NASA/TP–20220000355/Vol 3

NDARC
NASA Design and Analysis of Rotorcraft

Input

Wayne Johnson
Ames Research Center
Moffett Field, California

January 2022

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TP–20220000355/Vol 3

NDARC
NASA Design and Analysis of Rotorcraft

Input

Wayne Johnson
Ames Research Center
Moffett Field, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA 94035-1000

January 2022

NDARC

NASA Design and Analysis of Rotorcraft

Input

Release 1.16
January 2022

Wayne Johnson
NASA Ames Research Center, Moffett Field, CA

Contents

Chapter 1

Data Structures and Input

1–1 Overview

The NDARC code performs design and analysis tasks. The design task involves sizing the rotorcraft to satisfy specified design conditions and missions. The analysis tasks can include off-design mission performance analysis, flight performance calculation for point operating conditions, and generation of subsystem or component performance maps. Figure 1-1 illustrates the tasks. The principal tasks (sizing, mission analysis, flight performance analysis) are shown in the figure as boxes with heavy borders. Heavy arrows show control of subordinate tasks.

The aircraft description (figure 1-1) consists of all the information, input and derived, that defines the aircraft. The aircraft consists of a set of components, including fuselage, rotors, wings, tails, and propulsion. This information can be the result of the sizing task; can come entirely from input, for a fixed model; or can come from the sizing task in a previous case or previous job. The aircraft description information is available to all tasks and all solutions (indicated by light arrows).

The sizing task determines the dimensions, power, and weight of a rotorcraft that can perform a specified set of design conditions and missions. The aircraft size is characterized by parameters such as design gross weight, weight empty, rotor radius, and engine power available. The relations between dimensions, power, and weight generally require an iterative solution. From the design flight conditions and missions, the task can determine the total engine power or the rotor radius (or both power and radius can be fixed), as well as the design gross weight, maximum takeoff weight, drive system torque limit, and fuel tank capacity. For each propulsion group, the engine power or the rotor radius can be sized.

Missions are defined for the sizing task, and for the mission performance analysis. A mission consists of a number of mission segments, for which time, distance, and fuel burn are evaluated. For the sizing task, certain missions are designated to be used for design gross weight calculations; for transmission sizing; and for fuel tank sizing. The mission parameters include mission takeoff gross weight and useful load. For specified takeoff fuel weight with adjustable segments, the mission time or distance is adjusted so the fuel required for the mission (burned plus reserve) equals the takeoff fuel weight. The mission iteration is on fuel weight or energy.

Flight conditions are specified for the sizing task, and for the flight performance analysis. For the sizing task, certain flight conditions are designated to be used for design gross weight calculations; for transmission sizing; for maximum takeoff weight calculations; and for antitorque or auxiliary thrust rotor sizing. The flight condition parameters include gross weight and useful load.

For flight conditions and mission takeoff, the gross weight can be maximized, such that the power required equals the power available.

A flight state is defined for each mission segment and each flight condition. The aircraft performance can be analyzed for the specified state, or a maximum effort performance can be identified. The maximum effort is specified in terms of a quantity such as best endurance or best range, and a variable such as speed, rate of climb, or altitude. The aircraft must be trimmed, by solving for the controls and motion that produce equilibrium in the specified flight state. Different trim solution definitions are required for various flight states. Evaluating the rotor hub forces may require solution of the blade flap equations of motion.

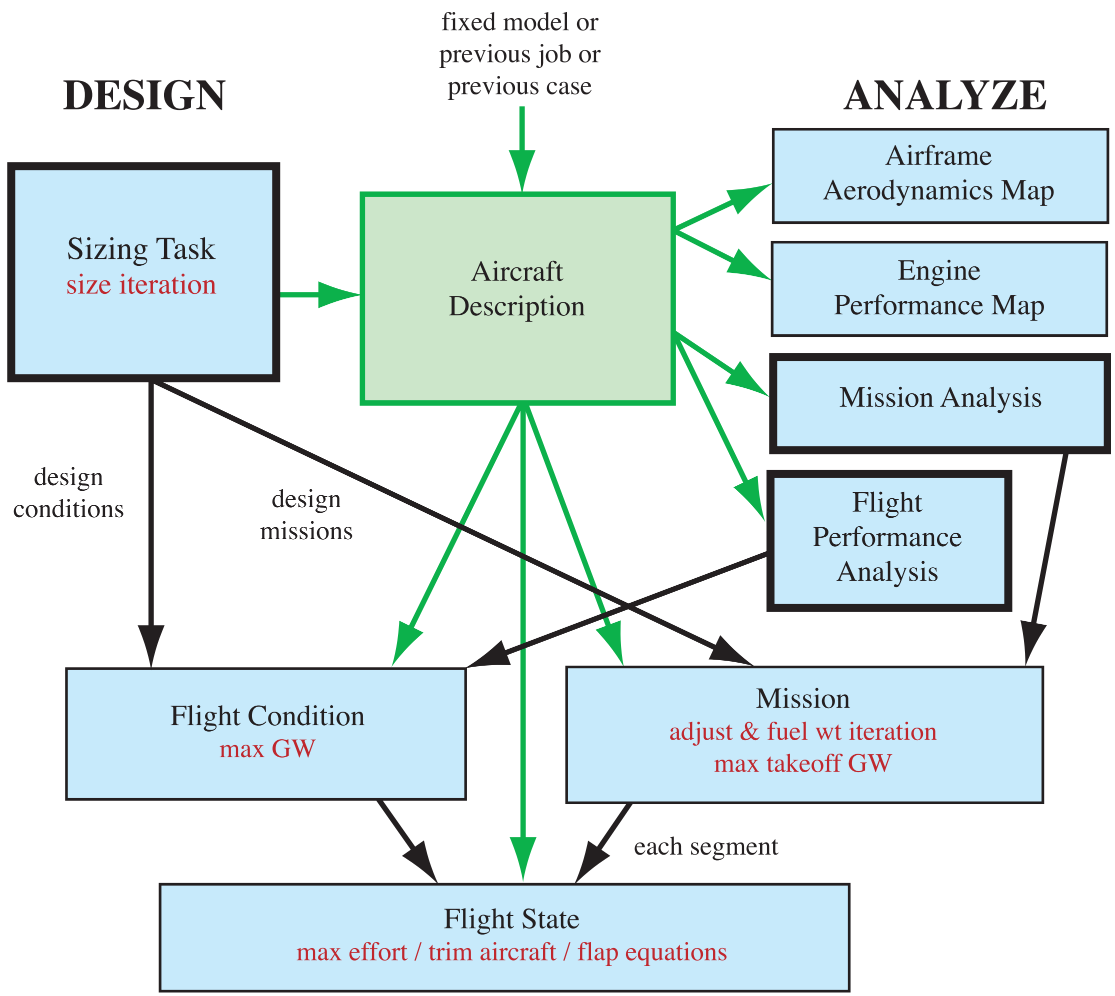

Figure 1-1 Outline of NDARC tasks.

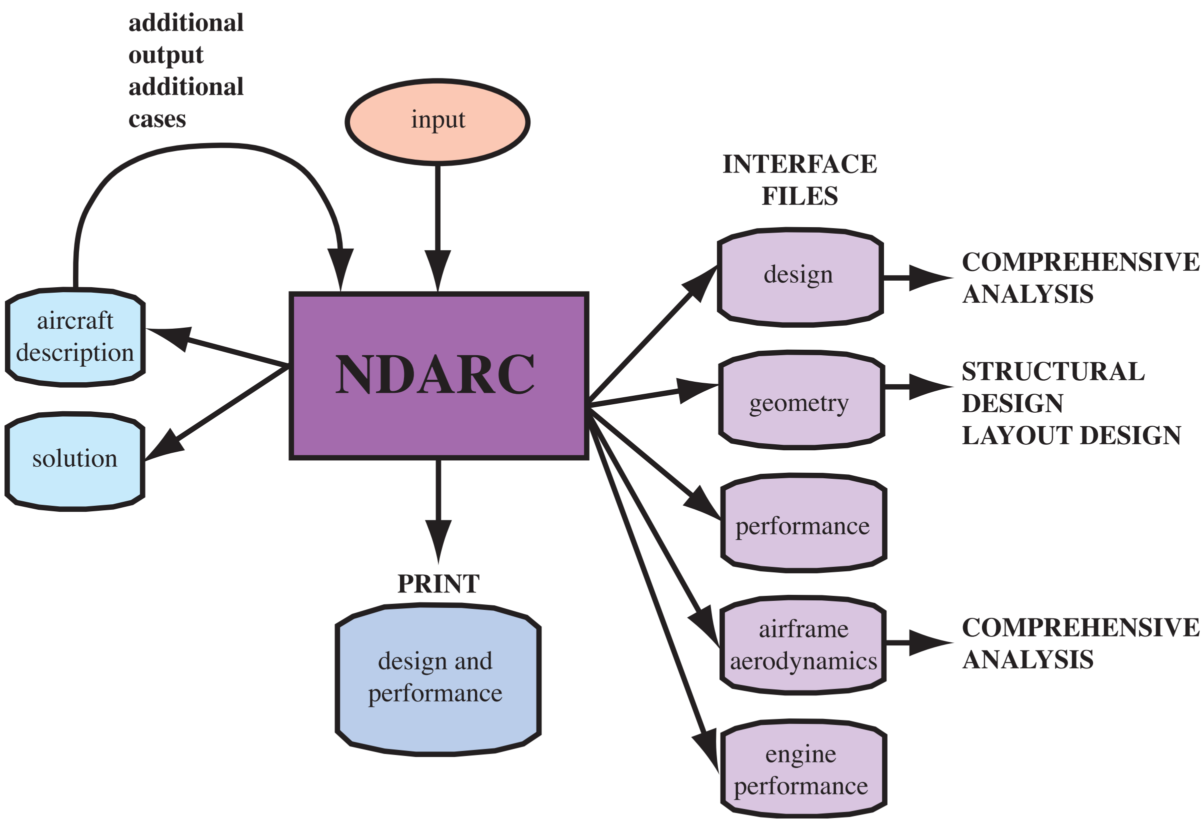

Figure 1-2 NDARC Interfaces.

```
&JOB INIT_input=0,INIT_data=0,&END
&DEFN action='ident',created='time-date',title='standard input',&END
!##########################################################################
&DEFN action='open file',file='engine.list',&END
&DEFN action='open file',file='helicopter.list',&END
!==========================================================================
&DEFN quant='Cases',&END
&VALUE title='Helicopter',TASK_size=0,TASK_mission=1,TASK_perf=1,&END
&DEFN quant='Size',&END
&VALUE nFltCond=0,nMission=0,&END
!==========================================================================
&DEFN quant='OffDesign',&END
&VALUE title='mission analysis',nMission=1,&END
&DEFN quant='OffMission',&END
&VALUE
        (one mission, mission segment parameters as arrays)
&END
!==========================================================================
&DEFN quant='Performance',&END
&VALUE title='performance analysis',nFltCond=2,&END
&DEFN quant='PerfCondition',&END
&VALUE
        (one condition)
&END
&DEFN quant='PerfCondition',&END
&VALUE
        (one condition)
&END
!==========================================================================
&DEFN action='endofcase',&END
!##########################################################################
&DEFN action='endofjob',&END
```

Figure 1-3a Illustration of NDARC input (primary input).

```
&DEFN action='ident',created='time-date',title='Helicopter',&END
!########################################################################
! default helicopter
&DEFN action='configuration',&END
&VALUE config='helicopter',rotate=1,&END
!========================================================================
&DEFN quant='Cases',&END
&VALUE title='Helicopter',FILE_design='helicopter.design',&END
&DEFN quant='Size',&END
&VALUE
    title='Helicopter',
    SIZE_perf='none',SET_rotor='radius+Vtip+sigma','radius+Vtip+sigma',
    FIX_DGW=1,SET_tank='input',SET_SDGW='input',SET_WMTO='input',
&END
&DEFN quant='Solution',&END
&VALUE &END
!========================================================================
&DEFN quant='Aircraft',&END
&VALUE (Aircraft parameters) &END
&DEFN quant='Geometry',&END
&VALUE (geometry) &END
&DEFN quant='Rotor 1',&END
&VALUE (Rotor 1 parameters) &END
!========================================================================
        (other parameters in other structures)
!========================================================================
&DEFN quant='TechFactors',&END
&VALUE (technology factors) &END
!########################################################################
&DEFN action='endoffile',&END
```

Figure 1-3b Illustration of NDARC input (secondary input file).

1–2 NDARC Input and Output

Figure 1-2 illustrates the input and output environment of NDARC. Table 1-1 lists the possible input and output files. A job reads input from one or more files. The primary input is obtained from standard input (perhaps redirected to a file). The primary input can direct the code to read other files, identified by file name or logical name. The input data are read in namelist format. Unit numbers are part of the job input. Output file names are part of the case input. Input files names are defined in the input itself.

Table 1-1. Input and output files.

	file logical name	unit number (and default)
INPUT		
Primary Input	standard input	nuin = 5
Secondary Input File	FILE	nufile = 40
Aircraft Description	FILE	nufile = 40
Solution	FILE	nufile = 40
OUTPUT		
Output	standard output	nuout = 6
Design	DESIGNn	nudesign = 41
Performance	PERFn	nuperf = 42
Airframe Aerodynamics	AEROn	nuaero = 43
Engine Performance	ENGINEn	nuengine = 44
Geometry	GEOMETRYn	nugeom = 45
Aircraft Description	AIRCRAFTn	nuacd = 46
Solution	SOLUTIONn	nusoln = 47
Sketch	SKETCHn	nusketch = 48
Errors	ERRORn	nuerror = 49

1-2.1 Input

Figure 1-3 illustrates NDARC input. The primary input starts with a JOB namelist, then DEFN namelists are read to define the action and contents of the subsequent information. The job parameters include initialization control, error action, and input/output unit numbers. Job parameters can be read during case input using QUANT='Job'. The initialization takes place before case input, so changed initialization parameters in QUANT='Job' input take effect for the next case. The DEFN namelist has the following parameters.

a) ACTION: character string (length = 32; case independent).

b) QUANT: character string (length = 32, case independent); corresponds to data structure in input; string includes structure number (1 or next condition/mission if absent).

c) SOURCE: integer; for copy action.

d) FILE: file name or logical name (length = 256).

e) CREATED: character string of creation time and date (length = 20).

f) TITLE: character string of title identifying input file (length = 80).

g) VERSION: code version number as character string (length = 6).

h) MODIFICATION: character string of code modification (length = 32).

Table 1-2 describes the options for the ACTION variable in the DEFN namelist. The code searches for the keyword in the ACTION character string. A solution file (text or binary) can be written by an NDARC job and then read by a subsequent job, restoring the solution to the state that existed when the file was created. Then additional output and additional cases can be obtained. An aircraft description file can be written by an NDARC job and then read by a subsequent job, restoring the aircraft model (but not the solution). A secondary input file has DEFN namelists to define action and contents. When ACTION='end' (or EOF) is encountered in a secondary input file, the file is closed and the code returns to primary input.

A DEFN namelist with ACTION='ident' identifies the file; probably there is only one identification per file, and only the last occurrence is stored. The identification consists of the CREATED, TITLE, VERSION, MODIFICATION variables. CREATED and TITLE are written when a file is created by NDARC, and read and stored for each input file. If present, VERSION and MODIFICATION are compared with the version and modification of the code, and input continues only if they match.

The parameter QUANT identifies the data structure to be read (namelist format), initialized, or copied. Table 1-3 describes the options. The input corresponds to the data structures of the analysis. The QUANT string includes the structure number; if absent, the number is 1, or the next condition or mission. Note that each mission, with the mission segment parameters as arrays, is input with QUANT='SizeMission' or QUANT='OffMission'; and each condition is input with QUANT='SizeCondition' or QUANT='PerfCondition'.

A case inherits input for flight conditions and missions from the previous case if INIT_input = last-case-input (default). A DEFN namelist with ACTION='delete' deletes this input as specified by QUANT='SizeCondition n', QUANT='SizeMission n', QUANT='OffMission n', or QUANT='PerfCondition n'. ACTION='delete all' deletes all (ignore structure number); ACTION='delete one' deletes structure n (all if number absent); ACTION='delete last' deletes structure n and subsequent structures (all if number absent).

For ACTION='nosize', input variables in the Size structure are set for no size iteration: SIZE_perf='none', SIZE_engine='none', SIZE_jet='none', SIZE_charge='none', SET_rotor='radius+Vtip+sigma', SET_wing='area+span', FIX_DGW=1, SET_tank='input', SET_limit_ds='input', SET_SDGW='input', SET_WMTO='input'.

Table 1-2. ACTION options.

ACTION	keyword	QUANT	function
Primary Input Only			
blank	—	blank	open and read secondary input file, name = FILE
'open file'	file, open		open and read secondary input file, name = FILE
'load aircraft'	aircraft, desc		load aircraft description file, name = FILE
'read solution'	solution	'text'	read complete solution file, name = FILE (text)
'read solution'	solution	not 'text'	read complete solution file, name = FILE (binary)
'end of case'	end+case		stop case input, execute case
'end of job'	end+job, quit		stop job input, execute case, exit code
Primary or Secondary Input			
blank	—	'structure'	read VALUE namelist
'read namelist'	list	'structure'	read VALUE namelist
'copy input'	copy	'structure'	copy input from source (same structure), SOURCE=SRCnumber
'initialize'	init	'structure'	set structure variables to default values
'delete all'	del+all	'structure'	delete all conditions or missions
'delete one'	del+one	'structure'	delete one condition or mission
'delete last'	del+last	'structure'	delete last conditions or missions
'configuration'	config		set input based on aircraft configuration
'nosize'	nosize		set input for no size iteration
'identification'	ident		identify file
'end'	end (or EOF)		Secondary: close file, return to primary input
'end'	end (or EOF)		Primary: same as ACTION='endofjob'

Table 1-3. QUANT options.

QUANT	data structures read	maximum n
'Job'	Job	
'Cases'	Cases	
'Size'	SizeParam	
'SizeCondition n'	one FltCond+FltState	nFltCond
'SizeMission n'	one MissParam, MissSeg+FltState as array	nMission
'OffDesign'	OffParam	
'OffMission n'	one MissParam, MissSeg+FltState as array	nMission
'Performance'	PerfParam	
'PerfCondition n'	one FltCond+FltState	nFltCond
'MapEngine'	MapEngine	
'MapAero'	MapAero	
'Solution'	Solution	
'Cost'	Cost	
'Emissions'	Emissions	
'Aircraft'	Aircraft	
'Systems'	Systems, WFltCont, WDeIce	
'Fuselage'	Fuselage, AFuse, WFuse	
'LandingGear'	LandingGear, AGear, WGear	
'Rotor n'	Rotor, PRotorInd, PRotorPro, PRotorTab, IRotor, DRotor, WRotor	nRotor
'Wing n'	Wing, AWing, WWing, WWingTR	nWing
'Tail n'	Tail, ATail, WTail	nTail
'FuelTank n'	FuelTank, WTank	nTank
'Propulsion n'	Propulsion, WDrive	nPropulsion
'EngineGroup n'	EngineGroup, DEngSys, WEngSys	nEngineGroup
'JetGroup n'	JetGroup, DJetSys, WJetSys	nJetGroup
'ChargeGroup n'	ChargeGroup, DChrgSys, WChrgSys	nChargeGroup
'EngineModel n'	EngineModel	nEngineModel
'EngineParamN n'	EngineParamN	nEngineParamN
'EngineTable n'	EngineTable	nEngineTable
'RecipModel n'	RecipModel	nRecipModel
'CompressorModel n'	CompressorModel	nCompressorModel
'MotorModel n'	MotorModel	nMotorModel
'JetModel n'	JetModel	nJetModel
'FuelCellModel n'	FuelCellModel	nFuelCellModel
'SolarCellModel n'	SolarCellModel	nSolarCellModel
'BatteryModel n'	BatteryModel	nBatteryModel
'TechFactors'	all TECH_xxx	
'Geometry'	all Location	

1-2.2 Formats

Namelist input has the following format (see also figure 1-3).

```
&DEFN action='IDENT',created='time-date',title='xxx',version='n.n',modification='xxx',&END
&DEFN quant='STRUCTURE n',&END
&VALUE param=value,&END
&DEFN action='NAMELIST',quant='STRUCTURE n',&END
&VALUE param=value,&END
&DEFN action='COPY',quant='STRUCTURE n',source=#,&END
```

An aircraft description file is written in a separate file by NDARC, from theDesign(kcase):

```
&DEFN action='IDENT',created='time-date',title='xxx',version='n.n',modification='xxx',&END
&VALUE_ADIMEN nrotor=m,nwing=m,ntail=m,ntank=m,npropulsion=m,nenginegroup=m,njetgroup=m,nchargegroup=m,
     nenginemodel=m,nengineparamn=m,nenginetable=m,nrecipmodel=m,ncompressormodel=m,nmotormodel=m,njetmodel=m,
     nfuelcellmodel=m,nsolarcellmodel=m,nbatterymodel=m,&END
&VALUE theStructure%xxx,&END
&VALUE theStructure%xxx,&END
&VALUE theStructure%xxx,&END
```

This aircraft description file is read by identifying it in the primary input:

```
&DEFN action='AIRCRAFT',file='aircraft.acd',&END
```

A solution file is written in a separate file by NDARC, from theDesign(kcase), in binary or text format:

```
&DEFN action='IDENT',created='time-date',title='xxx',version='n.n',modification='xxx',&END
&VALUE_ADIMEN nrotor=m,nwing=m,ntail=m,ntank=m,npropulsion=m,nenginegroup=m,njetgroup=m,nchargegroup=m,
     nenginemodel=m,nengineparamn=m,nenginetable=m,nrecipmodel=m,ncompressormodel=m,nmotormodel=m,njetmodel=m,
     nfuelcellmodel=m,nsolarcellmodel=m,nbatterymodel=m,&END
&VALUE_SDIMEN nsizecond=m,nsizemiss=m,nperfcond=m,noffmiss=m,&END
&VALUE theStructure%xxx,&END
&VALUE theStructure%xxx,&END
&VALUE theStructure%xxx,&END
```

This solution file is read by identifying it in the primary input, with QUANT identifying the file as text or binary:

```
&DEFN action='SOLUTION,quant='TEXT',file='aircraft.soln'&END
```

1-2.3 Conventions

Each flight condition (FltCond and FltState variables) is input in a separate SizeCondition or PerfCondition namelist.

Each mission (MissParam, MissSeg, and FltState variables) is input in a separate SizeMission or OffMission namelist. All mission segments are defined in this namelist, so MissSeg and FltState variables are arrays. Each variable gets one more dimension, with the first array index always segment number.

Geometry input includes Location variables, which are read as elements of the data structure (for example, loc_rotor%SL).

Variables can appear in more than one namelist. Specifically there are separate namelists for all technology factors (all TECH_xxx variables), and all geometry (all Location variables), with corresponding options for output. A variable that is a scalar in the Rotor, Wing, Tail, Propulsion, EngineGroup, JetGroup, or ChargeGroup input becomes an array in the TechFactors or Geometry input. Note that it is the Location variable that is the array (for example, loc_rotor(1)%SL).

Case is not important in character string input. Character string input consists of keywords; the code searches for the keywords in the string.

Default values are specified in the dictionary (blank implies a default of zero); all elements of arrays have the same default value.

Tasks, aircraft, and components have title variables. There are also notes variables (long character string) to record information about the input.

1–3 Software Tool

All information about data structures is contained in a dictionary file. This information includes the parameter name, dimension, type, default value, description, identification as input, and formats for write of the parameter. A software tool was created to manage the data, including construction of the module of data structures. The software tool reads this dictionary file and creates subroutines for the input process: namelist read, copy, print of input, initialization, set to default. This software tool is a program that manipulates character strings, to produce compilable module and subroutines for NDARC.

1–4 Data Structures

Table 1-4 outlines the data structures used for NDARC. The following chapters describe the contents of each structure. Note that a ”+” sign in the column between the type and description identifies input variables. Input variables can be changed by the analysis, so may not be the same at the end of a case as at the beginning. All variables, input and other, are initialized to zero or blank. If default values exist (only for input variables), they supersede that initialization.

Table 1-4. NDARC data structures.

Design	Fuselage	FuelTank(ntankmax)	FltState(nfltmax)
Cases	[Location]loc_fuselage	[Location]loc_auxtank(nauxtankmax)	FltAircraft
Size	AFuse	Weight	FltFuse
SizeParam	Weight	WTank	FltGear
FltCond(nfltmax)	WFuse	Propulsion(npropmax)	FltRotor(nrotormax)
FltState(nfltmax)	LandingGear	Weight	FltWing(nwingmax)
Mission(nmissmax)	[Location]loc_gear	WDrive	FltTail(ntailmax)
MissParam	AGear	EngineGroup(nengmax)	FltTank(ntankmax)
MissSeg(nsegmax)	Weight	[Location]loc_engine	FltProp(npropmax)
FltState(nsegmax)	WGear	DEngSys	FltEngn(nengmax)
OffDesign	Rotor(nrotormax)	Weight	FltJet(njetmax)
OffParam	[Location]loc_rotor	WEngSys	FltChrg(nchrgmax)
Mission(nmissmax)	[Location]loc_pylon	JetGroup(njetmax)	
MissParam	[Location]loc_pivot	[Location]loc_jet	
MissSeg(nsegmax)	[Location]loc_nac	DJetSys	
FltState(nsegmax)	PRotorInd	Weight	
Performance	PRotorPro	WJetSys	
PerfParam	PRotorTab	ChargeGroup(nchrgmax)	
FltCond(nfltmax)	IRotor	[Location]loc_charger	
FltState(nfltmax)	DRotor	DChrgSys	
MapEngine	Weight	Weight	
MapAero	WRotor	WChrgSys	
Solution	Wing(nwingmax)	EngineModel(nengmax)	
Cost	[Location]loc_wing	EngineParamN(nengpmax)	
Emissions	AWing	EngineTable(nengmax)	
Aircraft	Weight	RecipModel(nengmax)	
[Location]loc_cg	WWing	CompressorModel(nengmax)	
Weight	WWingTR	MotorModel(nengmax)	
XAircraft	Tail(ntailmax)	JetModel(njetmax)	
Systems	[Location]loc_tail	FuelCellModel(nchrgmax)	
Weight	ATail	SolarCellModel(nchrgmax)	
WFltCont	Weight	BatteryModel(ntankmax)	
WDeIce	WTail		

Chapter 2

Input Based on Configuration

The rotorcraft configuration is identified by the variable config in the QUANT='Aircraft' input. With ACTION='configuration', the analysis defines a number of input parameters in order to facilitate modelling of conventional configurations. The input required to execute ACTION='configuration' is:

```
&DEFN action='configuration',&END
&VALUE config='aaaa',nRotor=#,rotate=#,#,overlap_tandem=#,#,ang_multicopter=#,#,&END
```

The VALUE namelist contains only the parameters Aircraft%config (rotorcraft configuration), Aircraft%nRotor (number of rotors, only for multicopter), Rotor%rotate (direction of rotation, each rotor), Rotor%overlap_tandem (each rotor, only for tandem helicopter), and Rotor%ang_multicopter (each rotor, only for multicopter). The convention is that the first rotor is the main rotor for the helicopter or compound configuration; the front rotor for the tandem configuration; the right rotor for the tiltrotor configuration. This capability has been implemented for rotorcraft, helicopter, tandem, coaxial, tiltrotor, compound, multicopter, and airplane configurations. There is common input for all configurations, and special input for each except the rotorcraft. The analysis creates the following input, through information at the end of the NDARC structures file. Note that default values are defined for all input quantities.

2–1 All Configurations

a) Components: nRotor=2 (except multicopter), nWing=0, nTail=2; nPropulsion=1, nEngineGroup=1, nEngineModel=1, nJetGroup=0, nChargeGroup=0

b) Aircraft
Aircraft controls: ncontrol=7, IDENT_control='coll','latcyc','lngcyc','pedal','tailinc','elevator','rudder'
Control states: nstate_control=1
Trim states: nstate_trim=10, selected by FltAircraft%STATE_trim=IDENT_trim; compound state not active

	IDENT_trim	mtrim	trim_quant	trim_var
6-variable	'free'	6	'force x','force y','force z','moment x','moment y','moment z'	'coll','latcyc','lngcyc','pedal','pitch','roll'
longitudinal	'long'	4	'force x','force z','moment y','moment z'	'coll','lngcyc','pitch','pedal'
symmetric 3-variable	'symm'	3	'force x','force z','moment y'	'coll','lngcyc','pitch'
weight and drag	'force'	2	'force x','force z'	'coll','pitch'
hover thrust and torque	'hover'	2	'force z','moment z'	'coll','pedal'
hover thrust	'thrust'	1	'force z'	'coll'
hover rotor C_T/σ	'rotor'	1	'CTs rotor 1'	'coll'
wind tunnel	'windtunnel'	3	'CTs rotor 1','betac 1','betas 1'	'coll','latcyc','lngcyc'
full power	'power'	1	'P margin 1'	'coll'
ground run	'ground'	1	'force x'	'coll'
compound	'comp'	6	'force x','force y','force z','moment x','moment y','moment z'	'coll','latcyc','lngcyc','pedal','prop','roll'

c) Systems: MODEL_FWfc=0, MODEL_CVfc=0 (no fixed wing flight controls, no conversion controls)

d) Landing Gear: KIND_LG=0 (fixed gear), Wgear%nLG=3

e) Fuel Tank: place=1 (internal tank), Mauxtanksize=1, WTank%ntank_int=1, WTank%nplumb=2

f) Rotor
First rotor is primary: kPropulsion=1, KIND_xmsn=1
Second and other rotors are dependent: kPropulsion=1, KIND_xmsn=0, INPUT_gear=1 (input quantity is tip speed)
Configuration: direction='main'
Drag: SET_aeroaxes=1 (helicopter), Idrag=0. (not tilt); DRotor%SET_Dspin=1, DRotor%DoQ_spin=0. (no spinner drag)
Weight: WRotor%MODEL_config=1 (rotor), WRotor%KIND_rotor=2 (not tilting)
Control:
  INPUT_coll=0, INPUT_cyclic=0, INPUT_incid=0, INPUT_cant=0, INPUT_diam=0 (no connection to aircraft controls)
  T_coll=0., T_latcyc=0., T_lngcyc=0., T_incid=0., T_cant=0., T_diam=0. (all controls, all states)
  KIND_control=1 (1 for thrust and TPP command)
  KIND_coll=2 (1 for thrust, 2 for C_T/σ)
  KIND_cyclic=1 (1 for TPP tilt, 2 for hub moment, 3 for lift offset)
  KIND_tilt=0 (fixed shaft)

g) Wing
Control:
  INPUT_flap=0, INPUT_flaperon=0, INPUT_aileron=0, INPUT_incid=0 (no connection to aircraft controls)
  T_flap=0., T_flaperon=0., T_aileron=0., T_incid=0. (all controls, all states, all panels)
Drag: Idrag=0. (not tilt)

h) Tail
First tail is horizontal tail: KIND_tail=1, WTail%MODEL_Htail=1 (helicopter)
Second tail is vertical tail: KIND_tail=2, WTail%MODEL_Vtail=1 (helicopter)
Configuration: KIND_TailVol=2, TailVolRef=1 (rotor reference)
Control:
  INPUT_cont=1 (tail control connection to aircraft controls), INPUT_incid=0 (no connection of tail incidence to aircraft controls)
  T_cont=0., T_incid=0. (all controls, all states)

i) Propulsion: nGear=1, STATE_gear_wt=1, INPUT_DN=0

j) Engine Group
Configuration: kPropulsion=1, INPUT_gear=1 (gear ratio from N_spec), SET_power=0 (sized), fPsize=1., direction='x', SET_geom=0 (standard position)
Drag: MODEL_drag=1, Idrag=0. (not tilt)

k) Engine Group, Jet Group, Charge Group
Control:
INPUT_amp=0, INPUT_mode=0, INPUT_incid=0, INPUT_yaw=0 (no connection to aircraft controls)
T_amp=0., T_incid=0., T_yaw=0. (all controls, all states)

2–2 Helicopter

a) Rotor
First rotor is main rotor: config='main', fDGW=1., fArea=1., SET_geom='standard'
rotation: $r = 1$; if (Rotor(1)%rotate < 0) $r = -1$
control: INPUT_coll=1, INPUT_latcyc=1, INPUT_lngcyc=1 (rotor control connection to aircraft controls)
control: T_coll(1,1)=1., T_latcyc(2,1)= $-r$, T_lngcyc(3,1)=-1.
Second rotor is tail rotor: config='tail+antiQ', fThrust=1., fArea=0., SET_geom='tailrotor', mainRotor=1
direction='tail', WRotor%MODEL_config=2 (tail rotor)
rotation: $r = 1$; if (Rotor(1)%rotate < 0) $r = -1$
control: KIND_control=2 (thrust and NFP command); INPUT_coll=1, T_coll(4,1)= $-r$ (rotor collective connection to aircraft control 'pedal')
Performance: PRotorInd%MODEL_twin='none'
Drag: SET_Sspin=1, Swet_spin=0., DRotor%SET_Dspin=1, DRotor%DoQ_spin=0., DRotor%CD_spin=0. (no spinner drag)

b) Tail
Control: INPUT_incid=1 (tail incidence connection to aircraft controls)
Horizontal tail: T_incid(5,1)=1. (incidence connection to aircraft control 'tailinc'), T_cont(6,1)=1. (elevator direct control)
Vertical tail: T_cont(7,1)=1. (rudder direct control)

c) Propulsion: WDrive%ngearbox=2, WDrive%ndriveshaft=1, WDrive%fShaft=0.1, WDrive%fTorque=0.03, WDrive%fPower=0.15

2–3 Tandem

a) Components: nTail=0 (no tail)

b) Fuel Tank: place=2 (sponson)

c) Rotor
Configuration: config='main+tandem', fDGW=.5, SET_geom='tandem', fRadius=1.
fArea=$1 - m/2$, from $m = (2/\pi)(\cos^{-1} h - h\sqrt{1-h^2})$, $h = 1 -$ overlap_tandem
First rotor is front rotor: otherRotor=2
rotation: $r = 1$, if (Rotor(1)%rotate < 0) $r = -1$
control: INPUT_coll=1, INPUT_latcyc=1 (rotor control connection to aircraft controls)
control: T_coll(1,1)=1., T_coll(3,1)=-1., T_latcyc(2,1)=$-r$, T_latcyc(4,1)=$-r$
Second rotor is aft rotor: otherRotor=1, rotate=-Rotor(1)%rotate
rotation: $r = 1$, if (Rotor(1)%rotate < 0) $r = -1$; $r = -r$
control: INPUT_coll=1, INPUT_latcyc=1 (rotor control connection to aircraft controls)
control: T_coll(1,1)=1., T_coll(3,1)= 1., T_latcyc(2,1)=$-r$, T_latcyc(4,1)=r
Performance: PRotorInd%MODEL_twin='tandem', PRotorInd%Kh_twin=1., PRotorInd%Kf_twin=0.85, IRotor%MODEL_int_twin=2
Drag: SET_Sspin=1, Swet_spin=0., DRotor%SET_Dspin=1, DRotor%DoQ_spin=0., DRotor%CD_spin=0. (no spinner drag)

d) Propulsion: WDrive%ngearbox=2, WDrive%ndriveshaft=1, WDrive%fShaft=0.1; WDrive%fTorque=0.6, WDrive%fPower=0.6

2–4 Coaxial

a) Rotor
Configuration: config='main+coaxial', fDGW=.5, fArea=.5, SET_geom='coaxial', fRadius=1.
First rotor is lower rotor: otherRotor=2
rotation: $r = 1$, if (Rotor(1)%rotate < 0) $r = -1$
control: INPUT_coll=1, INPUT_latcyc=1, INPUT_lngcyc=1 (rotor control connection to aircraft controls)
control: T_coll(1,1)=1., T_coll(4,1)=r, T_latcyc(2,1)=$-r$, T_lngcyc(3,1)=-1.
Second rotor is upper rotor: otherRotor=1, rotate=-Rotor(1)%rotate
rotation: $r = 1$, if (Rotor(1)%rotate < 0) $r = -1$; $r = -r$
control: INPUT_coll=1, INPUT_latcyc=1, INPUT_lngcyc=1 (rotor control connection to aircraft controls)
control: T_coll(1,1)=1., T_coll(4,1)=r, T_latcyc(2,1)=$-r$, T_lngcyc(3,1)=-1.
Performance: PRotorInd%MODEL_twin='coaxial', PRotorInd%Kh_twin=1., PRotorInd%Kf_twin=0.85, IRotor%MODEL_int_twin=2
Drag: SET_Sspin=1, Swet_spin=0., DRotor%SET_Dspin=1, DRotor%DoQ_spin=0., DRotor%CD_spin=0. (no spinner drag)

b) Tail
Horizontal tail: T_cont(6,1)=1. (elevator direct control)
Vertical tail: T_cont(7,1)=1. (rudder direct control)

c) Propulsion: WDrive%ngearbox=1, WDrive%ndriveshaft=0, WDrive%fShaft=0.1; WDrive%fTorque=0.6, WDrive%fPower=0.6

2–5 Tiltrotor

a) Components: nWing=1, nEngineGroup=2 (engine at each nacelle)

b) Aircraft
Aircraft controls: ncontrol=10, IDENT_control='coll','latcyc','lngcyc','pedal','tilt','flap','flaperon','elevator','aileron','rudder'
Control states: nstate_control=2 (state 1 for helicopter mode, state 2 for airplane mode)
Control state in conversion: kcont_hover=1, kcont_conv=1, kcont_cruise=2
Drive state in conversion: kgear_hover(1)=1, kgear_conv(1)=1, kgear_cruise(1)=1

c) Systems: MODEL_FWfc=1, MODEL_CVfc=1 (fixed wing flight controls, conversion control)

d) Landing Gear: KIND_LG=1 (retractable)

e) Fuel Tank: place=3 (wing), fFuelWing(1)=1.

f) Rotor
Configuration: config='main+tiltrotor', fDGW=.5, fArea=1.; SET_geom='tiltrotor', KIND_TRgeom=1 (from clearance), fRadius=1., WingForRotor=1
First rotor is right rotor: otherRotor=2
helicopter mode control: INPUT_coll=1, INPUT_lngcyc=1 (rotor control connection to aircraft controls)
helicopter mode control: T_coll(1,1)=1., T_coll(2,1)=-1., T_lngcyc(3,1)=-1., T_lngcyc(4,1)=1.
Second rotor is left rotor: otherRotor=1, rotate=-Rotor(1)%rotate; INPUT_gear=2 (input quantity is gear ratio)
helicopter mode control: INPUT_coll=1, INPUT_lngcyc=1 (rotor control connection to aircraft controls)
helicopter mode control: T_coll(1,1)=1., T_coll(2,1)=1., T_lngcyc(3,1)=-1., T_lngcyc(4,1)=-1.
Airplane mode control state: T_coll(1,2)=1. (collective connection to aircraft control 'coll')
Tilt: KIND_tilt=1 (shaft control = incidence), incid_ref=90. (helicopter mode reference), SET_Wmove=1, fWmove=1. (wing tip weight move)
control: INPUT_incid=1, T_incid(5,1)=1., T_incid(5,2)=1. (incidence connection to aircraft control 'tilt')
Performance: PRotorInd%MODEL_twin='tiltrotor', PRotorInd%Kh_twin=1., PRotorInd%Kf_twin=1., IRotor%MODEL_int_twin=2
Weight: WRotor%KIND_rotor=1 (tilting)
Drag: SET_aeroaxes=2 (tiltrotor), Idrag=90. (tiltrotor)
DRotor%SET_Dhub=1, DRotor%DoQ_hub=0., DRotor%CD_hub=0., DRotor%SET_Vhub=1, DRotor%DoQV_hub=0., DRotor%CDV_hub=0. (no hub drag)

g) Wing
Configuration: fDGW=1., nRotorOnWing=2, RotorOnWing(1)=1, RotorOnWing(2)=2, SET_ext=0
Control: KIND_flaperon=3 (independent), nVincid=1
INPUT_flap=1, INPUT_flaperon=1, INPUT_aileron=1 (wing control connection to aircraft controls)
T_aileron(2,2)=-1. (airplane mode aileron connection to aircraft control 'latcyc')

T_flap(6,1)=1., T_flap(6,2)=1. (flap direct control)
T_flaperon(7,1)=1., T_flaperon(7,2)=1. (flaperon direct control)
T_aileron(9,1)=1., T_aileron(9,2)=1. (aileron direct control)
Weight: WWing%MODEL_wing=3 (tiltrotor)

h) Tail
Configuration: KIND_TailVol=1, TailVolRef=1 (wing reference); Wtail%MODEL_Htail=2, Wtail%MODEL_Vtail=2 (tiltrotor)
Horizontal tail control: nVincid=1
T_cont(3,2)=1. (airplane mode elevator connection to aircraft control 'lngcyc')
T_cont(8,1)=1., T_cont(8,2)=1. (elevator direct control)
Vertical tail control: nVincid=1
T_cont(4,2)=1. (airplane mode rudder connection to aircraft control 'pedal')
T_cont(10,1)=1., T_cont(10,2)=1. (rudder direct control)

i) Propulsion: WDrive%ngearbox=2, WDrive%ndriveshaft=1, WDrive%fShaft=0.1; WDrive%fTorque=0.6, WDrive%fPower=0.6

j) Engine Group
Configuration: fPsize=0.5, SET_geom=1 (tiltrotor)
First engine group: RotorForEngine=1
Second engine group: RotorForEngine=2
Control: INPUT_incid=1; T_incid(5,1)=1., T_incid(5,2)=1. (nacelle incidence connection to aircraft control 'tilt')
Drag: SET_Swet=1, Swet=0., MODEL_drag=0, Idrag=90. (no engine nacelle drag)
DEngSys%SET_drag=1, DEngSys%DoQ=0., DEngSys%CD=0.; DEngSys%SET_Vdrag=1, DEngSys%DoQV=0., DEngSys%CDV=0.

