# MATLAB Executable (MEX) Steady-State Solver and Linearization Tool for the Advanced Geared Turbofan 30,000lbf (AGTF30)

Thomas W. Kennings

NASA Glenn Research Center, Cleveland, Ohio

Donald L. Simon

NASA Glenn Research Center, Cleveland, Ohio

**Revisions List**

| Version | Date | User | Description |
|---------|------|------|-------------|
| V1.0 | 2024-05-30 | TWK | Initial Release |
| V1.1 | 2024-06-03 | TWK | Clarified description of engine states, inputs, and outputs |

# Contents

# Index of Figures

## Index of Tables

# Introduction

This document will help users of the MEX Steady-State and Linearization Tool software understand its capabilities and how to use the software effectively.

Controls development for gas turbine engines typically entails the design of multiple linear set point controllers spanning the operating envelope of the engine. Once designed, the resulting linear controllers can be combined in a piecewise linear fashion to provide full-envelope control functionality. This design process requires the generation of a linear state-space model at each operating point. This linearization procedure requires repeated iterative execution of simulation models to produce the required linear models. The generation of each individual linear model may only take a short amount of computer execution time, but oftentimes thousands of linear models are needed to support the development of controllers spanning the entire engine operating envelope.

Modeling and linearization of gas turbine engines is done using the NASA Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS). With the standard T-MATS linearization function, the process to generate a full-envelope batch of linear models takes an inordinate amount of time (multiple days on some computers). Repeated compilation of the Simulink model and overhead communications between the MATLAB workspace and Simulink represent a large portion of the time it takes to run linearization batches in T-MATS.

This software, the MATLAB Executable (MEX) Steady-State Solver and Linearization Tool, accelerates the controls development process by compiling the engine model into a standalone MEX function, which can be invoked repeatedly with little overhead. The user can specify a set of operating conditions, and, at each operating condition, the tool will solve the engine to steady-state and linearize the engine. A generic Newton-Raphson solver is included with the tool which can be adapted to other purposes. Program structure is summarized by Figure 1.
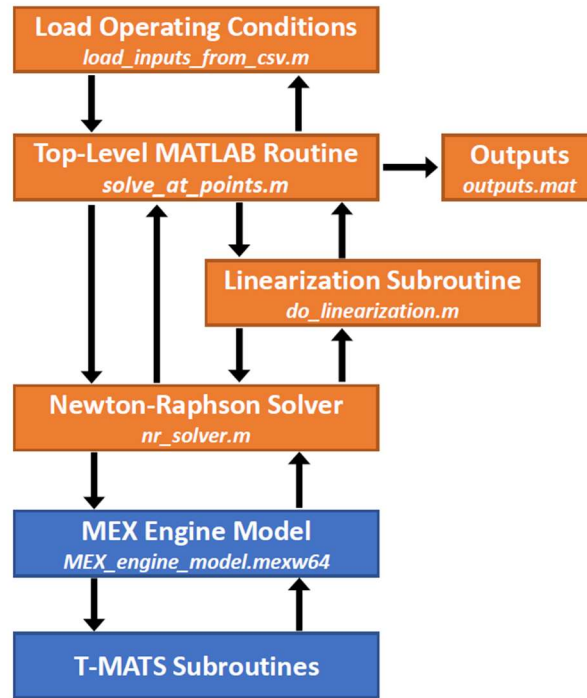
*Figure 1: Program flow diagram. Arrows indicate flow of data between subroutines. Italics text indicates the respective file for each subroutine. T-MATS subroutines are stored in numerous individual C files and are compiled into the MEX engine model.*

## AGTF30 Engine

The Advanced Geared Turbofan 30,000 (AGTF30) is a geared turbofan simulation that utilizes the Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS; Appendix D) to create a steady-state and dynamic engine model within MATLAB/Simulink. The engine model is based upon a futuristic geared turbofan concept and allows steady-state operation throughout the flight envelope. The AGTF30 model has two states: low-pressure shaft speed (N2) and high-pressure shaft speed (N3). Model inputs are enumerated in Appendix A, and model outputs in Appendix B. A link to download the un-modified AGTF30 engine model is provided in Appendix F. Locations within the engine are referred to by their station numbers, which are illustrated in Figure 2.
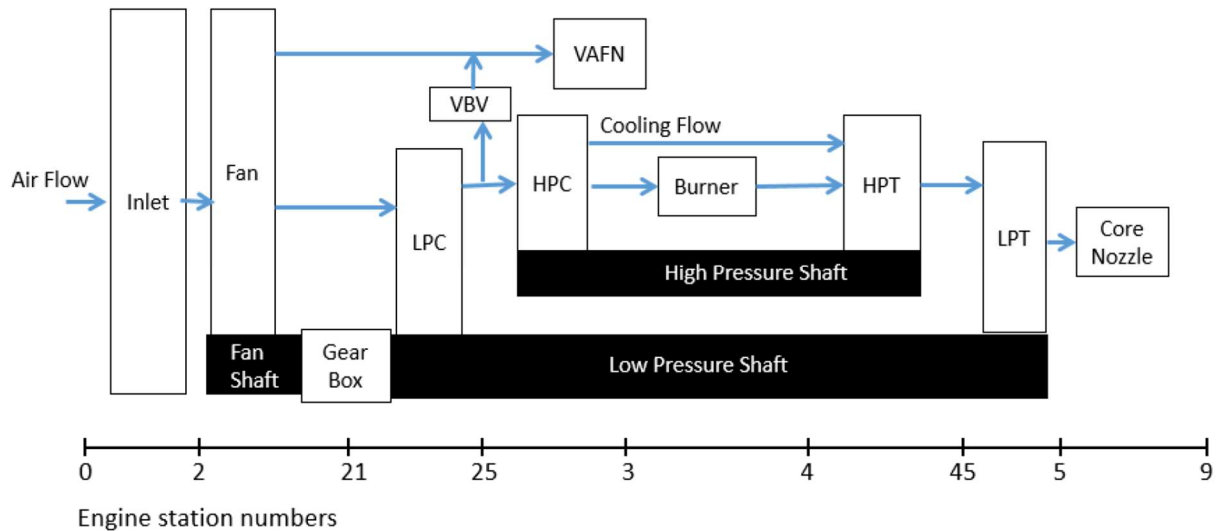
*Figure 2: AGTF30 engine station numbers. This figure is repeated, and its source listed, in Appendix H.*

**Electrified AGTF30 Engine**

The version of the AGTF30 engine modeled in the MEX-based solver and linearization tool includes optional electric motors coupled to the high- and low-pressure spool shafts as control actuators. These electric motors can inject or extract power from the shafts, which can be used to explore novel engine control methods. The mass and inertia of these motors are not modeled. This engine model is like an electrified version of the original AGTF30 engine model, called the "AGTF30-e" (Appendix G). Engine states are low- and high-pressure shaft speeds. Model inputs are enumerated in Appendix A, and model outputs in Appendix B.

# Requirements and Installation

To use the MEX Steady-State Solver and Linearization Tool, MATLAB must be installed. MATLAB versions 2021b through 2023b were tested and were compatible with the MEX solver tool. Other versions of MATLAB will most likely be compatible with the MEX solver tool, especially if the version is newer rather than older.

Once MATLAB is installed, clone the MEX Steady-State Solver and Linearization Tool from the NASA Github (https://github.com/nasa/-AGTF30-mex-solver-).

# Quick Start

### Specify Program Inputs

Specify a set of operating conditions and engine health parameters in the file *inputs.csv*. Each row represents a single operating condition which will be solved at steady state. To specify multiple operating conditions, put each operating condition on its own row. Each operating condition consists of the following information:

- Altitude
- Mach number
- Corrected fan speed (N1c)
- Ambient temperature differential from ISO standard day (Appendix C)
- Health parameters

The script *load_inputs_from_csv.m* will be called automatically to load the inputs from the comma separated value (CSV) file into a structured array in the MATLAB base workspace.

### Run the Solver

Run *solve_at_points.m* using MATLAB. The program will attempt to solve the engine system at each steady-state operating condition specified in *inputs.csv*. The program will perform linearization at each operating condition by methodically perturbing model inputs and measuring model outputs until the defined accuracy is achieved. If the solver is unable to find a solution for a given operating condition, it will give up and move on to the next operating condition.

