Online Trimming of Flight Dynamics Models Using the 2Simulate Realtime Simulation Framework

Jürgen Gotschlich[1]
DLR (German Aerospace Center), Institute of Flight Systems, Braunschweig, 38108, Germany
Michael Jones[2]
DLR (German Aerospace Center), Institute of Flight Systems, Braunschweig, 38108, Germany

Real-time simulation requires the definition of scenarios, which may be defined on the ground or in the air. In-flight scenarios usually begin in a trimmed flight condition, so that the simulation model is in a steady state. At the German Aerospace Center (DLR), the Air Vehicle Simulator (AVES) is operated for flight research and simulator studies. Research simulators require a high degree of flexibility to modify scenarios. Moreover, the flexibility to modify the simulation model poses a challenge to both the user interface and the trim procedure. Therefore, an adjustable user interface is required to allow the user to run the trim calculation online for immediate operation in a simulation run. This paper presents an implementation of a generic approach to trim flight dynamic models for use in research flight simulators. This implies both the implementation of a graphical user interface as well as the trim algorithm and their interaction. An example is demonstrated for a helicopter model used for research activities at DLR.

Nomenclature

2SimCC	=	2Simulate Control Center
2SimMC	=	2Simulate Model Control
2SimRT	=	2Simulate Realtime Framework
ACT/FHS	=	Active Control Technology/Flying Helicopter Simulator
AVES	=	Air Vehicle Simulator
COTS	=	Commercial off The Shelf
DLR	=	German Aerospace Center
FMI	=	Functional Mockup Interface
GUI	=	Graphical User Interface
SDK	=	Software Development Kit
TCP	=	Transmission Control Protocol
TSML	=	2Simulate Model Language
UDP	=	User Datagram Protocol

I. Introduction

GENERALLY, flight simulators must have the flexibility to facilitate different functions and scenarios (mass, position, attitude, velocities, flaps, slats, etc.) to support experimentation under specific circumstances. For each scenario, the simulation model is usually initialized in a steady state condition, also known as a trimmed state or equilibrium point. There are a number of possible ways to provide the dynamic model with the corresponding initial conditions. To trim the vehicle in flight one requires both the mathematical problem to calculate initial values and the simulation environment to configure these values.

[1] Research Scientist, Flight Dynamics and Simulation, Lilienthalplatz 7, 38108 Braunschweig, Germany, juergen.gotschlich@dlr.de.
[2] Research Engineer, Rotorcraft Department, Lilienthalplatz 7, 38108 Braunschweig, Germany, michael.jones@dlr.de.

For most simulation devices, it is sufficient to make use of pre-defined trim scenarios to start the simulation model in a steady-state condition. This calculation is performed in a pre-step, prior to full simulation. One option is to have different executable programs for the simulation model, each with its individual steady state. In this case the result of an offline trim calculation is used to set fixed initial values to the state and input variables. This requires the simulation to be restarted for any changes in conditions. A better approach is to store the trim result in pre-calculated datasets. The datasets are loaded and used later by the realtime model. These pre-calculated datasets are only valid as long as a configuration management ensures that no changes have occurred to other simulation parameters, particularly with regards to changes to the simulation model. Both of these solutions are labor intensive, and lack the flexibility required for a research simulation facility. These methods offer limited flexibility and cost significant time even for slight modifications (turnaround-time). Furthermore, it is possible that discrepancies exist between the models at the time of the trim calculation and the time of activation of the trim state (version discrepancy).

Another disadvantage, particularly for realtime simulation, is the need to keep an additional offline model ready to prepare trim calculations. This may also lead to version discrepancies. An essential improvement in terms of flexibility, expenditure of time and quality assurance is the ability to use an ‘online’ trim functionality, as presented in this paper. The term online describes direct access to the realtime simulation model. Unlike the approaches that use pre-defined trim data, this approach ensures that the calculated parameters match with the used model.

Full flight simulators used for research purposes have different requirements to flight simulators for training purposes. The demands for flexibility lead to additional challenges. Both essential modifications to the flight model must be possible for testing purposes whilst ensuring that validated standard operation is possible. Changes to the flight model may uncover different model behavior caused by varied parameters as well as modified input and output interfaces, which certainly influence the whole simulator infrastructure. In both cases the user would like to have an easily accessible and manageable interface that fulfills all requirements to restart the simulator from a newly trimmed steady state. Therefore the calculation of the steady state needs to be performed online if a new trim scenario has been specified. Allowing the user to specify a new trim scenario in such a flexible simulator infrastructure is an aspect which needs special attention and needs an adjustable graphical user interface (GUI).

Numerous papers can be found in the literature to calculate steady state conditions for flight dynamic models.[1] Some solve the mathematical problem with an analytic approach,[2,3] which requires a number of assumptions and restrictions, based on physical dependencies, which are not acceptable for a general approach as attempted here. Usually the trim problem is presented as an optimization problem. Some research is done in the area of new algorithm approaches. Ref. 4 and 5 give a good general overview. Some of these methods can be classified as neurocomputing algorithms. Although it is claimed that trim-routines are generally applicable, likewise a number of assumptions are made which would restrict the usage for a general approach. However, most applications use classical convex gradient methods using direct numerical calculations. Since the focus of this paper is not on the mathematical aspect of the trim algorithm but more the flexible usage, a classical numerical approach using Newton-Raphson iteration is used.

This paper describes an implementation of a general online trimming feature into an existing simulator infrastructure. Challenges are the user presentation of the trim problem of a general simulation model, as well as the procedure to calculate the trim values inside a flexible simulation environment.

II. AVES Software Infrastructure

A. AVES

At the DLR-Institute of Flight Systems in Braunschweig, Germany, the Air Vehicle Simulator (AVES) (Fig.1) is used for flight research and simulator studies.[6] It has been designed as a modular, flexible platform for research on both fixed- and rotary-winged flight vehicles. Currently, AVES is equipped with two interchangeable cockpits: an A320 cockpit and an EC135 cockpit. Both cockpits can be used on a six degree-of-freedom hexapod motion platform and are used to run a number of different simulation applications, including different vehicle simulation models.

