Michael Palman, Boris Leizeronok and Beni Cukurel*

Comparative study of numerical approaches to adaptive gas turbine cycle analysis

https://doi.org/10.1515/tjeng-2021-0021
Received June 6, 2021; accepted June 28, 2021;
published online July 9, 2021

Abstract: Significant increase in task complexity for modern gas-turbine propulsion systems drives the need for future advanced cycles' development. Further performance improvement can be achieved by increasing the number of engine controls. However, there is a lack of cycle analysis tools, suitable for the increased complexity of such engines. Towards bridging this gap, this work focuses on the computation time optimization of various mathematical approaches that could be implemented in future cycle-solving algorithms. At first, engine model is described as a set of engine variables and error functions, and is solved as an optimization problem. Then, the framework is updated to use advanced root-finding paradigms. Starting with Newton-Raphson, the model is improved by applying Broyden's and Miller's schemes and implementing solution existence validation. Finally, algorithms are compared in representative condition using increasingly complex turbojet and adaptive cycle turbofan configurations. As evaluation cases become more time consuming, associated time benefits also improve.

Keywords: adaptive cycle engine; engine modelling; numerical simulations; optimization; root-finding technique comparison.

Introduction

Modern gas turbine engine designs that are predominantly based on conventional engine architecture are typically designed to effectively operate in single flight condition. However, as the requirements from modern flight platforms reach into operational range with multitude of characteristic conditions, such as slow-speed loiter and high-speed cruise, contemporary propulsion systems are reaching the peak of their performance within current technological limits. Consequently, the natural evolution of jet engine technology will lead the market towards novel adaptive cycle engine concepts.

This growing interest in adaptive cycle research can already be observed in the work of gas turbine industry leaders [1,2] and throughout recent academic studies [3–6]. As these preliminary studies mostly focus on the adaptive cycle thermodynamics, the most common approach is to start the research with numerical simulation of the engine components and the complete cycle. These studies usually utilize commonly available engine simulation frameworks and commercial tools. However, as these codes are generally developed for well-established aerial and power gas turbine configurations, which are typically characterized by a small number of simulation variables, development of new adaptive cycle engines with numerous control parameters creates significantly enhanced computational load.

Motivation

In an attempt to alleviate this increasingly challenging problem, the present effort focuses on the computation time optimization of various mathematical approaches that could be implemented in future advanced cycle-solving algorithms. Drawing inspiration from previously developed enhanced numerical methods, this paper compares the performance of representative solvers for a set of increasingly complex engine architectures. For each configuration, the computational efficiency of each solver is depicted using typical flight conditions. The outcomes of this work are intended to provide basis for future development of optimized modal agile simulation tools for advanced thermodynamic cycles' analysis. According to the authors' best knowledge, this is a first comparison of various mathematical methods systematically applied to the field of gas turbines with comparative findings, applicable to a generic development platform.

***Corresponding author: Beni Cukurel**, Turbomachinery and Heat Transfer Laboratory, Department of Aerospace Engineering, Technion – Israel Institute of Technology, 3200003 Haifa, Israel, E-mail: beni@cukurel.org
Michael Palman and Boris Leizeronok, Turbomachinery and Heat Transfer Laboratory, Department of Aerospace Engineering, Technion – Israel Institute of Technology, 3200003 Haifa, Israel, E-mail: p.michael@campus.technion.ac.il (M. Palman), borisl@technion.ac.il (B. Leizeronok)

Engine numerical model

During jet engine design process, each engine module is designed, manufactured and tested independently. Similar approach should be applied in engine simulation framework, where each engine segment can be represented via independent modules. Such approach creates flexible toolbox that would be used towards designing different engine models. This way, a simple turbojet engine is represented by five major modules - intake, compressor, combustor, turbine and nozzle. In same framework, conversion of turbojet to a single-spool turbofan configuration would only require integrating an additional compressor module, which will stand for the fan. The two architectures are schematically described in Figure 1. Each segment will then evaluate the inlet and outlet conditions and calculate values for various parameters. The data is shared among the modules to complete the overall cycle calculation. An exemplary solver built using this methodology and the individual component thermodynamics are described in detail in Ref. [7]. The methodology can be easily expanded to two- and three-spool machines by introducing additional turbine modules.

In order to accurately characterize the performance of different turbomachinery components (fans, compressors and turbines), the modules can be supplied with component maps that describe the operational range in terms of mass flow rates, rotational speeds, pressure ratios and efficiencies. In the case of compressor module, it is theoretically possible to have same mass flow rate but different pressure ratio (choking) or same pressure ratio with different mass rate on the same speed line. Therefore, it is impractical to use pressure ratio and mass flow rate to directly read map data and new auxiliary β-line coordinate should be introduced. β-lines are non-crossing lines that completely cover the component map and cross each speed line only once. Thereby, a monotonous injective system of spool speed and β-line coordinates can be defined. Each set of these parameters describes a unique performance point on the component map in terms of efficiency, pressure ratio and mass flow rate. Detailed procedure of rebuilding maps with β-lines is described in Ref. [7].

Engine performance simulation

After defining individual components, all engine modules need to be synchronized to correctly evaluate the integral engine performance. To satisfy conservation of mass requirements, mass flow rates throughout the engine should be matched. Then, mechanical couplings must be added between relevant components. For example, the high-pressure compressor and the high-pressure turbine are coupled by a common shaft. In turbofan configuration, the fan can be connected to the same shaft directly or via a gearbox. Alternatively, it can be coupled to a second low-pressure compressor in twin- or triple-spool configuration. Finally, correct power balance between all components should be preserved. For instance, in simple single-spool architectures, the turbine is the only power source in the engine. Therefore, in this case, it needs to support both the engine compressor and any external loads such as fan and aircraft alternator. Power can be evaluated based on enthalpy difference through power consuming or generating component, multiplied by mass flow rate. Proper mechanical efficiencies of the coupling elements (shaft and gearbox) should also be taken into account. To illustrate this concept, power demand for exemplary single-spool turbojet architecture with an alternator can be calculated from

$$PW_{turb} = \frac{PW_{comp} + PW_{alt}}{\eta_m}, \tag{1}$$

$$(\dot{m}_a + \dot{m}_f) \cdot \Delta h_{turb} \cdot \eta_m = \dot{m}_a \cdot \Delta h_{comp} + PW_{alt}. \tag{2}$$