2–6 Compound

a) Components: nRotor=3, nWing=1

b) Aircraft
Aircraft controls: ncontrol=10, IDENT_control='coll','latcyc','lngcyc','pedal','tailinc','elevator','rudder','prop','aileron','flap'
Trim states: nstate_trim=11; compound state active

c) Rotor
First rotor is main rotor: config='main', fDGW=1., fArea=1., SET_geom='standard'
rotation: $r = 1$; if (Rotor(1)%rotate < 0) $r = -1$
control: INPUT_coll=1, INPUT_latcyc=1, INPUT_lngcyc=1 (rotor control connection to aircraft controls)

control: T_coll(1,1)=1., T_latcyc(2,1)= $-r$, T_lngcyc(3,1)=-1.
Second rotor is tail rotor: config='tail+antiQ', fThrust=1., fArea=0., SET_geom='tailrotor', mainRotor=1
direction='tail', WRotor%MODEL_config=2 (tail rotor)
rotation: $r=1$; if (Rotor(1)%rotate <0) $r=-1$
control: KIND_control=2 (thrust and NFP command); INPUT_coll=1, T_coll(4,1)= $-r$ (rotor collective connection to aircraft control 'pedal')
Third rotor is propeller: config='prop+auxT', fThrust=1., fArea=0., SET_geom='standard'
direction='prop', WRotor%MODEL_config=3 (auxiliary thrust)
control: KIND_control=2 (thrust and NFP command); INPUT_coll=1, T_coll(8,1)=1. (rotor collective connection to aircraft control 'prop')
Performance: PRotorInd%MODEL_twin='none'
Drag: SET_Sspin=1, Swet_spin=0., DRotor%SET_Dspin=1, DRotor%DoQ_spin=0., DRotor%CD_spin=0. (no spinner drag)

d) Wing
Configuration: fDGW=1.
Control: nVincid=1
INPUT_flap=1, INPUT_flaperon=1, INPUT_aileron=1 (wing control connection to aircraft controls)
T_aileron(9,1)=1. (aileron direct control)
T_flap(10,1)=1. (flap direct control)
Weight: WWing%MODEL_wing=2 (parametric)

e) Tail
Control: INPUT_incid=1 (tail incidence connection to aircraft controls)
Horizontal tail: T_incid(5,1)=1. (incidence connection to aircraft control 'tailinc'), T_cont(6,1)=1. (elevator direct control)
Vertical tail: T_cont(7,1)=1. (rudder direct control)

f) Propulsion: WDrive%ngearbox=3, WDrive%ndriveshaft=1, WDrive%fShaft=0.1, WDrive%fTorque=0.03, WDrive%fPower=0.15

2–7 Multicopter

a) Components: nTail=0 (no tail)

b) Rotor
Configuration: config='main+multirotor', fDGW=1/nRotor, fArea=1., SET_geom='multicopter'
Control: KIND_control=2 (thrust and NFP command); INPUT_coll=1
rotation: $r=1$; if (rotate <0) $r=-1$; a =ang_multicopter
T_coll(1,1)=1., T_coll(2,1)=$-\sin(a)$, T_coll(3,1)=$\cos(a)$, T_coll(4,1)=r (rotor collective connection to aircraft controls)

Performance: PRotorInd%MODEL_twin='multirotor'; xh_multi=0., xp_multi=0., xf_multi=0., except 1.0 for this rotor
Drag: SET_Sspin=1, Swet_spin=0., DRotor%SET_Dspin=1, DRotor%DoQ_spin=0., DRotor%CD_spin=0. (no spinner drag)

c) Propulsion: WDrive%ngearbox=nRotor, WDrive%ndriveshaft=nRotor-1, WDrive%fShaft=0.1; WDrive%fTorque=0.6, WDrive%fPower=0.6

2–8 Airplane

a) Components: nRotor=1, nWing=1

b) Solution: KIND_Lscale=2 (wing span reference)

c) Aircraft
Geometry: INPUT_geom=2, KIND_scale=2, kScale=1 (geometry scaled with wing span); KIND_Ref=2, kRef=1 (wing reference)
Aircraft controls: ncontrol=9, IDENT_control='coll','latcyc','lngcyc','pedal','tailinc','elevator','rudder','aileron','flap'
coll = propeller, latcyc = lateral stick, lngcyc = longitudinal stick

d) Systems: MODEL_FWfc=1 (fixed wing flight controls)

e) Rotor
Propeller: config='prop+auxT', fThrust=1., fDGW=0., SET_geom='standard'
direction='prop', WRotor%MODEL_config=3 (auxiliary thrust)
Control: KIND_control=2 (thrust and NFP command); INPUT_coll=1, T_coll(1,1)=1. (rotor collective connection to aircraft control 'coll')

f) Wing
Configuration: fDGW=1.
Control: nVincid=1
INPUT_flap=1, INPUT_aileron=1 (wing control connection to aircraft controls)
T_aileron(2,1)=1. (lateral stick), T_aileron(8,1)=1. (aileron direct control)
T_flap(9,1)=1. (flap direct control)
Weight: WWing%MODEL_wing=2 (parametric)

g) Tail: KIND_TailVol=1, TailVolRef=1 (wing reference)
Control: INPUT_incid=1 (tail incidence connection to aircraft controls)
Horizontal tail: T_incid(5,1)=1. (incidence connection to aircraft control 'tailinc'), T_cont(3,1)=1. (longitudinal stick), T_cont(6,1)=1. (elevator direct control)
Vertical tail: T_cont(4,1)=1. (pedal), T_cont(7,1)=1. (rudder direct control)

h) Propulsion: WDrive%ngearbox=1, WDrive%ndriveshaft=1, WDrive%fShaft=0.1

Chapter 3

Parameters

Parameters	Value				
ncasemax	10	ndesignmax	41	ngetab2max	20
nfilemax	40	ncontmax	20	npanelmax	5
nrotormax	16	nsweepmax	200	nauxtankmax	4
npropmax	16	qsweepmax	4	ngearmax	8
nengmax	16	ntrimstatemax	20	nratemax	20
njetmax	4	mtrimmax	16	nengtmax	20
nchrgmax	4	nvnemax	32	nengxmax	100
nstatemax	10	niasmax	40	nengkmax	6
nwingmax	8	nvelmax	20	nengrmax	40
ntailmax	6	ntablemax	32	nengpmax	20
ntankmax	4	nrmax	51	nengcmax	80
nmissmax	20	mrmax	40	nspeedmax	8
nsegmax	40	mpsimax	36	nrowmax	4000
nfltmax	21	ngetabmax	40	naeromax	100

Chapter 4

Common: Job

Variable	Type		Description	Default
		+	Initialization	
INIT_input	int	+	input parameters (0 default, 1 last case input, 2 last case solution)	1
INIT_data	int	+	other parameters (0 default, 1 start of last case, 2 end of last case)	0

INIT_input:
if default, all input variables set to default values
if last-case-input, then case inherits input at beginning of previous case
if last-case-solution, then case inherits input at end of previous case
use INIT_input=2 to analyze case #1 design in subsequent cases
INIT_data: if always start-last-case, then case starts from default
if default, all other variables set to default values

Variable	Type		Description	Default
		+	Errors	
ACT_error	int	+	action on error (0 none, 1 exit)	1
ACT_version	int	+	action on version mismatch in input (0 none, 1 exit)	0
		+	File open	
OPEN_status	int	+	status keyword for write (0 unknown, 1 replace, 2 new, 3 old)	2
		+	Input/output unit numbers	
		+	input	
nuin	int	+	standard input	5
nufile	int	+	secondary file input	40
		+	output	
nuout	int	+	standard output	6
nudesign	int	+	design (DESIGNn)	41

nuperf	int	+	performance (PERFn)	42
nuaero	int	+	airframe aerodynamics (AEROn)	43
nuengine	int	+	engine performance (ENGINEn)	44
nugeom	int	+	geometry output (GEOMETRYn)	45
nuacd	int	+	aircraft description (AIRCRAFTn)	46
nusoln	int	+	solution (SOLUTIONn)	47
nusketch	int	+	sketch output (SKETCHn)	48
nuerror	int	+	errors (ERRORn)	49

default input/output unit numbers usually acceptable
default OPEN_status can be changed as appropriate for computer OS

Chapter 5

Structure: Cases

Variable	Type		Description	Default
		+	Case Description	
title	c*100	+	title	
subtitle1	c*100	+	subtitle	
subtitle2	c*100	+	subtitle	
subtitle3	c*100	+	subtitle	
notes	c*1000	+	notes	
ident	c*32	+	identification	
		+	Case Tasks (0 for none)	
TASK_Size	int	+	size aircraft for design conditions	1
TASK_Mission	int	+	mission analysis	1
TASK_Perf	int	+	flight performance analysis	1
TASK_Map_engine	int	+	map of engine performance	0
TASK_Map_aero	int	+	map of airframe aerodynamics	0

Turn off all tasks to just initialize and check the model, including geometry and weights

Variable	Type		Description	Default
		+	Write Input Parameters	
WRITE_input	int	+	selection (0 none, 1 all, 2 first case)	2
WRITE_input_TechFactors	int	+	TechFactors (0 for none)	1
WRITE_input_Geometry	int	+	Geometry (0 for none)	1
		+	Output	
		+	selection (0 for none)	
OUT_design	int	+	design file	0

OUT_perf	int	+	performance file	0
OUT_geometry	int	+	geometry file	0
OUT_aircraft	int	+	aircraft description file	0
OUT_solution	int	+	solution file (1 text, 2 binary)	0
OUT_sketch	int	+	sketch file	0
OUT_error	int	+	errors file	0
		+	file name or logical name (blank for default logical name)	
FILE_design	c*256	+	design file (DESIGNn)	' '
FILE_perf	c*256	+	performance file (PERFn)	' '
FILE_geometry	c*256	+	geometry file (GEOMETRYn)	' '
FILE_aircraft	c*256	+	aircraft description file (AIRCRAFTn)	' '
FILE_solution	c*256	+	solution file (SOLUTIONn)	' '
FILE_sketch	c*256	+	sketch file (SKETCHn)	' '
FILE_engine	c*256	+	engine performance file (ENGINEn)	' '
FILE_aero	c*256	+	airframe aerodynamics file (AEROn)	' '
FILE_error	c*256	+	errors file (ERRORn)	' '
		+	formats	
WRITE_page	int	+	page control (0 none, 1 form feed, 2 extended Fortran)	1
WRITE_design	int	+	design (1 first case only, 2 all cases)	2
WRITE_wt_level	int	+	weight statement, max level (1 to 5)	5
WRITE_wt_long	int	+	weight statement, style (0 omit zero lines, 1 all lines)	0
WRITE_energy	int	+	fuel energy for burn weight (0 for none)	1
WRITE_flight	int	+	flight state, component loads (0 for none)	0
WRITE_files	int	+	design, performance, or geometry (1 single file of all cases)	0
WRITE_sketch_load	int	+	sketch component forces (0 none)	1
WRITE_sketch_cond	int	+	sketch flight condition (0 none, 1 design, 2 performance)	0
ksketch	int	+	flight condition number	0

---

selected files are generated for each case (n = case number in default name)
option single file of all cases for design, performance, or geometry (form feed between cases)
size and analysis tasks can produce design and performance files
same information as in standard output, in tab-delimited form
aircraft or solution file can be read by subsequent case or job
geometry file has information for graphics and other analyses

sketch file has information to check geometry and solution (DXF format)
flight condition required to use Euler angles, control and incidence, component forces
engine map task (TASK_Map_engine) produces engine performance file
airframe aerodynamics map task (TASK_Map_aero) produces airframe aerodynamics file
error messages to standard output (OUT_error=0) or separate file (OUT_error=1)

		+	Gravity	
SET_grav	int	+	specification (0 standard, 1 input)	0
grav	real	+	input gravitational acceleration g	
		+	Environment	
density_ref	real	+	reference density (0. for air at SLS)	0.
csound_ref	real	+	reference speed of sound (0. for air at SLS)	0.
		+	Units	
Units	int	+	analysis units (1 English, 2 SI)	1
		+	units for input of missions and flight conditions	
Units_miss	int	+	override default units (0 no, 1 yes)	0
Units_vel	int	+	velocity units (0 knots; 1 mile/hr, 2 km/hr, 3 ft/sec, 4 m/sec)	0
Units_alt	int	+	altitude units (0 ft or m; 1 ft, 2 m)	0
Units_pay	int	+	payload units (0 lb or kg; 1 lb, 2 kg)	0
Units_time	int	+	time units (0 minutes; 1 hours)	0
Units_dist	int	+	distance units (0 nm; 1 miles; 2 km)	0
Units_temp	int	+	temperature (0 F or C; 1 F, 2 C)	0
Units_drag	int	+	drag units (0 ft^2 or m^2; 1 ft^2, 2 m^2)	0
Units_ROC	int	+	rate of climb units (0 ft/min; 1 ft/sec, 2 m/sec)	0
		+	units for parameters	
Units_Dscale	int	+	input D/q scaled with gross weight (0 analysis default, 1 English, 2 SI)	0
Units_energy	int	+	units for energy input and output (1 MJ, 2 kWh)	1

Analysis units: must be same for all cases in job
English: ft-slug-sec-F; weights in lb, power in hp (internal units)
SI: m-kg-sec-C; weights in kg, power in kW (internal units)

Weight in the design description is actually mass
    pounds converted to slugs using reference gravitational acceleration
Default units for flight condition and mission: override with Units_xxx
    speed in knots, time in minutes, distance in nm, ROC in ft/min
Input Efuel_cap, Eaux_cap always MJ; internal energy units MJ

Structure: Size

Variable	Type		Description	Default
		+	Size Aircraft for Design Conditions and Missions	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Sizing Method	
SIZE_perf(npropmax)	c*16	+	quantity sized from performance	'engine'
SIZE_engine(nengmax)	c*16	+	engine group sized from performance	'none'
SIZE_jet(njetmax)	c*16	+	jet group sized from performance	'jet'
SIZE_charge(nchrgmax)	c*16	+	charge group sized from performance	'none'
SIZE_param	int	+	parameter iteration (0 not required)	0
SET_rotor(nrotormax)	c*32	+	rotor parameters	'DL+Vtip+CWs'
SET_wing(nwingmax)	c*16	+	wing parameters	'WL+aspect'
FIX_DGW	int	+	design gross weight (0 calculated, 1 fixed)	0
FIX_WE	int	+	weight empty (0 calculated, 1 fixed, 2 scaled)	0
SET_tank(ntankmax)	c*16	+	fuel tank capacity	'miss'
SET_SDGW	c*16	+	structural design gross weight	'f(DGW)'
SET_WMTO	c*16	+	maximum takeoff weight	'f(DGW)'
SET_limit_ds(npropmax)	c*16	+	drive system torque limit	'ratio'

---

size task (Cases%TASK_Size=1): at least one nFltCond or nMission
no size task (Cases%TASK_Size=0): size input specifies how fixed aircraft determined

SIZE_perf: size power-producing engines of propulsion group
- 'engine' = power from maximum of power required for all designated conditions and missions
- 'rotor' = radius from maximum of power required for all designated conditions and missions
- 'none' = power required not used to size engine/rotor
  - flight conditions and missions (max GW, max effort, or trim)
    - that have zero power margin are not used to size engine or rotor
    - that have zero torque margin are not used to size transmission

SIZE_engine: size power-consuming engines of engine group
'engine' = power from maximum of power required for all designated conditions and missions
flight conditions and missions (max GW, max effort, or trim)
that have zero power margin are not used to size engine group
designated only for engine groups that consume power
engine groups that produce power sized with propulsion group (SIZE_perf)
'none' = power required not used to size engine group
SIZE_jet:
'jet' = thrust from maximum of thrust required for all designated conditions and missions
'none' = thrust required not used to size jet group
flight conditions and missions (max GW, max effort, or trim)
that have zero thrust margin are not used to size jet group
SIZE_charge:
'charge' = power from maximum of power required for all designated conditions and missions
'none' = power required not used to size charge group
'SIZE_param': use to force parameter iteration

SET_rotor, rotor parameters: required for each rotor
rotor parameters: input three or two quantities, others derived
SET_rotor = input three of ('radius' or disk loading 'DL' or 'ratio'), 'CWs', 'Vtip', 'sigma'
except if SIZE_perf='rotor': SET_rotor = input two of 'CWs', 'Vtip', 'sigma' for one or more main rotors
SET_rotor = 'ratio+XX+XX' to calculate radius from radius of another rotor
tip speed is Vtip_ref for drive state #1
rotor parameters for an antitorque or aux thrust rotor:
SET_rotor = input three of ('radius' or 'DL' or 'ratio' or 'scale'), 'CWs', 'Vtip', 'sigma'
SET_rotor = 'scale+XX+XX' to calculate tail rotor radius from parametric equation,
using main rotor radius and disk loading
thrust from designated sizing conditions and missions (DESIGN_thrust)

SET_wing, wing parameters: for each wing; input two quantities, other two derived
SET_wing = input two of ('area' or wing loading 'WL'), ('span' or 'ratio' or 'radius'or 'width' or 'hub' or 'panel'),
'chord', aspect ratio 'aspect'
SET_wing = 'ratio+XX' to calculate span from span of another wing
SET_wing = 'radius+XX' to calculate span from rotor radius
SET_wing = 'width+XX' to calculate span from rotor radius, fuselage width, and clearance (tiltrotor)

SET_wing = 'hub+XX' to calculate span from rotor hub position (tiltrotor)
SET_wing = 'panel+XX' to calculate span from wing panel widths

FIX_DGW: input DGW restricts SIZE_perf, SET_GW parameters
FIX_WE: fixed or scaled weight empty obtained by adjusting contingency weight
scaled with design gross weight: W_E=dWE+fWE*W_D

SET_tank, fuel tank sizing: usable fuel capacity Wfuel_cap (weight) or Efuel_cap (energy)
'input' = input Wfuel_cap or Efuel_cap
'miss' = calculate from mission fuel used
Wfuel_cap or Efuel_cap = max(fFuel_cap*(maximum mission fuel), (maximum mission fuel)+(reserve fuel))
'f(miss)' = function of mission fuel used
Wfuel_cap or Efuel_cap = dFuel_cap + fFuel_cap*((maximum mission fuel)+(reserve fuel))
'used' = calculate from maximum fuel quantity in tank during mission
Wfuel_cap or Efuel_cap = dFuel_cap + fFuel_cap*(maximum fuel in tank)
'XX+power' = and calculate from mission battery discharge power

SET_SDGW, structural design gross weight:
'input' = input
'f(DGW)' = based on DGW; W_{SD}=dSDGW+fSDGW*W_D
'f(WMTO)' = based on WMTO; W_{SD}=dSDGW+fSDGW*W_{MTO}
'maxfuel' = based on fuel state; W_{SD}=dSDGW+fSDGW*W_G, $W_G = W_D$–Wfuel_DGW+fFuelSDGW*$W_{\text{fuel}-\text{cap}}$
'perf' = calculated from maximum gross weight at SDGW sizing conditions (DESIGN_sdgw)
Aircraft input parameters: dSDGW, fSDGW, fFuelSDGW
SET_WMTO, maximum takeoff weight:
'input' = input
'f(DGW)' = based on DGW; W_{MTO}=dWMTO+fWMTO*W_D
'f(SDGW)' = based on SDGW; W_{MTO}=dWMTO+fWMTO*W_{SD}
'maxfuel' = based on maximum fuel; W_{MTO}=dWMTO+fWMTO*W_G, $W_G = W_D$–Wfuel_DGW+$W_{\text{fuel}-\text{cap}}$
'perf' = calculated from maximum gross weight at WMTO sizing conditions (DESIGN_wmto)
Aircraft input parameters: dWMTO, fWMTO

SET_limit_ds, drive system torque limit: input (use Plimit_xx) or calculate (from fPlimit_xx)
'input' = Plimit_ds input
'ratio' = from takeoff power, fPlimit_ds$\sum(N_{\rm eng} P_{\rm eng})$
'Pav' = from engine power available at transmission sizing conditions and missions (DESIGN_xmsn)
fPlimit_ds$(\Omega_{\rm ref}/\Omega_{\rm prim}) \sum(N_{\rm eng} P_{av})$
'Preq' = from engine power required at transmission sizing conditions and missions (DESIGN_xmsn)
fPlimit_ds$(\Omega_{\rm ref}/\Omega_{\rm prim}) \sum(N_{\rm eng} P_{req})$
engine shaft limit also uses EngineGroup%SET_limit_es
rotor shaft limit also uses Rotor%SET_limit_rs, rotor limits only use power required (or input)

input required to transmit sized rotorcraft to another job (through aircraft description file) or to following case:
turn off sizing: Cases%TASK_size=0, Cases%TASK_mission=1, Cases%TASK_perf=1
fix aircraft: use ACTION='nosize', or
SIZE_perf='none', SIZE_engine='none', SIZE_jet='none', SIZE_charge='none'
SET_rotor='radius+Vtip+sigma', SET_wing='area+span', FIX_DGW=1
SET_tank='input', SET_limit_ds='input', SET_SDGW='input', SET_WMTO='input'
with wing panels: SET_wing='WL+panel', Wing%SET_panel='width+taper','span+taper'

		+	Sizing Flight Conditions	
nFltCond	int	+	number of conditions (maximum nfltmax)	0
		+	Design Missions	
nMission	int	+	number of missions (maximum nmissmax)	0

input one condition (FltCond and FltState variables) in SizeCondition namelist

input one mission (MissParam, MissSeg, and FltState variables) in SizeMission namelist
all mission segments are defined in this namelist, so MissSeg and FltState variables are arrays
each variable gets one more dimension, first array index is always segment number

Chapter 7

Structure: OffDesign

Variable	Type		Description	Default
		+	Mission Analysis	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Missions	
nMission	int	+	number of missions (maximum nmissmax)	0

mission analysis input required if Cases%TASK_Mission=1

input one mission (MissParam, MissSeg, and FltState variables) in OffMission namelist
all mission segments are defined in this namelist, so MissSeg and FltState variables are arrays
each variable gets one more dimension, first array index is always segment number

Chapter 8

Structure: Performance

Variable	Type		Description	Default
		+	Flight Performance Analysis	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Performance Flight Conditions	
nFltCond	int	+	number of conditions (maximum nfltmax)	0

flight performance analysis input required if Cases%TASK_Perf=1

input one condition (FltCond and FltState variables) in PerfCondition namelist

Chapter 9

Structure: MapEngine

Variable	Type		Description	Default
		+	Map of Engine Performance	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Identification	
kEngineGroup	int	+	engine group	1
KIND_map	int	+	Kind (1 performance, 2 model)	1

engine map only available for RPTEM model and reciprocating engine model (performance only)

engine map input required if Cases%TASK_Map_engine=1
only performance parameters or only model parameters used

Variable	Type		Description	Default
		+	Performance	
		+	independent variables (0 none, 1 altitude, 2 temperature, 3 flight speed, 4 engine speed, 5 power)	
SET_var(5)	int	+	first set	0
SET_var2(5)	int	+	second set	0
WRITE_header	int	+	output format (1 single header, 2 header for inner variable)	2
SET_atmos	c*12	+	atmosphere specification	'std'
		+	altitude h (Units_alt)	
altitude_min	real	+	minimum	0.
altitude_max	real	+	maximum	20000.
altitude_inc	real	+	increment	1000.
altitude_base	real	+	baseline	0.

		+	temperature τ or temperature increment ΔT (Units_temp)	
temp_min	real	+	minimum	0.
temp_max	real	+	maximum	100.
temp_inc	real	+	increment	10.
temp_base	real	+	baseline	0.
		+	flight speed V (TAS, Units_vel)	
Vkts_min	real	+	minimum	0.
Vkts_max	real	+	maximum	200.
Vkts_inc	real	+	increment	50.
Vkts_base	real	+	baseline	0.
SET_rpm	int	+	engine speed N (1 rpm, 2 percent)	2
Nturbine_min	real	+	minimum	90.
Nturbine_max	real	+	maximum	110.
Nturbine_inc	real	+	increment	5.
Nturbine_base	real	+	baseline	100.
SET_power	int	+	power required (1 power, 2 fraction of power available (0. to 1.+)	2
power_min	real	+	minimum	.1
power_max	real	+	maximum	1.
power_inc	real	+	increment	.1
power_base	real	+	baseline	1.
STATE_IRS	int	+	IR suppressor system state (0 off, hot exhaust; 1 on, suppressed exhaust)	0
KIND_loss	int	+	installation losses (0 for none)	0

---

independent variables: 1 to 5 variables, last is innermost loop; outer loop is always rating
quantities not identified as independent variables fixed at baseline values

SET_atmos, atmosphere specification:
determines whether temp_xxx is temperature or temperature increment
'std' = standard day at specified altitude (use altitude_xxx)
'temp' = standard day at specified altitude, and specified temperature (use altitude_xxx, temp_xxx)
'dtemp' = standard day at specified altitude, plus temperature increment (use altitude_xxx, temp_xxx)
see FltState%SET_atmos for other options (polar, tropical, and hot days)

		+	Model	
		+	flight speeds V (TAS, Units_vel)	
nV_model	int	+	number (maximum 10)	1
V_model(10)	real	+	values	0.
V_min	real	+	minimum	0.
V_max	real	+	maximum	400.
V_inc	real	+	increment	50.
		+	temperature ratio T/T_0	
ntheta_model	int	+	number (maximum 10)	1
theta_model(10)	real	+	values	1.
theta_min	real	+	minimum	.8
theta_max	real	+	maximum	1.1
theta_inc	real	+	increment	.02
		+	engine speed, $N/N_{\rm spec}$ (percent)	
fN_min	real	+	minimum	90.
fN_max	real	+	maximum	110.
fN_inc	real	+	increment	5.
		+	fraction static MCP power, P/P_{0C}	
fP_min	real	+	minimum	.1
fP_max	real	+	maximum	2.
fP_inc	real	+	increment	.1

RPTEM model
- performance: fuel flow, mass flow, net jet thrust, optimum turbine speed
  - vs power fraction and airspeed (use fP and V_model)
- turbine speed: power ratio vs turbine speed and airspeed (use fN and V_model)
- power available: specific power, mass flow, power, fuel flow
  - vs temperature ratio (use theta and V_model)
  - vs airspeed (use V and theta_model)

Chapter 10

Structure: MapAero

Variable	Type		Description	Default
		+	Map of Airframe Aerodynamics	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Tables	
KIND_table	int	+	kind (1 one-dimensional, 2 multi-dimensional)	1
		+	aerodynamic loads (0 for components off)	
SET_fuselage	int	+	fuselage and landing gear	1
SET_tail	int	+	tails	1
SET_wing	int	+	wings	1
SET_rotor	int	+	rotors	1
SET_engine	int	+	engines and fuel tank	1

airframe aerodynamics map input required if Cases%TASK_Map_aero=1

multi-dimensional: generate 6 files of three-dimensional tables
one file for each `load=DRAG, SIDE, LIFT, ROLL, PITCH, YAW`
filename=`FILE_aero//load` or `AEROn//load`
one-dimensional: generate 1 file of all six loads
function of single independent variable = var_lift(1)

Variable	Type		Description	Default
		+	Operating Condition	
STATE_control	int	+	aircraft control state	1
STATE_LG	c*12	+	landing gear state	'retract'
Nauxtank(nauxtankmax,ntankmax)	int	+	number of auxiliary fuel tanks N_{auxtank} (each aux tank size)	0
SET_extkit	int	+	wing extension kit on aircraft (0 none, 1 present)	1
KIND_alpha	int	+	angle of attack and sideslip angle representation (1 conventional, 2 reversed)	1

SET_comp_control	int	+	use component control (0 for $c = Tc_{AC}$; 1 for $c = Tc_{AC} + c_0$)	0
control(ncontmax)	real	+	aircraft controls	0.
tilt	real	+	tilt	0.
alpha	real	+	angle of attack α	0.
beta	real	+	sideslip angle β	0.

landing gear state: STATE_LG='extend', 'retract' (keyword = ext, ret)

		+	Independent variables	
var_lift(3)	c*16	+	lift	
var_drag(3)	c*16	+	drag	
var_side(3)	c*16	+	side force	
var_pitch(3)	c*16	+	pitch moment	
var_roll(3)	c*16	+	roll moment	
var_yaw(3)	c*16	+	yaw moment	
		+	Variable range	
		+	angle of attack and sideslip variation	
angle_lowinc	real	+	low range increment (deg)	2.
angle_highinc	real	+	high range increment (deg)	5.
angle_low	real	+	low range value (deg)	40.
angle_max	real	+	maximum value (deg)	180.
		+	control variation	
control_lowinc	real	+	low range increment (deg)	2.
control_highinc	real	+	high range increment (deg)	2.
control_low	real	+	low range value (deg)	45.
control_max	real	+	maximum value (deg)	90.
		+	third independent variable	
gamma_lowinc	real	+	low range increment (deg)	20.
gamma_highinc	real	+	high range increment (deg)	20.
gamma_low	real	+	low range value (deg)	60.
gamma_max	real	+	maximum value (deg)	60.

var_load identify independent variables
only var_lift(1) used for KIND_table=one-dimensional
values: 'alpha', 'beta', IDENT_control(ncontrol)
var_load(2) blank for 1D table, var_load(3) blank for 2D table
alpha/beta/controls/tilt fixed if not independent variable (tilt replace control(ktilt))
assume control system defined so aircraft controls connected to flaperon, elevator, aileron, rudder

angle, control, gamma variation: by lowinc for -low to +low; by highinc to -max and +max
maximum total values = naeromax

Chapter 11

Structure: FltCond

Variable	Type		Description	Default
		+	Sizing or Performance Flight Condition	
title	c*100	+	title	
label	c*8	+	label	
		+	Specification	
SET_GW	c*12	+	gross weight	'DGW'
GW	real	+	input gross weight W_G	0.
dGW	real	+	gross weight increment	0.
fGW	real	+	gross weight factor	1.
dPav(npropmax)	real	+	power increment, each propulsion group	0.
fPav(npropmax)	real	+	power factor, each propulsion group	1.
dTav(njetmax)	real	+	thrust increment, each jet group	0.
fTav(njetmax)	real	+	thrust factor, each jet group	1.
SET_Wlimit	c*12	+	gross weight limit	'none'
Wlimit	real	+	input gross weight limit	0.
SET_alt	int	+	altitude (0 input, 1 from KIND_source)	0
		+	source for gross weight and altitude	
KIND_source	int	+	kind (1 size mission, 2 size condition, 3 off design mission, 4 performance condition)	1
kSource	int	+	mission or condition number	0
kSegment	int	+	segment number	0
seg_source	int	+	segment (1 start, 2 midpoint)	1
SET_UL	c*12	+	useful load	'pay'
Wpay	real	+	input payload weight $W_{\rm pay}$ (Units_pay)	0.
Npass	int	+	number of passengers $N_{\rm pass}$	0
Wpay_cargo	real	+	cargo $W_{\rm cargo}$ (Units_pay)	0.
Wpay_extload	real	+	external load $W_{\rm ext-load}$ (Units_pay)	0.
Wpay_ammo	real	+	ammunition $W_{\rm ammo}$ (Units_pay)	0.
Wpay_weapons	real	+	weapons $W_{\rm weapons}$ (Units_pay)	0.

		+	fuel tank system	
dFuel(ntankmax)	real	+	fuel weight or energy increment	0.
fFuel(ntankmax)	real	+	fuel capacity factor	1.
SET_auxtank(ntankmax)	int	+	auxiliary fuel tanks (1 adjust Nauxtank, 2 only increase, 0 no change)	1
mauxtank(ntankmax)	int	+	tank size changed (–1 first, –2 first size already used, m for m-th size)	-1
dNauxtank(ntankmax)	int	+	number tanks added or dropped	1
Nauxtank(nauxtankmax,ntankmax)	int	+	number of auxiliary fuel tanks N_{auxtank} (each aux tank size)	
		+	fixed useful load	
dWcrew	real	+	crew weight increment	0.
dNcrew	int	+	number of crew increment δN_{crew}	0
dWoful(10)	real	+	other fixed useful load increment (nWoful categories)	0.
dWequip	real	+	equipment weight increment	0.
dNcrew_seat	int	+	crew seat increment $\delta N_{\text{crew-seat}}$	0
dNpass_seat	int	+	passenger seat increment $\delta N_{\text{pass-seat}}$	0
		+	kits on aircraft (0 none, 1 present)	
SET_foldkit	int	+	folding kit	1
SET_extkit(nwingmax)	int	+	wing extension kit	1
SET_wingkit(nwingmax)	int	+	wing kit on aircraft	1
SET_otherkit	int	+	other kit on aircraft	0
DESIGN_engine	int	+	design condition for power (1 to use for engine sizing)	1
DESIGN_jet	int	+	design condition for jet thrust (1 to use for jet group sizing)	1
DESIGN_charge	int	+	design condition for charge power (1 to use for charge group sizing)	1
DESIGN_GW	int	+	design condition for DGW (1 to use for DGW calculation)	1
DESIGN_xmsn	int	+	design condition for transmission (1 to use for transmission sizing)	1
DESIGN_sdgw	int	+	design condition for SDGW (1 to use for SDGW calculation)	1
DESIGN_wmto	int	+	design condition for WMTO (1 to use for WMTO calculation)	1
DESIGN_thrust	int	+	design condition for antitorque or aux thrust (1 to use for rotor sizing)	1

label is short description for output
sizing flight condition: use all parameters except sweep
fixed gross weight conditions not used to determine DGW, SDGW, WMTO
(set DESIGN_GW=0, DESIGN_sdgw=0, DESIGN_wmto=0)
condition not used to size engine or rotor if power margin fixed (max GW, max effort, or trim)
condition not used to size transmission if zero torque margin (max GW, max effort, or trim)

performance flight condition: not use DESIGN_xx
SET_GW, SET_UL values determine which input parameters used

SET_GW, set gross weight W_G:
- 'DGW' = design gross weight W_D; input (FIX_DGW) or calculated
- 'SDGW' = structural design gross weight W_{SD} (may depend on DGW)
- 'WMTO' = maximum takeoff gross weight W_{MTO} (may depend on DGW)
- 'f(DGW)' = function DGW: fGW*W_D+dGW
- 'f(SDGW)' = function SDGW: fGW*W_{SD}+dGW
- 'f(WMTO)' = function WMTO: fGW*W_{MTO}+dGW
- 'input' = input (use GW)
- 'source' = gross weight from specified mission segment or flight condition (KIND_source)
- 'f(source)' = function of source: fGW*W_{source}+dGW
- 'maxP', 'max' = maximum GW for power required equal specified power: P_{req} = fPavP_{av} + dPav
  $\min((fP_{avPG} + d) - P_{reqPG}) = 0$, over all propulsion groups
- 'maxQ' = maximum GW for transmission torque equal limit: zero torque margin
  $\min(P_{\text{limit}} - P_{req}) = 0$, over all propulsion groups, engine groups, and rotors
- 'maxPQ', 'maxQP' = maximum GW for power required equal specified power and transmission torque equal limit
  most restrictive of power and torque margins
- 'maxJ' = maximum GW for jet thrust required equal specified thrust: T_{req} = fTavT_{av} + dTav
  $\min((fT_{avJG} + d) - T_{reqJG}) = 0$, over all jet groups
- 'maxPJ', 'maxQJ', 'maxPQJ' = maximum GW for most restrictive of power, torque, and thrust margins
- 'pay+fuel' = input payload and fuel weights; gross weight fallout

SET_Wlimit: weight limit for SET_GW='max'
- 'none' = no limit
- 'f(DGW)' = function DGW: fGW*W_D+dGW
- 'f(SDGW)' = function SDGW: fGW*W_{SD}+dGW
- 'f(WMTO)' = function WMTO: fGW*W_{MTO}+dGW
- 'input' = input (use Wlimit)

SET_UL, set useful load: with fixed useful load adjustments in fallout weight
- 'pay' = input payload weight (Wpay); fuel weight fallout
- 'fuel' = input fuel weight (dFuel, fFuel, Nauxtank); payload weight fallout
- 'pay+fuel' = input payload and fuel weights; gross weight fallout

if SET_GW='pay+fuel', assume SET_UL same (actual SET_UL ignored)

KIND_source, source for gross weight or altitude: source must be solved before this condition
calculation order: size missions, size conditions, off design missions, performance conditions

input fuel weight: $W_{\rm fuel} = \min(\text{dFuel}+\text{fFuel}*W_{\rm fuel-cap}, W_{\rm fuel-cap}) + \sum \text{Nauxtank}*W_{\rm aux-cap}$

auxiliary fuel tanks: SET_auxtank used for fallout fuel weight (SET_UL='pay')
adjust Nauxtank for first fuel tank system with SET_auxtank > 0
otherwise number of auxiliary fuel tanks fixed at input value

payload: only Wpay used if SET_Wpayload = no details
crew: only dWcrew used if SET_Wcrew = no details
equipment: dNcrew_seat and dNpass_seat require non-zero weight per seat

		+	Parameter sweep	
SET_sweep	int	+	sweep (0 for none, 1 from list, 2 from range)	0
KIND_sweep	int	+	kind (1 single sweep sequence, 2 nested sweeps)	1
INIT_sweep	int	+	initialize trim (0 for not)	0
nquant_sweep	int	+	number of swept quantities (1 to qsweepmax)	1
quant_sweep(qsweepmax)	c*12	+	quantity (parameter name)	
		+	range	
sweep_first(qsweepmax)	real	+	first parameter value	
sweep_last(qsweepmax)	real	+	last parameter value	
sweep_inc(qsweepmax)	real	+	parameter increment	
		+	list	
nsweep(qsweepmax)	int	+	number of values (maximum nsweepmax)	
sweep(nsweepmax,qsweepmax)	real	+	parameter values	

Parameter sweep: only for performance flight conditions, not sizing flight conditions
maximum total number of values for all conditions is nsweepmax
KIND_sweep: single sweep, simultaneously varying nquant_sweep quantities; or nquant_sweep nested sweeps
Sweeps executed from sweep_last to sweep_first
sweep analyzed using single data structure, only solution for sweep_first saved (last value executed)
sweep_last (first value executed) should be condition that will converge
sign of parameter step determined by sign of (sweep_last-sweep_first); sign of sweep_inc ignored

Single sweep sequence: only use nsweep(1)
sweep_inc of first quantity determines number of values, sweep_inc of other quantities not used
INIT_sweep: control/pitch/roll values of trim iteration initialized from previous condition of sweep
Available parameters: quant_sweep = parameter name
GW, dGW, fGW, dPavn, fPavn, dTavn, fTavn, Wpay, dFueln, fFueln, dWcrew, dWequip
Vkts, Mach, ROC, climb, side, pitch, roll, rate_turn, nz_turn, bank_turn, rate_pullup, nz_pullup
ax_linear, ay_linear, az_linear, nx_linear, ny_linear, nz_linear
altitude, dtemp, temp, density, csound, viscosity, HAGL
controln, coll, latcyc, lngcyc, pedal, tilt, Vtipn, Npecn, fPower, fThrust, fCharge, fTorque
DoQ_pay, fDoQ_pay, DoQV_pay, dSLcg, dBLcg, dWLcg, trim_targetn
n = propulsion group (Vtip, Nspec, dPav, fPav), jet group (dTav, fTav), fuel tank system, control number, or trim quantity
n = 1 if absent from quant_sweep
for fPower, value is factor on input fPower for all engine groups, all propulsion groups
for fThrust, value is factor on input fThrust for all jet groups
for fCharge, value is factor on input fCharge for all charge groups
for fTorque, value is factor on input fTorque for for all propulsion groups

Chapter 12

Structure: Mission

Variable	Type		Description	Default
		+	Mission Profile	
title	c*100	+	title	
label	c*8	+	label	
		+	Specification	
SET_GW	c*16	+	mission takeoff gross weight W_G	'pay+miss'
GW	real	+	input gross weight	0.
dGW	real	+	gross weight increment	0.
fGW	real	+	gross weight factor	1.
SET_Wlimit	c*16	+	gross weight limit	'none'
Wlimit	real	+	input gross weight limit	0.
SET_UL	c*16	+	useful load	'pay+miss'
Wpay	real	+	input takeoff payload weight W_{pay} (Units_pay)	0.
Npass	int	+	number of passengers N_{pass}	0
Wpay_cargo	real	+	cargo W_{cargo} (Units_pay)	0.
Wpay_extload	real	+	external load $W_{\text{ext-load}}$ (Units_pay)	0.
Wpay_ammo	real	+	ammunition W_{ammo} (Units_pay)	0.
Wpay_weapons	real	+	weapons W_{weapons} (Units_pay)	0.
SET_pay	c*16	+	payload changes	'delta'
		+	fuel tank systems	
FIX_missfuel(ntankmax)	int	+	mission fuel weight (0 calculated, 1 fixed)	0
dFuel(ntankmax)	real	+	fuel weight or energy increment	0.
fFuel(ntankmax)	real	+	fuel capacity factor	1.
SET_auxtank(ntankmax)	int	+	auxiliary fuel tanks (1 adjust Nauxtank, 2 only increase, 3 increase at start and drop, 0 no change)	1
mauxtank(ntankmax)	int	+	tank size changed (–1 first, –2 first size already used, m for m-th size)	-1
dNauxtank(ntankmax)	int	+	number tanks added or dropped	1
Nauxtank(nauxtankmax,ntankmax)	int	+	number of auxiliary fuel tanks N_{auxtank} (each aux tank size)	

		+	fixed useful load	
SET_foldkit	int	+	folding kit on aircraft (0 none, 1 present)	1
SET_reserve	int	+	fuel reserve (1 fraction mission fuel, 2 fraction fuel capacity, 3 only mission segments)	1
fReserve	real	+	fuel reserve fraction $f_{\rm res}$	0.
		+	split segments	
dist_inc	real	+	distance increment (Units_dist)	100.
time_inc	real	+	time increment (Units_time)	30.
alt_inc	real	+	altitude increment (Units_alt)	2000.
VTO_inc	real	+	takeoff velocity increment	10.
hTO_inc	real	+	takeoff height increment	10.
DESIGN_engine	int	+	design mission for power (1 to use for engine sizing)	1
DESIGN_jet	int	+	design mission for jet thrust (1 to use for jet group sizing)	1
DESIGN_charge	int	+	design mission for charge power (1 to use for charge group sizing)	1
DESIGN_GW	int	+	design mission for DGW (1 to use for DGW calculation)	1
DESIGN_xmsn	int	+	design mission for transmission (1 to use for transmission sizing)	1
DESIGN_tank	int	+	design mission for fuel tank (1 to use for fuel tank capacity)	1
DESIGN_thrust	int	+	design mission for antitorque or aux thrust (1 to use for rotor sizing)	1

---

label is short description for output
sizing mission: use all parameters
- fixed gross weight missions not used to determine DGW (set DESIGN_GW=0)
- mission segment not used to size engine or rotor if power margin fixed (max GW, max effort, or trim)
- mission segment not used to size transmission if zero torque margin (max GW, max effort, or trim)
- mission segment not used for sizing if set MissSeg%SizeZZZ=0

off design mission: not use DESIGN_xx
SET_GW, SET_UL values determine which input parameters used

SET_GW, set mission takeoff gross weight W_G:
- 'DGW' = design gross weight W_D; input (FIX_DGW) or calculated
- 'SDGW' = structural design gross weight W_{SD} (may depend on DGW)
- 'WMTO' = maximum takeoff gross weight W_{MTO} (may depend on DGW)
- 'f(DGW)' = function DGW: fGW*W_D+dGW
- 'f(SDGW)' = function SDGW: fGW*W_{SD}+dGW
- 'f(WMTO)' = function WMTO: fGW*W_{MTO}+dGW

'input' = input (use GW)
'maxP', 'max' = maximum GW for power required equal specified power: $P_{req} = \text{fPav} P_{av} + \text{dPav}$
at mission segment MaxGW, minimum gross weight of designated segments
$\min((f P_{avPG} + d) - P_{reqPG}) = 0$, over all propulsion groups
'maxQ' = maximum GW for transmission torque equal limit: zero torque margin
at mission segment MaxGW, minimum gross weight of designated segments
$\min(P_{\text{limit}} - P_{req}) = 0$, over all propulsion groups, engine groups, and rotors
'maxPQ', 'maxQP' = maximum GW for power required equal specified power and transmission torque equal limit
at mission segment MaxGW, minimum gross weight of designated segments
most restrictive of power and torque margins
'maxJ' = maximum GW for jet thrust required equal specified thrust: $T_{req} = \text{fTav} T_{av} + \text{dTav}$
at mission segment MaxGW, minimum gross weight of designated segments
$\min((f T_{avJG} + d) - T_{reqJG}) = 0$, over all jet groups
'maxPJ', 'maxQJ', 'maxPQJ' = maximum GW for most restrictive of power, torque, and thrust margins
'pay+fuel' = input payload and fuel weights; gross weight fallout
'pay+miss' = input payload, fuel weight from mission; gross weight fallout

SET_Wlimit: weight limit for SET_GW='max'
'none' = no limit
'f(DGW)' = function DGW: fGW*W_D+dGW
'f(SDGW)' = function SDGW: fGW*W_{SD}+dGW
'f(WMTO)' = function WMTO: fGW*W_{MTO}+dGW
'input' = input (use Wlimit)

SET_UL, set useful load:
'pay' = input payload weight (Wpay); fuel weight fallout
'fuel' = input fuel weight (dFuel, fFuel, Nauxtank); initial payload weight fallout
'miss' = fuel weight from mission; initial payload weight fallout
'pay+fuel' = input payload and fuel weights; gross weight fallout
'pay+miss' = input payload, fuel weight from mission; gross weight fallout

if SET_GW='pay+fuel' or 'pay+miss', assume SET_UL same (actual SET_UL ignored)
FIX_missfuel only used for SET_UL='miss' or 'pay+miss', with more than one fuel tank system

SET_pay, set payload changes: mission segment payload (use of MissSeg%xWpay)
'none' = no changes
'input' = value; payload = xWpay (not use Wpay)
'delta' = increment; payload = (initial payload weight)+(xWPay–xWpay(seg1))
'scale' = factor; payload = (initial payload weight)*(xWPay/xWpay(seg1))
when SET_GW='max' and SET_UL='fuel' or 'miss' (so payload is fallout), payload (from SET_pay and xWpay) must not be zero at the maximum GW segments

payload: only Wpay and xWpay used if SET_Wpayload = no details

input fuel weight: $W_{\text{fuel}} = \min(\text{dFuel}+\text{fFuel}*W_{\text{fuel-cap}}, W_{\text{fuel-cap}}) + \sum \text{Nauxtank}*W_{\text{aux-cap}}$
for fallout fuel weight, this is the initial value for the mission iteration

auxiliary fuel tanks:
SET_auxtank options: fixed; or adjust Nauxtank for each segment; or
increase at mission start, then constant; or increase at start, then drop
for input fuel (SET_UL = 'fuel' or 'pay+fuel'), start with input Nauxtank, then drop
for mission fuel (SET_UL = 'miss' or 'pay+miss'), fixed W_{fuel} or E_{fuel} at start
for fallout (SET_UL = 'pay'), adjust W_{fuel} with change in Nauxtank (fixed $W_G - W_{\text{pay}} = W_O + W_{\text{fuel}}$)
for all SET_UL, adjust W_O with change in Nauxtank
fuel tank design mission: Nauxtank=0, allow W_{fuel} or E_{fuel} to exceed tank capacity

SET_reserve: maximum of fuel for designated reserve mission segments
and fraction of fuel ($f_{\text{res}} W_{\text{burn}}$ or $f_{\text{res}} E_{\text{burn}}$) or fraction of fuel capacity ($f_{\text{res}} W_{\text{fuel-cap}}$ or $f_{\text{res}} E_{\text{fuel-cap}}$)

		+	Segment integration	
KIND_SegInt	int	+	method (0 segment start, 1 segment midpoint, 2 trapezoidal)	1
		+	Mission iteration (supersede Solution input if nonzero)	
relax_miss	real	+	relaxation factor (mission fuel)	0.
relax_range	real	+	relaxation factor (range credit)	0.
relax_gw	real	+	relaxation factor (max takeoff GW)	0.
toler_miss	real	+	tolerance (fraction reference)	0.
trace_miss	int	+	trace iteration (0 for none)	0

		+	Mission Segments	
nSeg	int	+	number of mission segments (maximum nsegmax)	1

---

input all mission segments as arrays in single mission namelist

---

Chapter 13

Structure: MissSeg

Variable	Type		Description	Default
		+	Segment definition	
label_seg	c*8	+	label	' '
kind	c*12	+	kind	'dist'
dist	real	+	distance D (Units_dist)	0.
time	real	+	time T (Units_time)	0.
		+	segment	
reserve	int	+	reserve (0 for not)	0
adjust	int	+	adjustable for flexible mission (0 for not)	0
range_credit	int	+	segment number for range credit (0 for no reassignment)	0
ignore	int	+	ignore segment (0 for not)	0
copy	int	+	copy segment (source segment number)	0
split	int	+	split segment (number segments; –1 calculated; 0 for not split)	0
SET_tank(ntankmax)	int	+	segment fuel use or replace	0
dTank(ntankmax)	real	+	fuel increment	0.
fTank(ntankmax)	real	+	fuel factor	1.
SET_refuel(ntankmax)	int	+	refuel (0 not, 1 fill all tanks, 2/8 add fuel, 3/9 drop fuel, 4-5 fill/add below rWfuel, 6-7 fill/add below mWfuel)	0
xWfuel(ntankmax)	real	+	fuel weight or energy change	0.
rWfuel(ntankmax)	real	+	threshold fraction	0.
mWfuel(ntankmax)	real	+	threshold weight or energy	0.
		+	gross weight	
MaxGW	int	+	maximize gross weight (0 not)	0
dPav(npropmax)	real	+	power increment, each propulsion group	0.
fPav(npropmax)	real	+	power factor, each propulsion group	1.
dTav(njetmax)	real	+	thrust increment, each jet group	0.
fTav(njetmax)	real	+	thrust factor, each jet group	1.
		+	useful load	
xWpay	real	+	payload weight change (Units_pay)	0.
xNpass	int	+	number of passengers increment $\delta N_{\rm pass}$	0

		+	fixed useful load	
dWcrew	real	+	crew weight increment	0.
dNcrew	int	+	number of crew increment $\delta N_{\rm crew}$	0
dWoful(10)	real	+	other fixed useful load increment (nWoful categories)	0.
dWequip	real	+	equipment weight increment	0.
dNcrew_seat	int	+	crew seat increment $\delta N_{\rm crew-seat}$	0
dNpass_seat	int	+	passenger seat increment $\delta N_{\rm pass-seat}$	0
		+	kits on aircraft (0 none, 1 present)	
SET_extkit(nwingmax)	int	+	wing extension kit	1
SET_wingkit(nwingmax)	int	+	wing kit	1
SET_otherkit	int	+	other kit	0
SET_alt	int	+	altitude at start of segment (0 input, 1 from previous segment, 2 from kSeg_alt)	0
kSeg_alt	int	+	source of altitude	0
		+	design mission (0 to not use segment for sizing)	
SizeEngine	int	+	power	1
SizeJet	int	+	jet thrust	1
SizeCharge	int	+	charger power	1
SizeGW	int	+	DGW	1
SizeXmsn	int	+	transmission	1
SizeThrust	int	+	antitorque or aux thrust	1

---

segment kind

kind='taxi', 'idle': taxi/warm-up mission segment (use time)
kind='dist': fly segment for specified distance (use dist)
kind='time': fly segment for specified time (use time)
kind='hold', 'loiter': fly segment for specified time (use time), fuel burned but no distance added to range
kind='climb': climb/descend from present altitude to next segment altitude
kind='spiral': climb/descend from present altitude to next segment altitude, fuel burned but no dist added to range
kind='fuel': use or replace specified fuel amount, calculate time and distance
kind='burn', 'charge': use or replace specified fuel amount, calculate time but no distance added to range
kind='takeoff', 'TO': takeoff distance calculation

only one of reserve, adjust, range_credit designations for each segment
reserve: time and distance not included in block time and range

range credit: to facilitate specification of range
range calculated for this segment credited to segment = range_credit
range_credit segment must be kind='dist', specified distance is for group of segments
actual distance flown in range_credit segment is specified dist less distances from other segments
if credit to earlier segment, iteration required
adjustable: for SET_UL not 'miss', can adjust one or more segments
if more than one segment adjusted, must be all kind='dist' or all kind='time'/'hold'
adjust time or distance based on fuel burn (proportional to initial values)

split segment: number specified, or calculated from MissParam%dest_inc, time_inc, alt_inc

ignore segment: removed from input; segments using MaxGW, range_credit, FltCond%KIND_source can not be ignored