To allow flexible usage, testing and validation of the software, the AVES Software Development Kit (SDK) has been designed and plays a vital role by running applications either in AVES, software test devices or personal desktop computers.[7] Flexible configuration possibilities allow to distinguish between a software development environment (SDE) and runtime environment (RTE). Users have the possibility to operate appropriate simulation runs under all variations. Although differences exist in test environments (like different hardware, e.g. motion platform), the presented trim feature is in operation under all this configurations. Users can prepare and test trim scenarios at their desktop for later use in the flight simulator.

Figure 1. AVES Simulator

B. 2Simulate

For use in AVES, the software framework 2Simulate has been developed, which is used as the backbone for the simulation infrastructure.[8] It is a generic toolbox and is used for a number of projects and different purposes. 2Simulate is designed focusing on four main aspects:

- Realtime data processing and concurrency
- Simulation of dynamic models
- Scalability (runs on realtime simulator, software test device as well as on desktop)
- Interactive simulation control

To achieve these goals, 2Simulate is separated into three packages: 2SimRT, 2SimMC, 2SimCC. The 2Simulate Realtime Framework (2SimRT) comes as an API-library and enables the user to generate realtime applications, so called 2Simulate targets. They are used to fulfill realtime requirements according to the first two aspects (i.e. data processing and simulation of dynamic models, Fig. 2).

2SimRT and 2SimMC are available for the operating systems QNX, Windows and Linux, of course with different quality to the deterministic time behavior, which is essential for the realtime requirements. 2Simulate targets run a number of tasks with dedicated functionality, instantiated from one of the available task classes shown in the class diagram (Fig. 3). Users can add their dedicated needs within callback functions, e.g. for data processing, input/output data handling or processing complex transfer protocols. 2Simulate targets can run a number of simulation models.[9] Currently, possible model variants are Simulink models, compiled by Mathworks Simulink Coder as well as TSML-models (2Simulate Model Language). The simulation models are encapsulated within the 2Simulate Model Control Framework (2SimMC). Its main functionality is implemented in the TSimMcModelCtrl class which controls the model (set initial conditions, start, pause, trim, data management, etc.). Since Simulink models have an embedded solver inside the generated code, TSML models use 2Simulate's solver. The individual model variants inherit beside the general TSimMdlTask from dedicated model control classes and spread different 2SimRT model tasks (TSimSimulinkTask, TSimTsmlTask, etc.) with their realtime behavior.

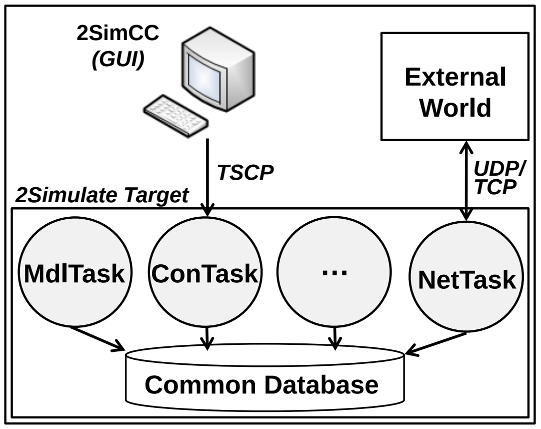


Figure 2. Typical 2Simulate Target Application

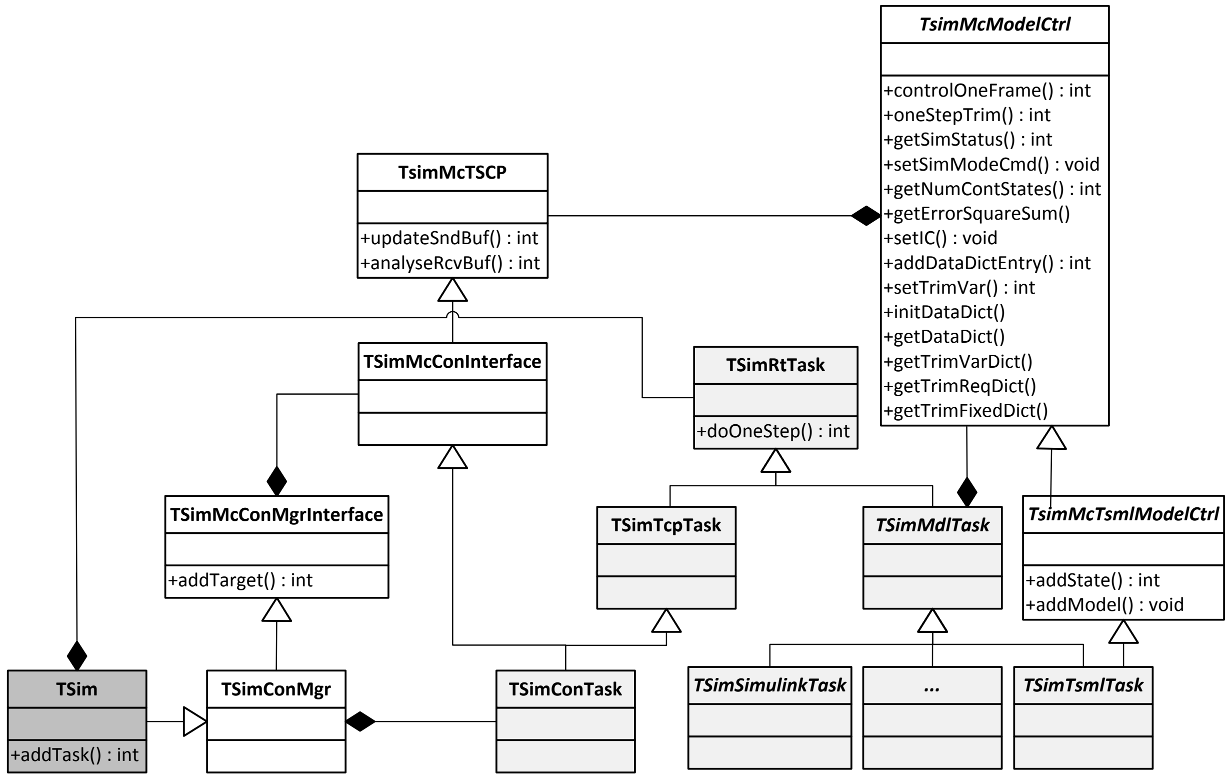
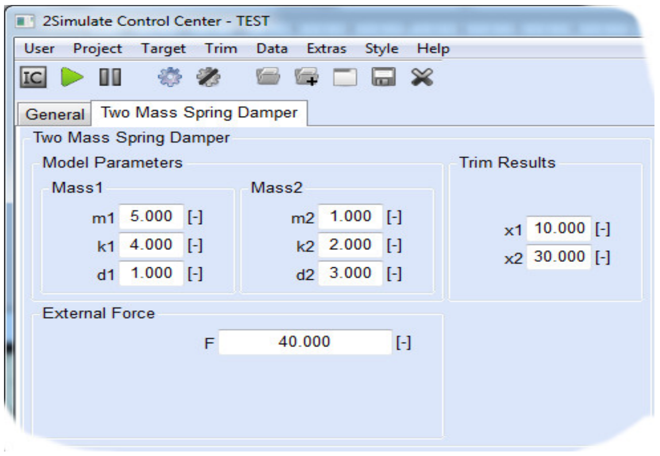


