

REAL TIME CAPABILITY FOR AN OFF-THE-SHELF FULL THERMODYNAMIC ENGINE PERFORMANCE COMPUTER PROGRAM

M. Klobe, C. Riegler and K. Salchow

MTU Aero Engines GmbH
Dachauerstrasse 665, 80995 München, Germany

Key words: Turbo engine, performance, full thermodynamic, real time.

Abstract: Engine performance models running real time are a key feature in many applications, e.g. for the propulsion system simulation in aircraft crew training aids or for the development and testing of engine control system software. Conventional approach to provide real time capability consists in the creation of simplified models based on a specific engine data set and derived from high fidelity full thermodynamic engine performance computer programs. These programs are used by manufacturers for engine development purposes anyway. They are able to handle a wide range of engine configurations and applications, e.g. engine design and off design calculations as well as engine test data analysis. Recent work at MTU Aero Engines GmbH is aiming at reducing the calculation time consumption of the in-house engine performance program in order to allow direct use of this already available and validated program for real time applications.

The high calculation time demand of full thermodynamic performance programs is mainly due to the matching of the component performance and therefore to the need to solve a system of non-linear equations. This system is usually solved by using Newton-Raphson or Broyden type iterative techniques requiring a time consuming thermodynamic calculation through all the engine components for each iteration loop.

The efforts in achieving real time capability described in this paper are focussing on output reduction, on the reduction of the calculation time demand for one iteration loop and on the improvement of the solver algorithm in order to reduce the number of iteration loops. Combining all these measures enables to perform calculations real time on a typical medium power CPU. This is demonstrated for a full thermodynamic engine model running in a crew training aid application.

1 INTRODUCTION

1.1 Real time applications

During the activities for design, development, entry-into-service and in-service of propulsion systems, especially of turbo engines for aircraft application, engine performance models are used for a variety of purposes. Besides the task to provide and analyze steady-state performance data for all relevant flight conditions and power settings, modelling and simulating the transient operation of the engine is increasingly important. A subset of transient simulation applications are those where a simulation model of the engine is coupled directly to any kind of device acting in real time. The latter could be for example a human being or some hard-

ware which is not simulated but physically integrated in the simulation loop. Those applications require the simulation solution for a single operating condition within a time frame set by the real time device, and hence in real time. Some typical examples for real time simulation applications are

- aircrew training aids where the aircraft and propulsion system behaviour are simulated in real time to provide data to the man-machine interface, see [2],
- control system testing where the control system hardware is tested in the loop with a simulation of the engine behaviour or
- performance seeking control activities where on-board engine performance models are used to provide not directly measured data as input to the control system in real time (embedded models).

A very profound overview of these applications can be found in [7].

When it comes to define the term ‘real time capability’ in conjunction with turbo engine performance models there is not only the calculation time but a number of other requirements which have to be fulfilled. Of course the simulation has to provide the system states and outputs at a given update rate, but also the results have to fulfil accuracy requirements and as a very important issue the simulation has to be absolutely stable in terms of numerics, i.e. the simulation has always to come back with a reasonable result. Due to the requirement of deterministic calculation time the time integration procedure used in real time models does usually abandon the option of using error estimation and time step control mechanisms. Hence in real time models one time step does require the calculation of one single operating point only.

When thinking about turbo engine design and development work real time is not a real boundary engineers like to be limited to. A variety of pure simulation applications without direct coupling to a real time limited device would benefit from simulation models running faster than real time. For example in control system development work a faster than real time simulation of the engine performance would allow for studying a greater number of control schedule options and thus for finding a better solution for a given development time and budget frame.

1.2 Available methods

The classical approach to establish real time capable engine performance models is based on either transfer function models or on piece-wise linear models usually implemented as state-space models, see [7] and [9]. Despite the advantage of being real time capable there are major drawbacks of these approaches, see [2] for details. As far as consistency is concerned, these model types face problems in all operating regimes apart from idle to maximum power. As for validation and maintainability usually big effort is necessary with these somewhat artificial model types.

Recently some work has been presented in open literature dealing with real time application of full thermodynamic engine performance models. For example in [1], [4] and [10] major achievements are presented by using full thermodynamic engine performance models within real time simulation even for complex engine configurations. Still there can be found some drawbacks in these references. In [4] and [10] the thermodynamic models are compromised to a certain extent compared to the models used for non real time applications in favour of reducing calculation time. The numerical method used in [10] to speed up the iterative solution of the full thermodynamic model is to pre-calculate the inverse Jacobian matrix as a function of corrected gas generator speed and to store the results as input for the dynamic simulation. This implies additional work which has to be repeated every time the model status changes. In

addition the simulation results presented in [10] show only poor accuracy in some operating conditions due to the concept of storing the inverse Jacobian matrix as a function of corrected gas generator speed only. Especially with variable geometry driven towards an end position in case of malfunctions this approach will imply numerical problems. A pre-calculated inverse Jacobian also means that for a new application with another iteration scheme necessary the pre-calculation has to be performed again. In [4] one way to reduce calculation time per time step is to get rid of equations in the iteration scheme which do not have a major contribution to the solution. This method introduces only small deviations compared to the non real time solution for the test case shown, but applied to malfunctions the relevance of the neglected equations might change. Another feature described in [4] is the use of pre-calculated partial derivatives between parameters which are guessed during iteration, and simulation input parameters, to improve variable guesses. As in [10] this implies additional work which has to be repeated every time the model status changes and will impose problems in case of malfunctions. In [1] no compromise in the physical content of the thermodynamic model is mentioned. The means to reduce calculation time is a Broyden/Powell type update of the inverse Jacobian without any pre-calculation. In addition the maximum number of loops per time step is limited to three. In [8] pretty much the same methods and results are reported than in [1], but in [8] a performance computer program especially implemented for the application considered is used.