SET_tank: segment fuel use or replace for kind='fuel' or 'burn'; distance and time calculated
SET_tank = 0: no requirement
SET_tank = 1: target dTank+fTank*$W_{\rm fuel-cap}$ or dTank+fTank*$E_{\rm fuel-cap}$
SET_tank = 2: target dTank+fTank*$W_{\rm fuel}$ or dTank+fTank*$E_{\rm fuel}$
SET_tank = 3: increment dTank+fTank*$W_{\rm fuel-cap}$ or dTank+fTank*$E_{\rm fuel-cap}$
SET_tank = 4: increment dTank+fTank*$W_{\rm fuel}$ or dTank+fTank*$E_{\rm fuel}$
charge if $\dot{E} < 0$ (not based on keyword, increment always positive)
target limited by capacity, if target already achieved then no requirement
increment limited by current fuel (use) or capacity minus current fuel (replace)

SET_refuel, refuel: change at start of segment; weight or energy; no contribution to distance or time
SET_refuel = 1: fill all tanks (including any auxiliary tanks installed)
SET_refuel = 2: add fuel xWfuel
SET_refuel = 3: drop fuel xWfuel
SET_refuel = 4: if below fraction rWfuel of fuel capacity (including auxiliary tanks), fill all tanks
SET_refuel = 5: if below fraction rWfuel of fuel capacity (including auxiliary tanks), add xWfuel
SET_refuel = 6: if below mWfuel, fill all tanks
SET_refuel = 7: if below mWfuel, add xWfuel
SET_refuel = 8: add fraction rWfuel of fuel capacity (including auxiliary tanks)
SET_refuel = 9: drop fraction rWfuel of fuel capacity (including auxiliary tanks)
added fuel limited by capacity (unless sizing fuel tank); not used for first segment
xWfuel positive (add or drop determined by SET_refuel)

maximize gross weight: MaxGW designate segments if SET_GW='maxP' or 'maxQ' or 'maxPQ'

climb/descend or spiral segment: end altitude is that of next segment; last segment kind can not be climb or spiral
begin altitude is that input for this segment (SET_alt=0), or altitude of previous segment (SET_alt=1),

payload: only Wpay and xWpay used if SET_Wpayload = no details
xNpass is change from MissParam%Npass
crew: only dWcrew used if SET_Wcrew = no details
equipment: dNcrew_seat and dNpass_seat require non-zero weight per seat

		+	Takeoff distance calculation	
SET_takeoff	c*12	+	takeoff segment kind	'none'
Vkts_takeoff	real	+	ground speed or climb speed (knots, CAS)	0.
climb_takeoff	real	+	climb angle relative ground γ (deg)	0.
height_takeoff	real	+	height during climb h (ft or m)	0.
slope_ground	real	+	slope of ground γ_G (+ for uphill; deg)	0.
friction	real	+	friction coefficient μ	0.04
t_decision	real	+	decision delay after engine failure t_1 (sec)	1.5
t_rotation	real	+	rotation time t_R (sec)	2.0
nz_transition	real	+	transition load factor n_{TR}	1.2

takeoff distance calculation: set of consecutive kind='takeoff' segments
first segment identified by SET_takeoff='start' ($V = 0$)
last segment if next segment is not kind='takeoff', or is SET_takeoff='start'
takeoff segment kind
SET_takeoff='start', 'ground run' (keyword = ground or run), 'engine fail' (keyword = eng or fail)
SET_takeoff='liftoff', 'rotation', 'transition', 'climb', 'brake'
each segment requires appropriate configuration, trim option, max effort specification
not use dist, time, reserve, adjust, range_credit, SET_refuel, MaxGW, SET_alt
max_var='alt' not allowed in maximum effort
velocity specification (SET_vel) and HAGL superseded; SET_turn=SET_pullup=0
can split segment (except start, rotation, transition): split height for climb, velocity for others
splitting liftoff or engine failure segment produces additional ground run segments
separate definition of multiple ground run, climb, brake segments allows configuration variations

define takeoff profile in terms of velocities
- integrate acceleration vs velocity to obtain time and distance
- segments correspond to ends of integration intervals
- analysis checks for consistency of input velocity and calculated acceleration
- analysis checks for consistency of input height and input/calculated climb angle

takeoff distance definition: includes SET_takeoff='liftoff' segment
- order: start, ground run, engine failure, ground run, liftoff, rotation, transition, climb
- only one liftoff; only one engine failure, rotation, transition (or none)
- engine failure before liftoff; all ground run before liftoff, all climb after liftoff

accelerate-stop distance definition: does not have SET_takeoff='liftoff' segment
- order: start, ground run, engine failure, brake
- only one engine failure (or none)

engine failure segment (if present) identifies point for decision delay
- until t_decision after engine failure segment, use engine rating, fPower, fraction of engine failure segment
- so engine failure segment corresponds to conditions before failure

number of inoperative engines specified by nEngInop for each segment
- if engine failure segment present, nEngInop specification must be consistent

---

Chapter 14

Structure: FltState

Variable	Type		Description	Default
		+	Flight State	
		+	Specification	
SET_max	int	+	maximum effort performance (maximum 2, 0 to analyze specified condition)	0
max_quant(2)	c*12	+	quantity	' '
max_var(2)	c*12	+	variable	' '
max_limit(2)	int	+	switch quantity if exceed limit (0 not, 1 power margin, 2 torque margin, 3 both)	0
max_Vlimit(2)	int	+	velocity limited by V_{NE} (0 not)	0
fVel(2)	real	+	flight speed factor	1.
SET_vel	c*12	+	flight speed	'general'
Vkts	real	+	horizontal velocity V_h (TAS or CAS or IAS, Units_vel)	0.
Mach	real	+	horizontal velocity M (Mach number)	0.
ROC	real	+	vertical rate of climb V_c (Units_ROC)	0.
climb	real	+	climb angle θ_V (deg)	0.
side	real	+	sideslip angle ψ_V (deg)	0.
		+	aircraft motion	
SET_pitch	int	+	pitch motion specification (0 Aircraft value, 1 FltState input)	1
SET_roll	int	+	roll motion specification (0 Aircraft value, 1 FltState input)	1
pitch	real	+	pitch θ_F	0.
roll	real	+	roll ϕ_F	0.
SET_turn	int	+	turn specification (0 zero, 1 turn rate, 2 load factor, 3 bank angle)	0
rate_turn	real	+	turn rate $\dot{\psi}_F$ (deg/sec)	0.
nz_turn	real	+	load factor n (g)	1.
bank_turn	real	+	bank angle ϕ_F (deg)	0.
SET_pullup	int	+	pullup specification (0 zero, 1 pitch rate, 2 load factor)	0
rate_pullup	real	+	pitch rate $\dot{\theta}_F$ (deg/sec)	0.
nz_pullup	real	+	load factor n (g)	1.
SET_acc	int	+	linear acceleration specification (0 zero, 1 acceleration, 2 load factor)	0
ax_linear	real	+	x-acceleration a_{ACx} (ft/sec^2 or m/sec^2)	0.

ay_linear	real	+	y-acceleration a_{ACy} (ft/sec^2 or m/sec^2)	0.
az_linear	real	+	z-acceleration a_{ACz} (ft/sec^2 or m/sec^2)	0.
nx_linear	real	+	x-load factor increment n_{Lx} (g)	0.
ny_linear	real	+	y-load factor increment n_{Ly} (g)	0.
nz_linear	real	+	z-load factor increment n_{Lz} (g)	0.
altitude	real	+	altitude h (Units_alt)	0.
SET_atmos	c*12	+	atmosphere specification	'std'
temp	real	+	temperature τ (Units_temp)	
dtemp	real	+	temperature increment ΔT (Units_temp)	0.
density	real	+	density ρ	
csound	real	+	speed of sound c_s	
viscosity	real	+	viscosity μ	
SET_wind	int	+	wind specification (0 none, 1 headwind, 2 tailwind)	0
dWind	real	+	wind increment, knots (dWind+fWind*altitude)	0.
fWind	real	+	wind gradient, knots (dWind+fWind*altitude)	0.
SET_GE	int	+	ground effect (0 OGE, 1 IGE)	0
HAGL	real	+	height of landing gear above ground level h_{LG}	999.
STATE_LG	c*12	+	landing gear state	'default'
STATE_control	int	+	aircraft control state	1
SET_control(ncontmax)	int	+	control specification (0 Aircraft value, 1 FltState input)	1
SET_coll	int	+	collective stick	1
SET_latcyc	int	+	lateral cyclic stick	1
SET_lngcyc	int	+	longitudinal cyclic stick	1
SET_pedal	int	+	pedal	1
SET_tilt	int	+	tilt (0 Aircraft value, 1 FltState input, 2 Aircraft conversion schedule)	1
control(ncontmax)	real	+	aircraft controls	
coll	real	+	collective stick c_{AC0}	0.
latcyc	real	+	lateral cyclic stick c_{ACc}	0.
lngcyc	real	+	longitudinal cyclic stick c_{ACs}	0.
pedal	real	+	pedal c_{ACp}	0.
tilt	real	+	tilt α_{tilt}	0.
SET_comp_control	int	+	use component control (0 for $c = Tc_{AC}$; 1 for $c = Tc_{AC} + c_0$)	1
SET_cg	int	+	center of gravity specification (0 baseline plus increment, 1 input)	0
dSLcg	real	+	stationline	0.

dBLcg	real	+	buttline	0.
dWLcg	real	+	waterline	0.
		+	Specification, each propulsion group	
SET_Vtip(npropmax)	c*12	+	rotor tip speed specification	'hover'
Vtip(npropmax)	real	+	tip speed	
Mtip(npropmax)	real	+	tip Mach number M_{tip}	
mu_Vtip(npropmax)	real	+	tip speed from μ	
Mat_Vtip(npropmax)	real	+	tip speed from M_{at}	
Nrotor(npropmax)	real	+	rotor speed (rpm)	
Nspec(npropmax)	real	+	engine speed (rpm)	
STATE_gear(npropmax)	int	+	drive system state	1
rating_ds(npropmax)	c*12	+	drive system rating	' '
fTorque(npropmax)	real	+	fraction of rated drive system torque limit f_Q (0. to 1.+)	1.
SET_Plimit(npropmax)	int	+	drive system limit (0 not applied to power available)	1
SET_Qlimit_rs(npropmax)	int	+	rotor shaft limit (0 not used for torque margin)	1
SET_Pmargin(npropmax)	int	+	power and torque margin (0 not used for maximum effort)	1
dPacc(npropmax)	real	+	accessory power increment dP_{acc}	0.
		+	Specification, each engine group	
rating(nengmax)	c*12	+	engine rating	'MCP'
fPower(nengmax)	real	+	fraction of rated engine power available f_P (0. to 1.+)	1.
nEngInop(nengmax)	int	+	number of inoperative engines N_{inop}	0
SET_Preq(nengmax)	int	+	power required (1 distributed, 2 fixed A, 3 fixed AP_{av}, 4 fixed AP_{eng})	1
STATE_IRS(nengmax)	int	+	IR suppressor system state (0 off, hot exhaust; 1 on, suppressed exhaust)	0
		+	Specification, each jet group	
rating_jet(njetmax)	c*12	+	jet rating	'MCT'
fThrust(njetmax)	real	+	fraction of rated jet thrust available f_T (0. to 1.+)	1.
nJetInop(njetmax)	int	+	number of inoperative jets N_{inop}	0
SET_Jreq(njetmax)	int	+	thrust required (1 from component, 2 fixed A, 3 fixed AT_{av}, 4 fixed AT_{jet})	2
STATE_IRS_jet(njetmax)	int	+	IR suppressor system state (0 off, hot exhaust; 1 on, suppressed exhaust)	0
		+	Specification, each charge group	
rating_charge(nchrgmax)	c*12	+	charger rating	'MCP'
fCharge(nchrgmax)	real	+	fraction of rated charger power available f_C (0. to 1.+)	1.
nChrgInop(nchrgmax)	int	+	number of inoperative chargers N_{inop}	0
SET_Creq(nchrgmax)	int	+	power required (2 fixed A, 4 fixed AP_{chrg})	2

dPeq(ntankmax)	real	+	Equipment power increment $dP_{\rm eq}$, each fuel tank	0.
		+	Specification, each fuel tank (battery)	
ffade(ntankmax)	real	+	battery capacity fade factor	1.
Tcell(ntankmax)	real	+	cell temperature (deg C)	20.
fcurrent(ntankmax)	real	+	maximum current (fraction x_{mbd} or $x_{CC{\rm max}}$)	1.
		+	Specification, each rotor	
STOP_rotor(nrotormax)	int	+	rotor stop/stow (0 not, 1 stop, 2 stop and stow, 3 stop as wing)	0
STATE_deice	int	+	Deice system state (0 off)	0
		+	Performance	
DoQ_pay	real	+	payload forward flight drag increment D/q (Units_drag)	0.
fDoQ_pay	real	+	payload drag increment scaling with weight $\Delta(D/q)/W_{\rm pay}$ (Units_drag, Units_pay)	0.
DoQV_pay	real	+	payload vertical drag increment D/q (Units_drag)	0.
		+	Rotor (nonzero to supersede rotor model)	
Ki(nrotormax)	real	+	induced power factor κ	0.
cdo(nrotormax)	real	+	profile power mean c_d	0.
MODEL_Ftpp(nrotormax)	int	+	inplane forces, tip-path plane axes (1 neglect, 2 blade-element theory)	0
MODEL_Fpro(nrotormax)	int	+	inplane forces, profile (1 simplified, 2 blade element theory, 3 neglect)	0
KIND_control(nrotormax)	int	+	control mode (1 thrust and TPP, 2 thrust and NFP, 3 pitch and TPP, 4 pitch and NFP)	0
		+	Trim solution	
STATE_trim	c*12	+	aircraft trim state (match IDENT_trim, 'none' for no trim)	'none'
trim_target(mtrimmax)	real	+	trim quantity targets	
		+	Iterations (supersede Solution input if nonzero)	
		+	relaxation factor	
relax_rotor	real	+	all rotors	0.
relax_trim	real	+	trim	0.
relax_fly(2)	real	+	maximum effort	0.
relax_maxgw	real	+	maximum gross weight	0.
		+	tolerance (fraction reference)	
toler_rotor	real	+	all rotors	0.
toler_trim	real	+	trim	0.
toler_fly(2)	real	+	maximum effort	0.
toler_maxgw	real	+	maximum gross weight	0.
		+	reinitialize aircraft controls (0 no, 1 force retrim)	
init_trim	int	+	trim	0

init_fly	int	+	maximum effort	0
		+	variable perturbation amplitude (fraction reference, 0. for no limit)	
perturb_trim	real	+	trim	0.
perturb_fly(2)	real	+	maximum effort	0.
perturb_maxgw	real	+	maximum gross weight	0.
		+	maximum derivative amplitude (0. for no limit)	
maxderiv_fly(2)	real	+	maximum effort	0.
maxderiv_maxgw	real	+	maximum gross weight	0.
		+	maximum increment fraction (0. for no limit)	
maxinc_fly(2)	real	+	maximum effort	0.
maxinc_maxgw	real	+	maximum gross weight	0.
		+	solution method	
method_flymax(2)	int	+	maximum effort	0
		+	trace iteration (0 for none)	
trace_rotor	int	+	all rotors	0
trace_trim	int	+	trim (2 for component controls)	0
trace_fly(2)	int	+	maximum effort	0
trace_maxgw	int	+	maximum gross weight	0

---

maximum effort performance: one or two quantity/variable identified; first is inner loop
two variables must be unique
two variables can be identified for same maximized quantity (endurance, range, climb)
quantity identified by max_quant maximized for endurance, range, climb, or ceiling; otherwise driven to zero

ROC or altitude can be outer loop quantity only if it is also inner loop variable
fVel is only used for max_var='speed' or 'ROC'
ceiling calculation should use 'Pmargin'/'alt' as inner loop, 'power'/'speed' as outer loop
best range calculation often requires maxinc_fly=0.1 for convergence
ROC for zero power margin initialized based on level flight power margin if input ROC=0
max_quant='rotor(s) n' uses Rotor%CTs_steady, max_quant='rotor(t) n' uses Rotor%CTs_tran
max_quant='rotor(e) n' uses equation for rotor thrust capability (Rotor%K0_limit and Rotor%K1_limit)
if energy burned (not weight) or multiple fuels, use equivalent fuel flow obtained from weighted energy flow
max_var='Vtip' or 'Nspec' requires FltAircraft%SET_Vtip='input'
if trailing "n" is absent, use first component (n=1)

max_limit: switch quantity to power and/or torque margin if margin negative; useful for best range

description	max_quant	
endurance	'end'	maximum (1/fuelflow)
range (high side)	'range'	0.99 maximum (V/fuelflow)
range	'range(100)'	maximum (V/fuelflow)
range (low side)	'range(low)'	0.99 maximum (V/fuelflow), low side
range (high side), ground speed	'rangeVg'	0.99 maximum (V_g/fuelflow)
range, ground speed	'range(100)Vg'	maximum (V_g/fuelflow)
range (low side), ground speed	'range(low)Vg'	0.99 maximum (V_g/fuelflow), low side
climb or descent rate	'climb', 'ROC'	maximum (ROC)
climb rate (power)	'power'	maximum (1/Power)
climb or descent angle	'angle'	maximum (ROC/V)
climb angle (power)	'power/V'	maximum (V/Power)
ceiling	'alt'	maximum (altitude)
power margin	'P margin'	$\min(P_{av} - P_{req}) = 0$ (all propulsion groups)
torque margin	'Q margin',	$\min(Q_{\rm limit} - Q_{req}) = 0$ (all limits)
jet thrust margin	'J margin',	$\min(T_{av} - T_{req}) = 0$ (all jet groups)
power and torque margin	'PQ margin',	most restrictive
power and thrust margin	'PJ margin',	most restrictive
torque and thrust margin	'QJ margin',	most restrictive
power, torque, thrust margin	'PQJ margin',	most restrictive
battery power margin	'B margin'	$\min(P_{\max} - \lvert\dot{E}_{\rm batt}\rvert) = 0$ (all fuel tanks)
rotor thrust margin	'rotor(t) n'	$(C_T/\sigma)_{\max} - \lvert C_T/\sigma\rvert = 0$ (transient)
rotor thrust margin	'rotor(s) n'	$(C_T/\sigma)_{\max} - \lvert C_T/\sigma\rvert = 0$ (sustained)
rotor thrust margin	'rotor(e) n'	$(C_T/\sigma)_{\max} - \lvert C_T/\sigma\rvert = 0$ (equation)
wing lift margin	'stall n'	$C_{L\max} - C_L = 0$

description	max_var	
horizontal velocity	'speed'	times fVel
vertical rate of climb	'ROC'	times fVel
aircraft velocity	'side'	sideslip angle
altitude	'alt'	
aircraft angular rate	'pullup', 'turn'	Euler angle rates
aircraft acceleration	'xacc', 'yacc', 'zacc'	linear, airframe axes
aircraft acceleration	'xaccI', 'yaccI', 'zaccI'	linear, inertial axes
aircraft acceleration	'xaccG', 'yaccG', 'zaccG'	linear, ground axes
aircraft control	match IDENT_control	
aircraft orientation	'pitch', 'roll'	body axes relative inertial axes
propulsion group tip speed	'Vtip n'	
propulsion group engine speed	'Nspec n'	

SET_vel, velocity specification:
'general' = general (use Vkts=horizontal, ROC, side)
'hover' = hover (zero velocity)
'vert' = hover or VROC (use ROC; Vkts=0, climb=0/+90/–90)
'right' = right sideward (use Vkts, ROC; side=90)
'left' = left sideward (use Vkts, ROC; side=–90)
'rear' = rearward (use Vkts, ROC, side=180)
'Vfwd' = general (use Vkts=forward velocity, ROC, side)
'Vmag' = general (use Vkts=velocity magnitude, ROC, side)
'climb' = general (use Vkts=velocity magnitude, climb, side)
'VNE' = never-exceed speed
'+Mach' = use Mach not Vkts
'+CAS' = Vkts is CAS not TAS
'+IAS' = Vkts is IAS not TAS
velocities: forward $V_f = V_h \cos(\text{side})$, side $V_s = V_h \sin(\text{side})$, climb $V_c = V_h \tan(\text{climb})$

aircraft motion:

orientation velocity relative inertial axes defined by climb and sideslip angles (θ_V, ψ_V)
- sideslip positive aircraft moving to right, climb positive aircraft moving up
- specify horizontal velocity, vertical rate of climb, and sideslip angle

orientation body relative inertial axes defined by Euler angles, yaw/pitch/roll (ψ_F, θ_F, ϕ_F)
- yaw positive to right, pitch positive nose up, roll positive to right

SET_PITCH and SET_roll, pitch and roll motion specification:
- Aircraft values (perhaps function speed) or flight state input
- initial values specified if motion is trim variable; otherwise fixed for flight state

SET_turn, bank angle and load factor in turn: use turn rate, load factor, or bank angle
- $\tan(\text{roll}) = \sqrt{n^2 - 1}$ = (turn)V/g; calculated using input Vkts for flight speed

SET_pullup, load factor in pullup: use pullup rate or load factor
- $n = 1 +$ (pullup)V/g; calculated using input Vkts for flight speed

SET_acc, linear acceleration: use acceleration or load factor

SET_atmos, atmosphere specification:
- 'std' = standard day at specified altitude (use altitude)
- 'polar' = polar day at specified altitude (use altitude)
- 'trop' = tropical day at specified altitude (use altitude)
- 'hot' = hot day at specified altitude (use altitude)
- 'xxx+dtemp' = specified altitude, plus temperature increment (use altitude, dtemp)
- 'xxx+temp' = specified altitude, and specified temperature (use altitude, temp)
- 'hot+table' = hot day table at specified altitude (use altitude)
- 'dens' = input density and temperature (use density, temp)
- 'input' = input density, speed of sound, and viscosity (use density, csound, viscosity)
- 'notair' = input, not air on earth (use density, csound, viscosity)

SET_GE: use HAGL; out-of-ground-effect (OGE) if rotor more than 1.5Diameter above ground
- height rotor = landing gear above ground + hub above landing gear = HAGL + (WL_hub–WL_gear+d_gear)

STATE_LG: 'default' (based on retraction speed), 'extend', 'retract' (keyword = def, ext, ret)

STATE_control, aircraft control state: identifies control matrix
STATE_control=0 to use conversion schedule, STATE_control=n (1 to nstate_control) to use state#n
SET_control, control specification: Aircraft values (perhaps function speed) or flight state input
coll/latcyc/lngcyc/pedal/tilt specification and values put in SET_control and control, based on IDENT_control
initial values specified if control is trim variable; otherwise fixed for flight state
SET_control=0 to use Aircraft%cont and Aircraft%Vcont; 1 to use FltState%control
SET_tilt: 0 to use Aircraft%tilt and Aircraft%Vtilt; 1 to use FltState%tilt
2 to use conversion speeds Aircraft%Vconv_hover and Aircraft%Vconv_cruise

SET_cg, center of gravity position: input for this flight state; or
baseline cg position plus shift due to nacelle tilt, plus input cg increment

tip speed, engine, transmission: for each propulsion group
SET_Vtip, primary rotor tip speed: for primary rotor of propulsion group
'input' = use input Vtip for this flight state
'Mtip' = use input Mtip for this flight state
'Nrotor' = use input Nrotor (rpm) for this flight state
'ref' = use Vtip_ref (for drive state STATE_gear)
'speed' = use default Vtip(speed)
'conv' = use conversion schedule (Vtip_hover or Vtip_cruise)
'hover' = use default Vtip_hover = Vtip_ref(1)
'cruise', 'man', 'OEI', 'xmsn' = use default Vtip_cruise, Vtip_man, Vtip_oei, Vtip_xmsn
'mu' = use tip speed from μ (mu_Vtip)
'Mat' = use tip speed from M_{at} (Mat_Vtip)
'xxx+Mat' = for tip speed limited by M_{at} (Mat_Vtip)
'xxx+diam' = for variable diameter rotor, scale $V_{\rm tip}$ with radius ratio
without rotors, specify engine group speed by SET_Vtip='input' (use input Nspec) or 'ref'
STATE_gear, drive system state: identifies gear ratio set for multiple speed transmissions
state=0 to use conversion schedule, state=n (1 to nGear) to use gear ratio #n
drive system rating: match rating designation in propulsion group; blank for same as rating of first engine group
rating_ds='speed' to use schedule with speed
if Propulsion%nrate_ds≤ 1, drive system rating not used
fTorque reduces drive system torque limit (fTorque = 0. to 1.; > 1 is an acceptable input)
SET_Plimit: usually should not be applied for flight conditions and mission segments that size transmission

engine rating: match rating designation in engine model; e.g. 'ERP','MRP','IRP','MCP'
or rating='idle' or rating='takeoff'
fPower reduces engine group power available (fPower = $0.$ to $1.$; > 1 is an acceptable input)
the engine model gives the power available, accounting for installation losses and mechanical limits
then the power available is reduced by the factor fPower
next torque limits are applied (unless SET_Plimit=off), first engine shaft limit and then drive system limit
for SET_GW='maxP' or 'maxPQ' (flight condition or mission), the gross weight is determined
such that $P_{reqPG} = fP_{avPG} + d$
either fPower or fPav can be used to reduce the available power
with identical results, unless the engine group is operating at a torque limit
nEngInop, number inoperative engines: 1 for one engine inoperative (OEI), maximum nEngine

SET_Preq: distribution of propulsion group power required among engine groups
distributed (SET_Preq=1): P_{reqEG} from P_{reqPG}, proportional P_{eng}
except for rotor reaction drive, P_{reqEG} from power needed to supply reaction force
and for fuselage or wing flow control, P_{reqEG} from power needed to supply momentum flux
fixed options use engine group amplitude control variable A, for each operable engine
engine group that consumes shaft power (generator or compressor) only uses fixed option
engine group that produces no shaft power (converted to turbo jet or reaction drive) only uses fixed option
EngineGroup%SET_Power, fPsize defines power distribution for sizing

jet rating: match rating designation in jet model; or rating_jet='idle' or rating_jet='takeoff'
fThrust reduces jet group thrust available (fThrust = 0 to 1; > 1 is an acceptable input)
nJetInop, number inoperative jets: 1 for one jet inoperative (OEI), maximum nJet
SET_Jreq: fixed options use jet group amplitude control variable A, for each operable jet
from component (SET_Jreq=1): only for reaction drive or flow control, T_{reqJG} from required F_{Greq}

charger rating: match rating designation in charger model; or rating_charge='idle' or rating_charge='takeoff'
fCharge reduces charger group power available (fCharge = 0 to 1; > 1 is an acceptable input)
nChrgInop, number inoperative chargers: 1 for one charger inoperative (OEI), maximum nCharge
SET_Creq: use charge group amplitude control variable A, for each operable charger

STOP_rotor: only for stoppable rotor; if stopped, model sets KIND_control=1, MODEL_Ftpp=1, MODEL_Fpro=3

STATE_trim, aircraft trim state: match IDENT_trim, 'none' for no trim
identifies trim variables and quantities
ACTION='configuration' defines trim states with following identification:
IDENT_trim='free', 'symm', 'hover', 'thrust', 'rotor', 'windtunnel', 'power', 'ground', 'comp'
requirement for trim_target depends on designation of Aircraft%trim_quant

Chapter 15

Structure: Solution

Variable	Type		Description	Default
		+	Solution Procedures	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Rotor	
		+	convergence control	
niter_rotor(nrotormax)	int	+	maximum number of iterations	40
toler_rotor(nrotormax)	real	+	tolerance (deg)	.01
relax_rotor(nrotormax)	real	+	relaxation factor	.5
deriv_rotor(nrotormax)	int	+	derivative (1 first order, 2 second order)	1
maxinc_rotor(nrotormax)	real	+	maximum increment amplitude (0. for no limit)	4.
trace_rotor(nrotormax)	int	+	trace iteration (0 for none)	0
		+	Trim	
		+	convergence control	
niter_trim	int	+	maximum number of iterations	40
toler_trim	real	+	tolerance (fraction reference)	.001
relax_trim	real	+	relaxation factor	.5
		+	perturbation identification of derivative matrix	
deriv_trim	int	+	perturbation (1 first order, 2 second order)	1
mpid_trim	int	+	number of iterations between identification (0 for never recalculated)	0
perturb_trim	real	+	variable perturbation amplitude (fraction reference)	.002
init_trim	int	+	reinitialize aircraft controls in maximum effort iteration (0 no, 1 force retrim)	0
start_trim	int	+	initialize controls from solution of previous case (0 no)	0
trace_trim	int	+	trace iteration (0 for none, 2 for component controls)	0

start_trim=1: initialize FltAircraft%control from FltAircraft%control_trim of previous case
require INIT_input=INIT_data=2 or read solution file; and same missions and conditions as previous case
requirements not checked

		+	Maximum effort	
method_fly	int	+	method (1 secant, 2 false position)	1
method_flymax	int	+	maximization method (1 secant, 2 false position, 3 golden section search, 4 curve fit)	3
		+	convergence control	
niter_fly	int	+	maximum number of iterations	80
toler_fly	real	+	tolerance (fraction reference)	.002
relax_fly	real	+	relaxation factor	.5
perturb_fly	real	+	variable perturbation amplitude (fraction reference)	.05
maxderiv_fly	real	+	maximum derivative amplitude (0. for no limit)	0.
maxinc_fly	real	+	maximum increment fraction (0. for no limit)	0.
rfit_fly	real	+	extent of curve fit (fraction maximum)	.98
nfit_fly	int	+	order of curve fit (2 quadradic, 3 cubic)	3
init_fly	int	+	reinitialize aircraft controls (0 no, 1 force retrim)	0
trace_fly	int	+	trace iteration (0 for none)	0
		+	Maximum gross weight (flight condition or mission takeoff)	
method_maxgw	int	+	method (1 secant, 2 false position)	1
		+	convergence control	
niter_maxgw	int	+	maximum number of iterations	40
toler_maxgw	real	+	tolerance (fraction reference)	.002
relax_maxgw	real	+	relaxation factor	.5
perturb_maxgw	real	+	variable perturbation amplitude (fraction reference)	.02
maxderiv_maxgw	real	+	maximum derivative amplitude (0. for no limit)	0.
maxinc_maxgw	real	+	maximum increment fraction (0. for no limit)	0.
trace_maxgw	int	+	trace iteration (0 for none)	0
		+	Mission	
		+	convergence control	
niter_miss	int	+	maximum number of iterations	40
toler_miss	real	+	tolerance (fraction reference)	.01
relax_miss	real	+	relaxation factor (mission fuel)	1.
relax_range	real	+	relaxation factor (range credit)	1.
relax_gw	real	+	relaxation factor (max takeoff GW)	1.
trace_miss	int	+	trace iteration (0 for none)	0

		+	Size aircraft	
		+	convergence control	
niter_size	int	+	maximum number of iterations (performance loop)	40
niter_param	int	+	maximum number of iterations (parameter loop)	40
toler_size	real	+	tolerance (fraction reference)	.01
		+	relaxation factors	
relax_size	real	+	power or radius	1.
relax_DGW	real	+	gross weight	1.
relax_xmsn	real	+	drive system limit	1.
relax_wmto	real	+	WMTO and SDGW	1.
relax_tank	real	+	fuel tank capacity	1.
relax_thrust	real	+	rotor thrust	1.
		+	maximum increment fraction (0. for no limit)	
maxinc_size	real	+	power or radius	0.
maxinc_DGW	real	+	gross weight	0.
maxinc_xmsn	real	+	drive system limit	0.
maxinc_wmto	real	+	WMTO and SDGW	0.
maxinc_tank	real	+	fuel tank capacity	0.
maxinc_thrust	real	+	rotor thrust	0.
trace_size	int	+	trace iteration (0 for none, 2 for power)	0

with niter_param=1, parameter iteration is part of performance loop (can be faster than niter_param > 1)

		+	Case	
trace_case	int	+	trace operation (0 for none, 1 trace, 2 for all iterations)	1
trace_start	int	+	counter at start trace of iterations	0

use trace_case=2 to identify point at which analysis diverges
counter written if trace_case=1 or 2; trace of iterations suppressed until counter > trace_start
then turn on trace selectively for mission/segment/condition

		+	Flight condition and mission segment	
toler_check	real	+	check Preq, Qlimit, Wfuel (fraction reference)	.005
		+	Tolerance and perturbation scales	
KIND_Wscale	int	+	weight scale (1 design gross weight, 2 nominal C_T/σ)	1
KIND_Pscale	int	+	power scale (1 aircraft power, 2 derived from weight scale)	1
KIND_Lscale	int	+	length scale (1 rotor radius, 2 wing span, 3 fuselage length)	1
scaleRotor	int	+	rotor number	1
scaleWing	int	+	wing number	1
		+	External solution procedure (0 for internal)	
SETextsol_size	int	+	size iteration	0
SETextsol_miss	int	+	mission iteration	0
SETextsol_trim	int	+	trim iteration	0
SETextsol_rotor	int	+	rotor iteration	0

---

for external solution procedure (SETextsol $= 1$), suppress iteration and calculate residual
 the solution problem (such as size parameters, trim variables) must still be defined
 residuals (and error ratios) are in structures SizeParam, MissParam, FltAircraft, FltRotor
with external solution for maximum gross weight or maximum effort, there is no residual; do not specify internal iteration

---

Chapter 16

Structure: Cost

Variable	Type		Description	Default
		+	Cost	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Inflation	
MODEL_inf	int	+	model (1 only input factor, 2 CPI, 3 DoD)	3
year_inf	int	+	year for internal inflation factor	2018
inflation	real	+	inflation factor (per cent, relative 1994 or year_inf)	100.00
EXTRAP_inf	int	+	year beyond CPI/DoD table data (0 error, 1 extrapolate factor)	1

inflation: F_i multiplies airframe purchase price and maintenance cost
factor inflation always used, even with internal table
CPI or DoD table: $F_i = \text{inflation}\big(F_{\text{table}}(\text{year\_inf})/F_{\text{table}}(1994)\big)$
input factor: $F_i = \text{inflation}$ (relative 1994)
cost factors and rates include technology and inflation, correspond to year_inf

Variable	Type		Description	Default
		+	Cost	
MODEL_cost	int	+	model (0 none)	1
FuelPrice(ntankmax)	real	+	fuel price G_{fuel} ($/gallon or $/liter)	5.0
EnergyPrice(ntankmax)	real	+	energy price G_{energy} ($/MJ or $/kWh, Units_energy)	0.04
EnergyCredit(ntankmax)	real	+	credit for generated energy ($/MJ or $/kWh, Units_energy)	0.
Npass	int	+	number of passengers N_{pass}	100

equivalent energy price for fuel burned: $/MJ≅($/gal)/126.2 (based on 42.8 MJ/kg and 6.5 lb/gal of JP-4/JP-8)
EnergyCredit=0. if no credit for generated energy

		+	Direct Operating Cost	
BlockHours	real	+	available block hours per year B	3751.
NonFlightTime	real	+	non-flight time per trip T_{NF} (min)	12.
DepPeriod	real	+	depreciation period D (years)	15.
LoanPeriod	real	+	loan period L (years)	15.
IntRate	real	+	interest rate i (%)	8.
ResidValue	real	+	residual value V (%)	10.
Spares	real	+	spares per aircraft S (% purchase price)	25.
LoadFactor	real	+	passenger load factor (%)	75.
		+	Technology Factors	
TECH_cost_af	real	+	airframe χ_{AF}	0.87
TECH_cost_maint	real	+	maintenance $\chi_{\rm maint}$	1.0
TECH_cost_comp	real	+	components $\chi_{\rm comp}$	1.0
		+	CTM rotorcraft cost model	
		+	Purchase Price	
MODEL_aircraft	int	+	aircraft (1 rotorcraft, 2 turboprop airliner)	1
KIND_engine	int	+	engine (1 turbine, 2 piston)	1
fmotor	real	+	weighting factor for electric motor or generator	0.5
		+	airframe	
rComp	real	+	additional cost rate $r_{\rm comp}$ for composite construction ($/lb or $/kg)	0.
fWcomp_body	real	+	composite weight in body (fraction body weight)	0.
fWcomp_tail	real	+	composite weight in tail (fraction tail weight)	0.
fWcomp_pylon	real	+	composite weight in pylon (fraction pylon weight)	0.
fWcomp_wing	real	+	composite weight in wing (fraction wing weight)	0.
		+	systems (fixed useful load)	
rFCE	real	+	cost factor $r_{\rm FCE}$, flight control electronics ($/lb or $/kg)	10000.
rMEP	real	+	cost factor $r_{\rm MEP}$, mission equipment package ($/lb or $/kg)	10000.
rBatt	real	+	cost factor $r_{\rm batt}$, battery ($/MJ or $/kWh, Units_energy)	50.

cost factors and rates include technology and inflation, correspond to year_inf
rComp negative for cost reduction

		+	Maintenance	
MODEL_maint	int	+	maintenance cost estimate (1 total only, 2 separate components)	2
rLabor	real	+	labor rate ($ per hour)	160.
MMHperFH	real	+	maintenance man hours per flight hour	0.
Mlabor	real	+	MMH/FH factor $M_{\rm labor}$	0.0017
Mparts	real	+	parts factor $M_{\rm parts}$	34.
Mengine	real	+	engine overhaul factor $M_{\rm engine}$	1.45
Mmajor	real	+	major periodic maintenance factor $M_{\rm major}$	18.
Mbatt	real	+	battery maintenance factor $M_{\rm batt}$ ($/MJ or $/kWh per hour, Units_energy)	.10

---

labor rate includes inflation, corresponds to year_inf
current best practice: Mlabor=0.0017, Mparts=34, Mengine=1.45, Mmajor=18
current average practice: Mlabor=0.0027, Mparts=56, Mengine=1.74, Mmajor=28

maintenance man hours per flight hour calculated from sum of fixed term (MMHperFH) and term scaling with weight empty (Mlabor)

		+	Direct Operating Cost	
MODEL_doc	int	+	crew+depreciation+insurance estimate (1 total only, 2 separate components)	2
Kcdi	real	+	crew+depreciation+insurance factor $K_{\rm cdi}$	1.0
Kcrew	real	+	crew cost factor $K_{\rm crew}$	1.0
Kins	real	+	insurance cost $K_{\rm ins}$ (fraction aircraft cost)	.0056
KETS	real	+	emissions trading scheme cost $K_{\rm ETS}$ ($/kg CO_2)	.02
		+	Scott rotorcraft component cost model	
		+	production	
year_proc	int	+	year of procurement (0 same as year_inf, not used if <1955)	0
Nprod	int	+	aircraft production number (0 not used)	0
Nlot	int	+	number aircraft in this production lot (0 not used)	0
Nprod_eng	int	+	engine production number (0 not used)	0
		+	systems	
drFCE	real	+	cost factor $\Delta r_{\rm FCE}$, additional flight control electronics ($/lb or $/kg)	0.
drMEP	real	+	cost factor $\Delta r_{\rm MEP}$, additional mission equipment package ($/lb or $/kg)	0.

		+	component cost models	
f_sec	real	+	fuselage, fraction of secondary fuselage weight	0.35
KIND_fuse_boom	int	+	fuselage, includes tail boom (0 not)	0
KIND_fuse_dev	int	+	fuselage, early LRIP of new design (0 not)	0
MODEL_eng	int	+	engine, turboshaft (0 not)	1
Kmotor	real	+	electric motor/generator cost $c = KP^X$, factor	0.2
Xmotor	real	+	electric motor/generator cost $c = KP^X$, exponent	1.0
Pr_avg	real	+	engine, stage-averaged compressor pressure ratio	1.6
TBO_eng	real	+	engine, time between overhaul (hours)	2000.
KIND_eng_mar	int	+	engine, marinized (0 not)	0
KIND_eng_FADEC	int	+	engine, FADEC equipped (0 not)	1
KIND_xmsn_rg	int	+	transmission, engine group includes reduction gearbox (0 direct drive)	0
KIND_xmsn_mar	int	+	transmission, marinized (0 not)	0
KIND_av_dev	int	+	avionics, early LRIP of new package (0 not)	0
KIND_av_UAV	int	+	avionics, unmanned medium to long endurance aircraft (0 not)	0
f_env	real	+	environmental group, fraction prime equipment cost	0.03
f_arm_furn_LH	real	+	armament provisions, furnishings, and load and handling groups, fraction fuselage cost	0.12
KIND_int_SE_prof	int	+	integration and assembly, systems engineering, and profit (1 government, 2 commercial)	1
f_int_SE_prof	real	+	integration and assembly, systems engineering, and profit (commercial), fraction prime equipment cost	0.25

Chapter 17

Structure: Emissions

Variable	Type		Description	Default
		+	Emissions	
title	c*100	+	title	
notes	c*1000	+	notes	
MODEL_emissions	int	+	Emissions model (0 none)	1
		+	Emissions Trading Scheme (ETS)	
Kfuel(ntankmax)	real	+	CO_2 emissions from fuel used, K_{fuel} (kg/kg)	3.75
Kenergy(ntankmax)	real	+	CO_2 emissions from energy used, K_{energy} (kg/MJ or kg/kWh, Units_energy)	0.14
		+	Average Temperature Response (ATR)	
H	real	+	aircraft operating lifetime H (yr)	30.
U	real	+	aircraft utilization rate U (missions/yr)	350.
r	real	+	ATR discount rate r	0.03
tmax	real	+	ATR integration period $t_{\max}$ (yr)	500.
		+	emission index (kg/kg)	
EI_CO2(ntankmax)	real	+	carbon dioxide, EI_{CO_2}	3.16
EI_H2O(ntankmax)	real	+	water vapor, $EI_{\text{H}_2\text{O}}$	1.26
EI_SO4(ntankmax)	real	+	sulphates, EI_{SO_4}	0.0002
EI_soot(ntankmax)	real	+	soot, EI_{soot}	0.00004
EI_NOx(ntankmax)	real	+	nitrogen oxides, EI_{NO_x}	0.01
MODEL_NOx(ntankmax)	int	+	turboshaft engine NOx emission model (0 input EI_{NO_x}, 1 DLR, 2 Swiss)	1
KIND_NOx(ntankmax)	int	+	model parameters (0 input, 1 low emissions, 2 high emissions)	1
KEI0(ntankmax)	real	+	DLR model, K_{EI0}	0.0036739
KEI1(ntankmax)	real	+	DLR model, K_{EI1}	0.00748
KEIs(ntankmax)	real	+	Swiss model, K_{EIs}	0.004
fAIC	real	+	aviation induced cloudiness factor, f_{AIC}	1.0
		+	energy emission factor (kg/MJ or kg/kWh, Units_energy)	
K_CO2(ntankmax)	real	+	carbon dioxide, K_{CO_2}	0.14
K_H2O(ntankmax)	real	+	water vapor, $K_{\text{H}_2\text{O}}$	0.

K_SO4(ntankmax)	real	+	sulphates, K_{SO_4}	0.
K_soot(ntankmax)	real	+	soot, K_{soot}	0.
K_NOx(ntankmax)	real	+	nitrogen oxides, K_{NO_x}	0.
SET_credit	int	+	Emissions credit for energy generated (0 for none)	1

EI default values are for turboshaft engine

emission index (EI and K_{fuel}) only used for tanks that store and use fuel as weight (SET_burn=1)
energy emission factor (K and K_{energy}) only used for tanks that store and use fuel as energy (SET_burn=2)

ATR discount rate: $r \geq 100000$ evaluated as $r = \infty$

Chapter 18

Structure: Aircraft

Variable	Type		Description	Default
		+	Aircraft	
title	c*100	+	title	
notes	c*1000	+	notes	
config	c*16	+	Configuration	'helicopter'

config: identifies rotorcraft configuration
config = 'rotorcraft', 'helicopter', 'tandem', 'coaxial', 'tiltrotor', 'compound', 'multicopter', 'airplane'

Variable	Type		Description	Default
		+	Aircraft Controls	
ncontrol	int	+	number of aircraft controls (maximum ncontmax)	4
IDENT_control(ncontmax)	c*16	+	labels of aircraft controls	
nstate_control	int	+	number of control states (maximum nstatemax)	1
			pilot's controls (control number)	
		+	control values (function speed)	
nVcont(ncontmax)	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	
nVcoll	int	+	collective stick	0
nVlatcyc	int	+	lateral cyclic stick	0
nVlngcyc	int	+	longitudinal stick	0
nVpedal	int	+	pedal	0
nVtilt	int	+	tilt	0
cont(nvelmax,ncontmax)	real	+	values	
coll(nvelmax)	real	+	collective stick c_{AC0}	
latcyc(nvelmax)	real	+	lateral cyclic stick c_{ACc}	
lngcyc(nvelmax)	real	+	longitudinal cyclic stick c_{ACs}	
pedal(nvelmax)	real	+	pedal c_{ACp}	
tilt(nvelmax)	real	+	tilt α_{tilt}	

Vcont(nvelmax,ncontmax)	real	+	speeds (CAS or TAS)
Vcoll(nvelmax)	real	+	collective stick
Vlatcyc(nvelmax)	real	+	lateral cyclic stick
Vlngcyc(nvelmax)	real	+	longitudinal cyclic stick
Vpedal(nvelmax)	real	+	pedal
Vtilt(nvelmax)	real	+	tilt

---

control system: set of aircraft controls c_{AC} defined
  aircraft controls connected to individual controls of each component, $c = Tc_{AC} + c_0$
  for each component control, define matrix T (for each control state) and value c_0
    flight state specifies control state, or that control state obtained from conversion schedule
  c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
    use of component control c_0 can be suppressed for flight state using SET_comp_control
aircraft controls: identified by IDENT_control
  typical aircraft controls are pilot's controls; default IDENT_control='coll','latcyc','lngcyc','pedal','tilt'
  available for trim (flight state specifies trim option)
    initial values specified if control is trim variable; otherwise fixed for flight state
  each aircraft control can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
    coll/latcyc/lngcyc/pedal/tilt input put in appropriate nVcont-cont-Vcont, based on IDENT_control
  flight state input can override
  by connecting aircraft control to component control, flight state can specify component control value
sign conventions for pilot's controls: collective + up, lat cyclic + right, long cyclic + forward, pedal + nose right
  rotor controls are positive Fourier series, with azimuth measured in direction of rotation

---

		+	Aircraft Motion
		+	aircraft pitch angle θ_F
nVpitch	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)
pitch(nvelmax)	real	+	values
Vpitch(nvelmax)	real	+	speeds (CAS or TAS)
		+	aircraft roll angle ϕ_F
nVroll	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)
roll(nvelmax)	real	+	values
Vroll(nvelmax)	real	+	speeds (CAS or TAS)

---

aircraft motion
 available for trim (depending on flight state)
 each motion can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
 flight state input can override; initial value if trim variable

---

		+	Conversion	
Vconv_hover	real	+	maximum speed for hover and helicopter mode (CAS or TAS)	
Vconv_cruise	real	+	minimum speed for cruise (CAS or TAS)	
		+	control state	
kcont_hover	int	+	hover and helicopter mode ($V \leq V_{\text{conv}-\text{hover}}$)	1
kcont_conv	int	+	conversion mode ($V_{\text{conv}-\text{hover}} < V < V_{\text{conv}-\text{cruise}}$)	1
kcont_cruise	int	+	cruise mode ($V \geq V_{\text{conv}-\text{cruise}}$)	1
		+	drive system state (each propulsion group)	
kgear_hover(npropmax)	int	+	hover and helicopter mode ($V \leq V_{\text{conv}-\text{hover}}$)	1
kgear_conv(npropmax)	int	+	conversion mode ($V_{\text{conv}-\text{hover}} < V < V_{\text{conv}-\text{cruise}}$)	1
kgear_cruise(npropmax)	int	+	cruise mode ($V \geq V_{\text{conv}-\text{cruise}}$)	1

---

conversion control: use depends on STATE_control, SET_tilt, SET_Vtip of FltState
 hover and helicopter mode ($V \leq V_{\text{conv}-\text{hover}}$): use tilt=90, Vtip_hover, kgear_hover, kcont_hover
 cruise mode ($V \geq V_{\text{conv}-\text{cruise}}$): use tilt=0, Vtip_cruise, kgear_cruise, kcont_cruise
 conversion mode: tilt linear with V, use Vtip_hover, kgear_conv, kcont_conv
 nacelle tilt angle: 0 for cruise, 90 deg for helicopter mode flight

---

		+	Never-exceed speed	
SET_VNE	c*32	+	model	'none'
		+	table	
KIND_VNE_table	int	+	velocity (0 TAS, 1 CAS, 2 IAS)	0
nwt_VNE	int	+	number of weights (maximum nvnemax)	
nalt_VNE	int	+	number of altitudes (maximum nvnemax)	
ntemp_VNE	int	+	number of temperatures (maximum nvnemax)	
rwt_VNE(nvnemax)	real	+	weight ratio $r_W = W_G/W_D$ (fraction DGW)	
alt_VNE(nvnemax,nvnemax)	real	+	density altitude h_d(nalt,nwt)	
temp_VNE(nvnemax)	real	+	temperature τ (deg C)	
VNE(nvnemax,nvnemax)	real	+	never-exceed speed V_{NEt}(nalt,nwt) (knots)	
VNE3(nvnemax,nvnemax,nvnemax)	real	+	never-exceed speed V_{NEt}(nalt,nwt,ntemp) (knots)	
KIND_VNE_stall(nrotormax)	int	+	stall model, each rotor (0 for no limit, 1 steady, 2 transient, 3 equation)	3
C_VNE(5)	real	+	compressibility limit constants C_n	
Mat_VNE(nrotormax)	real	+	advancing tip Mach number M_{at}, each rotor (0. for no limit)	1.
		+	limits (0. not used)	
VNEmaxTAS	real	+	TAS maximum (knots)	0.
VNEmaxIAS	real	+	IAS maximum (knots)	0.
VNEminTAS	real	+	TAS minimum (knots)	0.
VNEminIAS	real	+	IAS minimum (knots)	0.