Figure 3. 2Simulate Class Diagram

The 2Simulate Control Center (2SimCC) is the graphical user interface to control and manage targets and models. A global project, user and data display management is available. Since 2SimCC is adaptable to any 2Simulate target, it needs to have a generic interface to operate any kind and number of targets and models. Therefore 2Simulate targets communicate via its 2Simulate Communication Protocol (TSCP). TSCP is handled by a connection task (TSimConTask), which is based on TCP and uses a dynamic protocol layout (Fig. 3).

The 2SimCC has the capability to allow users to implement their own page layout for interaction to just control dedicated model values and parameters of interest and allows an operational service with a selected focus. This is based on a global data dictionary concept. 2SimCC has knowledge about all registered data signals inside all connected targets. Connecting a target with 2SimCC starts with a request to the local target data dictionary via TSCP. The targets then respond with their local dictionary. The data signals are shown in a data viewer selectively to have a numerical online presentation or used in several other menu tools, (e.g. graphical data scope). Individually designed notebook pages can be edited by the user with an integrated GUI-editor and show GUI-control-elements (numeric data, button, radio button, choice, slider, etc.) to allow bidirectional data access. Fig. 4 shows an example for a classical two mass spring damper model. All elements have the ability to connect to any data signal inside the global data dictionary. All data signals are referenced symbolically (i.e. with its fully detailed name path). Therefore, any revised version of the model will still correctly refer to the variables, unless their names are really modified as part of the modelling.


2Simulate Control Center - TEST
User Project Target Trim Data Extras Style Help
IC
General
Two Mass Spring Damper

Two Mass Spring Damper

Model Parameters

Mass1			Mass2		
m1	5.000	[-]	m2	1.000	[-]
k1	4.000	[-]	k2	2.000	[-]
d1	1.000	[-]	d2	3.000	[-]

Trim Results

x1	10.000	[-]
x2	30.000	[-]

External Force

F 40.000 [-]

Figure 4. Project Specific 2SimCC User Level Notebook Page for a Two Mass Spring Damper Example

III. Steady State Trim Analysis

A. Basics

A nonlinear dynamic system of n^th order can be written as a system of first-order differential equations by its vector differential equation and vector output equation[10]:

$$\dot{\boldsymbol{x}}(t) = \mathrm{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \boldsymbol{u}(t_0) = \boldsymbol{u}_0$$
$$\boldsymbol{y}(t) = \mathrm{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)),$$

with

- $\boldsymbol{x}$ state vector
- $\boldsymbol{u}$ input vector
- $\boldsymbol{y}$ output vector
- t time.

The stationary state of a dynamic system corresponds to the general idea of rest.[2,11] For a closer study of a stationary operating point (trim state) it is necessary to distinguish whether the system is in total rest or only in an acceleration-free condition. A simple spring-mass-damper system is at total rest when in a stationary state. That is not only its dynamic states are zero but also its kinematic states. In the case of systems of much higher complexity, such as an aircraft, this is only the case if it stays on the ground. In the air at least some of its kinematic states like the translational velocities are nonzero and take up a defined value, according to the specified trim state. Therefore it is reasonable to split the state vector into a kinematic portion $\boldsymbol{x_{kin}}$ and a dynamic portion $\boldsymbol{x_{dyn}}$[2]

$$\boldsymbol{x}(t) = \left[\boldsymbol{x_{dyn}}(t)^T, \boldsymbol{x_{kin}}(t)^T\right]^T,$$

with

$$\dot{\boldsymbol{x}}_{\boldsymbol{dyn}} = f_{dyn}\big(\boldsymbol{x_{dyn}}(t), \boldsymbol{x_{kin}}(t), \boldsymbol{u}\big)$$
$$\dot{\boldsymbol{x}}_{\boldsymbol{kin}} = f_{kin}\left(\boldsymbol{x_{dyn}}(t), \boldsymbol{x_{kin}}(t)\right) \quad .$$

The dynamic portion $\boldsymbol{x_{dyn}}$ includes the state variables, whose derivatives are dependent on the input vector $\boldsymbol{u}$, while the kinematic portion $\boldsymbol{x_{kin}}$ contains all other elements, i.e. whose derivatives are independent from $\boldsymbol{u}$. Generalizing the trim problem can be seen to establish a stationary condition of a time-invariant dynamic system as follows:

$$f\left(\left[\dot{\boldsymbol{x}}_{\boldsymbol{req}}^{\boldsymbol{T}}\ \boldsymbol{y}_{\boldsymbol{req}}^{\boldsymbol{T}}\right], \left[\boldsymbol{x}_{\boldsymbol{trim}}^{\boldsymbol{T}}\ \boldsymbol{u}_{\boldsymbol{trim}}^{\boldsymbol{T}}\right]\right) = 0$$

Hereby required values for the vectors $\dot{\boldsymbol{x}}_{\boldsymbol{req}}$ und $\boldsymbol{y_{req}}$ will be specified within the possible physical bounds. They are so called trim requirements. The elements of the state vector $\boldsymbol{x_{trim}}$ and input vector $\boldsymbol{u_{trim}}$, necessary to fulfil the trim requirements, are called trim variables. Generally this results in a nonlinear system of equations, which is not solvable analytically. Therefore a numerical iteration procedure will be used to solve it. The trim variables will be varied until all the trim requirements are fulfilled with a sufficient accuracy. The used Newton algorithm belongs to the class of gradient based iteration algorithms[12].