1.3 Approach presented in this paper

The main idea of the approach presented in this paper is to overcome the drawbacks in real time engine performance models presented in open literature so far by using a highly accurate full thermodynamic engine performance model based on an off-the-shelf performance calculation software directly in real time applications without any deviation to the software used in non real time applications. To this end the off-the-shelf performance calculation software has to be modified and qualified to run in real time. As a result of this approach in real time and in non real time dynamic simulations

- both performance calculation software and engine performance model are identical,
- accuracy and numerical stability requirements are identical,
- no engine model specific pre-calculated data is necessary to enhance convergence and
- the number of loops per time step has not to be limited at all, or at least the simulation will run to such a limit only in a very few number of cases.

With respect to now-a-days desktop CPUs available and taking into account the statements found in [1] and [4], the authors think that we are just on the edge to achieve real time capability taking into account these prerequisites. Of course every year in microprocessor development will relax the critical issues which might exist today. The major advantages of this kind of approach to provide real time engine performance models are

- high fidelity results, i.e. simulation results are identical in real time and non real time application,
- high flexibility compared to piece-wise linear models or any kind of pre-calculated data based models, e.g. in case of simulating malfunctions,
- low validation and maintenance effort because the real time software and the engine performance model are already used and maintained during the engine development program,
- no additional effort to switch from non real time to real time,
- achievement of real time capability for all existing engine models in the company because the underlying software itself is qualified, not the single engine model,

- additional benefit for non real time applications as for example control schedule optimization work in terms of numerical stability and calculation time.

2 OFF-THE-SHELF CALCULATION SOFTWARE

Performance calculation programs are used for aero-engine development purposes to perform design, off-design and test data analysis for a wide range of engine types and application cases. The calculations are aiming at determining overall engine data but also component performance for both steady-state and transient operation. To address the various performance relevant tasks encountered during engine development it is common standard in aero-engine industry to use full thermodynamic performance programs. With this type of program the engine is modelled on a component level, the degree of component detailing depending on the purpose of the calculation and the availability of component data. Generally the components are representing the engine functional groups, e.g. compressor, combustor, turbine etc.

At MTU the MOdular Performance Synthesis program MOPS (see [5]) has been used for more than 20 years. To perform a calculation for a specific engine the user must create an engine model by feeding the engine data into the performance program via special input files. The calculations are executed within a user-friendly software environment providing pre- and post-processing as well as diagnosis features. For special applications the basic off-the-shelf performance library can be isolated in order to be linked to or to be controlled by some top level program.

2.1 General calculation methodology

Within MOPS each engine component is represented by a module that contains a piece of source code describing the component specific aero-thermodynamic behaviour. Each module produces mean values of the mass flow and the thermodynamic variables of state at module exit as a function of the respective values at module entry. The calculations performed within the modules are based on component specific data and characteristics provided by the user and are taking into account fundamental physical laws such as the conservation of mass and energy. As shown in Figure 1, the module arrangement reflects the layout of the engine. The performance calculation typically starts at engine entry station, proceeds through the modules according to the main flow direction and ends at engine exit. A complete calculation pass from engine face to exit is quoted in this paper as a ‘thermodynamic loop’.

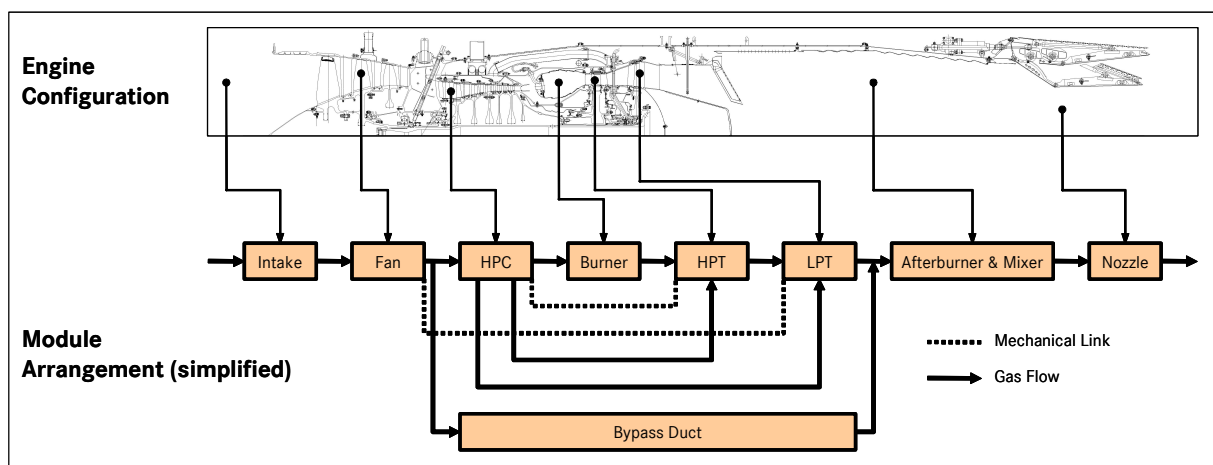


Figure 1: MOPS module arrangement for a given engine configuration.