---

never-exceed speed: calculate $V_{\rm NE}$ in knots TAS; depends on density altitude h_d, gross weight W_G (in terms of weight ratio $r_W = W_G/W_D$, fraction DGW), and temperature τ

SET_VNE = 'none', or one to four of ('table' or 'table3', 'stall', 'comp', 'Mat')

table limit (2D): $V_{{\rm NE}t}(h_d)$ for set of weights r_W (alt_VNE(nalt,nwt))

table limit (3D): $V_{{\rm NE}t}(h_d, r_W, \tau)$ (alt_VNE not depend on weight)

stall limit: $V_{{\rm NE}s}$ from rotor thrust capability (C_T/σ vs μ)

compressibility limit: $V_{{\rm NE}c} = C_1 - C_2 h_d + C_3\tau - C_4 V_{\rm tip} - C_5 r_W$ (knots IAS; temperature in deg C)

Mach number limit: $V_{{\rm NE}m}$ from advancing tip Mach number M_{at}

		+	Indicated airspeed correction	
nIAS	int	+	number of values (maximum niasmax, 0 no correction)	0
IAS(niasmax)	real	+	indicated airspeed (knots)	
CAS(niasmax)	real	+	calibrated airspeed (knots)	
SET_Vschedule	int	+	Velocity schedules (1 CAS, 2 TAS, 3 IAS)	1

indicated airspeed correction: IAS(1)=CAS(1)=0., both IAS and CAS unique and sequential
velocity schedules: all described as function CAS or TAS or IAS
conversion, controls and motion, rotor tip speed, landing gear retraction, trim targets, drive system ratings

		+	Trim states	
nstate_trim	int	+	number of trim states (maximum ntrimstatemax)	1
IDENT_trim(ntrimstatemax)	c*12	+	label of trim state	
mtrim(ntrimstatemax)	int	+	number of trim variables (maximum mtrimmax)	0
trim_quant(mtrimmax,ntrimstatemax)	c*16	+	trim quantity name	
trim_var(mtrimmax,ntrimstatemax)	c*16	+	trim variable name	
trim_target(mtrimmax,ntrimstatemax)	int	+	target source (1 FltState, 2 component)	1

trim state: one or more set of quantities and variables for trim iteration
FltState identifies trim state (STATE_trim match IDENT_trim),
trim variable:

description	trim_var	
aircraft control	match IDENT_control	
aircraft orientation	'pitch', 'roll'	body axes relative inertial axes
aircraft velocity	'speed', 'ROC'	horizontal, vertical flight speed
aircraft velocity	'side'	sideslip angle
aircraft angular rate	'pullup', 'turn'	Euler angle rates
propulsion group tip speed	'Vtip n'	
propulsion group engine speed	'Nspec n'	

trim quantity:

description	trim_quant	target
aircraft total force	'force x', 'force y', 'force z'	zero
aircraft total moment	'moment x', 'moment y', 'moment z'	zero
aircraft load factor	'nx', 'ny', 'nz'	FltState%trim_target
propulsion group power	'power n'	FltState%trim_target
power margin	'P margin n'	FltState%trim_target
torque margin	'Q margin n'	FltState%trim_target
engine group power	'power EG n'	FltState%trim_target
power margin	'E margin n'	FltState%trim_target
momentum margin	'FE margin n'	FltState%trim_target
jet group thrust	'jet n'	FltState%trim_target
jet thrust margin	'J margin n'	FltState%trim_target
momentum margin	'FJ margin n'	FltState%trim_target
charge group power	'charge n'	FltState%trim_target
charge power margin	'C margin n'	FltState%trim_target
fuel tank energy flow	'tank n'	FltState%trim_target
battery power margin	'B margin n'	FltState%trim_target
rotor lift	'lift rotor n', 'flift rotor n'	FltState%trim_target, Rotor%Klift
rotor lift	'CLs rotor n', 'vert rotor n'	FltState%trim_target, Rotor%Klift
rotor propulsive force	'prop rotor n', 'fprop rotor n'	FltState%trim_target, Rotor%Kprop
rotor propulsive force	'CXs rotor n', 'X/q rotor n'	FltState%trim_target, Rotor%Kprop
rotor thrust	'CTs rotor n'	FltState%trim_target, Rotor%Klift
rotor thrust margin	'T margin n'	FltState%trim_target
rotor thrust margin	'T margin tran n', 'T margin eqn n'	FltState%trim_target
rotor shaft power	'power rotor n'	FltState%trim_target
rotor flapping	'betac n', 'lngflap n'	FltState%trim_target
rotor flapping	'betas n', 'latflap n'	FltState%trim_target
rotor hub moment	'hub Mx n', 'roll n'	FltState%trim_target
rotor hub moment	'hub My n', 'pitch n'	FltState%trim_target
rotor torque	'hub Mz n', 'torque n'	FltState%trim_target
wing lift	'lift wing n', 'flift wing n'	FltState%trim_target, Wing%Klift
wing lift coefficient	'CL wing n'	FltState%trim_target, Wing%Klift
wing lift margin	'L margin n'	FltState%trim_target
tail lift	'lift tail n'	FltState%trim_target

if trim_target=1, trim quantity target value is FltState%trim_target; otherwise component Klift or Kprop used
if trailing "n" is absent, use first component (n=1)

trim_quant='flift rotor n' or trim_quant='flift wing n': target is fraction total aircraft lift (GW*nAC(3))
trim_quant='fprop rotor n': target is fraction total aircraft drag (qAC*DoQ)
trim_quant='T margin n' uses Rotor%CTs_steady, trim_quant='T margin tran n' uses Rotor%CTs_tran
trim_quant='T margin eqn n' uses equation for rotor thrust capability (Rotor%K0_limit and Rotor%K1_limit)

trim_var='Vtip' or 'Nspec': requires FltAircraft%SET_Vtip='input'

		+	Geometry	
INPUT_geom	int	+	input (1 fixed, SL/BL/WL; 2 scaled, from XoL/YoL/ZoL)	2
		+	scaled geometry	
		+	reference length	
KIND_scale	int	+	kind (1 rotor radius, 2 wing span, 3 fuselage length)	1
kScale	int	+	identification (component number)	1
		+	reference point	
KIND_Ref	int	+	kind (0 input, 1 rotor, 2 wing, 3 fuselage, 4 center of gravity)	0
kRef	int	+	identification (component number)	1
SL_Ref	real	+	stationline	
BL_Ref	real	+	buttline	
WL_Ref	real	+	waterline	
			calculated reference point (input or component)	
loc_cg	Location	+	baseline center of gravity location	

Geometry: Location for each component
  fixed geometry input (INPUT_geom = 1): dimensional SL/BL/WL
    stationline + aft, buttline + right, waterline + up; arbitary origin; units = ft or m
  scaled geometry input (INPUT_geom = 2): divided by reference length (KIND_scale, kScale)
    XoL + aft, YoL + right, ZoL + up; from reference point
    option to fix some geometry (FIX_geom in Location override INPUT_geom)
    option to specify reference length (KIND_scale in Location override this global KIND_scale)

reference point: KIND_Ref, kRef; input dimensional XX_Ref, or position of identified component
component reference must be fixed
certain Locations can be calculated from other parameters (configuration specific)
center of gravity: baseline is for nacelle angle = 90
flight state has calculated or input actual cg location

		+	Takeoff flight condition	
SET_atmos	c*12	+	atmosphere specification	'std'
temp	real	+	temperature τ	
dtemp	real	+	temperature increment ΔT	0.
density	real	+	density ρ	
csound	real	+	speed of sound c_s	
viscosity	real	+	viscosity μ	
altitude	real	+	altitude	

---

takeoff condition (density) used for C_T/σ in rotor sizing
SET_atmos, atmosphere specification:
'std' = standard day at specified altitude (use altitude)
'dtemp' = standard day at specified altitude, plus temperature increment (use altitude, dtemp)
'temp' = standard day at specified altitude, and specified temperature (use altitude, temp)
'dens' = input density and temperature (use density, temp)
'input' = input density, speed of sound, and viscosity (use density, csound, viscosity)
'notair' = input, not air on earth (use density, csound, viscosity)
see FltState%SET_atmos for other options (polar, tropical, and hot days)

		+	Weight	
DGW	real	+	design gross weight W_D	
Wfuel_DGW	real	+	mission fuel W_{fuel} corresponding to DGW	
Wpay_DGW	real	+	payload W_{pay} corresponding to DGW	
WE	real	+	weight empty W_E	
dWE	real	+	weight increment	
fWE	real	+	weight factor	
		+	structural design gross weight	
SDGW	real	+	structural design gross weight W_{SD}	
dSDGW	real	+	weight increment	0.
fSDGW	real	+	weight factor	1.
fFuelSDGW	real	+	fraction main fuel tanks filled at SDGW	1.
		+	maximum takeoff weight	
WMTO	real	+	maximum takeoff weight W_{MTO}	
dWMTO	real	+	weight increment	0.
fWMTO	real	+	weight factor	1.
nz_ult	real	+	design ultimate flight load factor $n_{z\text{ult}}$ at SDGW	6.0

---

input or calculated: design gross weight W_D (FIX_DGW), structural design gross weight W_{SD} (SET_SDGW), maximum takeoff weight W_{MTO} (SET_WMTO), weight empty W_E (FIX_WE)
if calculated, then input parameter is initial value

DGW, design gross weight: used for rotor disk loading and blade loading, wing loading, power loading, thrust loading
to obtain aircraft moments of inertia from radii of gyration
for tolerance and perturbation scales of the solution procedures
optionally to define structural design gross weight and maximum takeoff weight
optionally to specify the gross weight for missions and flight conditions
Wfuel_DGW and Wpay_DGW usually calculated (identified as input so inherited by next case)

FIX_WE: fixed or scaled weight empty obtained by adjusting contingency weight
scaled with design gross weight: W_E=dWE+fWE*W_D

SET_SDGW, structural design gross weight:
'input' = input
'f(DGW)' = based on DGW; W_{SD}=dSDGW+fSDGW*W_D

'f(WMTO)' = based on WMTO; W_{SD}=dSDGW+fSDGW*W_{MTO}
'maxfuel' = based on fuel state; W_{SD}=dSDGW+fSDGW*W_G, $W_G = W_D$–Wfuel_DGW+fFuelSDGW*$W_{\text{fuel-cap}}$
'perf' = calculated from maximum gross weight at SDGW sizing conditions (DESIGN_sdgw)
SET_WMTO, maximum takeoff weight:
'input' = input
'f(DGW)' = based on DGW; W_{MTO}=dWMTO+fWMTO*W_D
'f(SDGW)' = based on SDGW; W_{MTO}=dWMTO+fWMTO*W_{SD}
'maxfuel' = based on maximum fuel; W_{MTO}=dWMTO+fWMTO*W_G, $W_G = W_D$–Wfuel_DGW+$W_{\text{fuel-cap}}$
'perf' = calculated from maximum gross weight at WMTO sizing conditions (DESIGN_wmto)
SDGW used for weights (fuselage, rotor, wing)
WMTO used for cost, drag (scaled aircraft and hub drag), and weights (system, fuselage, landing gear, engine group)
nz_ult, design ultimate flight load factor at SDGW: used for weights (fuselage, rotor, wing)

		+	Weight
		+	moments of inertia (based on design gross weight, scaled with reference length)
kx	real	+	roll radius of gyration k_x/L
ky	real	+	pitch radius of gyration k_y/L
kz	real	+	yaw radius of gyration k_z/L

weight empty = structure + propulsion + systems and equipment + vibration + contingency
operating weight = weight empty + fixed useful load
weight statement defines fixed useful load and operating weight for design configuration
so for flight state, additional fixed useful load = auxiliary fuel tank and kits and increments
flight state can also increment crew weight or equipment weight
flight state: gross weight, useful load (payload, usable fuel, fixed useful load), operating weight
gross weight = weight empty + useful load = operating weight + payload + usable fuel
useful load = fixed useful load + payload + usable fuel

		+	Drag	
FIX_drag	int	+	total aircraft D/q (0 calculated; 1 fixed, input D/q; 2 scaled, input C_D; 3 scaled, from k)	0
DoQ	real	+	area D/q	0.
CD	real	+	coefficient C_D (based on rotor area, $D/q = A_{\text{ref}} C_D$)	0.008
kDrag	real	+	$k = (D/q)/(W_{MTO}/1000)^{2/3}$ (Units_Dscale)	2.5
FIX_DL	int	+	total aircraft download (0 calculated; 1 fixed, input D/q_V; 2 scaled, from k_{DL})	0
DoQV	real	+	area $(D/q)_V$	0.
kDL	real	+	$k_{DL} = (D/q)_V/A_{\text{ref}}$	0.05

fixed drag or download: obtained by adjusting contingency D/q or $(D/q)_V$
FIX_drag: minimum drag, excludes drag due to lift and angle of attack
use only one of input DoQ, CD, kDrag (others calculated)
A_{ref} = reference rotor area; units of kDrag are ft^2/klb$^{2/3}$ or m^2/Mg$^{2/3}$
CD = 0.02 for old helicopter, 0.008 for current low drag helicopters
kDrag = 9 for old helicopter, 2.5 for current low drag helicopters,
1.6 for current tiltrotors, 1.4 for turboprop aircraft (English units)
FIX_DL, download: A_{ref} = reference rotor area, kDL $\sim DL/T$
use only one of DoQV, kDL (other calculated)

		+	Aerodynamics	
KIND_alpha	int	+	angle of attack and sideslip angle representation (1 conventional, 2 reversed for sideward flight)	2

angle of attack and sideslip angle: reversed definition best for sideward flight

		+	Number of Components	
nRotor	int	+	rotors (maximum nrotormax)	2
nWing	int	+	wings (maximum nwingmax)	0
nTail	int	+	tails (maximum ntailmax)	1
nTank	int	+	fuel tank systems (maximum ntankmax)	1

nPropulsion	int	+	propulsion groups (maximum npropmax)	1
nEngineGroup	int	+	engine groups (maximum nengmax)	1
nJetGroup	int	+	jet groups (maximum njetmax)	0
nChargeGroup	int	+	charge groups (maximum nchrgmax)	0
nEngineModel	int	+	engine models (maximum nengmax)	1
nEngineParamN	int	+	engine model parameters (maximum nengpmax)	0
nEngineTable	int	+	engine tables (maximum nengmax)	0
nRecipModel	int	+	reciprocating engine models (maximum nengmax)	0
nCompressorModel	int	+	compressor models (maximum nengmax)	0
nMotorModel	int	+	motor models (maximum nengmax)	0
nJetModel	int	+	jet models (maximum njetmax)	0
nFuelCellModel	int	+	fuel cell models (maximum nchrgmax)	0
nSolarCellModel	int	+	solar cell models (maximum nchrgmax)	0
nBatteryModel	int	+	battery models (maximum ntankmax)	0

---

propulsion group is set of components and engine groups, connected by drive system
engine model or engine table or reciprocating engine or motor model describes particular engine,
used in one or more engine groups
jet model describes particular jet, used in one or more jet groups
fuel cell model or solar cell model describes particular charger, used in one or more charge groups
battery model describes particular battery, used in one or more fuel tanks

---

Chapter 19

Structure: Systems

Variable	Type		Description	Default
		+	Systems	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Weight	
SET_Wpayload	int	+	payload (1 no details; 2 all terms)	1
Upass	real	+	weight per passenger	
		+	fixed useful load	
SET_Wcrew	int	+	crew weight (1 no details; 2 all terms)	1
Wcrew	real	+	weight or adjustment	
Ucrew	real	+	weight per crew	
Ncrew	int	+	number of crew	
Wtrap	real	+	trapped fluids and engine oil weight	0.
		+	other fixed useful load	
nWoful	int	+	number of categories (0 for one value without name; maximum 10)	0
Woful_name(10)	c*24	+	category name	' '
Woful(10)	real	+	baseline weight	0.
Wotherkit	real	+	other kit	0.

SET_Wpayload: payload specified by flight condition or mission
SET_Wcrew: no details (only Wcrew) or all terms (Ucrew*Ncrew+Wcrew)

other fixed useful load: can include baggage, gun installations, weapons provisions, aircraft survivability equipment, survival kits, life rafts, oxygen

SET_fold	int	+	folding (0 none, 1 fold weights, 2 with kit)	0
		+	folding weight in kit f_{foldkit} (fraction wing/rotor/tail/body fold weight)	
fWfoldkitW(nwingmax)	real	+	wing	0.5
fWfoldkitR(nrotormax)	real	+	rotor	0.5
fWfoldkitT(ntailmax)	real	+	tail	0.5
fWfoldkitFw	real	+	body (wing and rotor fold)	0.5
fWfoldkitFt	real	+	body (tail fold)	0.5
SET_Wvib	int	+	vibration treatment weight (1 fraction weight empty, 2 input)	1
Wvib	real	+	weight W_{vib}	
fWvib	real	+	fraction weight empty f_{vib}	
SET_Wcont	int	+	contingency weight (1 fraction weight empty, 2 input)	1
Wcont	real	+	weight W_{cont}	
fWcont	real	+	fraction weight empty f_{cont}	

W_E = (structure + propulsion group + systems and equipment) + W_{vib} + W_{cont}
SET_Wvib: W_{vib} input or $W_{\text{vib}} = f_{\text{vib}} W_E$
SET_Wcont: W_{cont} input or $W_{\text{cont}} = f_{\text{cont}} W_E$; or adjust W_{cont} for input or scaled W_E (FIX_WE=1 or 2)

SET_fold, folding:
set component dWxxfold=0 and fWxxfold=0 for no rotor/wing/tail/body fold weight
fraction fWfoldkit of fold weight in fixed useful load as kit, remainder kept in component weight
kit weight removable, absent for specified flight conditions and missions

---

		+	systems and equipment	
Wauxpower	real	+	auxiliary power group (APU)	0.
Winstrument	real	+	instruments group	0.
Wpneumatic	real	+	pneumatic group	0.
Wenviron	real	+	environmental control group	0.
SET_Welectrical	int	+	electrical group (1 no details; 2 all terms)	1
Welectrical	real	+	aircraft	0.
Welect_supply	real	+	power supply	0.
Welect_conv	real	+	power conversion	0.
Welect_distrib	real	+	power distribution and controls	0.
Welect_lights	real	+	lights and signal devices	0.

Welect_support	real	+	equipment supports	0.
SET_WMEQ	int	+	avionics group (1 no details; 2 all terms)	1
WMEQ	real	+	avionics	0.
Wavionics_com	real	+	communications	0.
Wavionics_nav	real	+	navigation	0.
Wavionics_ident	real	+	identification	0.
Wavionics_disp	real	+	control and display	0.
Wavionics_survive	real	+	aircraft survivability	0.
Wavionics_mission	real	+	mission system equipment	0.
		+	armament group	
SET_Warmor	int	+	armor (1 no details; 2 all terms)	1
Warmor	real	+	armor	0.
Uarmor_floor	real	+	cabin floor armor weight per area	
Uarmor_wall	real	+	cabin wall armor weight per area	
Uarmor_crew	real	+	armor weight per crew	
SET_Warmprov	int	+	armament provisions (1 no details; 2 all terms)	1
Warmprov	real	+	armament provisions	0.
Warmprov_gun	real	+	gun provisions	0.
Warmprov_turret	real	+	turret systems	0.
Warmprov_expend	real	+	expendable weapons provisions	0.
Warm_elect	real	+	armament electronics (avionics group)	0.
SET_Wfurnish	int	+	furnishings and equipment group (1 no details; 2 all terms)	1
Wfurnish	real	+	furnishings and equipment	0.
		+	accommodations for personnel	
Useat_crew	real	+	each crew seat	
Useat_pass	real	+	each passenger seat	
Uaccom_crew	real	+	miscellaneous accommodation per crew seat	
Uaccom_pass	real	+	miscellaneous accommodation per passenger seat	
Uox_crew	real	+	oxygen system per crew seat	
Uox_pass	real	+	oxygen system per passenger seat	
Wfurnish_misc	real	+	miscellaneous equipment	0.
		+	furnishings	
Wfurnish_trim	real	+	trim	0.
Uinsulation	real	+	acoustic and thermal insulation weight per cabin area	

		+	emergency equipment	
Wemerg_fire	real	+	fire detection and extinguishing	0.
Wemerg_other	real	+	other emergency equipment	0.
SET_Wload	int	+	load and handling group (1 no details; 2 all terms)	1
Wload	real	+	load and handling	0.
Whandling_aircraft	real	+	aircraft handling	0.
		+	load handling	
Uhandling_cargo	real	+	cargo handling weight per cabin floor area	
Wload_hoist	real	+	hoist	0.
Wload_extprov	real	+	external load provisions	0.
		+	systems and equipment	
Ncrew_seat	int	+	number of crew seats	0
Npass_seat	int	+	number of passenger seats	0
Ucrew_seat_inc	real	+	equipment weight increment per crew seat (0. for default)	0.
Upass_seat_inc	real	+	equipment weight increment per passenger seat (0. for default)	0.

SET_Welectrical=1: only Welectrical+WDIelect
SET_WMEQ=1: only WMEQ; equipment weights include installation
SET_Warmor=1: only Warmor
SET_Warmprov=1: only Warmprov
SET_Wfurnish=1: only Wfurnish
miscellaneous accommodation includes galleys and toilets
miscellaneous equipment includes cockpit displays
trim includes floor covering, partitions, crash padding, acoustic and thermal insulation
excluding vibration absorbers
other emergency equipment includes first aid, survival kit, life raft
SET_Wload=1: only Wload

equipment weight increment is for flight condition and mission; default (if SET_furnish=2 and SET_armor=2):
Ucrew_seat_inc=Useat_crew+Uaccom_crew+Uox_crew+Uarmor_crew
Upass_seat_inc=Useat_pass+Uaccom_pass+Uox_pass

		+	Weight	
		+	systems and equipment	
		+	flight control group and hydraulic group	
MODEL_fc	int	+	model (0 input, 1 NDARC, 2 custom)	1
MODEL_RWfc	int	+	rotary wing flight controls (0 not present, 1 global, 2 for each rotor)	1
refRotor	int	+	reference rotor number for global	1
MODEL_FWfc	int	+	fixed wing flight controls (0 for not present)	1
MODEL_CVfc	int	+	conversion controls (0 for not present)	1
		+	flight control weight increment	
dWRWfc_b	real	+	rotary wing, boosted	0.
dWRWfc_mb	real	+	rotary wing, control boost mechanisms	0.
dWRWfc_nb	real	+	rotary wing, non-boosted	0.
dWFWfc_mb	real	+	fixed wing, control boost mechanisms	0.
dWFWfc_nb	real	+	fixed wing, non-boosted	0.
dWCVfc_mb	real	+	conversion, boosted	0.
dWCVfc_nb	real	+	conversion, control boost mechanisms	0.
		+	fixed flight controls	
Wfc_cc	real	+	cockpit controls	0.
Wfc_afcs	real	+	automatic flight control system	0.
		+	hydraulic weight increment	
dWRWhyd	real	+	rotary wing	0.
dWFWhyd	real	+	fixed wing	0.
dWCVhyd	real	+	conversion	0.
WEQhyd	real	+	equipment hydraulics	0.
		+	anti-icing group	
MODEL_DI	int	+	model (0 input, 1 NDARC, 2 custom)	1
		+	weight increment	
dWDIelect	real	+	electrical system	0.
dWDIsys	real	+	anti-ice system	0.

---

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

MODEL_RWfc: global option is based on just main rotors
"for each rotor" option sums separate contributions from all rotors

tiltrotor wing weight model requires weight on wing tip: distributed to designated rotor;
sum rotary wing and conversion flight controls, hydraulic group, trapped fluids

		+	Technology Factors	
		+	rotary wing flight control weight	
TECH_RWfc_b	real	+	boosted χ_{RWb}	1.0
TECH_RWfc_mb	real	+	control boost mechanisms χ_{RWmb}	1.0
TECH_RWfc_nb	real	+	non-boosted χ_{RWnb}	1.0
		+	fixed wing flight control weight	
TECH_FWfc_mb	real	+	control boost mechanisms χ_{FWmb}	1.0
TECH_FWfc_nb	real	+	non-boosted χ_{FWnb}	1.0
		+	conversion flight control weight	
TECH_CVfc_mb	real	+	control boost mechanisms χ_{CVmb}	1.0
TECH_CVfc_nb	real	+	non-boosted χ_{CVnb}	1.0
		+	flight control hydraulics	
TECH_RWhyd	real	+	rotary wing $\chi_{RW\text{hyd}}$	1.0
TECH_FWhyd	real	+	fixed wing $\chi_{FW\text{hyd}}$	1.0
TECH_CVhyd	real	+	conversion $\chi_{CV\text{hyd}}$	1.0
		+	anti-icing	
TECH_DIelect	real	+	electrical system $\chi_{DI\text{elect}}$	1.0
TECH_DIsys	real	+	anti-ice system $\chi_{DI\text{sys}}$	1.0

		+	Flight Control Group, NDARC Weight Model	
		+	rotary wing flight controls	
MODEL_WRWfc	int	+	model (1 fraction, 2 parametric, 3 Boeing, 4 GARTEUR, 5 Tishchenko, 6 generic)	1
fRWfc_nb	real	+	AFDD: non-boosted control weight f_{RWnb} (fraction boost mechanisms weight)	0.6
xRWfc_red	real	+	AFDD: hydraulic system redundancy/complexity factor $f_{RW\text{red}}$	3.0
KIND_WRWfc	int	+	AFDD: survivability (1 baseline, 2 UTTAS/AAH level of survivability)	2
fRWfc_b	real	+	Boeing, GARTEUR, Tishchenko, or generic: boosted weight f_{RWb} (fraction boosted + boost mech, or total)	0.2
fRWfc_mb	real	+	GARTEUR, Tishchenko, or generic: boost mechanisms weight f_{RWmb} (fraction total weight)	0.2
KRW	real	+	generic: factor K_{RW}	0.

XRWN	real	+	exponent X_{RWN}	0.
XRWR	real	+	exponent X_{RWR}	0.
XRWc	real	+	exponent X_{RWc}	0.
XRWW	real	+	exponent X_{RWW}	0.
XRWb	real	+	exponent X_{RWb}	0.
		+	fixed wing flight controls	
MODEL_WFWfc	int	+	model (1 full controls, 2 only on hor tail, 3 GARTEUR, Raymer (4 transport, 5 general aviation), 6 generic)	1
fFWfc_nb	real	+	non-boosted weight f_{FWnb} (fraction total fixed wing flight control weight)	0.10
nfunction	int	+	Raymer: number of control functions	6
fmech	real	+	Raymer: number of mechanical functions (fraction total)	0.2
KFW	real	+	generic, factor K_{FW}	0.
XFW	real	+	exponent X_{FW}	0.
		+	conversion controls	
fCVfc_mb	real	+	boost mechanisms weight f_{CVmb} (fraction maximum takeoff weight)	0.02
fCVfc_nb	real	+	non-boosted weight f_{CVnb} (fraction boost mechanisms weight)	0.10
		+	cockpit controls	
MODEL_cc	int	+	model (1 fixed Wfc_cc, 2 scaled with DGW)	1
Kcc	real	+	factor K_{cc}	1.7
Xcc	real	+	exponent X_{cc}	0.41
		+	Hydraulic Group, NDARC Model	
		+	flight control hydraulics	
fRWhyd	real	+	rotary wing $f_{RW\mathrm{hyd}}$ (fraction rotary wing boost mechanisms + hydraulic weight)	0.40
fFWhyd	real	+	fixed wing $f_{FW\mathrm{hyd}}$ (fraction fixed wing boost mechanisms weight)	0.10
fCVhyd	real	+	conversion $f_{CV\mathrm{hyd}}$ (fraction conversion boost mechanisms weight)	0.10

---

flight controls = non-boosted (do not see aero surface or rotor loads) + boost mechanisms (actuators) + boosted

MODEL_WRWfc = fraction: parametric except for non-boosted controls (from fRWfc_nb)

typically fRWfc_nb = 0.6 (data range 0.3 to 1.8), fRWhyd = 0.4
xRWfc_red = 1.0 to 3.0

		+	Custom Weight Model	
WtParam_fc(8)	real	+	parameters	0.
		+	Anti-Icing Group, NDARC Weight Model	
kDeIce_elec(nrotormax)	real	+	weight factor for electrical system $K_{\rm elec}$ (lb/ft^2 or kg/m^2)	0.25
kDeIce_rotor(nrotormax)	real	+	weight factor for main rotor $K_{\rm rotor}$ (lb/ft^2 or kg/m^2)	0.25
kDeIce_wing(nwingmax)	real	+	weight factor for wing $K_{\rm wing}$ (lb/ft or kg/m)	0.
kDeIce_air(nengmax)	real	+	weight factor for engine air intake $K_{\rm air}$ (lb/lb or kg/kg)	0.006
kDeIce_jet(njetmax)	real	+	weight factor for jet air intake $K_{\rm jet}$ (lb/lb or kg/kg)	0.006
		+	Custom Weight Model	
WtParam_DI(8)	real	+	parameters	0.

Chapter 20

Structure: Fuselage

Variable	Type		Description	Default
		+	Fuselage	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Geometry	
loc_fuselage	Location	+	fuselage location	
SET_length	int	+	fuselage length (1 input, 2 calculated, 3 from rotor and tail only, 4 from rotor only)	1
Length_fus	real	+	length ℓ_{fus}	
SET_nose	int	+	nose length (distance forward of hub; 1 input, 2 calculated)	1
Length_nose	real	+	nose length ℓ_{nose}	
fLength_nose	real	+	nose length (fraction reference length)	
SET_aft	int	+	aft length (distance aft of hub; 1 input, 2 calculated)	1
Length_aft	real	+	aft length ℓ_{aft}	
fLength_aft	real	+	aft length (fraction reference length)	
fRef_fus	real	+	fuselage SL location relative nose f_{ref} (fraction fuselage length)	
Width_fus	real	+	fuselage width w_{fus}	
SET_Swet	int	+	fuselage wetted area (1 input, 2 input plus boom, 3 from nose length, 4 from fuselage length, 5 from weight)	2
Swet	real	+	wetted area S_{wet}	
Sproj	real	+	projected area S_{proj}	
fSwet	real	+	factor for wetted area f_{wet} or k_{wet}	1.
fSproj	real	+	factor for projected area f_{proj} or k_{proj}	1.
Height_fus	real	+	fuselage height h_{fus}	
Circum_boom	real	+	tail boom effective circumference C_{boom}	
Width_boom	real	+	tail boom effective width w_{boom}	
SET_Scabin	int	+	cabin area (1 input, 2 calculated)	2
Scabin	real	+	total cabin surface area S_{cabin}	
Scabin_floor	real	+	cabin floor area $S_{\text{cabin-floor}}$	
Scabin_wall	real	+	cabin wall area $S_{\text{cabin-wall}}$	

fScabin	real	+	factor for total cabin surface area $f_{\rm cabin}$	0.6
fScabin_floor	real	+	factor for cabin floor area $f_{\rm cabin-floor}$	0.6
fScabin_wall	real	+	factor for cabin wall area $f_{\rm cabin-wall}$	0.6
KIND_scale	int	+	reference length (1 rotor radius, 2 wing span, 3 fuselage length)	1
refRotor	int	+	rotor number (for rotor radius)	1
refWing	int	+	wing number (for wing span)	1

SET_length: input (use Length_fus) or calculated (from nose and aft lengths)
calculated uses rotor, tail, wing locations; or just rotor and tail, or just rotor
which can not then be scaled with fuselage length
SET_nose: input (use Length_nose) or calculated (from fLength_nose); used for Length_fus and Swet
SET_aft: input (use Length_aft) or calculated (from fLength_aft); used for Length_fus
fRef_fus=(SL_fuselage–SL_nose)/Length_fus; used for operating length and sketch
input required if SET_length = input, otherwise calculated

SET_Swet: both wetted area and projected area; input (use Swet, Sproj),
or calculated (from fSwet, fSproj, Width_fus, Height_fus, and fuselage or nose length)
or from weight, units of $k_{\rm wet}$ = fSwet and $k_{\rm proj}$ = fSproj are ft^2/klb$^{2/3}$ or m^2/Mg$^{2/3}$
boom circumference and width used if SET_Swet not input and not from weight (set to zero if no boom)

SET_Scabin: cabin areas used for systems and equipment weights

---

		+	Geometry (for graphics)	
Height_ramp	real	+	height of cargo ramp	
fLength_cargo	real	+	fraction of fuselage length used for cargo	0.60
		+	Controls	
		+	flow control momentum coefficient C_μ	
INPUT_flow	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_flow(ncontmax,nstatemax)	real	+	control matrix	
nVflow	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
flow(nvelmax)	real	+	values	
Vflow(nvelmax)	real	+	speeds (CAS or TAS)	

---

aircraft controls connected to individual controls of component, $c = Tc_{AC} + c_0$
for each component control, define matrix T (for each control state) and value c_0
flight state specifies control state, or that control state obtained from conversion schedule
c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
by connecting aircraft control to comp control, flight state can specify comp control value
initial values if control is connected to trim variable; otherwise fixed for flight state

---

		+	Aerodynamics	
MODEL_aero	int	+	model (0 none, 1 standard)	1
DoQ_cont	real	+	contingency drag, area $(D/q)_{\rm cont}$	0.
DoQV_cont	real	+	contingency vertical drag, area $(D/q)_{V{\rm cont}}$	0.

DoQ_cont calculated if total drag fixed (Aircraft FIX_drag); otherwise input
DoQV_cont calculated if total download fixed (Aircraft FIX_DL); otherwise input

---

		+	Weight	
		+	fuselage group	
MODEL_weight	int	+	fuselage group model (0 input, 1 NDARC, 2 custom)	1
		+	weight increment	
dWbody	real	+	basic body	0.
dWmar	real	+	body marinization	0.
dWpress	real	+	pressurization	0.
dWcrash	real	+	body crashworthiness	0.
dWftfold	real	+	tail fold	0.
dWfwfold	real	+	wing fold	0.
		+	Technology Factors	
TECH_body	real	+	basic body $\chi_{\rm basic}$	1.0
TECH_mar	real	+	body marinization $\chi_{\rm mar}$	1.0
TECH_press	real	+	pressurization $\chi_{\rm press}$	1.0
TECH_crash	real	+	body crashworthiness $\chi_{\rm cw}$	1.0
TECH_ftfold	real	+	tail fold $\chi_{\rm tfold}$	1.0
TECH_fwfold	real	+	wing fold $\chi_{\rm wfold}$	1.0

			weight model result multiplied by technology factor and increment added: Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.	
		+	Aerodynamics, Standard Model	
AoA_zl	real	+	zero lift angle of attack α_{zl} (deg)	0.
AoA_max	real	+	angle of attack for maximum lift $\alpha_{\max}$ (deg)	10.
		+	lift	
SET_lift	int	+	specification (1 fixed, L/q; 2 scaled, C_L)	2
dLoQda	real	+	lift slope, $d(L/q)/d\alpha$ (per rad)	0.
dCLda	real	+	lift slope, $C_{L\alpha} = dC_L/d\alpha$ (per rad; based on wetted area, $L/q = SC_L$)	0.
		+	pitch moment	
SET_moment	int	+	specification (1 fixed, M/q; 2 scaled, C_M)	2
MoQ0	real	+	moment at zero lift, $(M/q)_0$	0.
CM0	real	+	moment at zero lift, C_{M0} (based on wetted area and fuselage length, $M/q = S\ell C_M$)	0.
dMoQda	real	+	moment slope, $d(M/q)/d\alpha$ (per rad)	0.
dCMda	real	+	moment slope, $C_{M\alpha} = dC_M/d\alpha$ (per rad; based on wetted area and fuselage length, $M/q = S\ell C_M$)	0.
SS_zy	real	+	sideslip angle for zero side force β_{zy} (deg)	0.
SS_max	real	+	sideslip angle for maximum side force $\beta_{\max}$ (deg)	10.
		+	side force	
SET_side	int	+	specification (1 fixed, Y/q; 2 scaled, C_Y)	2
dYoQdb	real	+	side force slope, $d(Y/q)/d\beta$ (per rad)	0.
dCYdb	real	+	side force slope, $C_{Y\beta} = dC_Y/d\beta$ (per rad; based on wetted area, $Y/q = SC_Y$)	0.
		+	yaw moment	
SET_yaw	int	+	specification (1 fixed, N/q; 2 scaled, C_N)	2
NoQ0	real	+	moment at zero lift, $(N/q)_0$	0.
CN0	real	+	moment at zero lift, C_{N0} (based on wetted area and fuselage length, $N/q = S\ell C_N$)	0.
dNoQdb	real	+	moment slope, $d(N/q)/d\beta$ (per rad)	0.
dCNdb	real	+	moment slope, $C_{N\beta} = dC_N/d\beta$ (per rad; based on wetted area and fuselage length, $N/q = S\ell C_N$)	0.
			SET_xxx: fixed (use XoQ) or scaled (use CX); other parameter calculated	

		+	Drag, Standard Model	
		+	forward flight drag	
SET_drag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ	real	+	area $(D/q)_0$	
CD	real	+	coefficient C_{D0} (based on wetted area, $D/q = SC_D$)	0.005
		+	fixtures and fittings	
SET_Dfit	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ_fit	real	+	area $(D/q)_{\rm fit}$	
CD_fit	real	+	coefficient $C_{D{\rm fit}}$ (based on wetted area, $D/q = SC_D$)	0.
		+	rotor-body interference	
SET_Drb	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ_rb(nrotormax)	real	+	area $(D/q)_{rb}$	
CD_rb(nrotormax)	real	+	coefficient C_{Drb} (based on wetted area, $D/q = SC_D$)	0.
		+	vertical drag	
SET_Vdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV	real	+	area $(D/q)_V$	
CDV	real	+	coefficient C_{DV} (based on projected area, $D/q = S_{\rm proj} C_D$)	0.
		+	sideward drag	
SET_Sdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQS	real	+	area $(D/q)_S$	
CDS	real	+	coefficient C_{DS} (based on wetted area, $D/q = SC_D$)	0.
		+	drag variation with angle of attack	
MODEL_drag	int	+	model (0 none, 1 general, 2 quadratic)	2
AoA_Dmin	real	+	angle of attack for fuselage minimum drag $C_{D{\rm min}}$ (deg)	0.
Kdrag	real	+	drag increment K_d, $\Delta C_D = C_{D0} K_d \lvert\alpha_e\rvert^{X_d}$	0.
Xdrag	real	+	drag increment X_d, $\Delta C_D = C_{D0} K_d \lvert\alpha_e\rvert^{X_d}$	2.
		+	transition from forward flight drag to vertical drag	
MODEL_trans	int	+	model (1 input transition angle of attack, 2 calculate for quadratic)	1
AoA_tran	real	+	angle of attack for transition α_t (deg)	25.
		+	Flow Control; $\Delta C_L = C_{L\alpha}(L_{\mu s}\sqrt{C_\mu} + L_{\mu 1} C_\mu + L_{\mu 2} C_\mu^2)$, $\Delta C_{L{\rm max}} = X_\mu C_\mu$, $\Delta C_M = M_\mu C_\mu$, $\Delta C_D = D_\mu C_\mu$	
MODEL_flow	int	+	model (0 none)	0
Lmus	real	+	lift $L_{\mu s}$	0.0
Lmu1	real	+	lift $L_{\mu 1}$	0.0

Lmu2	real	+	lift $L_{\mu 2}$	0.0
Xmu	real	+	maximum lift X_μ	1.0
Mmu	real	+	moment M_μ	0.0
Dmu	real	+	drag D_μ	0.0
Cmu_limit	real	+	flow limit $C_{\mu\text{limit}}$	1.0
		+	Fuselage Group, NDARC Weight Model	
MODEL_body	int	+	model (1 AFDD84, 2 AFDD82, 3 other)	1
MODEL_other	int	+	model (1 Boeing, GARTEUR (2 air, 3 hel), 4 Tishchenko, 5 Torenbeek, Raymer (6 transport, 7 gen av), 8 generic)	
KIND_ramp	int	+	AFDD: rear cargo ramp (0 none)	0
fLength_crg	real	+	Boeing: cabin length + ramp length + cg range (fraction fuselage length)	0.6
Vdive	real	+	Boeing or Torenbeek or Raymer: design dive speed V_{dive} (knots)	200.
ndoor	int	+	Raymer: number of cargo doors	0
Pdelta	real	+	Raymer: cabin pressure differential (psi)	8.
Kfus	real	+	generic: factor K_{fus}	0.
XfusW	real	+	exponent $X_{\text{fus}W}$	0.
Xfusn	real	+	exponent $X_{\text{fus}n}$	0.
XfusS	real	+	exponent $X_{\text{fus}S}$	0.
Xfusl	real	+	exponent $X_{\text{fus}\ell}$	0.
fWbody_mar	real	+	body weight for marinization f_{mar} (fraction basic body weight)	0.
fWbody_press	real	+	body weight for pressurization f_{press} (fraction basic body weight)	0.
fWbody_crash	real	+	body weight for crashworthiness f_{cw} (fraction body weight)	0.
fWbody_tfold	real	+	tail fold weight f_{tfold} (fraction tail (AFDD84 or other) or body (AFDD82) weight)	0.
fWbody_wfold	real	+	wing fold weight f_{wfold} (fraction wing+tip (AFDD84 or other) or body+tailfold (AFDD82) weight)	0.

AFDD84 (UNIV) is universal body weight model, for tiltrotor and tiltwing as well as for helicopters
AFDD82 (HELO) is helicopter body weight model, should not be used for tiltrotor or tiltwing
dive speed: V_{max} = SLS max speed, Vdive $= 1.25 V_{\text{max}}$
fLength_crg = $(\ell_c + \ell_r + \Delta CG)/\ell_{\text{body}} \cong$ 1.0 for tandem, 0.3-0.6 for single main rotor (0.7-0.8 with ramp)
typically fWbody_crash = 0.06
typically fWbody_tfold = 0.30 (AFDD84 or other) or 0.05 (AFDD82) for folding tail

		+	Custom Weight Model	
WtParam_fuse(8)	real	+	parameters	0.

Chapter 21

Structure: LandingGear

Variable	Type		Description	Default
		+	Landing Gear	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Geometry	
loc_gear	Location	+	landing gear location	
d_gear	real	+	distance from bottom of landing gear to WL_gear d_{LG}	0.
place	int	+	placement (1 located on body, 2 located on wing)	1
KIND_LG	int	+	retraction (0 fixed, 1 retracts)	1
speed	real	+	retraction speed (CAS or TAS, knots)	

landing gear location: with HAGL (FltState) determines rotor height above ground level
height rotor = landing gear above ground + hub above landing gear = HAGL + (WL_hub–WL_gear+d_gear)
place: used for weight (fuselage and wing)

Variable	Type		Description	Default
		+	Aerodynamics	
MODEL_aero	int	+	model (0 none, 1 standard)	1
		+	Weight	
		+	alighting gear group	
MODEL_weight	int	+	alighting gear group model (0 input, 1 NDARC, 2 custom)	1
		+	weight increment	
dWLG	real	+	basic landing gear	0.
dWLGret	real	+	retraction	0.
dWLGcrash	real	+	crashworthiness	0.

		+	Technology Factors	
TECH_LG	real	+	basic landing gear χ_{LG}	1.0
TECH_LGret	real	+	retraction $\chi_{LG\mathrm{ret}}$	1.0
TECH_LGcrash	real	+	crashworthiness $\chi_{LG\mathrm{cw}}$	1.0

---
weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

		+	Drag, Standard Model	
DoQ	real	+	drag area extended, D/q	

		+	Landing Gear Group, NDARC Weight Model	
MODEL_LG	int	+	model (1 fraction, 2 parametric rotary wing, 3 parametric fixed wing)	2
nLG	int	+	number of landing gear assemblies N_{LG}	3
fWLG_basic	real	+	basic landing gear weight f_{LG} (fraction maximum takeoff weight)	0.0325
fWLG_ret	real	+	landing gear weight for retraction $f_{LG\mathrm{ret}}$ (fraction basic weight)	0.08
fWLG_crash	real	+	landing gear weight for crashworthiness $f_{LG\mathrm{cw}}$ (fraction basic+retraction weight)	0.14

---
only MODEL_LG=fraction uses fWLG_basic
typically fWLG_basic = 0.0325 (fraction method)
typically fWLG_ret = 0.08, fWLG_crash = 0.14

		+	Custom Weight Model	
WtParam_gear(8)	real	+	parameters	0.