### Observe Program Outputs

The program will write outputs to a MATLAB data file called *outputs.mat*. These outputs will be organized in a structured array, and will include:

- Altitude
- Mach number
- Corrected fan speed (N1c)
- Ambient temperature differential
- Health parameters
- Solved model steady-state conditions, including:
    - Independent variables
    - Input vector, U
    - State vector, X
    - Output vector, Y
- Linear state-space matrices (A, B, C, D)
- Solver diagnostic information

The output structured array can be viewed in MATLAB by double-clicking on the *outputs* object in the MATLAB workspace.

9

## Example of Program Execution

As an example, the definitions of five operating conditions are illustrated in Figure 3.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Altitude | Mach Number | N1c | dTamb | fan_WcMod | fan_PRMod | fan_EffMod | lpc_WcMod | lpc_PRMod | lpc_EffMod | hpc_WcMod | hpc_PRMod | hpc_EffMod | hpt_WcMod | hpt_EffMod | lpt_WcMod | lpt_EffMod |
| 2 | 0 | 0 | 1000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0.3 | 1500 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 10000 | 0.5 | 1750 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 25000 | 0.6 | 2000 | -10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 40000 | 0.8 | 2300 | 0 | -0.00441 | 0 | -0.0322 | -0.022 | 0 | -0.0175 | -0.00552 | 0 | -0.05985 | 0.00132 | -0.00225 | 0.0014 | -0.0027 |
| 7 | | | | | | | | | | | | | | | | | |

*Figure 3: Five operating conditions specified for solving and linearization.*

Once *inputs.csv* is created and saved, *solve_at_points.m* is opened in MATLAB (tested on R2021b-R2023b). Running *solve_at_points.m* produces the result shown in Figure 4.
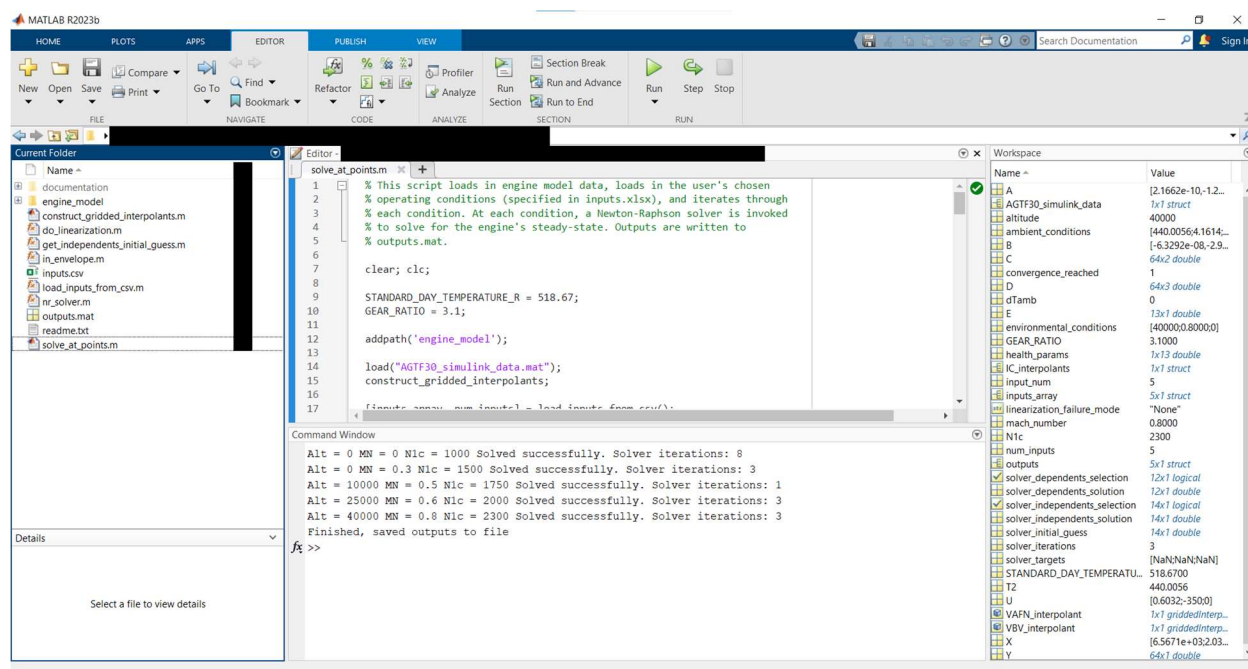


*Figure 4: Results of running "solve_at_points.m."*

The *outputs* structured array can be inspected by double-clicking on it in the workspace, as shown in Figure 5. Verification of generated linear models is demonstrated in Appendix I.
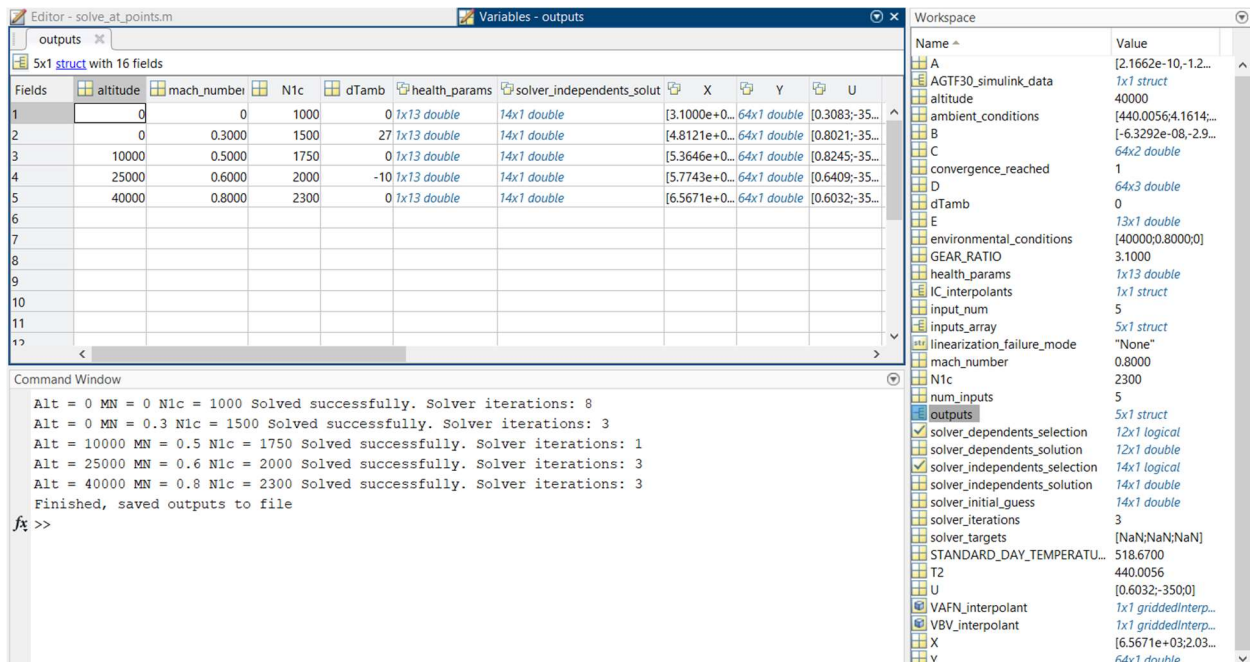


*Figure 5: Inspecting the "outputs" structured array in the MATLAB workspace.*

# Program Inputs

This section describes variables included in the specification of engine operating conditions. This section also discusses how users can change the method by which operating conditions are specified to the MEX solver program. The specification of operating conditions includes information about the engine's flight condition, as well as about the health of the engine.