Although the calculation within one thermodynamic loop is more or less straight forward, it is necessary to guess some parameters at certain points of the calculation process. Such a situation occurs for example in the compressor when reading the compressor characteristic. Mass flow, pressure ratio and efficiency depend on spool speed and compressor loading but the latter parameters are not known a priori and must be guessed in order to proceed with the calculation. On the other hand the components do not act independently from each other. Physical constraints such as flow continuity across the engine components and power balance between compressor and turbine must be respected. These matching conditions are used to fix the parameters that have been guessed previously.

To calculate the output for a single operating point a set of input variables must be specified, depending on the type of calculation to be performed. A steady-state operating point is defined by the flight conditions, the customer required installation conditions (e.g. power off-take) and the actuator output (e.g. fuel flow). A single transient operating represents a snapshot within an overall transient manoeuvre governed by transient phenomena (e.g. rotor inertia). Therefore additional state variables (e.g. spool speed) are necessary to define a transient operating point.

2.2 Numerics

From a mathematical point of view, finding a set of guess parameter values that are fulfilling the matching conditions requires to solve a system of analytical equations. Since the relations describing the engine behaviour are mostly non-linear and partly given by tables, an analytical solution can not be found and hence an iterative technique must be used instead. Within this technique the guess parameters are acting as iteration variables and the matching conditions are representing the iteration errors. Based on a (guessed) set of initial iteration variable values the general iteration procedure consists in modifying the iteration variables and performing iteration loops through the aero-thermodynamic calculation until the resulting iteration error norm falls below a pre-defined level of tolerance. When this is the case the calculation is considered as being converged and iteration is stopped. Figure 2 summarizes the variables and computations involved in a single operating point calculation with MOPS.

The key numerical feature when calculating a single operating point is the algorithm used to compute the matrix used to update the iteration variables within the iteration procedure. MOPS uses the open source HYBRID algorithm described in [6]. It combines the classical Newton-Raphson technique with an efficient Broyden/Powell type matrix update algorithm. A single operating point calculation is performed in three steps, as shown in Figure 2:

- Step 1: Based on the operating point input data and on an initial set of iteration variables a reference calculation (single thermodynamic loop) is performed. If convergence is achieved no further iteration is necessary. The calculation proceeds with the next operating point without execution of step 2 and step 3.
- Step 2: If step 1 does not produce a converged solution as well as for the very first transient operating point, an initial Jacobian matrix of the system is calculated. The calculation is performed via small perturbations of the iteration variables and requires as many loops through the thermodynamic calculation as there are iteration errors. Finally a QR decomposition is performed which yields matrices being equivalent to an inverted Jacobian matrix (see [6]).
- Step 3: With the QR matrices the iteration is started, aiming at the reduction of the initial iteration error norm by updating the iteration variables. For the first iteration the initial QR matrices as determined in step 2 are used. Due to the non-linearity of the system it is likely that this first Newton-Raphson type iteration will not lead to a converged solution

and further iteration is necessary. In the subsequent iterations the iteration variable updates are calculated with an update of the QR matrices. For the update of the matrices a Broyden/Powell type algorithm is used that produces approximations of the true inverted Jacobian matrix based on the information from the previous iteration loops.

As shown in Figure 2, three types of thermodynamic loops can be distinguished during the calculation of a single operating point. The ‘reference loop’ for reference point calculation, the ‘matrix loops’ necessary to compute the initial Jacobian matrix and the ‘iteration loops’ necessary to reduce the iteration errors by updating the iteration variables.