Chapter 22

Structure: Rotor

Variable	Type		Description	Default
		+	Rotor	
title	c*100	+	title	
notes	c*1000	+	notes	
config	c*32	+	Configuration	'main'

configuration designation: principal designation required, rest identify special characteristics
principal designation = 'main', 'tail', 'prop'
antitorque = 'antiQ', 'auxT'
twin rotor = 'coaxial', 'tandem', 'tiltrotor' (keyword = tan, coax, tilt)
others = 'variable diameter', 'stop', 'ducted fan', 'reaction drive', 'multirotor' (keyword = var, stop, duct, react, multi)
principal designation determines where weight put in weight statement, and designates main rotors (isMainRotor)
separately specify appropriate performance and weight models
multiple rotor configurations have special options for geometry and performance
options defined by variables SET_geom, MODEL_twin, MODEL_int_twin
antitorque or aux thrust rotor has special options for sizing
options defined by variables SET_rotor, fThrust, Tdesign
reaction drive still requires propulsion group

Variable	Type		Description	Default
		+	Propulsion group	
kPropulsion	int	+	group number	1
KIND_xmsn	int	+	drive system branch (1 primary, 0 dependent)	1
Vtip_ref(ngearmax)	real	+	reference tip speed	
INPUT_gear	int	+	gear ratio input for dependent branch (1 Vtip_ref, 2 gear)	1
gear(ngearmax)	real	+	gear ratio $r = \Omega_{\rm dep}/\Omega_{\rm prim}$ (ratio rpm to rpm of primary rotor)	1.0
		+	Reaction drive	
r_react	real	+	effective radial station of force (fraction Radius)	1.0

---

drive system branch: only one primary rotor per propulsion group
  tip speed and gear ratio required for each drive system state
  primary: specify Vtip_ref and default tip speeds; $V_{\text{tip}-\text{hover}}$ = Vtip_ref(1)
  dependent: specify gear ratio, or specify Vtip_ref and calculate gear (depend on rotor radius)
    can not specify gear ratio if sizing changes dependent rotor V_{tip} (SET_rotor)
  if size task changes Vtip_ref(1), then rVtip_ref used to change Vtip_ref(n) for n>1
variable speed transmission: for drive system state STATE_gear_var, gear ratio factor f_{gear} (control) included
  when evaluate rotational speed of dependent rotor

reaction drive requires one and only one propulsion system (engine group or jet group)

		+	Default rotor tip speeds (primary rotor)	
INPUT_Vtip	int	+	input form (1 tip speed, 2 hover V_{tip} and rpm ratio)	1
		+	function of flight speed	
nVrpm	int	+	number of speeds (1 constant; ≥ 2 piecewise linear, maximum nvelmax)	1
Vrpm(nvelmax)	real	+	speeds (CAS or TAS)	
		+	tip speed	
Vtip_cruise	real	+	cruise	
Vtip_man	real	+	maneuvering flight	
Vtip_oei	real	+	OEI	
Vtip_xmsn	real	+	transmission sizing	
Vtip(nvelmax)	real	+	function of flight speed	
		+	rpm ratio ($V_{\text{tip}}/V_{\text{tip}-\text{hover}}$)	
fRPM_cruise	real	+	cruise	1.
fRPM_man	real	+	maneuvering flight	1.
fRPM_oei	real	+	OEI	1.
fRPM_xmsn	real	+	transmission sizing	1.
fRPM(nvelmax)	real	+	function of flight speed	1.

---

default rotor tip speeds (including conversion): selectable by SET_Vtip of FltState
  only for primary rotor; V_{tip} calculated from gear(state) for dependent branch

---

		+	Drive system torque limit	
SET_limit_rs	int	+	rotor shaft (0 input, 1 fraction power, 2 fraction drive system limit)	1
Plimit_rs	real	+	rotor shaft power limit $P_{RS\text{limit}}$	
fPlimit_rs	real	+	rotor shaft power limit factor	1.

---

drive system torque limit: Size%SET_limit_ds = input (use Plimit_rs) or calculated (from fPlimit_rs)
SET_limit_ds='input': Plimit_rs input
SET_limit_ds≠'input': from rotor power required at transmission sizing flight conditions (DESIGN_xmsn)
rotor shaft: options for SET_limit_ds≠'input'
SET_limit_rs=0: Plimit_rs
SET_limit_rs=1: fPlimit_rs × (rotor P_{req})
SET_limit_rs=2: fPlimit_rs $\times P_{DS\text{limit}}$
rotor shaft power limit: corresponds to one rotor
can be used for max effort in flight state (max_quant='Q margin')
can be used for max gross weight in flight condition or mission (SET_GW='maxQ' or 'maxPQ')
always check and print whether exceed torque limit

		+	Parameters	
diskload	real	+	disk loading (lb/ft^2 or N/m^2)	
fArea	real	+	fraction rotor area for reference disk area f_A	
fDGW	real	+	fraction DGW f_W (for disk loading and blade loading)	
fThrust	real	+	thrust factor (antitorque or aux thrust rotor)	1.0
Radius	real	+	radius R	
CWs	real	+	blade loading C_W/σ (thrust-weighted)	
sigma	real	+	solidity $\sigma = Nc/\pi R$ (thrust-weighted)	
Tdesign	real	+	thrust for antitorque or aux thrust rotor	
Pdesign	real	+	power for antitorque or aux thrust rotor	
Ndesign	real	+	rotor speed (rpm) at Pdesign	
SET_thrust	int	+	rotor thrust for disk loading and blade loading (0 default; 1 fDGW*DGW, 2 fThrust*Tdesign)	0

---

rotor disk loading = T/A; aircraft disk loading = W_D/A_{ref}, $A_{\text{ref}} = \sum(f_A A)$
$W = f_W W_D$ (main rotor) or fThrust*Tdesign (antitorque or aux thrust rotor); can specify using SET_thrust
Tdesign and Pdesign obtained from thrust design conditions and missions (DESIGN_thrust)

if rotor sized from disk loading (SET_rotor='DL+xx+xx'), area = T/diskload
if SET_rotor specify 'Vtip', use Vtip_ref(1)
if SET_rotor not specify 'Vtip', calculate Vtip_ref(1), and then Vtip_ref for dependent rotors
if SET_rotor='CWs+xx+xx', then C_W/σ from fDGW*DGW, takeoff condition, Vtip_ref, and thrust-weighted solidity

for antitorque or aux thrust rotor, need design conditions and missions (DESIGN_thrust) to identify Tdesign
otherwise use fDGW and design gross weight
Tdesign and Pdesign generally calculated (identified as input so inherited by next case)

		+	Geometry	
SET_geom	c*12	+	position (standard, tiltrotor, coaxial, tandem, tailrotor, multicopter)	'std'
KIND_TRgeom	int	+	tiltrotor (1 from clearance, 2 at wing tip, 3 at wing panel edge)	0
		+	twin rotors	
fRadius	real	+	ratio rotor radius to that of other rotor	1.0
otherRotor	int	+	other rotor number	
positionOfRotor	int	+	rotor position (+1/–1 for right/left, lower/upper, front/aft)	0
WingForRotor	int	+	wing number	1
PanelForRotor	int	+	wing panel number	1
clearance_fus	real	+	tiltrotor clearance between rotor and fuselage d_{fus}	0.6
fclearance_fus	real	+	tiltrotor clearance factor	1.0
sep_coaxial	real	+	coaxial rotor separation s (fraction Diameter)	0.08
overlap_tandem	real	+	tandem rotor overlap o (fraction Diameter)	0.25
		+	tail rotor	
mainRotor	int	+	main rotor number	1
fRadius_tr	real	+	radius scale factor	1.0
clearance_tr	real	+	clearance between tail rotor and main rotor d_{tr}	0.5
		+	multicopter	
ang_multicopter	real	+	angle ψ (clockwise from forward, deg)	0.
len_multicopter	real	+	arm length ℓ (fraction Radius)	1.5
		+	variable diameter rotor	
SET_VarDiam	int	+	set diameter (1 conversion schedule, 2 function speed)	
fRcruise	real	+	ratio cruise radius to hover radius (variable diameter only)	
		+	rotor stopped as wing	
StopAsWing	int	+	wing number (0 not)	0

SET_geom: calculation override part of location input
- SET_geom='tiltrotor': calculate lateral position (BL)
  - KIND_TRgeom=clearance: from WingForRotor, Width_fus, clearance_fus, fclearance_fus
  - KIND_TRgeom=wing tip: from WingForRotor, wing span
  - KIND_TRgeom=wing panel edge: from WingForRotor, PanelForRotor, panel edge and wing span
  - positionOnRotor specifies right or left position
  - BL or YoL in loc_pylon, loc_pivot, loc_naccg is relative calculated loc_rotor BL
- SET_geom='coaxial': calculate position from sep_coaxial
  - same sep_coaxial for otherRotor, positionOnRotor specifies lower or upper position
  - loc_rotor (SL,BL,WL or XoL,YoL,ZoL) is midpoint between hubs
  - loc_pylon (SL,BL,WL or XoL,YoL,ZoL) is relative calculated loc_rotor
- SET_geom='tandem': calculate longitudinal position (SL) from overlap_tandem
  - same overlap_tandem for otherRotor, positionOnRotor specifies front or aft position
  - loc_rotor (SL or XoL only) is midpoint between hubs
  - loc_pylon SL or XoL is relative calculated loc_rotor
- SET_geom='tailrotor': calculate longitudinal position (SL) from clearance_tr, mainRotor
  - loc_pylon SL or XoL is relative calculated loc_rotor
- SET_geom='multicopter': calculate longitudinal and lateral position from ang_multicopter, len_multicopter
  - loc_rotor (SL,BL or XoL,YoL) is center of rotors
  - loc_pylon (SL,BL,WL or XoL,YoL,ZoL) is relative calculated loc_rotor
  - ang_multicopter also used for Aircraft%config='multicopter' to define control
- if rotor number ≤ 2 and positionOnRotor=0: first rotor is right/lower/front, second rotor is left/upper/aft

sizing:
- if SET_rotor='ratio', Radius=fRadius*Radius(otherRotor); otherRotor not SET_rotor='ratio'

twin rotors: config identify as twin rotor

antitorque: config identify as antitorque rotor
- if SET_rotor='scale', Radius=fRadius_tr*(main rotor Radius)*function(DiskLoad)

variable diameter: Radius is hover or reference radius; can be commanded by aircraft controls
- conversion schedule: R =Radius in hover and helicopter mode ($V \leq V_{\mathrm{conv-hover}}$)
  - R =Radius*fRcruise in cruise mode ($V \geq V_{\mathrm{conv-cruise}}$); linear with V in conversion mode
- function of speed: use nVdiam, fdiam, Vdiam to calculate R

stoppable rotor: zero rotor flapping, forces, and power when stopped
stopped (FltAircraft%STOP_rotor=1) uses stopped rotor hub and blade drag
stopped and stowed (FltAircraft%STOP_rotor=2) uses stowed rotor hub drag
stopped as wing (FltAircraft%STOP_rotor=3) uses wing aero (wing number StopAsWing) with zero hub drag

---

		+	Geometry	
rotate	int	+	direction of rotation (1 counter-clockwise, –1 clockwise)	1
nBlade	int	+	number of blades N	
		+	planform and twist	
SET_chord	int	+	chord distribution (1 linear from fTWsigma, 2 linear from taper, 3 nonlinear from fchord)	1
fTWsigma	real	+	ratio thrust-weighted solidity to geometric solidity $f = \sigma_t/\sigma_g$	1.
taper	real	+	taper ratio t (tip chord/root chord)	1.
SET_twist	int	+	twist distribution (1 linear from twistL, 2 nonlinear from twist)	1
twistL	real	+	linear twist θ_L (deg, root to tip)	-10.
nprop	int	+	number of radial stations (maximum nrmax)	2
rprop(nrmax)	real	+	radial stations ($r_{\rm root}/R$)	
fchord(nrmax)	real	+	chord distribution $c(r)/c_{\rm ref}$	1.
twist(nrmax)	real	+	twist $\theta_{tw}(r)$ (deg)	
		+	flap dynamics	
KIND_hub	int	+	hub type (1 articulated, 2 hingeless)	1
flapfreq	real	+	first flapwise natural frequency ν (per-rev at hover tip speed)	1.04
conefreq	real	+	coning natural frequency ν (0. to use flapfreq)	0.
gamma	real	+	blade Lock number γ	8.
precone	real	+	precone β_p (deg)	0.
delta3	real	+	pitch-flap coupling δ_3 (deg)	0.
		+	aerodynamics	
dclda	real	+	blade section 2D lift-curve slope $a = c_{\ell\alpha}$ (per-rad)	5.7
tiploss	real	+	tip loss factor B (lift zero from BR to tip)	0.97
xroot	real	+	root cutout ($r_{\rm root}/R$)	0.1

SET_chord: use one of fTWsigma, taper, or fchord(r); others calculated (including root cutout)
fTWsigma = sigma_tw/sigma_geom
from fTWsigma: calculate equivalent linear taper, and $f_c = c/c_{\rm ref}$

from taper (linear): calculate fTWsigma, and $f_c = c/c_{\text{ref}}$
from fchord(r): integrate for c_g and c_t, fTWsigma= c_t/c_g, calculate taper, f_c = scaled fchord

SET_twist: use one of twistL or twist(r); other calculated
for nonlinear distribution, twist relative $0.75R$ obtained from input

flap frequency and Lock number are used for flap dynamics and hub moments due to flap
specified for hover radius and rotational speed
KIND_hub determines how flap frequency and hub moment spring vary with rotor speed and R
weight models can have separate blade and hub values for flap frequency

blade Lock number gamma: for SLS density, $a = 5.7$, thrust-weighted chord
SET_Iblade determines whether Lock number input or calculated

---

		+	Geometry (for graphics)	
thick	real	+	blade thickness-to-chord ratio	0.12
		+	Blade element theory solution	
		+	integration	
mr	int	+	number of radial stations (xroot to 1; maximum mrmax)	4
mpsi	int	+	number of azimuth angles (maximum mpsimax)	8
		+	Geometry	
loc_rotor	Location	+	hub location	
loc_pylon	Location	+	pylon location	
loc_pivot	Location	+	pivot location	
loc_naccg	Location	+	nacelle cg location	
direction	c*16	+	nominal orientation ('+x', '–x', '+y', '–y', '+z', '–z'; 'main' (–z), 'tail' (ry), 'prop' (x))	'main'
KIND_tilt	int	+	shaft control (0 fixed shaft, 1 incidence, 2 cant, 3 both controls)	0
		+	orientation of rotor shaft	
incid_hub	real	+	incidence θ_h (deg)	0.
cant_hub	real	+	cant angle ϕ_h (deg)	0.
		+	orientation of pivot axes	
dihedral_pivot	real	+	pivot dihedral angle ϕ_p (deg)	
pitch_pivot	real	+	pivot pitch angle θ_p (deg)	
sweep_pivot	real	+	pivot sweep angle ψ_p (deg)	

		+	reference shaft control	
incid_ref	real	+	incidence $i_{\rm ref}$ (deg)	0.
cant_ref	real	+	cant angle $c_{\rm ref}$ (deg)	0.
		+	moving weight for cg shift	
SET_Wmove	int	+	weight (1 wing tip weight, 2 W_{gbrs}, 3 W_{gbrs} and W_{ES})	1
fWmove	real	+	fraction moving weight	1.
dz_hub(3)	real	+	hub position increment due to tilt $\Delta z^F_{\rm hub}$ (SL/BL/WL)	0.

loc_naccg, loc_pivot, orientation of pivot axes, and reference shaft control angles not used for KIND_tilt=fixed shaft
for tiltrotor, locations and orientation specified in helicopter mode, so incid_ref = 90
SET_Wmove: cg shift calculated using incidence and cant rotation of loc_naccg relative loc_pivot
moving weight fWmove*Wmove, Wmove = Wtip_total/nRotorOnWing or $w/N_{\rm rotor}$
$w = W_{gbrs}$ (drive system) or $W_{gbrs} + \sum(W_{ES})$ (drive system and engine system)

		+	Controls	
KIND_control	int	+	rotor control mode (1 thrust and TPP, 2 thrust and NFP, 3 pitch and TPP, 4 pitch and NFP)	1
KIND_cyclic	int	+	cyclic input (1 tip-path-plane tilt, 2 hub moment, 3 lift offset)	1
KIND_coll	int	+	collective input (1 thrust, 2 C_T/σ)	2
SCALE_coll	int	+	scale collective T matrix (0 for none)	1
		+	collective (magnitude of thrust vector)	
INPUT_coll	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_coll(ncontmax,nstatemax)	real	+	control matrix	
nVcoll	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
coll(nvelmax)	real	+	values	
Vcoll(nvelmax)	real	+	speeds (CAS or TAS)	
		+	longitudinal cyclic (tip-path plane tilt or no-feathering plane tilt)	
INPUT_lngcyc	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_lngcyc(ncontmax,nstatemax)	real	+	control matrix	
nVlngcyc	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
lngcyc(nvelmax)	real	+	values	
Vlngcyc(nvelmax)	real	+	speeds (CAS or TAS)	

		+	lateral cyclic (tip-path plane tilt or no-feathering plane tilt)	
INPUT_latcyc	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_latcyc(ncontmax,nstatemax)	real	+	control matrix	
nVlatcyc	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
latcyc(nvelmax)	real	+	values	
Vlatcyc(nvelmax)	real	+	speeds (CAS or TAS)	
		+	incidence i (nacelle tilt)	
INPUT_incid	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_incid(ncontmax,nstatemax)	real	+	control matrix	
nVincid	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
incid(nvelmax)	real	+	values	
Vincid(nvelmax)	real	+	speeds (CAS or TAS)	
		+	cant c	
INPUT_cant	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_cant(ncontmax,nstatemax)	real	+	control matrix	
nVcant	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
cant(nvelmax)	real	+	values	
Vcant(nvelmax)	real	+	speeds (CAS or TAS)	
		+	diameter $f_{\rm diam}$ (variable diameter only)	
INPUT_diam	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_diam(ncontmax,nstatemax)	real	+	control matrix	
nVdiam	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
fdiam(nvelmax)	real	+	values	
Vdiam(nvelmax)	real	+	speeds (CAS or TAS)	
		+	gear ratio factor $f_{\rm gear}$ (variable speed transmission only)	
INPUT_fgear	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_fgear(ncontmax,nstatemax)	real	+	control matrix	
nVfgear	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
fgear(nvelmax)	real	+	values	
Vfgear(nvelmax)	real	+	speeds (CAS or TAS)	

		+	reaction drive net force F_{react}	
INPUT_Freact	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_Freact(ncontmax,nstatemax)	real	+	control matrix	
nVFreact	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
Freact(nvelmax)	real	+	values	
VFreact(nvelmax)	real	+	speeds (CAS or TAS)	

---

aircraft controls connected to individual controls of component, $c = Tc_{AC} + c_0$

for each component control, define matrix T (for each control state) and value c_0

flight state specifies control state, or that control state obtained from conversion schedule

c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)

by connecting aircraft control to component control, flight state can specify component control value

initial values if control is connected to trim variable; otherwise fixed for flight state

pylon moves with rotor; nontilting part is engine nacelle

		+	Trim Targets
		+	rotor lift
nVlift	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)
Klift(nvelmax)	real	+	target
Vlift(nvelmax)	real	+	speeds (CAS or TAS)
		+	rotor propulsive force
nVprop	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)
Kprop(nvelmax)	real	+	target
Vprop(nvelmax)	real	+	speeds (CAS or TAS)

---

target definition determined by Aircraft%trim_quant

Klift can be fraction total aircraft lift, lift, C_L/σ, or C_T/σ

Kprop can be fraction total aircraft drag, propulsive force $-X$, $-C_X/\sigma$, or $-X/q$)

		+	Rotor Thrust Capability (C_T/σ vs μ)	
		+	sustained	
nsteady	int	+	number of points (maximum 20)	16
mu_steady(20)	real	+	advance ratio	
CTs_steady(20)	real	+	C_T/σ	
		+	transient	
ntran	int	+	number of points (maximum 20)	16
mu_tran(20)	real	+	advance ratio	
CTs_tran(20)	real	+	C_T/σ	
		+	equation, $C_T/\sigma = K_0 - K_1\mu^2$	
K0_limit	real	+	constant K_0	0.17
K1_limit	real	+	constant K_1	0.25

CTs_steady, CTS_tran used to calculate rotor thrust margin, which available for max effort or trim
defaults used if CTs(1)=0.
default CTs_steady = .170,.168,.161,.149,.131,.109,.084,.050,.049,.048,.047,.046,.045,.044,.043,.042
default CTs_tran = .200,.197,.190,.177,.156,.135,.110,.080,.075,.070,.065,.060,.055,.050,.045,.040
default mu_steady = 0.,.10,.20,.30,.40,.50,.60,.70,.71,.72,.73,.74,.75,.76,.77,.78
default mu_tran = 0.,.10,.20,.30,.40,.50,.60,.70,.72,.74,.76,.78,.80,.82,.84,.86

		+	Performance	
MODEL_perf	int	+	power model (1 standard, 2 table model)	1
MODEL_Ftpp	int	+	inplane forces, tip-path plane axes (1 neglect, 2 blade-element theory)	2
MODEL_Fpro	int	+	inplane forces, profile (1 simplified, 2 blade element theory, 3 neglect)	2

if thrust and TPP command, and neglect inplane forces relative TPP, then pitch control angles not required

		+	Interference	
MODEL_int	int	+	model (0 none, 1 standard, 2 with transition)	1
		+	transition	
Vint_low	real	+	low velocity (knots)	0.
Vint_high	real	+	high velocity (knots)	0.

Kint=0 to suppress interference at component; MODEL_int=0 for no interference at all
with transition: interference factors linearly vary from Kint at $V \leq$ Vint_low to 0 at $V \geq$ Vint_high

		+	Geometry	
SET_aeroaxes	int	+	hub/pylon aerodynamic axes (0 input pitch, 1 helicopter, 2 propeller or tiltrotor)	1
pitch_aero	real	+	pitch relative shaft axes $\theta_{\rm ref}$, $C^{BS} = Y_{-\theta_{\rm ref}}$	0.
SET_Spylon	int	+	pylon wetted area (1 fixed, input Swet; 2 scaled, W_{gbrs}; 3 scaled, W_{gbrs} and W_{ES})	2
Swet_pylon	real	+	area $S_{\rm pylon}$	0.
kSwet_pylon	real	+	factor, $k = S_{\rm pylon}/(w/N_{\rm rotor})^{2/3}$ (Units_Dscale)	1.0
SET_Sduct	int	+	duct area (1 fixed, input S_duct; 2 scaled, from fLength_duct)	2
S_duct	real	+	area $S_{\rm duct}$	0.
fLength_duct	real	+	duct length (fraction rotor radius)	1.2
SET_Sspin	int	+	spinner wetted area (1 fixed, input Swet; 2 scaled, from fSwet)	2
Swet_spin	real	+	area $S_{\rm spin}$	0.
fSwet_spin	real	+	factor, $k = S_{\rm spin}/A_{\rm spin}$	1.0
fRadius_spin	real	+	spinner radius (fraction rotor radius)	0.

only SET_aeroaxes=input uses pitch_aero; pitch_aero=180 for helicopter, 90 for propeller

SET_Spylon, pylon wetted area: input (use Swet_pylon) or calculated (from kSwet_pylon)
units of kSwet are ft^2/lb$^{2/3}$ or m^2/kg$^{2/3}$
$w = W_{gbrs}$ (drive system) or $W_{gbrs} + \sum W_{ES}$ (drive system and engine system)
pylon wetted area used for pylon drag
rotor pylon must be consistent with engine group nacelle

SET_Sduct, duct area: input (use S_duct) or calculated (from fLength_duct)
$S_{\rm duct} = (2\pi R)\ell_{\rm duct}$, $\ell_{\rm duct}$ =fLength_duct*R; used for drag (wetted area $2S_{\rm duct}$) and weight

SET_Sspin, spinner wetted area: (use Swet_spin) or calculated (from fSwet_spin)
$A_{\rm spin} = \pi R_{\rm spin}^2$ = spinner frontal area (from fRadius_spin*R); spinner radius used for drag and weight

		+	Drag	
MODEL_drag	int	+	model (0 none, 1 standard)	1
Idrag	real	+	incidence angle for helicopter nominal drag (deg; 0 for not tilt)	0.
		+	Download and blockage	
MODEL_download	int	+	model (0 none, 1 blockage, 2 download, 3 both)	0
download	real	+	download $DL = \Delta T/T$	0.
blockage	real	+	blockage $B = \Delta T/T$	0.
muDL	real	+	advance ratio μ_{DL} (0. for no correction)	0.16
zDL	real	+	height above ground $(z_g/D)_{DL}$ (fraction diameter, 0. for no correction)	0.41
aDL	real	+	forward flight constant a_{DL}	1.04
bDL	real	+	ground effect constant b_{DL}	0.23

download: rotor induced and profile power evaluated at thrust increased by $f_{DL} = 1/(1 - \Delta T/T)$
blockage: force acting on aircraft includes $f_B = (\Delta T/T)T$ opposing thrust
download DL and blockage B are for hover, out of ground effect
download and blockage zero for $\mu > \mu_{DL}$ or $z_g/D < (z_g/D)_{DL}$

		+	Weight	
		+	rotor group (or empennage or propulsion group)	
MODEL_weight	int	+	model (0 input, 1 NDARC, 2 custom)	1
		+	weight increment	
dWblade	real	+	blade	0.
dWhub	real	+	hub and hinge	0.
dWshaft	real	+	inter-rotor shaft	0.
dWspin	real	+	fairing/spinner	0.
dWrfold	real	+	blade fold	0.
dWtr	real	+	tail rotor	0.
dWaux	real	+	auxiliary thrust	0.
dWrsupt	real	+	rotor support structure	0.
dWduct	real	+	duct	0.

SET_Iblade	int	+	blade moment of inertia (0 from Lock number, 1 from blade wt, 2 tip wt from Lock number, 3 tip wt from AI)	1
AI	real	+	autorotation index $KE/P = \frac{1}{2} N_{\text{blade}} I_{\text{blade}} \Omega^2/P$ (sec)	3.0
Wblade_tip	real	+	tip weight (per blade)	0.
rWblade_tip	real	+	location tip weight (fraction blade radius)	0.9
fWblade_tip	real	+	distributed weight for centrifugal force (fraction Wblade_tip)	1.0
rblade	real	+	radius of gyration for distributed mass (fraction blade radius)	0.6
xWblade	real	+	blade weight (fraction total tail rotor or auxiliary thrust rotor weight)	0.55
		+	Technology Factors	
TECH_blade	real	+	blade weight χ_{blade}	1.0
TECH_hub	real	+	hub and hinge weight χ_{hub}	1.0
TECH_shaft	real	+	inter-rotor shaft χ_{shaft}	1.0
TECH_spin	real	+	fairing/spinner weight χ_{spin}	1.0
TECH_rfold	real	+	blade fold weight χ_{fold}	1.0
TECH_tr	real	+	tail rotor weight χ_{tr}	1.0
TECH_aux	real	+	auxiliary thrust weight χ_{at}	1.0
TECH_rsupt	real	+	rotor support structure weight χ_{supt}	1.0
TECH_duct	real	+	duct weight χ_{duct}	1.0

---

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

blade weight: $W_{\text{blade}} = \chi_{\text{blade}} w_{\text{blade}} + dW_{\text{blade}} + (1+f) W_{\text{tip}} N_{\text{blade}}$
SET_Iblade: calculate blade moment of inertia Iblade
0 from Lock number gamma, independent of blade weight
1 from blade weight
2 from Lock number gamma, tip weight Wblade_tip calculated from Iblade
3 from autorotation index AI, tip weight Wblade_tip calculated from Iblade
for tail rotor or aux thrust weight model (MODEL_config = 2 or 3), blade weight W_{blade} = xWblade*W_{tr} or xWblade*W_{at}

rotor weight = blade + hub + spinner + fold + shaft + support + duct
rotor config determines where weight put in weight statement
main rotor: rotor group
tail rotor: empennage group (tail rotor)
propeller: propulsion group (propeller/fan installation)

		+	Rotor Induced Power, Standard Energy Performance Method	
MODEL_ind	int	+	model (0 none, 1 constant, 2 standard)	2
		+	induced velocity factors (ratio to momentum theory induced velocity)	
Ki_hover	real	+	hover $\kappa_{\rm hover}$	1.12
Ki_climb	real	+	axial climb $\kappa_{\rm climb}$	1.08
Ki_prop	real	+	axial cruise (propeller) $\kappa_{\rm prop}$	2.0
Ki_edge	real	+	edgewise flight (helicopter) $\kappa_{\rm edge}$	2.0
		+	variation with thrust	
CTs_Hind	real	+	$(C_T/\sigma)_{\rm ind}$ for hover κ_h variation	0.08
kh1	real	+	coefficient k_{h1} for κ_h	0.
kh2	real	+	coefficient k_{h2} for κ_h	0.
Xh2	real	+	exponent X_{h2} for κ_h	2.
CTs_Pind	real	+	$(C_T/\sigma)_{\rm ind}$ for axial κ_p variation	0.08
kp1	real	+	coefficient k_{p1} for κ_p	0.
kp2	real	+	coefficient k_{p2} for κ_p	0.
Xp2	real	+	exponent X_{p2} for κ_p	2.
CTs_Tind	real	+	$(C_T/\sigma)_{\rm ind}$ for edgewise κ_e variation	0.08
kt1	real	+	coefficient k_{t1} for κ_e	0.
kt2	real	+	coefficient k_{t2} for κ_e	0.
Xt2	real	+	exponent X_{t2} for κ_e	2.
		+	variation with shaft angle	
kpa	real	+	coefficient $k_{p\alpha}$ for κ_p	0.
Xpa	real	+	exponent $X_{p\alpha}$ for κ_p	2.
		+	variation with propulsive force	
kpx	real	+	coefficient k_{px} for κ_p	0.
Xpx	real	+	exponent X_{px} for κ_p	1.
		+	axial flight transition	
Maxial	real	+	constant $M_{\rm axial}$ from hover to climb	1.176
Xaxial	real	+	exponent $X_{\rm axial}$ from hover to climb	0.65
mu_axtran	real	+	advance ratio $\mu_{z\rm tran}$ from hover to axial	0.
		+	variation with axial velocity	
mu_prop	real	+	advance ratio $\mu_{z\rm prop}$ for Ki_prop	1.0
ka1	real	+	coefficient k_{a1} for $\kappa(\mu_z)$ (linear)	0.
ka2	real	+	coefficient k_{a2} for $\kappa(\mu_z)$ (quadratic)	0.

ka3	real	+	coefficient k_{a3} for $\kappa(\mu_z)$	0.
Xa	real	+	exponent X_a for $\kappa(\mu_z)$	4.5
		+	variation with edgewise velocity	
MODEL_edge	int	+	model for edgewise κ relative axial κ (0 replace, 1 sum)	0
mu_edge	real	+	advance ratio μ_{edge} for Ki_edge	0.35
ke1	real	+	coefficient k_{e1} for $\kappa(\mu)$ (linear)	0.8
ke2	real	+	coefficient k_{e2} for $\kappa(\mu)$ (quadratic)	0.
ke3	real	+	coefficient k_{e3} for $\kappa(\mu)$	1.
Xe	real	+	exponent X_e for $\kappa(\mu)$	4.5
kea	real	+	variation with rotor drag $k_{e\alpha}$	0.
		+	variation with lift offset	
ko1	real	+	coefficient k_{o1} for f_{off}	0.
ko2	real	+	factor k_{o2} for f_{off}	8.
Ki_min	real	+	minimum $\kappa_{\min}$	1.
Ki_max	real	+	maximum $\kappa_{\max}$	10.

MODEL_ind=constant uses only Ki_hover, Ki_prop, Ki_edge
nonzero values of Ki in FltState supersede calculated value

		+	Climb power	
MODEL_climb	int	+	model (0 for no climb power increment, 1 vertical, 2 edgewise, 3 both)	0
		+	vertical flight	
nclimb_vert	int	+	number of climb values (maximum 20)	
Vclimb_vert(20)	real	+	climb speed V_c/v_h	
fclimb_vert(20)	real	+	climb power factor f	
		+	edgewise forward flight	
nclimb_edge	int	+	number of climb values (maximum 20)	
Vclimb_edge(20)	real	+	climb speed V_c/v_h	
fclimb_edge(20)	real	+	climb power factor f	

climb power factor $f(V_c/v_h)$ gives $P_{\text{climb}} - P_{\text{level}} = TV_cf$
including TV_c and effect of climb on induced and profile power
intended for use with table model for level flight power

		+	Momentum theory	
MODEL_grad	int	+	inflow gradient in forward flight (0 none, 1 White and Blake, 2 Coleman and Feingold)	1
fGradx	real	+	longitudinal gradient factor f_x	1.
fGrady	real	+	lateral gradient factor f_y	1.
fGradm	real	+	hub moment inflow gradient factor f_m	1.
		+	Ground effect	
MODEL_GE	int	+	model (0 none, 1 Cheeseman, 2 BE Cheeseman, 3 Law, 4 Hayden, 5 Zbrozek, 6 Maryland, 7 T table, 8 P table)	3
Cge	real	+	effective height correction C_g	1.
		+	table	
KIND_GEtable	int	+	table kind (2 2D, 3 3D)	2
nCTsGE	int	+	number of C_T/σ values (maximum ngetabmax)	0
nhGE	int	+	number of h/D values (maximum ngetab2max)	0
nMtipGE	int	+	number of $M_{\rm tip}$ values (maximum ngetab2max)	0
CTsGE(ngetabmax)	real	+	blade loading C_T/σ	
hGE(ngetab2max)	real	+	rotor height above ground h/D	
MtipGE(ngetab2max)	real	+	rotor tip Mach number $M_{\rm tip}$	
xGE(ngetabmax,ngetab2max)	real	+	ground effect factor $\kappa_g = x(C_T/\sigma, h/D)$ or $f_g = x(C_T/\sigma, h/D)$	
xGE3(ngetabmax,ngetab2max,ngetab2max)	real	+	ground effect factor $\kappa_g = x(C_T/\sigma, h/D, M_{\rm tip})$ or $f_g = x(C_T/\sigma, h/D, M_{\rm tip})$	

MODEL_GE: table options for $\kappa_g = T/T_\infty$ or $f_g = P/P_\infty$
as function of blade loading C_T/σ and rotor height above ground h/D (fraction rotor diameter),
and perhaps tip Mach number $M_{\rm tip}$

Cge: for tiltrotors, typically $C_g = 0.5$; smaller effective height accounting for increased influence of ground compared to isolated rotor

---

		+	Ducted fan	
MODEL_duct	int	+	model (1 specify area ratio, 2 specify thrust ratio)	1
fDuctA	real	+	area ratio f_A (fan area/far wake area)	1.
fDuctT	real	+	thrust ratio f_T (rotor thrust/total thrust)	0.5

fDuctVx	real	+	velocity ratio f_{Vx} (fan edgewise velocity/free stream velocity)	1.
fDuctVz	real	+	velocity ratio f_{Vz} (fan axial velocity/free stream velocity)	1.
eta_duct	real	+	duct efficiency η_D (total pressure loss through duct)	1.

ducted fan model used only if config='duct'

		+	Twin rotors	
MODEL_twin	c*12	+	model (based on config, none, side-by-side, coaxial, tandem, multirotor)	'config'
Kh_twin	real	+	ideal induced velocity correction for hover $\kappa_{h\text{twin}}$	1.00
Kp_twin	real	+	ideal induced velocity correction for propeller $\kappa_{p\text{twin}}$	1.00
Kf_twin	real	+	ideal induced velocity correction for forward flight $\kappa_{f\text{twin}}$	0.85
Cind_twin	real	+	constant C in axial to forward flight transition	1.0
Caxial_twin	real	+	constant C_a in hover to propeller transition	1.0
A_coaxial	real	+	coaxial rotor nonuniform disk loading factor $\bar{\alpha}$	1.05
xh_multi(nrotormax)	real	+	multirotor thrust factor x_h for hover	1.0
xp_multi(nrotormax)	real	+	multirotor thrust factor x_p for propeller	1.0
xf_multi(nrotormax)	real	+	multirotor thrust factor x_f for forward flgiht	1.0

MODEL_twin: 'config', 'none', 'side-by-side' or 'tiltrotor', 'coaxial', 'tandem', or 'multirotor'
'config' must identify rotor as twin or multiple rotors
coaxial: MODEL_twin='coaxial' (use A_coaxial; Kh_twin not used)
or MODEL_twin='tandem' with zero horizontal separation (typically Kh_twin=0.90)
coaxial and tandem: Kf_twin = 0.88 to 0.81 for rotor separation $0.06D$ to $0.12D$
thrust factors x calculated for twin rotors, input for multiple rotors
correction factors and transition constants ($\kappa_{\text{twin}}, C, C_a$) used for twin or multiple rotors

		+	Rotor Profile Power, Standard Energy Performance Method	
		+	Technology factor	
TECH_drag	real	+	profile power χ	1.0
Re_ref	real	+	Reference Reynolds number $Re_{\rm ref}$ (0. for no correction)	0.
X_Re	real	+	exponent for Reynolds number correction $X_{\rm Re}$	0.2
MODEL_basic	int	+	Basic model $c_{d\rm basic}$ (0 none, 1 array, 2 equation)	2
		+	array (c_d vs thrust-weighted C_T/σ)	
ncd	int	+	number of points (maximum 24)	24
CTs_cd(24)	real	+	blade loading	
cd(24)	real	+	drag coefficient	
		+	equation	
CTs_Dmin	real	+	$(C_T/\sigma)_{D\min}$ for minimum profile drag ($\Delta = \lvert C_T/\sigma - (C_T/\sigma)_{D\min}\rvert$)	0.07
d0_hel	real	+	coefficient $d_{0\rm hel}$ in drag, $c_{dh} = d_{0\rm hel} + d_{1\rm hel}\Delta + d_{2\rm hel}\Delta^2 + \Delta c_{d\rm sep}$ (hover/edgewise)	0.009
d1_hel	real	+	coefficient $d_{1\rm hel}$ in drag (hover/edgewise)	0.
d2_hel	real	+	coefficient $d_{2\rm hel}$ in drag (hover/edgewise)	0.5
d0_prop	real	+	coefficient $d_{0\rm prop}$ in drag, $c_{dp} = d_{0\rm prop} + d_{1\rm prop}\Delta + d_{2\rm prop}\Delta^2 + \Delta c_{d\rm sep}$ (axial)	0.009
d1_prop	real	+	coefficient $d_{1\rm prop}$ in drag (axial)	0.
d2_prop	real	+	coefficient $d_{2\rm prop}$ in drag (axial)	0.5
dprop	real	+	variation with shaft angle, coefficient $d_{p\alpha}$ for c_{dp}	0.
Xprop	real	+	variation with shaft angle, exponent $X_{p\alpha}$ for c_{dp}	2.
CTs_sep	real	+	$(C_T/\sigma)_{\rm sep}$ for separation ($\Delta c_{d\rm sep} = d_{\rm sep}(\lvert C_T/\sigma\rvert - (C_T/\sigma)_{\rm sep})^{X_{\rm sep}}$)	0.07
dsep	real	+	factor $d_{\rm sep}$ in drag increment	4.0
Xsep	real	+	exponent $X_{\rm sep}$ in drag increment	3.0
df1	real	+	variation with edgewise velocity, coefficient d_{f1}	0.
df2	real	+	variation with edgewise velocity, coefficient d_{f2}	0.
Xf	real	+	variation with edgewise velocity, exponent X_f	2.
dz1	real	+	variation with axial velocity, coefficient d_{z1}	0.
dz2	real	+	variation with axial velocity, coefficient d_{z2}	0.
Xz	real	+	variation with axial velocity, exponent X_z	2.

---

default array (cd(1)=0.): $C_T/\sigma = 0.$ to 0.23 (uniform increments)

cd = .01100,.01075,.01025,.01000,.01010,.01070,.01050,.00975,.00925,.00926,.00938,.00977,
.01048,.01152,.01336,.01593,.01920,.02381,.03014,.04000,.08000,.16000,.32000,1.0000

nonzero values of cdo in FltState supersede calculated cdmean

---

MODEL_stall	int	+	Stall model $c_{d\text{stall}}$ (0 none)	1
		+	C_T/σ at stall ($\Delta_s = \lvert C_T/\sigma\rvert - (f_s/f_\alpha f_{\text{off}})(C_T/\sigma)_s$, $\Delta c_d = d_{s1}\Delta_s^{X_{s1}} + d_{s2}\Delta_s^{X_{s2}}$)	
nstall	int	+	number of points (maximum 20)	10
mu_stall(20)	real	+	advance ratio V/V_{tip}	
CTs_stall(20)	real	+	$(C_T/\sigma)_s$	
fstall	real	+	constant f_s in stall drag increment	1.0
dstall1	real	+	factor d_{s1} in stall drag increment	2.
dstall2	real	+	factor d_{s2} in stall drag increment	40.
Xstall1	real	+	exponent X_{s1} in stall drag increment	2.0
Xstall2	real	+	exponent X_{s2} in stall drag increment	3.0
		+	variation with lift offset	
do1	real	+	coefficient d_{o1} for f_{off}	0.
do2	real	+	factor d_{o2} for f_{off}	8.
dsa	real	+	variation with rotor drag $d_{s\alpha}$	0.

---

default used if CTs_stall(1)=0.
default CTs_stall = 0.17,0.16,0.15,0.14,0.13,0.12,0.11,0.10,0.10,0.10
default mu_stall = 0.00,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.80

MODEL_comp	int	+	Compressibility model $c_{d\text{comp}}$ (0 none, 1 drag divergence, 2 similarity, 3 tip Mach number)	1
MODEL_comp_ff	int	+	compressibility increment (0 only used for hover or axial flight)	1
		+	similarity model	
fSim	real	+	factor f	1.0
thick_tip	real	+	blade tip thickness-to-chord ratio τ	0.08
		+	drag divergence model ($\Delta_m = M_{at} - M_{dd}$, $\Delta c_d = d_{m1}\Delta_m + d_{m2}\Delta_m^{X_m}$)	
dm1	real	+	coefficient d_{m1} in drag increment	0.056
dm2	real	+	coefficient d_{m2} in drag increment	0.416
Xm	real	+	exponent X_m in drag increment	2.0
		+	drag divergence Mach number ($M_{dd} =$ Mdd0 $-$ Mddcl c_ℓ)	
Mdd0	real	+	M_{dd0} at zero lift	0.88
Mddcl	real	+	derivative with lift $\kappa = \partial M_{dd}/\partial c_\ell$	0.16
		+	tip Mach number model	
dmt	real	+	coefficient d_{mt}	
Mtip_limit	real	+	tip Mach number limit M_{tiplimit}	
CT_limit	real	+	thrust coefficient limit $C_{T\text{limit}}$	
Mtip_ref	real	+	reference tip Mach number M_{tipref}	
MODEL_propeff	int	+	Propulsive force efficiency (0 none)	0
DoQ_ref	real	+	reference propulsive force $(D/q)_{\text{ref}}$	
nCTs_eff	int	+	number of blade loading values (maximum 20)	
nV_eff	int	+	number of rotor velocity values (maximum 20)	
CTs_eff(20)	real	+	blade loading C_T/σ	
V_eff(20)	real	+	rotor velocity V/V_{tip}	
propeff(20,20)	real	+	efficiency for propulsive force increment $\eta(C_T/\sigma, V/V_{\text{tip}})$	

---

propeff: efficiency η gives $\Delta P_o = V\,\Delta D(1/\eta - 1)$

DoQ_ref corresponds to baseline profile and induced power models
intended for use with table model for power at baseline propulsive force

---

		+	Performance, Table Method	
MODEL_indTab	int	+	induced power model (0 standard, 1 table, 2 table with equations)	1
nvar_ind	int	+	number independent variables (1 to 3)	0
var_ind(3)	c*12	+	variables	' '
nv_ind(3)	int	+	number of variable values (maximum ntablemax)	0
v_ind(ntablemax,3)	real	+	independent variable	
MODEL_proTab	int	+	profile power model (0 standard, 1 table, 2 table with equations)	1
KIND_proTab	int	+	profile power model (0 standard, 1 table $c_{d\text{mean}}$, 2 table $c_{d\text{mean}}F = 8C_{Po}/\sigma$)	1
nvar_pro	int	+	number independent variables (1 to 3)	0
var_pro(3)	c*12	+	variables	' '
nv_pro(3)	int	+	number of variable values (maximum ntablemax)	0
v_pro(ntablemax,3)	real	+	independent variable	
MODEL_geTab	int	+	ground effect model (0 inflow, 1 table thrust, 2 table power)	0
		+	table	
Ki(ntablemax,ntablemax,ntablemax)	real	+	induced power factor κ	
cdo(ntablemax,ntablemax,ntablemax)	real	+	profile power mean c_d	

---

independent variables: var_ind and var_pro

- 'V': flight speed V/V_{tip}
- 'Vh': horizontal speed V_h/V_{tip}
- 'mu', 'muHP': edgewise advance ratio μ (hub plane)
- 'muz', 'muzHP': axial velocity ratio μ_z (hub plane)
- 'alpha', 'alphaHP': shaft angle-of-attack $\alpha = \tan^{-1}(\mu_z/\mu)$ (hub plane)
- 'muTPP': edgewise advance ratio μ (tip-path plane)
- 'muzTPP': axial velocity ratio μ_z (tip-path plane)
- 'alphaTPP': shaft angle-of-attack $\alpha = \tan^{-1}(\mu_z/\mu)$ (tip-path plane)
- 'CTs', 'CT/s': blade loading C_T/σ
- 'Mx', 'offset': lift offset M_x/TR
- 'Mtip': tip Mach number M_{tip}
- 'Mat': advancing tip Mach number M_{at}

MODEL_geTab: ground effect included in inflow, or table power evaluated at thrust decreased by κ_g, or table power decreased by f_g

MODEL_download≥ 2: table induced and profile power evaluated at thrust increased by $f_{DL} = 1/(1 - \Delta T/T)$

nonzero values of Ki and/or cdo in FltState supersede table (or table with equations) values

		+	Rotor Drag, Standard Model	
		+	forward flight drag	
SET_Dhub	int	+	hub drag specification (1 fixed, D/q; 2 scaled, C_D; 3 scaled, squared-cubed; 4 scaled, square-root)	2
DoQ_hub	real	+	area $(D/q)_{\rm hub}$	
CD_hub	real	+	coefficient $C_{D\rm hub}$ (based on rotor area, $D/q = SC_D$)	0.0024
kDrag_hub	real	+	$k = (D/q)/(W/1000)^{2/3}$ or $(D/q)/W^{1/2}$ (Units_Dscale)	0.8
SET_Dpylon	int	+	pylon drag specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ_pylon	real	+	area $(D/q)_{\rm pylon}$	
CD_pylon	real	+	coefficient $C_{D\rm pylon}$ (based on pylon wetted area, $D/q = SC_D$)	0.
SET_Dduct	int	+	duct drag specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ_duct	real	+	area $(D/q)_{\rm duct}$	
CD_duct	real	+	coefficient $C_{D\rm duct}$ (based on duct wetted area, $D/q = SC_D$)	0.
SET_Dspin	int	+	spinner drag specification (1 fixed, D/q; 2 scaled, C_D)	1
DoQ_spin	real	+	area $(D/q)_{\rm spin}$	0.
CD_spin	real	+	coefficient $C_{D\rm spin}$ (based on spinner wetted area, $D/q = SC_D$)	0.
		+	vertical drag	
SET_Vhub	int	+	hub drag specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV_hub	real	+	area $(D/q)_{V\rm hub}$	
CDV_hub	real	+	coefficient $C_{DV\rm hub}$ (based on rotor area, $D/q = SC_D$)	0.
SET_Vpylon	int	+	pylon drag specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV_pylon	real	+	area $(D/q)_{V\rm pylon}$	
CDV_pylon	real	+	coefficient $C_{DV\rm pylon}$ (based on pylon wetted area, $D/q = SC_D$)	0.
SET_Vduct	int	+	duct drag specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV_duct	real	+	area $(D/q)_{V\rm duct}$	
CDV_duct	real	+	coefficient $C_{DV\rm duct}$ (based on duct wetted area, $D/q = SC_D$)	0.