### Flight Condition Inputs

Flight conditions are described by four variables:

- Altitude above sea level
- Mach number of inlet air
- Fan speed, corrected for inlet air temperature ("N1c")
- Ambient temperature differential from ISO standard day ("dTamb"; Appendix C)

Figure 6 shows the (altitude, Mach) pairs for which the AGTF30 engine is defined. Attempting to operate the engine at an undefined (altitude, Mach) pair will generate lots of warnings in the MATLAB terminal and will likely cause the program to throw an error.
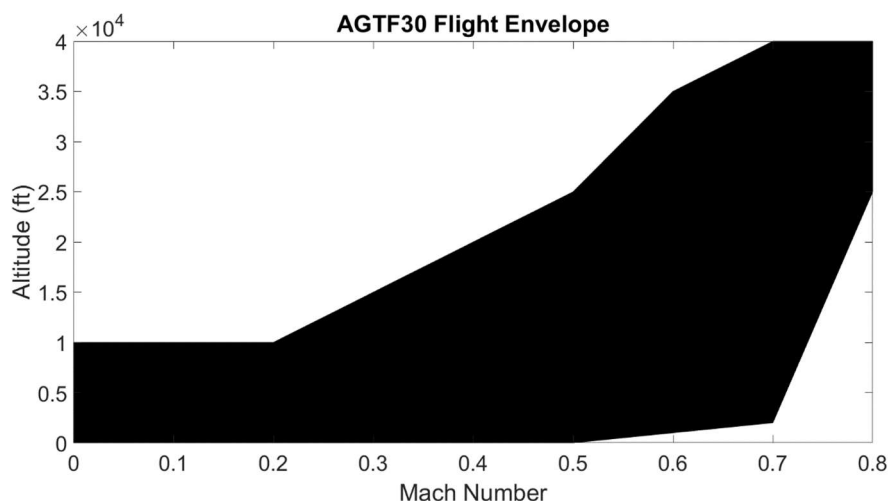


*Figure 6: AGTF30 engine flight envelope. The shaded region illustrates the set of altitude and Mach number pairs for which engine behavior is defined. Exceeding the shaded region will lead to undefined behavior of the AGTF30 engine.*

The solver will generally converge when corrected fan speeds between 850 and 2550 RPM are specified. However, requesting fan speeds which are unrealistic for the altitude and Mach number may cause the solver to fail to converge. This is mostly limited to the following two cases:

- Very low fan speed at a high altitude and high Mach number
- Very high fan speed at a low altitude and low Mach number

Ambient temperature differential was limited to +/- 30 degrees Fahrenheit in the original AGTF30 model, but this hard-limit was removed for the MEX engine model since it constrained the linearization process. Values of +/- 60 degrees Fahrenheit worked properly during initial testing with accurate results.

## Health Parameter Inputs

Health parameter values represent a percentage change of a given component's performance characteristics. For a new nominal engine, the health parameters are equal zero. Degradation is represented as a shift from the nominal value. For example, "lpc_WcMod = -0.022" will decrease the low-pressure compressor's flow capacity by 2.2%. These inputs allow end users to produce linear models under degradation conditions if so desired. The 13 available AGTF30 health parameters are summarized in Table 1.

*Table 1: Summary of health parameter program inputs. All inputs are floating point numbers representing fractional modification of a component's characteristic.*

| Index | Variable | Component | Quantity |
|---|---|---|---|
| 1 | fan_WcMod | Fan | Flow capacity |
| 2 | fan_PRMod | Fan | Pressure ratio |
| 3 | fan_EffMod | Fan | Efficiency |
| 4 | lpc_WcMod | Low-pressure compressor | Flow capacity |
| 5 | lpc_PRMod | Low-pressure compressor | Pressure ratio |
| 6 | lpc_EffMod | Low-pressure compressor | Efficiency |
| 7 | hpc_WcMod | High-pressure compressor | Flow capacity |
| 8 | hpc_PRMod | High-pressure compressor | Pressure ratio |
| 9 | hpc_EffMod | High-pressure compressor | Efficiency |
| 10 | hpt_WcMod | High-pressure turbine | Flow capacity |
| 11 | hpt_EffMod | High-pressure turbine | Pressure ratio |
| 12 | lpt_WcMod | Low-pressure turbine | Flow capacity |
| 13 | lpt_EffMod | Low-pressure turbine | Pressure ratio |

## Generating Inputs in Other Ways

The Solver and Linearization program comes packaged with a subroutine (*load_inputs_from_csv.m*) for loading inputs from the file *inputs.csv*. This subroutine is invoked in *solve_at_points.m* to populate the variables *inputs_array* and *num_inputs*. Users could substitute their own method of input-generation by replacing this function call. For example, operating conditions could be programmatically specified across the AGTF30's entire flight envelope to generate a piecewise-linear engine model. The solver program is agnostic to the method used to supply program inputs.

Inputs must be loaded into a structured array called *inputs_array* with the structure shown in Table 2. The *num_inputs* variable must also be specified and contain the number of operating conditions.

*Table 2: Overview of inputs structured array and fields for compatibility with the Solver and Linearization routines. All data types are doubles. The index "(i)" indicates the ith operating condition being specified.*

| Field Name | Dimensions |
|---|---|
| inputs_array(i).altitude | 1x1 |
| inputs_array(i).mach_number | 1x1 |
| inputs_array(i).N1c | 1x1 |
| inputs_array(i).dTamb | 1x1 |
| inputs_array(i).health_params | 1x13 |

# Program Outputs

The outputs are stored in a structured array, *outputs*, whose fields are summarized in Table 3. The structured array is written to a file *outputs.mat* in the same folder as *solve_at_points.m*.

*Table 3: Summary of program outputs. Appendix A details the contents of the X, Y, U, and E vectors. The index "(i)" indicates the ith operating condition evaluated.*

| Field Name | Description |
| --- | --- |
| outputs(i).altitude | Altitude, specified in program input. |
| outputs(i).mach_number | Mach number, specified in program input. |
| outputs(i).N1c | Corrected fan speed, specified in program input. |
| outputs(i).dTamb | Ambient temperature differential from ISO standard day (Appendix C), specified in program input. |
| outputs(i).health_params | Engine health parameters, specified in program input. |
| outputs(i).solver_independents_solution | Trim Newton-Raphson independent variables. |
| outputs(i).X | Solved model trim state vector. |
| outputs(i).Y | Solved model trim output vector conditions. |
| outputs(i).U | Solved model trim input vector. |
| outputs(i).E | Additional information about solution, for debugging. |
| outputs(i).A | Linear state-space matrix at operating condition. |
| outputs(i).B | Linear state-space matrix at operating condition. |
| outputs(i).C | Linear state-space matrix at operating condition. |
| outputs(i).D | Linear state-space matrix at operating condition. |
| outputs(i).linearization_failure_mode | Linearization failure reason, if applicable. |
| outputs(i).converged | Whether or not the steady-state solver succeeded. |

By default, the electric motors are excluded from the control input vector *U* to maintain parity with the original AGTF30 engine model. This means that the control vector only includes the fuel flow rate and the high- and low-pressure shaft power extractions are a constant 350 and 0 horsepower, respectively.

Electric motors can be included in the control input vector *U* by setting the *DO_ELECTRIC_MOTORS* constant to a value of *true* in *solve_at_points.m*. Electric motor power extractions are fixed at the same values as the non-electrified AGTF30 (-350 and 0 horsepower), but these constants could be modified by changing their values in *get_initial_guess.m*. Electric motor power extractions could also be modified by the solver if they were activated as solver independents. The process of activating solver independent variables is described in the "Flexible Newton-Raphson Solver" section (pg. 15).

# Flexible Newton-Raphson Solver

The Newton-Raphson solver script (*nr_solver.m*) iteratively adjusts model inputs (independent variables) to drive certain model outputs (dependent variables) to zero. Users can choose which independent and dependent variables to active to achieve different goals. Users can also specify targets which the solver will try to obtain.

**Independent and Dependent Variables**

"Active" independent variables are those which the solver is allowed to vary. Inactive independent variables will not be adjusted by the solver and will retain their values from the initial guess supplied to the solver.

The solver has converged when all active dependent variables have values less than their respective tolerances in the *Dtol* vector (Table 7). "Active" dependent variables are those which the solver tries to drive to zero. Inactive dependent variables are not used when calculating solver convergence.

Activation of independent and dependent variables is done using the vectors *solver_independents_selection* and *solver_dependents_selection*, respectively, in *solve_at_points.m*. In these vectors, *true* elements indicate the activation of an independent or dependent variable. The *solver_independents_selection* vector is summarized in Table 4 (pg. 16), and the *solver_dependents_selection* vector is summarized in Table 5 (pg. 16).

*The same number of independent and dependent variables must be activated, otherwise the solver will be under- or over-constrained and will throw an error accordingly.*

To maintain conservation of mass, dependent variables pertaining to flow errors (*W21err*, etc.) must be activated. Shaft acceleration dependent variables (*N2dot* and *N3dot*) should be activated if the user wishes to solve the engine model to steady-state operation. Dependent variables which are errors from solver targets should be activated only to operate the engine at those target values.