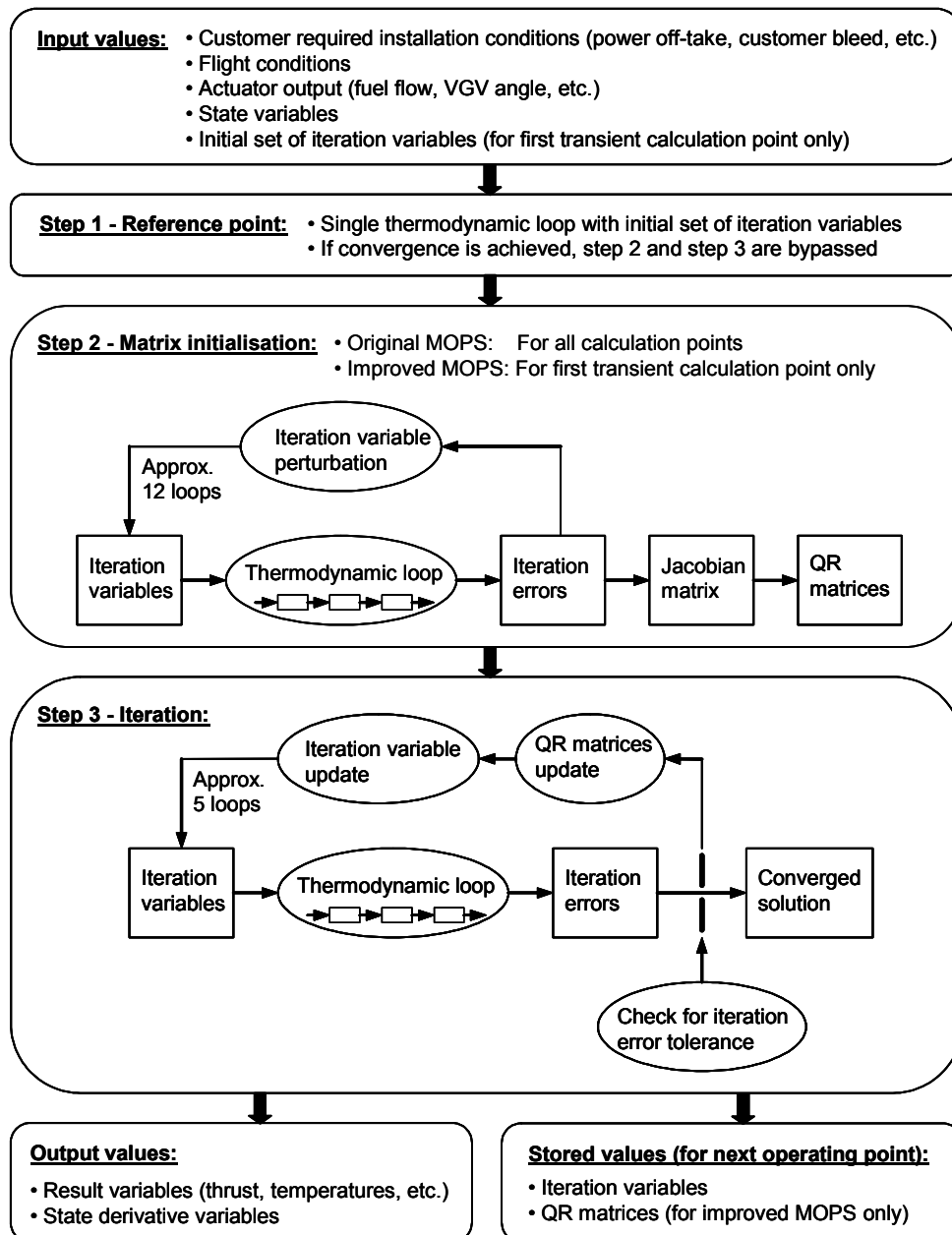


Figure 2: Single transient operating point calculation procedure.

2.3 Reference application for real time qualification

The flexibility of the MOPS program goes along with some penalties concerning the execution time and improvement is necessary to achieve real time capability. To qualify the code

improvements discussed in the following chapters it is necessary to consider a special engine model, in this case the engine model of the aircrew training aid system described in [2]. The rather complex engine configuration is shown in Figure 1 and consists of a two spool low bypass ratio turbofan with flow mixing and afterburning. The corresponding non-linear system (and thus the iteration scheme) comprises 12 equations. The transient simulation was developed within the MATLAB/Simulink® environment. The corresponding model is used in a crew training aid application and consists of a MOPS thermodynamic engine sub-model running closed-loop with sub-models for the engine control unit, the actuator and the sensor system.

3 REAL TIME CAPABILITY CONCEPT

3.1 Boundary conditions and requirements

The effort to qualify the off-the-shelf engine performance computer program MOPS for real time applications was based on some boundary conditions and requirements. These have been set by the aircrew synthetic training aid development activity already mentioned in chapter 1 which is based on the reference application described in chapter 2.3. The detailed requirements comprise

- a given CPU being an 833 MHz ALPHA processor,
- the absolute robustness of the simulation ensuring a reasonable answer within the given update rate in all flight conditions and power settings (from start to maximum power) including engine and control system malfunctions,
- simulation accuracy as in non real time application, so no trade between calculation time and accuracy was allowed,
- a given update rate of 70 Hz for the propulsion system yielding to a calculation time available for the engine simulation of around 10 ms per simulated time step.

3.2 Dismissed speed up measures

A wide range of measures as described in the open literature discussed in chapter 1 can be implemented to speed up MOPS execution. However, not all of these measures are efficient and in line with the requirements mentioned above. For sake of completeness the measures that have been considered but not retained are summarised in the following list:

- Suppression of file input: Although any file reading action is time consuming a transient calculation will read the engine specific input file only once at the beginning of the calculation, thus time savings are marginal. Furthermore MOPS already provides special interface functions to handle the data transfer with top-level programs during transient simulation via shared memory.
- Code efficiency: The extent of the thermodynamic calculations within a MOPS model depends mainly on the engine configuration. Obviously a simple turbojet will take less calculation time than a three spool mixed flow turbofan with an afterburner. The time consumption also depends on the calculation path the user wants to perform within the modules. All these items can not be influenced since they are requirements for the calculation. Still, execution speed can be improved if efficient algorithms are used, e.g. fast fluid model functions, smart table interpolation and calculation procedures which avoid as much as possible internal iteration. Any overhead in the calculation should be reduced, ensuring that only those calculations are performed which are strictly necessary to produce the results required for the simulation. However, weak point identification and implementation of these measures would be very time consuming since MOPS source code profiling has revealed that the calculation load is more or less evenly distributed over all the MOPS subroutines

and no subroutine has shown to take more than 10 % of the overall calculation time for its own. Therefore none of these measures have been applied.

- Improved matrix calculation: According to Figure 2 the calculation proceeds through all the modules for each matrix loop. As mentioned in [7] this is not necessary since the modules ‘upstream’ of the iteration variable position within the module arrangement are not affected by the iteration variable perturbation. Therefore the calculations within these modules can be skipped and execution time may be saved. However, the savings are possible only for the matrix loops and not for the iteration loops. Since the effort for implementation and validation is high, this measure has not been applied to MOPS.
- Reduction of the number of iteration loops: Typically the number of iteration loops necessary to produce a converged solution varies between 1 and 10, depending on
 - the operating point (sub-idle operation with mass flow and speed approaching zero may lead to poor convergence),
 - the demanded accuracy (the higher the demanded accuracy the higher the number of iteration loops),
 - the complexity of the iteration scheme (number of iteration errors),
 - the condition of the system of non-linear equations and
 - the quality of the initial iteration variable guess.