		+	stopped/stowed rotor	
		+	forward flight hub drag	
DoQ_hubstop	real	+	area $(D/q)_{\rm hub-stop}$	0.
CD_hubstop	real	+	coefficient $C_{D\rm hub-stop}$ (based on rotor area, $D/q = SC_D$)	0.
DoQ_hubstow	real	+	area $(D/q)_{\rm hub-stow}$	0.
CD_hubstow	real	+	coefficient $C_{D\rm hub-stow}$ (based on rotor area, $D/q = SC_D$)	0.
		+	vertical hub drag	
DoQV_hubstop	real	+	area $(D/q)_{V\rm hub-stop}$	0.
CDV_hubstop	real	+	coefficient $C_{DV\rm hub-stop}$ (based on rotor area, $D/q = SC_D$)	0.
DoQV_hubstow	real	+	area $(D/q)_{V\rm hub-stow}$	0.
CDV_hubstow	real	+	coefficient $C_{DV\rm hub-stow}$ (based on rotor area, $D/q = SC_D$)	0.
		+	stopped blade drag	
CD_bladestop	real	+	coefficient $C_{D\rm blade}$ (based on blade area, $D/q = SC_D$)	0.
		+	transition from forward flight drag to vertical drag	
MODEL_Dhub	int	+	hub drag model (0 none, 1 general, 2 quadratic)	2
MODEL_Dpylon	int	+	pylon drag model (0 none, 1 general, 2 quadratic)	2
MODEL_Dduct	int	+	duct drag model (0 none, 1 general, 2 quadratic)	2
X_hub	real	+	hub drag, transition exponent X_d	2.
X_pylon	real	+	pylon drag, transition exponent X_d	2.
X_duct	real	+	duct drag, transition exponent X_d	2.

SET_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated
component drag contributions must be consistent; pylon is rotor support, and nacelle is engine support
tiltrotor with tilting engines use pylon drag (and no nacelle drag), since pylon connected to rotor shaft axes
tiltrotor with nontilting engines: use nacelle drag as well
rotor with a spinner (such as on a tiltrotor aircraft) likely not have hub drag
SET_Dhub, hub drag: use one of DoQ_hub, CD_hub, kDrag_hub
units of kDrag are ft^2/klb$^{2/3}$ or m^2/Mg$^{2/3}$; ft^2/lb$^{1/2}$ or m^2/kg$^{1/2}$
CD = 0.0040 for typical hubs, 0.0024 for current low drag hubs, 0.0015 for faired hubs
kDrag (2/3 power) = 1.4 for typical hubs, 0.8 for current low drag hubs, 0.5 for faired hubs (English units)
kDrag (1/2 power) = 0.074 for single rotor helicopters, 0.049 for tandem helicopters,
0.038 for hingeless rotors, 0.027 for faired hubs (English units)
$W = f_W W_{MTO}$ (main rotor) or fThrust*Tdesign (antitorque or aux thrust rotor)
stopped/stowed rotor: areas or coefficients (based on SET_Dhub and SET_Vhub) replace hub drag

		+	Rotor Interference, Standard Model	
		+	model	
MODEL_develop	int	+	development along wake axis (1 step function, 2 nominal, 3 input Xdevelop)	3
Xdevelop	real	+	rate parameter t	0.2
MODEL_boundary	int	+	immersion in wake (1 step function, 2 always immersed, 3 input Xboundary)	3
MODEL_contract	int	+	far wake contraction (0 no, 1 yes)	1
Xboundary	real	+	boundary transition s (fraction contracted radius)	0.2
MODEL_int_twin	int	+	twin rotor interference (1 no correction, 2 nominal, 3 input Ktwin)	1
Ktwin	real	+	velocity factor in overlap region K_T	1.4142
Nint_wing(nwingmax)	int	+	number wing span stations	6
Nint_tail(ntailmax)	int	+	number tail span stations	2
		+	interference factors $K_{\rm int}$ (0. for no interference)	
Kint_fus	real	+	at fuselage	1.0
Kint_wing(nwingmax)	real	+	at wing	1.0
Kint_tail(ntailmax)	real	+	at tail	1.0

Kint=0 to suppress interference at component; MODEL_int=0 for no interference at all
interference factor linearly transition from Kint at $V \leq$ Vint_low to 0 at $V \geq$ Vint_high

to account for wing or tail area in wake, interference averaged at Nint points along span

MODEL_develop: step function same as Xdevelop=0; nominal same as Xdevelop=1.
MODEL_boundary: step function same as Xboundary=0; always immersed same as Xboundary=∞
MODEL_twin: only for coaxial or tandem or side-by-side; nominal same as Ktwin=$\sqrt{2}$

		+	Induced power interference at wing	
KIND_int_wing	int	+	kind (1 wing-like, 2 propeller-like)	1
Cint_wing(nwingmax)	real	+	factor $C_{\rm int}$ (0. for no interference)	0.

For tiltrotors, typically the interference is wing-like, with $C_{\rm int} \cong -0.06$

		+	Rotor Group, NDARC Weight Model	
MODEL_config	int	+	model (1 rotor, 2 tail rotor, 3 auxiliary thrust)	1
MODEL_Wblade	int	+	blade weight model (1 AFDD82, 2 AFDD00, 3 lift offset, 4 Boeing, 5 GARTEUR, 6 Tishchenko, 7 generic)	1
MODEL_Whub	int	+	hub and hinge weight model (1 AFDD82, 2 AFDD00, 3 lift offset, 4 Boeing, 5 GARTEUR, 6 Tishchenko, 7 generic)	1
MODEL_Wshaft	int	+	inter-rotor shaft weight (0 none, 1 from lift offset, 2 from shaft length)	0
		+	AFDD00 weight models	
MODEL_type	int	+	hub weight equation depend on blade weight (for hub weight; 0 no, 1 yes)	1
KIND_rotor	int	+	rotor kind (for blade weight; 1 tilting, 2 not)	2
		+	AFDD00 and AFDD82: first flapwise natural frequency ν (per-rev at hover tip speed)	
flapfreq_blade	real	+	blade (0. to use flapfreq)	0.
flapfreq_hub	real	+	hub (0. to use flapfreq_blade)	0.
		+	lift offset rotor	
MODEL_offset	int	+	rotor tip clearance (for blade weight; 1 scaled, 2 fixed)	1
offset	real	+	design lift offset L (roll moment$/TR$)	0.3
thick20	real	+	blade airfoil thickness-to-chord ratio $\tau_{.2R}$ (at 20%R)	0.21
clearance_tip	real	+	tip clearance, scaled s/R or fixed s (ft or m)	0.05
thick25	real	+	Boeing: blade airfoil thickness-to-chord ratio $\tau_{.25R}$ (at 25%R)	0.15
rattach	real	+	Boeing (blade, hub, tail rotor, aux thrust): blade attachment (fraction rotor radius)	0.09
		+	generic blade	
Kblade	real	+	factor $K_{\rm blade}$	0.
XbldN	real	+	exponent $X_{{\rm bld}N}$	0.
XbldR	real	+	exponent $X_{{\rm bld}R}$	0.
Xbldc	real	+	exponent $X_{{\rm bld}c}$	0.
XbldV	real	+	exponent $X_{{\rm bld}V}$	0.
Xbldf	real	+	exponent $X_{{\rm bld}\nu}$	0.
XbldW	real	+	exponent $X_{{\rm bld}W}$	0.
		+	generic hub	
Khub	real	+	factor $K_{\rm hub}$	0.
XhubN	real	+	exponent $X_{{\rm hub}N}$	0.
XhubR	real	+	exponent $X_{{\rm hub}R}$	0.
Xhubc	real	+	exponent $X_{{\rm hub}c}$	0.
XhubV	real	+	exponent $X_{{\rm hub}V}$	0.
Xhubf	real	+	exponent $X_{{\rm hub}\nu}$	0.
XhubW	real	+	exponent $X_{{\rm hub}W}$	0.

MODEL_tr	int	+	tail rotor weight model (1 AFDD, 2 Boeing, 3 GARTEUR)	1
thick70	real	+	GARTEUR: blade airfoil thickness-to-chord ratio $\tau_{.7R}$ (at 70%R)	0.11
MODEL_aux	int	+	auxiliary thrust weight model (1 AFDD10, 2 AFDD82, 3 Boeing, 4 GARTEUR, 5 Torenbeek, 6 generic)	1
thrust_aux	real	+	AFDD82: design maximum thrust T_{at}	0.
power_aux	real	+	AFDD10: design maximum power P_{at}	0.
material_aux	real	+	AFDD10: material factor f_m	1.
		+	generic propeller	
Kat	real	+	factor K_{at}	0.
XatN	real	+	exponent $X_{\mathrm{at}N}$	0.
XatR	real	+	exponent $X_{\mathrm{at}R}$	0.
Xatc	real	+	exponent $X_{\mathrm{at}c}$	0.
XatV	real	+	exponent $X_{\mathrm{at}V}$	0.
XatP	real	+	exponent $X_{\mathrm{at}P}$	0.
fWfold	real	+	blade fold weight f_{fold} (fraction total blade weight)	0.
fWsupt	real	+	rotor support structure weight (fraction maximum takeoff weight)	0.
Usupt	real	+	rotor support weight per length U_{supt} (lb/ft or kg/m)	0.
fshaft	real	+	rotor shaft length (fraction rotor radius) f_{shaft}	0.
Ushaft	real	+	rotor shaft weight per length U_{shaft} (lb/ft or kg/m)	0.
Uduct	real	+	duct weight per area U_{duct} (lb/ft^2 or kg/m^2)	1.5

MODEL_config: tail rotor and auxiliary thrust models use only rotor, support, and duct weights (not shaft, fold, or separate blade and hub weights)
duct weight only used for ducted fan configuration

for teetering and gimballed rotors, the flap frequency flapfreq_blade should be the coning frequency

The AFDD00 hub weight equation using the calculated blade weight (MODEL_type = 0) results in a lower average error, and best represents legacy rotor systems.
Using the actual actual blade weight (MODEL_type = 1) is best for advanced technology rotors with blades lighter than trend.

if thrust_aux$\neq 0$, supersedes design maximum thrust of rotor from sizing task
if power_aux$\neq 0$, supersedes design maximum power of rotor from sizing task
material_aux$=1$ for composite construction, 1.20 for wood, 1.31 for aluminum spar, 1.44 for aluminum construction
default Ω_{prop} is the reference rotor speed

typically fWfold = 0.04 for manual fold, 0.28 for automatic fold

rotor support structure weight must be consistent with engine support and pylon support weights of engine section

---

		+	Custom Weight Model	
WtParam_rotor(8)	real	+	parameters	0.

Chapter 23

Structure: Wing

Variable	Type		Description	Default
		+	Wing	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Geometry	
wingload	real	+	wing loading $W/S = f_W W_D/S$	
fDGW	real	+	fraction DGW f_W (for wing loading)	1.0
area	real	+	area S	
span	real	+	span b	
chord	real	+	chord c	
AspectRatio	real	+	aspect ratio AR	

---

wing parameters: for each wing; input two quantities, other two derived (SizeParam input)
SET_wing = input two of ('area' or wing loading 'WL'), ('span' or 'ratio' or 'radius' or 'width' or 'hub' or 'panel'), 'chord', aspect ratio 'aspect'
SET_wing = 'ratio+XX' to calculate span from span of another wing
SET_wing = 'radius+XX' to calculate span from rotor radius
SET_wing = 'width+XX' to calculate span from rotor radius, fuselage width, and clearance (tiltrotor)
SET_wing = 'hub+XX' to calculate span from rotor hub position (tiltrotor)
SET_wing = 'panel+XX' to calculate span from wing panel widths
if wing sized from wing loading (SET_wing='WL+xx'), area = fDGW*DGW/wingload

rotor stopped as wing: identified by wing number Rotor%StopAsWing for stoppable rotor
use SET_wing='area+span', area = blade geometric area, span = $2R$, nPanel=1, zero weight
wing aerodynamic loads calculated when FltAircraft%STOP_rotor = stopped as wing

---

		+	Geometry	
		+	rotors	
nRotorOnWing	int	+	number of rotors mounted on wing	0
RotorOnWing(nrotormax)	int	+	rotor numbers	
		+	span calculation	
fSpan	real	+	ratio wing span to span of other wing, or to rotor radius	1.0
otherWing	int	+	other wing number	0
RotorForSpan	int	+	rotor number for span (if nRotorOnWing=0)	0
RotorOnPanel(npanelmax)	int	+	rotor at wing panel edge	
thick	real	+	thickness ratio τ_w	.23
fWidth_box	real	+	wing torque box chord w_{tb} (fraction wing chord)	0.45
SET_ac	int	+	aerodynamic center offset from pivot, at zero incidence (0 none, 1 fixed, 2 scale with chord)	0
dSLac	real	+	stationline	0.
dBLac	real	+	buttline	0.
dWLac	real	+	waterline	0.
SET_cg	int	+	center of gravity offset from pivot, at zero incidence (0 none, 1 fixed, 2 scale with chord)	0
dSLcg	real	+	stationline	0.
dWLcg	real	+	waterline	0.

RotorOnWing required for SET_wing = 'radius' or 'width' or 'hub'; MODEL_wing = tiltrotor; SET_Vdrag = airfoil c_{d90}
RotorOnPanel required for SET_panel = 'radius' or 'width' or 'hub'
SET_wing = 'radius' gets radius from RotorOnWing or RotorForSpan

taper, sweep, thickness used by weight equations
taper and sweep calculated for entire wing from wing panel geometry
fWidth_box used by tiltrotor weight equations
thick and fWidth_box used for fuel in wing

		+	Geometry (for graphics)	
twist	real	+	twist	0.

		+	Geometry	
loc_wing	Location	+	aerodynamic center location	
nPanel	int	+	number of wing panels (maximum npanelmax)	1
KIND_ACoffset	int	+	aero center offset (1 fixed, 2 fraction root chord, 3 fraction inboard chord)	1
		+	Wing Panels	
SET_panel(npanelmax)	c*24	+	panel parameters	'span+taper'
span_panel(npanelmax)	real	+	span (one side), b_p	
area_panel(npanelmax)	real	+	area (both sides), S_p	
chord_panel(npanelmax)	real	+	mean chord, c_p	
fspan_panel(npanelmax)	real	+	ratio span to wing span (one side), $b_p/(b/2)$	1.
farea_panel(npanelmax)	real	+	ratio area to wing area (both sides), S_p/S	1.
fchord_panel(npanelmax)	real	+	ratio mean chord to wing chord, c_p/c	1.
		+	panel edges	
edge_panel(npanelmax)	real	+	outboard edge, y_E	
fedge_panel(npanelmax)	real	+	outboard edge, $\eta_E = y/(b/2)$	1.
lambdaI(npanelmax)	real	+	inboard chord ratio, c_I/c_{ref}	1.
lambdaO(npanelmax)	real	+	outboard chord ratio, c_O/c_{ref}	1.
		+	aerodynamic center locus	
sweep_panel(npanelmax)	real	+	sweep Λ_p (deg, + aft)	0.
dihedral_panel(npanelmax)	real	+	dihedral δ_p (deg, + up)	0.
dxAC_panel(npanelmax)	real	+	chordwise offset at panel inboard edge x_{Ip} (+ aft)	0.
dzAC_panel(npanelmax)	real	+	vertical offset at panel inboard edge z_{Ip} (+ up)	0.
		+	control surfaces	
fchord_flap(npanelmax)	real	+	flap chord $\ell_F = c_F/c_p$ (fraction panel chord)	0.25
fchord_flaperon(npanelmax)	real	+	flaperon/aileron chord $\ell_f = c_f/c_p$ (fraction panel chord)	0.25
fspan_flap(npanelmax)	real	+	flap span $f_b = b_F/b_p$ (fraction panel span)	0.5
fspan_flaperon(npanelmax)	real	+	flaperon/aileron span $f_b = b_f/b_p$ (fraction panel span)	0.5
fAC_aileron(npanelmax)	real	+	aileron aerodynamic center lateral position y	0.7

SET_wing, wing parameters: for each wing; input two quantities, other two derived
SET_wing = input two of ('area' or wing loading 'WL'), ('span' or 'ratio' or 'radius' or 'width' or 'hub' or 'panel')
SET_wing = 'chord', aspect ratio 'aspect'
SET_wing = 'ratio+XX' to calculate span from span of another wing
SET_wing = 'radius+XX' to calculate span from rotor radius

SET_wing = 'width+XX' to calculate span from rotor radius, fuselage width, and clearance (tiltrotor)
SET_wing = 'hub+XX' to calculate span from rotor hub position (tiltrotor)
SET_wing = 'panel+XX' to calculate span from wing panel widths

wing panels: SET_panel not required with only one panel
SET_panel: specify consistent definition of panels (span, edge, area, chord)
panel span: 'span' or 'bratio', else free
'span' = input span_panel, b_p
'bratio' = input ratio to wing span, fspan_panel, $b_p/(b/2)$
panel outboard edge: 'edge', 'station', 'width', 'hub', or 'adjust' (not used for tip panel)
'edge' = input edge_panel, y_E
'station' = input fraction wing semispan fedge_panel, $\eta_E = y/(b/2)$
'radius' = from rotor radius
'width' = from rotor radius, fuselage width, and clearance (tiltrotor)
'hub' = from rotor hub position (tiltrotor)
'adjust' = from adjacent input panel span or span ratio
panel area or chord: 'area', 'Sratio', 'chord', 'cratio', 'taper', else free
'area' = input area_panel, S_p
'Sratio' = input ratio to wing area, farea_panel, S_p/S
'chord' = input chord_panel, c_p
'cratio' = input ratio to wing chord, fchord_panel, c_p/c
'taper' = from chord ratios lambdaI and lambdaO

require consistent definition of panel spans and outboard edges, and consistent with SET_wing
all edges known (from input edge or station, or from adjacent panel span or span ratio)
resulting edges unique and sequential
if wing span calculated from panel widths:
one and only one input panel span or span ratio that not used to define edge
if known span: no input panel span or span ratio that not used to define edge
usually best that any free span defined for inboard panel, not outboard panel
panel area or chord:
if one or more taper (and no free), calculate c_{ref} from wing area
if one (and only one) free, calculate S_p from wing area

fAC_aileron: from panel inboard edge, fraction panel span
for nPanel=1, from centerline and fraction wing semispan

Example input for typical wing geometry

Tiltrotor, one panel:

```
Size: SET_wing='WL+width', ! span from radius, fuselage width, and clearance; and wing loading
Rotor: SET_geom='tiltrotor',KIND_TRgeom=1, ! rotor lateral position (BL) from clearance
    WingForRotor=1,otherRotor=1/2,
    clearance_fus=x.,
    fclearance_fus=1.,
Fuselage: Width_fus=x.,
Wing: wingload=x.,
    nRotorOnWing=2,RotorOnWing=1,2,
    nPanel=1,
    SET_panel='span+taper',lambdaI=1.,lambdaO=1., ! not required with only one panel
```

Tiltrotor with wing extension, two panels

```
Size: SET_wing='WL+panel', ! span from wing panel widths; and wing loading
Rotor: SET_geom='tiltrotor',KIND_TRgeom=1, ! rotor lateral position (BL) from clearance
    WingForRotor=1,otherRotor=1/2,PanelForRotor=1,
    clearance_fus=x.,
    fclearance_fus=1.,
Fuselage: Width_fus=x.,
Wing: wingload=x.,
    nRotorOnWing=2,RotorOnWing=1,2,
    nPanel=2,
    SET_panel='width+taper','span+taper', ! outboard edge from R, Width_fus, and clearance; from span_panel
    RotorOnPanel=1, 0,
    span_panel=0., x.,
    lambdaI=1., 1.,
    lambdaO=1., x.,
    sweep_panel=x., x.,
    dihedral_panel=x., x.,
    SET_ext=1,kPanel_ext=2,KIT_ext=0, ! wing extension
```

General wing, two panels, define chord and span of both
Size: SET_wing='panel+area', ! span from wing panel widths; and wing area
Rotor: SET_geom='standard',
Wing: area=x.,
nPanel=2,
SET_panel='span+chord','span+free', ! span from span_panel; chord from inboard chord_panel and area
span_panel=x., x.,
chord_panel=x., x.,

Tiltwing, three panels, four rotors
inboard hub at 1.75R (R + .25R clearance + .50R fuselage)
outboard hub at 3.6R (1.85R between hubs, overlap = .075)
wing tip at 4.2R (0.6R from outboard hub)
Size: SET_wing='WL+radius', ! calculate span from rotor radius; and wing loading
Rotor: right/right-inboard/left-inboard/left
SET_geom='tiltrotor',KIND_TRgeom=3, ! rotor lateral position (BL) from wing panel edge
WingForRotor=1,
positionOfRotor=1/1/-1/-1, ! right/left
PanelForRotor=2/1/1/2,
Wing: wingload=x.,
nRotorOnWing=4,RotorOnWing=1,2,3,4,
fSpan=4.2, ! fSpan = b/D
nPanel=3,
SET_panel='station+cratio','station+cratio','station+free',
fedge_panel=0.4167, 0.8571, 1., ! inboard-rotor/semispan, outboard-rotor/semispan, 1
fchord_panel=1., 1., 1.,

		+	Wing Extensions	
SET_ext	int	+	extension (0 for none)	0
kPanel_ext	int	+	wing panel number	2
KIT_ext	int	+	wing extension as kit (0 not kit)	0
		+	Wing Kit	
KIT_wing	int	+	wing as kit (0 not, 1 kit, 2 kit as fixed useful load)	0
fWkit	real	+	kit weight (fraction total wing weight)	0.

		+	Controls (each panel)	
		+	kind deflection	
KIND_flap(npanelmax)	int	+	flap (1 fraction root flap; 2 increment relative root flap; 3 independent)	3
KIND_aileron(npanelmax)	int	+	aileron (1 fraction root aileron; 2 increment relative root aileron; 3 independent)	3
KIND_incid(npanelmax)	int	+	incidence (1 fraction root incidence; 2 increment relative root incidence; 3 independent)	3
KIND_flaperon(npanelmax)	int	+	kind flaperon deflection (1 fraction flap; 2 increment relative flap; 3 independent)	1
		+	flap δ_{Fp}	
INPUT_flap(npanelmax)	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_flap(ncontmax,nstatemax,npanelmax)	real	+	control matrix	
nVflap(npanelmax)	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
flap(nvelmax,npanelmax)	real	+	values	
Vflap(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)	
		+	flaperon δ_{fp}	
INPUT_flaperon(npanelmax)	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_flaperon(ncontmax,nstatemax,npanelmax)	real	+	control matrix	
nVflaperon(npanelmax)	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
flaperon(nvelmax,npanelmax)	real	+	values	
Vflaperon(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)	
		+	aileron δ_{ap}	
INPUT_aileron(npanelmax)	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_aileron(ncontmax,nstatemax,npanelmax)	real	+	control matrix	
nVaileron(npanelmax)	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
aileron(nvelmax,npanelmax)	real	+	values	
Vaileron(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)	

		+	incidence i_p	
INPUT_incid(npanelmax)	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_incid(ncontmax,nstatemax,npanelmax)	real	+	control matrix	
nVincid(npanelmax)	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
incid(nvelmax,npanelmax)	real	+	values	
Vincid(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)	
		+	flow control momentum coefficient C_μ	
INPUT_flow(npanelmax)	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_flow(ncontmax,nstatemax,npanelmax)	real	+	control matrix	
nVflow(npanelmax)	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
flow(nvelmax,npanelmax)	real	+	values	
Vflow(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)	

aircraft controls connected to individual controls of component, $c = Tc_{AC} + c_0$
for each component control, define matrix T (for each control state) and value c_0
flight state specifies control state, or that control state obtained from conversion schedule
c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
by connecting aircraft control to comp control, flight state can specify comp control value
initial values if control is connected to trim variable; otherwise fixed for flight state

		+	Trim Target	
		+	wing lift	
nVlift	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	
Klift(nvelmax)	real	+	target	
Vlift(nvelmax)	real	+	speeds (CAS or TAS)	

target definition determined by Aircraft%trim_quant
Klift can be fraction total aircraft lift, lift, or C_L

		+	Aerodynamics	
MODEL_aero	int	+	model (0 none, 1 standard)	1
Idrag	real	+	incidence angle i for helicopter nominal drag (deg; 0 for not tilt)	0.

		+	Weight	
		+	wing group	
MODEL_weight	int	+	model (0 input, 1 NDARC, 2 custom)	1
		+	weight increment	
dWprim	real	+	wing primary structure	0.
dWext	real	+	wing extension	0.
dWfair	real	+	fairing	0.
dWfit	real	+	fittings	0.
dWflap	real	+	flaps and control surfaces	0.
dWwfold	real	+	wing fold	0.
dWefold	real	+	wing extension fold	0.
		+	tiltrotor model	
fWtip	real	+	factor for weight on wing tips	1.
xWtip	real	+	increment for weight on wing tips	0.
		+	Technology Factors	
TECH_prim	real	+	wing primary structure (torque box) weight $\chi_{\rm prim}$	1.0
TECH_ext	real	+	wing extension weight $\chi_{\rm ext}$	1.0
TECH_fair	real	+	fairing weight $\chi_{\rm fair}$	1.0
TECH_fit	real	+	fittings weight $\chi_{\rm fit}$	1.0
TECH_flap	real	+	flaps and control surfaces weight $\chi_{\rm flap}$	1.0
TECH_wfold	real	+	wing fold weight $\chi_{\rm fold}$	1.0
TECH_efold	real	+	wing extension fold weight $\chi_{\rm efold}$	1.0

---

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

tiltrotor model requires weight on wing tips: both sides; calculated as sum of
rotor group, engine section or nacelle group, air induction group,
engine system, drive system (less drive shaft), rotary wing and conversion flight controls,
hydraulic group, trapped fluids, wing tip extensions
fWtip and xWtip adjust Wtip_total, without changing weight statements
negative increment required when engine and transmission not at tip location with rotor

---

		+	Wing Aerodynamics, Standard Model	
AoA_zl	real	+	zero lift angle of attack α_{zl} (deg)	0.
CLmax	real	+	maximum lift coefficient $C_{L\max}$	1.5
SET_compress	int	+	compressibility correction (0 none, 1 lift, 2 drag, 3 both)	0
		+	lift	
SET_lift	int	+	specification (2 2D $dC_L/d\alpha$; 3 3D $dC_L/d\alpha$)	2
dCLda	real	+	lift curve slope $C_{L\alpha} = dC_L/d\alpha$ (per rad)	5.73
Tind	real	+	lift curve slope non-elliptical loading correction τ	0.25
Eind	real	+	Oswald or span efficiency e ($C_{Di} = (C_L - C_{L0})^2/(\pi e AR)$)	0.8
CL_Dmin	real	+	lift coefficient for minimum induced drag C_{L0}	0.
Mdiv	real	+	lift-divergence Mach number M_{div}	0.75
		+	control (each wing panel)	
eta0(npanelmax)	real	+	lift effectiveness factor η_0, $\eta_0 - \eta_1\|\delta\|$	0.85
eta1(npanelmax)	real	+	lift effectiveness factor η_1, $\eta_0 - \eta_1\|\delta\|$	0.43
Kconl(npanelmax)	real	+	calibration or correction factor for lift K_ℓ	1.
Kconm(npanelmax)	real	+	calibration or correction factor for moment K_m	1.
Kcond(npanelmax)	real	+	calibration or correction factor for drag K_d	1.
Kconx(npanelmax)	real	+	calibration or correction factor for maximum lift K_x	1.
		+	pitch moment	
CMac	real	+	pitch moment coefficient about aerodynamic center C_{Mac}	0.
		+	Wing Drag, Standard Model	
		+	forward flight drag	
SET_drag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ	real	+	area $(D/q)_0$	
CD	real	+	coefficient C_{D0} (based on wing area, $D/q = SC_D$)	0.012
		+	vertical drag	
SET_Vdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D; 3 airfoil c_{d90})	2
DoQV	real	+	area $(D/q)_V$	
CDV	real	+	coefficient, C_{DV} (based on wing area, $D/q = SC_D$)	2.
cd90	real	+	airfoil drag coefficient c_{d90} (–90 deg)	1.4
fd90	real	+	airfoil drag coefficient flap effectiveness factor f_{d90}	2.5
CDcc	real	+	compressibility drag increment C_{Dcc} at M_{cc}	0.0011
Mcc0	real	+	critical Mach number constant M_{cc0}	0.74
Mcc1	real	+	critical Mach number constant M_{cc1}	0.31

SET_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

---

		+	drag variation with angle of attack	
MODEL_drag	int	+	model (0 none, 1 general, 2 quadratic) $\Delta C_D = C_{D0} K_d \lvert\alpha_e\rvert^{X_d}$	2
AoA_Dmin	real	+	angle of attack for wing minimum drag $\alpha_{D\min}$ (deg)	0.
Kdrag	real	+	drag increment K_d	0.
Xdrag	real	+	drag increment X_d	2.
MODEL_sep	int	+	separated flow model (0 none, 1 general, 2 quadratic, 3 cubic) $\Delta C_D = C_{D0} K_s (\lvert\alpha_e\rvert - \alpha_s)^{X_s}$	3
AoA_sep	real	+	angle of attack for separation α_s (deg)	10.
Ksep	real	+	drag increment K_s	0.
Xsep	real	+	drag increment X_s	2.
		+	transition from forward flight drag to vertical drag	
AoA_tran	real	+	angle of attack for transition α_t (deg)	25.

Conventionally the Oswald efficiency e represents the wing parasite drag variation with lift, as well as the induced drag. If C_{Dp} varies with angle-of-attack, then e is just the span efficiency factor for the induced power (and C_{L0} should be zero).

		+	wing-body interference drag	
SET_wb	int	+	specification (1 fixed, D/q 2 scaled, C_D)	1
DoQ_wb	real	+	area $(D/q)_{wb}$	0.
CD_wb	real	+	coefficient C_{Dwb} (based on wing area, $D/q = SC_D$)	0.
		+	Interference velocity	
Etail(ntailmax)	real	+	angle of attack change at tail, $E = d\epsilon/d\alpha$ (rad/rad)	0.
Kint_wing(nwingmax)	real	+	interference factor $K_{\rm int}$ at other wings (0. for no interference)	0.
		+	interference power factor $K_{\rm int}$ at rotors (0. for no interference)	
Kintn_rotor(nrotormax)	real	+	normal (helicopter)	0.
Kintp_rotor(nrotormax)	real	+	inplane (propeller)	0.

---

for tandem wings, typically
 Kint_wing(aftwing)=2. for front-on-aft interference
 Kint_wing(frontwing)=0. for aft-on-front interference
for biplane wings, typically Kint_wing(otherwing)=0.7
with mutual interference (as for biplane), require trim or other iteration for convergence

interference power: inplane (propeller) factor Kintp_rotor negative for favorable

---

		+	Flow Control; $\Delta C_L = C_{L\alpha}(L_{\mu s}\sqrt{C_\mu} + L_{\mu 1}C_\mu + L_{\mu 2}C_\mu^2)$, $\Delta C_{L\max} = X_\mu C_\mu$, $\Delta C_M = M_\mu C_\mu$, $\Delta C_D = D_\mu C_\mu$	
MODEL_flow	int	+	model (0 none)	0
Lmus(npanelmax)	real	+	lift $L_{\mu s}$	1.4
Lmu1(npanelmax)	real	+	lift $L_{\mu 1}$	0.0
Lmu2(npanelmax)	real	+	lift $L_{\mu 2}$	0.0
Xmu(npanelmax)	real	+	maximum lift X_μ	1.0
Mmu(npanelmax)	real	+	moment M_μ	0.0
Dmu(npanelmax)	real	+	drag D_μ	0.0
Cmu_limit(npanelmax)	real	+	flow limit $C_{\mu\text{limit}}$	1.0

		+	Wing Group, NDARC Weight Model	
MODEL_wing	int	+	model (1 area, 2 parametric, 3 tiltrotor, 4 other)	2
MODEL_other	int	+	model (1 Boeing, 2 GARTEUR, Torenbeek (3 light, 4 transport), Raymer (5 transport, 6 general aviation))	
fLift	real	+	lift factor	1.0
bFold	real	+	parametric method: fraction wing span that folds b_{fold} (0 to 1)	0.
wfus	real	+	Boeing: maximum fuselage width (fraction wing span)	
Vdive	real	+	Boeing or Raymer: design dive speed V_{dive} (knots)	200.
rflaplift	real	+	GARTEUR: ratio maximum lift with and without flaps	
		+	area method	
Uprim	real	+	weight per area U_{prim}, wing primary structure (lb/ft^2 or kg/m^2)	5.
Uext	real	+	weight per area U_{ext}, wing extension (lb/ft^2 or kg/m^2)	3.

		+	weight factors (fraction total wing weight)	
fWfair	real	+	fairing $f_{\rm fair}$	0.10
fWfit	real	+	fittings $f_{\rm fit}$	0.12
fWflap	real	+	flaps and control surfaces $f_{\rm flap}$	0.10
fWfold	real	+	wing fold $f_{\rm fold}$	0.
fWefold	real	+	wing extension fold $f_{\rm efold}$ (fraction wing extension weight)	0.
		+	Custom Weight Model	
WtParam_wing(8)	real	+	parameters	0.
		+	Wing Group, NDARC Tiltrotor Weight Model	
		+	jump takeoff condition	
CTs_jump	real	+	rotor maximum blade loading C_T/σ	0.20
n_jump	real	+	load factor $n_{\rm jump}$ at SDGW	2.0
Vtip_jump	real	+	rotor tip speed (0. to use hover $V_{\rm tip}$)	750.0
thickTR	real	+	wing airfoil thickness-to-chord ratio τ_w	0.23
		+	width of wing structural attachments to body	
SET_Attach	int	+	definition (0 input wAttach, 1 fraction fuselage width, 2 fraction wing span)	1
fAttach	real	+	fraction width $w_{\rm attach}/w_{\rm fus}$	1.
wAttach	real	+	width $w_{\rm attach}$ (ft or m)	0.
fRG_pylon	real	+	pylon radius of gyration $r_{\rm pylon}/R$ (fraction rotor radius)	0.30
		+	wing mode frequencies (per rev, fraction rotor speed)	
freq_beam	real	+	beam bending frequency ω_B	0.5
freq_chord	real	+	chord bending frequency ω_C	0.8
freq_tors	real	+	torsion frequency ω_T	0.9
SET_refrpm	int	+	reference rotor speed (0 from input Vtip_freq, 1 hover $V_{\rm tip}$, 2 cruise $V_{\rm tip}$)	0
Vtip_freq	real	+	rotor tip speed	600.
MODEL_form	int	+	form factors (1 calculate, 2 input)	1
form_beam	real	+	torque box beam bending F_B	0.6048
form_chord	real	+	torque box chord bending F_C	0.4874
form_tors	real	+	torque box torsion F_T	1.6384
form_spar	real	+	spar caps vertical/horizontal bending F_{VH}	0.5018
eff_spar	real	+	spar structural efficiency e_{sp}	0.8
eff_box	real	+	torque box structural efficiency e_{tb}	0.8

		+	tapered spar cap correction factors	
C_t	real	+	weight correction C_t (equivalent stiffness)	0.75
C_j	real	+	weight correction C_j (equivalent strength)	0.50
C_m	real	+	strength correction C_m (equivalent stiffness)	1.5
		+	material (lb/in^2, in/in, lb/in^3; or N/m^2, m/m, kg/m^3)	
E_spar	real	+	spar modulus E_{sp}	10.E6
E_box	real	+	torque box modulus E_{tb}	10.E6
G_box	real	+	torque box shear modulus G_{tb}	4.0E6
StrainU_spar	real	+	spar ultimate strain allowable ϵ_U	0.01
StrainU_box	real	+	torque box ultimate strain allowable ϵ_U	0.01
density_spar	real	+	density spar cap ρ_{sp}	0.06
density_box	real	+	density torque box ρ_{tb}	0.06
		+	weight per area (lb/ft^2 or kg/m^2)	
Ufair	real	+	fairing U_{fair}	2.
Uflap	real	+	flaps and control surfaces U_{flap}	3.
UextTR	real	+	wing extension U_{ext}	3.
		+	weight factor	
fWfitTR	real	+	fittings f_{fit} (fraction maximum thrust of one rotor)	0.01
fWfoldTR	real	+	wing fold f_{fold} (fraction total wing weight excluding fold)	0.
fWefoldTR	real	+	wing extension fold f_{efold} (fraction wing extension weight)	0.

---

jump takeoff: hover V_{tip} obtained from RotorOnWing(1) rotor

wing frequencies: reference rotor rotation speed from rotor V_{tip} and radius from RotorOnWing(1) rotor; hover tip speed Vtip_ref(1), cruise Vtip_cruise

thickTR only used for tiltrotor wing weight

SET_Attach: attachment width used for both torsion stiffness and fairing area

---

		+	Custom Weight Model	
WtParam_wingtr(8)	real	+	parameters	0.

Chapter 24

Structure: Tail

Variable	Type		Description	Default
		+	Empennage	
title	c*100	+	title	
notes	c*1000	+	notes	
KIND_tail	int	+	kind (1 horizontal tail, 2 vertical tail, 3 V-tail horizontal, 4 V-tail vertical)	1
		+	Geometry	
SET_tail	c*16	+	specification	'vol+aspect'
area	real	+	area S	
span	real	+	span b	
chord	real	+	chord c	
AspectRatio	real	+	aspect ratio AR	
TailVol	real	+	tail volume V	
KIND_TailVol	int	+	tail volume reference (1 wing, 2 rotor)	2
TailVolRef	int	+	wing or rotor number for tail volume	1
otherVtail	int	+	other V-tail number	

KIND_tail used for geometry, baseline orientation, tail volume, tail weight model
tail parameters: input two quantities, others calculated
  SET_tail = input two of ('area' or tail volume 'vol'), ('span' or aspect ratio 'aspect' or 'chord')
tail volume reference: tail volume $V = S\ell/RA$ (tailarea * taillength / (diskarea * radius))
  or horizontal tail volume $V = S\ell/S_w c_w$ (tailarea * taillength / (wingarea * wingchord))
  or vertical tail volume $V = S\ell/S_w b_w$ (tailarea * taillength / (wingarea * wingspan))
V-tail: modeled as pair of horizontal and vertical tails (identified by otherVtail)
  separately sized, aerodynamic loads for each; dihedral calculated, cant set to zero
  weight only for second tail, based on V-tail area and aspect ratio

		+	Geometry (for graphics and weights)	
taper	real	+	taper ratio	1.0
sweep	real	+	sweep (+ aft, deg)	0.
dihedral	real	+	dihedral (deg)	0.
thick	real	+	thickness ratio	.12
		+	Geometry	
loc_tail	Location	+	aerodynamic center location	
cant	real	+	cant angle ϕ (deg)	0.
fchord_cont	real	+	control surface chord c_f/c (fraction tail chord)	0.25
fspan_cont	real	+	control surface span b_f/b (fraction tail span)	1.0
		+	Controls	
		+	elevator δ_e or rudder δ_r	
INPUT_cont	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_cont(ncontmax,nstatemax)	real	+	control matrix	
nVcont	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
cont(nvelmax)	real	+	values	
Vcont(nvelmax)	real	+	speeds (CAS or TAS)	
		+	incidence i	
INPUT_incid	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_incid(ncontmax,nstatemax)	real	+	control matrix	
nVincid	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
incid(nvelmax)	real	+	values	
Vincid(nvelmax)	real	+	speeds (CAS or TAS)	

---

horizontal tail cant angle: + to left (vertical tail for cant = 90)
vertical tail cant angle: + to right (horizontal tail for cant = 90)

aircraft controls connected to individual controls of component, $c = Tc_{AC} + c_0$
for each component control, define matrix T (for each control state) and value c_0
flight state specifies control state, or that control state obtained from conversion schedule
c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
by connecting aircraft control to comp control, flight state can specify comp control value
initial values if control is connected to trim variable; otherwise fixed for flight state

---

		+	Aerodynamics	
MODEL_aero	int	+	model (0 none, 1 standard)	1
		+	Weight	
		+	tail (empennage group)	
MODEL_weight	int	+	model (0 input, 1 NDARC, 2 custom)	1
		+	weight increment	
dWtail	real	+	basic	0.
dWfold	real	+	fold	0.
		+	Technology Factors	
TECH_tail	real	+	tail weight χ_{ht} or χ_{vt}	1.0
TECH_tfold	real	+	fold weight χ_{fold}	1.0

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

		+	Tail Aerodynamics, Standard Model	
AoA_zl	real	+	zero lift angle of attack α_{zl} (deg)	0.
CLmax	real	+	maximum lift coefficient $C_{L\max}$	1.
SET_compress	int	+	compressibility correction (0 none, 1 lift, 2 drag, 3 both)	0
		+	lift	
SET_lift	int	+	specification (2 2D $dC_L/d\alpha$; 3 3D $dC_L/d\alpha$)	2
dCLda	real	+	lift curve slope $C_{L\alpha} = dC_L/d\alpha$ (per rad)	5.73
Tind	real	+	lift curve slope non-elliptical loading correction τ	0.25
Eind	real	+	Oswald efficiency e ($C_{Di} = (C_L - C_{L0})^2/(\pi e AR)$)	0.8
CL_Dmin	real	+	lift coefficient for minimum induced drag C_{L0}	0.
Mdiv	real	+	lift-divergence Mach number M_{div}	0.75
		+	control	
eta0	real	+	lift effectiveness factor η_0, $\eta_0 - \eta_1\|\delta\|$	0.85
eta1	real	+	lift effectiveness factor η_1, $\eta_0 - \eta_1\|\delta\|$	0.43

Kconl	real	+	calibration or correction factor for lift K_ℓ	1.
Kconm	real	+	calibration or correction factor for moment K_m	1.
Kcond	real	+	calibration or correction factor for drag K_d	1.
Kconx	real	+	calibration or correction factor for maximum lift K_x	1.
		+	Tail Drag, Standard Model	
		+	forward flight drag	
SET_drag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ	real	+	area $(D/q)_0$	
CD	real	+	coefficient C_{D0} (based on tail area, $D/q = SC_D$)	0.011
		+	vertical drag	
SET_Vdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV	real	+	area $(D/q)_V$	
CDV	real	+	coefficient C_{DV} (based on tail area, $D/q = SC_D$)	1.
CDcc	real	+	compressibility drag increment C_{Dcc} at M_{cc}	0.0011
Mcc0	real	+	critical Mach number constant M_{cc0}	0.74
Mcc1	real	+	critical Mach number constant M_{cc1}	0.31

SET_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

		+	drag variation with angle of attack	
MODEL_drag	int	+	model (0 none, 1 general, 2 quadratic) $\Delta C_D = C_{D0} K_d \lvert\alpha_e\rvert^{X_d}$	2
AoA_Dmin	real	+	angle of attack for tail minimum drag $\alpha_{D\min}$ (deg)	0.
Kdrag	real	+	drag increment K_d	0.
Xdrag	real	+	drag increment X_d	2.
		+	transition from forward flight drag to vertical drag	
AoA_tran	real	+	angle of attack for transition α_t (deg)	25.

		+	Tail, NDARC Weight Model	
MODEL_tail	int	+	model (1 horizontal tail, 2 vertical tail, 3 based on KIND_tail)	3
		+	horizontal tail	
MODEL_Htail	int	+	model (1 helicopter or compound, 2 tiltrotor or tiltwing, 3 area, 4 other)	1
MODEL_Hother	int	+	model (1 GARTEUR, Torenbeek (2 low speed, 3 transport), Raymer (4 transport, 5 general aviation))	
KIND_Htail	int	+	Torenbeek or Raymer: kind (1 fixed, 2 variable incidence)	1
wfus	real	+	Raymer: fuselage width at horizontal tail w_f/b_{ht} (fraction span)	0.2
		+	vertical tail	
MODEL_Vtail	int	+	model (1 helicopter or compound, 2 tiltrotor or tiltwing, 3 area, 4 other)	1
MODEL_Vother	int	+	model (1 GARTEUR, Torenbeek (2 low speed, 3 transport), Raymer (4 transport, 5 general aviation))	
place_AntiQ	int	+	AFDD: antitorque placement (0 none, 1 on tail boom, 2 on vertical tail)	1
KIND_Vtail	int	+	Torenbeek or Raymer: kind (1 conventional, 2 T-tail)	1
fTtail	real	+	Torenbeek: T-tail factor $(S_{ht}h_{ht})/(S_{vt}b_{vt})$	0.8
Vdive	real	+	design dive speed $V_{\rm dive}$ (knots)	200.
		+	area method	
Utail	real	+	weight per area $U_{\rm tail}$ (lb/ft^2 or kg/m^2)	3.
fTfold	real	+	fold weight factor $f_{\rm fold}$ (fraction total tail weight excluding fold)	0.

weight models can use taper ratio, sweep, and thickness ratio
dive speed: $V_{\rm max}$ = SLS max speed, Vdive $= 1.25V_{\rm max}$

		+	Custom Weight Model	
WtParam_tail(8)	real	+	parameters	0.