If geared turbofan configuration is considered instead, the power requirement should be calculated from:

$$PW_{turb} = \frac{PW_{comp} + PW_{alt}}{\eta_m} + \frac{PW_{fan}}{\eta_m \cdot \eta_{gb}}, \tag{3}$$

$$(\dot{m}_a + \dot{m}_f) \cdot \Delta h_{turb} \cdot \eta_m = \dot{m}_{comp} \cdot \Delta h_{comp} + \dot{m}_{fan} \cdot \Delta h_{fan}/\eta_{gb} + PW_{alt}. \tag{4}$$

When every component and the necessary couplings are fully defined, simulation is initialized using known inlet conditions. One of the established approaches is to

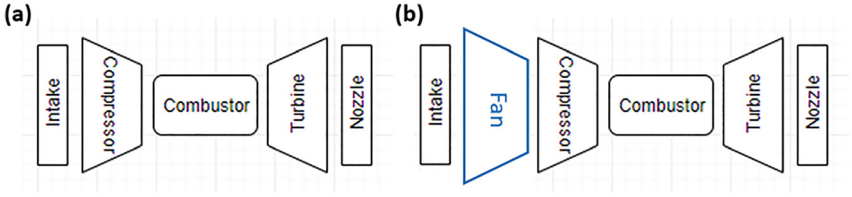

Figure 1: Engine block diagram - (a) turbojet engine, (b) single-spool turbofan engine.

select initial operating point guess on the compressor map, thereby imposing initial β-line coordinate. This β-line coordinate then becomes the first simulation variable. In following, the engine combustion chamber can be solved by estimating combustion temperature, which becomes the second variable. Now, turbine inlet mass flow rate is imposed by mass flow conservation, while turbine outlet mass flow rate is calculated based on required turbine load. The two values must match, and their difference can be used to define convergence error via two surfaces.

An illustrative example of such surfaces is charted in Figure 2, where it was calculated for ground conditions at the design spool speed of a typical turbojet engine, using representative component maps from open literature [8]. The green surface depicts turbine inlet mass flow rate, which is imposed by the compressor flow rate and fuel addition in the combustor. Turbine outlet mass flow rate is plotted using blue surface. The mass flow difference between the two surfaces at each combination of variables is the error surface, labeled as $ERROR_1$.

In following, the turbine outlet mass flow rate should be matched with the nozzle outlet mass flow rate, imposed by nozzle pressure ratio. The two mass flow surfaces are presented in Figure 2. The delta between the two values is the second convergence error, $ERROR_2$. The problem is now fully defined by two variables and two error functions, and cross-section spline between the two error surfaces can be defined, dashed red line in Figure 3. The solution converges when the errors decay below a predefined convergence criterion. When converged, the spline should cross zero plane (grey surface in Figure 3) and change sign. This point, marked with red circle in Figure 3, is the converged point of the simulation, where both errors are equal to zero. The module-by-module simulation algorithm is summarized as a block diagram in Figure 4.

Turbofan configuration can be solved in a similar manner. The simulation starts with the core of the engine, which is described like a regular turbojet. Additional load of the fan and changing core inlet conditions are taken into account. The flow that is not ingested into the core is evaluated in bypass nozzle.

This straight-forward algorithm has one prominent flaw - it is extremely high computational inefficiency. In the scope of this paper, all mass flow rate surfaces are resolved with resolution of 40 β-coordinates and temperature steps of 10 K. Thus, evaluation of each turbojet spool speed requires more than 2500 calculations. Consequently, a single turbojet operating line that consists of 25 spool speeds takes 62 s to converge using MATLAB on a modern computer equipped with Intel Core i7-8700 CPU and 32 GB DDR4 2667 MHz RAM. Considering that turbojet has a single control variable (fuel flow rate, which imposes the spool

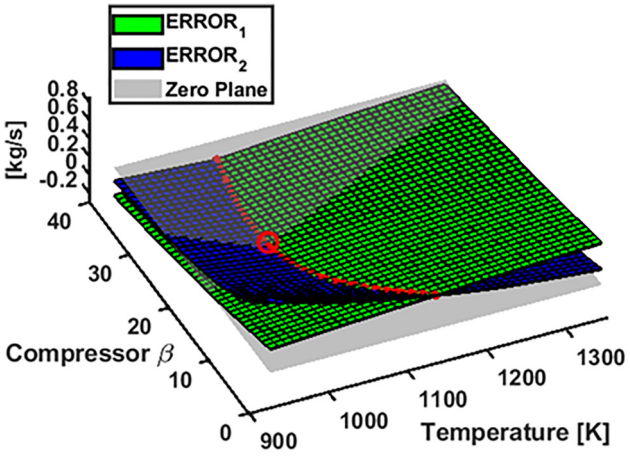

Figure 3: Two error surfaces and the converged point of the simulation algorithm.

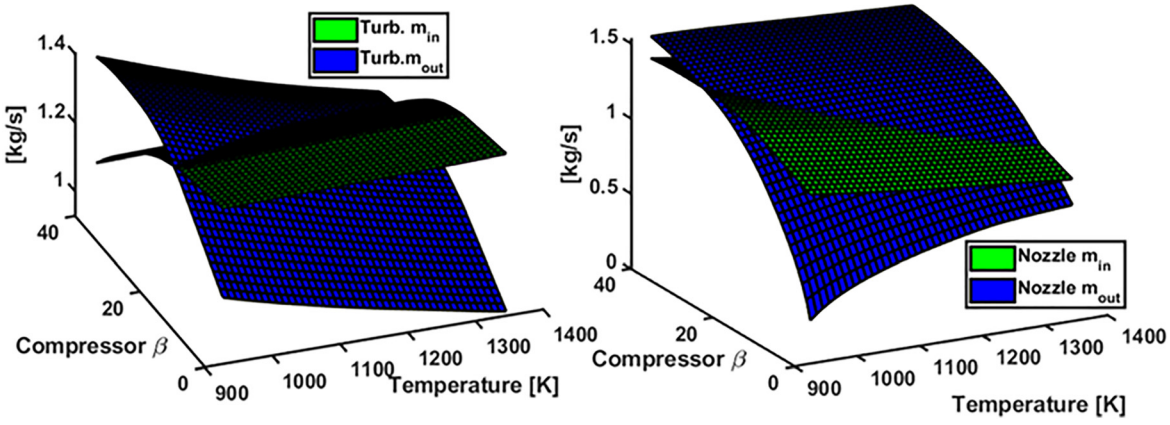

Figure 2: Turbine inlet and outlet (left), nozzle inlet and outlet (right) mass flow rates.