The first three items are mainly fixed by the application and thus can not be influenced. The condition of the system of non-linear equations depends mainly on the mutual interaction between the iteration variables and the iteration errors. To improve convergence, the user must take care when setting up the iteration scheme, e.g. by avoiding situations where the iteration variables tends to be zero. Concerning the quality of the initial iteration variable guesses, MOPS provides the possibility to use pre-calculated tables with guess values for the iteration variables. However, during a transient simulation the input variables do not vary significantly from one operating point to the next, especially if the update rate is high. This is due to the pilot’s limited reaction capability, to the controlled reaction rate of the actuators and to the engine spool inertia. This offers the opportunity to simply use the iteration variable values of a converged operating point as initial set for the subsequent operating point. As shown in Figure 2 this procedure is already implemented in MOPS.

3.3 Retained speed up measures

Based on the real time engine model approach described in chapter 1.3, on the requirements listed above and on the starting point provided by the MOPS software the following items were identified as major steps towards a real time capable full thermodynamic engine performance computer program:

- Suppression of file output: Reduce the output provided in the huge development environment for post-processing tools. Many of the features used in non real time performance work are not necessary for real time applications.
- Reduction of time per thermodynamic loop: Reduce the time necessary to evaluate the engine thermodynamics from inlet to nozzle once.
- Reduction of the number of thermodynamic loops: Improve the numerical solution of the non-linear equation scheme by applying intelligent and appropriate solver algorithms.

4 ACHIEVEMENT OF REAL TIME CAPABILITY

4.1 Suppression of file output

Performance development work usually includes post-processing activities. For this purpose the results of the MOPS calculation are stored in special data base files. The associated file

creation and writing activities are consuming a significant amount of execution time. In real time applications the MOPS output required by the top-level program or device is updated with each time step via special interface functions. Therefore no data storage is necessary and to save execution time MOPS provides the capability to switch off the data base generation. The relative savings depend on the amount of data to be stored and on the number of thermodynamic loops per operating point: the higher the number of thermodynamic loops, the lower the relative savings since the share of the data base storing activities within the overall execution time diminishes. In case of the reference application, switching off the data base storage has contributed to improve the execution time by a factor of 1.13.

A similar measure concerns the MOPS logging file which contains information about errors and warnings encountered during the calculation, e.g. lacking input data and table extrapolation. This information is provided for each operating point and has shown to be useful in an early phase of the performance development work when the engine models are set up. In real time applications validated and robust running models are used. Hence there is no urgent need for the information stored in the logging file, which could not be evaluated during real time application anyway. Again, the savings when switching off the logging file generation depends on the amount of data to be stored and on the number of thermodynamic loops. In case of the reference application the execution speed can be increased by a factor of 1.08.

4.2 Reduction of the execution time per thermodynamic loop

MOPS is implemented in FORTRAN, a 3rd generation programming language known as execution time efficient especially for scientific and technical software. A large part of the MOPS code has been implemented and modified over more than 20 years by quite a number of engineers who are naturally familiar with the physical behaviour of the engine but who are less experienced with the details of an execution time efficient program implementation. As a result MOPS certainly features some potential for optimisation for both basic code implementation and general program structure.

A simple means to improve code efficiency and thus execution time without extensive code modification consists in using the optimisation options offered by a compiler. By default MOPS is compiled in debug mode to provide full diagnosis possibility for the code development staff and for the program user during model development. With this compilation mode the executable contains additional debug information that significantly slows down the program execution and that inhibits any code optimisation by the compiler. Since a real time application will not allow the user to exploit the debug information the corresponding compiler option can be disabled in favour of the optimisation option. With the optimisation option the executable contains no debug information and the code is automatically re-arranged by the compiler in a more efficient fashion leading to a significantly reduced calculation time. Although the savings depend to some extent on the compiler used (f77 or g77), tests have shown that execution speed can be improved by a factor of 3.4 if full optimisation is enabled. It should be noticed that the output generated by a fully optimised executable may change slightly due to the changed sequence of the calculation (code re-arrangement). However, this phenomenon has not been observed during the qualification tests.

A further possibility to save computation time consists in using a single precision version of the MOPS executable instead of a double precision version. Again, the compiler options can be used for this purpose. Tests have shown that execution speed can be increased slightly by a factor of 1.15 without compromising the accuracy of the results as demanded by typical real time applications.

4.3 Reduction of the number of thermodynamic loops

According to Figure 2 the overall number of thermodynamic loops is equal to the sum of the matrix loops, the iteration loops and the single reference loop. Reduction of the number of iteration loops has already been discussed in chapter 3.2 and the now retained improvement are focussing on the reduction of the matrix loop number.

The number of matrix loops depends only on the number of iteration errors within the iteration scheme. A typical MOPS off-design iteration scheme as used in the reference application features 12 iteration errors. To reduce the number of matrix loops in case of a transient calculation, a similar approach as for the computation of the initial set of iteration variable values can be applied. Since the operating point input does not vary significantly between two subsequent operating points, the behaviour of the engine described by the Jacobian matrix will also not change significantly. Therefore it is possible to simply use the last updated QR matrices from the previous operating point as initial matrices for the actual operating point. By doing this, the initial QR matrices calculation as performed in step 2 of Figure 2 is skipped and the QR matrices are continuously updated with the Broyden/Powell algorithm. With the improved method no matrix loops at all are necessary and the total number of thermodynamic loops is now equal to the number of iteration loops plus the single reference loop. Thus the execution speed of a typical application featuring 12 matrix loops and 5 iteration loops can be improved by a factor of 3.4.

