

**NASA/TP–2009-215402**



**NDARC**  
**NASA Design and Analysis of Rotorcraft**

**Input**

Appendix 8  
Release 1.9  
February 2015

*Wayne Johnson*  
*NASA Ames Research Center, Moffett Field, CA*

---

**February 2015**

## The NASA STI Program Office . . . 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 Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published 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's counterpart 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 cosponsored 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 that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at <http://www.sti.nasa.gov>
- E-mail your question via the Internet to [help@sti.nasa.gov](mailto:help@sti.nasa.gov)
- Fax your question to the NASA Access Help Desk at (301) 621-0134
- Telephone the NASA Access Help Desk at (301) 621-0390
- Write to:  
NASA Access Help Desk  
NASA Center for AeroSpace Information  
7115 Standard Drive  
Hanover, MD 21076-1320

**NASA/TP–2009-215402**



**NDARC**  
**NASA Design and Analysis of Rotorcraft**

**Input**

*Wayne Johnson*  
*NASA Ames Research Center, Moffett Field, CA*

National Aeronautics and  
Space Administration

Ames Research Center  
Moffett Field, California 94035-1000

---

**February 2015**

Available from:

NASA Center for AeroSpace Information  
7115 Standard Drive  
Hanover, MD 21076-1320  
(301) 621-0390

National Technical Information Service  
5285 Port Royal Road  
Springfield, VA 22161  
(703) 487-4650

## Contents

---

1. Data Structures and Input .....	1
2. Input Based on Configuration .....	13
3. Parameters and Constants .....	20
<hr/>	
4. <b>Job</b> .....	24
5. <b>Design</b> .....	31
6. <b>Cases</b> .....	32
<hr/>	
7. <b>Size</b> .....	37
8. SizeParam .....	38
9. <b>OffDesign</b> .....	43
10. OffParam .....	44
11. <b>Performance</b> .....	45
12. PerfParam .....	46
13. <b>MapEngine</b> .....	47
14. <b>MapAero</b> .....	50
15. <b>FltCond</b> .....	53
16. <b>Mission</b> .....	58
17. MissParam .....	59
18. MissSeg .....	65
<hr/>	
19. <b>FltState</b> .....	73
20. FltAircraft .....	74

21. FltFuse .....	91
22. FltGear .....	92
23. FltRotor .....	93
24. FltWing .....	99
25. FltTail .....	102
26. FltTank .....	103
27. FltProp .....	105
28. FltEngn .....	107
29. FltJet .....	110
30. FltChrg .....	113
<hr/>	
31. <b>Solution</b> .....	116
<hr/>	
32. <b>Cost</b> .....	120
33. CostCTM .....	122
<hr/>	
34. <b>Aircraft</b> .....	124
35. XAircraft .....	139
36. <b>Systems</b> .....	141
37. WFltCont .....	148
38. WDeIce .....	149
39. <b>Fuselage</b> .....	150
40. AFuse .....	153
41. WFuse .....	156
42. <b>LandingGear</b> .....	157
43. AGear .....	159
44. WGear .....	160

45. <b>Rotor</b> .....	161
46. PRotorInd .....	177
47. PRotorPro .....	180
48. PRotorTab .....	183
49. DRotor .....	185
50. IRotor .....	187
51. WRotor .....	189
52. <b>Wing</b> .....	191
53. AWing .....	200
54. WWing .....	203
55. WWingTR .....	204
56. <b>Tail</b> .....	206
57. ATail .....	210
58. Wtail .....	212
<hr/>	
59. <b>FuelTank</b> .....	213
60. WTank .....	216
61. <b>Propulsion</b> .....	217
62. WDrive .....	222
63. <b>EngineGroup</b> .....	223
64. DEngSys .....	232
65. WEngSys .....	233
66. <b>JetGroup</b> .....	234
67. DJetSys .....	240
68. WJetSys .....	241
69. <b>ChargeGroup</b> .....	242
70. DChrgSys .....	247

71. WChrgSys .....	248
<hr/>	
72. <b>EngineModel</b> .....	249
73. EngineParam .....	254
74. <b>EngineTable</b> .....	257
75. <b>CompressorModel</b> .....	259
76. <b>MotorModel</b> .....	261
77. <b>JetModel</b> .....	264
78. <b>FuelCellModel</b> .....	267
79. <b>SolarCellModel</b> .....	269
80. <b>BatteryModel</b> .....	271
<hr/>	
81. <b>Location</b> .....	273
82. <b>Weight</b> .....	275



## 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.

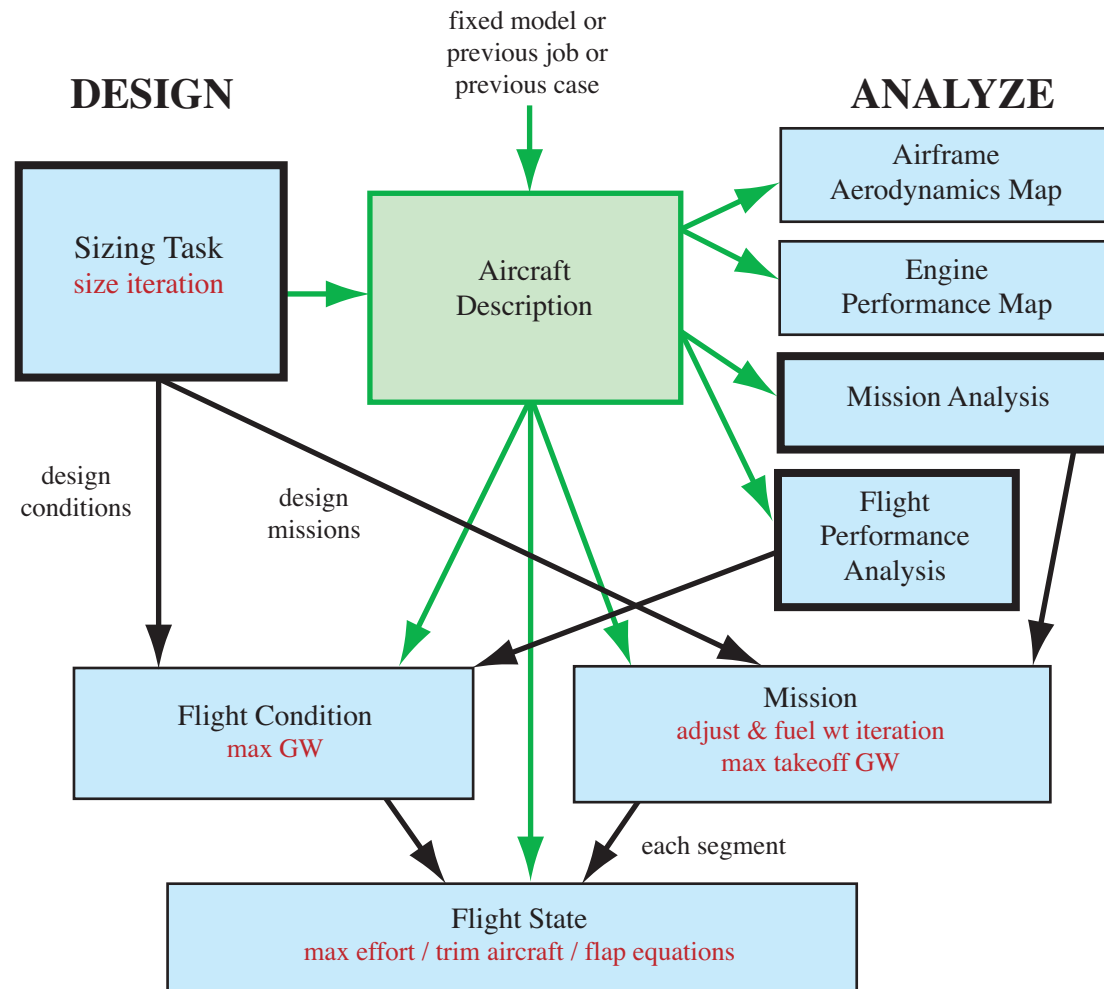


Figure 1-1 Outline of NDARC tasks.

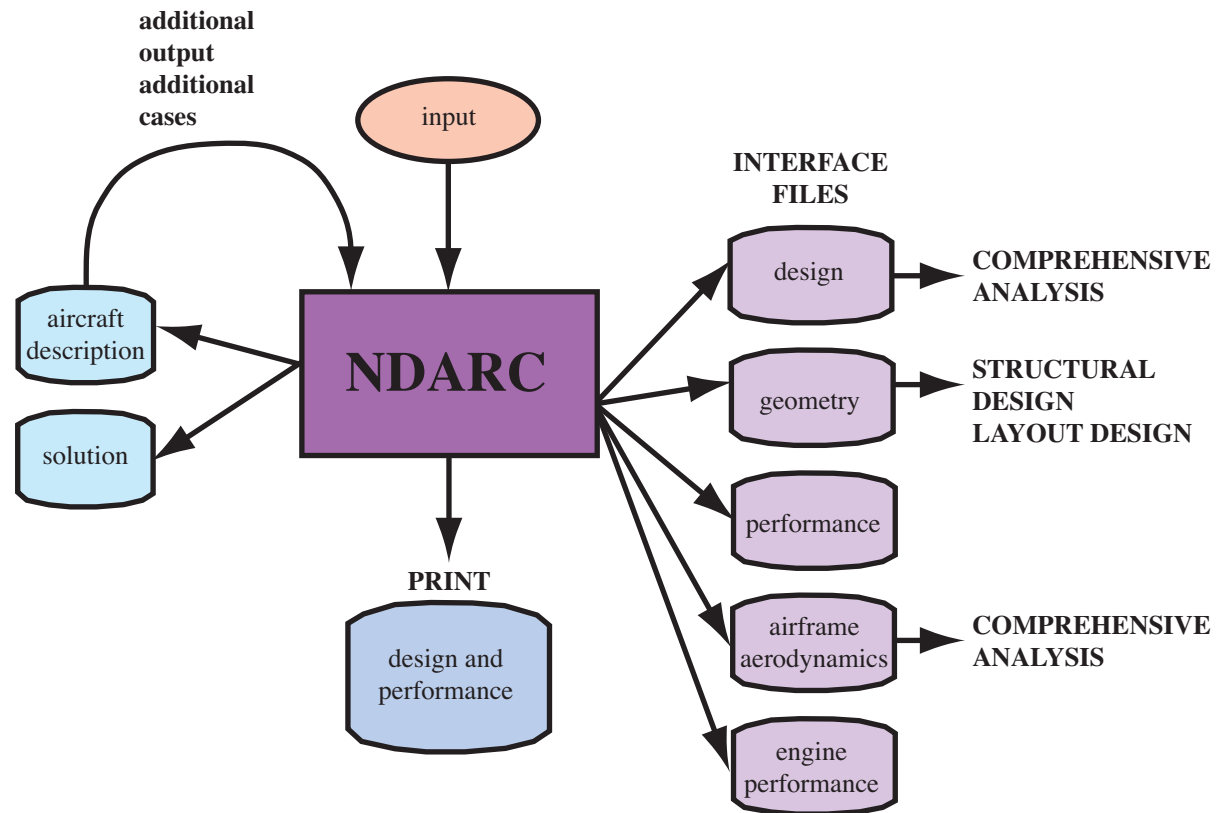


Figure 1-2 NDARC Interfaces.

```

&JOB INIT_input=0,INIT_data=0,&END
&DEFN action='ident',created='time-date',title='standard input',&END
!#####
&DEFN action='read file',file='engine.list',&END
&DEFN action='read 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 quant='Aircraft',&END
&VALUE config='helicopter',&END
&DEFN quant='Rotor 1',&END
&VALUE rotate=1,&END
&DEFN action='configuration',&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) PARENT: integer; engine model number for QUANT='EngineParam'; value is 1 if absent; input variable can be PARENT or ENGINEMODEL.
- e) FILE: file name or logical name (length = 256).
- f) CREATED: character string of creation time and date (length = 20).
- g) TITLE: character string of title identifying input file (length = 80).
- h) VERSION: code version number as character string (length = 6).
- i) 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. Parent structures may be required: engine model number for QUANT='EngineParam'; the parent number is 1 if PARENT is absent. 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\_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
<b>Primary Input Only</b>			
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
<b>Primary or Secondary Input</b>			
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	PARENT
'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, CostCTM		
'Aircraft'	Aircraft		
'Systems'	Systems, WFltCont, WDelce		
'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, EngineParam	nEngineModel	
'EngineParamN n'	EngineParam	nspeed	EngineModel number
'EngineTable n'	EngineTable	nEngineTable	
'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,nenginetable=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,nenginetable=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	[EngineParam]Param	
CostCTM	AWing	[EngineParam]ParamN(nspeedmax)	
Aircraft	Weight	EngineTable(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)	
WDelce	WTail		

## 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 minimum input required to execute `ACTION='configuration'` is:

