# Advanced Solvers for General High Performance Transient Gas Turbine Simulation Tools

# Advanced Solvers for General High Performance Transient Gas Turbine Simulation Tools

Tomas U.J. Grönstedt*
*Chalmers University of Technology, SE-41296 Gothenburg, Sweden*

**The numerical simulation of a turbofan acceleration transient is performed, evaluating the performance of a number of solvers ranging from a simple forward Euler method to advanced fifth order differential algebraic schemes. Three different levels of complexity in dynamic modeling for the turbofan model is studied. It is shown that differential algebraic solvers are ideally suited for developing general transient simulation tools intended for freely building up complex gas turbine systems. Furthermore, it is demonstrated that the direct solution of the differential algebraic equation system can be done very efficiently, outperforming the frequently used indirect approach. To make the engine modeling assumptions completely transparent, the full nonlinear analytic test model is given within the paper. An interpolation method substantially reducing the typical loss of performance of high order solvers when applied to a real engine performance code is demonstrated.**

## Nomenclature

| | |
|---|---|
| A | component design area |
| b | fuel flow |
| $C_p$ | specific heat |
| $C_{pc}$ | cold engine parts = 1005.0 |
| $C_{ph}$ | hot engine parts = 1148.0 |
| f | fuel air ratio |
| G | torque |
| h | stagnation entalphy |
| HPC | high pressure compressor |
| HPT | high pressure turbine |
| LPC | low pressure compressor |
| LPT | low pressure turbine |
| M | Mach number |
| $m$ | mass flow |
| $\widetilde{m}$ | $\frac{m\sqrt{\delta}}{\theta}$ = corrected mass flow |
| n | corrected rotational speed ($\frac{n_s}{\sqrt{\theta}}$) |
| $n_s$ | mechanical rotational speed (n_s) |
| ODE | Ordinary Differential Equation system |
| ODAE | Ordinary Differential Algebraic Equation system |
| $P$ | stagnation pressure |
| $P_\infty$ | ambient pressure (p_a) |
| $R$ | the gas constant = 287.0 |
| SLS | Sea Level Static |
| $T$ | stagnation temperature |

**Greek**

| | |
|---|---|
| $\alpha$ | bypass ratio |
| $\gamma$ | specific heat ratio |
| $\gamma_c$ | cold engine parts = 1.400 |
| $\gamma_h$ | hot engine parts = 1.333 |
| $\pi$ | pressure ratio |
| $\chi(M,\gamma)$ | $\sqrt{\gamma}M\left(1+\frac{\gamma-1}{2}M^2\right)^{-\frac{\gamma+1}{2(\gamma-1)}}$ |
| $\chi(\pi,\gamma)$ | $\sqrt{\frac{2\gamma(\pi^{\frac{\gamma-1}{\gamma}}-1)}{(\gamma-1)\pi^{\frac{\gamma+1}{\gamma}}}}$ |
| $\chi_c$ | $\sqrt{\gamma}\frac{\gamma+1}{2}^{-\frac{\gamma+1}{2(\gamma-1)}}$ |
| $\phi_i$ | component design parameter number i |
| $\delta$ | $\frac{P_1}{101325.0}$ |
| $\theta$ | $\frac{T_1}{288.15}$ |

**Subscripts**

| | |
|---|---|
| 1,3 | inflow to component |
| 2,4 | outflow from component |
| dp | design point |
| sl | speed line |
| s | static property |
| cr | cooling rotor |
| cs | cooling stator |
| id | ideal |
| $\infty$ | ambient conditions |

**Superscripts**

| | |
|---|---|
| $'$ | State derivative operator |

## Introduction

Dynamic gas turbine modeling has been in use for almost five decades[1] and digital dynamic engine simulation tools for more than 20 years.[2,3] Early on it was concluded that jet engines constitute stiff systems and that the speed of integration can profit greatly from the implementation of implicit solvers.[4] Although several authors have reported the successful use of low order (first/second) implicit ODE (Ordinary Differential Equations) solvers,[5–7] very little attention has been paid to the direct use of differential algebraic

*Ph.D. Student, M.Sc., thgr@tfd.chalmers.se

system solvers and quantifying the improvements of higher order integration techniques.

A critical matter concerning the modeling of a gas turbine system is to decide whether an ODE or an ODAE (ODE with coupled algebraic equations) model is going to be applied. This choice will have quite a substantial impact not only on the required computation time but also on the complexity and effort necessary for assembling the model and getting it running.

The performance of high order ordinary differential equation solvers, is strongly dependent on the smoothness of the engine model. Most performance codes use component map data, from which the required variables are interpolated. The selection of a poor interpolation method can have a devastating effect on the solver. This paper measures to which extent the smoothness of the interpolation method influences the performance of high order solvers on a typical gas turbine transient. By introducing first, third and fifth order splines[8] to approximate all map data in the performance model the smoothness can be increased in a step wise manner, and consequently the effect of the interpolation method on the solver performance can be studied.

The main guide-line for deriving the engine models used for testing the numerical solvers, was to obtain a nonlinear gas turbine system model with as realistic dynamic performance as possible, at a minimum of complexity. E.g. using constant specific heat ratios $\gamma = \gamma_h$=1.333 for engine components with a fuel air ratio $\neq 0$ and $\gamma = \gamma_c$=1.400 otherwise, was considered acceptable since it was anticipated to have little effect on how the solvers would perform. Effects of mechanical losses were also neglected. Although metal air heat transfer effects are very important for transient modeling in general no such effects were included here, since these phenomena do not add very much to the burden for the solvers. The characteristic time scales are much larger than those for rotor and especially mass and thermal gas dynamics, with eigenvalues having small negative real parts causing little trouble for ODE/ODAE solvers.