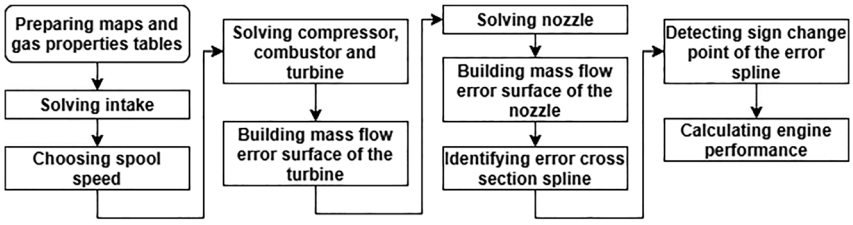

Figure 4: Module-by-module simulation approach.

speed), introduction of additional control parameters would exponentially increase computational complexity, resulting in simulation times beyond reasonable. Therefore, in order to simulate future adaptive cycles with numerous degrees of freedom, present approach is impractical and different path should be adopted.

Surrogate and particle swarm optimization model

In an attempt to resolve the high computation time problem, one of the logical approaches is to convert the simulation into an optimization problem. Then, the goal of the problem becomes to find the minimum in multidimensional error surface, which is defined as the Euclidian norm of the two error functions. Minimal error point is considered converged if this vertex is below the predefined allowable error. An exemplary error surface is depicted in Figure 5, with the solution marked with red circle. The advantage of this approach is that it significantly simplifies simulation programming as only the variables' ranges and engine error function need to be prepared. The algorithm of this approach is described in Figure 6.

Additional benefit of this method lies in the fact that many sub-routines already exist for various optimization schemes. For example, one of the most advanced and popular schemes, known as the "Surrogate Optimization", has premade implementations in MATLAB, C++, Python and other programming environments. This method, described in details in Refs. [9–12], has two major phases - creation of a surrogate surface and evaluation of the built surface minimum. At first, the algorithm takes samples of an error surface at random points within the bounds. Based on this sampling, and using interpolation, it creates surrogate error surface. Then, the algorithm searches for a minimum of the created surface by sampling several thousand random points within the bounds. Next, error values on the error surface are calculated at the same points. Based on the differences between the surrogate and error surfaces, best points are chosen to iteratively update the surrogate surface. The iterations are stopped when the found minimum falls into prescribed allowable error bounds.

Using MATLAB, implementation of this function in the optimization problem resulted in calculation time of 47.1 s. To speed up calculations, parallel computing can be used to divide a given job to several data packages and solve them simultaneously using different processor cores. However, in the present case, it increased the simulation time to 79 s as the problem division and preparation of data packages for each core also consumes significant time. Therefore, parallelization remains beneficial only in higher-dimension problems.

In a bid to further reduce optimization time, less numerically demanding particle swarm optimization can be considered. Particle swarm is a population-based algorithm, described in details in Refs. [13–15]. Calculation starts with spread of "particles" over the error surface inside prescribed bounds. Then error is evaluated for each particle and they get prescribed velocity based on the

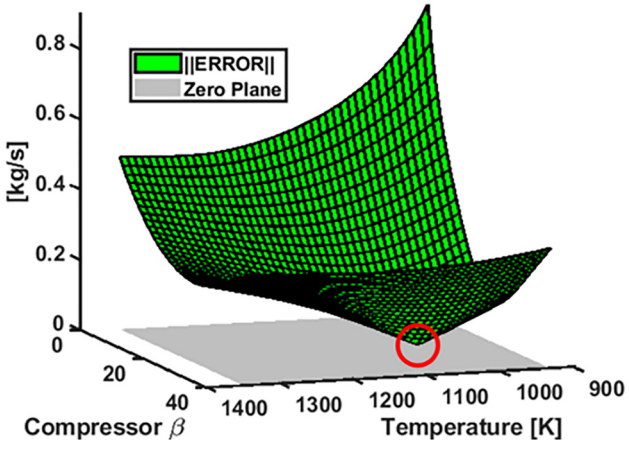

Figure 5: Error norm in an optimization problem.

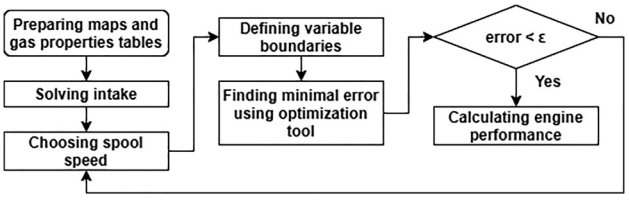

Figure 6: Optimization problem algorithm.

distance travelled at each iteration. After movement on the error surface each particle is re-evaluated. The particles are attracted to some degree to the best location they have found individually and to the best location found by any member of the swarm. Using MATLAB, implementation of this function in the optimization problem resulted in calculation time of 21.6 s. Similar to first optimization attempt, parallel computing resulted in calculation time increase to 34.1 s.

Newton-Raphson algorithm

Although optimization approach significantly reduced individual calculation time, introduction of additional engine control parameters would still result in significant computational load. Therefore, instead of solving the engine model as an optimization problem, an attempt can be made to improve the solution efficiency by modifying and optimizing the solver for root-finding formulation. Instead of resorting to grid search over entire error surfaces, thereby obtaining zero-point through numerous calculations, several efficient numerical methods can be implemented.

One of the most efficient ways to numerically find the function root is the Newton - Raphson (NR) method [16]. It is a powerful tool that solves equations based on the idea of linear approximation. This approach relies on the notion that any function can be rewritten using Taylor series expansion:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n. \tag{5}$$

If x_0 is an estimated root of the function, shortening the series to first two significant terms results in:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) = 0. \tag{6}$$

This way, the root can be obtained using an iterative procedure

$$x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}, \tag{7}$$

where the function derivative can be found using forward-stepping finite difference formula. Multivariable NR method is direct extension of the single variable formulation. This time, it can be used to solve system of equations that has the form of

$$\begin{matrix} f_1(\overline{x}) = 0 \\ \vdots \\ f_n(\overline{x}) = 0 \end{matrix}, \tag{8}$$

where $\overline{x}$ is the variables vector $\overline{x} = [x_1, \ldots, x_n]$. In this formulation, partial function derivatives take the form of Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{9}$$

Iterative procedure for a system of multivariable nonlinear equations is represented by

$$x^{(i+1)} = x^{(i)} - J^{-1}(x^{(i)}) \cdot f(x^{(i)}). \tag{10}$$

The advantages of the NR method are its fast convergence and simplicity of implementation. However, this method has several disadvantages, one of which is the absence of stopping criteria in cases with no solution. Therefore, to prevent infinite loop, existence of a solution in computational domain should be confirmed before execution of this technique.