```
&DEFN quant='Aircraft',&END
&VALUE config='helicopter',&END
&DEFN quant='Rotor 1',&END
&VALUE rotate=#,overlap_tandem=0.25,&END
&DEFN quant='Rotor 2',&END
&VALUE rotate=#,overlap_tandem=0.25,&END
&DEFN action='configuration',&END
```

where `rotate` specifies the direction of rotation, and `overlap_tandem` is only required for tandem helicopters. The convention is that the first rotor is the main rotor for the helicopter 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, and tiltrotor configurations. The analysis creates the following input, through the code in the file `input_config.f90`. Note that all input quantities have default values.

### 2-1 All Configurations

a) Components: `nRotor=2, 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=9, selected by FltAircraft%STATE_trim=IDENT_trim`

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

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), Mauxtanks=1, WTank%ntank\_int=1, WTank%nplumb=2

f) Rotor

First rotor is primary: kPropulsion=1, KIND\_xmsn=1

Second rotor is dependent: kPropulsion=1, KIND\_xmsn=0, INPUT\_gear=2 (input quantity is gear ratio)

Configuration: direction='main'

Drag: SET\_aeroaxes=1 (helicopter), ldrag=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\_latcyc=0, INPUT\_lngcyc=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  $C_T/\sigma$ )

KIND\_lngcyc=1, KIND\_latcyc=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: ldrag=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:  $nGear=1$ ,  $STATE\_gear\_wt=1$ 

## 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:  $MODEL\_drag=1$ ,  $ldrag=0.$  (not tilt)

Control:

$INPUT\_incid=0$ ,  $INPUT\_yaw=0$  (no connection to aircraft controls)

$T\_incid=0.$ ,  $T\_yaw=0.$  (all controls, all states)

## k) Jet Group, Charge Group

Control:

$INPUT\_amp=0$ ,  $INPUT\_incid=0$ ,  $INPUT\_yaw=0$  (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:  $INPUT\_coll=1$ ,  $INPUT\_latcyc=1$ ,  $INPUT\_Ingcyc=1$  (rotor control connection to aircraft controls)

control:  $T\_coll(1,1)=1.$ ,  $T\_latcyc(2,1) = -r$ ,  $T\_Ingcyc(3,1)=-1.$

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  $m = (2/\pi)(\cos^{-1} h - h\sqrt{1-h^2})$ ,  $h = 1 - \text{overlap\_tandem}$

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) Tail

Horizontal tail: T\_cont(6,1)=1. (elevator direct control)

Vertical tail: T\_cont(7,1)=1. (rudder direct control)

e) 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\_ingcyc=1 (rotor control connection to aircraft controls)

control: T\_coll(1,1)=1., T\_coll(4,1)= $r$ , T\_latcyc(2,1)=  $-r$ , T\_ingcyc(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\_ingcyc=1 (rotor control connection to aircraft controls)

control: T\_coll(1,1)=1., T\_coll(4,1)= $r$ , T\_latcyc(2,1)=  $-r$ , T\_ingcyc(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','ingcyc','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\_ingcyc=1 (rotor control connection to aircraft controls)

helicopter mode control: T\_coll(1,1)=1., T\_coll(2,1)=-1., T\_ingcyc(3,1)=-1., T\_ingcyc(4,1)=1.

Second rotor is left rotor: otherRotor=1, rotate=-Rotor(1)%rotate

helicopter mode control: INPUT\_coll=1, INPUT\_ingcyc=1 (rotor control connection to aircraft controls)

helicopter mode control: T\_coll(1,1)=1., T\_coll(2,1)=1., T\_ingcyc(3,1)=-1., T\_ingcyc(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), ldrag=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 'ingcyc')

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, ldrag=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.

## Chapter 3

**Parameters and Constants**

Parameters	Value				
ncasemax	10	nmissmax	20	mrmax	40
nfilemax	40	nsegmax	20	mpsimax	36
nrotormax	8	nfltmax	21	npanelmax	5
npropmax	4	ndesignmax	41	nauxtankmax	4
nengmax	8	ncontmax	20	ngearmax	8
njetmax	4	nsweepmax	200	nratemax	20
nchrmax	4	ntrimstatemax	20	nengtmax	10
nstatemax	10	mtrimmax	16	nengkmax	6
nwingmax	8	nvelmax	20	nspeedmax	5
ntailmax	6	ntablemax	16	nrowmax	4000
ntankmax	4	nrmx	51	naeromax	100

Constants	Value				
ACTION_error	0	SET_takeoff_liftoff	4	TRIM_QUANT_Jmargin	16
ACTION_file	1	SET_takeoff_rotation	5	TRIM_QUANT_charge	17
ACTION_ident	2	SET_takeoff_transition	6	TRIM_QUANT_Cmargin	18
ACTION_list	3	SET_takeoff_climb	7	TRIM_QUANT_tank	19
ACTION_copy	4	SET_takeoff_brake	8	TRIM_QUANT_Bmargin	20
ACTION_init	5	MAX_QUANT_none	0	TRIM_QUANT_rotorL	21
ACTION_delete	6	MAX_QUANT_end	1	TRIM_QUANT_rotorFL	22
ACTION_delone	7	MAX_QUANT_range	2	TRIM_QUANT_CLs	23
ACTION_dellast	8	MAX_QUANT_rangelow	3	TRIM_QUANT_rotorV	24
ACTION_config	9	MAX_QUANT_range100	4	TRIM_QUANT_rotorX	25
ACTION_nosize	10	MAX_QUANT_climb	5	TRIM_QUANT_rotorFX	26
ACTION_desc	11	MAX_QUANT_angle	6	TRIM_QUANT_CXs	27

ACTION_soln	12	MAX_QUANT_power	7	TRIM_QUANT_XoQ	28
ACTION_endfile	13	MAX_QUANT_PoV	8	TRIM_QUANT_CTs	29
ACTION_endcase	14	MAX_QUANT_alt	9	TRIM_QUANT_Tmargs	30
ACTION_endjob	15	MAX_QUANT_Pmargin	10	TRIM_QUANT_Tmargt	31
STATE_newcase	1	MAX_QUANT_Qmargin	11	TRIM_QUANT_betac	32
STATE_onecase	2	MAX_QUANT_PQmargin	12	TRIM_QUANT_betas	33
STATE_endofjob	3	MAX_QUANT_Jmargin	13	TRIM_QUANT_hubMx	34
STATE_init	1	MAX_QUANT_PJmargin	14	TRIM_QUANT_hubMy	35
STATE_size	2	MAX_QUANT_QJmargin	15	TRIM_QUANT_hubQ	36
STATE_miss	3	MAX_QUANT_PQJmargin	16	TRIM_QUANT_wingL	37
STATE_perf	4	MAX_QUANT_Bmargin	17	TRIM_QUANT_wingfL	38
STATE_maps	5	MAX_QUANT_Lmargin	18	TRIM_QUANT_CL	39
STATE_out	6	MAX_QUANT_Tmargs	19	TRIM_QUANT_Lmargin	40
SIZE_perf_engine	1	MAX_QUANT_Tmargt	20	TRIM_QUANT_tailL	41
SIZE_perf_rotor	2	MAX_VAR_none	0	TRIM_VAR_not_found	0
SIZE_perf_none	3	MAX_VAR_vel	-1	TRIM_VAR_pitch	-1
SIZE_jet_jet	1	MAX_VAR_ROC	-2	TRIM_VAR_roll	-2
SIZE_jet_none	2	MAX_VAR_side	-3	TRIM_VAR_ROC	-3
SIZE_charge_chrg	1	MAX_VAR_alt	-4	TRIM_VAR_side	-4
SIZE_charge_none	2	MAX_VAR_turn	-5	TRIM_VAR_speed	-5
SIZE_rotor_none	1	MAX_VAR_pullup	-6	TRIM_VAR_turn	-6
SIZE_rotor_radius	2	MAX_VAR_xaccF	-7	TRIM_VAR_pullup	-7
SIZE_rotor_thrust	3	MAX_VAR_yaccF	-8	TRIM_VAR_Vtip	-8
SET_rotor_radius	1	MAX_VAR_zaccF	-9	TRIM_VAR_Nspec	-9
SET_rotor_DL	2	MAX_VAR_xaccI	-10	AERO_VAR_none	0
SET_rotor_ratio	3	MAX_VAR_yaccI	-11	AERO_VAR_not_found	-1
SET_rotor_scale	4	MAX_VAR_zaccI	-12	AERO_VAR_alpha	-2
SET_rotor_not_radius	5	MAX_VAR_xaccG	-13	AERO_VAR_beta	-3
SET_wing_area	1	MAX_VAR_yaccG	-14	RCCONFIG_rotorcrafft	0
SET_wing_WL	2	MAX_VAR_zaccG	-15	RCCONFIG_helicopter	1
SET_wing_not_area	3	MAX_VAR_pitch	-16	RCCONFIG_tandem	2
SET_wing_span	4	MAX_VAR_roll	-17	RCCONFIG_coaxial	3
SET_wing_ratio	5	MAX_VAR_Vtip	-18	RCCONFIG_tiltrotor	4
SET_wing_radius	6	MAX_VAR_Nspec	-19	ROTORCONFIG_main	1

SET_wing_width	7	SET_vel_general	1	ROTORCONFIG_tail	2
SET_wing_hub	8	SET_vel_hover	2	ROTORCONFIG_prop	3
SET_wing_panel	9	SET_vel_vert	3	ROTORCONFIG_tandem	4
SET_wing_not_span	10	SET_vel_right	4	ROTORCONFIG_coaxial	5
SET_tank_input	1	SET_vel_left	5	ROTORCONFIG_tiltrotor	6
SET_tank_miss	2	SET_vel_rear	6	ROTORCONFIG_not_twin	7
SET_tank_misspower	3	SET_vel_Vfwd	7	SET_geom_standard	0
SET_SDGW_input	1	SET_vel_Vmag	8	SET_geom_tiltrotor	1
SET_SDGW_fDGW	2	SET_vel_climb	9	SET_geom_coaxial	2
SET_SDGW_maxfuel	3	SET_vel_takeoff	10	SET_geom_tandem	3
SET_SDGW_perf	4	SET_vel2_TAS	1	SET_geom_tailrotor	4
SET_WMTO_input	1	SET_vel2_CAS	2	MODEL_twin_none	0
SET_WMTO_fDGW	2	SET_vel2_Mach	3	MODEL_twin_sidebyside	1
SET_WMTO_maxfuel	3	SET_atmos_input	-1	MODEL_twin_coaxial	2
SET_WMTO_perf	4	SET_atmos_dens	-2	MODEL_twin_tandem	3
SET_limit_input	1	SET_atmos_std	1	tablevar_none	0
SET_limit_Ratio	2	SET_atmos_std_dtemp	2	tablevar_mu	1
SET_limit_Pav	3	SET_atmos_std_temp	3	tablevar_muz	2
SET_limit_Preq	4	SET_atmos_polar	4	tablevar_alpha	3
SET_GW_DGW	1	SET_atmos_polar_dtem	5	tablevar_muTPP	4
SET_GW_SDGW	2	SET_atmos_polar_temp	6	tablevar_muzTPP	5
SET_GW_WMTO	3	SET_atmos_trop	7	tablevar_alphaTPP	6
SET_GW_fDGW	4	SET_atmos_trop_dtemp	8	tablevar_CT_s	7
SET_GW_fSDGW	5	SET_atmos_trop_temp	9	tablevar_Mx	8
SET_GW_fWMTO	6	SET_atmos_hot	10	tablevar_Mat	9
SET_GW_input	7	SET_atmos_hot_dtemp	11	SET_panel_free	0
SET_GW_maxP	8	SET_atmos_hot_temp	12	SET_panel_span	1
SET_GW_maxQ	9	SET_atmos_hot_table	13	SET_panel_bratio	2
SET_GW_maxPQ	10	SET_Vtip_input	1	SET_panel_edge	3
SET_GW_maxJ	11	SET_Vtip_ref	2	SET_panel_station	4
SET_GW_maxPJ	12	SET_Vtip_speed	3	SET_panel_radius	5
SET_GW_maxQJ	13	SET_Vtip_conv	4	SET_panel_width	6
SET_GW_maxPQJ	14	SET_Vtip_hover	5	SET_panel_hub	7
SET_GW_Source	15	SET_Vtip_cruise	6	SET_panel_adjust	8

SET_GW_payfuel	16	SET_Vtip_man	7	SET_panel_area	9
SET_GW_paymiss	17	SET_Vtip_OEI	8	SET_panel_Sratio	10
SET_UL_pay	1	SET_Vtip_xmsn	9	SET_panel_chord	11
SET_UL_fuel	2	SET_Vtip_mu	10	SET_panel_cratio	12
SET_UL_payfuel	3	SET_Vtip_Mat	11	SET_panel_taper	13
SET_UL_miss	4	STATE_LG_default	0	SET_tail_area	1
SET_UL_paymiss	5	STATE_LG_extend	1	SET_tail_vol	2
SET_pay_none	1	STATE_LG_retract	2	SET_tail_span	3
SET_pay_input	2	TRIM_QUANT_not_found	0	SET_tail_AR	4
SET_pay_delta	3	TRIM_QUANT_forcex	1	SET_tail_chord	5
SET_pay_scale	4	TRIM_QUANT_forcey	2	MODEL_engine_RPTM	1
KIND_MissSeg_taxi	1	TRIM_QUANT_forcez	3	MODEL_engine_table	2
KIND_MissSeg_dist	2	TRIM_QUANT_momentx	4	MODEL_engine_recip	3
KIND_MissSeg_time	3	TRIM_QUANT_momenty	5	MODEL_engine_comp	4
KIND_MissSeg_hold	4	TRIM_QUANT_momentz	6	MODEL_engine_compreact	5
KIND_MissSeg_climb	5	TRIM_QUANT_nz	7	MODEL_engine_motor	6
KIND_MissSeg_spiral	6	TRIM_QUANT_nx	8	MODEL_engine_gen	7
KIND_MissSeg_fuel	7	TRIM_QUANT_ny	9	MODEL_engine_motorgen	8
KIND_MissSeg_burn	8	TRIM_QUANT_power	10	MODEL_engine_motorcell	9
KIND_MissSeg_takeoff	9	TRIM_QUANT_Pmargin	11	MODEL_jet_RPJEM	1
SET_takeoff_none	0	TRIM_QUANT_Qmargin	12	MODEL_jet_react	2
SET_takeoff_start	1	TRIM_QUANT_powerEG	13	MODEL_jet_simple	3
SET_takeoff_groundrun	2	TRIM_QUANT_PEGmarg	14	MODEL_charge_fuelcell	1
SET_takeoff_enginefail	3	TRIM_QUANT_thrust	15	MODEL_charge_solarcell	2

## Chapter 4

**Common: Job**

Variable	Type	Description	Default
		NDARC	
		Version (set by main program)	
version	c*6	number n.n	
modification	c*32	modification	
versionout	c*64	string for headers (Version n.n, modification "xxx")	
		+ 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
<hr/>			
		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	
<hr/>			
		+ 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

---

### Analysis

kcase	int	current case number
ncase	int	number of cases (maximum ncasemax)
case_state	int	case state
job_state	int	job state
out_design_state	int	design output state (1 file open)
out_perf_state	int	performance output state (1 file open)
out_geom_state	int	geometry output state (1 file open)
out_error_state	int	errors output state (1 file open)
nuinit	int	nuout or nuerror
fscratch	FltState	scratch structure

		<b>Input</b>
kind_input	int	file input status (0 for primary file, 1 for secondary file, 2 for aircraft or solution file)
nread	int	unit number for input (nuin for primary file, nufile for secondary file)
		Input file identification (stored from action=IDENT data)
ninputfile	int	number of identifications (maximum nfilemax; first is standard input)
input_title(nfilemax)	c*80	title
input_created(nfilemax)	c*20	creation date
theDesign(ncasemax)	Design	Design
theInput	Design	Input
theLastCaseInput	Design	Input from last case

---

system data = Job + theDesign(ncase) + theInput + theLastCaseInput  
 all data structure parameters = input (can be changed by analysis) or other (generated by analysis)  
 theInput used for input (not changed by analysis)  
 theLastCaseInput used to print only what changed from last case  
 after case input concluded, kcase incremented and theInput copied to theDesign(kcase)

---

		<b>CPU time</b>
CPUtime_case_start(ncasemax)		
	real	case start
CPUtime_case_end(ncasemax)	real	case end
CPUtime_case(ncasemax)	real	case
CPUtime_job	real	job
		<b>Clock time</b>
DateTime_case_start(8,ncasemax)		
	int	case start
DateTime_case_end(8,ncasemax)		
	int	case end
ElapsedTime_case(ncasemax)	real	case
ElapsedTime_job	real	job

		Case dimensions
nrotor_case	int	number of rotors (Aircraft)
nwing_case	int	number of wings (Aircraft)
ntail_case	int	number of tails (Aircraft)
ntank_case	int	number of fuel tank systems (Aircraft)
npropulsion_case	int	number of propulsion groups (Aircraft)
nenginegroup_case	int	number of engine groups (Aircraft)
njetgroup_case	int	number of jet groups (Aircraft)
nchargegroup_case	int	number of charge groups (Aircraft)
nenginemodel_case	int	number of engine models (Aircraft)
nenginetable_case	int	number of engine tables (Aircraft)
ncompressormodel_case	int	number of compressor models (Aircraft)
nmotormodel_case	int	number of motor models (Aircraft)
njetmodel_case	int	number of jet models (Aircraft)
nfuelcellmodel_case	int	number of fuel cell models (Aircraft)
nsolarcellmodel_case	int	number of solar cell models (Aircraft)
nbatterymodel_case	int	number of battery models (Aircraft)
ncontrol_case	int	number of controls (Aircraft)
nstate_control_case	int	number of control states (Aircraft)
npanel_case(nwingmax)	int	number of wing panels (Wing)
mauxtanksizes_case(ntankmax)	int	number of aux tank sizes (FuelTank)
ngear_case(npropmax)	int	number of drive system states (Propulsion)
nstate_trim_case	int	number of trim states (Aircraft)
mtrim_case(ntrimstatemax)	int	number of trim variables (Aircraft)
nwoful_case	int	number of other fixed useful load categories (System)
nrate_case	int	number of engine ratings (for EngineParam output)