All numerical tests have been conducted with a new general transient and steady state code, GESTPAN[9] (GEneral Stationary Transient Propulsion ANalysis), developed at Chalmers University of Technology, The Royal Institute of Technology and at Volvo Aero Corporation. The tool has been developed using Fortran 90. This made the inclusion of the solvers, Fortran 77 code freely available for down-loading from NETLIB,[10] very straight forward.

## The Inter-Component Volume method

To simulate transient engine behavior, dynamic components are introduced into the engine model. Volume components are used between the static engine components to simulate storage of thermal energy and gas mass, and rotor components are used to model rotor dynamics.

## Gas turbine dynamics and ODAE:s

If an attempt to model an arbitrary gas turbine system is made the most probable system that will emerge is a semi-explicit ordinary differential algebraic system, i.e.:

$$\begin{aligned} x' &= f(t, x, z) \\ 0 &= g(t, x, z) \end{aligned} \tag{1}$$

This means that unless special care is taken during the modeling process some algebraic equations (algebraic loops) will arise coupled to the differential equation system. Note that local equations completely contained within a component do not constitute algebraic equations. In the rest of this paper the $x$ variables in Eq. 1 will be referred to as differential variables and the $z$ variables as algebraic variables.

The semi-explicit ODAE represented by Eq. 1 is very general indeed. Shampine et al.[11] shows that a system of equations can be represented in a SIMULINK block diagram if and only if the system can be written as an ODAE in the form represented by Eq. 1.

### Methods for solving the ODAE problem

Basically three main strategies for solving Eq. 1 exist:

1. The "direct approach"
2. The "ODE approach" = "the indirect approach"
3. Transforming the ODAE model to an ODE model by algebraic manipulations.

The direct approach is studied in this work by the use of the DASSL solver.[13] This code uses a kth order Backward Differentiation Formula (BDF), where k varies from one to five, to approximate the derivatives of a more general expression than Eq. 1. The DASSL code then solves the resulting equation system directly, i.e. it solves for the differential and algebraic variables simultaneously.

The ODE approach, or the indirect approach, is based on solving the algebraic equations in Eq. 1 for every function evaluation required by the ODE solver. Here, the algebraic equations are solved with a globally convergent Broyden method.[14] Three different ODE solvers have been tested. One implicit method (also a BDF method),[15] and two explicit methods; a variable order Adams Bashforth method[15] and the forward Euler method.

### Transforming the ODAE to ODE

Using the Inter-Component Volume Method for transient gas turbine modeling sometimes allows the

algebraic equations of Eq. 1 to be eliminated through manipulations of the component formulas. This has been done in the "Engine 3" test case studied in this paper (see burner, nozzle and mixer component sections in the Appendix).

#### System advantages with ODAE formulations

When working with a general gas turbine simulation tool the use of such algebraic manipulations as those described above are unfortunate from the system complexity perspective. In order to make a specific engine model free from algebraic equations component physics has to be duplicated in more than one algorithm, making the simulation system more complex without any direct benefits. It is highly desirable to be able to use the most straight forward and robust way of formulating the component physics for all engine models, including both transient and steady state formulations.

## Engine models

The nonlinear analytic engine component models described in the Appendix have been used to generate three turbofan models:

1. Turbofan - rotor dynamics - ODAE
2. Turbofan - rotor/volume dynamics - ODAE
3. Turbofan - rotor/volume dynamics - ODE

The wiring diagrams in Fig. 1 and Fig. 2(a) as well as in Fig. 2(b) illustrate how the engine components are interconnected. The differential and algebraic variables used during the integration, as well as their initial value for the test transient, are given in Table 1. The index numbering and the referencing to algebraic variables given in Table 1 are clear from the wiring diagrams and the abbreviations given in the table text.