The improved method has been implemented in MOPS and successfully used in the crew training aid application described in [1]. Figure 3 is showing the input as well as some results plotted versus time for the simulation of a typical transient manoeuvre performed with the reference application running with an update frequency of 70 Hz. As shown in the power lever demand history (Figure 3a) the manoeuvre features a slam deceleration from maximum dry rating to idle rating immediately followed by a slam acceleration to maximum dry rating again. The engine response is shown in Figure 3b. In the relevant time range the thrust decreases gradually down to a value of almost zero and increases again during the acceleration.

To illustrate the effect of the improved solver method, the manoeuvre has been simulated for three different test cases:

- Case 1 (black line) is based on the original solver method which includes the calculation of the initial QR matrices.
- Case 2 (blue line) is based on the improved solver method which goes without the initial matrices calculation.
- Case 3 (red line) is also based on the improved solver method but in addition the number of iteration loops is limited to 5.

As shown in Figure 3c, case 1 features the highest number of total loops. With the improved solver method (blue line) the total loop number can be significantly reduced which presents the main advantage of the improved solver. However, at certain operating points case 2 still requires up to 9 thermodynamic loops which may be too much for the achievement of real time capability. To ensure return of data within the required time frame in any operating point the number of iteration loops must be limited, as shown by the red line. Although cutting the number of iteration loops is necessary for three operating points it should be emphasised that more than 99.6% of the relevant operating points are not concerned by the iteration loop limitation.

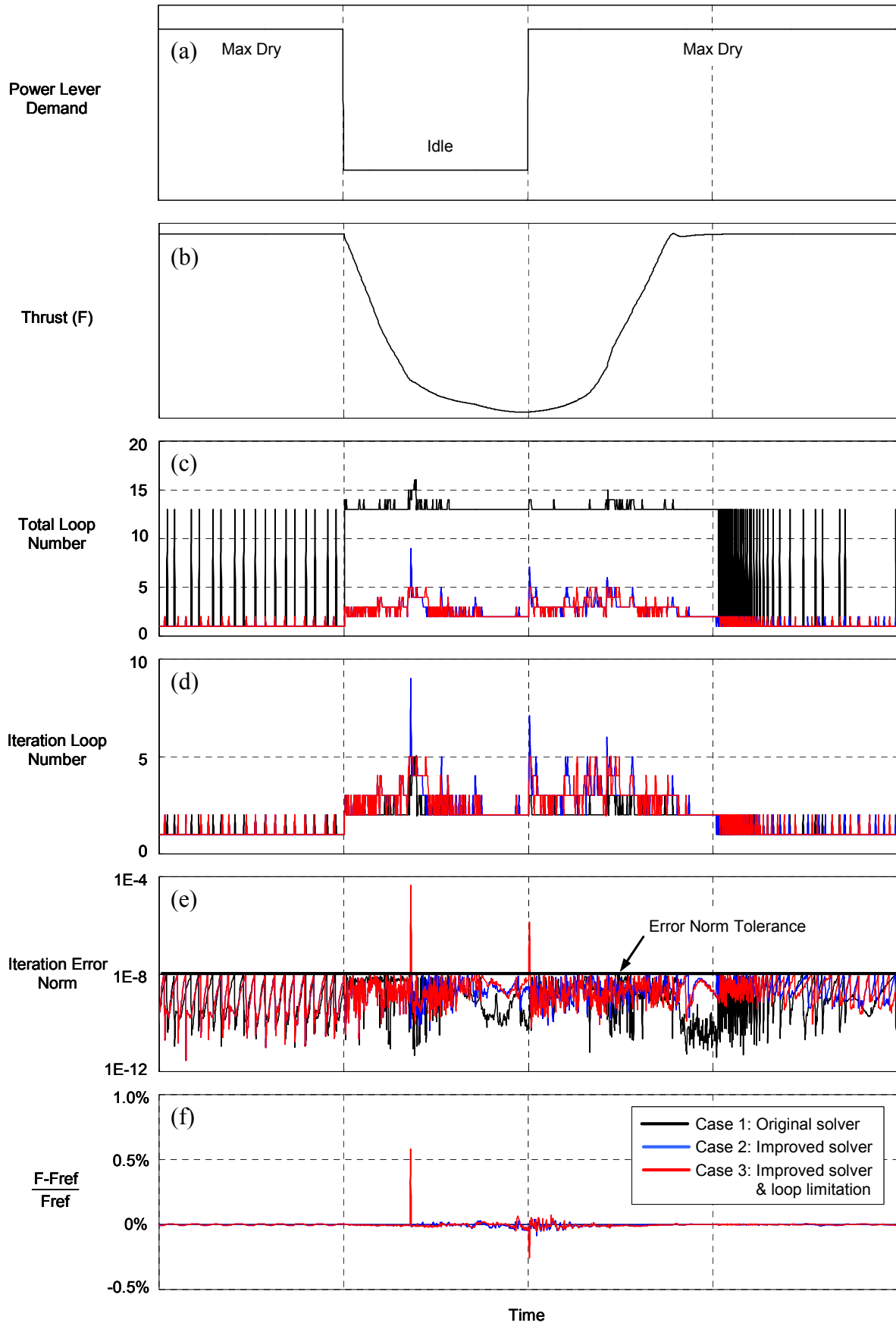


Figure 3: Transient simulation with original and improved initial QR matrices calculation method.