B. Trim procedure

It is important to understand that the procedure of trimming a generic dynamic system can be partitioned into two consecutive steps[13]:

1. Establishing the trim law.
2. Performing the trim calculation based on the trim law using actual values.

Establishing the trim law describes the selection of trim variables and trim requirements, no values will be assigned:

1.a To specify the trim requirements, a selection of elements of the derivative vector $\dot{\boldsymbol{x}}$ and the output vector $\boldsymbol{y}$ is needed.
1.b To specify the trim variables, a selection of elements of the state vector $\boldsymbol{x}$ and the input vector $\boldsymbol{u}$ is needed.

REMARK: The number of trim variables and trim requirements must match to get unique solutions to the trim problem!

To conduct the trim calculation based on the trim law for any specific trim state the following applies:

2.a The trim requirements need to be specified with nominal values.
2.b All elements of the state vector $\boldsymbol{x}$ and the input vector $\boldsymbol{u}$ shall be defined. If it is a trim variable, the value will be used as a start value for the iterative solution of the system of equations. Otherwise the value will be used as a fixed value to the trim calculation and is also the initial value for the subsequent simulation run.

Performing the trim calculation will then determine the trim variables. Furthermore it automatically determines all those elements of the derivative vector $\dot{\boldsymbol{x}}$ and the output vector $\boldsymbol{y}$, which are not subjected to any trim requirement.

IV. 2Simulate Implementation

The control and handling of any trim functionality is realized in 2SimCC. It is worth mentioning that besides the online trim feature, the primary topic of this paper, 2SimCC is also able to load predefined trim points, as discussed in section 1. This is obtainable by a standard file load (*Load Static Model Data*, Fig. 5). Static trim point files are stored and loaded in Mathworks .mat format and are generated in MATLAB. In this case it is up to the modeler using MATLAB to generate a steady state condition for the simulation model before storing these static trim point files.

The implementation of the online trim procedure includes the presentation of the trim problem, the operational functionality and the implementation of the trim algorithm itself.

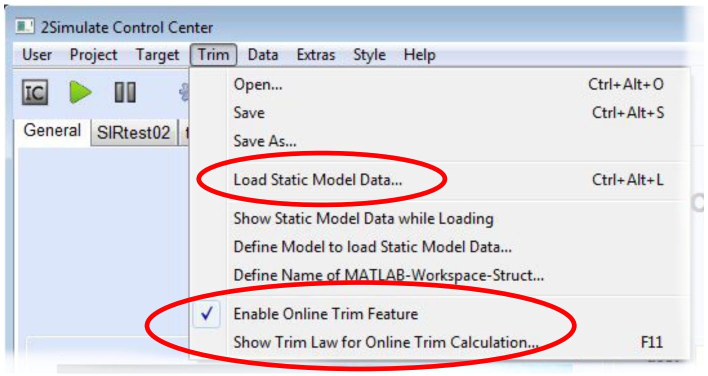

2Simulate Control Center
User
Project
Target
Trim
Data
Extras
Style
Help
Open... Ctrl+Alt+O
Save Ctrl+Alt+S
Save As...
Load Static Model Data... Ctrl+Alt+L
Show Static Model Data while Loading
Define Model to load Static Model Data...
Define Name of MATLAB-Workspace-Struct...
Enable Online Trim Feature
Show Trim Law for Online Trim Calculation... F11
General
SIRtest02

Figure 5. 2SimCC Trim Menu

A. Trim problem presentation and control

The implementation of the online trim functionality into 2Simulate allows operation on two different levels (Fig. 6). Firstly, a modeler is able to define and operate the trim law for even complex models based on the design of her/his simulation model. Secondly, a user is able to use the previously stored trim law to operate the simulator without detailed knowledge of the trim algorithm and settings. Defining and storing the trim law is implemented inside an editable trim overview menu, which itself is subdivided into a trim law menu and a trim control menu. Besides editing the trim overview menus, the modeler is able to fully manage and observe the trim calculation process.

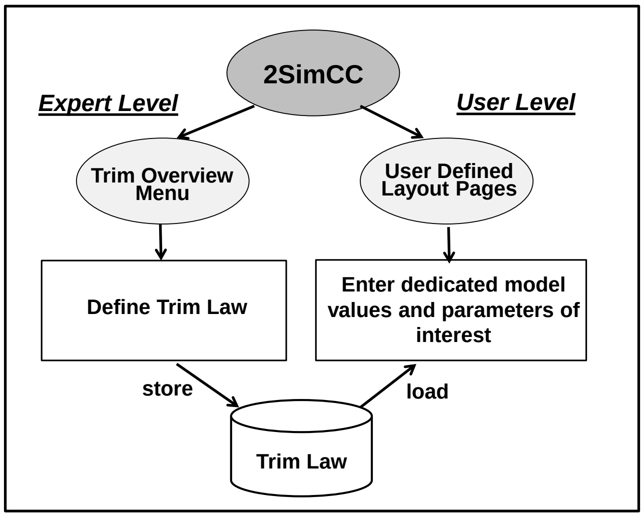

2SimCC
Expert Level
User Level
Trim Overview Menu
User Defined Layout Pages
Define Trim Law
Enter dedicated model values and parameters of interest
store
load
Trim Law