Computational domain

Computational domain can be defined by subtraction between the two error surfaces. As the spool speed decreases, nozzle inlet pressure drops. If it reaches sub-atmospheric values, solution no longer exists, limiting the range of feasible computations. The possible solutions range is further limited by β-coordinate, which not only further affects nozzle pressure, but also imposes minimal combustor inlet temperature. The aggregate influence of the two effects results in a shift of computational domain towards top-right corner, Figure 7.

The suggested process of finding the computational domain boundaries is illustrated in Figure 8 (left) for representative flight conditions of Mach 0.3 and altitude of 5000 m, and relative spool speed of 0.38. At first, the theoretical area of all β-lines and temperatures is truncated to only include possible turbine inlet temperatures for given spool speed. Then, the remaining region is covered by fanning auxiliary beams, originating from bottom-left corner, Figure 8 (left). Along each beam, interactive probing is used to identify the computational field's boundaries, magenta marks in Figure 8 (left). As each beam can only cross the boundary once or twice at best, the crossings provide lower and upper bounds of the computational domain. Together, the found points form the computational polygon of the simulation.

In the exemplary case of Figure 8 (left), 13 beams are crossing the region and the polygon consists of 26 vertices.

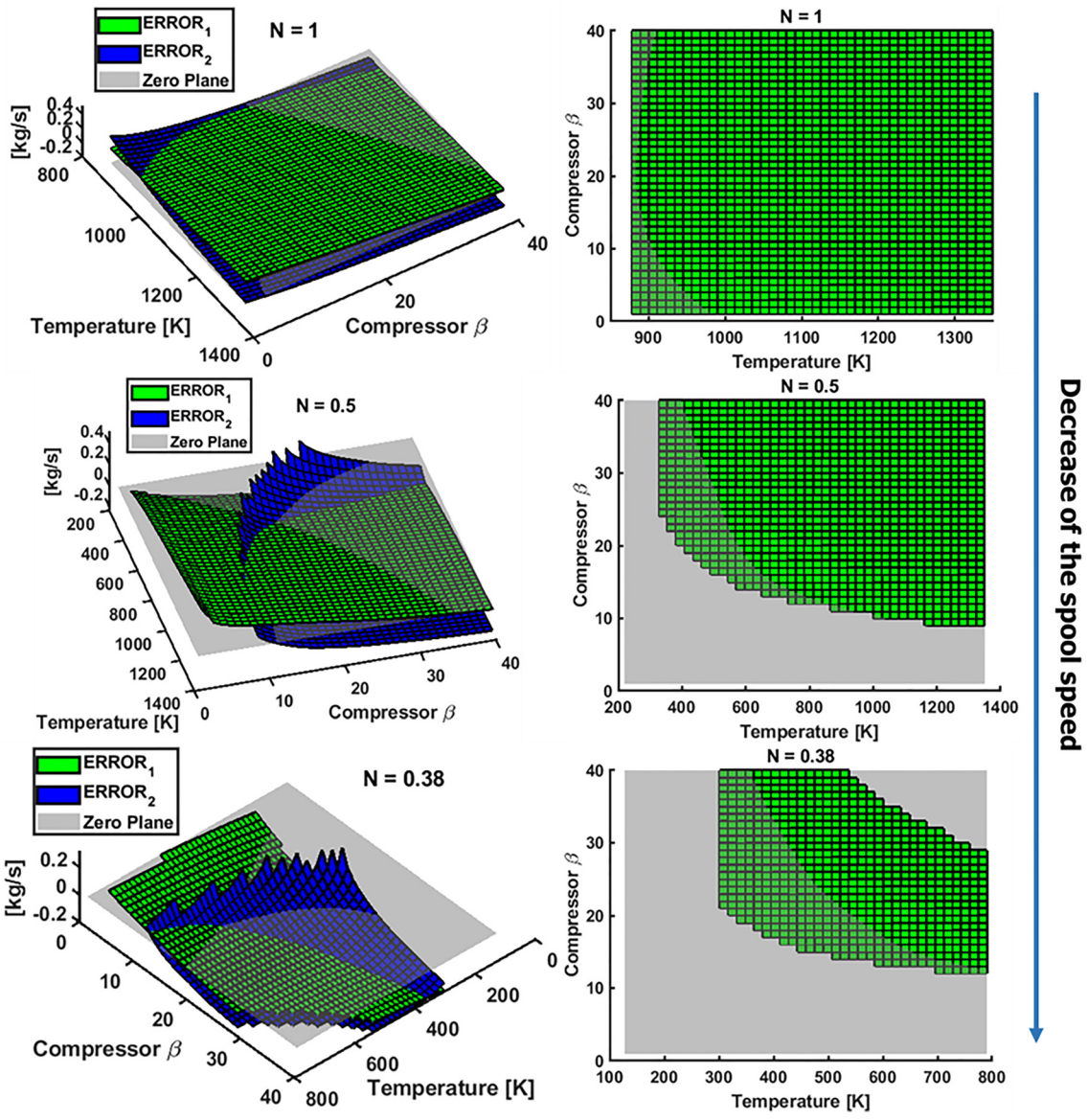

Figure 7: Error surface and computational domain with respect to changes in spool speed for representative flight conditions of Mach 0.3 and altitude of 5000 m.

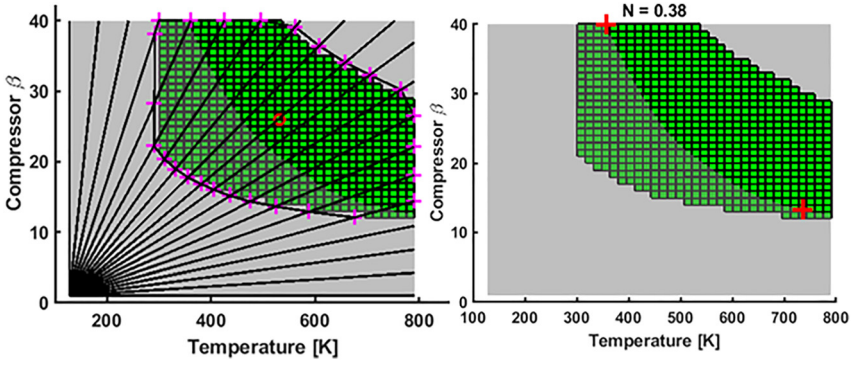

Figure 8: Example of computational domain polygon identification (left) and edges of the zero-plane spline (right) at M = 0.3, H = 5000 m, N = 0.38.