		Job constants
pi	real	$\pi$
twopi	real	$2\pi$
halfpi	real	$\pi/2$
degrad	real	degree/radian = $180/\pi$
raddeg	real	radian/degree = $\pi/180$

		Case constants
gravity	real	gravity $g$ (ft/sec <sup>2</sup> or m/sec <sup>2</sup> )
density_sls	real	SLS density $\rho_0$ (slug/ft <sup>3</sup> or kg/m <sup>3</sup> )
		Conversion factors
powerconv	real	power (hp from ft-lb/sec; kW from m-N/sec)
knotsconv	real	speed (knots from ft/sec or m/sec)
nmconv	real	range (nm from ft or m)
weightconv	real	weight (lb from lb; kg from N)
massconv	real	mass (slug from lb; kg from kg)
volumeconv	real	volume (gal from ft <sup>3</sup> ; liter from m <sup>3</sup> )
		Conversion factors for scaled $D/q$
DoQconv23	real	$D/q = kW^{2/3}$ (ft <sup>2</sup> from $k=m^2/kg^{2/3}$ ; m <sup>2</sup> from $k=ft^2/lb^{2/3}$ ; depending on Units_Dscale)
DoQconv12	real	$D/q = kW^{1/2}$ (ft <sup>2</sup> from $k=m^2/kg^{1/2}$ ; m <sup>2</sup> from $k=ft^2/lb^{1/2}$ ; depending on Units_Dscale)
		Conversion factors for mission and flight condition input
uconv_vel	real	velocity (knots from input)
uconv_alt	real	altitude (ft or m from input)
uconv_pay	real	payload (lb or kg from input)
uconv_time	real	time (minutes from input)
uconv_dist	real	distance (nm from input)
uconv_drag	real	drag (ft <sup>2</sup> or m <sup>2</sup> from input)
uconv_ROC	real	rate of climb (ft/sec or m/sec from input)
		Conversion factors for weight equations
wtconv_hp	real	power (hp from hp or kW)
wtconv_lb	real	weight (lb from lb or kg)
wtconv_frc	real	force (lb from lb or N)
wtconv_ft	real	length (ft from ft or m)
wtconv_ft2	real	area (ft <sup>2</sup> from ft <sup>2</sup> or m <sup>2</sup> )
wtconv_gal	real	fuel (gal from gal or liter)
wtconv_slug	real	slug (slug/lb or kg/kg)
wtconv_in	real	inches (in/ft or m/m)
wtconv_kW	real	power (kW from hp or kW)
wtconv_m	real	meter (m from ft or m)
		Conversion factors for energy
Econv_kg	real	weight (kg from lb or kg)

Econv_L	real	volume (liter from gal or liter)
Econv_dE	real	energyflow (MJ/hr from hp or kW)
Conversion factors		
DLconv	real	disk loading (lb/ft <sup>2</sup> from lb/ft <sup>2</sup> or kg/m <sup>2</sup> )
tonconv	real	ton (from lb or kg)
rangeconv	real	range for fuel=1%GW (nm from 1/(lb/hp-hr) or 1/(kg/kW-hr), times ln(1/.99))
endconv	real	endurance for fuel=1%GW (min from hr, times ln(1/.99))
Output		
WRITEenergy_case	int	write fuel energy for burn weight
Units for output		
Uwrite	int	analysis units (from Cases)
Uwrite_temp	int	mission units, temperature (from Cases)
Ukts	c*10	speed (knots, mph, kph, ft/sec, m/sec); uconv_vel
UROC	c*10	rate of climb (ft/min, ft/sec, m/sec); uconv_ROC
Udist	c*10	distance (nm, mile, km); uconv_dist
Utime	c*10	time (min, hr); uconv_time
UDoQ	c*10	drag (ft <sup>2</sup> , m <sup>2</sup> ); uconv_drag
Upay	c*10	payload (lb, kg); uconv_pay
Ualt	c*10	altitude (ft, m); uconv_alt
Ulen	c*10	length
Uarea	c*10	area
Uvel	c*10	velocity
Utemp	c*10	temperature
Uwt	c*10	weight
Upwr	c*10	power
Ufuelflow	c*10	fuel flow
Umassflow	c*10	mass flow
Uscf	c*10	sfc
Utsfc	c*10	thrust sfc
Uspecrange	c*10	specific range
Ufuelff	c*10	fuel efficiency
Uproductivity	c*10	productivity
Ufrc	c*10	force
Umom	c*10	moment

Uque	$c*10$	dynamic pressure
Udens	$c*10$	density

## Chapter 5

**Structure: Design**

Variable	Type	Description	Default
Cases	Cases	Cases	
Size	Size	Size Aircraft for Design Conditions and Missions	
OffDesign	OffDesign	Mission Analysis	
Performance	Performance	Flight Performance Analysis	
MapEngine	MapEngine	Map of Engine Performance	
MapAero	MapAero	Map of Airframe Aerodynamics	
Solution	Solution	Solution Procedures	
Cost	Cost	Cost	
Aircraft	Aircraft	Aircraft	
Systems	Systems	Systems	
Fuselage	Fuselage	Fuselage	
LandingGear	LandingGear	Landing Gear	
Rotor(nrotormax)	Rotor	Rotors	
Wing(nwingmax)	Wing	Wings	
Tail(ntailmax)	Tail	Tails	
FuelTank(ntankmax)	FuelTank	Fuel Tank Systems	
Propulsion(npropmax)	Propulsion	Propulsion Groups	
EngineGroup(nengmax)	EngineGroup	Engine Groups	
JetGroup(njetmax)	JetGroup	Jet Groups	
ChargeGroup(nchrgmax)	ChargeGroup	Charge Groups	
EngineModel(nengmax)	EngineModel	Engine Models	
EngineTable(nengmax)	EngineTable	Engine Tables	
CompressorModel(nengmax)	CompressorModel	Compressor Models	
MotorModel(nengmax)	MotorModel	Motor Models	
JetModel(njetmax)	JetModel	Jet Models	
FuelCellModel(nchrgmax)	FuelCellModel	Fuel Cell Models	
SolarCellModel(nchrgmax)	SolarCellModel	Solar Cell Models	
BatteryModel(ntankmax)	BatteryModel	Battery Models	

## Chapter 6

**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	
timedate	c*20	+ time-date 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			
		+ Write Input Parameters	
WRITE_input	int	+ selection (0 none, 1 all, 2 first case, 3 group selection)	2
		+ group selection	
WRITE_input_Job	int	+ Job	0
WRITE_input_Cases	int	+ Cases	0
WRITE_input_Size	int	+ Size	0
WRITE_input_OffDesign	int	+ OffDesign	0



WRITE_input_Performance	int	+	Performance	0
WRITE_input_MapEngine	int	+	MapEngine	0
WRITE_input_MapAero	int	+	MapAero	0
WRITE_input_Solution	int	+	Solution	0
WRITE_input_Cost	int	+	Cost	0
WRITE_input_Aircraft	int	+	Aircraft	0
WRITE_input_Systems	int	+	Systems	0
WRITE_input_Fuselage	int	+	Fuselage	0
WRITE_input_LandingGear	int	+	LandingGear	0
WRITE_input_Rotor(nrotormax)	int	+	Rotor	0
WRITE_input_Wing(nwingmax)	int	+	Wing	0
WRITE_input_Tail(ntailmax)	int	+	Tail	0
WRITE_input_FuelTank(ntankmax)	int	+	FuelTank	0
WRITE_input_Propulsion(npropmax)	int	+	Propulsion	0
WRITE_input_EngineGroup(nengmax)	int	+	EngineGroup	0
WRITE_input_JetGroup(njetmax)	int	+	JetGroup	0
WRITE_input_ChargeGroup(nchrgmax)	int	+	ChargeGroup	0
WRITE_input_EngineModel(nengmax)	int	+	EngineModel	0
WRITE_input_EngineTable(nengmax)	int	+	EngineTable	0
WRITE_input_CompressorMo(nengmax)	int	+	CompressorModel	0
WRITE_input_MotorModel(nengmax)	int	+	MotorModel	0
WRITE_input_JetModel(njetmax)	int	+	JetModel	0
WRITE_input_FuelCellMode(nchrgmax)				

	int	+	FuelCellModel	0
WRITE_input_SolarCellMod(nchrgmax)	int	+	SolarCellModel	0
WRITE_input_BatteryModel(ntankmax)	int	+	BatteryModel	0
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_wt_comp	int	+	weight statement, component (0 for none)	1
WRITE_energy	int	+	fuel energy for burn weight (0 for none)	0
WRITE_flight	int	+	flight state, component loads (0 for none)	1

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 $g$	
		+	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 <sup>2</sup> or m <sup>2</sup> ; 1 ft <sup>2</sup> , 2 m <sup>2</sup> )	0
Units_ROC	int	+	rate of climb units (0 ft/min; 1 ft/sec, 2 m/sec)	0

Units_Dscale	int	+	units for parameters	
		+	input $D/q$ scaled with gross weight (0 analysis default, 1 English, 2 SI)	0

---

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 for case	
inCases	int		Cases	
inSize	int		Size	
inSizeCondition(nfltmax)	int		SizeCondition	
inSizeMission(nmissmax)	int		SizeMission	
inOffDesign	int		OffDesign	
inOffMission(nmissmax)	int		OffMission	
inPerformance	int		Performance	
inPerfCondition(nfltmax)	int		PerfCondition	
inMapEngine	int		MapEngine	
inMapAero	int		MapAero	
inSolution	int		Solution	
			Last input	
lastSizeCondition	int		SizeCondition	
lastSizeMission	int		SizeMission	
lastOffMission	int		OffMission	
lastPerfCondition	int		PerfCondition	

---

case input of other structures recorded in Aircraft structure  
 there must be input for systems, fuselage, landing gear, fuel tank  
 there must be input for all structures used

---

## Chapter 7

---

**Structure: Size**

Variable	Type	Description	Default
SizeParam	SizeParam	Size Aircraft for Design Conditions and Missions Parameters Sizing Flight Conditions	
FltCond(nfltmax)	FltCond	conditions	
FltState(nfltmax)	FltState	conditions	
Mission(nmissmax)	Mission	Design Missions missions	

## Chapter 8

**Structure: SizeParam**

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_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)	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:

'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\_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 weight empty obtained by adjusting contingency weight

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(fFuelCap\*(maximum mission fuel), (maximum mission fuel)+(reserve fuel))

'miss+power' = calculate from mission fuel used and mission battery discharge power

SET\_SDGW, structural design gross weight:

'input' = input

'f(DGW)' = based on DGW; SDGW=dSDGW+fSDGW\*DGW

'maxfuel' = based on fuel state; SDGW=dSDGW+fSDGW\*GW, GW=DGW-Wfuel+fFuelSDGW\*Wfuel\_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; WMTO=dWMTO+fWMTO\*DGW

'maxfuel' = based on maximum fuel; WMTO=dWMTO+fWMTO\*GW, GW=DGW-Wfuel+Wfuel\_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,  $fPlimit\_ds \sum (N_{eng} P_{eng})$

'Pav' = from engine power available at transmission sizing conditions and missions (DESIGN\_xmsn)

$fPlimit\_ds (\Omega_{ref} / \Omega_{prim}) \sum (N_{eng} P_{av})$

'Preq' = from engine power required at transmission sizing conditions and missions (DESIGN\_xmsn)

$fPlimit\_ds (\Omega_{ref} / \Omega_{prim}) \sum (N_{eng} P_{req})$

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)

convergence may be improved if do not apply drive system limits to power available (FltState%SET\_Plimit=off)

for transmission sizing conditions and mission segments (DESIGN\_xmsn)

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\_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'

---

		Specification
iSIZE_perf(npropmax)	int	performance (SIZE_perf_engine, rotor, param, none)
iSIZE_jet(njetmax)	int	performance (SIZE_jet_jet, param, none)
iSIZE_charge(nchrgmax)	int	performance (SIZE_charge_chrg, param, none)
iSIZE_rotor(nrotormax)	int	rotor sized (SIZE_rotor_radius, thrust, none)
iSET_rotor_radius(nrotormax)	int	rotor radius (SET_rotor_radius, DL, ratio, scale, not_radius)
FIX_rotor_CWs(nrotormax)	int	rotor $C_W/\sigma$ (1 fixed, 0 not)
FIX_rotor_Vtip(nrotormax)	int	rotor $V_{tip}$ (1 fixed, 0 not)
FIX_rotor_sigma(nrotormax)	int	rotor $\sigma$ (1 fixed, 0 not)
iSET_wing_area(nwingmax)	int	wing area (SET_wing_area, WL, not_area)
iSET_wing_span(nwingmax)	int	wing span (SET_wing_span, ratio, radius, width, hub, panel, not_span)
FIX_wing_chord(nwingmax)	int	wing chord (1 fixed, 0 not)
FIX_wing_AR(nwingmax)	int	wing aspect ratio (1 fixed, 0 not)
iSET_tank(ntankmax)	int	fuel tank (SET_tank_input, miss, misspower)
iSET_SDGW	int	SDGW (SET_SDGW_input, fDGW, maxfuel, perf)
iSET_WMTO	int	WMTO (SET_WMTO_input, fDGW, maxfuel, perf)
iSET_limit_ds(npropmax)	int	drive system torque limit (SET_limit_input, ratio, Pav, Preq)
nSIZE_engine(npropmax)	int	conditions and missions for size engine or rotor
nSIZE_jet(njetmax)	int	conditions and missions for size jet group
nSIZE_charge(nchrgmax)	int	conditions and missions for size charge group
nDESIGN_GW	int	design conditions and missions for DGW
nDESIGN_xmsn(npropmax)	int	design conditions and missions for transmission
nDESIGN_sdgw	int	design conditions for SDGW
nDESIGN_wmto	int	design conditions for WMTO
nDESIGN_tank	int	design missions for fuel tank
nDESIGN_thrust	int	design conditions and missions for rotor thrust
		Size aircraft
kind_iter_size	int	kind iteration, performance (0 none, 1 size engine or radius or jet group or charge group)

kind_iter_param	int	kind iteration, parameters (0 none, 1 calculate parameters)	
issizeconv	int	converged (0 not)	
count_size	int	number of iterations, performance loop	
count_param	int	number of iterations, parameter loop	
count_total	int	total number of iterations	
error_engine(nengmax)	real	error ratio, engine	
error_jet(njetmax)	real	error ratio, jet	
error_charge(nchrgmax)	real	error ratio, charge	
error_rotor(nrotormax)	real	error ratio, rotor	
error_DGW	real	error ratio, DGW	
error_xmsn(npropmax)	real	error ratio, Plimit	
error_sdgw	real	error ratio, structural design gross weight	
error_wmto	real	error ratio, maximum takeoff weight	
error_tank	real	error ratio, Wfuelcap	
error_thrust(nrotormax)	real	error ratio, thrust	
error_WE	real	error ratio, WE	
Pratio(npropmax)	real	ratio $P_{reqPG}/P_{avPG}$ (max all sizing conditions and missions)	
PratioEG(nengmax)	real	ratio $P_{reqEG}/P_{avEG}$ (max all sizing conditions and missions)	
Jratio(njetmax)	real	ratio $T_{reqJG}/T_{avJG}$ (max all sizing conditions and missions)	
Cratio(nchrgmax)	real	ratio $P_{reqCG}/P_{avCG}$ (max all sizing conditions and missions)	
nFltCond_out	int	number of conditions for output	
nMission_out	int	number of missions for output	
		+ 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 9

---

**Structure: OffDesign**

Variable	Type	Description	Default
OffParam	OffParam	Mission Analysis Parameters	
Mission(nmissmax)	Mission	Missions	

## Chapter 10

**Structure: OffParam**

Variable	Type	Description	Default
		+ Mission Analysis	
title	c*100	+ title	
notes	c*1000	+ notes	
nMission_out	int	Analyze mission number of missions for output	
nMission	int	+ Missions number of missions (maximum nmissmax)	0
<hr/> <p>mission analysis input required if Cases%TASK_Mission=1</p> <p>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</p> <hr/>			

---

**Structure: Performance**

Variable	Type	Description	Default
PerfParam	PerfParam	Flight Performance Analysis Parameters	
FltCond(nfltmax)	FltCond	Performance Flight Conditions conditions	
FltState(nfltmax)	FltState	conditions	