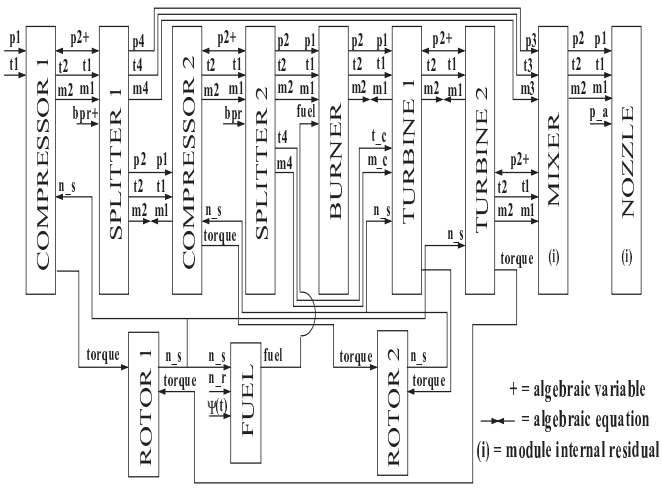



**Fig. 1 Engine 1 - rotor dynamics - ODAE model**

| | Engine 1,2,3 | |
|---|---|---|
| $x_1$ | $n_{s,R1} = 124.29$ | |
| $x_2$ | $n_{s,R2} = 223.79$ | |
| | **Engine 2,3** | |
| $x_3$ | $T_{V1} = 411.62$ | |
| $x_4$ | $m_{V1} = 0.4124$ | |
| $x_5$ | $T_{V2} = 727.07$ | |
| $x_6$ | $m_{V2} = 1.3426$ | |
| $x_7$ | $T_{V3} = 1373.4$ | |
| $x_8$ | $m_{V3} = 0.6743$ | |
| $x_9$ | $T_{V4} = 1061.7$ | |
| $x_{10}$ | $m_{V4} = 0.3099$ | |
| $x_{11}$ | $T_{V5} = 911.78$ | |
| $x_{12}$ | $m_{V5} = 0.4401$ | |
| $x_{13}$ | $T_{V6} = 767.61$ | |
| $x_{14}$ | $m_{V6} = 0.5270$ | |
| | **Engine 1** | **Engine 2** |
| $z_1$ | $P_{2,LPC} = 2.43E5$ | $bpr_{S1} = 0.4122$ |
| $z_2$ | $bpr_{S1} = 0.4122$ | $m_{3,M} = 11.712$ |
| $z_3$ | $P_{2,HPC} = 1.40E6$ | $m_{2,V2} = 26.601$ |
| $z_4$ | $P_{2,HPT} = 4.72E5$ | $m_{2,V5} = 28.922$ |
| $z_5$ | $P_{2,LPT} = 2.30E5$ | $m_{2,V6} = 40.634$ |

**Table 1 Differential and algebraic variables used during integration with their initial conditions (V=Volume, R=Rotor, M=Mixer)**

## Solver comparisons - Methodology

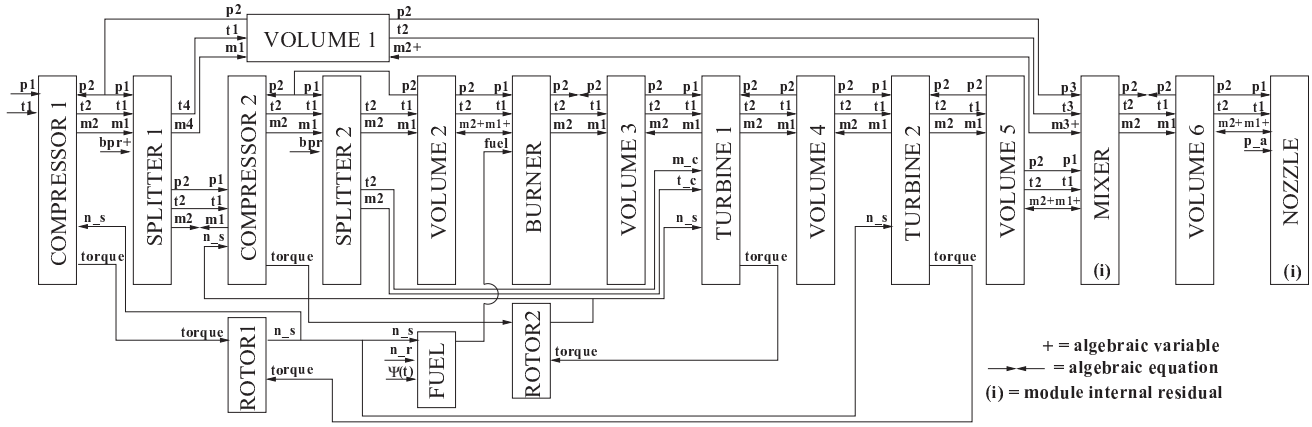
#### Test transients and accuracy requirements

The engine test transient selected for the measurement of the solver performance is an engine acceleration trajectory from 67% to 95% of engine maximum rotational speed.

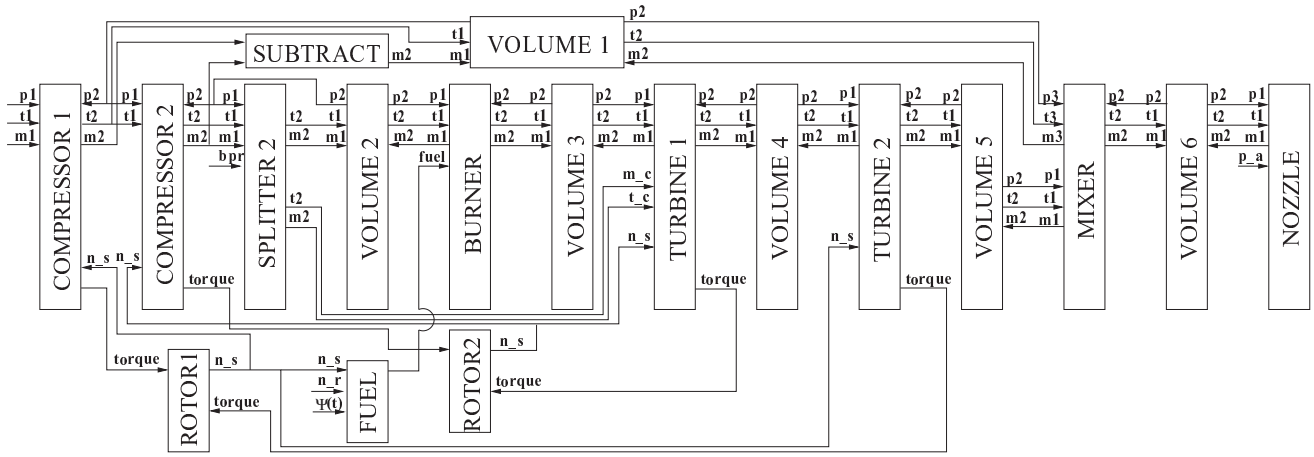
#### Error control

To establish a converged solution integration was carried out for a decreasing series of tolerances. The three codes having a proper error control (not the forward Euler implementation) were tested in this way and produced the same solutions for the for all three engines. Note that the converged solution of Engine 2 and Engine 3 will be the same but Engine 1 will differ due to the absence of volumes.