Figure 6. 2SimCC Trim Levels

The trim law is presented in a 4-quadrant SIDO-structure (States, Input, Derivatives and Output) inside the trim law menu (Fig.7). All the variables registered in the data dictionary, explained in section II-B, are shown here. The two quadrants on the left side allow the user to select the trim variables which are varied during the trim calculation, whereas the two quadrants on the right side allow the user to determine the trim requirements. The upper left quadrant presents the elements of the state vector underneath which the elements of the input vector are presented. A checkbox allows each variable to be selected as a trim variable. On the right side the upper quadrant presents the elements of the derivative vector and the lower presents those of the output vector. Presentation of the derivatives is performed by the name of the state variable followed by a trailing apostrophe. A checkbox allows each variable to be selected as a trim requirement. Information about the number of selected trim variables and trim requirements is shown underneath the quadrants. Selecting adequate start values for the trim variables generally significantly influences the trim calculation, either influencing the number of iterations or any successful result at all.

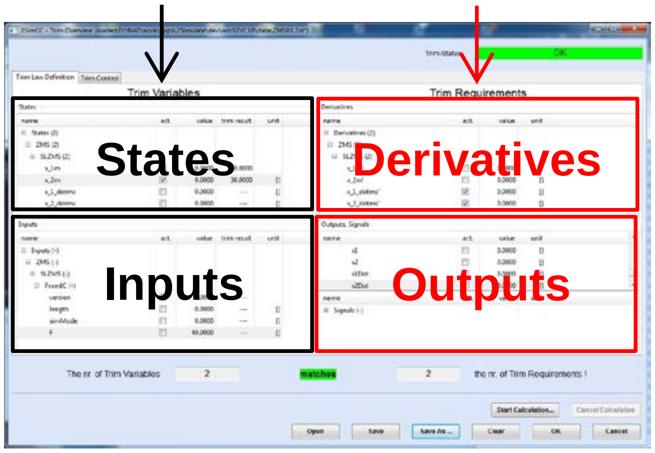


Figure 7. 2SimCC Trim Law Menu

The user level trim functionality is implemented within the user editable notebook pages, presented in section II-B (Fig. 4). The described data control functionality has been extended so that all GUI-control-elements can be connected in addition to any trim variable prior defined in the trim law menu. Activating a new trim calculation with its 2SimCC toolbar-icon allows an operational service with a selected focus without the expert knowledge of the modeler. The prior defined trim law is used internally and doesn't need to be opened in the trim overview menu. This is shown in the example in Fig. 13.

B. Trim operation

The operation of the trim calculation as depicted in Fig. 8 starts with a user event from 2SimCC, either from a toolbar icon inside the main menu or from the trim overview menu. Then all relevant trim data is generated as TSCP protocol and sent from 2SimCC's connection task to the targets connection task. Here the TSCP protocol is analyzed. This causes the setting of trim variables and activates the trim mode inside the model task.

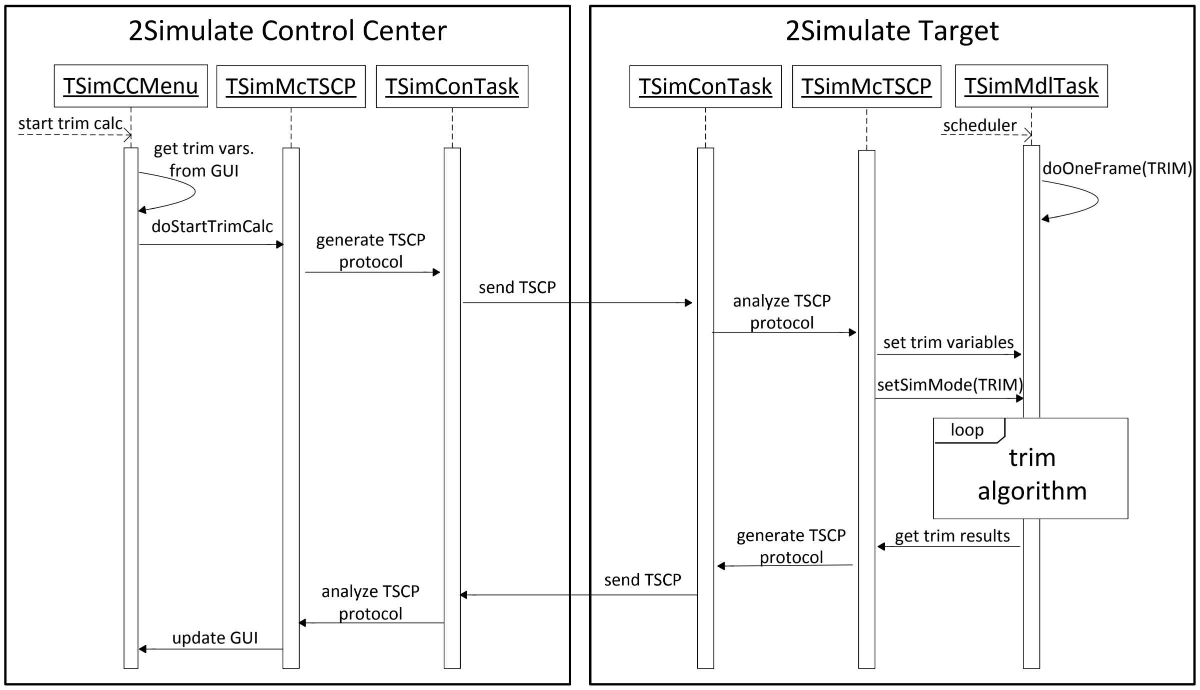


Figure 8. 2Simulate Trim Operation

C. Trim algorithm

The trim algorithm is implemented into 2Simulate's model control framework 2SimMC. 2Simulate uses a classical numerical Newton-Raphson iteration algorithm.[12] The main control parameters are the accuracy (EPS) and stop criteria (LIM) as well as the perturbation fraction for calculating the jacobian matrix. They are specified in the 2SimCC trim control menu. The accuracy is tested with a cost-function, giving the relative error of the trim result based on the standard least squares method. This is calculated for each two consecutive iterations. A general flow control of the trim algorithm is shown in Fig. 9.