**Structure: PerfParam**

Variable	Type	Description	Default
		+ Flight Performance Analysis	
title	c*100	+ title	
notes	c*1000	+ notes	
		Analyze performance	
nFltCond_out	int	number of conditions for output (including sweeps)	
nsweep_total	int	total number of sweep conditions	
		+ Performance Flight Conditions	
nFltCond	int	+ number of conditions (maximum nfltmax)	0
<hr/>			
flight performance analysis input required if Cases%TASK_Perf=1			
input one condition (FltCond and FltState variables) in PerfCondition namelist			
<hr/>			

## Chapter 13

**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
<hr/> engine map only available for RPTEM engine model  engine map input required if Cases%TASK_Map_engine=1 only performance parameters or model parameters used <hr/>			
		+ Performance	
SET_var(5)	int	+ independent variables (0 none, 1 altitude, 2 temperature, 3 flight speed, 4 turbine speed, 5 power factor)	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.
		+ temperature $\tau$ or temperature increment $\Delta T$ (Units_temp)	
temp_min	real	+ minimum	0.
temp_max	real	+ maximum	100.
temp_inc	real	+ increment	10.

Structure: MapEngine

48

		+	flight speed $V$ (TAS, Units_vel)	
Vkts_min	real	+	minimum	0.
Vkts_max	real	+	maximum	200.
Vkts_inc	real	+	increment	50.
SET_rpm	int	+	engine turbine speed $N$ (1 rpm, 2 percent)	2
Nturbine_min	real	+	minimum	90.
Nturbine_max	real	+	maximum	110.
Nturbine_inc	real	+	increment	5.
		+	fraction of rated engine power available $f_P$ (0. to 1.+)	
fPower_min	real	+	minimum	.1
fPower_max	real	+	maximum	1.
fPower_inc	real	+	increment	.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 minimum 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.



Structure: MapEngine

49

		+	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 turbine speed, $N/N_{spec}$ (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)

---

		Specification
kEngineModel	int	engine model
iSET_atmos	int	atmosphere (SET_atmos_xxx)

## Chapter 14

**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
<hr/>			
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)			
<hr/>			
		+ 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(ncntmax)	real	+	aircraft controls	0.
tilt	real	+	tilt	0.
alpha	real	+	angle of attack $\alpha$	0.
beta	real	+	sideslip angle $\beta$	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

---

iSTATE_LG	int	Operating Condition landing gear state (STATE_LG_extend, retract)
nvar(6)	int	Independent variables (AERO_VAR_none, alpha, beta or control number)
ivar(3,6)	int	number of independent variables variables (drag, side, lift, roll, pitch, yaw)
nang	int	Tables number of angles (maximum naeromax)
ang(naeromax)	real	angle values
ncnt	int	number of controls (maximum naeromax)
cnt(naeromax)	real	control values
ngam	int	number of gamma (maximum naeromax)
gam(naeromax)	real	gamma values

## Chapter 15

**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_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 $W_{\text{pay}}$ (Units_pay)	0.
Npass	int	+ number of passengers $N_{\text{pass}}$	0
Wpay_cargo	real	+ cargo $W_{\text{cargo}}$ (Units_pay)	0.
Wpay_extload	real	+ external load $W_{\text{ext-load}}$ (Units_pay)	0.
Wpay_ammo	real	+ ammunition $W_{\text{ammo}}$ (Units_pay)	0.
Wpay_weapons	real	+ weapons $W_{\text{weapons}}$ (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 $N_{\text{auxtank}}$ (each aux tank size)	
		+	fixed useful load	
dWcrew	real	+	crew weight increment	0.
dNcrew	int	+	number of crew increment $\delta N_{\text{crew}}$	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)

'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

$\min(P_{limit} - P_{req}) = 0$ , 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\_UL, set useful load: with fixed useful load adjustments in fallout weight

'pay' = input payload weight ( $W_{pay}$ ); fuel weight fallout

'fuel' = input fuel weight ( $dFuel$ ,  $fFuel$ ,  $N_{auxtank}$ ); 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:  $W_{fuel} = \min(dFuel + fFuel * W_{fuel-cap}, W_{fuel-cap}) + \sum N_{auxtank} * W_{aux-cap}$

auxiliary fuel tanks: SET\_auxtank used for fallout fuel weight (SET\_UL='pay')

adjust  $N_{auxtank}$  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
nquant_sweep	int	+	number of swept quantities (1 to 3)	1
nsweep	int	+	list, number of values (maximum nsweepmax)	
quant_sweep(3)	c*12	+	quantity (parameter name)	
		+	range	
sweep_first(3)	real	+	first parameter value	
sweep_last(3)	real	+	last parameter value	
sweep_inc(3)	real	+	parameter increment	
		+	list	
sweep(nsweepmax)	real	+	parameter 1 values	
sweep2(nsweepmax)	real	+	parameter 2 values	
sweep3(nsweepmax)	real	+	parameter 3 values	

---

Parameter sweep: only for performance flight conditions, not sizing flight conditions

maximum total number of values for all conditions is nsweepmax

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

sweep\_inc of first quantity determines number of values, sweep\_inc of other quantities not used

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

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;  
 1 if absent  
 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

---

parent	int	parent (1 Size, 2 Performance)
kFltCond	int	FltCond number
kcol_out	int	performance output column (first for sweep)
Specification		
iSET_GW	int	gross weight (SET_GW_xxx)
iSET_maxGW	int	max gross weight (0 no iteration; SET_GW_maxP, maxQ, maxPQ, maxJ, maxPJ, maxQJ, maxPQJ)
iSET_UL	int	useful load (SET_UL_pay, fuel, payfuel)
iSETPmargin(npropmax)	int	power margin as quantity (3 max GW, 2 max effort, 1 trim); not used to size engine or rotor
iSETQmargin(npropmax)	int	torque margin as quantity (3 max GW, 2 max effort, 1 trim); not used to size transmission
iSETJmargin(njetmax)	int	jet thrust margin as quantity (3 max GW, 2 max effort, 1 trim); not used to size jet group
iSETCmargin(nchrgmax)	int	charger power margin as quantity (1 trim); not used to size charge group
iSETBmargin(ntankmax)	int	battery power margin as quantity (2 max effort, 1 trim); not used to size fuel tank
isFIX_GW	int	fixed gross weight; DESIGN_GW=0, DESIGN_sdgw=0, DESIGN_wmto=0
Parameter sweep		
kquant_sweep(3)	int	quantity number
label_sweep	c*8	quantity column label
vsweep(3,nsweepmax)	real	parameter values
fPower_original(nengmax)	real	fraction of rated engine power available
fThrust_original(njetmax)	real	fraction of rated jet thrust available
fCharge_original(nchrgmax)	real	fraction of rated charger power available

## Chapter 16

---

**Structure: Mission**

Variable	Type	Description	Default
MissParam	MissParam	Mission Profile Parameters	
MissSeg(nsegmax)	MissSeg	Mission Segments mission segments	
FltState(nsegmax)	FltState	flight conditions	

## Chapter 17

**Structure: MissParam**

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_UL	c*16	+ useful load	'pay+miss'
Wpay	real	+ input takeoff payload weight $W_{\text{pay}}$ (Units_pay)	0.
Npass	int	+ number of passengers $N_{\text{pass}}$	0
Wpay_cargo	real	+ cargo $W_{\text{cargo}}$ (Units_pay)	0.
Wpay_extload	real	+ external load $W_{\text{ext-load}}$ (Units_pay)	0.
Wpay_ammo	real	+ ammunition $W_{\text{ammo}}$ (Units_pay)	0.
Wpay_weapons	real	+ weapons $W_{\text{weapons}}$ (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 $N_{\text{auxtank}}$ (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 $f_{res}$	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:  $P_{req} = fPavP_{av} + dPav$   
 at mission segment MaxGW, minimum gross weight of designated segments  
 $\min((fP_{avPG} + d) - P_{reqPG}) = 0$ , 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  
 $\min(P_{limit} - P_{req}) = 0$ , 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:  $T_{req} = fTavT_{av} + dTav$   
 at mission segment MaxGW, minimum gross weight of designated segments  
 $\min((fT_{avJG} + d) - T_{reqJG}) = 0$ , 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

'pay+miss' = input payload, fuel weight from mission; gross weight fallout

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:  $W_{fuel} = \min(dFuel + fFuel * W_{fuel-cap}, W_{fuel-cap}) + \sum N_{auxtank} * W_{aux-cap}$   
 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  $W_G - W_{pay} = W_O + W_{fuel}$ )  
 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 ( $f_{res} W_{burn}$  or  $f_{res} E_{burn}$ ) or fraction of fuel capacity ( $f_{res} W_{fuel-cap}$  or  $f_{res} E_{fuel-cap}$ )

		+ 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

parent	int	parent (1 Size, 2 OffDesign)
kMission	int	Mission number
kcol_out	int	performance output column

		Specification
iSET_GW	int	gross weight (SET_GW_xxx)
iSET_maxGW	int	max gross weight (SET_GW_maxP, maxQ, maxPQ, maxJ, maxPJ, maxQJ, maxPQJ)
nSET_maxGW	int	number max gross weight segments
iSET_UL	int	useful load (SET_UL_pay, fuel, payfuel, miss, paymiss)
iSET_pay	int	payload changes (SET_pay_none, input, delta, scale)
iSETPmargin(npropmax)	int	power margin as quantity (all mission segments); not used to size engine or rotor
iSETQmargin(npropmax)	int	torque margin as quantity (all mission segments); not used to size transmission
iSETJmargin(njetmax)	int	jet thrust margin as quantity (all mission segments); not used to size jet group
iSETCmargin(nchrgmax)	int	charger power margin as quantity (all mission segments); not used to size charge group
iSETBmargin(ntankmax)	int	battery power margin as quantity (all mission segments); not used to size fuel tank
isFIX_GW	int	fixed gross weight; DESIGN_GW=0
		Segments
nreserve	int	number reserve segments
nadjust	int	number adjustable segments
kind_adjust	int	kind adjustable (0 none, 1 distance, 2 time)
kind_range	int	kind range credit (0 none, 1 all forward, 2 all backward, 3 both)
ntakeoff	int	number takeoff segments
		Iteration
kind_iter	int	kind iteration (0 none, 1 calculate mission fuel, 2 adjust mission, 3 only range credit or integration)
ismissconv	int	converged (0 not)
count_miss	int	number of iterations
error_miss(3)	real	error ratio (Wfuel, range credit, takeoff GW)
		Mission quantities
isFirstSol	int	first solution (initialize GW_to and Wfuel_to)
GW_to	real	takeoff gross weight (start of mission)
GW_endmiss	real	gross weight (end of mission, excluding reserve segments; last non-reserve segment)
GW_end	real	gross weight (end of mission; last segment)
Wfuel_to(ntankmax)	real	takeoff fuel weight (start of mission)
Wfuel_add(ntankmax)	real	added fuel weight (fill/add/drop during mission)
Wfuel_endmiss(ntankmax)	real	fuel weight (end of mission, excluding reserve segments; last non-reserve segment)
Wfuel_end(ntankmax)	real	fuel weight (end of mission; last segment)
Wburn(ntankmax)	real	weight fuel burned $W_{burn}$

Wres(ntankmax)	real	weight reserve fuel $W_{res}$ (maximum of fraction or reserve segments)
Wfuel_miss(ntankmax)	real	calculated mission fuel weight ( $W_{burn} + W_{res}$ )
Efuel_to(ntankmax)	real	takeoff fuel energy (start of mission)
Efuel_add(ntankmax)	real	added fuel energy (fill/add/drop during mission)
Efuel_endmiss(ntankmax)	real	fuel energy (end of mission, excluding reserve segments; last non-reserve segment)
Efuel_end(ntankmax)	real	fuel energy (end of mission; last segment)
Eburn(ntankmax)	real	energy fuel burned $E_{burn}$
Eres(ntankmax)	real	energy reserve fuel $E_{res}$ (maximum of fraction or reserve segments)
Efuel_miss(ntankmax)	real	calculated mission fuel energy ( $E_{burn} + E_{res}$ )
exceedP	int	exceed power available: any mission segment $P_{reqPG} > (1 + \epsilon)P_{avPG}$
exceedQ	int	exceed torque available: any mission segment $P_{reqPG} > (1 + \epsilon)P_{DSlimit}$
exceedJ	int	exceed jet thrust available: any mission segment $T_{reqJG} > (1 + \epsilon)T_{avJG}$
exceedC	int	exceed charger power available: any mission segment $P_{reqCG} > (1 + \epsilon)P_{avCG}$
exceedWf	int	exceed fuel capacity: any mission segment $W_{fuel} > (1 + \epsilon)W_{fuel-cap}$ or $E_{fuel} > (1 + \epsilon)E_{fuel-cap}$
exceedB	int	exceed battery power: any mission segment $ \dot{E}_{batt}  > (1 + \epsilon)P_{max}$
		Total mission, excluding reserve segments
endurance	real	endurance $E$ , block time (min)
range	real	range $R$ (nm)
airdist	real	air distance (nm)
blockspeed	real	block speed (kts; range/endurance)
range_factor	real	range factor $RF = R / \ln(W_{to}/(W_{to} - W_{burn}))$ (nm)
end_factor	real	efficiency factor $EF = E/2((W_{to}/(W_{to} - W_{burn}))^{3/2} - 1)$ (min)
fuel_eff	real	fuel efficiency $e = W_{pay}R/W_{burn}$ (ton-nm/lb or ton-nm/kg)
productivity_o	real	productivity $p = W_{pay}V/W_O$ (ton-kt/lb or ton-kt/kg)
productivity_f	real	productivity $p = W_{pay}V/W_{burn}$ (ton-kt/lb or ton-kt/kg)
		Cost
ASM	real	available seat miles
DOC	real	direct operating cost



## Chapter 18

**Structure: MissSeg**

Variable	Type	Description	Default
		+ Segment definition	
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 add fuel, 3 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 $\delta N_{\text{pass}}$	0

		+	fixed useful load	
dWcrew	real	+	crew weight increment	0.
dNcrew	int	+	number of crew increment $\delta N_{\text{crew}}$	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_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
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.
		+	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'