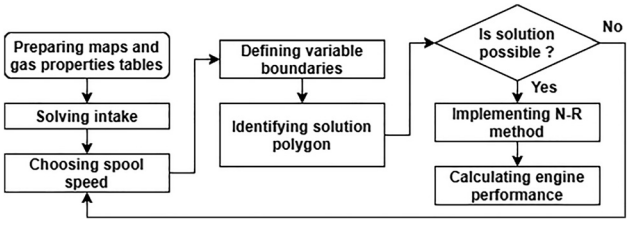

Figure 9: NR simulation algorithm for turbojet engine.

If all of these points have the same sign, the two error surfaces do not have a crossing in the computational domain and the solution does not exist, allowing the algorithm to quickly shift to next spool speed. Else, the edges of the spline that lies on the zero plane can be identified, Figure 8 (right), defining the crossing between the two error surfaces. Following same logic, if the sign of the two end points is the same, there is no solution. Therefore, only if the curve edges have opposite signs, solution exists and it must lie within the computational polygon.

As present work relies on iterative solvers, this check negates the risk of running into infinite iteration loop, which is inherent in initial condition-sensitive iterative solvers. This boundary-defining method is also useful in the frameworks of engine control systems and operational envelope definitions, where the boundaries of engine operation are of larger interest rather than its actual performance.

After identifying the computational polygon and confirming solution existence, NR method can be implemented in confidence. Center of the field, red circle in Figure 8 (left), is a good first guess for initializing the simulation. The complete algorithm is described in Figure 9. Implementation of this algorithm in MATLAB yields 4.3 s simulation time for single-spool turbojet.

Broyden's method

One of the numerical disadvantages of the NR method, is the inherent requirement to reevaluate the Jacobian matrix during each iteration. For systems with n variables, Jacobian matrix has the size of $n \times n$. Therefore, n^2 derivatives are needed to reconstruct the matrix during each iteration. As the derivatives are calculated using finite differences, Jacobian matrix construction can entail up to $2n^2$ error function evaluations. The actual number of calculations will be somewhat lower, as engine model provides two errors simultaneously and the previously calculated point can also be retained. To ease this computational burden, it is possible to implement a quasi-Newton technique. One of the most successful and commonly used quasi-Newton methods was developed by Broyden [17]. This approach requires Jacobian matrix calculation only once when the iterative procedure is launched. Then, in the process of finding the root for a given problem, Jacobian matrix is approximated from previous iteration, significantly reducing the number of computations. The iterative procedure for Broyden's method is identical to that used in NR method, with the only difference of using approximation matrix B instead of Jacobian matrix:

$$x^{(i+1)} = x^{(i)} - B^{-1}\left(x^{(i)}\right) \cdot f\left(x^{(i)}\right). \tag{11}$$

The approximation matrix is defined as:

$$B^{(i)} = B^{(i-1)} + \frac{y^{(i)} - B^{(i-1)}s^{(i)}}{\|s^{(i)}\|_2^2}\left(s^{(i)}\right)^T, \tag{12}$$

where

$$y^{(i)} = f\left(X^{(i)}\right) - f\left(X^{(i-1)}\right),\ s^{(i)} = X^{(i)} - X^{(i-1)}. \tag{13}$$

Implementation of Broyden's method in MATLAB for same engine architecture further reduced simulation time to 4.09 s.

Grid resolution

Beyond the efforts to reduce calculation times for root-finding problem, major part of the computational burden

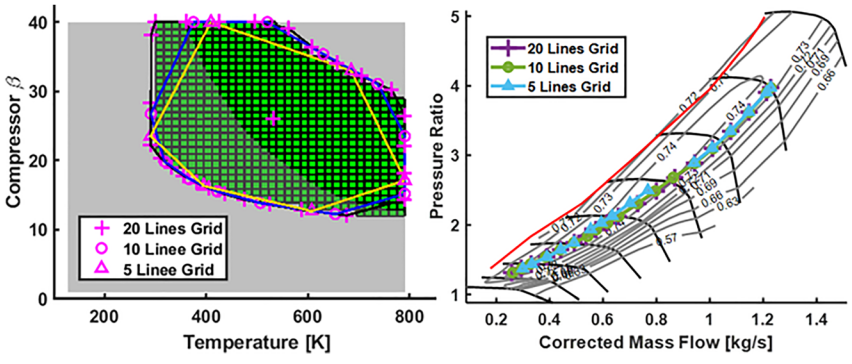

Figure 10: Computational polygon resolution comparison. Polygon at N = 038 (left) and operating lines (right) at M = 0.3, h = 5000 m.

lies on solution existence validation. The number of calculations that are used to identify computational is directly proportional to resolution of auxiliary beams. Therefore, coarse resolution can significantly decrease simulation time, however it should be approached with caution as insufficient beam number would be detrimental to simulation performance. Figure 10 (left) presents comparison of found computational field for grid resolutions of 20, ten and five lines. It can be seen that while reduction to 10 lines still results in reasonable polygon shape, highly inaccurate shape can be observed after further reduction to five line resolution. This is reflected in engine simulation performance. When comparing operating line on compressor map as found by the simulation for different auxiliary beam resolution, Figure 10 (right), ten-line values correlate well to higher resolution data, whereas there are visible breakdowns in the five-line simulation that couldn't converge in several cases.