*Table 4: Summary of available solver independent variables.*

| Index | Variable | Units | Description |
|---|---|---|---|
| 1 | WIn | lbm/s | Mass flow rate entering inlet |
| 2 | FAN_RLIn | Dimensionless | R-line of fan |
| 3 | LPC_RLIn | Dimensionless | R-line of low-pressure compressor |
| 4 | HPC_RLIn | Dimensionless | R-line of high-pressure compressor |
| 5 | BPR | Dimensionless | Bypass/Core flow ratio |
| 6 | HPT_PR | Dimensionless | Pressure ratio of high-pressure turbine |
| 7 | LPT_PR | Dimensionless | Pressure ratio of low-pressure turbine |
| 8 | WfIn | lbm/hr | Fuel input |
| 9 | VAFNIn | Square inches | Area of variable area fan nozzle |
| 10 | VBVIn | Dimensionless | Fractional variable bleed valve position |
| 11 | N2In | RPM | Low-pressure shaft speed |
| 12 | N3In | RPM | High-pressure shaft speed |
| 13 | HPpwrIn | Horsepower | Electric motor power input to high-pressure shaft. Power extraction = negative value. |
| 14 | LPpwrIn | Horsepower | Electric motor power input to low-pressure shaft. Power extraction = negative value. |

*Table 5: Summary of available solver dependent variables.*

| Index | Variable | Units | Description |
|---|---|---|---|
| 1 | W21err | Dimensionless | Fan normalized flow error |
| 2 | W24err | Dimensionless | Low-pressure compressor normalized flow error |
| 3 | W36err | Dimensionless | High-pressure compressor normalized flow error. |
| 4 | W45err | Dimensionless | High-pressure turbine normalized flow error |
| 5 | W5err | Dimensionless | Low-pressure turbine normalized flow error |
| 6 | W8err | Dimensionless | Core nozzle normalized flow error |
| 7 | W18err | Dimensionless | Bypass nozzle normalized flow error |
| 8 | N2dot | RPM/s | Low-pressure shaft acceleration |
| 9 | N3dot | RPM/s | High-pressure shaft acceleration |
| 10 | LPC SM error | % | Error from *LPC_SM_target* solver target |
| 11 | Fnet error | lbf | Error from *Fnet_target* solver target |
| 12 | T45 error | Deg. Fahrenheit | Error from *T45_target* solver target |

To simulate the engine at a particular fuel input rate, fuel input should be *deactivated as an independent variable* so that the solver does not change the fuel input rate. This is done by setting the eighth element of the *solver_independents_selection* vector to *false*. The desired value of fuel flow should then be inserted into the *solver_initial_guess* array immediately before the solver is invoked, as demonstrated in Figure 7. This will mean overwriting the fuel flow rate initial guess which was supplied by the function *get_initial_guess*.

```matlab
%% Get initial guess for solver
solver_initial_guess = get_initial_guess(altitude, mach_number, N1c, dTamb, IC_interpolants);

solver_initial_guess(8) = desired_fuel_rate;

%% Run the solver
[solver_dependents_solution, solver_independents_solution, X, U, Y, E, convergence_reached, ...
    solver_iterations] = nr_solver(environmental_conditions, ...
    solver_initial_guess, solver_targets, health_params, ...
    solver_independents_selection, solver_dependents_selection);
```

*Figure 7: Demonstration of inserting desired fuel rate after "get_initial_guess" is called, and before the solver is invoked.*

**Solver Targets**

Solver targets are the targeted values for the respective quantity. For example, setting *Fnet_target* to 15,000 lbf (and activating the *Fnet_error* dependent variable) will cause the solver to drive the engine to a state at which it is producing a net 15,000 lbf of thrust. Users should consider whether their specified targets and/or their combination are reasonable, otherwise the solver will not be able to converge to the specified operating point (e.g., the AGTF30 cannot produce 1 million lbf of thrust, nor can the solver simultaneously achieve both a target *N2* and a target *Fnet*). Table 6 lists the solver targets available by default. More targets could be added for different objectives, as described in the "Adding Solver Targets" section (pg. 19).

*Table 6: Summary of solver targets available by default.*

| Index | Solver Target | Units | Description |
|-------|---------------|-------|-------------|
| 1 | **LPC_SM_target** | % | Low-pressure compressor stall margin |
| 2 | **Fnet_target** | lbf | The net thrust of the engine |
| 3 | **T45_target** | R | Total temperature at engine station 45 |

## Solver Parameters

Besides activation of independent and dependent variables, several parameters governing the solver process are available to modify in *nr_solver.m*. Table 7 explains the function of each setting.

*Table 7: Description of solver parameters available for modification in "nr_solver.m."*

| Parameter | Description |
|---|---|
| IMinMax | Matrix containing minimum and maximum bounds for each independent variable. These bounds ensure that the solver can only select valid independent variable values residing within the bounds of component maps and specified actuator limits. |
| Dtol | Vector containing tolerances for each dependent variable. If all active dependent variables are less than their respective tolerances, then convergence has been reached. |
| MapRange | Matrix containing minimum and maximum bounds for each engine component's *NcMap* (a dimensionless quantity representing the component speed). If these bounds are exceeded, the solver will throw an error. |
| MaxIter | The maximum number of solver iterations before the solver halts attempts to find a converged solution. |
| NRASS | Number of solver iterations before recalculating the Jacobian. Recalculating the Jacobian can help when the solver gets far from its initial condition. |
| JPerSS | Jacobian perturbation size, or iteration step size. A higher value will cause the solver to step towards a solution faster, but setting too high can cause the solver to overshoot solutions. |
| NumJPerSS | Number of solver iterations before *JPerSS* is reduced. This setting is only used in the 'second try' section of the solver script. Reducing *JPerSS* occasionally can help in situations where the solver is repeatedly overshooting a solution. |


## Multiple Convergence Attempts

The Newton-Raphson solver can attempt to solve the model multiple times, each time with a different set of solver parameters. This allows for the solver to, for instance, use parameters conducive to quick solutions for most problems. For conditions which are not solved using the 'quick' parameters, the solver can then try using a second set of parameters which may help reach convergence.

To specify sets of parameters, specify the parameters as members of their respective arrays near the top of *nr_solver.m*. The nth element in each array corresponds to the set of solver parameters used on the nth convergence attempt. For example, to specify the following rule:

*Use a Jacobian perturbation size of 0.001 on the first convergence attempt, then try increasing the perturbation size to 0.002 if convergence is not reached.*

*JPerSS_array* would need to be…

*JPerSS_array = [0.001, 0.002];*

All other parameters' arrays would need to be lengthened to match the length of the longest parameter array (2, in this case).

## Adding Solver Targets

This section demonstrates the addition of a new *T45* (total temperature after the high-pressure turbine) target to the solver. To not over- or under-constrain the solver, an additional independent must be activated (or a dependent disabled), such that the number of active independent and dependent variables are equal. If the solver is under- or over-constrained, the program will throw an error. For this example, an *N2* independent variable was activated in addition to the *T45* target dependent variable. This allowed the solver to vary the speed of the low-pressure shaft (and thus the fan speed) along with the previously defined independents to achieve the specified *T45* target value.

1. Open *C_files/MEX_engine_model.c*
2. Add the new solver target variable, *T45_target*, to the declaration of targets as demonstrated in Figure 8.

```
56
57        double *tar;
58        double LPC_SM_target, Fnet_target, T45_target;
59
```

*Figure 8: Adding a declaration of the "T_45" target variable.*

3. Update the maximum limit on the size of variables *m* and *n* to become 3 as shown in Figure 9 and indicated by the red boxes.

```
1168
1169        m = mxGetM(TAR_OUT);
1170        n = mxGetN(TAR_OUT);
1171        if (!mxIsDouble(TAR_OUT) || mxIsComplex(TAR_OUT) ||
1172        (MAX(m,n) != 3) || (MIN(m,n) != 1)) {
1173        mexErrMsgTxt("Requires that TAR_OUT be a 3 x 1 vector.");
1174        }
1175
```

*Figure 9: Updating the expected length of the "TAR_OUT" vector commensurate with the added solver target. This code will throw an error if it receives a "TAR_OUT" vector which is of different length than it expects.*

4. Increase the length of the *DEP_OUT* vector by one for each solver target added. Figure 10 illustrates increasing the length from 11 to 12, indicated by the red box.

```
1182
1183        /* Create a matrix for the return argument */
1184        DEP_OUT = mxCreateDoubleMatrix(12, 1, mxREAL);
1185        X_OUT = mxCreateDoubleMatrix(2, 1, mxREAL);
```