SET\_tank = 0: no requirement

SET\_tank = 1: target  $dTank+fTank*W_{fuel-cap}$  or  $dTank+fTank*E_{fuel-cap}$

SET\_tank = 2: target  $dTank+fTank*W_{fuel}$  or  $dTank+fTank*E_{fuel}$

SET\_tank = 3: increment  $dTank+fTank*W_{fuel-cap}$  or  $dTank+fTank*E_{fuel-cap}$

SET\_tank = 4: increment  $dTank+fTank*W_{fuel}$  or  $dTank+fTank*E_{fuel}$

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

SET\_refuel = 1: fill all tanks (including any auxiliary tanks installed)

SET\_refuel = 2: add fuel  $xW_{fuel}$

SET\_refuel = 3: drop fuel  $xW_{fuel}$

SET\_refuel = 4: if below fraction  $rW_{fuel}$  of fuel capacity (including auxiliary tanks), fill all tanks

SET\_refuel = 5: if below fraction  $rW_{fuel}$  of fuel capacity (including auxiliary tanks), add  $xW_{fuel}$

SET\_refuel = 6: if below  $mW_{fuel}$ , fill all tanks

SET\_refuel = 7: if below  $mW_{fuel}$ , add  $xW_{fuel}$

added fuel limited by capacity; not used for first segment

$xW_{fuel}$  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 $\gamma$ (deg)	0.
height_takeoff	real	+	height during climb $h$ (ft or m)	0.
slope_ground	real	+	slope of ground $\gamma_G$ (+ for uphill; deg)	0.
friction	real	+	friction coefficient $\mu$	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

---

parent	int	parent (1 Size, 2 OffDesign)
kMission	int	Mission number
kMissSeg	int	MissSeg number
kcol_out	int	performance output column
		Specification
ikind	int	kind (MissSeg_kind_taxi, dist, time, hold, climb, spiral, fuel, burn)
SET_foldkit	int	folding kit on aircraft (0 none, 1 present)

kind_range	int	this segment receives range credit (0 not, 1 source forward, 2 source backward, 3 both)
fadjust	real	adjustment ratio (initial time or dist ratio)
wassplit	int	split segment (number segments; 0 for not split)
ksplit_first	int	first segment after split
ksplit_last	int	last segment after split
dWpay	real	payload increment ( $\times W_{\text{pay}} - \times W_{\text{pay}}(\text{seg1})$ ) or factor ( $\times W_{\text{pay}} / \times W_{\text{pay}}(\text{seg1})$ )
iSET_maxGW	int	max gross weight (0 no iteration; SET_GW_maxP, maxQ, maxPQ, maxJ, maxPJ, maxQJ, maxPQJ + maxGW)
iSETPmargin(npropmax)	int	power margin as quantity (3 max GW, 2 max effort, 1 trim)
iSETQmargin(npropmax)	int	torque margin as quantity (3 max GW, 2 max effort, 1 trim)
iSETJmargin(njetmax)	int	jet thrust margin as quantity (3 max GW, 2 max effort, 1 trim)
iSETCmargin(nchrgmax)	int	charger power margin as quantity (1 trim)
iSETBmargin(ntankmax)	int	battery power margin as quantity (2 max effort, 1 trim)
<b>Segments</b>		
<b>Maximum gross weight</b>		
ismaxgwconv	int	converged (0 not)
count_maxgw	int	number of iterations
error_maxgw	real	error ratio
GW_inc	real	gross weight increment
<b>Takeoff distance calculation</b>		
iSET_takeoff	int	takeoff segment kind (SET_takeoff_xxx)
VCAS_TO	real	ground speed or climb speed (CAS)
V_TO	real	ground speed (ft/sec or m/sec)
climb_TO	real	angle relative ground (deg)
isConsistent_TO	int	consistent acc and V change, climb and h change
FxG_TO	real	net force $T - D$ (ground axes)
FzG_TO	real	net force $W - L$ (ground axes)
FzGmu_TO	real	friction drag $\mu F_{zG}$
acc_TO	real	acceleration (ground axes)
h_TO	real	height (ft or m)
t_TO	real	time (sec)
s_TO	real	distance (ft or m)
time_TO	real	cumulative time (sec)
dist_TO	real	cumulative distance (ft or m)

		original value for engine failure decision (from FltAircraft)
rating_original(nengmax)	c*12	engine rating
krate_original(nengmax)	int	engine rating
fPower_original(nengmax)	real	fraction of rated engine power available
rating_jet_original(njetmax)	c*12	jet rating
krate_jet_original(njetmax)	int	jet rating
fThrust_original(njetmax)	real	fraction of rated jet thrust available
rating_charge_original(nchrgmax)	c*12	charger rating
krate_charge_original(nchrgmax)	int	charger rating
fCharge_original(nchrgmax)	real	fraction of rated charger power available
friction_original	real	friction coefficient
kSegEF_TO	int	engine failure segment (0 for none)
		Performance (from FltState; at start or midpoint)
speed	real	horizontal speed $V_h$ (knots)
Vclimb	real	climb velocity $V_c$ (ft/sec or m/sec)
fuelflow(ntankmax)	real	fuel flow $\dot{w}$ (lb/hr or kg/hr)
energyflow(ntankmax)	real	energy flow $\dot{E}$ (MJ/hr)
		trapezoidal integration
speed_start	real	horizontal speed $V_h$
Vclimb_start	real	climb velocity $V_c$
fuelflow_start(ntankmax)	real	fuel flow $\dot{w}$
energyflow_start(ntankmax)	real	energy flow $\dot{E}$
speed_end	real	horizontal speed $V_h$
Vclimb_end	real	climb velocity $V_c$
fuelflow_end(ntankmax)	real	fuel flow $\dot{w}$
energyflow_end(ntankmax)	real	energy flow $\dot{E}$
alt_start	real	altitude $h$ at start of segment (ft or m)
alt_end	real	altitude $h$ at end of segment (from start of next segment, only used for kind='climb' or 'spiral')
Wind	real	Headwind $V_w$ (knots)
groundspeed	real	Ground speed ( $V_h - V_w$ ) (knots)

Mission segment quantities		
T	real	time $T$ (minutes)
D	real	ground distance $D$ (nm)
otherDpast	real	distance from past range credit (nm)
otherDfuture	real	distance from future range credit (nm)
dR	real	range contribution $dR$ (nm)
airdist	real	air distance (nm)
Wburn(ntankmax)	real	fuel burned $W_{\text{burn}}$ (lb or kg)
Wfuel_add(ntankmax)	real	fuel added at start of segment
Wfuel_start(ntankmax)	real	fuel weight $W_{\text{fuel}}$ (segment start)
Eburn(ntankmax)	real	fuel burned $E_{\text{burn}}$ (MJ)
Efuel_add(ntankmax)	real	fuel added at start of segment
Efuel_start(ntankmax)	real	fuel energy $E_{\text{fuel}}$ (segment start)
GW_start	real	gross weight $W_G$ (segment start)



## Chapter 19

**Structure: FltState**

Variable	Type	Description	Default
		Flight State	
FltAircraft	FltAircraft	Aircraft	
		Components	
FltFuse	FltFuse	fuselage	
FltGear	FltGear	landing gear	
FltRotor(nrotormax)	FltRotor	rotors	
FltWing(nwingmax)	FltWing	wings	
FltTail(ntailmax)	FltTail	tails	
FltTank(ntankmax)	FltTank	fuel tank systems	
FltProp(npropmax)	FltProp	propulsion groups	
FltEngn(nengmax)	FltEngn	engine groups	
FltJet(njetmax)	FltJet	jet groups	
FltChrg(nchrgmax)	FltChrg	charge groups	

**Structure: FltAircraft**

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
fVel(2)	real	+ flight speed factor	1.
SET_vel	c*12	+ flight speed	'general'
Vkts	real	+ horizontal velocity $V_h$ (TAS or CAS, 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 $\theta_V$ (deg)	0.
side	real	+ sideslip angle $\psi_V$ (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 $\theta_F$	0.
roll	real	+ roll $\phi_F$	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 $\phi_F$ (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 <sup>2</sup> or m/sec <sup>2</sup> )	0.
ay_linear	real	+ y-acceleration $a_{ACy}$ (ft/sec <sup>2</sup> or m/sec <sup>2</sup> )	0.

az_linear	real	+	z-acceleration $a_{ACz}$ (ft/sec <sup>2</sup> or m/sec <sup>2</sup> )	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 $\tau$ (Units_temp)	
dtemp	real	+	temperature increment $\Delta T$ (Units_temp)	0.
density	real	+	density $\rho$	
csound	real	+	speed of sound $c_s$	
viscosity	real	+	viscosity $\mu$	
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(ncntmax)	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(ncntmax)	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 $\alpha_{tilt}$	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	
mu_Vtip(npropmax)	real	+ tip speed from $\mu$	
Mat_Vtip(npropmax)	real	+ tip speed from $M_{at}$	
Nspec(npropmax)	real	+ engine turbine speed (rpm)	
STATE_gear(npropmax)	int	+ drive system state	1
rating_ds(npropmax)	c*12	+ drive system rating	' '
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
STATE_deice(npropmax)	int	+ deice system state (0 off)	0
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
		+ 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 $x_{CCmax}$ )	1.

		+ Performance	
DoQ_pay	real	+ payload forward flight drag increment $D/q$ (Units_drag)	0.
fDoQ_pay	real	+ payload drag increment scaling with weight $\Delta(D/q)/W_{\text{pay}}$ (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 $\kappa$	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)	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)

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

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
climb or descent rate	'climb'	maximum (ROC)
climb rate (power)	'power'	maximum (1/Power)

description	max_quant	
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',	$\min(Q_{limit} - Q_{req}) = 0$ (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'	$\min(P_{max} -  \dot{E}_{batt} ) = 0$ (all fuel tanks)
rotor thrust margin	'rotor(t) n'	$(C_T/\sigma)_{max} - C_T/\sigma = 0$ (transient)
rotor thrust margin	'rotor(s) n'	$(C_T/\sigma)_{max} - C_T/\sigma = 0$ (sustained)
wing lift margin	'stall n'	$C_{Lmax} - C_L = 0$

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	'xaccl', 'yaccl', 'zaccl'	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)
- '+Mach' = use Mach not Vkts
- '+CAS' = Vkts is CAS 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 ( $\theta_V, \psi_V$ )
- 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 ( $\psi_F, \theta_F, \phi_F$ )

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

$\tan(\text{roll}) = \sqrt{n^2 - 1} = (\text{turn})V/g$ ; calculated using input Vkts for flight speed

SET\_pullup, load factor in pullup: use pullup rate or load factor

$n = 1 + (\text{pullup})V/g$ ; 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)

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  
 '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  $\mu$  ( $\mu$ \_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  $V_{tip}$  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  
if Propulsion%rate\_ds ≤ 1, drive system rating not used

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 reaction drive,  $P_{reqEG}$  equals rotor power required  
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,  $T_{reqJG} = F_{react}$  of rotor

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

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'  
 requirement for trim\_target depends on designation of Aircraft%trim\_quant

---

parent	int	parent (1 SizeCond, 2 SizeMiss, 3 OffMiss, 4 PerfCond)
kMission	int	Mission number
kMissSeg	int	MissSeg number
kFltState	int	FltState number
kcol_out	int	performance output column
Maximum effort		
imax_quant(2)	int	quantity (MAX_QUANT_xxx)
imax_quantn(2)	int	quantity structure number
imax_isslope(2)	int	quantity is slope (maximize)
imax_var(2)	int	variable (MAX_VAR_xxx, or control number)
imax_varn(2)	int	variable structure number
Specification		
iSET_vel	int	velocity (SET_vel_xxx)
iSET_vel2	int	velocity (SET_vel2_TAS, SET_vel2_CAS, SET_vel2_Mach)
isSideward	int	sideward flight (1 for sideward flight)
iSET_atmos	int	atmosphere (SET_atmos_xxx)
iSTATE_LG	int	landing gear state (STATE_LG_default, extend, retract)
iSTATE_trim	int	aircraft trim state (number, 0 for no trim)
Specification, each propulsion group		
iSET_Vtip(npropmax)	int	rotor tip speed (SET_Vtip_input, ref, speed, conv, hover, cruise, man, OEI, xmsn, mu, Mat)
iSET_Vtip_Mat(npropmax)	int	rotor tip speed limited by $M_{at}$
iSET_Vtip_VarDiam(npropmax)	int	rotor tip speed for variable diameter rotor (1 to scale $V_{tip}$ with radius ratio)
iSETPmargin(npropmax)	int	power margin as quantity (2 maximum effort, 1 trim)
iSETQmargin(npropmax)	int	torque margin as quantity (2 maximum effort, 1 trim)
krate_ds(npropmax)	int	drive system rating number

		Specification
krate(nengmax)	int	engine rating number
krate_jet(njetmax)	int	jet rating number
krate_charge(nchrgmax)	int	charger rating number
iSETJmargin(njetmax)	int	jet thrust margin as quantity (2 maximum effort, 1 trim)
iSETCmargin(nchrgmax)	int	charger power margin as quantity (1 trim)
iSETBmargin(ntankmax)	int	battery power margin as quantity (2 maximum effort, 1 trim)
		Weight
GW	real	gross weight $W_G$
Wfuel_total	real	usable fuel weight $W_{fuel}$
Wfuel(ntankmax)	real	usable fuel weight
Wfuel_std(ntankmax)	real	standard tanks
Wfuel_aux(ntankmax)	real	auxiliary tanks
Wpayload	real	payload weight $W_{pay}$
Wpay_pass	real	passengers $W_{pass}$
Wpay_cargo	real	cargo $W_{cargo}$
Wpay_extload	real	external load $W_{ext-load}$
Wpay_ammo	real	ammunition $W_{ammo}$
Wpay_weapons	real	weapons $W_{weapons}$
Wpay_other	real	other $W_{other}$
WFixUL	real	fixed useful load $W_{FUL}$
dW_fixUL	real	fixed useful load increment (relative weight statement $W_{fixUL}$ )
Wcrew	real	crew (replace weight statement $W_{fixUL\_crew}$ )
Wauxtank	real	auxiliary fuel tanks (replace weight statement $W_{fixUL\_auxtank}$ )
W_fixUL_other	real	other fixed useful load (replace weight statement $W_{fixUL\_other}$ )
Woful(10)	real	categories
Wequip	real	equipment increment (replace weight statement $W_{fixUL\_equip}$ )
Wfoldkit	real	folding kit (replace weight statement $W_{fixUL\_foldkit}$ )
Wextkit	real	wing extension kit (replace weight statement $W_{fixUL\_extkit}$ )
Wwingkit	real	wing kit (replace weight statement $W_{fixUL\_wingkit}$ )
Wotherkit	real	other kit (replace weight statement $W_{fixUL\_otherkit}$ )
WO	real	operating weight $W_O$
Ncrew	int	number of crew
Npass	int	number of passengers

Ncrew_seat	int	number of crew seats
Npass_seat	int	number of passenger seats