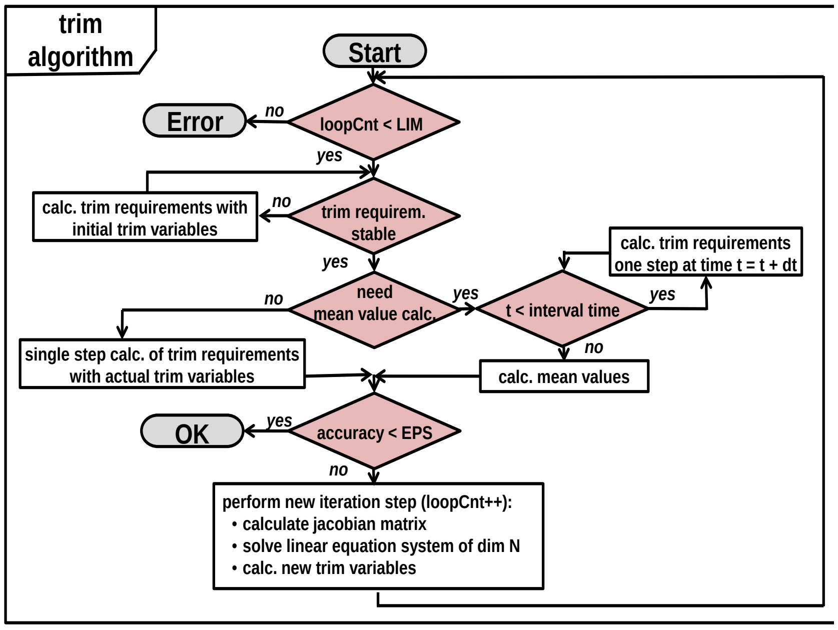

trim algorithm
Start
Error
no
loopCnt < LIM
yes
calc. trim requirements with initial trim variables
no
trim requirem. stable
yes
calc. trim requirements one step at time t = t + dt
no
need mean value calc.
yes
t < interval time
yes
no
single step calc. of trim requirements with actual trim variables
calc. mean values
OK
yes
accuracy < EPS
no
perform new iteration step (loopCnt++):
• calculate jacobian matrix
• solve linear equation system of dim N
• calc. new trim variables

Figure 9. Flow Control of Trim Algorithm

V. Application using the ACT/FHS simulation model

DLR operates the Active Flight Control/Flying Helicopter Simulator (ACT/FHS). The aircraft is a highly modified version of the EC135. For this reason, its performance and qualities do not reflect standard operational variants of the aircraft type. Vehicle configurational data and flight test and performance data have been used to develop the model operational in AVES. Parameter tuning was conducted to better reflect data obtained from flight test data[14]. The model features a rigid, four bladed, constant velocity main rotor, with Pitt-Peters dynamic inflow. A simple disk tail-rotor model is used along with a simple engine model. The helicopter model features additional elements, specific to testing conducted by researchers at DLR. This includes a slung-load model, used for flight testing of control systems to reduce pilot workload during hoist operations[15], and Control Equivalent Turbulence Input (CETI) models[16], for turbulence simulation. Both have been validated using flight test data collected from the ACT/FHS.

The trim law for the model is based on six trim variables and six trim requirements (Fig. 10). The trim requirements are the translational and rotational accelerations of the rigid body. The trim variables are the four controls plus the two Euler angles, theta and phi.

Trim Variables	Trim Requirements
pitchCmd	q’ = 0.0
rollCmd	p’ = 0.0
yawCmd	r’ = 0.0
collCmd	ukf’ = 0.0
phi	vkf’ = 0.0
theta	wkf’ = 0.0

Figure 10. EC135 ACT/FHS Trim Law

The simulation model is realized as a 2Simulate TSML model. It is subdivided into six submodels, each derived from TSimMcTsmlModelCtrl: actuator, body, engine, main rotor, tail rotor, and sensor. Fig. 11 shows code excerpts of the main class EC135 to illustrate the submodel initializations and data dictionary entries for the four control inputs, used as trim variables.

```
//====================
void EC135::initSubModels( void ) {
  m_pActuator = new EC135Actuator( "Actuator" );
  m_pEngine = new EC135Engine( "Engine" );
  m_pMainRotor = new EC135MainRotor( "MainRotor" );
  m_pTailRotor = new EC135TailRotor( "TailRotor" );
  m_pBody = new EC135Body( "Body" );
  m_pSensor = new EC135Sensor( "Sensor" );
  addModel( m_pActuator );
  addModel( m_pEngine );
  addModel( m_pMainRotor );
  addModel( m_pTailRotor );
  addModel( m_pBody );
  addModel( m_pSensor );
  ...
}
//====================
void EC135::initDataDict( void ) {
  addDataDictInputEntry( "clsPitCmd", "pitch cmd. (0=FW, 100=BA)", "-",
                         &aInData.clsPitCmd, MCTRL_DATATYP_real32, 1 );
  addDataDictInputEntry( "clsRolCmd", "roll cmd. (0=LH, 100=RH)" , "-",
                         &aInData.clsRolCmd, MCTRL_DATATYP_real32, 1 );
  addDataDictInputEntry( "clsYawCmd", "yaw cmd. (0=LH, 100=RH)"  , "-",
                         &aInData.clsYawCmd, MCTRL_DATATYP_real32, 1 );
  addDataDictInputEntry( "clsColCmd", "coll. cmd. (0=DN, 100=UP)", "-",
                         &aInData.clsColCmd, MCTRL_DATATYP_real32, 1  );
  ...
}
```