A meaningful solver performance measurement has to be carried out at a specific error tolerance. For gas turbine engines a solution which is more accurate than the maximum attainable accuracy of a tuned system model would be wasteful. Another relevant aspect is that the way errors are transported along the solution trajectory depend on the dynamic system itself. A numerical error made at one point might be attenuated along the trajectory. With all this in mind a global test was formed based on all the points along the integration compared to the converged solution, i.e. it was required of a solution to pass the following test:




**a) ODAE model**




**b) ODE model**

**Fig. 2 Engine 2 and Engine 3 - rotor and volume dynamics**

$$\sqrt{\frac{\sum\left(\frac{y-y_{conv}}{y_{conv}}\right)^2}{nps}} < 0.005 \qquad (2)$$

where $nps$ is the number of points on the trajectory. All cases were sampled at 100 Hz, i.e. 500 test points were used. The error was checked on the fuel scheduled by the feedback controller, i.e. the fuel component. $y_{conv}$ was obtained using the DASSL code with a pure relative error control using a tolerance of $10^{-10}$.

The transient was started from the initial conditions given in Table 1 and a required rotational speed was set to $n_r$=180.0. The fuel scheduling trajectory relevant for Engine 2 and Engine 3 is displayed in Fig. 3.

## Solver comparisons - Results

The number of function evaluations, i.e. engine evaluations, given in Table 2 below are the minimum number necessary for the solution to pass the error test defined by Eq. 2.

A number of observations can be made based on Table 2.

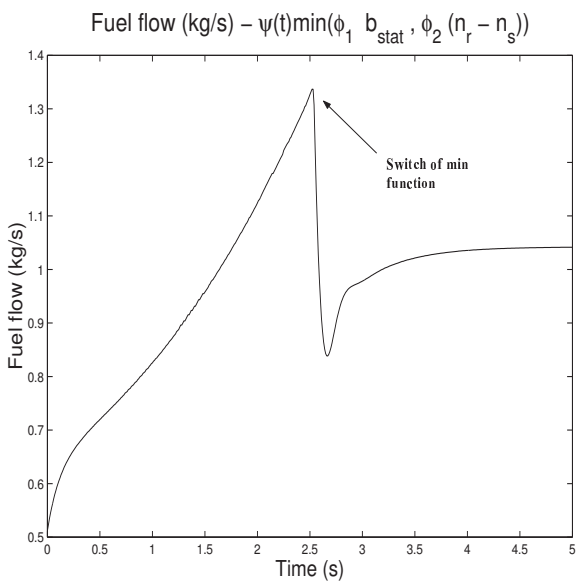



**Fig. 3 Fuel scheduled by the control unit**

Elimination of algebraic equations: In Engine 3 the formulas of the mixer, the burner and the nozzle were rearranged in order to eliminate the algebraic equations present in Engine 2. This operation resulted in

| Engine | F. Eul. | LSODE | DASSL | Adams |
|---|---|---|---|---|
| 1 | 11240 | 3362 | 761 | 3898 |
| 2 | 625143 | 11879 | 1585 | 2393325 |
| 3 | 62502 | 1313 | 1147 | 286078 |

**Table 2 Performance of the solvers**

an increase in computational speed with about an order of magnitude, for all solvers except for the DASSL code. The reason for this is that the Forward Euler, the LSODE and the Adams Bashforth implementations use an "ODE approach" to solve the differential algebraic problem, i.e. the solution of a nonlinear equation system has to be performed for every function evaluation required by the ODE-solver. The error tolerance for the equation solver was set to a fraction $10^{-3}$ of the local error requirement of the ODE-solver. This seemed, after some trade-off studies, to be close to the optimal choice. A larger value increased the number of function evaluations, since the inaccurately solved equation system would disturb the ODE-solver.

The DASSL code, on the other hand solves the problem by the "direct approach" treating the differential and algebraic variables simultaneously. Note also that the number of function evaluations required by DASSL is reduced only with about 30% as the the system is transformed from the ODAE system represented by Engine 2 to the ODE system represented by Engine 3.

Engine 1 - non-stiff case: In the Engine 1 case all the volumes have been eliminated. Since these are the main source of stiffness (large negative real parts of the eigenvalues on the system Jacobian) the explicit solvers perform fairly well.

Engine 2 and 3: Judging from the performance measurements of the variable order Adams method and the forward Euler method the use of explicit methods for inter-component volume models seems completely out of the question. However, there is one less appealing remedy to this problem. By increasing the size of the volume giving rise to the pole with the largest negative real part the stiffness of the model can be reduced. Of course, such a trick could also introduce some unwanted physical effects.[2]

A comment about the Forward Euler method: The reason for the fact that the forward Euler method is actually faster than the Adams method for Engine 2 and Engine 3 is that this implementation has no automatic stability control. The engine transients were recalculated repeatedly for the Euler method in order to obtain the largest possible step size without making the solution unstable. As a matter of fact the order used by the Adams method throughout the entire integration was one (this is the forward Euler method) and since the Adams code monitors the stability automatically it is reasonable for the method to be slower. However, in the Engine case 1, the problem is no longer stiff and the stability requirements will no longer limit the selection of order for the Adams method. Up to order 6 was then observed for the Adams code during integration.

## "Real" performance code effects

The major difference between a real performance code and the nonlinear test equations given in the Appendix, is that the performance code will most likely use a number of empirical maps for estimating the component performance. When interpolation methods are used on these maps discontinuities in higher derivatives will be introduced in the table break points. More refined codes will then use smaller step sizes around these points to get an accurate solution. This can reduce the performance of high order solvers quite drastically, as is shown below.