Efuel_total	real	usable fuel energy $E_{\text{fuel}}$
Efuel(ntankmax)	real	usable fuel energy
Efuel_std(ntankmax)	real	standard tanks
Efuel_aux(ntankmax)	real	auxiliary tanks
Weight at mission segment start		
GW_start	real	gross weight $W_G$
Wfuel_start(ntankmax)	real	usable fuel weight $W_{\text{fuel}}$
Wfuel_std_start(ntankmax)	real	standard tanks
Wfuel_aux_start(ntankmax)	real	auxiliary tanks
Efuel_start(ntankmax)	real	usable fuel energy $E_{\text{fuel}}$
Efuel_std_start(ntankmax)	real	standard tanks
Efuel_aux_start(ntankmax)	real	auxiliary tanks
zcg(3)	real	Center of gravity position
SLcg	real	stationline
BLcg	real	buttline
WLCg	real	waterline
Moments of inertia		
lxx	real	$I_{xx}$
lyy	real	$I_{yy}$
lzz	real	$I_{zz}$
lxy	real	$I_{xy}$
lyz	real	$I_{yz}$
lxz	real	$I_{xz}$

---

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

gross weight = weight empty + useful load = operating weight + payload + usable fuel

useful load = fixed useful load + payload + usable fuel

operating weight = weight empty + fixed useful load

---

		Atmosphere
alt	real	altitude $h$
tmp	real	temperature $\tau$
dtmp	real	temperature increment $\Delta T$
sigma	real	density ratio $\rho/\rho_0$
theta	real	temperature ratio $T/T_0$
delta	real	pressure ratio $p/p_0$
kinvis	real	kinematic viscosity $\nu = \mu/\rho$
altdens	real	density altitude $h_d$
altpress	real	pressure altitude $h_p$
		Flight condition
radius(nrotormax)	real	rotor radius $R$
		rotational speeds
Vtip_trim(nrotormax)	real	rotor tip speed $\Omega R$
rpm_trim(nrotormax)	real	rotor rpm $\Omega$
rN_trim_rotor(nrotormax)	real	rotor $\Omega/\Omega_{\text{ref}}$
N_trim(nengmax)	real	engine turbine rpm $N$
rN_trim_eng(nengmax)	real	engine $N/N_{\text{spec}}$
rN_trim_ref(npropmax)	real	propulsion group reference speed ratio
		flight speed
speed	real	horizontal speed $V_h$ (knots)
Vclimb	real	climb velocity $V_c$ (ft/sec or m/sec)
side_trim	real	sideslip angle $\psi_V$ (deg)
		derived
Vhoriz	real	horizontal velocity $V_h$ (ft/sec or m/sec)
climb_trim	real	climb angle $\theta_V$ (deg)
Vside	real	sideward velocity $V_s$ (ft/sec or m/sec)
Vmag	real	velocity magnitude $ V $
Vfwd	real	forward velocity $V_f$ (ft/sec or m/sec)
VCAS	real	calibrated airspeed $V_{\text{cal}}$ (knots) ( $V\sqrt{\sigma}f(\delta, M)$ )
VAC(3)	real	$v_{AC}$ in F axes
ed(3)	real	drag vector, $-v_{AC}/ v_{AC} $ in F axes
qAC	real	$q_{AC}$

		angular velocity
turn_trim	real	turn $\dot{\psi}_F$ (yaw rate)
pullup_trim	real	pullup $\dot{\theta}_F$ (pitch rate)
turnRadius	real	turn radius $R_T$
wAC(3)	real	$\omega_{AC}$ in F axes
		acceleration
aAC(3)	real	$a_{AC}$ in F axes (linear)
nAC(3)	real	load factor $n_{AC}$ (linear acc and angular rate)
KIND_alpha	int	angle of attack and sideslip angle representation (1 conventional, 2 reversed)
		orientation of body axes relative inertial axes
pitch_trim	real	pitch angle $\theta_F$ (deg)
roll_trim	real	roll angle $\phi_F$ (deg)
		rotation matrices
CFI(3,3)	real	$C^{FI}$ , velocity axes relative inertial axes
CVI(3,3)	real	$C^{VI}$ , body axes relative inertial axes
CFV(3,3)	real	$C^{FV}$ , body axes relative velocity axes
control_trim(ncntmax)	real	aircraft controls
Nauxtank(nauxtankmax,ntankmax)	int	number of auxiliary fuel tanks $N_{auxtank}$ (each aux tank size), from FltCond or MissSeg
SET_extkit(nwingmax)	int	wing extension kit on aircraft (0 none, 1 present)
SET_wingkit(nwingmax)	int	wing kit on aircraft (0 none, 1 present)
Wfuel_cap(ntankmax)	real	total fuel capacity $W_{fuel-cap}$ , including auxiliary tanks
Efuel_cap(ntankmax)	real	total fuel capacity $E_{fuel-cap}$ , including auxiliary tanks
slope_ground	real	slope of ground $\gamma_G$ (+uphill; deg), from MissSeg
SET_sweep	int	parameter sweep, from FltCond

---

angle of attack and sideslip angle representation: from Aircraft and isSideward

orientation body relative inertial axes defined by Euler angles, with yaw/pitch/roll sequence  $(\psi_F, \theta_F, \phi_F)$

yaw positive to right, pitch positive nose up, roll positive to right

$$C^{FI} = X_{roll} Y_{pitch} Z_{yaw}, \text{ yaw angle} = (\text{turn}) \cdot \text{time}$$

orientation velocity relative inertial axes defined by climb and sideslip angles  $(\theta_V, \psi_V)$

sideslip positive aircraft moving to right, climb positive aircraft moving up

$$C^{VI} = Y_{climb} Z_{side} Z_{yaw}$$

orientation body relative velocity axes:  $C^{FV} = X_{\text{roll}} Y_{\text{pitch}} Z_{\text{-side}} Y_{\text{-climb}}$

---

		Trim (last)
istrimconv	int	converged (0 not)
count_trim	int	number of iterations
error_trim(mtrimmax)	real	error ratio
gain_trim(mtrimmax,mtrimmax)	real	gain matrix
		Maximum effort (principal iteration, 99% range iteration; inner, outer loops)
isflyconv(2,2)	int	converged (0 not)
count_fly(2,2)	int	number of iterations
error_fly(2,2)	real	error ratio
isSwitched(2)	int	quantity switched (1 P margin, 2 Q margin, 3 both)
		Maximum gross weight (flight condition or mission takeoff)
ismaxgwconv	int	converged (0 not)
count_maxgw	int	number of iterations
error_maxgw	real	error ratio
		Rotor flap equation (all converged or any not converged)
isrotorconv	int	converged (0 not, -1 no iteration)
		Solution state
count_control	int	count of solution (0 at start, get aircraft controls)
trim_deriv_exist	int	trim derivative matrix exist (0 for not)
		Loads
		forces (F axes, about cg)
Faero(3)	real	aerodynamic $F_{\text{aero}}^F$ (fuselage, rotor, wing, tail, tank, engine, jet, charger)
Frotor(3)	real	rotor $F_{\text{rotor}}^F$
Fengine(3)	real	engine groups $F_{\text{eng}}^F$ (jet thrust, momentum drag)
Fjet(3)	real	jet groups $F_{\text{jet}}^F$
Fchrg(3)	real	charge groups $F_{\text{charge}}^F$
Fgrav(3)	real	gravitational $F_{\text{grav}}^F$
Finertia(3)	real	inertial $F_{\text{inertial}}^F$ (turn)



		moments (F axes, about cg)
Maero(3)	real	aerodynamic $M_{aero}^F$ (fuselage, rotor, wing, tail, tank, engine, jet, charger)
Mrotor(3)	real	rotor $M_{rotor}^F$
Mengine(3)	real	engine groups $M_{engine}^F$ (jet thrust, momentum drag)
Mjet(3)	real	jet groups $M_{jet}^F$
Mchrg(3)	real	charge groups $M_{charge}^F$
Minertia(3)	real	inertial $M_{inertia}^F$ (turn)
Ftotal(3)	real	total force (F axes, about cg); $F + F_{grav} - F_{inertia}$
Mtotal(3)	real	total moment (F axes, about cg); $M - M_{inertia}$
Download	real	download, aero $F_z$ (I axes); set to 0 if $V > 10$ knots
Thrust	real	rotor thrust, rotor $-F_z$ (I axes; sum Fvert)
DLoT	real	$DL/T$
DLoW	real	$DL/W$
		Aircraft performance
		power
Preq	real	power required $P_{req}$ (engine groups)
Pmargin	real	power margin, $\min(P_{av} - P_{req})$ (propulsion groups and converted engine groups)
Qmargin	real	torque margin, $\min(P_{limit} - P_{req})$
exceedP	int	exceed power available: any propulsion group $P_{reqPG} > (1 + \epsilon)P_{avPG}$
exceedQ	int	exceed torque available: any propulsion group $P_{reqPG} > (1 + \epsilon)P_{DSlimit}$
		thrust
Tjet	real	thrust required $T_{jet}$ (jet groups)
Jmargin	real	jet thrust margin, $\min(T_{av} - T_{req})$
exceedJ	int	exceed jet thrust available: any jet group $T_{reqJG} > (1 + \epsilon)T_{avJG}$
		charging
Pchrg	real	power required $P_{chrg}$ (charge groups)
Cmargin	real	charger power margin, $\min(P_{av} - P_{req})$
exceedC	int	exceed charger power available: any charge group $P_{reqCG} > (1 + \epsilon)P_{avCG}$
Pequiv	real	equivalent aircraft power required $P = P_{req} + VT_{jet}$
Pclimb	real	climb power, $V_{climb}W$
fuelflow(ntankmax)	real	fuel flow $\dot{w}$
fuelflow_total	real	total fuel flow $\dot{w}$
fuelflow_equiv	real	equivalent fuel flow $\dot{w}_{equiv}$ , from energy flow
energyflow(ntankmax)	real	energy flow $\dot{E}$

energyflow_total	real	total energy flow $\dot{E}$
exceedWf	int	exceed fuel capacity: $W_{\text{fuel}} > (1 + \epsilon)W_{\text{fuel-cap}}$ or $E_{\text{fuel}} > (1 + \epsilon)E_{\text{fuel-cap}}$ battery
Bmargin	real	battery power margin, $\min(P_{\text{max}} -  \dot{E}_{\text{batt}} )$ (MJ/hr)
exceedB	int	exceed battery power: any fuel tank $ \dot{E}_{\text{batt}}  > (1 + \epsilon)P_{\text{max}}$
sfc	real	sfc, $P_{\text{equiv}}/\dot{w}_{\text{equiv}}$
spec_range	real	specific range, $V/\dot{w}_{\text{equiv}}$
Performance indices		
FM	real	figure of merit $FM = W\sqrt{W/(2\rho A_{\text{ref}})}/P$
LoDe	real	$L/D_e = WV/P$
Drage	real	effective drag $D_e = P/V$
DragAC	real	aircraft drag $D_{AC}$
DoQAC	real	$D/q = D_{AC}/q_{AC}$ ; set to 0 if $V < 10$ knots
WoP	real	power loading $W/P$
range_onepcW	real	range for fuel=1%GW (nm)
fuel_eff	real	fuel efficiency $e = W_{\text{pay}}V/\dot{w}_{\text{equiv}}$ (ton-nm/lb or ton-nm/kg)
productivity	real	productivity $p = W_{\text{pay}}V/W_O$ (ton-kt/lb or ton-kt/kg)
Operating size		
length_op	real	length
width_op	real	width
area_op	real	area

**Structure: FltFuse**

Variable	Type	Description	Default
Flight State - Fuselage			
aerodynamics			
VintR(3,nrotormax)	real	interference velocity $v_{int}^F$ , from rotors (F axes)	
Vaero(3)	real	total velocity relative air $v^F$ (F axes)	
VB(3)	real	total velocity relative air $v^B$ (B axes)	
alpha	real	angle of attack $\alpha$ (deg)	
beta	real	sideslip angle $\beta$ (deg)	
CBA(3,3)	real	$C^{BA}$	
Vmag	real	velocity magnitude	
q	real	dynamic pressure	
DoQ_pay	real	payload $D/q$	
DoQ_cont	real	contingency $D/q$	
CL	real	lift coefficient $C_L$	
CM	real	pitch moment coefficient $C_M$	
CD	real	drag coefficient $C_D$	
CY	real	side force coefficient $C_Y$	
CN	real	yaw moment coefficient $C_N$	
L	real	lift	
M	real	pitch moment	
D	real	drag	
Y	real	side force	
N	real	yaw moment	
loads			
Faero(3)	real	aerodynamic force $F_{aero}^F$ (F axes, about cg)	
Maero(3)	real	aerodynamic moment $M_{aero}^F$ (F axes, about cg)	
Drag	real	drag $e_d^T F_{aero}^F$	
Download	real	download, aero $F_z$ (I axes)	

**Structure: FltGear**

Variable	Type	Description	Default
		Flight State - Landing Gear	
		aerodynamics	
iSTATE_LG	int	landing gear state (STATE_LG_extended, retracted)	
Vaero(3)	real	total velocity relative air $v^F$ (F axes)	
Vmag	real	velocity magnitude	
q	real	dynamic pressure	
ed(3)	real	drag vector, $-v/ v $ in F axes	
Faero(3)	real	aerodynamic force $F_{aero}^F$ (F axes, about cg)	
Maero(3)	real	aerodynamic moment $M_{aero}^F$ (F axes, about cg)	
Drag	real	drag $e_d^T F_{aero}^F$	
Download	real	download, aero $F_z$ (I axes)	

**Structure: FltRotor**

Variable	Type	Description	Default
		Flight State - Rotor	
		control mode	
KIND_control_coll	int	collective control mode (1 thrust command, 2 pitch command)	
KIND_control_cyc	int	cyclic control mode (1 TPP command, 2 NFP command)	
Scoll	real	collective $T$ matrix scale factor $S$ (1, $a/6$ , $\rho V_{\text{tip}}^2 A_{\text{blade}} a/6$ )	
Scyc	real	cyclic $T$ matrix scale factor $S$ (-1 TPP command, 1 NFP command)	
		controls	
coll	real	collective	
lngcyc	real	longitudinal cyclic	
latcyc	real	lateral cyclic	
incid	real	incidence	
cant	real	cant	
diam	real	diameter	
fgear	real	gear ratio factor	
		geometry	
Ccont(3,3)	real	shaft control, $C_{\text{cont}}$	
CSF(3,3)	real	shaft relative airframe, $C^{SF}$	
zhub(3)	real	hub position, $z_{\text{hub}}$	
zpylon(3)	real	pylon position, $z_{\text{pylon}}$	
znac(3)	real	nacelle cg position, $z_{\text{nac}}$	
CBF(3,3)	real	pylon relative airframe, $C^{BF}$	
		condition	
radius	real	radius $R$	
Vtip	real	tip speed $V_{\text{tip}} = \Omega R$	
Omega	real	rotational speed $\Omega$	
Mtip	real	tip Mach number $M_{\text{tip}}$	
Mat	real	maximum Mach number $M_{at}$ (advancing tip or helical)	
sigma	real	solidity $\sigma$ (thrust weighted)	

gamma	real	Lock number $\gamma$
lblade	real	blade moment of inertia $I_{\text{blade}}$
flapfreq	real	flap frequency $\nu$
conefreq	real	coning frequency $\nu$
Khub	real	hub stiffness $K_{\text{hub}}$
		performance
		shaft axis loads
T	real	thrust
H	real	drag force
Y	real	side force
Mx	real	roll moment
My	real	pitch moment
Q	real	torque
CT	real	thrust coefficient $C_T$
CH	real	drag force coefficient $C_H$
CY	real	side force coefficient $C_Y$
CMx	real	roll moment coefficient $C_{Mx}$
CMy	real	pitch moment coefficient $C_{My}$
CQ	real	torque coefficient $C_Q$
		control and motion
theta75	real	collective pitch $\theta_{0.75}$ (0.75R)
thetas	real	longitudinal cyclic pitch $\theta_s$
thetac	real	lateral cyclic pitch $\theta_c$
beta0	real	coning $\beta_0$
betac	real	longitudinal flapping $\beta_c$
betas	real	lateral flapping $\beta_s$
lambda0	real	inflow $\lambda_0 = \kappa \lambda_i$
CPS(3,3)	real	tip-path plane relative shaft, $C^{PS}$
		velocity and inflow
VoVtip	real	$V/V_{\text{tip}}$
VF(3)	real	total velocity relative air $v^F$ (F axes)
VS(3)	real	total velocity relative air $v^S$ (S axes)
mux	real	$\mu_x$
muy	real	$\mu_y$

muz	real	$\mu_z$
omegaS(3)	real	angular velocity $\omega^S$ (S axes)
dax	real	$\dot{\alpha}_x$
day	real	$\dot{\alpha}_y$
mu	real	$\mu = \sqrt{\mu_x^2 + \mu_y^2}$
alphas	real	$\alpha = \tan^{-1}(\mu_z/\mu)$
fDuctA	real	ducted fan area ratio $f_A$
fDuctT	real	ducted fan thrust ratio $f_T$
fDuctW	real	ducted fan far wake ratio $f_W$
zg	real	height rotor hub above ground, $z_g/D$
zge	real	effective height, $z_g C_g / (D \cos \epsilon)$
fg	real	ground effect inflow ratio $f_g$
CTe	real	$C_T$ for inflow solution
lambdah	real	reference $\lambda_h = \sqrt{C_T/2}$
lambda_ideal	real	ideal induced velocity $\lambda_i$
CPideal	real	ideal induced power $C_{Pideal} = C_T \lambda_i$
kappax	real	inflow gradient $\kappa_x$