*Figure 10: Lengthening the "DEP_OUT" vector to a length of 12.*

5. Define the target from the input *tar* vector as shown in Figure 11 and indicated by the red box.

```
1223          Fnet_target  = tar[1]; /* Fnet target */
1224          T45_target = tar[2]; /* T45 target */
1225
```

*Figure 11: Setting the value of the "T45_target" variable.*

6. Add a new dependent in the program outputs section, near the end of the file. The new dependent variable should be the difference between the data of interest and its target, as illustrated in Figure 12. Tt45 is a variable which was calculated by the T-MATS subroutines earlier in the file, and is output in the Y-vector.

```
1745          DEP[10] = Fnet - Fnet_target;
1746          DEP[11] = Tt45 - T45_target;
1747
```

*Figure 12: Added dependent, which is the "T45" error from its target.*

7. Run *C_files/make_file_engine.m* to update the MEX engine model.
8. Open *nr_solver.m* and add tolerances for the new dependent in the *Dtol* vector. Figure 13 indicates the added tolerance in the red box, which is the same as all the others (1e-5).

```
41
42      %--- Define Dependent Vector Tolerances ----
43      Dtol = [1e-5;  %--- W21err ---
44              1e-5;  %--- W24aerr ---
45              1e-5;  %--- W36err ---
46              1e-5;  %--- W45err ---
47              1e-5;  %--- W5err ---
48              1e-5;  %--- W8err ---
49              1e-5;  %--- W18err ---
50              1e-5;  %--- N2dot ---
51              1e-5;  %--- N3dot ---
52              1e-5;  %--- LPC SM error ---
53              1e-5;  %--- Fnet error ---
54              1e-5]; %--- T45 error ---
55
```

*Figure 13: Expanding the "Dtol" vector in "nr_solver.m."*

9. Add the new target output to the *solver_targets* vector and *solver_dependents_selection* vector in *solve_at_points.m*. To enable the solver target, set the corresponding element in the *solver_dependents_selection* vector to *true*. Figure 14 indicates the relevant code by red boxes.

```
50
51    solver_dependents_selection = [ true;   %--- W21err ---
52          true;    %--- W24aerr ---
53          true;    %--- W36err ---
54          true;    %--- W45err ---
55          true;    %--- W5err ---
56          true;    %--- W8err ---
57          true;    %--- W18err ---
58          true;    %--- N2dot ---
59          true;    %--- N3dot ---
60          false;   %--- LPC SM error ---
61          false;   %--- Fnet error ---
62          true];   %--- T45_target ---
63
64    solver_targets = [  NaN;     %--- LPC_SM_target ---
65                        NaN;     %--- Fnet_target ---
66                        2400];   %--- T45_target ---
67
```

*Figure 14: Expanding the "solver_dependents_selection" and "solver_targets" vectors.*

10. Activate an additional independent, or deactivate a dependent, such that the number of active independent and dependent variables are equal. For this example, *N2* (low-pressure shaft speed) was activated as an independent as shown in Figure 15. This allowed the solver to vary *N2* (and, due to the engine's gearbox, the fan speed) to achieve the *T45* target.

```
35
36      solver_independents_selection = [   true;  %--- WIn ---
37          true;  %--- FAN_RLIn ---
38          true;  %--- LPC_RLIn ---
39          true;  %--- HPC_RLIn ---
40          true;  %--- BPR ---
41          true;  %--- HPT_PR ---
42          true;  %--- LPT_PR ---
43          true;  %--- WfIn ---
44          false;  %--- VAFNIn ---
45          false;  %--- VBVIn ---
46          true;  %--- N2In ---
47          true;  %--- N3In ---
48          false;  %--- HPpwrIn ---
49          false]; %--- LPpwrIn ---
50
```

*Figure 15: Activating "N2" as an independent so that the number of active independent and dependent variables match.*

11. Specify inputs and run *solve_at_points.m*. Figure 16 shows results with the solver target enabled.

```
Command Window
   Alt = 0 MN = 0 N1c = 1700 Solved successfully. Solver iterations: 11
   Finished, saved outputs to file
   T45 = 2400; Solved N1c = 2217.3469
fx >>
```

*Figure 16: Solver results with a "T45" target of 2400 degrees Rankine. "T45" was driven to 2400 degrees Rankine and corrected fan speed "N1c" was varied from its starting value of 1700.*

# Linearization Routine

## Algorithm Overview

If the steady-state solver converges to a solution at a given operating condition, the linearization function *do_linearization* is invoked to linearize the engine model about the operating condition. The linearization function returns state-space matrices A, B, C, and D. Note that any partial derivatives in the matrices involving shaft speeds use mechanical shaft speeds, rather than any type of corrected speeds.

The AGTF30 MEX engine model includes two states: high- and low-pressure shaft speeds. The model has three control inputs: fuel flow, and electric motor power extractions on the high- and low-pressure shafts. The electric motor inputs are optional and are disabled by default. The variable bleed valve and variable-area fan nozzle are not included in the control input vector, and are assumed to always run on-schedule, as was the case in the original AGTF30 model.

The A- and C-matrices are obtained by systematically perturbing each model state and measuring the change in model state derivatives and outputs. Each state is perturbed in both a positive and a negative direction by the same amount to form 'positive' and 'negative' versions of A- and C-matrices. These 'positive' and 'negative' versions are then averaged to calculate the final A- and C-matrices returned by the linearization routine.

The B- and D-matrices are obtained like the A- and C-matrices are, except that model inputs are perturbed instead of model states. In the AGTF30, the electric motor on the low-pressure shaft does not normally exert torque on the shaft (0 power extraction). This means that perturbing its power by any percentage would result in zero power change. To get around this, the low-pressure motor power is perturbed by the same amount as the high-pressure motor.

Verification of generated linear models is demonstrated in Appendix I.


## Settings

Users can change the amount by which the linearization script perturbs model states and inputs by editing the *PERTURBATION_FRACTION* parameter in *do_linearization.m*. By default, the perturbation fraction is 0.0003, or 0.03%. Setting a small perturbation fraction generally improves linearization accuracy. Using too large of a perturbation fraction can lead to the linearization function not capturing local behavior accurately.

# MEX Engine Model

The MATLAB executable (MEX) engine model and related files are stored in the *engine_model* folder. The main driver file, *MEX_engine_model.c*, invokes T-MATS component functions which are also stored in their own *.c files in the *engine_model* folder. The driver file also contains definitions of the engine components in the upper part of the file.

## Component Definitions

AGTF30 components are defined in the upper section of *MEX_engine_model.c*. Component definitions consist of variables, vectors, and matrices of data which govern the components' performance at different operating conditions. For an in-depth explanation of engine components and their data, refer to the NASA T-MATS documentation, available from the T-MATS Github (Appendix D). T-MATS documentation is also available as a NASA Technical Memorandum (Appendix E). Data used in component definitions was taken from the original AGTF30 Simulink model (Appendix F).

## Calling T-MATS Functions

In the middle section of the driver function, *MEX_engine_model.c*, T-MATS functions are invoked beginning with the most forward components and moving aft. Before invoking each T-MATS function, data are packaged into vectors as required for input/output to the T-MATS function. After executing the T-MATS function, output data are unpacked from the function output. These data are then ready to be used in the next T-MATS function and/or to be output from the engine model.

## Input/Output

Model inputs are processed and unpacked just after the definition of components in the driver function *MEX_engine_model.c*. This section of code checks that the model input has the expected dimensions and unpacks model input into separate variables. An enumeration of model input vectors is provided in Appendix A.

Model outputs are packaged into their respective vectors near the end of *MEX_engine_model.c*. An enumeration of model output vectors is provided in Appendix A.

If users modify the structure of model input or output vectors, they must be sure to update the definitions of the input/output vectors to have the appropriate length. Otherwise, the MEX function will throw an error because the vectors are of unexpected size. If input vector lengths are changed, update the dimension-checks which are performed as inputs are unpacked in *MEX_engine_model.c*.

**Tweaking the MEX Engine Model**

This section describes the process of modifying component definitions to produce an engine which has different performance characteristics than the default AGTF30. The Simulink Coder might be useful for accelerating the process of converting a Simulink model into a MEX file, though this was not attempted.

1. Before making modifications, it is recommended to create a backup of *MEX_engine_model.c*.
2. Open *MEX_engine_model.c*.
3. Locate where the component of interest is defined. Components are defined in the upper part of *MEX_engine_model.c*.
4. Change values in the components' definitions as desired. Refer to T-MATS documentation (Appendix A) for description of how components' variables affect their performance.
5. Save *MEX_engine_model.c*.
6. Run *make_file_engine.m* to compile a new MEX file.
7. The engine model is now updated and ready to use. Test the new engine model to confirm that changes produced reasonable results.