Chapter 25

Structure: FuelTank

Variable	Type		Description	Default
		+	Fuel Tank System	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Configuration	
SET_burn	int	+	fuel quantity stored and used (1 weight, 2 energy)	1
		+	fuel weight properties	
fuel_density	real	+	fuel weight per volume $\rho_{\rm fuel}$ (lb/gallon or kg/liter)	6.5
specific_energy	real	+	fuel energy per weight $e_{\rm fuel}$ (MJ/kg)	42.8
fFuelWing(nwingmax)	real	+	fraction wing torque box filled by fuel tanks	1.0
		+	fuel tank sizing	
Wfuel_cap	real	+	fuel capacity $W_{\rm fuel-cap}$ (weight, lb or kg)	
Efuel_cap	real	+	fuel capacity $E_{\rm fuel-cap}$ (energy, MJ)	
fFuel_cap	real	+	ratio capacity to mission fuel $f_{\rm fuel-cap}$	1.0
dFuel_cap	real	+	capacity increment $d_{\rm fuel-cap}$	0.
IDENT_battery	c*16	+	battery identification	' '

store and use weight: energy calculated from weight; capacity is usable fuel weight
use Wfuel_cap, Waux_cap, fuel_density, specific_energy, fFuelWing; fWtank, fWauxtank, other weight parameters
units of specific_energy = MJ/kg, regardless of Units_energy

store and use energy: fuel weight zero; capacity is usable fuel energy
use Efuel_cap, Eaux_cap, IDENT_battery; eWtank, eWauxtank, energy_density, other weight parameters
units of Efuel_cap, Eaux_cap = MJ, regardless of Units_energy

fuel tank sizing: usable fuel capacity Wfuel_cap (weight) or Efuel_cap (energy)
SET_tank='input': input Wfuel_cap or Efuel_cap
SET_tank='miss': calculate from mission fuel used
Wfuel_cap or Efuel_cap = max(fFuel_cap*(maximum mission fuel), (maximum mission fuel)+(reserve fuel))
SET_tank='miss+power' = calculate from mission fuel used and mission battery discharge power
SET_tank='f(miss)' = function of mission fuel used
Wfuel_cap or Efuel_cap = dFuel_cap + fFuel_cap*((maximum mission fuel)+(reserve fuel))

battery identification: energy storage only, match ident of BatteryModel

		+	Geometry	
loc_tank	Location	+	location	
place	int	+	placement (for graphics; 1 internal, 2 sponson, 3 wing, 4 combination)	1
SET_length_wire	int	+	wiring length (1 input, 2 from component positions)	1
Length_wire	real	+	length $\ell_{\rm wire}$	
fLength_wire	real	+	factor	1.0
		+	Auxiliary Fuel Tank	
Mauxtanksize	int	+	number of auxiliary tank sizes (minimum 1, maximum nauxtankmax)	1
Waux_cap(nauxtankmax)	real	+	fuel capacity $W_{\rm aux-cap}$ (weight)	1000.
Eaux_cap(nauxtankmax)	real	+	fuel capacity $E_{\rm aux-cap}$ (energy)	20000.
fWauxtank(nauxtankmax)	real	+	tank weight $f_{\rm auxtank}$ (fraction auxiliary fuel weight)	0.
eWauxtank(nauxtankmax)	real	+	tank weight $e_{\rm auxtank}$ (MJ/kg or kWh/kg, Units_energy)	0.
DoQ_auxtank(nauxtankmax)	real	+	drag $(D/q)_{\rm auxtank}$ (each tank)	
loc_auxtank(nauxtankmax)	Location	+	location	
		+	Equipment power	
MODEL_Peq	int	+	model (0 for none)	0
sfc	real	+	specific fuel consumption	0.
Peq_0	real	+	power loss $P_{\rm eq0}$, constant	0.
Peq_d	real	+	power loss $P_{{\rm eq}d}$, scale with density	0.
Peq_t	real	+	power loss $P_{{\rm eq}t}$, scale with temperature	0.
KPeq_w	real	+	power loss $P_{{\rm eq}w}$, weight factor	0.

XPeq_w	real	+	power loss $P_{\mathrm{eq}w}$, weight exponent	0.
Peq_deice	real	+	deice power loss $P_{\mathrm{eq}i}$	0.

specific fuel consumption: weight (lb/hp-hr or kg/kWh) or energy (hp/hp or kW/kW)

		+	Thermal management system	
SET_TMS	int	+	design rejected power $P_{\mathrm{rej-design}}$ (0 none, 1 input, 2 fraction P_{cap})	0
Prej_design	real	+	power (hp or kW)	0.
fPrej_design	real	+	fraction	0.02
SET_FN	int	+	net jet force (0 for no force)	1
		+	Power distribution losses	
eta_dist	real	+	efficiency at P_{cap}	1.
		+	Cooling drag	
DoQ_cool	real	+	area $(D/q)_{\mathrm{cool}}$	0.
		+	Weight	
		+	fuel system (propulsion group)	
MODEL_weight	int	+	model (0 input, 1 NDARC, 2 custom)	1
		+	weight increment	
dWtank	real	+	tanks and support; battery management system	0.
dWplumb	real	+	plumbing; power distribution (wiring)	0.
		+	Technology Factors	
TECH_tank	real	+	fuel tank weight χ_{tank}	1.0
TECH_plumb	real	+	plumbing weight χ_{plumb}	1.0

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

		+	Fuel System, NDARC Weight Model	
		+	weight storage	
		+	fuel tank	
MODEL_tank	int	+	model (1 fraction, 2 parametric, Torenbeek (3 integral, 4 generic), Raymer (5 transport, 6 general aviation))	2
ntank_int	int	+	number of internal tanks N_{int}	4
fWtank	real	+	tank weight f_{tank} (fraction fuel capacity weight)	0.09
Ktoler	real	+	parametric: ballistic tolerance factor f_{bt} (1.0 to 2.5)	2.5
KIND_crash	int	+	parametric: survivability (1 baseline, 2 UTTAS/AAH level of survivability)	2
Ktank	real	+	Torenbeek (generic): factor K_{tank}	3.2
Xtank	real	+	Torenbeek (generic): exponent X_{tank}	0.727
fint	real	+	Raymer: integral tank capacity (fraction total)	1.0
fprot	real	+	Raymer: protected tank capacity (fraction total)	1.0
		+	plumbing	
MODEL_plumb	int	+	model (1 fraction, 2 parametric)	2
nplumb	int	+	total number of fuel tanks (internal and auxiliary) for plumbing N_{plumb}	4
K0_plumb	real	+	weight increment $K_{0\text{plumb}}$ (lb)	150.
K1_plumb	real	+	weight factor $K_{1\text{plumb}}$ (lb)	2.0
fWplumb	real	+	plumbing weight f_{plumb} (fraction total fuel system weight)	0.4

MODEL_tank: fraction method uses fWtank; parametric method uses ntank_int, Ktoler, KIND_crash

K1_plumb is a crashworthiness and survivability factor; typically K1_plumb = 2.
K0_plumb is the sum of weights for auxiliary fuel, in-flight refueling, pressure refueling, inerting system, etc.; typically K0_plumb = 50 to 250 lb

---

		+	energy storage	
eWtank	real	+	tank weight e_{tank} (MJ/kg or kWh/kg, Units_energy)	
energy_density	real	+	tank volume density ρ_{tank} (MJ/liter or kWh/liter, Units_energy)	
fBMS	real	+	battery management system (fraction basic tank weight)	0.2
		+	power distribution (wiring) weight	
Uwire	real	+	weight per length	0.62
fwire	real	+	fraction basic tank weight	0.2

specific energy e_{tank} and energy density ρ_{tank} based on usable fuel capacity (consistent with $d_{\max} - d_{\min}$)

		+	Custom Weight Model	
WtParam_tank(8)	real	+	parameters	0.

Chapter 26

Structure: Propulsion

Variable	Type		Description	Default
		+	Propulsion Group	
title	c*100	+	title	
notes	c*1000	+	notes	

propulsion group is set of components and engine groups, connected by drive system
components (rotors) define power required, engine groups define power available
drive system defines ratio of rotational speeds of components (relative primary rotor speed)

Variable	Type		Description	Default
		+	Drive system	
nGear	int	+	number of states (maximum ngearmax)	1
STATE_gear_var	int	+	drive system state for variable speed transmisson (0 for none)	0

drive system branches: one primary rotor per propulsion group (specify $V_{\rm tip}$), others dependent (specify gear ratio)
specify primary engine group only if no rotors in propulsion group
drive system state: identifies gear ratio set for multiple speed transmissions
state=0 to use conversion schedule, state=n (1 to nGear) to use gear ratio #n
variable speed transmission: for drive system state STATE_gear_var, gear ratio factor $f_{\rm gear}$ (control) included
when evaluate rotational speed of dependent rotors and engines

Variable	Type		Description	Default
		+	Transmission losses	
MODEL_Xloss	int	+	model (1 fraction component power required; 2 with function drive shaft limit)	2
fPloss_xmsn	real	+	gear box loss $\ell_{\rm xmsn}$ (fraction total component power required)	0.04
Ploss_windage	real	+	power loss due to windage $P_{\rm windage}$	0.

		+	Accessory losses	
Pacc_0	real	+	power loss P_{acc0}, constant	0.
Pacc_d	real	+	power loss $P_{\text{acc}d}$, scale with density	0.
Pacc_n	real	+	power loss $P_{\text{acc}n}$, scale with density and rpm	0.
Pacc_deice	real	+	deice power loss $P_{\text{acc}i}$	0.
fPacc_ECU	real	+	ECU (etc.) power loss ℓ_{acc} (fraction component+transmission power)	0.
fPacc_IRfan	real	+	IRS fan loss ℓ_{IRfan} (fraction total engine power)	0.
		+	Geometry	
SET_length	int	+	drive shaft length (1 input, 2 from hub positions, 3 scale with radius)	2
Length_ds	real	+	length ℓ_{DS}	
fLength_ds	real	+	factor	0.9

SET_length: input (use Length_ds) or calculated (from fLength_ds)

		+	Drive system torque limit	
Plimit_ds	real	+	drive system power limit $P_{DS\text{limit}}$	
fPlimit_ds	real	+	drive system power limit factor	1.0
SET_Plimit_size	int	+	drive system limit when sizing transmission (0 not applied to power available)	0
		+	Drive system ratings	
nrate_ds	int	+	number of ratings (maximum nratemax)	1
rating_ds(nratemax)	c*12	+	drive system rating designation	' '
frating_ds(nratemax)	real	+	torque limit factor	1.0
		+	schedule	
Vdrive_hover	real	+	maximum speed for hover and helicopter mode (CAS or TAS)	
Vdrive_cruise	real	+	minimum speed for cruise (CAS or TAS)	
rating_ds_hover	c*12	+	rating for hover and helicopter mode ($V \leq V_{\text{drive-hover}}$)	' '
rating_ds_conv	c*12	+	rating for conversion mode ($V_{\text{drive-hover}} < V < V_{\text{drive-cruise}}$)	' '
rating_ds_cruise	c*12	+	rating for cruise mode ($V \geq V_{\text{drive-cruise}}$)	' '

drive system torque limits: SET_limit_ds = input (use Plimit_xx) or calculate (from fPlimit_xx)
SET_limit_ds='input': Plimit_ds input
SET_limit_ds='ratio': from takeoff power, fPlimit_ds$\sum(N_{\text{eng}}P_{\text{eng}})$

SET_limit_ds='Pav': from engine power available at transmission sizing conditions and missions (DESIGN_xmsn)
fPlimit_ds$(\Omega_{\rm ref}/\Omega_{\rm prim})\sum(N_{\rm eng}P_{av})$
SET_limit_ds='Preq': from engine power required at transmission sizing conditions and missions (DESIGN_xmsn)
fPlimit_ds$(\Omega_{\rm ref}/\Omega_{\rm prim})\sum(N_{\rm eng}P_{req})$
engine shaft: options for SET_limit_ds≠'input'
SET_limit_es=0: Plimit_es
SET_limit_es=1: fPlimit_es × (engine group $P_{\rm eng}$ or P_{av} or P_{req}, depending on SET_limit_ds)
SET_limit_es=2: fPlimit_es $\times P_{DS\rm limit}(P_{{\rm eng}EG}/P_{{\rm eng}PG})$

drive system power limit: corresponds to power of all engines of propulsion group (all engine groups)
can be used for trim (trim_quant='Q margin')
used for drive system weight, tail rotor weight, transmission losses
limits propulsion group P_{av} (if FltState%SET_Plimit=on)
engine shaft power limit: corresponds to all engines of engine group (nEngine × Peng)
limits engine group P_{av} (if FltState%SET_Plimit=on)
rotor shaft power limit: corresponds to one rotor
all limits
can be used for max effort in flight state (max_quant='Q margin')
can be used for max gross weight in flight condition or mission (SET_GW='maxQ' or 'maxPQ')
always check and print whether exceed torque limit

the engine model gives the power available, accounting for installation losses and mechanical limits
then the power available is reduced by the factor FltState%fPower
next torque limits are applied (unless FltState%SET_Plimit=off), first engine shaft limit and then drive system limit

SET_Plimit_size=0: drive system limits are not applied for transmission sizing conditions and mission segments (DESIGN_xmsn); otherwise use FltState%SET_Plimit

drive system ratings: blank to use engine ratings of first engine group
limit at flight state is $rxf_QP_{\rm limit}$, where r is the rotor speed ratio and x is the rating factor frating_ds
if nrate_ds≤ 1, drive system rating not used
schedule used if FltAircraft%rating_ds='speed'

		+	Control	
		+	rotational speed increment ΔN, primary rotor or primary engine (rpm)	
INPUT_DN	int	+	connection to aircraft controls (0 none, 1 input T matrix)	0
T_DN(ncontmax,nstatemax)	real	+	control matrix	
nVDN	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
DN(nvelmax)	real	+	values	
VDN(nvelmax)	real	+	speeds (CAS or TAS)	

---

aircraft controls connected to individual controls of component, $c = Tc_{AC} + c_0$
for each component control, define matrix T (for each control state) and value c_0
flight state specifies control state, or that control state obtained from conversion schedule
c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
by connecting aircraft control to comp control, flight state can specify comp control value
initial values if control is connected to trim variable; otherwise fixed for flight state

---

		+	Weight	
		+	drive system (propulsion group)	
MODEL_DS	int	+	model (0 input, 1 NDARC, 2 custom)	1
		+	weight increment	
dWgb	real	+	gear box	0.
dWrs	real	+	rotor shaft	0.
dWds	real	+	drive shaft	0.
dWrb	real	+	rotor brake	0.
dWcl	real	+	clutch	0.
dWgd	real	+	gas drive	0.
STATE_gear_wt	int	+	drive system state for weight	1
kEngineGroup_wt(2)	int	+	EngineGroup for weight (input, output)	1
		+	Technology Factors	
TECH_gb	real	+	gear box weight χ_{gb}	1.0
TECH_rs	real	+	rotor shaft weight χ_{rs}	1.0
TECH_ds	real	+	drive shaft weight χ_{ds}	1.0
TECH_rb	real	+	rotor brake weight χ_{rb}	1.0

TECH_cl	real	+	clutch weight χ_{cl}	1.0
TECH_gd	real	+	gas drive weight χ_{gd}	1.0

---

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

kEngineGroup_wt: always identify engine group for drive system input
if propulsion group has rotors, primary rotor speed used for drive system output
if propulsion group does not have rotors, must identify engine group for drive system output

drive system weight = gear box (including rotor shaft) + drive shaft + rotor brake + clutch + gas drive
tiltrotor wing weight model requires weight on wing tip (drive system, without rotor shaft)

		+	Drive System, NDARC Weight Model	
		+	gear box (including rotor shafts)	
MODEL_gbrs	int	+	model (1 AFDD83, 2 AFDD00, 3 other)	1
MODEL_other	int	+	model (1 Boeing, 2 Boeing (alternate), GARTEUR (3 helicopter, 4 tiltrotor), 5 Tishchenko, 6 generic)	
fShaft	real	+	rotor shaft weight f_{rs} (fraction gear box and rotor shaft weight)	0.13
ngearbox	int	+	AFDD83: number of gear boxes N_{gb}	7
fTorque	real	+	AFDD83: second (main or tail) rotor rated torque f_Q (fraction total drive system rated torque)	0.03
nstage	int	+	Boeing: number of stages in main-rotor drive	4
		+	generic gearbox	
Kgbrs	real	+	factor K_{gbrs}	0.
XgbP	real	+	exponent X_{gbP}	0.
Xgbe	real	+	exponent X_{gbe}	0.
Xgbr	real	+	exponent X_{gbr}	0.
KIND_other	int	+	other: separate tail rotor drive weight increment (0 none)	0
Ktrgb	real	+	tail rotor drive weight increment factor K_{trgb}	1.0
fPower_tr	real	+	tail rotor power (fraction total drive system rated power)	0.15
gear_tr	real	+	tail rotor gear ratio	5.0

		+	drive shaft and rotor brake	
MODEL_dsrb	int	+	model (0 none, 1 AFDD82)	1
ndriveshaft	int	+	AFDD82: number of intermediate drive shafts N_{ds} (excluding rotor shafts)	6
fPower	real	+	AFDD82: second (main or tail) rotor rated power f_P (fraction total drive system rated power)	0.15

---

fPower = fTorque*(otherrotor RPM)/(mainrotor RPM)
typically fTorque=fPower=0.6 for twin main rotors (tandem, coaxial, tiltrotor)
for single main rotor and tail rotor, fTorque = 0.03, fPower = 0.15 (0.18 for 2-bladed teeter)

typically fShaft = 0.13 (data range 0.06 to 0.20)

		+	Custom Weight Model	
WtParam_drive(8)	real	+	parameters	0.

Chapter 27

Structure: EngineGroup

Variable	Type		Description	Default
		+	Engine Group	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Description	
MODEL_engine	c*32	+	engine model	'RPTEM'
IDENT_engine	c*16	+	engine identification	'Engine'
IDENT_system2	c*16	+	second system identification	' '
nEngine	int	+	number of engines N_{eng}	1
nEngine_main	int	+	number of main engines	1
Peng	real	+	engine power P_{eng} (SLS static at takeoff rating, $0.$ for P0_ref(rating_to))	0.
rating_to	c*12	+	takeoff power rating	'MCP'
rating_idle	c*12	+	idle power rating	'MCP'
kFuelTank	int	+	fuel tank system number	1
kRotor_react	int	+	rotor number for reaction drive	
fuselage_flow	int	+	fuselage flow control (0 not)	1
wing_flow(nwingmax)	int	+	wing flow control (0 not)	1
		+	Propulsion Group	
kPropulsion	int	+	group number	1
KIND_xmsn	int	+	drive system branch (1 primary, 0 dependent)	0
INPUT_gear	int	+	gear ratio input (1 from Nspec, 2 gear)	1
gear(ngearmax)	real	+	engine gear ratio $r = \Omega_{\text{spec}}/\Omega_{\text{prim}}$ (ratio rpm to rpm of primary rotor in propulsion group)	1.0

MODEL_engine: engine model
- 'RPTEM', 'shaft' = turboshaft engine (RPTEM); IDENT_engine $\rightarrow$ EngineModel; fuel is weight
- 'table' = turboshaft engine (table); IDENT_engine $\rightarrow$ EngineTable; fuel is weight
- 'recip' = reciprocating engine; IDENT_engine $\rightarrow$ RecipModel; fuel is weight
- 'comp' = compressor; IDENT_engine $\rightarrow$ CompressorModel; not use fuel

'comp+react' = compressor for reaction drive; IDENT_engine → CompressorModel; not use fuel
'comp+flow' = compressor for flow control; IDENT_engine → CompressorModel; not use fuel
'motor' = electric motor; IDENT_engine → MotorModel; fuel is energy
'gen' = electric generator; IDENT_engine → MotorModel; fuel is energy (generated, not burned)
'motor+gen' = motor + generator (mode $B \geq 0$ for motor); IDENT_engine → MotorModel; fuel is energy
MODEL_engine: convertible engine; only with turboshaft
'+react' = reaction drive (mode $B = 1$); IDENT_system2 → EngineModel
'+jet', '+fan' = turbojet/turbofan (mode $B = 1$); IDENT_system2 → EngineModel

engine identification: match ident of EngineModel or EngineTable or RecipModel or CompressorModel or MotorModel
second system identification: match ident of EngineModel; not use weight
number of main engines: for fuel tank weight

for fixed engine: use $P_{\rm eng} = 0.$ and no size task (or engine power not sized)
takeoff power rating: for engine scaling, aircraft power loading, fuel tank weight
FltState%rating can be set to 'idle' (rating_idle) or 'takeoff' (rating_to)

fuel tank system identified for burn must store and use weight (turboshaft, reciprocating)
or energy (motor, may have BatteryModel)
fuel tank system identified for generation must store and use energy (may have BatteryModel)

drive system branch: primary engine group only designated if no rotors for propulsion group
INPUT_gear: calculate gear from Nspec and Vtip_ref of primary rotor of propulsion group, or specify gear ratio
variable speed transmission: for drive system state STATE_gear_var, gear ratio factor $f_{\rm gear}$ (control) included
when evaluate rotational speed of engine

		+	Sizing	
SET_power	int	+	specification (0 sized, 1 fixed)	0
fPsize	real	+	sized power ratio f_n	1.0
SET_Pother	int	+	sized power from other engine group (0 not)	0
fEsize(nengmax)	real	+	fraction other engine group power f_E	0.

SET_power: if SIZE_perf='engine', used to distribute propulsion group power required among engine groups
$P_{\text{eng}} = f_n P_{\text{sized}}/N_{\text{eng}}$ for n-th engine group, $P_{\text{sized}} = P_{PG} - \sum_{\text{fixed}} N_{\text{eng}} P_{\text{eng}}$
must size at least first engine group, so SET_power and fPsize values not used for first group
fPsize calculated for first engine group, must be > 0.
not used (SET_power=1) if group consumes power (compressor or generator, which sized if SIZE_engine='engine')
FltState%SET_Preq specifies distribution of power required for flight state

SET_Pother: size power from engine group of other propulsion groups, $\max(P_{\text{eng}}, f_E P_{\text{eng}-\text{other}})$

		+	Drive system torque limit	
SET_limit_es	int	+	engine shaft (0 input, 1 fraction power, 2 fraction drive system limit)	1
Plimit_es	real	+	engine shaft power limit $P_{ES\text{limit}}$	
fPlimit_es	real	+	engine shaft power limit factor	1.0

drive system torque limits: SET_limit_ds = input (use Plimit_es) or calculated (from fPlimit_es)
SET_limit_ds='input': Plimit_ds input
SET_limit_ds='ratio': from takeoff power, fPlimit_ds$\sum(N_{\text{eng}} P_{\text{eng}})$
SET_limit_ds='Pav': from engine power available at transmission sizing conditions and missions (DESIGN_xmsn)
fPlimit_ds$(\Omega_{\text{ref}}/\Omega_{\text{prim}}) \sum(N_{\text{eng}} P_{av})$
SET_limit_ds='Preq': from engine power required at transmission sizing conditions and missions (DESIGN_xmsn)
fPlimit_ds$(\Omega_{\text{ref}}/\Omega_{\text{prim}}) \sum(N_{\text{eng}} P_{req})$
engine shaft: options for SET_limit_ds≠'input'
SET_limit_es=0: Plimit_es
SET_limit_es=1: fPlimit_es × (engine group P_{eng} or P_{av} or P_{req}, depending on SET_limit_ds)
SET_limit_es=2: fPlimit_es $\times P_{DS\text{limit}}(P_{\text{eng}EG}/P_{\text{eng}PG})$

engine shaft power limit: corresponds to all engines of engine group (nEngine × Peng)
limits engine group P_{av} (if FltState%SET_Plimit=on)
can be used for max effort in flight state (max_quant='Q margin')
can be used for max gross weight in flight condition or mission (SET_GW='maxQ' or 'maxPQ')
always check and print whether exceed torque limit

		+	Installation	
Kffd	real	+	deterioration factor on engine fuel flow or performance K_{ffd}	1.05
eta_d	real	+	engine inlet efficiency η_d (fraction, for δ_M)	0.98
		+	power losses (fraction power available, $P_{\rm loss}/P_a$)	
fPloss_inlet	real	+	engine inlet loss ℓ_{in}	0.
fPloss_ps	real	+	inlet particle separator loss ℓ_{in}	0.
fPloss_exh	real	+	engine exhaust loss ℓ_{ex} (IRS off)	0.015
		+	auxiliary air momentum drag (IRS off)	
fMF_auxair	real	+	mass flow $f_{\rm aux}$ (fraction engine mass flow)	0.007
eta_auxair	real	+	ram recovery efficiency $\eta_{\rm aux}$	0.75
		+	IR suppressor	
		+	power losses (IRS on)	
fPloss_exh_IRon	real	+	engine exhaust loss ℓ_{ex}	0.030
		+	auxiliary air momentum drag (IRS on)	
fMF_auxair_IRon	real	+	mass flow $f_{\rm aux}$ (fraction engine mass flow)	0.01
eta_auxair_IRon	real	+	ram recovery efficiency $\eta_{\rm aux}$	0.75
		+	Convertible	
Kffd_conv	real	+	deterioration factor on engine fuel flow or performance K_{ffd}	1.05
		+	power losses (fraction power available, $P_{\rm loss}/P_a$)	
fPloss_exh_conv	real	+	engine exhaust loss ℓ_{ex}	0.015
		+	Thermal management system	
SET_TMS	int	+	design rejected power $P_{\rm rej-design}$ for one engine (0 none, 1 input, 2 fraction $P_{\rm eng}$)	0
Prej_design	real	+	power (hp or kW)	0.
fPrej_design	real	+	fraction	0.02
		+	Model	
SET_FN	int	+	net jet force (0 for no force)	1
SET_Daux	int	+	auxiliary air momentum drag (0 for no drag)	1

installation power losses = inlet + particle separator + exhaust (including IRS)
IR suppressor state specified by STATE_IRS in operating condition
motor or generator: only use Kffd, thermal management system

		+	Geometry	
loc_engine	Location	+	location	
direction	c*16	+	nominal orientation ('+x', '–x', '+y', '–y', '+z', '–z')	'x'
SET_geom	int	+	position (0 standard, 1 tiltrotor, 2 rotor)	0
RotorForEngine	int	+	rotor number	1
SET_Swet	int	+	nacelle/cowling wetted area (1 fixed, input Swet; 2 scaled, W_{ES}; 3 scaled, W_{ES} and W_{gbrs})	2
Swet	real	+	area S_{wet} (per engine)	0.
kSwet	real	+	factor, $k = S_{\text{wet}}/(w/N_{\text{eng}})^{2/3}$ (Units_Dscale)	0.8

SET_geom: calculation override part of location input
SET_geom=tiltrotor: calculate lateral position (BL) from RotorForEngine
SET_geom=rotor: (SL,BL,WL or XoL,YoL,ZoL) is relative loc_rotor(RotorForEngine)
SET_Swet, wetted area: input (use Swet) or calculated (from kSwet)
units of kSwet are ft^2/lb$^{2/3}$ or m^2/kg$^{2/3}$
$w = W_{ES}$ (engine system) or $W_{ES} + W_{gbrs}/N_{EG}$ (engine system and drive system)
nacelle wetted area used for nacelle drag, and for cowling weight
engine group nacelle must be consistent with rotor pylon

		+	Controls	
		+	amplitude A (fixed engine group power)	
INPUT_amp	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_amp(ncontmax,nstatemax)	real	+	control matrix	
nVamp	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
amp(nvelmax)	real	+	values	
Vamp(nvelmax)	real	+	speeds (CAS or TAS)	

		+	mode B	
INPUT_mode	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_mode(ncontmax,nstatemax)	real	+	control matrix	
nVmode	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
mode(nvelmax)	real	+	values	
Vmode(nvelmax)	real	+	speeds (CAS or TAS)	
		+	incidence i (tilt)	
INPUT_incid	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_incid(ncontmax,nstatemax)	real	+	control matrix	
nVincid	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
incid(nvelmax)	real	+	values	
Vincid(nvelmax)	real	+	speeds (CAS or TAS)	
		+	yaw ψ	
INPUT_yaw	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_yaw(ncontmax,nstatemax)	real	+	control matrix	
nVyaw	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
yaw(nvelmax)	real	+	values	
Vyaw(nvelmax)	real	+	speeds (CAS or TAS)	
		+	gear ratio factor f_{gear} (variable speed transmission only)	
INPUT_fgear	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_fgear(ncontmax,nstatemax)	real	+	control matrix	
nVfgear	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
fgear(nvelmax)	real	+	values	
Vfgear(nvelmax)	real	+	speeds (CAS or TAS)	

---

aircraft controls connected to individual controls of component, $c = Tc_{AC} + c_0$
- for each component control, define matrix T (for each control state) and value c_0
  - flight state specifies control state, or that control state obtained from conversion schedule
- c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
- by connecting aircraft control to comp control, flight state can specify comp control value
- initial values if control is connected to trim variable; otherwise fixed for flight state

---

		+	Nacelle Drag	
MODEL_drag	int	+	model (0 none, 1 standard)	1
Idrag	real	+	incidence angle i for helicopter nominal drag (deg; 0 for not tilt)	0.

---

component drag contributions must be consistent
pylon is rotor support, and nacelle is engine support
tiltrotor with tilting engines use pylon drag (and no nacelle drag),
since pylon connected to rotor shaft axes
tiltrotor with nontilting engines, use nacelle drag as well

		+	Weight	
		+	engine weight	
MODEL_weight	int	+	model (0 input, 1 RPTEM or NASA, 2 custom)	1
dWEng	real	+	weight increment (all engines)	0.
		+	engine system (except engine), engine section or nacelle group, air induction group	
		+	model (0 input, 1 NDARC, 2 custom)	
MODEL_sys	int	+	engine system	1
MODEL_nac	int	+	engine section or nacelle	1
MODEL_air	int	+	air induction	1
		+	weight increment	
dWexh	real	+	exhaust	0.
dWacc	real	+	accessories	0.
dWsupt	real	+	engine support	0.
dWcowl	real	+	engine cowling	0.
dWpylon	real	+	pylon support	0.
dWair	real	+	air induction	0.
		+	Technology Factors	
TECH_eng	real	+	engine weight $\chi_{\rm eng}$	1.0
TECH_cowl	real	+	engine cowling weight $\chi_{\rm cowl}$	1.0
TECH_pylon	real	+	pylon structure weight $\chi_{\rm pylon}$	1.0
TECH_supt	real	+	engine support structure weight $\chi_{\rm supt}$	1.0
TECH_air	real	+	air induction system weight $\chi_{\rm airind}$	1.0

TECH_exh	real	+	exhaust system weight χ_{exh}	1.0
TECH_acc	real	+	engine accessories weight χ_{acc}	1.0

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

engine system weight = engine + exhaust + accessory (WES used for rotor pylon wetted area, engine nacelle wetted area, rotor moving weight)
nacelle weight = support + cowl + pylon
engine weight parameters in EngineModel

tiltrotor wing weight model requires weight on wing tip:
engine section or nacelle group, air induction group, engine system

		+	Nacelle Drag, Standard Model	
		+	forward flight drag	
SET_drag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ	real	+	area $(D/q)_0$	
CD	real	+	coefficient C_{D0} (based on wetted area, $D/q = SC_D$)	
		+	vertical drag	
SET_Vdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV	real	+	area $(D/q)_V$	
CDV	real	+	coefficient C_{DV} (based on wetted area, $D/q = SC_D$)	
		+	transition from forward flight drag to vertical drag	
MODEL_Deng	int	+	model (0 none)	1
Xdrag	real	+	exponent X_d	2.0

SET_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

		+	Cooling Drag	
DoQ_cool	real	+	area $(D/q)_{\rm cool}$	0.
		+	Engine Section or Nacelle Group, NDARC Weight Model	
MODEL_nacelle	int	+	model (1 parametric, 2 scale with power, 3 Boeing, 4 Raymer (transport))	1
fWpylon	real	+	pylon support structure weight $f_{\rm pylon}$ (fraction maximum takeoff weight)	0.
		+	nacelle group weight, W vs P_{0C}	
Knac	real	+	factor $K_{\rm nac}$	
Xnac	real	+	exponent $X_{\rm nac}$	
n_clf	real	+	Boeing: crash load factor	20.
fWidth_nac	real	+	Raymer: nacelle width (fraction nacelle length)	0.2
		+	Air Induction Group, NDARC Weight Model	
MODEL_airind	int	+	model (1 parametric, 2 area)	1
fWair	real	+	air induction weight $f_{\rm airind}$ (fraction engine support plus air induction weight)	0.3
Uair	real	+	weight per nacelle area $U_{\rm airind}$ (lb/ft^2 or kg/m^2)	
		+	Engine System, NDARC Model	
		+	exhaust system weight, per engine, including IR suppressor; $W_{\rm exh}$ vs P_{0C}	
Kwt0_exh	real	+	$K_{0\rm exh}$	0.
Kwt1_exh	real	+	$K_{1\rm exh}$	0.002
		+	engine accessories	
MODEL_lub	int	+	lubrication system weight (1 in engine weight, 2 in accessory weight)	1

---

typically fWair = 0.3 (data range 0.1 to 0.6)

engine support and pylon support weights must be consistent with rotor support structure weight

---

		+	Custom Weight Model	
WtParam_engsys(8)	real	+	parameters	0.

Structure: JetGroup

Variable	Type		Description	Default
		+	Jet Group	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Description	
MODEL_jet	c*32	+	jet model	'RPJEM'
IDENT_jet	c*16	+	jet identification	'Jet'
IDENT_system2	c*16	+	second system identification	' '
nJet	int	+	number of jets N_{jet}	1
Tjet	real	+	jet thrust T_{jet} (SLS static at takeoff rating, $0.$ for T0_ref(rating_to))	0.
rating_to	c*12	+	takeoff thrust rating	'MCT'
rating_idle	c*12	+	idle thrust rating	'MCT'
kFuelTank	int	+	fuel tank system number	1
kRotor_react	int	+	rotor number for reaction drive	
fuselage_flow	int	+	fuselage flow control (0 not)	1
wing_flow(nwingmax)	int	+	wing flow control (0 not)	1

MODEL_jet: jet model
- 'RPJEM', 'jet', 'fan' = turbojet/turbofan engine (RPJEM); IDENT_jet → JetModel; fuel is weight
- 'react' = reaction drive (RPJEM)); IDENT_jet → JetModel; fuel is weight
- 'flow' = flow control (RPJEM)); IDENT_jet → JetModel; fuel is weight
- 'simple' = simple force generator; no model identified; fuel is weight or energy

MODEL_jet: convertible engine; only with turbojet/turbofan
- '+react' = reaction drive (mode $B = 1$); IDENT_system2 → JetModel

jet identification: match ident of JetModel
second system identification: match ident of JetModel; not use weight

for fixed jet: use $T_{\text{jet}} = 0.$ and no size task (or jet thrust not sized)

		+	Installation	
Kffd	real	+	deterioration factor on jet fuel flow K_{ffd}	1.05
eta_d	real	+	jet inlet efficiency η_d (fraction, for δ_M)	0.98
		+	power losses (fraction thrust available, T_{loss}/T_a)	
fTloss_inlet	real	+	engine inlet loss ℓ_{in}	0.
fTloss_exh	real	+	engine exhaust loss ℓ_{ex} (IRS off)	0.01
		+	auxiliary air momentum drag (IRS off)	
fMF_auxair	real	+	mass flow f_{aux} (fraction engine mass flow)	0.007
eta_auxair	real	+	ram recovery efficiency η_{aux}	0.75
		+	IR suppressor	
		+	power losses (IRS on)	
fTloss_exh_IRon	real	+	engine exhaust loss ℓ_{ex}	0.03
		+	auxiliary air momentum drag (IRS on)	
fMF_auxair_IRon	real	+	mass flow f_{aux} (fraction engine mass flow)	0.01
eta_auxair_IRon	real	+	ram recovery efficiency η_{aux}	0.75
		+	Convertible	
Kffd_conv	real	+	deterioration factor on jet fuel flow K_{ffd}	1.05
		+	power losses (fraction thrust available, T_{loss}/T_a)	
fTloss_exh_conv	real	+	engine exhaust loss ℓ_{ex}	0.01

installation power losses = inlet + exhaust (including IRS)
IR suppressor state specified by STATE_IRS_jet in operating condition

		+	Simple force generator	
Tmax	real	+	design maximum thrust $T_{\max}$	0.
SET_burn	int	+	fuel quantity used (1 weight, 2 energy)	1
sfc	real	+	thrust specific fuel consumption (weight or energy)	1.0
SW	real	+	specific weight S (per jet)	
KIND_simple	int	+	weight group (1 engine system, 2 propeller/fan installation, 3 tail rotor)	1

fuel tank system identified must be consistent with SET_burn

		+	Geometry	
loc_jet	Location	+	location	
direction	c*16	+	nominal orientation ('+x', '−x', '+y', '−y', '+z', '−z')	'x'
SET_Swet	int	+	nacelle/cowling wetted area (1 fixed, input Swet; 2 scaled)	2
Swet	real	+	area S_{wet} (per jet)	0.
kSwet	real	+	factor, $k = S_{\text{wet}}/(W_{ES}/N_{\text{jet}})^{2/3}$ (Units_Dscale)	0.8

SET_Swet, wetted area: input (use Swet) or calculated (from kSwet)
units of kSwet are $\text{ft}^2/\text{lb}^{2/3}$ or $\text{m}^2/\text{kg}^{2/3}$
nacelle wetted area used for nacelle drag, and for cowling weight

		+	Controls	
		+	amplitude A	
INPUT_amp	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_amp(ncontmax,nstatemax)	real	+	control matrix	
nVamp	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
amp(nvelmax)	real	+	values	
Vamp(nvelmax)	real	+	speeds (CAS or TAS)	
		+	mode B	
INPUT_mode	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_mode(ncontmax,nstatemax)	real	+	control matrix	
nVmode	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
mode(nvelmax)	real	+	values	
Vmode(nvelmax)	real	+	speeds (CAS or TAS)	

		+	incidence i (tilt)	
INPUT_incid	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_incid(ncontmax,nstatemax)	real	+	control matrix	
nVincid	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
incid(nvelmax)	real	+	values	
Vincid(nvelmax)	real	+	speeds (CAS or TAS)	
		+	yaw ψ	
INPUT_yaw	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_yaw(ncontmax,nstatemax)	real	+	control matrix	
nVyaw	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
yaw(nvelmax)	real	+	values	
Vyaw(nvelmax)	real	+	speeds (CAS or TAS)	

---

aircraft controls connected to individual controls of component, $c = Tc_{AC} + c_0$
for each component control, define matrix T (for each control state) and value c_0
flight state specifies control state, or that control state obtained from conversion schedule
c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
by connecting aircraft control to comp control, flight state can specify comp control value
initial values if control is connected to trim variable; otherwise fixed for flight state

---

		+	Nacelle Drag	
MODEL_drag	int	+	model (0 none, 1 standard)	1
Idrag	real	+	incidence angle i for helicopter nominal drag (deg; 0 for not tilt)	0.
		+	Weight	
		+	jet weight	
MODEL_weight	int	+	model (0 input, 1 RPJEM, 2 custom)	1
dWJet	real	+	weight increment (all jets)	0.
		+	engine system (except jet), engine section or nacelle group, air induction group	
		+	model (0 input, 1 NDARC, 2 custom)	
MODEL_sys	int	+	engine system	1

MODEL_nac	int	+	engine section or nacelle	1
MODEL_air	int	+	air induction	1
		+	weight increment	
dWexh	real	+	exhaust	0.
dWacc	real	+	accessories	0.
dWsupt	real	+	engine support	0.
dWcowl	real	+	engine cowling	0.
dWpylon	real	+	pylon support	0.
dWair	real	+	air induction	0.
		+	Technology Factors	
TECH_jet	real	+	jet weight $\chi_{\rm jet}$	1.0
TECH_jetcowl	real	+	engine cowling weight $\chi_{\rm cowl}$	1.0
TECH_jetpylon	real	+	pylon structure weight $\chi_{\rm pylon}$	1.0
TECH_jetsupt	real	+	engine support structure weight $\chi_{\rm supt}$	1.0
TECH_jetair	real	+	air induction system weight $\chi_{\rm airind}$	1.0
TECH_jetexh	real	+	exhaust system weight $\chi_{\rm exh}$	1.0
TECH_jetacc	real	+	engine accessories weight $\chi_{\rm acc}$	1.0

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

engine system weight = engine + exhaust + accessory (WES used for nacelle wetted area)
nacelle weight = support + cowl + pylon
jet weight parameters in JetModel

		+	Nacelle Drag, Standard Model	
		+	forward flight drag	
SET_drag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ	real	+	area $(D/q)_0$	
CD	real	+	coefficient C_{D0} (based on wetted area, $D/q = SC_D$)	
		+	vertical drag	
SET_Vdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV	real	+	area $(D/q)_V$	
CDV	real	+	coefficient C_{DV} (based on wetted area, $D/q = SC_D$)	

		+	transition from forward flight drag to vertical drag	
MODEL_Djet	int	+	model (0 none)	1
Xdrag	real	+	exponent X_d	2.0

SET_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

		+	Cooling Drag	
DoQ_cool	real	+	area $(D/q)_{\mathrm{cool}}$	0.
		+	Engine Section or Nacelle Group, NDARC Weight Model	
MODEL_nacelle	int	+	model (1 parametric, 2 scale with thrust, 3 Boeing, 4 Raymer (transport))	1
fWpylon	real	+	pylon support structure weight f_{pylon} (fraction maximum takeoff weight)	0.
		+	nacelle group weight, W vs T_{0C}	
Knac	real	+	factor K_{nac}	
Xnac	real	+	exponent X_{nac}	
n_clf	real	+	Boeing: crash load factor	20.
fWidth_nac	real	+	Raymer: nacelle width (fraction nacelle length)	0.2
		+	Air Induction Group, NDARC Weight Model	
MODEL_airind	int	+	model (1 parametric, 2 area)	1
fWair	real	+	air induction weight f_{airind} (fraction engine support plus air induction weight)	0.3
Uair	real	+	weight per nacelle area U_{airind} (lb/ft^2 or kg/m^2)	
		+	Engine System, NDARC Model	
		+	exhaust system weight, per jet; W_{exh} vs T_{0C}	
Kwt0_exh	real	+	$K_{0\mathrm{exh}}$	0.
Kwt1_exh	real	+	$K_{1\mathrm{exh}}$	0.002
		+	engine accessories	
MODEL_lub	int	+	lubrication system weight (1 in jet weight, 2 in accessory weight)	1
		+	Custom Weight Model	
WtParam_jetsys(8)	real	+	parameters	0.

Chapter 29

Structure: ChargeGroup

Variable	Type		Description	Default
		+	Charge Group	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Description	
MODEL_charge	c*32	+	charger model	' '
IDENT_charge	c*16	+	charger identification	'Charge'
nCharge	int	+	number of chargers $N_{\rm chrg}$	1
Pchrg	real	+	charger power $P_{\rm chrg}$ (SLS static at takeoff rating, $0.$ for P0_ref(rating_to))	0.
rating_to	c*12	+	takeoff power rating	'MCP'
rating_idle	c*12	+	idle power rating	'MCP'
kFuelTank	int	+	fuel tank system number (generated)	1
kFuelTank_burn	int	+	fuel tank system number (burned)	

MODEL_charge: charger model
'fuel' = fuel cell; IDENT_charge → FuelCellModel; fuel generated is energy; fuel burned is weight (kFuelTank_burn)
'solar' = solar cell; IDENT_charge → SolarCellModel; fuel generated is energy
'simple' = simple charger; no model identified; fuel generated is energy

charger identification: match ident of FuelCellModel or SolarCellModel

for fixed charger: use $P_{\rm chrg} = 0.$ and no size task (or charger power not sized)

fuel tank system identified for generation must store and use energy (may have BatteryModel)
fuel tank system identified for burn must store and use weight

		+	Installation	
Kffd	real	+	deterioration factor on charger fuel flow or performance K_{ffd}	1.05
eta_d	real	+	charger inlet efficiency η_d (fraction, for δ_M)	0.98
		+	auxiliary air momentum drag	
fMF_auxair	real	+	mass flow $f_{\rm aux}$ (fraction charger mass flow)	0.007
eta_auxair	real	+	ram recovery efficiency $\eta_{\rm aux}$	0.75
		+	Simple charger	
Pmax	real	+	design maximum power $P_{\rm max}$	0.
eta_chrg	real	+	efficiency $\eta_{\rm chrg}$	1.0
SW	real	+	specific weight S (per charger)	
		+	Geometry	
loc_charger	Location	+	location	
direction	c*16	+	nominal orientation ('+x', '–x', '+y', '–y', '+z', '–z')	'x'
SET_Swet	int	+	nacelle/cowling wetted area (1 fixed, input Swet; 2 scaled)	2
Swet	real	+	area $S_{\rm wet}$ (per charger)	0.
kSwet	real	+	factor, $k = S_{\rm wet}/(W/N_{\rm chrg})^{2/3}$ (Units_Dscale)	0.8

SET_Swet, wetted area: input (use Swet) or calculated (from kSwet)
units of kSwet are ft^2/lb$^{2/3}$ or m^2/kg$^{2/3}$
nacelle wetted area used for nacelle drag

		+	Controls	
		+	amplitude A	
INPUT_amp	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_amp(ncontmax,nstatemax)	real	+	control matrix	
nVamp	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
amp(nvelmax)	real	+	values	
Vamp(nvelmax)	real	+	speeds (CAS or TAS)	

		+	mode B	
INPUT_mode	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_mode(ncontmax,nstatemax)	real	+	control matrix	
nVmode	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
mode(nvelmax)	real	+	values	
Vmode(nvelmax)	real	+	speeds (CAS or TAS)	
		+	incidence i (tilt)	
INPUT_incid	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_incid(ncontmax,nstatemax)	real	+	control matrix	
nVincid	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
incid(nvelmax)	real	+	values	
Vincid(nvelmax)	real	+	speeds (CAS or TAS)	
		+	yaw ψ	
INPUT_yaw	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_yaw(ncontmax,nstatemax)	real	+	control matrix	
nVyaw	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
yaw(nvelmax)	real	+	values	
Vyaw(nvelmax)	real	+	speeds (CAS or TAS)	

aircraft controls connected to individual controls of component, $c = Tc_{AC} + c_0$
for each component control, define matrix T (for each control state) and value c_0
flight state specifies control state, or that control state obtained from conversion schedule
c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
by connecting aircraft control to comp control, flight state can specify comp control value
initial values if control is connected to trim variable; otherwise fixed for flight state

		+	Nacelle Drag	
MODEL_drag	int	+	model (0 none, 1 standard)	1
Idrag	real	+	incidence angle i for helicopter nominal drag (deg; 0 for not tilt)	0.

		+	Weight	
		+	charger weight	
MODEL_weight	int	+	model (0 input, 1 NDARC, 2 custom)	1
dWChrg	real	+	weight increment (all chargers)	0.
		+	Technology Factors	
TECH_chrg	real	+	charger weight $\chi_{\rm chrg}$	1.0

weight model result multiplied by technology factor and increment added:
Wxx = TECH_xx*Wxx_model + dWxx; for fixed (input) weight use MODEL_xx=0 or TECH_xx=0.

engine system weight = engine + exhaust + accessory = charge group weight (WES used for nacelle wetted area)
charger weight parameters in FuelCellModel or SolarCellModel

		+	Nacelle Drag, Standard Model	
		+	forward flight drag	
SET_drag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ	real	+	area $(D/q)_0$	
CD	real	+	coefficient C_{D0} (based on wetted area, $D/q = SC_D$)	
		+	vertical drag	
SET_Vdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV	real	+	area $(D/q)_V$	
CDV	real	+	coefficient C_{DV} (based on wetted area, $D/q = SC_D$)	
		+	transition from forward flight drag to vertical drag	
MODEL_Dchrg	int	+	model (0 none)	1
Xdrag	real	+	exponent X_d	2.0

SET_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

		+	Cooling Drag	
DoQ_cool	real	+	area $(D/q)_{\text{cool}}$	0.
		+	Custom Weight Model	
WtParam_chrgsys(8)	real	+	parameters	0.

