] Zanger, J., Krummrein, T., Siebel, T., and Roth, J. "Characterization of an aircraft auxiliary power unit test rig for cycle optimization studies." *Proceedings of ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition (GT 2018).* GT2018-76377. Oslo, Norway, June 11–15, 2018.

[28] P. Stephan et al., eds. *VDI Heat Atlas.* Springer, Berlin, Heidelberg (2010).

[29] P. Rabinowitz, ed. *Numerical methods for nonlinear algebraic equations: a conference held at the University of Essex on January 6 and 7, 1969.* Gordon and Breach, London (Jan. 6–7, 1969).