Figure 11. EC135 Class Code Excerpts of EC135 ACT/FHS Model

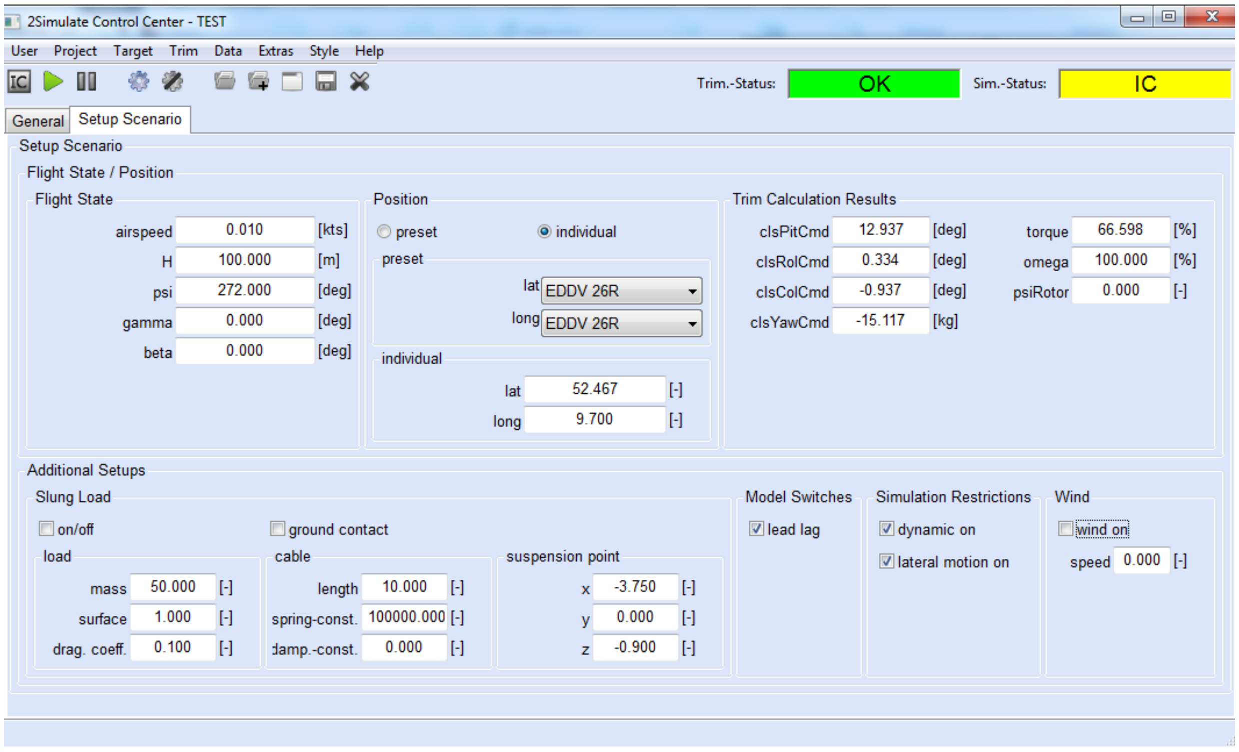
Fig. 12 shows excerpts from the code of the subclass EC135Body. It illustrates the data dictionary entries for the two trim variables theta and phi. Furthermore it shows the definition of the state variables and their corresponding derivatives, which are the trim requirements (i.e. the body-fixed translational velocity derivatives V_B_VEC’ and the rotational velocity derivatives P_B_VEC’) implemented as three-dimensional vectors. The trim requirements are presented in the derivatives quadrant of the 2SimCC trim law menu, whereas the trim variables are presented in the inputs quadrant shown in Fig. 7. The EC135Body code excerpts also show the relevant right hand side calculations of the two acceleration vectors.

A helicopter is a complex system with a large amount of cross coupling between control inputs and states. The system is highly nonlinear and changes significantly as a function of operating conditions. At the start of a simulation run the main rotor has a transient vibration caused by some algebraic loops, which complicates obtaining the accelerations of the rigid body. The main rotor requires a number of rotations to allow for the calculation of stable mean values for these accelerations.[1] 2SimCC permits the calculation of mean values for the trim requirement values. This will be configured in the 2SimCC trim control menu, defining an interval time for which the mean values will be calculated and a possible interval time multiplier. Here, the interval is set to one rotation and an interval time multiplier of four according to four rotations.

A corresponding user level scenario page has been edited for user level operation (Fig. 13). A scenario setup in the upper left area allows the definition of elementary flight state values: airspeed, height, heading, flight path angle and position. Specific model parameters (e.g. for the slung load model), which influence the model behavior and therefore the trim result, can be defined within additional setup areas at the bottom. The page also shows the trim results in the upper right area.

```
//=====================
void EC135Body::initDataDict( void ) {
  addDataDictInputEntry( "tht_IC", "tht_IC", "rad", &tht_IC, MCTRL_DATATYP_real, 1   );
  addDataDictInputEntry( "phi_IC", "phi_IC", "rad", &phi_IC, MCTRL_DATATYP_real, 1   );
  ...
}
//=====================
void EC135Body::defineStates( void ) {
  addState( V_B_VEC, V_B_DOT_VEC, "V_B_VEC", "translational velocities", "m/s" );
  addState( P_B_VEC, P_B_DOT_VEC, "P_B_VEC", "rotational velocities", "rad/s" );
  ...
}
//=====================
int EC135Body::doOneFrame( void ) {
  ...
 // ## translational accelerations
 V_B_DOT_VEC = F_TOT_VEC/MASS_DYN + BODY_EULER_M*G_VEC - P_B_VEC.crossProduct( V_B_VEC );
 // ## rotational accelerations
 P_B_DOT_VEC = I_INV_M.inverse() * ( M_TOT_VEC - P_B_VEC.crossProduct( I_DYN_M*P_B_VEC ) );
 ...
}
```

Figure 12. EC135Body Class Code Excerpts of EC135 ACT/FHS Model


2Simulate Control Center - TEST
User Project Target Trim Data Extras Style Help
General
Setup Scenario

Trim.-Status: OK
Sim.-Status: IC

Setup Scenario
Flight State / Position

Flight State
airspeed	0.010	[kts]
H	100.000	[m]
psi	272.000	[deg]
gamma	0.000	[deg]
beta	0.000	[deg]