**Ambient MEX File**

Inlet conditions (enthalpy, temperature, pressure, etc.) are calculated based on the specified altitude, Mach number, and ambient temperature differential from ISO standard day (Appendix C). For more information about how the T-MATS ambient calculations are performed, refer to T-MATS documentation (Appendix D).

Ambient calculations are performed in the MEX engine model before other components' T-MATS functions are invoked. Ambient calculations are also split into a standalone MEX function. The standalone function is invoked in *solve_at_points.m* to determine the temperature at the inlet, which is used when specifying initial conditions for the MEX engine model.

# Troubleshooting

**I'm Getting the Error Message: "Must have same number of Independents and Dependents!"**

This error message appears when the *solver_independents_selection* and *solver_dependents_selection* vectors have different numbers of *true* elements. Having excess active independent variables under-constrains the solver, and having excess active dependent variables over-constrains the solver.

**I'm Seeing "Component Map Violation…" in the Terminal Output!**

This message appears when a component exceeds the range of speeds for which its behavior is defined. This can happen when running the engine at an extreme condition. An example of an extreme condition is when a very high altitude and Mach number is used, but the fan speed is low. In this case, the requested fan speed is likely to be less than the lower bound of corrected speed on the fan map. A solution in this case would be to increase the requested fan speed. An analogous issue could occur if the altitude and Mach number were very low, and the fan speed was very high.

**I'm Seeing "Cannot form invertible Jacobian matrix." in the Terminal Output!**

This usually happens when the solver reaches a point which produces a Jacobian Matrix with *N/A* elements. This can be due to a variety of factors, including strange behavior around specific points in the AGTF30 engine model. When dealing with this issue, start by observing the Jacobian which is calculated in *nr_solver.m* to see what is making the Jacobian non-invertible in your case.

**I'm Seeing "…beyond engine flight envelope…" in the Terminal Output!**

This happens when the requested altitude, Mach number, and/or difference from standard ambient temperature exceeds the envelope over which the AGTF30 engine model is defined. The AGTF30 flight envelope is described in the "Program Inputs" section of this document (pg. 12).

**The Solver isn't Converging at my Requested Operating Conditions!**

Solver non-convergence is likely accompanied by other messages, so check the terminal output at that operating condition for clues about the root cause. Tweaking solver parameters, such as the step size, may help the solver converge more reliably. During testing, there were *very rare* operating conditions for which the solver inexplicably could not converge despite the developers' best efforts.

**I'm Getting an Error About the "cd" Function Not Working!**

Ensure that you are running *solve_at_points.m* within the proper directory in the MATLAB IDE. The file *solve_at_points.m* should be visible in the left-hand pane. To quickly navigate to the correct directory, right-click the *solve_at_points.m* tab on the middle pane and select "Change current folder to…"

**The Electric Motors Aren't Included in the U Vector!**

By default, the electric motors are excluded from the control input vector *U* to maintain parity with the original AGTF30 engine model. Electric motors can be included in the control input vector *U* by setting the *DO_ELECTRIC_MOTORS* constant to a value of *true* in *solve_at_points.m*.

# Appendices

## Appendix A: MEX AGTF30 Engine Model Inputs

MEX engine model inputs are itemized in this section. Note that these are different from the MEX solver and linearization tool *program* inputs, which are described in the section "Program Inputs" (pg. 12) Outputs are itemized in Appendix B (pg. 29). Some tables are duplicated from earlier sections for the reader's convenience. Appendix H (pg. 32) may be useful for understanding station numbering in the AGTF30 engine.

*Table 8: Itemization of environmental conditions vector "ENV_IN."*

| Index | Variable | Units | Desc |
|---|---|---|---|
| 1 | Altitude | Feet | Altitude above sea level, in feet |
| 2 | Mach Number | Dimensionless | Mach number of inlet air |
| 3 | dTamb | Deg. Fahrenheit | Ambient temperature differential from ISO standard day (Appendix C) |

*Table 9: Itemization of independent variables vector "CMD_IN."*

| Index | Variable | Units | Description |
|---|---|---|---|
| 1 | WIn | lbm/s | Mass flow rate entering inlet |
| 2 | FAN_RLIn | Dimensionless | R-line of fan |
| 3 | LPC_RLIn | Dimensionless | R-line of low-pressure compressor |
| 4 | HPC_RLIn | Dimensionless | R-line of high-pressure compressor |
| 5 | BPR | Dimensionless | Bypass ratio; Ratio between the mass flow rate of the bypass stream to the mass flow rate entering the core |
| 6 | HPT_PR | Dimensionless | Pressure ratio of high-pressure turbine |
| 7 | LPT_PR | Dimensionless | Pressure ration of low-pressure turbine |
| 8 | WfIn | lbm/hr | Fuel input |
| 9 | VAFNIn | Square inches | Area of variable area fan nozzle |
| 10 | VBVIn | Dimensionless | Fractional variable bleed valve position |
| 11 | N2In | RPM | Low-pressure shaft speed |
| 12 | N3In | RPM | High-pressure shaft speed |
| 13 | HPpwrIn | Horsepower | Electric motor power input to high-pressure shaft. Power extraction = negative value. |
| 14 | LPpwrIn | Horsepower | Electric motor power input to low-pressure shaft. Power extraction = negative value. |

*Table 10: Itemization of solver targets vector "TAR_OUT."*

| Index | Solver Target | Units | Description |
|---|---|---|---|
| 1 | **LPC_SM_target** | % | Low-pressure compressor stall margin |
| 2 | **Fnet_target** | lbf | The net thrust of the engine |
| 3 | **T45_target** | R | Gas temperature at engine station 45 |

*Table 11: Itemization of health parameter vector "HEALTH_PARAMS_IN."*

| Index | Variable | Component | Quantity |
|------:|----------|-----------|----------|
| 1 | fan_WcMod | Fan | Flow capacity |
| 2 | fan_PRMod | Fan | Pressure ratio |
| 3 | fan_EffMod | Fan | Efficiency |
| 4 | lpc_WcMod | Low-pressure compressor | Flow capacity |
| 5 | lpc_PRMod | Low-pressure compressor | Pressure ratio |
| 6 | lpc_EffMod | Low-pressure compressor | Efficiency |
| 7 | hpc_WcMod | High-pressure compressor | Flow capacity |
| 8 | hpc_PRMod | High-pressure compressor | Pressure ratio |
| 9 | hpc_EffMod | High-pressure compressor | Efficiency |
| 10 | hpt_WcMod | High-pressure turbine | Flow capacity |
| 11 | hpt_EffMod | High-pressure turbine | Pressure ratio |
| 12 | lpt_WcMod | Low-pressure turbine | Flow capacity |
| 13 | lpt_EffMod | Low-pressure turbine | Pressure ratio |

## Appendix B: MEX AGTF30 Engine Model Outputs

MEX engine model outputs are itemized in this section. Inputs are itemized in Appendix A (pg. 27). Some tables are duplicated from earlier sections for the reader's convenience. Appendix H (pg. 32) may be useful for understanding station numbering in the AGTF30 engine.