One may ask if the improved solver method with its continuous QR matrices update without any recalculation of the true Jacobian matrix may worsen the convergence. After all, the QR matrices are never refreshed and it might be questionable whether the updates are still representative for the inverse Jacobian matrix especially after a long simulation period. Figure 3d gives the answer, showing the pure iteration loop number for the three test cases. The original solver (black line) features a quick convergence, never requiring more than 5 iteration loops. Compared to this reference case, the improved solver (blue line and red line) sometimes needs one or two additional loops and at singular points even more. Therefore the improved solver indeed presents some penalties in terms of iteration loop number but in total the savings of the matrix loops largely outweigh the increase of the iteration loop number. Any sort of ‘matrix aging’ effects due to a lacking refreshment of the QR matrices at regular time intervals can not be observed at all.

Figure 3e is showing the impact of the improved solver on the accuracy of the calculation in terms of iteration error norm. The iteration error norm is equal to the sum of the square of all the iteration errors included in the iteration scheme. Convergence is achieved if the error norm is lower than the tolerance level, a typical value being 10^{-8} . Whereas convergence is always achieved for case 1 and case 2, case 3 fails to converge at least in two operating points out of 700. This is due to the iteration loop limitation. The impact on the main result of the calculation can be seen in Figure 3f which is showing the percentage deviation between the thrust response for cases 2 and 3 and the thrust for the reference case 1. The difference is lower than 0.1% for more than 99.6% of the operating points and even in cases where convergence is not reached, the difference does not exceed 1%. This result is well in line with the accuracy requirements for the reference application.

4.4 Summary of the achievements

Table 1 summarises the benefits in terms of execution speed improvement which can be obtained when the above mentioned measures are applied. For the aircrew training aid reference application all the measures listed in the table with the exception of measure no. 4 have been used enabling an overall speed up factor of 14. For a typical engine development application measures no. 3 and 5 can be used, yielding a speed up factor of 11.

no.	Measure	Speed up factor
1	Disabling data base generation.	1.13
2	Disabling log file generation.	1.08
3	Compiler options for optimised code.	3.40
4	Single precision instead of double precision executable.	1.15
5	Pure Broyden/Powell QR matrices update.	3.40

Table 1: Summary of the benefits.

4.5 Trades with CPU rate and update rate

Although the reference application features fixed requirements as listed in chapter 3.1, the requirements might change if other real time applications are considered. This is especially true for the desktop CPU hardware which is subjected to continuous progress and for the required update rate which depends on the special application. In this context the impact of both items on the execution speed was investigated to assess the need for further improvement.

Increasing the performance of the CPU hardware obviously reduces execution time. The trade between CPU rate and MOPS execution speed has been investigated in detail for a calculation

of 6000 operating points performed with the reference application. The calculation has been performed on several SGI RISC workstations with various CPU rates ranging from 180 MHz to 866 MHz. As shown in Figure 4 the major outcome of this investigation is that the execution speed of an operating point is proportional to the CPU rate. In other words, execution time can be reduced by half if the CPU rate is doubled. Since the development of the past 10 years has shown that the desktop CPU rate (and thus the speed) is doubled every 18 month, any extensive and expensive measure aiming at speeding up the MOPS source code should be assessed carefully.

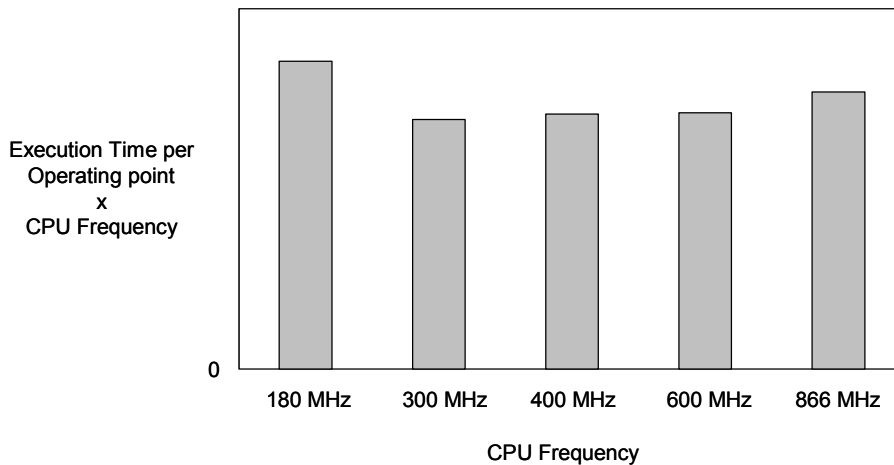


Figure 4: Influence of CPU rate on MOPS execution speed.

However, these results apply only for the average execution time. A more detailed analysis performed on a SGI RISC 400 MHz CPU has revealed a fluctuation for a single operating point execution time. As can be seen in Figure 5 the bandwidth of these fluctuations is approximately 20%, meaning that a single operating point may take up to 20% more execution time than the average. This effect occurs approximately for every 500th operating point and is probably due to some CPU internal memory re-allocation activity which is forced when cache memory is full. The fluctuations are problematic if the MOPS calculation is asked to return data at regular time intervals. The only solution to overcome the problem is to take into account some spare time to cover the fluctuations.