Position
( ) preset (● individual)
preset: lat EDDV 26R; long EDDV 26R
individual: lat 52.467 [-]; long 9.700 [-]

Trim Calculation Results
clsPitCmd	12.937	[deg]	torque	66.598	[%]
clsRolCmd	0.334	[deg]	omega	100.000	[%]
clsColCmd	-0.937	[deg]	psiRotor	0.000	[-]
clsYawCmd	-15.117	[kg]			

Additional Setups

Slung Load
☐ on/off
☐ ground contact

load			cable			suspension point		
mass	50.000	[-]	length	10.000	[-]	x	-3.750	[-]
surface	1.000	[-]	spring-const.	100000.000	[-]	y	0.000	[-]
drag. coeff.	0.100	[-]	damp.-const.	0.000	[-]	z	-0.900	[-]

Model Switches
☑ lead lag

Simulation Restrictions
☑ dynamic on
☑ lateral motion on

Wind
☐ wind on
speed 0.000 [-]

Figure 13. EC135 ACT/FHS Simulation Setup Scenario Notebook Page

The elapsed time for a trim calculation run is two seconds on a COTS-PC. In case of a hover scenario the simulation has a stable behavior on a desktop-PC using a joystick as control inputs for over 15 seconds (assuming no pilot commands). This is due to the ideal, noiseless input signals from a joystick. Running the simulation at the AVES full flight simulator, using its active control load system, the hover state is stable for about five seconds due to the noise of the input signals. Results also depend on the quality of the trimmed position of the control load system. Positioning the control load system according to the previously calculated trim state is done with an accuracy of 0.5% due to the hardware possibilities and therefore has some minor inaccuracy.

VI. Conclusion

An online trim procedure has been successfully implemented within the AVES simulation infrastructure using the 2Simulate simulation framework. This is the first implementation of an online trimming function within 2Simulate, and in addition replaces a less flexible use of the method for the ACT/FHS Helicopter Simulator model. In former implementations the procedure was hard coded inside the simulation model and therefore was not usable for other models. The new software was the result of more progressive user demands. The challenge for the new implementation was the combination of a flexible user interface, where users have different knowledge levels as well as the underlying implementation of the trim algorithm. It allows different user interfaces within a GUI, whilst fulfilling strong mathematical constraints. The trim procedure is designed to operate general simulations. It is able to work with any dynamic simulation model within the supported model languages. There is still some potential for optimization of the classical iteration algorithm, as well as possible implementation of new approaches using the same tool design. Further work will also include an enhanced presentation of the iteration process and possible linear dependencies causing a singular jacobian matrix. These measures will further increase performance and flexibility. The presented online trimming feature is planned to be used for a number of projects at AVES. A test version is currently in use for the trim law of the AVES A320 simulation model. This model requires higher complexity in the trim routine, which is now provided by the online trim capabilities within 2Simulate.

References

[1]McFarland, R.E., “Trimming an Aircraft Model for Flight Simulation,” *NASA Technical Memorandum*, Ames Research Center, Moffett Field, California, 1987

[2]De Marco, A., Duke, E., Berndt, J., “A General Solution to the Aircraft Trim Problem,” *AIAA Modeling and Simulation Technologies Conference*, Hilton Head, SC, 2007

[3]Elgersma, M. R., Morton, B. G., “Nonlinear Six-Degree-of-Freedom Aircraft Trim,” *Journal of Guidance, Control and Dynamics*, Vol.23, No.2, 2000

[4]Tupy J., Zelinka I., “Search for equilibrium state flight,” *Innovations and Advances in Computer Sciences and Engineering, edited by Sobh T.*, Springer, Dordrecht, 2010

[5]Enns, R., Si, J., “Helicopter trimming and tracking control using direct neural dynamic programming,”
IEEE Transactions on Neural Networks, Vol.14, Issue 4, 2003

[6]Duda, H., Gerlach, T., Advani S., Potter M. “Design of the DLR AVES Research Flight Simulator,” *AIAA Modeling and Simulation Technologies Conference*, Boston, MA, 2013

[7]Gerlach, T., Durak, U., “AVES SDK: Bridging the Gap between Simulator and Flight Systems Designer,” *AIAA Modeling and Simulation Technologies Conference*, Dallas, TX, 2015

[8]Gerlach, T., Durak, U., Gotschlich, J., “Model Integration Workflow for Keeping Models up to Date in a Research Simulator,” *Proceedings of the 4th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, SCITEPRESS, Wien, Austria, 2014, pp. 125-132

[9]Gotschlich, J., Gerlach, T., Durak, U. “2Simulate: A Distributed Real-Time Simulation Framework,” *ASIM STS/GMMS Workshop 2014*, Reutlingen, Germany, 2014

[10]Stevens, B., Lewis, F., *Aircraft Control and Simulation*, 2nd ed., Wiley, 2003

[11]Sgarioto, D. E., “Steady State Trim and Open Loop Stability Analysis for the REMUS Autonomous Underwater Vehicle,” *IEEE International Conference on Control and Automation*, Christchurch, New Zealand, 2009

[12]Stör, J. *Einführung in die Numerische Mathematik*, Springer, 1979

[13]Buchholz, J., “Regelungstechnik und Flugregler,“ *Vorlesungsmanuskript*, Hochschule Bremen, 2016, URL:http://buchholz.hs-bremen.de/rtfr/skript/skript10.pdf [cited 27 November 2017]

[14]Hamers, M., von Grünhagen, W., “Nonlinear Helicopter Model Validation Applied to Realtime Simulations,“ *53rd American Helicopter Society Annual Forum*, Virginia Beach, Virginia, 1997.

[15]Nonnenmacher, D., Jones, M., “Handling Qualities Evaluation of an Automatic Slung Load Stabilisation System for the ACT/FHS,” *CEAS Aeronautical Journal*, Vol. 7, 4, pp 587-606, 2016.

[16]Lehmann, P., Jones, M., Höfinger, M., “Impact of turbulence and degraded visual environment on pilot workload,” *CEAS Aeronautical Journal*, Vol. 8, 3, pp 413-428, 2017.