By the use of higher order approximating spline routines this problem can be alleviated, and in some cases measurement noise can be filtered away at the same time. The methodology for obtaining the spline approximations are described in great detail by Dierckx.[8]

Except for the nonlinear test equations described in this paper the GESTPAN simulation system also has a number of standard component performance models based on measurement data or more accurate models. A system with components corresponding to Engine 3 was set up (the same wiring diagram). In this case the model used 26 empirical tables. Three sets of spline coefficient approximations were defined for every table: linear, cubic and quintic approximating splines. The performance model was tuned to give roughly the same off-design performance as the analytical model and the code was started from close to the same initial conditions.

Since it is part of the conclusions of this paper that the DASSL solver is the most suitable solver for developing general gas turbine simulation systems it was selected for this study. Table 3 gives the results for the runs.

| Smoothness | DASSL |
|---|---|
| Linear interpolation | 10504 |
| Linear spline approximations | 9902 |
| Cubic spline approximations | 2387 |
| Quintic spline approximations | 2323 |

**Table 3 Effect of map data smoothness on solver performance**

It is seen that a poor interpolation method can increase the number of function evaluations between four and five times.

The DASSL solver can be set up to limit the order of the BDF method. This was done reducing the variable order BDF-method to the backward Euler method. This increased the number of necessary function evaluations to 17256 in the linear interpolation case and 15246 in the quintic spline case. This demonstrates

that the benefits of high order derivative smoothness is small for low order BDF methods. Likewise, the benefits of high order BDF methods applied to models with poor smoothness is limited. In fact, if linear interpolation was used the optimal BDF for this problem was of order two.

## Using Matlab and SIMULINK

Until recently, the methods for solving ODAE equations in SIMULINK were very crude indeed. As Shampine observes[11] "These versions had a limited capability for solving models with algebraic loops, so users had to resort to ad hoc changes to models in order to solve DAE:s beyond the capabilities of the language". However, Shampine et al.[11] have now introduced improved methods for the solution of ODAE:s both in Matlab and SIMULINK,[12] including a method using the direct approach for the Matlab environment. The SIMULINK environment still uses the "indirect approach".

Although the LSODE and DASSL solvers show comparable performance for the Engine 3 case, the DASSL code is about 4.4 times more efficient than LSODE in the Engine 1 case and 7.5 times in the Engine 2 case. This indicates that the indirect approach used by SIMULINK for dealing with ODAE:s corresponds to a considerable loss in performance compared to the direct method.

## Conclusions

The direct approach for solving the ODAE system arising from the acceleration test transient, has been shown to work very efficiently. The DASSL solver was about 4.4 times more efficient in the Engine 1 case and 7.5 times in the Engine 2 case, compared to the most efficient solver using the indirect method. Furthermore, the DASSL code solved the full inter-component volume model represented by Engine 2, using less than half the number of function evaluations requiered by the Adams method to solve the rotor dynamic model represented by Engine 1. This indicates that the justifiable use of models with only rotor dynamics is limited to cases when a minimum of complexity is required.

The application of the direct method allows the complexity of the simulation system to be kept at a minimum, by using the same engine component formulations for all engine models maintained by the system, including both steady state and transient formulations.

It has been demonstrated that high order BDF techniques can give increased performance with as much as a factor of 6.6 compared to the BDF method of order one, i.e. the backward Euler method, if suitable methods for data interpolation are used.

## Appendix: Engine component models

In this section all the nonlinear equations for the test modules are defined. Every engine component has its own set of design parameters $\phi_1, \phi_2, ..., \phi_n$. These are specified together with their equations in the component section below, and their values are given in Table 4. Note that specifying the engine design point simply means giving values to all the component design parameters.

**Compressor component**

The compressor maps are generated using ellipses with the minor and major axes depending on the rotational speed ratio $\frac{n}{n_{dp}}$ according to:

$$(\frac{\pi}{f(\frac{n}{n_{dp}})})^2 + (\frac{\widetilde{m}}{g(\frac{n}{n_{dp}})})^2 = 1 \tag{3}$$

where $f(\frac{n}{n_{dp}})$ and $g(\frac{n}{n_{dp}})$ are identified using the following requirements:

$$\begin{aligned} f(n=0) &= 1 \\ f(n=n_{dp}=\phi_6) &= \phi_9\pi_{dp} = \phi_9\phi_7 \\ \frac{\partial f(n=0)}{\partial n} &= 0 \\ g(n=0) &= 0 \\ g(n=n_{dp}) &= \phi_{10} \\ \frac{\partial g(n=n_c)}{\partial n} &= 0 \end{aligned}$$

The value used here for $\phi_9$ ($\phi_9$=8.0) defines a rather large extension for the $\frac{n}{n_{dp}} = 1.0$ ellipsis on the y-axis. This gives a very steep rotational speed line, which physically corresponds to a compressor close to choke.

A number of elementary functions could be used to fulfill the equations above. f and g, given by Eq. 4 and Eq. 5 satisfy the equations, and give a reasonable rotational speed line distribution.

$$f(n) = 1 + (A-1)(\frac{n}{n_{dp}})^4 \tag{4}$$

$$g(n) = B\frac{n}{n_{dp}}(2 - \frac{n}{n_{dp}}) \tag{5}$$

Determining the stability limits: The surge line is defined as the parabola in $\widetilde{m}$ and $\pi$ intersecting (0.0,1.0) and ($\widetilde{m}_{\mathrm{dp}}$, $\phi_3\pi_{\mathrm{dp}}$) and the choke line as the parabola intersecting (0.0,1.0) and ($\widetilde{m}_{\mathrm{dp}}$ , $\phi_4\pi_{\mathrm{dp}}$).