Muller's method

Additional approach to reduce calculation time during solution existence validation can be by focusing on improving the edge search for cross section line. Instead of using simple bisection method, which is known to be one of the less efficient root finding methods, a faster convergence technique can be implemented. One of the most agile methodologies is Muller's Method [18]. This algorithm starts with an initial guess of three points. Then, parabola is constructed through these locations. One of the parabola roots becomes the next root estimation. In following, value of the error spline at the new root is used to construct new parabola. The procedure is repeated until convergence. Overall, the iteration procedure is described as

$$X^{(i+1)} = X^{(i)} - \frac{2C}{B \pm \sqrt{B^2 - 4AC}}, \tag{14}$$

where A, B and C are the parabola coefficients that can be calculated according to

$$A = qF\left(X^{(i)}\right) - q(1+q)F\left(X^{(i-1)}\right) + q^2F\left(X^{(i-2)}\right),$$

$$B = (2q+1)F\left(X^{(i)}\right) - (1+q)^2F\left(X^{(i-1)}\right) + q^2F\left(X^{(i-2)}\right),$$

$$C = (1+q)F\left(X^{(i)}\right), \tag{15}$$

$$q = \frac{X^{(i)} - X^{(i-1)}}{X^{(i-1)} - X^{(i-2)}}. \tag{16}$$

Clear advantage of Mueller's method can be demonstrated using exemplary case study of Figure 8 (right). Figure 11 describes iterative procedure for the upper and lower spline edges search along polygon boundary, using both bisection and Muller's methods. In both cases, Muller's method is significantly advantageous to classical bisection approach. During upper edge search, Muller's algorithm converged within three steps, whereas bisection method required 14 iterations. Same tendencies are present in lower edge calculation, where it took Muller's technique only four iterations to identify the edge compared to 14 steps of bisection convergence.

Optimized algorithm validation

The proposed numerical approaches can be implemented into optimized simulation tool, which can be validated against commercial GasTurb13 software. Same single-spool turbojet engine is modeled in GasTurb13 and the results of the commercial and in-house codes are compared in Figures 12 and 13 in terms of operating line, thrust rating and fuel consumption respectively. The findings are in close agreement with 1.35% average deviation in operating line, 1.84% average deviation in thrust rating and 0.64% average deviation in fuel consumption. This deviation magnitude suggests that the algorithm is

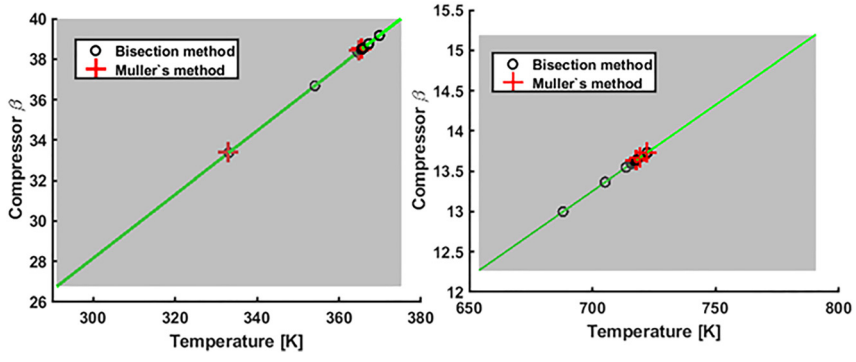

Figure 11: Search for the upper (left) and lower (right) spline edge (M = 0.3, H = 5000 m, N = 0.38).

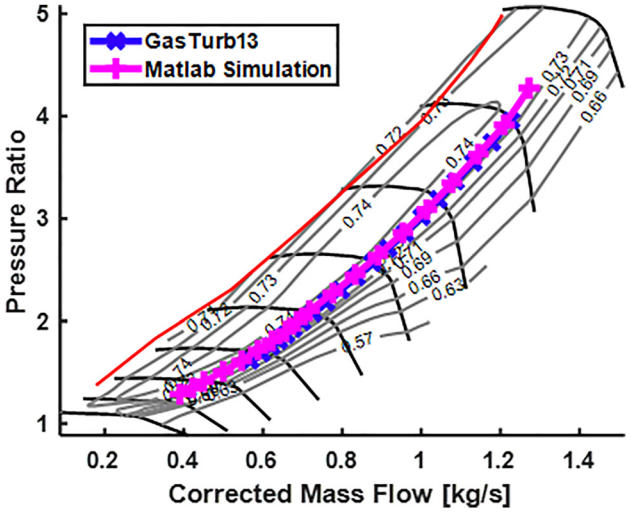

Figure 12: GasTurb13 and in-house code operating line comparison.

capable of accurately resolving the cycle performance. Overall, the suggested simulation approach provides reliable outputs and can be used to outline the benefits of each numerical method in significantly more complex cycles.

Algorithms comparison in adaptive cycle simulation

As present effort focuses on exploring the paths towards high-speed modular advanced cycle simulation, having already compared the performance of the different methods in single-spool turbojet scenario, same methods must be compared in higher order models. One of such previously conceptualized architectures is an adaptive cycle turbofan with variable bypass and gearbox ratio. Compared to conventional turbojet, which can only variate its fuel flow rate via throttle, this design, described in detail in Ref. [6], has two additional control parameters (gearbox ratio and variable bypass nozzle).

Towards results consistency, all comparisons are done for same flight conditions using single CPU core on same hardware and software. The NR, Broyden, NR coupled with Muller and Broyden coupled with Muller approaches are evaluated for both engine configurations using 10- and 20-line auxiliary beams grid. The simulation times for all cases are summarized in Table 1.

It is interesting to note that although Broyden's method is known to have slower convergence rate then NR, it achieves lower computational load due to estimation of Jacobian matrix. Exemplary solution path for both Broyden's and NR algorithms is presented in Figure 14 using the previously discussed case study. In this case, NR algorithm converged in four iterations. In comparison, although Broyden's method converged in 11 steps, due to Jacobian matrix evaluation only in the starting path point, reduced number of error evaluations directly translates to decreased simulation time in both engine configurations. As expected, further time reduction is achieved by reducing auxiliary grid resolution and implementing Muller's method during solution existence validation.

Table 1: Root finding approaches simulation times (in CPU time units).

		Turbojet	Adaptive Cycle Turbofan
10 lines grid	**Newton-Raphson**	2.46 s	25.9 min
	Broyden	2.34 s	24.2 min
	Newton-Raphson & Muller	2.25 s	22.8 min
	Broyden & Muller	2.21 s	22.4 min
20 lines grid	**Newton-Raphson**	4.3 s	35.1 min
	Broyden	4.09 s	34.3 min
	Newton-Raphson & Muller	3.84 s	33.9 min
	Broyden & Muller	3.78 s	33.7 min

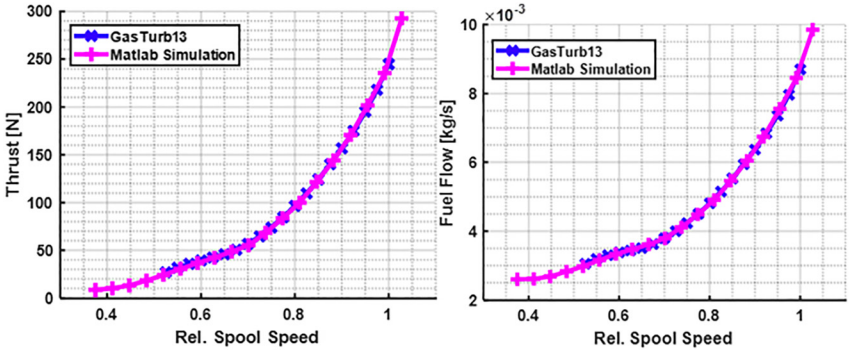

Figure 13: GasTurb13 and in-house code thrust rating (left) and fuel consumption (right) comparison.