*Table 12: Itemization of model dependent variables vector "DEP."*

| Index | Variable | Units | Description |
|------:|----------|--------------|-------------|
| 1 | W21err | Dimensionless | Fan normalized flow error |
| 2 | W24err | Dimensionless | Low-pressure compressor normalized flow error |
| 3 | W36err | Dimensionless | High-pressure compressor normalized flow error. |
| 4 | W45err | Dimensionless | High-pressure turbine normalized flow error |
| 5 | W5err | Dimensionless | Low-pressure turbine normalized flow error |
| 6 | W8err | Dimensionless | Core nozzle normalized flow error |
| 7 | W18err | Dimensionless | Bypass nozzle normalized flow error |
| 8 | N2dot | RPM/s | Low-pressure shaft acceleration |
| 9 | N3dot | RPM/s | High-pressure shaft acceleration |
| 10 | LPC SM error | % | Error from *LPC_SM_target* solver target |
| 11 | Fnet error | lbf | Error from *Fnet_target* solver target |
| 12 | T45 error | R | Error from *T45_target* solver target |

*Table 13: Itemization of state vector "X."*

| Index | Variable | Units | Description |
|------:|----------|-------|-------------|
| 1 | N2 | RPM | Low-pressure shaft speed |
| 2 | N3 | RPM | High-pressure shaft speed |

*Table 14: Itemization of model input vector "U" for the electrified AGTF30 model. The non-electrified AGTF30 model will not have electric motor elements. Note that while the electrified engine model will always output this vector, the MEX solver program will remove the HPpwrIn and LPpwrIn elements unless "DO_ELECTRIC_MOTORS" is set to "true" in solve_at_points.m.*

| Index | Variable | Units | Description |
|------:|----------|------------|-------------|
| 1 | WfIn | lbm/hr | Fuel input rate |
| 2 | HPpwrIn | Horsepower | Power added to the high-pressure shaft by the electric motor |
| 3 | LPpwrIn | Horsepower | Power added to the low-pressure shaft by the electric motor |

*Table 15: Itemization of conventional output vector "Y."*

| Index | Variable | Units | Component/Location | Quantity |
|------:|----------|-------|--------------------|----------|
| 1 | N1mech | RPM | Fan | Mechanical speed |
| 2 | N2mech | RPM | Low-pressure shaft | Mechanical speed |
| 3 | N3mech | RPM | High-pressure shaft | Mechanical speed |
| 4 | W0 | lbm/s | Station 0 | Mass flow rate |
| 5 | Tt0 | R | Station 0 | Total temperature |
| 6 | Pt0 | psi | Station 0 | Total pressure |
| 7 | W2 | lbm/s | Station 2 | Mass flow rate |
| 8 | Tt2 | R | Station 2 | Total temperature |
| 9 | Pt2 | psi | Station 2 | Total pressure |
| 10 | W21 | lbm/s | Station 21 | Mass flow rate |
| 11 | Tt21 | R | Station 21 | Total temperature |
| 12 | Pt21 | psi | Station 21 | Total pressure |
| 13 | W13 | lbm/s | Station 13 | Mass flow rate |
| 14 | Tt13 | R | Station 13 | Total temperature |
| 15 | Pt13 | psi | Station 13 | Total pressure |
| 16 | W15 | lbm/s | Station 15 | Mass flow rate |
| 17 | Tt15 | R | Station 15 | Total temperature |
| 18 | Pt15 | psi | Station 15 | Total pressure |
| 19 | W17 | lbm/s | Station 17 | Mass flow rate |
| 20 | Tt17 | R | Station 17 | Total temperature |
| 21 | Pt17 | psi | Station 17 | Total pressure |
| 22 | W22 | lbm/s | Station 22 | Mass flow rate |
| 23 | Tt22 | R | Station 22 | Total temperature |
| 24 | Pt22 | psi | Station 22 | Total pressure |
| 25 | W23 | lbm/s | Station 23 | Mass flow rate |
| 26 | Tt23 | R | Station 23 | Total temperature |
| 27 | Pt23 | psi | Station 23 | Total pressure |
| 28 | W24a | lbm/s | Station 24a | Mass flow rate |
| 29 | Tt24a | R | Station 24a | Total temperature |
| 30 | Pt24a | psi | Station 24a | Total pressure |
| 31 | W24 | lbm/s | Station 24 | Mass flow rate |
| 32 | Tt24 | R | Station 24 | Total temperature |
| 33 | Pt24 | psi | Station 24 | Total pressure |
| 34 | W25 | lbm/s | Station 25 | Mass flow rate |
| 35 | Tt25 | R | Station 25 | Total temperature |
| 36 | Pt25 | psi | Station 25 | Total pressure |
| 37 | W36 | lbm/s | Station 36 | Mass flow rate |
| 38 | Tt36 | R | Station 36 | Total temperature |
| 39 | Pt36 | psi | Station 36 | Total pressure |
| 40 | Ps36 | psi | Station 36 | Static pressure |
| 41 | W4 | lbm/s | Station 4 | Mass flow rate |
| 42 | Tt4 | R | Station 4 | Total temperature |
| 43 | Pt4 | psi | Station 4 | Total pressure |
| 44 | W45 | lbm/s | Station 45 | Mass flow rate |
| 45 | Tt45 | R | Station 45 | Total temperature |

| Index | Variable | Units | Component/Location | Quantity |
|---|---|---|---|---|
| 46 | Pt45 | psi | Station 45 | Total pressure |
| 47 | W48 | lbm/s | Station 48 | Mass flow rate |
| 48 | Tt48 | R | Station 48 | Total temperature |
| 49 | Pt48 | psi | Station 48 | Total pressure |
| 50 | W5 | lbm/s | Station 5 | Mass flow rate |
| 51 | Tt5 | R | Station 5 | Total temperature |
| 52 | Pt5 | psi | Station 5 | Total pressure |
| 53 | W7 | lbm/s | Station 7 | Mass flow rate |
| 54 | Tt7 | R | Station 7 | Total temperature |
| 55 | Pt7 | psi | Station 7 | Total pressure |
| 56 | Fdrag | lbf | N/A | Drag force |
| 57 | Fg18 + Fg8 | lbf | N/A | Gross engine thrust (bypass + core) |
| 58 | Fnet | lbf | N/A | Net thrust |
| 59 | Fg18 | lbf | N/A | Gross bypass thrust |
| 60 | Fg8 | lbf | N/A | Gross core thrust |
| 61 | TSFC | (lbm/hr)/lbf | N/A | Thrust-specific fuel consumption |
| 62 | SMMap21 | % | Fan | Stall margin map |
| 63 | SMMap24a | % | Low-pressure compressor | Stall margin map |
| 64 | SMMap36 | % | High-pressure compressor | Stall margin map |

*Table 16: Itemization of diagnostic output vector "E."*

| Index | Output | Units | Description |
|---|---|---|---|
| 1 | Nc24a | RPM | Corrected low-pressure compressor speed |
| 2 | Nc36 | RPM | Corrected high-pressure compressor speed |
| 3 | NcMap21 | % | Fan speed map |
| 4 | NcMap24a | % | Low-pressure compressor speed map |
| 5 | NcMap36 | % | High-pressure compressor speed map |
| 6 | NcMap45 | % | High-pressure turbine speed map |
| 7 | NcMap5 | % | Low-pressure turbine speed map |
| 8 | Trq21 | lb-ft | Torque applied to low-pressure shaft due to fan |
| 9 | Trq24a | lb-ft | Torque applied to low-pressure shaft due to low-pressure compressor |
| 10 | Trq36 | lb-ft | Torque applied to high-pressure shaft due to high-pressure compressor |
| 11 | Trq45 | lb-ft | Torque applied to high-pressure shaft due to high-pressure turbine |
| 12 | Trq5 | lb-ft | Torque applied to low-pressure shaft due to low-pressure turbine |
| 13 | Ps0 | psi | Static pressure at station 0. |

## Appendix C: International Standard Atmosphere

Information about the International Standard Atmosphere is published by the International Organization for Standards (ISO). The official publication is found at https://www.iso.org/standard/7472.html

## Appendix D: NASA Toolbox for the Modeling of Thermodynamic Systems (T-MATS) Github

https://github.com/nasa/T-MATS

## Appendix E: NASA T-MATS Users Guide Technical Memorandum

https://ntrs.nasa.gov/citations/20140012486

## Appendix F: NASA Advanced Geared Turbofan 30,00lbf Engine (AGTF30)

https://github.com/nasa/AGTF30

## Appendix G: NASA Electrified AGTF30 (AGTF30-e)

https://github.com/nasa/AGTF30/tree/AGTF30e

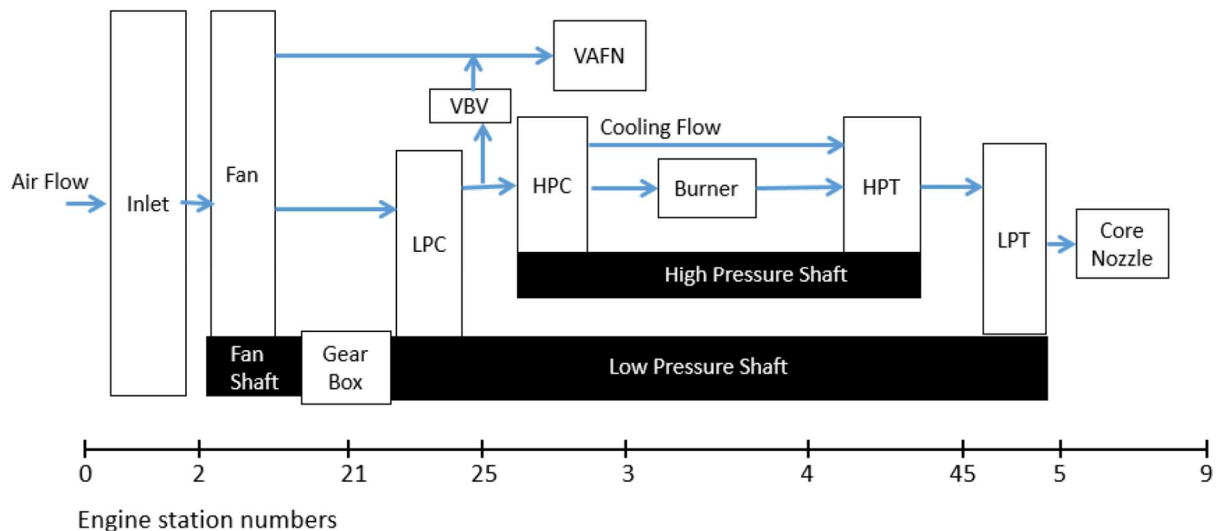## Appendix H: AGTF30 Station Numbers and Their Locations