Variable geometry: Variable geometry effects is included by the use of

$$h = (1-\phi_{12})\left(\frac{n}{n_{dp}}\right)^{\phi_{11}} + \phi_{12} \tag{6}$$

where h is a factor multiplying the corrected mass flows. The low pressure compressor map is shown in Fig. 4:

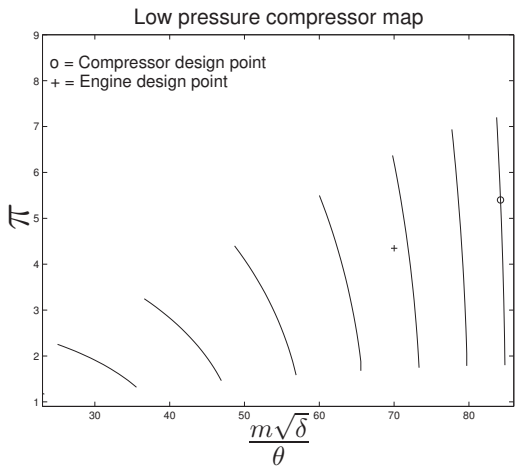



**Fig. 4 LPC with variabel geometry**

Compressor efficiency model: A third parabola, the backbone of the compressor, is introduced through (0.0,1.0) and ($\widetilde{m}_{\mathrm{dp}}$ , $\pi_{\mathrm{dp}}$) where the peak polytropic efficiency for every rotational speed is assumed to occur. The polytropic efficiency in the design point, $\phi_8$, is set to a fraction $\phi_2$ of the global compressor polytropic efficiency peak value, which is assumed to occur at $\frac{n}{n_{dp}} = \phi_1$. The variation in polytropic efficiency along the backbone is determined using:

$$\eta_{bb} = a + b\frac{n}{n_{dp}} + c(\frac{n}{n_{dp}})^2 \tag{7}$$

where a,b and c are determined from the known efficiency values in $\frac{n}{n_{dp}} = \phi_1$, $\frac{n}{n_{dp}} = 1$ and the requirement of a maximum value in $\frac{n}{n_{dp}} = \phi_1$. The variation on every rotational speed line is set by an analogous approach:

$$\eta_{sl} = a + b(\frac{\pi - \pi_{ch}}{\pi_{su} - \pi_{ch}}) + c(\frac{\pi - \pi_{ch}}{\pi_{su} - \pi_{ch}})^2 \tag{8}$$

having a peak polytropic efficiency in the backbone point, a fraction $\phi_5$ of that polytropic efficiency in the choke point and a maximum value in the backbone point.

**Burner component**

The burner efficiency is computed using an empirical reaction rate parameter, $\sigma$, defined according to

$$\sigma = \frac{P_1^{1.75} \cdot e^{\frac{T_1}{300.0}}}{m_1} \tag{9}$$

The ratio between the computed $\sigma$ value and the value in the design point $\sigma_{dp} = \phi_1$, i.e. $\frac{\sigma}{\sigma_{dp}}$, is used to compute the burner efficiency, $\eta_b$, according to

$$\eta_b = \eta_{b,dp}(1.0 - e^{-\phi_4 \frac{\sigma}{\sigma_{dp}}}) \tag{10}$$

| | LPC | HPC | HPT | LPT |
|---|---|---|---|---|
| $\phi_1$ | 0.85000 | 0.85000 | 0.90000 | 0.90000 |
| $\phi_2$ | 0.95000 | 0.95000 | 2.10000 | 2.20000 |
| $\phi_3$ | 1.33333 | 1.33333 | 0.41654 | 0.39336 |
| $\phi_4$ | 0.33333 | 0.33333 | 0.00075 | 0.00199 |
| $\phi_5$ | 0.90000 | 0.90000 | 0.50000 | |
| $\phi_6$ | 232.342 | 252.465 | 0.50000 | |
| $\phi_7$ | 5.40000 | 8.00000 | 0.50000 | |
| $\phi_8$ | 0.86200 | 0.86000 | | |
| $\phi_9$ | 8.00000 | 8.00000 | | |
| $\phi_{10}$ | 84.8916 | 18.6036 | | |
| $\phi_{11}$ | 0.70000 | 0.70000 | | |
| $\phi_{12}$ | 0.20000 | 0.20000 | | |
| | **Burner** | **Fuel** | **Volume** | **Mixer** |
| $\phi_1$ | 6.18E10 | 0.25000 | 0.2 (V1) | 0.16181 |
| $\phi_2$ | 0.99000 | 1.30000 | 0.2 (V2) | 0.03429 |
| $\phi_3$ | 1.952E5 | 4.10653 | 0.2 (V3) | |
| $\phi_4$ | 10.0000 | -1.177E-1 | 0.2 (V4) | |
| $\phi_5$ | | 1.2512E-3 | 0.5 (V5) | |
| $\phi_6$ | | -5.397E-6 | 0.5 (V6) | |
| $\phi_7$ | | 8.6744E-9 | | |
| | **Nozzle** | **Rotor 1** | **Rotor 2** | |
| $\phi_1$ | 2127.28 | 10.0 | 5.0 | |
| $\phi_2$ | 0.12843 | | | |

**Table 4 Design parameters of the engines**

where $\eta_{b,dp} = \phi_2$. A temperature increase based on ideal combustion is then computed according to:

$$\begin{aligned}\Delta t_{id} = &(\text{-}2.9429 \cdot 10^6 - 252.4827 t_1 + 0.9789 t_1^2) f^3 + \\ &(1.2888 \cdot 10^5 + 55.6336 t_1 - 0.1461 t_1^2) f^2 + \\ &(3.1273 \cdot 10^4 - 0.5387 t_1 + 8.4160 \cdot 10^{-4} t_1^2) f\end{aligned} \tag{11}$$

The real temperature is then computed according to

$$t_2 = \Delta t_{id} \cdot \eta_b + t_1 \tag{12}$$

The pressure loss is computed using:

$$P_2 = P_1 \cdot \left(1.0 - \phi_3 \cdot \left(\frac{m_1\sqrt{T_1}}{P_1}\right)^2\right) \tag{13}$$

In the Engine 3 case this formula is used to compute the mass flow as a function of the module pressure drop.