kappay	real	inflow gradient $\kappa_y$
kappam	real	inflow gradient $\kappa_m = (\sigma a/8) f_m / U$
CTs	real	thrust coefficient/solidity, $ C_T/\sigma $
FPpro	real	profile power factor $F_P$
FHpro	real	profile drag factor $F_H$
VintWn(3,nwingmax)	real	interference velocity $v_{int}^F$ from wings, normal (F axes)
VintWp(3,nwingmax)	real	interference velocity $v_{int}^F$ from wings, inplane (F axes)
		inplane forces
CHtpp	real	drag force $C_H$ , tpp
CYtpp	real	side force $C_Y$ , tpp
CHo	real	drag force $C_H$ , profile
CYo	real	side force $C_Y$ , profile
		rotor flap equations
isrotorconv	int	converged (0 not, -1 no iteration)
count_rotor	int	iteration count
error_rotor(3)	real	error ratio ( $E_t, E_c, E_s$ )
rotor_deriv_exist	int	rotor derivative matrix exist (0 for not)

		loads
Frotor(3)	real	rotor force $F_{rotor}^F$ (F axes, about cg)
Mrotor(3)	real	rotor moment $M_{rotor}^F$ (F axes, about cg)
L	real	lift (wind axis)
X	real	drag (wind axis)
CL	real	lift coefficient $C_L$
CX	real	drag coefficient $C_X$
Fvert	real	vertical force (inertia axes)
CTs_steady	real	max $C_T/\sigma$ (sustained)
CTs_tran	real	max $C_T/\sigma$ (transient)
Tmargin_steady	real	thrust margin, $(C_T/\sigma)_{max} - C_T/\sigma$ (sustained)
Tmargin_tran	real	thrust margin, $(C_T/\sigma)_{max} - C_T/\sigma$ (transient)
Plimit_rs	real	drive system limit $P_{RSlimit}$ (at rpm_trim and rating_ds)
Qmargin_rs	real	torque margin, $P_{RSlimit} - P$
exceedQ_rs	int	exceed torque available: $P > (1 + \epsilon)P_{RSlimit}$
		power
P	real	rotor power $P$
Pind	real	induced power $P_i$
Pint	real	interference power $P_t$
Ppro	real	profile power $P_o$
Ppar	real	parasite power $P_p$
CP	real	rotor power coefficient $C_P$
CPind	real	induced power coefficient $C_{P_i}$
CPint	real	interference power coefficient $C_{P_t}$
CPpro	real	profile power coefficient $C_{P_o}$
CPpar	real	parasite power coefficient $C_{P_p}$
lambda	real	induced velocity $\lambda$
lambdat	real	interference velocity $\lambda_t = C_{P_t}/C_F$
Ki	real	induced power factor $\kappa$
cdmean	real	mean drag coefficient $c_{dmean}$
FM	real	hover figure of merit, $Tf_Dv/P$
etaprop	real	propulsive efficiency, $TV/P$
etamom	real	momentum efficiency, $T(V + f_Dv)/P$
CDe	real	effective drag, $(C_{P_i} + C_{P_o})/(V/V_{tip})$



LoDe	real	effective lift-to-drag, $C_L/C_{De}$
Pshaft	real	shaft power $P_{\text{shaft}}$
Preact	real	reaction drive power $P_{\text{react}}$
Freact	real	reaction drive net force $F_{\text{react}} = P_{\text{react}}/\Omega r_{\text{react}}$
rOmegareact	real	blade velocity $\Omega r_{\text{react}}$
aerodynamics		
hub		
Vaero_hub(3)	real	total velocity relative air $v^F$ (F axes)
Vmag_hub	real	velocity magnitude
q_hub	real	dynamic pressure
ed_hub(3)	real	drag vector, $-v/ v $ in F axes
VB_hub(3)	real	total velocity relative air $v^B$ (B axes)
alpha_hub	real	angle of attack $\alpha$ (deg)
pylon		
Vaero_pylon(3)	real	total velocity relative air $v^F$ (F axes)
Vmag_pylon	real	velocity magnitude
q_pylon	real	dynamic pressure
ed_pylon(3)	real	drag vector, $-v/ v $ in F axes
VB_pylon(3)	real	total velocity relative air $v^B$ (B axes)
alpha_pylon	real	angle of attack $\alpha$ (deg)
CDhub	real	drag coefficient, hub $C_{D\text{hub}}$
CDpylon	real	drag coefficient, pylon $C_{D\text{pylon}}$
CDduct	real	drag coefficient, duct $C_{D\text{duct}}$
Dhub	real	drag, hub $D_{\text{hub}}$
Dpylon	real	drag, pylon $D_{\text{pylon}}$
Dduct	real	drag, duct $D_{\text{duct}}$
Dspin	real	drag, spinner $D_{\text{spin}}$
loads		
Faero(3)	real	aerodynamic force $F_{\text{aero}}^F$ (F axes, about cg)
Maero(3)	real	aerodynamic moment $M_{\text{aero}}^F$ (F axes, about cg)
Drag	real	drag $e_d^T F_{\text{aero}}^F$
Download	real	download, aero $F_z$ (I axes)

lambda_int	real	interference	ideal induced velocity $\lambda_i$ (from $C_T$ )
vind(3)	real		induced velocity $v_{ind}^F$ (F axes)
chi_wake	real		wake angle $\chi$
Fint_fus	real		interference factor $f_W f_z f_r f_t$ at fuselage
Fint_wingLp(nwingmax, npanelmax)	real		interference factor $f_W f_z f_r f_t$ at wing, left panel
Fint_wingRp(nwingmax, npanelmax)	real		interference factor $f_W f_z f_r f_t$ at wing, right panel
Fint_tail(ntailmax)	real		interference factor $f_W f_z f_r f_t$ at tail
isInWake_fus	int		fuselage inside wake
isInWake_wingLp(nwingmax, npanelmax)	int		wing inside wake, left panel
isInWake_wingRp(nwingmax, npanelmax)	int		wing inside wake, right panel
isInWake_tail(ntailmax)	int		tail inside wake
ftwin	real		twin rotor factor $f_t$
Aint_wing(nwingmax)	real		induced power interference at wing $\alpha_{int}$

## Chapter 24

**Structure: FltWing**

Variable	Type	Description	Default
		Flight State - Wing controls	
flap(npanelmax)	real	flap $\delta_F$	
flaperon(npanelmax)	real	flaperon $\delta_f$	
aileron(npanelmax)	real	aileron $\delta_a$	
incid(npanelmax)	real	incidence $i$	
		aerodynamics	
VintR_Lp(3,nrotormax,npanelmax)	real	interference velocity $v_{int}^F$ at left wing panel, from rotors (F axes)	
VintR_Rp(3,nrotormax,npanelmax)	real	interference velocity $v_{int}^F$ at right wing panel, from rotors (F axes)	
VintR(3,nrotormax)	real	interference velocity $v_{int}^F$ (panel area weighted), from rotors (F axes)	
VintW(3,nwingmax)	real	interference velocity $v_{int}^F$ , from other wings (F axes)	
AintW(nwingmax)	real	interference angle $\alpha_{int}$ , from other wings	
AintR(nrotormax)	real	induced power interference $\alpha_{int}$ , from rotors	
		with mean interference	
Vaero(3)	real	total velocity relative air $v^F$ (F axes)	
VB(3)	real	total velocity relative air $v^B$ (B axes)	
alpha	real	angle of attack $\alpha$ (deg)	
beta	real	sideslip angle $\beta$ (deg)	
CBA(3,3)	real	$C^{BA}$	
Vmag	real	velocity magnitude	
q	real	dynamic pressure	
alpha_int	real	angle of attack $\alpha$ , with interference (deg)	
CDV	real	vertical drag coefficient $C_{DV}$	
		left panel	
Vaero_Lp(3,npanelmax)	real	total velocity relative air $v^F$ (F axes)	

VB_Lp(3,npanelmax)	real	total velocity relative air $v^B$ (B axes)
alpha_Lp(npanelmax)	real	angle of attack $\alpha$ (deg)
beta_Lp(npanelmax)	real	sideslip angle $\beta$ (deg)
CBA_Lp(3,3,npanelmax)	real	$C^{BA}$
Vmag_Lp(npanelmax)	real	velocity magnitude
q_Lp(npanelmax)	real	dynamic pressure
CL_Lp(npanelmax)	real	lift coefficient $C_{Lp}$
CDp_Lp(npanelmax)	real	drag coefficient, parasite $C_{Dpp}$
CM_Lp(npanelmax)	real	pitch moment coefficient $C_{Mp}$
CR_Lp(npanelmax)	real	roll moment coefficient $C_{lp}$
L_Lp(npanelmax)	real	lift
Dp_Lp(npanelmax)	real	drag, parasite
M_Lp(npanelmax)	real	pitch moment
R_Lp(npanelmax)	real	roll moment
		right panel
Vaero_Rp(3,npanelmax)	real	total velocity relative air $v^F$ (F axes)
VB_Rp(3,npanelmax)	real	total velocity relative air $v^B$ (B axes)
alpha_Rp(npanelmax)	real	angle of attack $\alpha$ (deg)
beta_Rp(npanelmax)	real	sideslip angle $\beta$ (deg)
CBA_Rp(3,3,npanelmax)	real	$C^{BA}$
Vmag_Rp(npanelmax)	real	velocity magnitude
q_Rp(npanelmax)	real	dynamic pressure
CL_Rp(npanelmax)	real	lift coefficient $C_{Lp}$
CDp_Rp(npanelmax)	real	drag coefficient, parasite $C_{Dpp}$
CM_Rp(npanelmax)	real	pitch moment coefficient $C_{Mp}$
CR_Rp(npanelmax)	real	roll moment coefficient $C_{lp}$
L_Rp(npanelmax)	real	lift
Dp_Rp(npanelmax)	real	drag, parasite
M_Rp(npanelmax)	real	pitch moment
R_Rp(npanelmax)	real	roll moment
qS	real	$qS$ (sum over panels)
qeff	real	$(qS)/S$ (weighted by panel area)
CL	real	lift coefficient $C_L$
CDp	real	drag coefficient, parasite $C_{Dp}$

CDi	real	drag coefficient, induced $C_{Di}$
CM	real	pitch moment coefficient $C_M$
CR	real	roll moment coefficient $C_\ell$
CLmax	real	maximum lift coefficient $C_{Lmax}$
L	real	lift
Dp	real	drag, parasite
Di	real	drag, induced
D	real	drag
M	real	pitch moment
R	real	roll moment
Lmargin	real	stall margin, $C_{Lmax} - C_L$
loads		
Faero(3)	real	aerodynamic force $F_{aero}^F$ (F axes, about cg)
Maero(3)	real	aerodynamic moment $M_{aero}^F$ (F axes, about cg)
Drag	real	drag $e_d^T F_{aero}^F$
Download	real	download, aero $F_z$ (I axes)
interference		
Vint_tail(3,ntailmax)	real	velocity at tail $v_{int}^F$ (F axes)
vind(3)	real	induced velocity $v_{ind}^F$ (F axes)
Vint_wing(3,nwingmax)	real	velocity at other wing $v_{int}^F$ (F axes)
Aint_wing(nwingmax)	real	angle at other wing ( $\alpha_{int} = v_{int}/v^B = K_{int}v_{ind}/v^B$ )
Vintn_rotor(3,nrotormax)	real	velocity at rotor $v_{int}^F$ , normal (F axes)
Vintp_rotor(3,nrotormax)	real	velocity at rotor $v_{int}^F$ , inplane (F axes)

## Chapter 25

**Structure: FltTail**

Variable	Type	Description	Default
		Flight State - Tail	
		controls	
cont	real	control $\delta$	
incid	real	incidence $i$	
		aerodynamics	
VintR(3,nrotormax)	real	interference velocity $v_{int}^F$ , from rotors (F axes)	
VintW(3,nwingmax)	real	interference velocity $v_{int}^F$ , from wings (W axes)	
Vaero(3)	real	total velocity relative air $v^F$ (F axes)	
VB(3)	real	total velocity relative air $v^B$ (B axes)	
alpha	real	angle of attack $\alpha$ (deg)	
beta	real	sideslip angle $\beta$ (deg)	
CBA(3,3)	real	$C^{BA}$	
Vmag	real	velocity magnitude	
q	real	dynamic pressure	
CL	real	lift coefficient $C_L$	
CDp	real	drag coefficient, parasite $C_{Dp}$	
CDi	real	drag coefficient, induced $C_{Di}$	
CLmax	real	maximum lift coefficient $C_{Lmax}$	
L	real	lift	
D	real	drag	
		loads	
Faero(3)	real	aerodynamic force $F_{aero}^F$ (F axes, about cg)	
Maero(3)	real	aerodynamic moment $M_{aero}^F$ (F axes, about cg)	
Drag	real	drag $e_d^T F_{aero}^F$	
Download	real	download, aero $F_z$ (I axes)	

**Structure: FltTank**

Variable	Type	Description	Default
		Flight State - Fuel Tank Systems	
		all tanks (standard plus auxiliary)	
Wfuel	real	usable fuel weight	
Efuel	real	usable fuel energy	
Wfuel_cap	real	fuel weight capacity	
Efuel_cap	real	fuel energy capacity	
rWfuel	real	fraction weight capacity	
rEfuel	real	fraction energy capacity = state-of-charge = 1 - depth-of-discharge	
		battery ( $\dot{E} > 0$ discharge, $\dot{E} < 0$ charge; power and current positive)	
Pfuel_cap	real	power capacity $P_{\text{cap}} = x_{\text{mbd}} E_{\text{fuel-cap}}$ (MJ/hr)	
state	int	state (1 discharging, -1 CC charge, -2 CV charge)	
x	real	current $x$ (1/hr)	
xi	real	current $\xi = x/x_{\text{mbd}}$	
V	real	voltage $V$	
Edotcomp	real	component energy flow $\dot{E}_{\text{comp}}$ (MJ/hr)	
etabatt	real	battery efficiency $\eta_{\text{batt}}$	
Ploss	real	power loss $P_{\text{loss}}$ (MJ/hr)	
Edotbatt	real	battery energy flow $\dot{E}_{\text{batt}}$ (MJ/hr)	
dcrit	real	effective capacity factor $d_{\text{crit}}$	
Edoteff	real	effective energy flow $\dot{E}_{\text{eff}}$ (MJ/hr)	
xmax	real	maximum current $x_{\text{max}}$ (1/hr)	
Pmax	real	maximum power (for $x_{\text{max}}$ ) (MJ/hr)	
Bmargin	real	battery power margin $P_{\text{max}} -  \dot{E}_{\text{batt}} $ (MJ/hr)	
exceedB	int	exceed battery power: $ \dot{E}_{\text{batt}}  > (1 + \epsilon)P_{\text{max}}$	
		auxiliary tanks	
		aerodynamics	
Vaero(3,nauxtankmax)	real	total velocity relative air $v^F$ (F axes)	

Structure: FltTank

Vmag(nauxtankmax)	real	velocity magnitude
q(nauxtankmax)	real	dynamic pressure
ed(3,nauxtankmax)	real	drag vector, $-v/ v $ in F axes
D(nauxtankmax)	real	drag $D$
DL(nauxtankmax)	real	download, aero $F_z$ (I axes)
		loads
Faero(3)	real	aerodynamic force $F_{\text{aero}}^F$ (F axes, about cg)
Maero(3)	real	aerodynamic moment $M_{\text{aero}}^F$ (F axes, about cg)
Drag	real	drag $e_d^T F_{\text{aero}}^F$
Download	real	download, aero $F_z$ (I axes)



**Structure: FltProp**

Variable	Type	Description	Default
STATE_gear	int	Flight State - Propulsion Group drive system state control	
DN_trim	real	rotational speed increment, primary rotor or primary engine (rpm) power	
Pcomp	real	power required $P_{comp}$ , all components	
Pcomp_rotor	real	rotor	
Pcomp_eng	real	engine groups	
Pxmsn	real	transmission losses $P_{xmsn}$	
Pacc	real	accessory power $P_{acc}$	
PreqPG	real	power required $P_{reqPG} = P_{comp} + P_{xmsn} + P_{acc}$ , propulsion group	
PavPG	real	power available $P_{avPG}$ , propulsion group (sum all engine groups producing shaft power)	
PavElsum	real	engine installed power available $P_{avEI}$ (sum all engine groups producing shaft power)	
PavEGsum	real	engine group power available $P_{avEG}$ (sum all engine groups producing shaft power)	
Pratio	real	$P_{reqPG}/P_{avPG}$ , propulsion group	
Plimit_ds	real	drive system limit $P_{DSLlimit}$ (at rpm_trim(primary) and rating_ds)	
atPlimit_ds	int	at drive system limit ( $P_{avPG}$ limited by $P_{DSLlimit}$ )	
Qmargin_ds	real	torque margin, $P_{DSLlimit} - P_{reqPG}$	
Pmargin	real	power margin, $P_{avPG} - P_{reqPG}$	
exceedP	int	exceed power available: $P_{reqPG} > (1 + \epsilon)P_{avPG}$	
exceedQ_ds	int	exceed torque available: $P_{reqPG} > (1 + \epsilon)P_{DSLlimit}$	
Qmargin	real	torque margin, min(propulsion group, engine groups, rotors)	
exceedQ	int	exceed torque available: any propulsion group, engine groups, rotors propulsion group engines	
fuelflow(ntankmax)	real	fuel flow $\dot{w}$	
fuelflow_total	real	total fuel flow $\dot{w}$	
fuelflow_equiv	real	equivalent fuel flow $\dot{w}_{equiv}$ , from energy flow	
energyflow(ntankmax)	real	energy flow $\dot{E}$	

energyflow_total	real	total energy flow $\dot{E}$
sfc	real	specific fuel consumption $sfc = \dot{w}_{equiv}/P_{req}$
Fprop(3)	real	jet thrust and momentum drag force $F_{prop}^F$ (F axes, about cg)