To further highlight the difference between NR and Broyden methods, additional complexity can be added to engine architecture. For example, addition of recuperation to the cycle will impose another computational variable into the algorithm. A simplified heat exchanger model can be added according to Ref. [19].

Based on recuperator design condition, off-design efficiency and pressure losses on the cold and the hot sides can be evaluated. Efficiency is defined as

$$\eta = 1 - \frac{\dot{m}}{\dot{m}_{ds}}(1 - \eta_{ds}), \tag{17}$$

cold side pressure loss is evaluated from

$$\frac{P_{in} - P_{out}}{P_{in}} = \left(\frac{P_{in} - P_{out}}{P_{in}}\right)_{ds} \frac{\left(\frac{\dot{m}_{in}}{P_{in}}\right)^2 \frac{T_{out}^{1.55}}{T_{in}^{0.55}}}{\left(\frac{\dot{m}_{in}}{P_{in}}\right)^2_{ds} \frac{T_{out,ds}^{1.55}}{T_{in,ds}^{0.55}}}, \tag{18}$$

and hot side pressure loss is calculated via

$$\frac{P_{in} - P_{out}}{P_{in}} = \left(\frac{P_{in} - P_{out}}{P_{in}}\right)_{ds} \frac{\dot{m}_{in}^2 T_{in}}{\left(\dot{m}_{in}^2 T_{in}\right)_{ds}}. \tag{19}$$

In recuperated engine cycle, combustor inlet temperature, which is the heat exchanger cold side outlet, is the new simulation variable. Evaluated off-design efficiency provides hot side inlet temperature directly from heat exchanger efficiency definition

$$\eta = \frac{T_{cold,out} - T_{cold,in}}{T_{hot,in} - T_{cold,in}}. \tag{20}$$

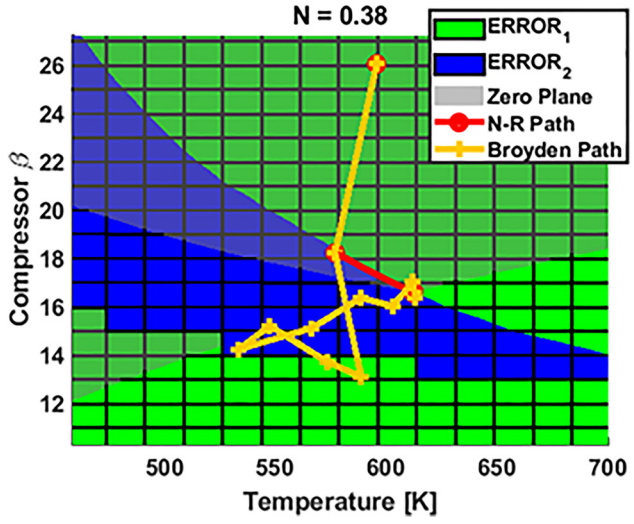

Figure 14: Converged solution paths of Broyden's and NR methods (M = 0.3, h = 5000 m, N = 0.38).

After solving the combustion chamber and the turbine modules, iterated recuperator efficiency is compared to initial estimate. The difference between the two values is additional simulation error. Thus, engine model formulation Should now be described by three variables and three error functions. In this case, the size of Jacobian matrix becomes 3×3, significantly increasing the number of error function evaluations.

This effect of adding heat exchanger module is evaluated in both engine configurations. Design point efficiency of 80% with 3% pressure drop on both cold and hot side is assumed. The recuperated cycles are evaluated only using 10-line auxiliary beams grid. Outcome of this investigation is summarized in Table 2.

It can be observed that despite increased size of the Jacobian matrix, due to pressure losses in the heat exchanger, the simulation times are close to non-recuperated engines. As pressure loss is directly proportional to engine flow rate, higher spool speeds impose higher pressure loss. This phenomenon significantly reduces the number of converged operating points, with unsuitable points filtered out by solution existence check. However, regardless of this effect, an increasing gap between NR and Broyden's methods can still be registered.

To highlight the usefulness of this simulation approach, preliminary analysis of conceptual micro-turbojet to micro-turbofan conversion project can be considered. Using the same engine core, this would require increased power extraction from the turbine to support the fan stage. In this scenario, increased core inlet pressure would not only allow higher turbine power, but would also result in higher thermodynamic cycle efficiency. Although large-scale turbofan engines typically have booster stages to achieve this effect, a different concept can be adopted in this hypothetical micro gas turbine project, where small spatial scales could potentially prevent booster installation. Instead, hub loaded fan with higher hub pressure ratio can perform as a booster stage. In this atypical case, the fan hub and tip regions will have separate fan maps,

Table 2: Simulation time of recuperated engine configurations (in CPU time units).

	Turbojet with HEX	Adaptive Cycle Turbofan with HEX
Newton-Raphson	2.44 s	25.2 min
Broyden	2.27 s	23.1 min
Newton-Raphson & Muller	2.17 s	22.7 min
Broyden & Muller	2.1 s	22 min

which need to be individually accounted by separate sets of β-coordinates, leading to increased simulation times.

To emphasize the differences between the solvers in this case, they can be evaluated in realistic mission design space. In this case, the engine model needs to be solved for ranges of Mach numbers and altitudes. A realistic flight envelope covers Mach range of 0–0.9 and altitude range of 0–9 km and can be resolved with Mach number steps of 0.1 and altitude steps of 1 km. Thus, the design space would result in 81 distinct flight conditions. Knowing the representative solution time for each flight condition allows evaluation of simulation time for the complete envelope. The results for each solver are summarized in Table 3. Clearly, in this case, simulation time is of tremendous importance and optimized simulation methodology becomes increasingly crucial tool.

It is important to note that although all cases were compared using MATLAB towards simplicity and faster implementation, it is an interpretive programing language, which is relatively slow. For example, it is known to be up to 11 times slower than C++ programing language [20, 21]. Thus, using compiled programming language and further optimizing the code will lead to significantly reduced simulation times akin to those, which are typical in various commercial cycle simulation codes. Regardless, the benefits of the presented methods will also scale accordingly.