**Turbine component**

The turbine inlet mass flow is a function of the turbine pressure ratio according to:

$$\frac{\dot{m}_1\sqrt{T_1}}{P_1} = \begin{cases} \left(\frac{\dot{m}_1\sqrt{T_1}}{P_1}\right)^* \sqrt{1 - \left(\frac{\pi^* - \pi}{\pi\pi^* - \pi}\right)^2}, & \pi < \pi^* \\ \left(\frac{\dot{m}_1\sqrt{T_1}}{P_1}\right)^*, & \pi > \pi^* \end{cases} \tag{14}$$

where $\pi^* = \phi_2$ is the assumed choking pressure ratio and $\left(\frac{\dot{m}\sqrt{T_1}}{P_1}\right)^* = \phi_4$ the corrected choking mass flow. The turbine polytropic efficiency is estimated using:

$$\eta = \eta_{dp}\left(1 - \left(\frac{\frac{N}{\sqrt{\Delta h}}}{\left(\frac{N}{\sqrt{\Delta h}}\right)_{dp}} - 1\right)^2\right) \tag{15}$$

where $\eta_{dp} = \phi_1$ and $\left(\frac{N}{\sqrt{\Delta h}}\right)_{dp} = \phi_3$

Airflow cooling scheme: The stator and rotor cooling air are computed according to:

$$m_{cs} = \phi_5 m_c$$
$$m_{cr} = \phi_6 m_c$$

The turbine torque , $G_t$, is computed according to:

$$G_t = \frac{\Delta h\left(m_1 + m_{cs} + \phi_7(m_c - m_{cs} - m_{cr})\right)}{2\pi n}$$

The enthalpy upstream of the rotor, $h_{1r}$, is given by

$$h_{1r} = \frac{h_{1r}m_1 + h_c\left(m_{cs} + \phi_7(m_c - m_{cs} - m_{cr})\right)}{m_1 + m_{cs} + \phi_7(m_c - m_{cs} - m_{cr})}$$

The turbine rotor exit temperature, $t_{2r}$, is obtained according to

$$t_{2r} = \frac{t_{1r}}{\pi^{\eta\frac{\gamma-1}{\gamma}}} \tag{16}$$

The exhaust temperature $t_2$, obtained after all the cooling air has been introduced, is found from the rotor outlet entalphy $t_2 = \frac{h_2}{c_p}$, which is given by:

$$h_2 = \frac{m_{2r}h_{2r} + \left((1.0 - \phi_7)(m_c - m_{cs} - m_{cr}) + m_{cr}\right)h_c}{m_1 + m_c} \tag{17}$$

**Nozzle component**

The exhaust stagnation pressure is determined from

$$P_2 = P_1\left(1.0 - \phi_1 \cdot \left(\frac{m_1\sqrt{t_1}}{P_1}\right)^2\right) \tag{18}$$

The 1D continuity equation states ($\pi = \frac{P_2}{P_\infty}$):

$$\frac{m_1\sqrt{RT_2}}{AP_2} = \chi(\pi, \gamma_2) \tag{19}$$

Where A = $\phi_2$. In the Engine 1 and Engine 2 cases this relation is used to form an internal residual, and in the Engine 3 case the equation is used to compute the mass flow.

**Splitter component**

The splitter splits the flow into two streams preserving the thermodynamic properties of the gas. The bypass ratio $\alpha$ is defined as $\alpha = \frac{m_{\text{bypass}}}{m_{\text{core}}}$ according to:

$$m_2 = \frac{m_1}{\alpha + 1} \tag{20}$$
$$m_4 = m_1 - m_2 \tag{21}$$

Engine 1 and Engine 2 have two splitters one for the bypass duct and one for cooling flow splitter. The ODE model only has the latter. In the bypass splitter $\alpha$ is an iteration variable (algebraic variable). In the cooling flow splitter, $\alpha$ is set to 0.068.

**Mixer component**

The mixer adds the bypass stream to the core flow conserving mass, energy and momentum. The Mach number of the core flow, $M_1$, and the Mach number in the bypass duct, $M_3$, are computed using the 1-D compressible continuity equation:

$$\frac{m_1\sqrt{RT_1}}{A_1P_1} = \chi(\gamma_h, M_1) \tag{22}$$
$$\frac{m_3\sqrt{RT_3}}{A_3P_3} = \chi(\gamma_h, M_3) \tag{23}$$

where $A_1 = \phi_1$ and $A_2 = \phi_2$. A Kutta condition (equal static pressure) has to be satisfied. In Engine 1 and Engine 2 the mass flow is specified and a residual is formed equating $P_{s1}$ and $P_{s3}$, and for Engine 3 this relationship is used directly to compute $m_3$ after $M_1$ has been guessed. To find the Mach number after mixing the impulse function[17] is introduced:

$$F = P_sA + \gamma P_sAM^2 \tag{24}$$

Assert that:

$$F_2 = F_1 + F_3 \tag{25}$$

Introducing Eq. 24 into Eq. 25 yields:

$$P_{2s}A_2\left(1 + \gamma_h M_2^2\right) = P_{1s}A_1\left(1 + \gamma_h M_1^2\right) + P_{3s}A_3\left(1 + \gamma_h M_3^2\right) \tag{26}$$

Using the 1-D compressible continuity equation for $P_2A_2$ yields:

$$A_2P_2 = \frac{m_2\sqrt{RT_2}}{\chi(\gamma_h, M_2)} \tag{27}$$

$T_2$ can be computed since the enthalpy of the mixed stream is a mass weighted average of its two constituents. Thus, $M_2$ can be found iteratively, and $P_2$ is then obtained from the continuity equation.