Chapter 30

Structure: EngineModel

Variable	Type		Description	Default
		+	Engine Model	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Engine'

---

engine identification: used by IDENT_engine of EngineGroup input (eg 'T800')

installed: power available P_{av}, power required P_{req}, gross jet thrust F_G, net jet thrust F_N
uninstalled: power available P_a, power required P_q, gross jet thrust F_g, net jet thrust F_n
"0" = SLS static; "C" = MCP
mass flow = power / specific power ($SP = P/\dot{m}$); fuel flow = specific fuel consumption * power ($\text{sfc} = \dot{w}/P$)

engine model can be used by more than one engine group, so all parameters fixed

as model for turbojet or reaction drive of convertible engine:
only use sfc0C_ref, sfc0C_ref, and parameters for optimum speed, thrust available, and performance
P0_ref and SP0_ref required, but not used; weight, ratings, technology, and scaling variables not used

---

Variable	Type		Description	Default
		+	Weight	
MODEL_weight	int	+	RPTEM model (0 fixed, 1 $W(P)$, 2 $SW(\dot{m})$)	1
Weng	real	+	engine weight (fixed)	0.
		+	engine weight, W_{eng} vs P_{eng} model ($W = K_{0\text{eng}} + K_{1\text{eng}}P + K_{2\text{eng}}P^{X_{\text{eng}}}$)	
Kwt0_eng	real	+	constant $K_{0\text{eng}}$	0.
Kwt1_eng	real	+	constant $K_{1\text{eng}}$	0.25
Kwt2_eng	real	+	constant $K_{2\text{eng}}$	0.
Xwt_eng	real	+	exponent X_{eng}	0.
		+	engine weight, $SW = P_{\text{eng}}/W_{\text{eng}}$ vs $\dot{m}_{0C}$ model	
SW_ref	real	+	specific weight reference SW_{ref} ($\dot{m} = \dot{m}_{\text{tech}}$)	4.
SW_limit	real	+	specific weight limit SW_{lim} ($\dot{m} = \dot{m}_{\text{lim}}$)	5.

		+	Custom Weight Model	
WtParam_engine(8)	real	+	parameters	0.
		+	Parameters	
		+	Engine Ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCP'
		+	Reference	
P0_ref(nratemax)	real	+	power (P_0)	2000.
SP0_ref(nratemax)	real	+	specific power (SP_0)	150.
Pmech_ref(nratemax)	real	+	mechanical limit of power ($P_{\rm mech}$)	2500.
sfc0C_ref	real	+	specific fuel consumption at MCP (${\rm sfc}_{0C}$)	0.45
SF0C_ref	real	+	specific jet thrust ($F_{g0C} = SF_{0C}\dot{m}_{0C}$)	10.
Nspec_ref	real	+	specification turbine speed ($N_{\rm spec}$)	20000.
Nopt0C_ref	real	+	optimum turbine speed at MCP ($N_{{\rm opt}0C}$)	20000.

Reference Engine Rating: SLS, static
if MCP scaled, ratios to MCP values kept constant
engine rating: match rating designation in FltState; typically designated as
'ERP' = Emergency Rated Power (OEI power)
'CRP' = Contigency Rated Power (2.5 min)
'MRP' = Maximum Rated Power (5 or 10 min)
'IRP' = Intermediate Rated Power (30 min)
'MCP' = Maximum Continuous Power (normal operations)
engine model being used may not contain data for all ratings

		+	Technology	
SP0C_tech	real	+	specific power at MCP $SP_{\rm tech}$ (0. for SP0_ref(MCP))	0.
sfc0C_tech	real	+	specific fuel consumption at MCP ${\rm sfc}_{\rm tech}$ (0. for sfc0C_ref)	0.
Nspec_tech	real	+	specification turbine speed $N_{\rm tech}$ (0. for Nspec_ref)	0.

		+	Scaling	
FIX_size	int	+	engine size (0 scaled, 1 fixed)	0
MF_limit	real	+	mass flow at limit SP and sfc ($\dot{m}_{\rm lim}$)	30.
SP0C_limit	real	+	specific power limit $SP_{\rm lim}$	200.
sfc0C_limit	real	+	specific fuel consumption limit ${\rm sfc}_{\rm lim}$	0.34
KNspec	real	+	specification turbine speed variation (K_{Ns2})	0.

SP and sfc functions are defined by values SP0C_tech, sfc0C_tech, $\dot{m}_{\rm tech}$=P0C_ref/SP0C_tech
and limits SP0C_limit, sfc0C_limit, MF_limit
defaults SP0C_tech=SP0_ref(MCP), sfc0C_tech=sfc0C_ref, Nspec_tech=Nspec_ref
require $\dot{m}_{\rm tech} < \dot{m}_{\rm lim}$ (otherwise get SP_{0C} = SP0C_tech and ${\rm sfc}_{0C}$ = sfc0C_tech)

for no variation of SP, sfc, and SW with scale, use FIX_size=1 or MF_limit=0.
engine weight scaling determined by MODEL_weight

		+	Optimum Power Turbine Speed	
MODEL_OptN	int	+	model (0 none, 1 linear, 2 cubic)	1
		+	linear, $N_{\rm opt}/N_{\rm spec}$ vs P_q/P_0	
KNoptA	real	+	constant $K_{N{\rm opt}A}$	1.
KNoptB	real	+	constant $K_{N{\rm opt}B}$	0.
		+	cubic, $N_{\rm opt}/N_{{\rm opt}0C}$ vs P_q/P_{0C}	
KNopt0	real	+	constant $K_{N{\rm opt}0}$	1.
KNopt1	real	+	constant $K_{N{\rm opt}1}$	0.
KNopt2	real	+	constant $K_{N{\rm opt}2}$	0.
KNopt3	real	+	constant $K_{N{\rm opt}3}$	0.
XNopt	real	+	exponent $X_{N{\rm opt}}$	0.
		+	power turbine efficiency function, $\eta_t(N)/\eta_t(N_{\rm spec})$	
XNeta	real	+	exponent $X_{N\eta}$	2.0

engine power and performance variation with power turbine speed determined by $N_{\rm opt}$ and $X_{N\eta}$
used only for INPUT_param = single set; no variation if MODEL_OptN=0

		+	Power Available and Power Required Parameters	
MODEL_Pav	int	+	power available (0 constant, 1 referred, 2 general)	2
MODEL_perf	int	+	performance at power required (1 referred, 2 general)	2
INPUT_param	int	+	parameter input form (1 single set; 2 function of engine speed)	1
		+	function of engine speed	
nspeed	int	+	number of engine speeds (maximum nspeedmax)	1
rNeng(nspeedmax)	real	+	engine speed ratio, N/N_{spec}	1.
kEngineParamN(nspeedmax)	int	+	identification of parameter sets	1

constant or referred model does not use parameters, does not include effect of turbine speed
general model uses parameters for effects of temperature and ram, can include effect of turbine speed

function of engine speed (INPUT_param=2): parameters interpolated, rNeng unique and sequential

simple model: constant (MODEL_Pav=0) or constant referred (MODEL_Pav=1) power available
constant specific fuel consumption (MODEL_perf=1, sfc0C_tech=0., MF_limit=0.)
no jet force (EngineGroup%SET_FN=0), no auxiliary air momentum drag (EngineGroup%SET_Daux=0)

		+	Power Available	
INPUT_lin	int	+	input form (1 coefficients K_0, K_1; 2 values θ_b, K_b)	1
		+	referred specific power available, SP_a/SP_0 vs temperature	
Nspa(nratemax)	int	+	number of regions (maximum nengkmax-1)	0
Kspa0(nengkmax,nratemax)	real	+	K_{spa0} (piecewise linear $K_{spa} = K_0 + K_1\theta$)	3.5
Kspa1(nengkmax,nratemax)	real	+	K_{spa1} (piecewise linear $K_{spa} = K_0 + K_1\theta$)	-2.5
Tspak(nengkmax,nratemax)	real	+	θ_b	
Kspab(nengkmax,nratemax)	real	+	K_{spa-b}	
Xspa0(nengkmax,nratemax)	real	+	X_{spa0} (piecewise linear $X_{spa} = X_0 + X_1\theta$)	-.2
Xspa1(nengkmax,nratemax)	real	+	X_{spa1} (piecewise linear $X_{spa} = X_0 + X_1\theta$)	0.
Tspax(nengkmax,nratemax)	real	+	θ_b	
Xspab(nengkmax,nratemax)	real	+	X_{spa-b}	
		+	referred mass flow at power available, $\dot{m}_a/\dot{m}_0$ vs temperature	
Nmfa(nratemax)	int	+	number of regions (maximum nengkmax-1)	0

Kmfa0(nengkmax,nratemax)	real	+	K_{mfa0} (piecewise linear $K_{mfa} = K_0 + K_1\theta$)	.3
Kmfa1(nengkmax,nratemax)	real	+	K_{mfa1} (piecewise linear $K_{mfa} = K_0 + K_1\theta$)	-.3
Tmfak(nengkmax,nratemax)	real	+	θ_b	
Kmfab(nengkmax,nratemax)	real	+	K_{mfa-b}	
Xmfa0(nengkmax,nratemax)	real	+	X_{mfa0} (piecewise linear $X_{mfa} = X_0 + X_1\theta$)	1.
Xmfa1(nengkmax,nratemax)	real	+	X_{mfa1} (piecewise linear $X_{mfa} = X_0 + X_1\theta$)	0.
Tmfax(nengkmax,nratemax)	real	+	θ_b	
Xmfab(nengkmax,nratemax)	real	+	X_{mfa-b}	

---

piecewise linear function:
- input form = coefficients K_0, K_1 (N sets) or values θ_b, K_b (N+1 values)
- form not input is calculated (N-1 θ_b, K_b or N K_0, K_1)
- input K_0, K_1: adjacent K_1 different, resulting θ_b unique and sequential
- input θ_b, K_b: θ_b unique and sequential

N_{spec} = specification power turbine speed
$SP_a, \dot{m}_a$ = referred specific power and mass flow available, at N_{spec}
$SP_0, \dot{m}_0$ = referred specific power and mass flow available, at N_{spec}, SLS static
N = power turbine speed, N_{opt} = optimum power turbine speed
η_t = power turbine efficiency; assume gas power available $P_G = P_a/\eta_t$ insensitive to N, so $\eta_t(N)$ give $P_a(N)$

		+	Performance at Power Required	
		+	referred fuel flow at power required, $\dot{w}_{req}/\dot{w}_{0C}$ vs P_q/P_{0C}	
Kffq0	real	+	constant K_{ffq0}	.2
Kffq1	real	+	constant K_{ffq1}	.8
Kffq2	real	+	constant K_{ffq2}	0.
Kffq3	real	+	constant K_{ffq3}	0.
Xffq	real	+	exponent X_{ffq}	1.3
		+	referred mass flow at power required, $\dot{m}_{req}/\dot{m}_{0C}$ vs P_q/P_{0C}	
Kmfq0	real	+	constant K_{mfq0}	.6
Kmfq1	real	+	constant K_{mfq1}	.78
Kmfq2	real	+	constant K_{mfq2}	-.48

Kmfq3	real	+	constant K_{mfq3}	.1
Xmfq	real	+	exponent X_{mfq}	3.5
		+	gross jet thrust at power required, F_g/F_{g0C} vs P_q/P_{0C}	
Kfgq0	real	+	constant K_{fgq0}	.2
Kfgq1	real	+	constant K_{fgq1}	.8
Kfgq2	real	+	constant K_{fgq2}	0.
Kfgq3	real	+	constant K_{fgq3}	0.
Xfgq	real	+	exponent X_{fgq}	2.0
		+	installed net jet thrust at power required, F_G/F_g (installed thrust loss) vs ℓ_{ex}	
Kfgr0	real	+	constant K_{fgr0}	.8
Kfgr1	real	+	constant K_{fgr1}	.6
Kfgr2	real	+	constant K_{fgr2}	0.
Kfgr3	real	+	constant K_{fgr3}	0.

Structure: EngineParamN

Variable	Type		Description	Default
		+	Engine Model Parameters	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
		+	Power Available	
nrate	int	+	number of ratings	1
INPUT_lin	int	+	input form (1 coefficients K_0, K_1; 2 values θ_b, K_b)	1
		+	referred specific power available, SP_a/SP_0 vs temperature	
Nspa(nratemax)	int	+	number of regions (maximum nengkmax-1)	0
Kspa0(nengkmax,nratemax)	real	+	K_{spa0} (piecewise linear $K_{spa} = K_0 + K_1\theta$)	3.5
Kspa1(nengkmax,nratemax)	real	+	K_{spa1} (piecewise linear $K_{spa} = K_0 + K_1\theta$)	-2.5
Tspak(nengkmax,nratemax)	real	+	θ_b	
Kspab(nengkmax,nratemax)	real	+	K_{spa-b}	
Xspa0(nengkmax,nratemax)	real	+	X_{spa0} (piecewise linear $X_{spa} = X_0 + X_1\theta$)	-.2
Xspa1(nengkmax,nratemax)	real	+	X_{spa1} (piecewise linear $X_{spa} = X_0 + X_1\theta$)	0.
Tspax(nengkmax,nratemax)	real	+	θ_b	
Xspab(nengkmax,nratemax)	real	+	X_{spa-b}	
		+	referred mass flow at power available, $\dot{m}_a/\dot{m}_0$ vs temperature	
Nmfa(nratemax)	int	+	number of regions (maximum nengkmax-1)	0
Kmfa0(nengkmax,nratemax)	real	+	K_{mfa0} (piecewise linear $K_{mfa} = K_0 + K_1\theta$)	.3
Kmfa1(nengkmax,nratemax)	real	+	K_{mfa1} (piecewise linear $K_{mfa} = K_0 + K_1\theta$)	-.3
Tmfak(nengkmax,nratemax)	real	+	θ_b	
Kmfab(nengkmax,nratemax)	real	+	K_{mfa-b}	
Xmfa0(nengkmax,nratemax)	real	+	X_{mfa0} (piecewise linear $X_{mfa} = X_0 + X_1\theta$)	1.
Xmfa1(nengkmax,nratemax)	real	+	X_{mfa1} (piecewise linear $X_{mfa} = X_0 + X_1\theta$)	0.
Tmfax(nengkmax,nratemax)	real	+	θ_b	
Xmfab(nengkmax,nratemax)	real	+	X_{mfa-b}	

number of ratings consistent with EngineModel

		+	Performance at Power Required	
		+	referred fuel flow at power required, $\dot{w}_{req}/\dot{w}_{0C}$ vs P_q/P_{0C}	
Kffq0	real	+	constant K_{ffq0}	.2
Kffq1	real	+	constant K_{ffq1}	.8
Kffq2	real	+	constant K_{ffq2}	0.
Kffq3	real	+	constant K_{ffq3}	0.
Xffq	real	+	exponent X_{ffq}	1.3
		+	referred mass flow at power required, $\dot{m}_{req}/\dot{m}_{0C}$ vs P_q/P_{0C}	
Kmfq0	real	+	constant K_{mfq0}	.6
Kmfq1	real	+	constant K_{mfq1}	.78
Kmfq2	real	+	constant K_{mfq2}	-.48
Kmfq3	real	+	constant K_{mfq3}	.1
Xmfq	real	+	exponent X_{mfq}	3.5
		+	gross jet thrust at power required, F_g/F_{g0C} vs P_q/P_{0C}	
Kfgq0	real	+	constant K_{fgq0}	.2
Kfgq1	real	+	constant K_{fgq1}	.8
Kfgq2	real	+	constant K_{fgq2}	0.
Kfgq3	real	+	constant K_{fgq3}	0.
Xfgq	real	+	exponent X_{fgq}	2.0
		+	installed net jet thrust at power required, F_G/F_g (installed thrust loss) vs ℓ_{ex}	
Kfgr0	real	+	constant K_{fgr0}	.8
Kfgr1	real	+	constant K_{fgr1}	.6
Kfgr2	real	+	constant K_{fgr2}	0.
Kfgr3	real	+	constant K_{fgr3}	0.

Chapter 32

Structure: EngineTable

Variable	Type		Description	Default
		+	Engine Table	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Engine'

engine identification: used by IDENT_engine of EngineGroup input

engine table can be used by more than one engine group, so all parameters fixed

engine not scaled (SET_power, fPsize not used); eta_d not used
fixed engine weight dWEng (MODEL_weight=0)
no mass flow value, so no momentum drag of auxillary air flow (fMF_auxair, eta_auxair not used)
obtain Peng from table; mechanical limits included in power available data
tables intended for installed engine, including losses (fPloss_inlet, fPloss_ps, fPloss_exh not used)
fuel flow multiplied by Kffd, accounting for deterioration of engine efficiency

Variable	Type		Description	Default
		+	Engine ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCP'
Nspec	real	+	Specification turbine speed ($N_{\rm spec}$)	

		+	Table	
KIND_table	int	+	format (1 E, 2 H)	1
nalt	int	+	number of altitudes (maximum nengtmax)	
ntemp	int	+	number of temperatures (maximum nengxmax)	
nspeed	int	+	number of speeds (maximum nengtmax)	
nalt_ram	int	+	number of altitudes for $f_{\rm RAM}$ (maximum nengtmax)	
ntemp_ram	int	+	number of temperatures for $f_{\rm RAM}$ (maximum nengxmax)	
alt(nengtmax)	real	+	altitude h	
temp(nengxmax)	real	+	temperature τ	
speed(nengtmax)	real	+	speed V (TAS)	
alt_ram(nengtmax)	real	+	altitude h for $f_{\rm RAM}$	
temp_ram(nengxmax)	real	+	temperature τ for $f_{\rm RAM}$	

table format E: use alt, speed
table format H: use alt, temp; and for $f_{\rm RAM}$ use speed, alt_ram, temp_ram; no jet thrust

		+	Technology factors	
Kp	real	+	power available	1.0
Kw	real	+	fuel flow	1.0
Kf	real	+	net thrust	1.0
		+	Table format E	
Tp(nengtmax,nengtmax,nratemax)	real	+	power available $P_a(h,V,R)$	
Tw(nengtmax,nengtmax,nratemax)	real	+	fuel flow $\dot{w}(h,V,R)$	
Tf(nengtmax,nengtmax,nratemax)	real	+	net thrust $F_N(h,V,R)$	
		+	Table format H	
KIND_temp	int	+	temperature units (0 F or C based on Units; 1 F, 2 C)	0
		+	power available	

P0(nengtmax,nengxmax,nratemax)	real	+	static power $P_0(h,\tau,R)$	
KIND_ram	int	+	ram factor (1 table, 2 referred)	1
fRAM(nengtmax,nengxmax,nengtmax)	real	+	table ram factor $f_{\mathrm{RAM}}(V,\tau,h)$	
Xpa(nratemax)	real	+	referred ram factor $f_{\mathrm{RAM}} = (\delta_M\sqrt{\theta_M})^{X_{pa}}$, exponent X_{pa}	1.
		+	fuel flow	
KIND_fuelflow	int	+	kind (1 reference $\dot{w}_{\mathrm{ref}}(P_{q\mathrm{ref}})$, 2 table $\dot{w}(P_q,h,\tau)$)	1
		+	reference	
nfuelflow	int	+	number of fuel flow values (maximum nengxmax)	
Pq_ref(nengxmax)	real	+	reference power required $P_q/\delta^{X_{dp}}\theta^{X_{rp}}$	
ff_ref(nengxmax)	real	+	reference fuel flow $\dot{w}/\delta^{X_{df}}\theta^{X_{rf}}$	
Xdp	real	+	reference power, pressure exponent X_{dp}	1.0
Xrp	real	+	reference power, temperature exponent X_{rp}	0.5
Xdf	real	+	reference fuel flow, pressure exponent X_{df}	1.0
Xrf	real	+	reference fuel flow, temperature exponent X_{rf}	0.5
		+	table	
npower_ff	int	+	number of power required values (maximum nengtmax)	
nalt_ff	int	+	number of altitudes (maximum nengtmax)	
ntemp_ff	int	+	number of temperatures (maximum nengtmax)	
power_ff(nengtmax)	real	+	power required P_q	
alt_ff(nengtmax)	real	+	altitude h	
temp_ff(nengtmax)	real	+	temperature τ	
ff(nengtmax,nengtmax,nengtmax)	real	+	fuel flow $\dot{w}(P_q,h,\tau)$	

Chapter 33

Structure: RecipModel

Variable	Type		Description	Default
		+	Reciprocating Engine Model	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Engine'

engine identification: used by IDENT_engine of EngineGroup input

installed: power available P_{av}, power required P_{req}, gross jet thrust F_G, net jet thrust F_N
uninstalled: power available P_a, power required P_q, gross jet thrust F_g, net jet thrust F_n
fuel flow = specific fuel consumption * power ($\text{sfc} = \dot{w}/P$); mass flow = fuel flow / fuel-air ratio

reciprocating engine model can be used by more than one engine group, so all parameters fixed

Variable	Type		Description	Default
		+	Weight	
MODEL_weight	int	+	model (0 fixed, 1 $W(P)$)	1
Weng	real	+	engine weight (fixed)	0.
		+	engine weight, W_{eng} vs P_{eng} model ($W = K_{0\text{eng}} + K_{1\text{eng}}P + K_{2\text{eng}}P^{X_{\text{eng}}}$)	
Kwt0_eng	real	+	constant $K_{0\text{eng}}$	0.
Kwt1_eng	real	+	constant $K_{1\text{eng}}$	0.25
Kwt2_eng	real	+	constant $K_{2\text{eng}}$	0.
Xwt_eng	real	+	exponent X_{eng}	0.
		+	Custom Weight Model	
WtParam_recip(8)	real	+	parameters	0.

		+	Parameters	
		+	Engine Ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCP'
		+	Reference	
P0_ref(nratemax)	real	+	power (P_0)	1000.
sfc0_ref(nratemax)	real	+	specific fuel consumption (sfc_0)	0.60
F0_ref(nratemax)	real	+	fuel-air ratio (F_0)	0.08
SF0_ref(nratemax)	real	+	specific jet thrust ($F_g = SF_0\dot{m}$)	0.
Pmep_ref(nratemax)	real	+	mean effective pressure limit (P_{mep})	1000.
Pcrit_ref(nratemax)	real	+	critical (throttle) limit (P_{crit})	1000.
N0_ref(nratemax)	real	+	reference engine speed (N_0)	2000.
Nspec_ref	real	+	specification engine speed (N_{spec})	2000.

Reference Engine Rating: SLS, static
if MCP scaled, ratios to MCP values kept constant
engine rating: match rating designation in FltState; typically designated as
'MRP' = Maximum Rated Power (5 or 10 min)
'MCP' = Maximum Continuous Power (normal operations)
ratings encompass mixture settings and supercharger speeds

Pmep_ref: zero for no mechanical (mep) limit
Pcrit_ref: zero for no critical (throttle) limit; Xcrit = 0. for limit independent of engine speed

		+	Scaling	
FIX_size	int	+	engine size (0 scaled, 1 fixed)	0
Xo	real	+	specific output exponent X_o	0.2
Xs	real	+	mean piston speed exponent X_s	0.3
Xf	real	+	specific fuel consumption exponent X_f	0.1
Ksfc1	real	+	specific fuel consumption constant K_{sfc1}	1.
Ksfc2	real	+	specific fuel consumption constant K_{sfc2}	0.
KN1	real	+	engine speed constant $K_{N\mathrm{spec1}}$	1.
KN2	real	+	engine speed constant $K_{N\mathrm{spec2}}$	0.

		+	Power Available	
MODEL_Pav	int	+	model (0 constant P_a)	1
Kp(nratemax)	real	+	factor K_p	1.
Kram(nratemax)	real	+	constant $K_{\rm ram}$	1.
XpN(nratemax)	real	+	exponent X_{pN}	1.
Xpt(nratemax)	real	+	exponent $X_{p\theta}$	0.5
Xcrit(nratemax)	real	+	exponent $X_{\rm crit}$	3.0
		+	Performance at Power Required	
		+	fuel flow, $\dot{w}_{req}/\dot{w}_0$ vs P_q/P_0	
MODEL_Kffq	int	+	model (1 polynomial, 2 piecewise linear)	1
		+	polynomial	
Kffq0(nratemax)	real	+	constant K_{ffq0}	0.
Kffq1(nratemax)	real	+	constant K_{ffq1}	1.
Kffq2(nratemax)	real	+	constant K_{ffq2}	0.
Kffq3(nratemax)	real	+	constant K_{ffq3}	0.
		+	piecewise linear	
Nffq(nratemax)	int	+	number of values (maximum nengrmax)	0
Pffq(nengrmax,nratemax)	real	+	power ratio P_q/P_0	
Kffq(nengrmax,nratemax)	real	+	factor K_{ffq}	
Xffq(nratemax)	real	+	exponent X_{ffq}	0.
Xffs(nratemax)	real	+	exponent $X_{ff\sigma}$	0.
		+	fuel-air ratio, F_{req}/F_0 vs P_q/P_0	
MODEL_KFq	int	+	model (1 polynomial, 2 piecewise linear)	1
		+	polynomial	
KFq0(nratemax)	real	+	constant K_{Fq0}	1.
KFq1(nratemax)	real	+	constant K_{Fq1}	0.
KFq2(nratemax)	real	+	constant K_{Fq2}	0.
KFq3(nratemax)	real	+	constant K_{Fq3}	0.
		+	piecewise linear	
NFq(nratemax)	int	+	number of values (maximum nengrmax)	0
PFq(nengrmax,nratemax)	real	+	power ratio P_q/P_0	
KFq(nengrmax,nratemax)	real	+	factor K_{Fq}	
XFq(nratemax)	real	+	exponent X_{Fq}	0.

		+	installed net jet thrust, $K_{fgr} = F_G/F_g$ (installed thrust loss)	
Kfgr(nratemax)	real	+	constant K_{fgr}	1.

Simple model: constant power available (MODEL_Pav=0)
constant specific fuel consumption (defaults Kffq1=1. and Xffq=0., and Xf=0.)
constant fuel-air ratio (defaults KFq0=1. and XFq=0.)
no jet force (EngineGroup%SET_FN=0), no auxiliary air momentum drag (EngineGroup%SET_Daux=0)

Chapter 34

Structure: CompressorModel

Variable	Type		Description	Default
		+	Compressor Model	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Comp'

compressor identification: used by IDENT_engine of EngineGroup input

"0" = SLS static; "C" = MCP
mass flow = power / specific power ($SP = P/\dot{m}$); gross thrust = specific thrust * mass flow ($ST = T/\dot{m}$)

compressor model can be used by more than one engine group, so all parameters fixed

Variable	Type		Description	Default
		+	Weight	
MODEL_weight	int	+	model (0 fixed, 1 $W(P)$)	1
Wcomp	real	+	compressor weight (fixed)	0.
		+	compressor weight, W_{comp} vs P_{eng} model ($W = K_{0\text{comp}} + K_{1\text{comp}}P + K_{2\text{comp}}P^{X_{\text{comp}}}$)	
Kwt0_comp	real	+	constant $K_{0\text{comp}}$	0.
Kwt1_comp	real	+	constant $K_{1\text{comp}}$	0.2
Kwt2_comp	real	+	constant $K_{2\text{comp}}$	0.
Xwt_comp	real	+	exponent X_{comp}	0.
		+	Custom Weight Model	
WtParam_comp(8)	real	+	parameters	0.

		+	Parameters	
		+	Compressor Ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCP'
		+	Reference	
P0_ref(nratemax)	real	+	power (P_0)	
SP0_ref(nratemax)	real	+	specific power (SP_0)	
Pmech_ref(nratemax)	real	+	mechanical limit of power (P_{mech})	
SF0C_ref	real	+	specific jet thrust ($F_{g0C} = SF_{0C}\dot{m}_{0C}$)	
Nspec_ref	real	+	specification compressor speed (N_{spec})	

Reference Compressor Rating: SLS, static
if MCP scaled, ratios to MCP values kept constant
compressor rating: match rating designation in FltState

		+	Power Available	
		+	referred specific power available, SP_a/SP_0	
Xspa	real	+	exponent X_{spa}	1.
		+	referred mass flow at power available, $\dot{m}_a/\dot{m}_0$	
Xmfa	real	+	exponent X_{mfa}	1.
		+	Performance at Power Required	
		+	referred mass flow at power required, $\dot{m}_{req}/\dot{m}_{0C}$ vs P_q/P_{0C}	
Kmfq0	real	+	constant K_{mfq0}	
Kmfq1	real	+	constant K_{mfq1}	
Kmfq2	real	+	constant K_{mfq2}	
Kmfq3	real	+	constant K_{mfq3}	
Xmfq	real	+	exponent X_{mfq}	1.
		+	gross jet thrust at power required, F_g/F_{g0C} vs P_q/P_{0C}	
Kfgq0	real	+	constant K_{fgq0}	1.
Kfgq1	real	+	constant K_{fgq1}	0.
Kfgq2	real	+	constant K_{fgq2}	0.
Kfgq3	real	+	constant K_{fgq3}	0.
Xfgq	real	+	exponent X_{fgq}	2.0

Chapter 35

Structure: MotorModel

Variable	Type		Description	Default
		+	Motor Model	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Motor'

---

motor identification: used by IDENT_engine of EngineGroup input

"0" = SLS static; "C" = MCP

motor model can be used by more than one engine group, so all parameters fixed

---

Variable	Type		Description	Default
		+	Weight	
MODEL_weight	int	+	NASA model (0 fixed, 1 $W(P)$, 2 $W(Q)$)	2
Wmotor	real	+	motor weight (fixed)	0.
		+	motor weight, $W_{\rm motor}$ vs $P_{\rm eng}$ model ($W = K_{0\rm motor} + K_{1\rm motor}P + K_{2\rm motor}P^{X_{\rm motor}}Q^{X_{q\rm motor}}S^{X_{s\rm motor}}$)	
Kwt0_motor	real	+	constant $K_{0\rm motor}$	0.
Kwt1_motor	real	+	constant $K_{1\rm motor}$	0.
Kwt2_motor	real	+	constant $K_{2\rm motor}$	0.
Xwt_motor	real	+	exponent $X_{\rm motor}$	0.
Xwtq_motor	real	+	exponent $X_{q\rm motor}$	0.
Xwts_motor	real	+	exponent $X_{s\rm motor}$	0.
		+	motor weight, $W_{\rm motor}$ vs $Q_{\rm peak}$ model	
KIND_design	int	+	torque-to-weight design (0 only high Q/W; 1 high Q/W, 2 low Q/W factor)	0
		+	controller weight ($\Delta W = K_{\rm ESC}P^{X_{\rm ESC}}$)	
Kwt_ESC	real	+	constant $K_{\rm ESC}$	0.
Xwt_ESC	real	+	exponent $X_{\rm ESC}$	0.
		+	Custom Weight Model	
WtParam_motor(8)	real	+	parameters	0.

		+	Parameters	
		+	Motor Ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCP'
		+	Reference	
P0_ref(nratemax)	real	+	power (P_0)	0.
Ppeak_ref(nratemax)	real	+	mechanical limit of power ($P_{\rm peak}$)	
Nspec_ref	real	+	specification motor speed ($N_{\rm spec}$)	

Reference Motor Rating: SLS, static
if MCP scaled, ratios to MCP values kept constant
motor rating: match rating designation in FltState

		+	Performance	
		+	Motor/Generator Efficiency	
KIND_eff	int	+	kind (1 fixed, 2 function power, 3 map)	2
		+	fixed or function power	
eta_motor	real	+	reference efficiency (at $P_{\rm eng}$)	1.00
loss_motor	real	+	power loss (fraction $P_{\rm eng}$)	0.00
		+	efficiency map ($P_{\rm loss} = P_{\rm eng} f_{\rm loss} \sum_{i=0}^{3} \sum_{j=0}^{3} C_{ij} t^i n^j$)	
Closs(4,4)	real	+	loss coefficients Closs(i+1,j+1)= C_{ij}	0.00
floss	real	+	factor $f_{\rm loss}$	1.00
eta_cont	real	+	controller efficiency	1.00
		+	Scaling	
KNspec	real	+	specification motor speed variation (K_{Ns})	0.
KNbase	real	+	base motor speed variation (K_{Nb})	0.

$N_{\rm spec}$ used by efficiency map; $N_{\rm base}$ affects $P_{\rm peak}$ scaling
for no variation of motor speeds with scale, use KNspec = KNbase = 0.

		+	Thermal Management System	
		+	mass flow (lb/sec or kg/sec) from rejected heat (hp or kW)	
KTMSm0	real	+	constant $K_{\text{TMS}m0}$	0.
KTMSm1	real	+	constant $K_{\text{TMS}m1}$	0.07
XTMSm	real	+	exponent $X_{\text{TMS}m}$	1.
		+	power (hp or kW) from mass flow (lb/sec or kg/sec)	
KTMSp0	real	+	constant $K_{\text{TMS}p0}$	0.
KTMSp1	real	+	constant $K_{\text{TMS}p1}$	0.6
XTMSp	real	+	exponent $X_{\text{TMS}p}$	1.
		+	gross jet force (lb or N) from mass flow (lb/sec or kg/sec)	
KTMSf0	real	+	constant $K_{\text{TMS}f0}$	0.
KTMSf1	real	+	constant $K_{\text{TMS}f1}$	6.0
XTMSf	real	+	exponent $X_{\text{TMS}f}$	1.
		+	weight (lb or kg)	
KTMSw0	real	+	constant $K_{\text{TMS}w0}$	4.0
KTMSw1	real	+	constant $K_{\text{TMS}w1}$	0.3
XTMSwp	real	+	exponent $X_{\text{TMS}wp}$	1.
XTMSwm	real	+	exponent $X_{\text{TMS}wm}$	0.

Chapter 36

Structure: JetModel

Variable	Type		Description	Default
		+	Jet Model	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Jet'

jet identification: used by IDENT_jet of JetGroup input

installed: thrust available T_{av}, thrust required T_{req}
uninstalled: thrust available T_a, thrust required T_q
"0" = SLS static; "C" = MCT
mass flow = thrust / specific thrust ($ST = T/\dot{m}$); fuel flow = specific fuel consumption * thrust ($\mathrm{sfc} = \dot{w}/T$)

jet model can be used by more than one jet group, so all parameters fixed

as model for reaction drive of convertible engine:
only use sfc0C_ref and parameters for thrust available and performance at thrust required
T0_ref and ST0_ref required, but not used; weight, ratings, technology, and scaling variables not used

Variable	Type		Description	Default
		+	Weight	
MODEL_weight	int	+	RPJEM model (0 fixed, 1 $W(T)$)	1
Wjet	real	+	jet weight (fixed)	0.
		+	jet weight, W_{jet} vs T_{jet} model ($W = K_{0\mathrm{jet}} + K_{1\mathrm{jet}}T + K_{2\mathrm{jet}}T^{X_{\mathrm{jet}}}$)	
Kwt0_jet	real	+	constant $K_{0\mathrm{jet}}$	0.
Kwt1_jet	real	+	constant $K_{1\mathrm{jet}}$	0.2
Kwt2_jet	real	+	constant $K_{2\mathrm{jet}}$	0.
Xwt_jet	real	+	exponent X_{jet}	0.
		+	Custom Weight Model	
WtParam_jet(8)	real	+	parameters	0.

		+	Parameters	
		+	Jet Ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCT'
		+	Reference	
T0_ref(nratemax)	real	+	thrust (T_0)	0.
ST0_ref(nratemax)	real	+	specific thrust (ST_0)	
Tmech_ref(nratemax)	real	+	mechanical limit of thrust ($T_{\rm mech}$)	
sfc0C_ref	real	+	specific fuel consumption at MCT (sfc_{0C})	

Reference Jet Rating: SLS, static
if MCT scaled, ratios to MCT values kept constant
jet rating: match rating designation in FltState

		+	Technology	
ST0C_tech	real	+	specific thrust at MCT $ST_{\rm tech}$ ($0.$ for ST0_ref(MCT))	0.
sfc0C_tech	real	+	specific fuel consumption at MCT $\mathrm{sfc}_{\rm tech}$ ($0.$ for sfc0C_ref)	0.
		+	Scaling	
FIX_size	int	+	engine size (0 scaled, 1 fixed)	0
MF_limit	real	+	mass flow at limit ST and sfc ($\dot{m}_{\rm lim}$)	0.
ST0C_limit	real	+	specific thrust limit $ST_{\rm lim}$	0.
sfc0C_limit	real	+	specific fuel consumption limit $\mathrm{sfc}_{\rm lim}$	

ST and sfc functions are defined by values ST0C_tech, sfc0C_tech, $\dot{m}_{\rm tech}$=T0C_ref/ST0C_tech
and limits ST0C_limit, sfc0C_limit, MF_limit
defaults ST0C_tech=ST0_ref(MCT), sfc0C_tech=sfc0C_ref
require $\dot{m}_{\rm tech} < \dot{m}_{\rm lim}$ (otherwise get ST_{0C} = ST0C_tech and sfc_{0C} = sfc0C_tech)

for no variation of ST and sfc with scale, use FIX_size=1 or MF_limit=0.

bypass	real	+	Turbofan bypass ratio (0. for turbojet)	0.
		+	Thrust Available	
		+	referred specific thrust available, ST_a/ST_0	
Xsta	real	+	exponent X_{sta}	1.
		+	referred mass flow at thrust available, $\dot{m}_a/\dot{m}_0$	
Xmfa	real	+	exponent X_{mfa}	1.
		+	Performance at Thrust Required	
		+	referred fuel flow at thrust required, $\dot{w}_{req}/\dot{w}_{0C}$ vs T_q/T_{0C}	
Kffq0	real	+	constant K_{ffq0}	0.
Kffq1	real	+	constant K_{ffq1}	1.
Kffq2	real	+	constant K_{ffq2}	0.
Kffq3	real	+	constant K_{ffq3}	0.
Xffq	real	+	exponent X_{ffq}	1.
		+	referred mass flow at thrust required, $\dot{m}_{req}/\dot{m}_{0C}$ vs T_q/T_{0C}	
Kmfq0	real	+	constant K_{mfq0}	0.
Kmfq1	real	+	constant K_{mfq1}	1.
Kmfq2	real	+	constant K_{mfq2}	0.
Kmfq3	real	+	constant K_{mfq3}	0.
Xmfq	real	+	exponent X_{mfq}	1.

Chapter 37

Structure: FuelCellModel

Variable	Type		Description	Default
		+	Fuel Cell Model	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Cell'

fuel cell identification: used by IDENT_charge of ChargerGroup input

"0" = SLS static; "C" = MCP

fuel cell model can be used by more than one charger group, so all parameters fixed

Variable	Type		Description	Default
		+	Weight	
MODEL_weight	int	+	model (0 fixed, 1 $W(P)$)	1
Wcell	real	+	fuel cell weight (fixed)	0.
		+	fuel cell weight, W_{cell} vs P_{chrg} model ($W = K_{0\text{cell}} + K_{1\text{cell}}P + K_{2\text{cell}}P^{X_{\text{cell}}}$)	
Kwt0_cell	real	+	constant $K_{0\text{cell}}$	0.
Kwt1_cell	real	+	constant $K_{1\text{cell}}$	0.
Kwt2_cell	real	+	constant $K_{2\text{cell}}$	0.
Xwt_cell	real	+	exponent X_{cell}	0.
		+	Custom Weight Model	
WtParam_fuelcell(8)	real	+	parameters	0.

		+	Parameters	
		+	Fuel Cell Ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCP'
		+	Reference	
P0_ref(nratemax)	real	+	power (P_0)	0.
sfc0C_ref	real	+	specific fuel consumption at MCP (sfc_{0C})	0.

Reference Fuel Cell Rating: SLS, static
if MCP scaled, ratios to MCP values kept constant
fuel cell rating: match rating designation in FltState

		+	Performance	
idesign	real	+	design current density i_d	
pi_comp	real	+	compressor pressure ratio π_C	
		+	cell characteristics (at cell pressure $\delta_c = 1$)	
ncell	int	+	number of values (maximum nengcmax)	1
icell(nengcmax)	real	+	current density i_c	1.
vcell(nengcmax)	real	+	voltage v_c	1.
Xfc	real	+	pressure scaling exponent X_{fc}	0.38
Kmf	real	+	mass flow ratio ($\dot{m}/\dot{w}$)	86.

---

reference sfc corresponds to fuel specific energy and design cell current, including technology impact
units of idesign and icell must be consistent

icell values unique and sequential; icell(1)=0.
vcell monotonically decreasing (reversed vcell unique and sequential)

simple model: define power P0_ref and specific fuel consumption sfc0C_ref, mass flow from Kmf
ncell=1 for constant v_c, hence constant efficiency, constant power and sfc (idesign, pi_comp, Xfc not used)

Chapter 38

Structure: SolarCellModel

Variable	Type		Description	Default
		+	Solar Cell Model	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Cell'

solar cell identification: used by IDENT_charge of ChargerGroup input

"0" = SLS static; "C" = MCP

solar cell model can be used by more than one charge group, so all parameters fixed

Variable	Type		Description	Default
		+	Weight	
MODEL_weight	int	+	model (0 fixed, 1 $W(A)$)	1
Wsolar	real	+	solar cell weight (fixed)	0.
ssolar	real	+	weight density (kg/m^2)	
		+	Custom Weight Model	
WtParam_solarcell(8)	real	+	parameters	0.
		+	Parameters	
		+	Solar Cell Ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCP'
		+	Reference	
P0_ref(nratemax)	real	+	power (P_0)	0.

Reference Solar Cell Rating: SLS, static
if MCP scaled, ratios to MCP values kept constant
solar cell rating: match rating designation in FltState

		+	Performance	
esolar	real	+	power density (W/m^2)	
		+	Efficiency	
KIND_eff	int	+	kind (1 fixed, 2 function power)	2
eta_cell	real	+	reference efficiency (at $P_{\rm chrg}$)	1.00
loss_cell	real	+	power loss (fraction $P_{\rm chrg}$)	0.00

---

simple model: power density esolar and weight density ssolar; with efficiency in esolar (KIND_eff=1 and eta_cell=1.)

Chapter 39

Structure: BatteryModel

Variable	Type		Description	Default
		+	Battery Model	
title	c*100	+	title	'Default'
notes	c*1000	+	notes	
ident	c*16	+	identification	'Battery'

battery identification: used by IDENT_battery of FuelTank input

battery model can be used by more than one fuel tank system, so all parameters fixed

Variable	Type		Description	Default
		+	Performance	
MODEL_battery	int	+	model (1 equivalent circuit, 2 lithium-ion)	1
Vref	real	+	reference voltage V_{ref}	4.2
xmbd	real	+	maximum burst discharge current x_{mbd} (1/hr)	20.
xCCmax	real	+	maximum charge current $x_{CC\max}$ (1/hr)	4.
		+	actual cell depth-of-discharge ($d_{\text{act}} = d_{\min} + (d_{\max} - d_{\min})d_{\text{use}}$)	
DoDmin	real	+	minimum $d_{\min}$	0.0
DoDmax	real	+	maximum $d_{\max}$	0.8
		+	Thermal Management System	
		+	mass flow (lb/sec or kg/sec) from rejected heat (hp or kW)	
KTMSm0	real	+	constant $K_{\text{TMS}m0}$	0.
KTMSm1	real	+	constant $K_{\text{TMS}m1}$	0.07
XTMSm	real	+	exponent $X_{\text{TMS}m}$	1.
		+	power (hp or kW) from mass flow (lb/sec or kg/sec)	
KTMSp0	real	+	constant $K_{\text{TMS}p0}$	0.
KTMSp1	real	+	constant $K_{\text{TMS}p1}$	0.6
XTMSp	real	+	exponent $X_{\text{TMS}p}$	1.

		+	gross jet force (lb or N) from mass flow (lb/sec or kg/sec)	
KTMSf0	real	+	constant $K_{\mathrm{TMS}f0}$	0.
KTMSf1	real	+	constant $K_{\mathrm{TMS}f1}$	6.0
XTMSf	real	+	exponent $X_{\mathrm{TMS}f}$	1.
		+	weight (lb or kg)	
KTMSw0	real	+	constant $K_{\mathrm{TMS}w0}$	4.0
KTMSw1	real	+	constant $K_{\mathrm{TMS}w1}$	0.3
XTMSwp	real	+	exponent $X_{\mathrm{TMS}wp}$	1.
XTMSwm	real	+	exponent $X_{\mathrm{TMS}wm}$	0.
		+	Equivalent Circuit Model	
KIND_eff	int	+	kind (1 fixed, 2 function power)	2
		+	discharge	
eta_dischrg	real	+	reference efficiency (at P_{ref})	1.00
loss_dischrg	real	+	power loss (fraction P_{ref})	0.00
		+	charge	
eta_chrg	real	+	reference efficiency (at P_{ref})	1.00
loss_chrg	real	+	power loss (fraction P_{ref})	0.00

simple model: constant efficiencies eta_dischrg and eta_chrg (KIND_eff=1)

		+	Lithium-Ion Model	
		+	discharge	
fcrit	real	+	critical voltage factor ($F_V = f_{\mathrm{crit}}$ is capacity)	0.6
fd	real	+	nominal discharge voltage ($V_d = f_d V_{\mathrm{ref}}$)	1.0
		+	open circuit voltage ratio ($V_o = V_d F_V(d)$)	
nFV	int	+	number of points (maximum 40)	0
DoD(40)	real	+	depth-of-discharge d (fraction)	0.
FV(40)	real	+	F_V	0.
Tref	real	+	reference temperature T_{ref} (deg C)	20.
fTC	real	+	temperature control power loss f_{TC} (fraction component power)	0.01

		+	current influence on discharge voltage	
R	real	+	internal resistance $x_{mbd}CR/V_{\rm ref}$	0.1
kdI	real	+	depth-of-discharge $k_{dI}x_{mbd}C$	0.05
		+	temperature influence on discharge voltage	
kVT	real	+	voltage increment k_{VT}	0.005
kdT	real	+	depth-of-discharge k_{dT}	0.000005
		+	charge	
fc	real	+	nominal charge voltage ($V_c = f_c V_{\rm ref}$)	1.0
kcV	real	+	CC phase starting voltage decrement k_{cV}	0.1
ks	real	+	CV phase parameter k_σ	0.2

open circuit voltage ratio: monotonically decreasing; default used if nFV=0
default DoD = 0.,.1,.2,.3,.4,.5,.6,.7,.8,.9,.91,.92,.93,.94,.95,.96,.97,.98,.99,1.,1.01,1.02
default FV = 1.,.97,.95,.93,.915,.90,.89,.88,.87,.85,.847,.842,.835,.826,.815,.8,.78,.75,.7,.6,.4,0.

Structure: Location

Variable	Type		Description	Default
		+	Location	
		+	input	
		+	fixed (dimensional, arbitrary origin)	
FIX_geom	c*8	+	input	' '
SL	real	+	stationline	
BL	real	+	buttline	
WL	real	+	waterline	
		+	scaled (based on reference length, from reference point)	
XoL	real	+	x/L	
YoL	real	+	y/L	
ZoL	real	+	z/L	
		+	reference length	
KIND_scale	int	+	kind (0 global, 1 rotor radius, 2 wing span, 3 fuselage length)	0
kScale	int	+	identification (component number)	1

Fixed input: FIX_geom = 'x', 'y', 'z' (or combination) to override INPUT_geom=2
Geometry: Location for each component
fixed geometry input (INPUT_geom = 1): dimensional SL/BL/WL
stationline + aft, buttline + right, waterline + up; arbitary origin; units = ft or m
scaled geometry input (INPUT_geom = 2): divided by reference length (KIND_scale, kScale)
XoL + aft, YoL + right, ZoL + up; from reference point
option to fix some geometry (FIX_geom in Location override INPUT_geom)
option to specify reference length (KIND_scale in Location override global KIND_scale)
Reference point: KIND_Ref, kRef; input dimensional XX_Ref, or position of identified component
component reference must be fixed
Locations can be calculated from other parameters (configuration specific)