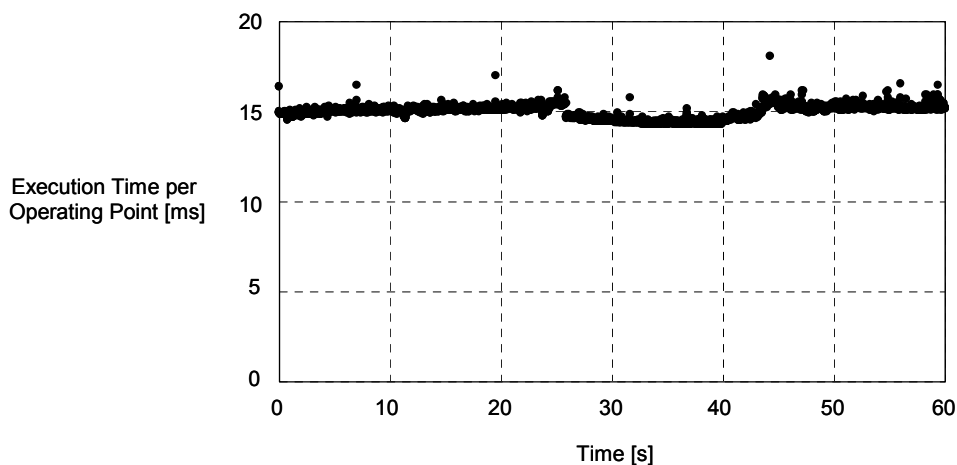


Figure 5: Operating point execution time fluctuation.

With increasing update rate the available calculation time decreases but the engine state and control input changes versus one simulated time step become smaller and thus the required number of iteration loops decreases, too. So in terms of required calculation time it is not clear a priori which update rate is best suited. Investigations with the reference application with various update rates ranging from 2 Hz to 200 Hz have shown calculation time continuously increasing with update rate. Only for very low update rates (1 Hz and lower) the solver for the thermodynamic iteration loops runs into poor convergence resulting in increasing calculation time again. Real time application relevant update rates are typically in the range of 50 Hz and higher, which corresponds to a time step size small enough to impose no problems at all to an adequate thermodynamic solver setup. The higher number of time steps and thus operating points in this update rate regime outweighs the reduced iteration loops per time step by far. So generally speaking the lowest update rate possible is the best one in terms of calculation time.

There are two important restrictions to lowering update rate. First of all the required update rate in simulation applications is usually set by other simulation elements than the engine so despite any optimum the engine simulation has to cope with the requirement set. The other restriction might be time integration and thus transient simulation data accuracy because lower update rate in a one step solver results in lower integration accuracy.

5 CONCLUSIONS

An off-the-shelf full thermodynamic engine performance computer program has been qualified for real time transient simulations. This was achieved exclusively by improving the source code itself without compromising the general applicability for various engine configurations. The three main areas of improvement have been output reduction, reducing the execution time per loop and reducing the number of thermodynamic loops per time step.

The achievements are demonstrated using the example of a complex fighter engine simulation model used as a sub-model for an aircrew synthetic training aid. The required update rate is 70 Hz yielding to a calculation time available for the engine simulation of around 10 ms per simulated time step. On a 833 MHz ALPHA processor the model requires a mean calculation time per loop of lower than 2 ms allowing a number of loops per simulated time step of at least 5. The simulation results show an accuracy which is identical to the required accuracy for the non real time performance work for more than 99.6% of the simulated operating conditions. In other words for more than 99.6% of the calculated points in time the real time simulation converges to the usually used accuracy of about 0.01 %. The remaining non converged solutions are very close to convergence so the overall performance results as thrust, spool speeds, mass flows, etc. are not compromised at all.

The microprocessor development will further improve the situation in the future. To give an indication today off-the-shelf desktop processors operate at clock times of above 2000 MHz which improves the situation by at least a factor of two compared to the results shown here. This development allows to get rid of some constraints, as for example inaccuracies due to the loop limitation, already today.

Despite the microprocessor development the code speed up will keep to be a challenge for full thermodynamic engine performance computer programs because

- for on-board applications (for example embedded models for performance seeking control) the available (airworthy) CPUs are not at all comparable in performance to today's desktop CPUs,

- engine performance models are developed in the direction of higher level of detail (zooming to 1D, 2D and 3D CFD models, see [3]) requiring more CPU power,
- for the faster than real time applications as for example control schedule development the engine performance models will never run fast enough.

6 REFERENCES

- [1] B. Biraud, A. Despierre, S. Gayraud, ‘Simulation of the WR-21 Advanced Cycle Engine’, ASME 2001-GT-0020.
- [2] M. Bolívar et. al., 2003, ‘Advanced Propulsion System Simulation Model for a Modern Fighter Aircraft Training Aid’, AIAA-2003-5374.
- [3] R. W. Claus et al, 1991, ‘Numerical Propulsion System Simulation’, in *Computing Systems in Engineering*, Vol. 2, No. 4.
- [4] C. Crainic, A. Thompson, R. Harvey, 1997, ‘Real Time Thermodynamic Transient Model for Three Spool Turboprop Engine’, ASME 97-GT-223.
- [5] J. Kurzke, 1992, ‘Calculation of Installation Effects within Performance Computer Programs’, in *AGARD Lecture Series 183*.
- [6] J. More, B. Garbow, K. Hillstrom, 1980, ‘User Guide for MINPACK-1’ , Argonne National Laboratory, ANL-80-74.
- [7] NN, 2001, ‘Real-Time Modelling Methods for Gas Turbine Engine Performance’, SAE AIR-4548-Rev. A.
- [8] NN, 2002, ‘On Board Identification, Diagnosis and Control of Gas Turbine Engines (OBIDICOTE)’, confidential final report.
- [9] N. Sugiyama, 1992, ‘Derivation of ABCD System Matrices from Nonlinear Dynamic Simulation of Jet Engines’, AIAA-92-3319.
- [10] W. Visser, M. Broomhead, J. van der Vorst, 2001, ‘ TERTS, a Generic Real-Time Gas Turbine Simulation Environment’, ASME 2001-GT-0446.