**Fuel component - Control unit**

The fuel module serves as a simple proportional controller scheduling the fuel flow, b, during the test transient according to:

$$b = \Psi(t)\ min(\phi_1 b_{stat}, \phi_2(n_r - n_s))$$
$$b_{stat} = \phi_3 + \phi_4 n_s + \phi_5 {n_s}^2 + \phi_6 {n_s}^3 + \phi_7 {n_s}^4$$
$$\Psi(t) = 1.0 - e^{-5tln5} \quad (28)$$

where $n_r$ is the required rotational speed and $n_s$ the mechanical speed. The expression for $b_{stat}$ is a result from fitting a fourth degree polynomial to a number of steady state fuel consumption operating points. The expression for $\Psi$ ramps the fuel from $b_{stat}$ at $t = 0$ approaching the $\phi_1 b_{stat}$ exponentially fast. The fuel flow reaches 80% of $\phi_1 b_{stat}$ in 0.2 seconds.

**The volume component**

The volume components are used to simulate storage of energy and mass in the engine according to:

$$m'_V = m_1 - m_2 \quad (29)$$
$$T'_V = \frac{m_1 T_1 - m_1 T_V}{m_V} \quad (30)$$

where $m'_V$ and $T'_V$ are the time derivatives of the integrated temperature, $T_V$, and the integrated mass, $m_V$, respectively. The output pressure of the module is computed using the ideal gas law and the states according to:

$$p = \frac{m_V R T_V}{v} \quad (31)$$

where the volume $v = \phi_1$ is a design parameter of the volume component.

**The rotor component**

The rotor acceleration, $n'$, is calculated according to:

$$n' = \frac{\sum G_t - \sum G_c}{2\pi I} \quad (32)$$

where $\sum G_t$ is the sum of the turbine torques, $\sum G_c$ is the sum of the compressor torques and $I$ is the moment of inertia of the shaft.

## Acknowledgments

This work was funded by NFFP (the National Flight Research Program). The author would also like to thank Ulf Håll and Anders Lundbladh for ideas, comments and support.

## References

[1]Gold, H. and Rosenzweig, S., "A Method for Estimating Speed Response of Gas Turbine Engines," Tech. rep., NACA-RM-E51K21, 1952.

[2]Fawke, A. J. and Saravanamuttoo, H. I. H., "Digital Computer Simulation of the Dynamic Response of a Twin-Spool Turbofan with Mixed Exhausts," *Aeronautical Journal*, 1973.

[3]Sellers, J. F. and Daniele, C. J., "DYNGEN - A Program for Calculating Steady-State and Transient Performance of Turbojet and Turbofan Engines," Tech. rep., NASA TN D-7901, 1975.

[4]Daniele, C. J., Krosel, M. S., John, R. S., and Westerkamp, E. J., "Digital Computer Program for Generating Dynamic Turbofan Engine Models (DIGTEM)," Tech. rep., NASA-TM-83446, 1983.

[5]Schobeiri, M. T., Attia, M., and Lippke, C., "GETRAN: A Generic, Modularly Structured Computer Code for Simulation of Dynamic Behavior of Aero- and Power Generation Gas Turbine Engines," *Journal of Engineneering for Gas Turbines and Power*, 1994.

[6]Garrard, D., Davis, M. J., Hale, A., Chalk, J., and Savelle, S., "Analysis of gas turbine engine operability with the Aerodynamic Turbine Engine Code," *ISABE97-7034*, AIAA, September 1997, pp. 223–232.

[7]Chappel, M. A. and McLaughlin, P. W., "Approach to Modeling Continuous Turbine Engine Operation from Startup to Shutdown," *Journal of Propulsion and Power*, 1993.

[8]Dierckx, P., *Curve and Surface Fitting with Splines*, Oxford Science Publications, 1993.

[9]Grönstedt, U. T. J., *On Advanced Jet Engine Concepts*, Licentiate thesis, Chalmers University of Technology, 1996.

[10]http://www.netlib.org/.

[11]Shampine, L. F., Reichelt, M. W., and Kierzenka, J. A., *Solving Index 1 DAES in Matlab and Simulink*, Draft - http://www.smu.edu/ lshampin/current.html, 1999.

[12]*Matlab 5.3 and Simulink 2.3*.

[13]Brenan, K. E., Campbell, S. L., and Petzold, L. R., *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Elsevier Science Publishing Co., Inc., 1989.

[14]Press, W. H., Flannery, B., Teukosky, S., and Vetterling, W., *Numerical Recipies*, Cambridge University Press, 1994.

[15]Hindmarsch, A. C., "LSODE and LSODI, Two New Initial Value Ordinary Differential Equation solvers," *ACM Signum Newsletter*, 1980.

[16]Petzold, L., *Numerical methods for differential-algebraic equations-current status and future directions*, Clarendon Press, 1992, pp. 259–273.

[17]Shapiro, A. H., *The dynamics and thermodynamics of compressible fluid flow*, The Ronald Press Company, 1953.