Figure source: Jeffryes W. Chapman and Jonathan S. Litt. "Control Design for an Advanced Geared Turbofan Engine", 53rd AIAA/SAE/ASEE Joint Propulsion Conference, AIAA Propulsion and Energy Forum, (AIAA 2017-4820)

## Appendix I: Linear Model Verification

The accuracy of generated linear models was verified by generating a piecewise-linear model spanning the AGTF30's flight envelope and comparing its response to that of the nonlinear AGTF30 T-MATS model. Figure 17 through Figure 19 compare transient responses using the nonlinear and piecewise-linear model, showing excellent agreement. In the figures, altitude, Mach number, and power lever angle were dramatically varied.
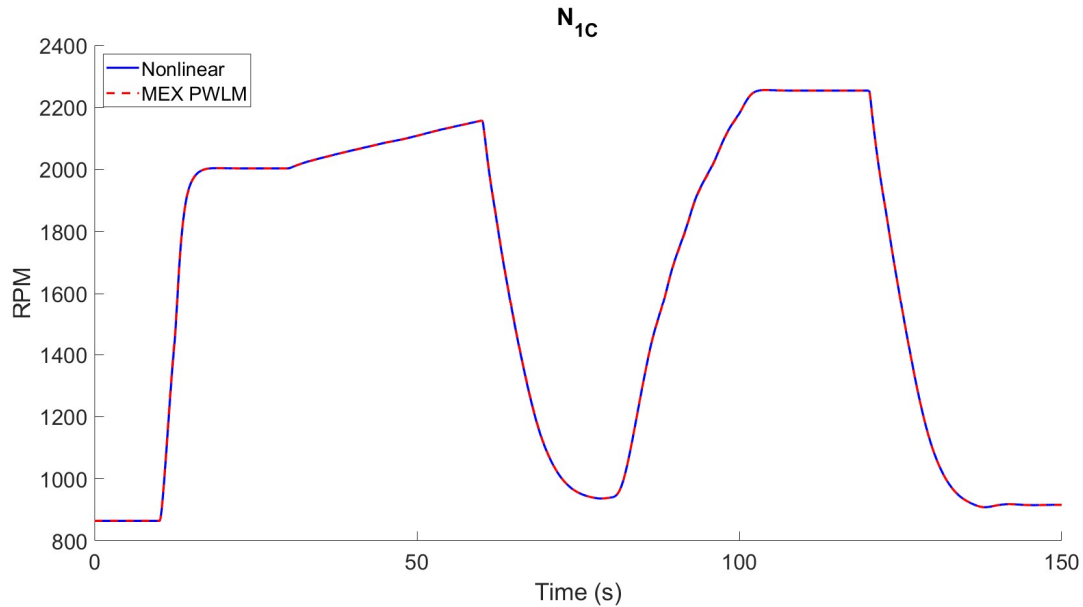


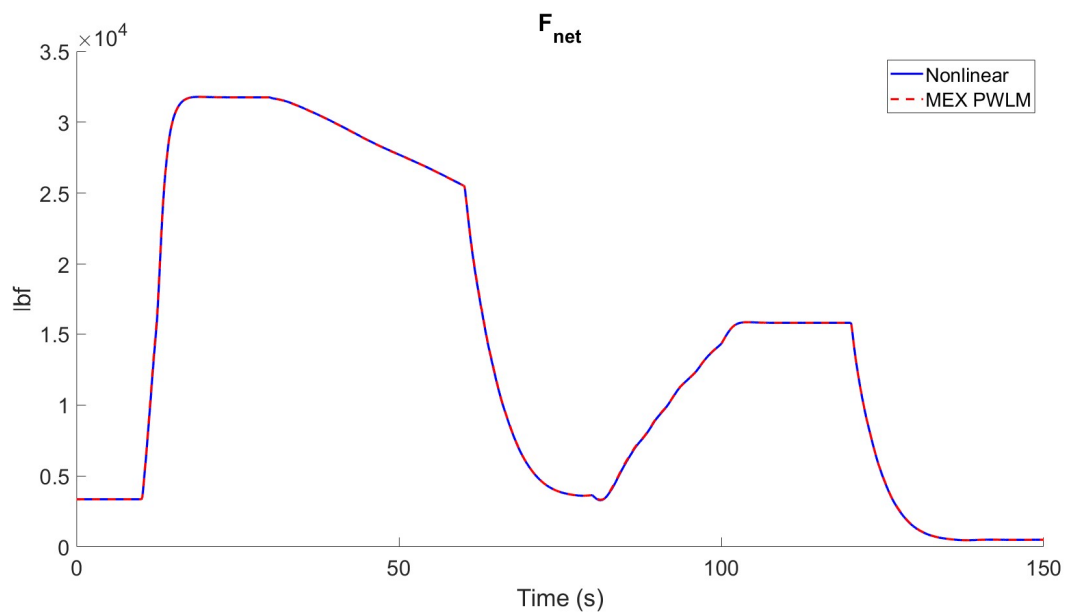*Figure 17: Piecewise-linear model vs. nonlinear engine model corrected fan speed.*



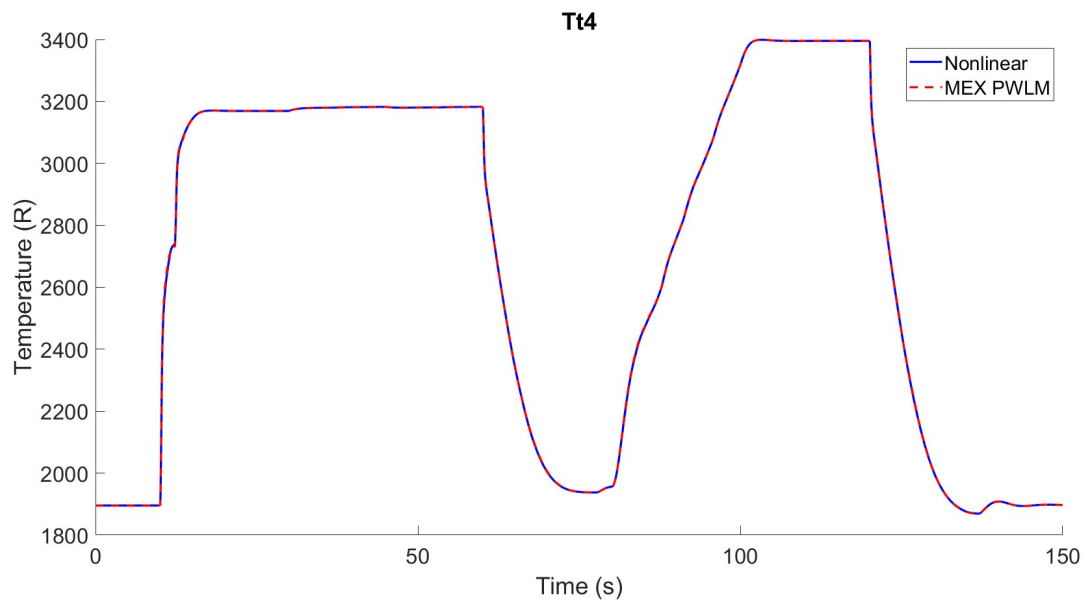*Figure 18: Piecewise-linear model vs. nonlinear engine model net thrust.*

*Figure 19: Piecewise-linear model vs. nonlinear engine model total temperature at combustor.*