Summary and conclusions

The present paper focuses on comparison of various numerical approaches in the framework of simulation development for increasingly complex engine models. Starting with outlining the numerical challenges of typical modular engine simulation tool, several paths are suggested to improve computation times for engine operating point. At first, the engine model is solved as an optimization problem using surrogate and particle swarm approaches. Although this formulation reduces simulation time, the outcomes are insignificant on the grandscale. Moreover, parallelization attempts result in higher times when compared to single core formulation.

Table 3: Adaptive cycle turbofan with booster flight envelope simulation.

	Adaptive Cycle Turbofan with Booster	Flight Envelope Simulation
Newton-Raphson	18.93 h	58.37 day
Broyden	17.9 h	55.19 day
Newton-Raphson & Muller	17.16 h	52.91 day
Broyden & Muller	16.6 h	51.18 day

In following, several suggestions are made to enhance the classical solution procedure. NR algorithm is implemented to shorten the root finding time. To further reduce computational load, a solution existence check methodology is implemented and NR is replaced with significantly faster Broyden's procedure. Additional improvement is done by reducing auxiliary grid resolution and applying Muller's technique during computational domain identification phase. The combined effect of Muller's and Broyden's techniques reduces the overall flight envelope simulation time by 3.3, 7.3 and 12.3% when compared to combination of Newton-Raphson and Muller's methods, only Broyden's method and only Newton-Raphson method, respectively.

The impact of sequential code improvement is highlighted in representative single-spool turbojet and adaptive cycle turbofan configurations. As adaptive cycle engine has additional gear and bypass ratio control parameters when compared to the turbojet architecture, which can only variate spool speed, the increased model complexity results in significantly increased simulation times. The engine model is then further expanded to include recuperation. Although addition of heat exchanger imposes another degree of freedom, the pressure losses reduce the number of converged engine operating points. Therefore, due to solution existence validation procedure, which filters un-convergable points prior to simulation start, the computation time remains unchanged. Finally, micro-turbofan with hub loaded fan is considered and is used to highlight the code differences in realistic design space.

According to the best knowledge, the present work is the first quantitative comparison of individual solvers and their cumulative effect, particularly geared towards advanced gas turbine cycle simulations. It is the hope of the authors that the findings of this study would help establish the path towards future optimized simulation codes, regardless of the selected implementation framework.

Author contributions: All the authors have accepted responsibility for the entire content of this submitted manuscript and approved submission.
Research funding: The present research effort was partially supported by the U.S. Office of Naval Research Global under award number N62909-17-1-217; Peter Munk Research

Institute under award number 110101; and the Bernard M. Gordon Center for Systems Engineering under award number 1017930. Also, supported by Minerva Research Center (Max Planck Society Contract No. AZ5746940764).
Conflict of interest statement: The authors declare no conflicts of interest regarding this article.

References

1. WILLIAM K. Pratt and Whitney receives $437M for continued adaptive engine development. Available from: https://www.sae.org/news/2018/09/pratt–whitney-receives-usd437m-for-continued-adaptive-engine-development.
2. Rolls-Royce. Rolls-Royce North American Technologies, Inc. Selected by U.S. Air Force to complete ADVENT research demonstrator Program. Available from: http://www.defense-aerospace.com/articles-view/release/3/109107/rolls_royce-selected-for-advent-demonstrator.html.
3. Kowalski M. Adaptive jet engines. J KONES Powertrain Transport 2011;18.
4. Patel HR, Wilson DR. Parametric cycle analysis of adaptive. AIAA Propul Energy 2018. 2018 Joint Propulsion Conference, July 9-11, 2018, Cincinnati, Ohio, US. https://doi.org/10.2514/6.2018-4521.
5. Visser MOWPJ, Kogenhop O. A generic approach for gas turbine adaptive modeling. J Eng Gas Turbines Power 2006;128:13–19.
6. Palman M, Leizeronok B, Cukurel B. Mission analysis and operational optimization of adaptive cycle microturbofan engine in surveillance and firefighting scenarios. J Eng Gas Turbines Power 2019;141. https://doi.org/10.1115/1.4040734.
7. Kadosh K Cukurel B. Micro-turbojet to turbofan conversion via continuously variable transmission: thermodynamic performance study. ASME J Eng Gas Turbines Power 2017;139. https://doi.org/10.1115/1.4034262.
8. Lichtsinder M, Levy Y. Jet engine model for control and real-time simulations. J Eng Gas Turbines Power 2006;128:745–53.
9. Powell MJD. The theory of radial basis function approximation in 1990. In: Light WA, editor. Advances in numerical analysis, volume 2, wavelets, subdivision algorithms, and radial basis functions. Oxford: Clarendon Press; 1992:105–210 pp.
10. Regis RG, Shoemaker CA. A stochastic radial basis function method for the global optimization of expensive functions. Inf J Comput 2007;19:497–509.
11. Wang Y, Shoemaker CA. A general stochastic algorithm framework for minimizing expensive black box objective functions based on surrogate models and sensitivity analysis. 2014. arXiv:1410.6271v1. Available from: https://arxiv.org/pdf/1410.6271.
12. Gutmann H-M. A radial basis function method for global optimization. J Global Optim 2001;19:201–27.
13. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. 1995:1942–5.
14. Mezura-Montes E, Coello CA. Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm and Evolutionary Computation 2011;1:173–94.
15. Pedersen ME. Good parameters for particle swarm optimization. Luxembourg City, Luxembourg: Hvass Laboratories; 2010.
16. Ryaben'kii VS, v Tsynkov S. A theoretical introduction to numerical analysis. Boca Raton, Florida, US: CRC Press; 2006:243 p. ISBN 9781584886075.
17. Broyden CG. A class of methods for solving nonlinear simultaneous equations math. Math Comput 1965;19:577–93.
18. Muller DE. A method for solving algebraic equations using an automatic computer. MTAC; 1956. p. 208–15. https://doi.org/10.1090/s0025-5718-1956-0083822-0.
19. Walsh PP, Fletcher P. Gas turbine performance. Hoboken, New Jersey, US: Blackwell Publishing; 2004.
20. Gouy I. The computer language benchmarks game. Available from: https://benchmarksgame-team.pages.debian.net/benchmarksgame.
21. Aruoba SB, Fernansez-Villaverde J. A comparison of programming languages in macroeconomics. J Econ Dyn Control 2015;58:265–73.