Mprop(3)	real	jet thrust and momentum drag moment $M_{prop}^F$ (F axes, about cg)
Faero(3)	real	aerodynamic force $F_{aero}^F$ (F axes, about cg)
Maero(3)	real	aerodynamic moment $M_{aero}^F$ (F axes, about cg)
Drag	real	drag $e_d^T F_{aero}^F$
Download	real	download, aero $F_z$ (I axes)

**Structure: FltEngn**

Variable	Type	Description	Default
		Flight State - Engine Group	
		controls	
amp	real	amplitude $A$	
mode	real	mode $B$	
incid	real	incidence $i$	
yaw	real	yaw $\psi$	
fgear	real	gear ratio factor $f_{\text{gear}}$	
		geometry	
CBF(3,3)	real	engine relative airframe, $C^{BF}$	
ef(3)	real	engine direction, $e_f$	
		engine	
Pq	real	uninstalled power required, $P_q$	
Plossq	real	installation loss $P_{\text{loss}}$	
etalossq	real	installation efficiency $\eta_{\text{loss}}$	
Preq_eng	real	installed power required, $P_{\text{req-eng}}$	
N_trim	real	engine turbine rpm $N$	
mdot	real	mass flow $\dot{m}$	
wdot	real	fuel flow $\dot{w}$	
Edot	real	energy flow $\dot{E}$	
FG	real	gross installed jet thrust $F_G$	
Fmom	real	momentum thrust $F_{\text{mom}}$	
FN	real	net installed jet thrust $F_N$	
Daux	real	momentum drag of auxiliary air flow $D_{\text{aux}}$	
Pa	real	uninstalled power available, $P_a$	
Plossa	real	installation loss $P_{\text{loss}}$	
etalossa	real	installation efficiency $\eta_{\text{loss}}$	
Pav_eng	real	installed power available, $P_{\text{av-eng}}$	

Pmech	real	engine mechanical limit $P_{\text{mech}}$ (at $N_{\text{trim}}$ )
atPmech	int	at mechanical limit ( $P_{\text{av-eng}}$ limited by $P_{\text{mech}}$ )
etamotor	real	motor/generator efficiency $\eta_{\text{motor}}$
etacell	real	fuel cell efficiency $\eta_{\text{cell}}$
		engine group
ReactionMode	int	reaction drive mode (MODEL_engine_compreact or converted)
Converted	int	converted (KIND=RPTEM with mode=1; 0 shaft power, 1 reaction, 2 jet)
ProducePower	int	shaft power (0 consumed (generator or compressor), 1 produced)
Pcomp	real	component power $P_{\text{comp}}$ (generator or compressor); (Nengine-NEngInOp) $P_q K_{ffd}$
Preq	real	power required $P_{\text{reqEG}}$
PavEI	real	engine installed power available $P_{\text{avEI}}$ ; (Nengine-NEngInOp) $P_{\text{av-eng}}$
Pav	real	power available, $P_{\text{avEG}}$ ; fPower(Nengine-NEngInOp) $P_{\text{av-eng}}$
Qreq	real	torque required $Q_{\text{req}}$ (at $N_{\text{trim}}$ )
Pratio	real	$P_{\text{reqEG}}/P_{\text{avEG}}$
Pmargin	real	power margin, $P_{\text{avEG}} - P_{\text{reqEG}}$
Plimit_es	real	drive system limit $P_{\text{ESlimit}}$ (at $N_{\text{trim}}$ and rating_ds)
atPlimit_es	int	at drive system limit ( $P_{\text{avEG}}$ limited by $P_{\text{ESlimit}}$ )
Qmargin_es	real	torque margin, $P_{\text{ESlimit}} - P_{\text{reqEG}}$
exceedQ_es	int	exceed torque available: $P_{\text{reqEG}} > (1 + \epsilon)P_{\text{ESlimit}}$
fuelflow	real	fuel flow $\dot{w}$ (negative if generated)
energyflow	real	energy flow $\dot{E}$ (negative if generated)
fuelflow_equiv	real	equivalent fuel flow $\dot{w}_{\text{equiv}}$ , from energy flow
sfc	real	specific fuel consumption $\text{sfc} = \dot{w}_{\text{equiv}}/P_{\text{req}}$
FNEG	real	net installed jet thrust $F_N$
DauxEG	real	momentum drag of auxiliary air flow $D_{\text{aux}}$
Fjet(3)	real	jet thrust force $F_{\text{jet}}^F$ (F axes, about cg)
Mjet(3)	real	jet thrust moment $M_{\text{jet}}^F$ (F axes, about cg)
Faux(3)	real	momentum drag force $F_{\text{aux}}^F$ (F axes, about cg)
Maux(3)	real	momentum drag moment $M_{\text{aux}}^F$ (F axes, about cg)
		aerodynamics
Vaero(3)	real	total velocity relative air $v^F$ (F axes)
Vmag	real	velocity magnitude
q	real	dynamic pressure
ed(3)	real	drag vector, $-v/ v $ in F axes

VB(3)	real	total velocity relative air $v^B$ (B axes)
alpha	real	angle of attack $\alpha$ (deg)
CD	real	drag coefficient $C_D$
D	real	drag
		load
Faero(3)	real	aerodynamic force $F_{\text{aero}}^F$ (F axes, about cg)
Maero(3)	real	aerodynamic moment $M_{\text{aero}}^F$ (F axes, about cg)
Drag	real	drag $e_d^T F_{\text{aero}}^F$
Download	real	download, aero $F_z$ (I axes)

**Structure: FltJet**

Variable	Type	Description	Default
		Flight State - Jet Group	
		controls	
amp	real	amplitude $A$	
mode	real	mode $B$	
incid	real	incidence $i$	
yaw	real	yaw $\psi$	
		geometry	
CBF(3,3)	real	jet relative airframe, $C^{BF}$	
ef(3)	real	jet direction, $e_f$	
		jet	
Tq	real	uninstalled thrust required, $T_q$	
Tlossq	real	installation loss $T_{loss}$	
etalossq	real	installation efficiency $\eta_{loss}$	
Treq_jet	real	installed thrust required, $T_{req-jet}$	
mdot	real	mass flow $\dot{m}$	
wdot	real	fuel flow $\dot{w}$	
Edot	real	energy flow $\dot{E}$	
FG	real	gross installed jet thrust $F_G$	
Fmom	real	momentum thrust $F_{mom}$	
FN	real	net installed jet thrust $F_N$	
Daux	real	momentum drag of auxiliary air flow $D_{aux}$	
Ta	real	uninstalled thrust available, $T_a$	
Tlossa	real	installation loss $T_{loss}$	
etalossa	real	installation efficiency $\eta_{loss}$	
Tav_jet	real	installed thrust available, $T_{av-jet}$	
Tmech	real	jet mechanical limit $T_{mech}$	
atTmech	int	at mechanical limit ( $T_{av-jet}$ limited by $T_{mech}$ )	

ReactionMode	int	jet group reaction drive mode (MODEL_jet_react or converted)
Converted	int	converted (RPJEM with mode=1; 0 jet, 1 reaction)
Treq	real	thrust required $T_{reqJG}$
TavJI	real	jet installed thrust available $T_{avJI}$ ; (Njet-NJetInOp) $T_{av-jet}$
Tav	real	thrust available, $T_{avJG}$ ; fThrust(Njet-NJetInOp) $T_{av-jet}$
Jratio	real	$T_{reqJG}/T_{avJG}$
Jmargin	real	thrust margin, $T_{avJG} - T_{reqJG}$
exceedJ	int	exceed thrust available: $T_{reqJG} > (1 + \epsilon)T_{avJG}$
fuelflow	real	fuel flow $\dot{w}$ (negative if generated)
energyflow	real	energy flow $\dot{E}$ (negative if generated)
fuelflow_equiv	real	equivalent fuel flow $\dot{w}_{equiv}$ , from energy flow
sfc	real	specific fuel consumption $sfc = \dot{w}_{equiv}/T_{req}$
FNJG	real	net installed jet thrust $F_N$
DauxJG	real	momentum drag of auxiliary air flow $D_{aux}$
Fjet(3)	real	jet thrust force $F_{jet}^F$ (F axes, about cg)
Mjet(3)	real	jet thrust moment $M_{jet}^F$ (F axes, about cg)
Faux(3)	real	momentum drag force $F_{aux}^F$ (F axes, about cg)
Maux(3)	real	momentum drag moment $M_{aux}^F$ (F axes, about cg)
		loads
F(3)	real	force $F_{jet}^F$ (F axes)
M(3)	real	moment $M_{jet}^F$ (F axes)
		aerodynamics
Vaero(3)	real	total velocity relative air $v^F$ (F axes)
Vmag	real	velocity magnitude
q	real	dynamic pressure
ed(3)	real	drag vector, $-v/ v $ in F axes
VB(3)	real	total velocity relative air $v^B$ (B axes)
alpha	real	angle of attack $\alpha$ (deg)
CD	real	drag coefficient $C_D$
D	real	drag
		load
Faero(3)	real	aerodynamic force $F_{aero}^F$ (F axes, about cg)
Maero(3)	real	aerodynamic moment $M_{aero}^F$ (F axes, about cg)

Structure: FltJet

112

Drag	real	drag $e_d^T F_{\text{aero}}^F$
Download	real	download, aero $F_z$ (I axes)



**Structure: FltChrg**

Variable	Type	Description	Default
		Flight State - Charge Group	
		controls	
amp	real	amplitude $A$	
mode	real	mode $B$	
incid	real	incidence $i$	
yaw	real	yaw $\psi$	
		geometry	
CBF(3,3)	real	charger relative airframe, $C^{BF}$	
ef(3)	real	charger direction, $e_f$	
		charger	
Pacell	real	cell power available $\dot{E}_{acell} = P_0$	
Edotacell	real	cell energy flow available $\dot{E}_{acell} = P_0$	
Edotqcell	real	cell energy flow required $\dot{E}_{qcell} = \dot{E}_{req}/\eta_{chrg}$	
Pqcell	real	cell power required $\dot{E}_{qcell} = \dot{E}_{req}/\eta_{chrg}$	
etachrg	real	charger efficiency $\eta_{chrg}$	
Preq	real	installed power required $\dot{E}_{req} = \dot{E}_{reqCG} K_{ffd}/(Ncharge - NChrgInOp)$	
		charger, fuel cell	
mdot_burn	real	mass flow $\dot{m}$	
wdot_burn	real	fuel flow $\dot{w}$	
FG	real	gross installed jet thrust $F_G$	
Fmom	real	momentum thrust $F_{mom}$	
FN	real	net installed jet thrust $F_N$	
Daux	real	momentum drag of auxiliary air flow $D_{aux}$	
		charge group	
Pchrg	real	power required $P_{reqCG} = \dot{E}_{reqCG}$	
Edotchrg	real	energy flow required $P_{reqCG} = \dot{E}_{reqCG}$	
Preqtotal	real	total cell power required $P_{reqtotal}; (Ncharge - NChrgInOp)\dot{E}_{qcell}$	

Pavtotal	real	total cell power available $P_{avtotal}$ ; fCharge(Ncharge–NChrgInOp) $\dot{E}_{acell}$
Cratio	real	$P_{reqtotal}/P_{avtotal}$
Cmargin	real	power margin, $P_{avtotal} - P_{reqtotal}$
exceedC	int	exceed power available: $P_{reqtotal} > (1 + \epsilon)P_{avtotal}$
energyflow	real	energy flow $\dot{E}$ (negative if generated)
fuelflow_equiv	real	equivalent fuel flow $\dot{w}_{equiv}$ , from energy flow
		charger, fuel cell
		fuel burn
fuelflow_burn	real	fuel flow $\dot{w}$
energyflow_burn	real	energy flow $\dot{E}$
fuelflow_equiv_burn	real	equivalent fuel flow $\dot{w}_{equiv}$ , from energy flow
sfc	real	specific fuel consumption $sfc = \dot{w}_{equiv}/P_{req}$
FNCG	real	net installed jet thrust $F_N$
DauxCG	real	momentum drag of auxiliary air flow $D_{aux}$
Fjet(3)	real	jet thrust force $F_{jet}^F$ (F axes, about cg)
Mjet(3)	real	jet thrust moment $M_{jet}^F$ (F axes, about cg)
Faux(3)	real	momentum drag force $F_{aux}^F$ (F axes, about cg)
Maux(3)	real	momentum drag moment $M_{aux}^F$ (F axes, about cg)
		loads
F(3)	real	force $F_{chrg}^F$ (F axes)
M(3)	real	moment $M_{chrg}^F$ (F axes)
		aerodynamics
Vaero(3)	real	total velocity relative air $v^F$ (F axes)
Vmag	real	velocity magnitude
q	real	dynamic pressure
ed(3)	real	drag vector, $-v/ v $ in F axes
VB(3)	real	total velocity relative air $v^B$ (B axes)
alpha	real	angle of attack $\alpha$ (deg)
CD	real	drag coefficient $C_D$
D	real	drag
		load
Faero(3)	real	aerodynamic force $F_{aero}^F$ (F axes, about cg)
Maero(3)	real	aerodynamic moment $M_{aero}^F$ (F axes, about cg)

Structure: FltChrg

115

Drag	real	drag $e_d^T F_{aero}^F$
Download	real	download, aero $F_z$ (I axes)

## Chapter 31

**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 (0 no, 1 force retrim)	0
trace_trim	int	+ trace iteration (0 for none, 2 for component controls)	0
		+ 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 quadratic, 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
trace_count	int		counter	

---

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 $C_T/\sigma$ )	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
		Derived	
Wscale	real	weight scale	
Pscale	real	power scale	
Lscale	real	length scale	
Ascale	real	angle scale	
Fscale	real	force scale	
Mscale	real	moment scale	
Vscale	real	horizontal velocity scale	
Rscale	real	vertical velocity scale	
Oscale	real	angular velocity scale	
Tscale	real	$C_T/\sigma$ scale	
Cscale	real	$C_L$ scale	
Hscale	real	altitude scale	
Gscale	real	acceleration scale	
Xscale	real	range scale	

## Chapter 32

**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	1994
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
<hr/> inflation: $F_i$ multiplies airframe purchase price and maintenance cost factor inflation always used, even with internal table CPI or DoD table: $F_i = \text{inflation}(F_{\text{table}}(\text{year\_inf})/F_{\text{table}}(1994))$ input factor: $F_i = \text{inflation}(\text{relative } 1994)$ cost factors and rates include technology and inflation, correspond to year_inf <hr/>			
		+ Cost	
MODEL_cost	int	+ model (1 CTM)	1
CostCTM	CostCTM	CTM cost model	
FuelPrice(ntaxmax)	real	+ fuel price $G$ (\$/gallon or \$/liter; or \$/MJ)	5.0
Npass	int	+ number of passengers $N_{\text{pass}}$	100
		+ 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.



Structure: Cost

121

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.
		+	Technology Factors	
TECH_cost_af	real	+	airframe $\chi_{AF}$	0.87
TECH_cost_maint	real	+	maintenance $\chi_{\text{maint}}$	1.0

## Chapter 33

**Structure: CostCTM**

Variable	Type	Description	Default
		+ 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
		+ airframe	
rComp	real	+ additional cost rate $r_{comp}$ for composite construction (\$/lb or \$/kg)	0.0
fWcomp_body	real	+ composite weight in body (fraction body weight)	0.0
fWcomp_tail	real	+ composite weight in tail (fraction tail weight)	0.0
fWcomp_pylon	real	+ composite weight in pylon (fraction pylon weight)	0.0
fWcomp_wing	real	+ composite weight in wing (fraction wing weight)	0.0
		+ systems (fixed useful load)	
rFCE	real	+ cost factor $r_{FCD}$ , flight control electronics (\$/lb or \$/kg)	10000.
rMEP	real	+ cost factor $r_{MEP}$ , mission equipment package (\$/lb or \$/kg)	10000.
<hr/> cost factors and rates include technology and inflation, correspond to year_inf rComp negative for cost reduction <hr/>			
		+ 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 $M_{labor}$	0.0017
Mparts	real	+ parts factor $M_{parts}$	34.
Mengine	real	+ engine overhaul factor $M_{engine}$	1.45
Mmajor	real	+ major periodic maintenance factor $M_{major}$	18.

---

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 $K_{cdi}$	1.0
Kcrew	real	+	crew cost factor $K_{crew}$	1.0

## Chapter 34

**Structure: Aircraft**

Variable	Type	Description	Default
		+ Aircraft	
title	c*100	+ title	
notes	c*1000	+ notes	
config	c*16	+ Configuration	'helicopter'
RCconfig	int	configuration (RCconfig_rotorcra <sub>f</sub> t, helicopter, tandem, coaxial, tiltrotor)	
nRotor_main	int	number of main rotors	
<hr/> config: identifies rotorcraft configuration config = 'rotorcra <sub>f</sub> t', 'helicopter', 'tandem', 'coaxial', 'tiltrotor'			
		+ 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) pilot's controls (control number)	1
kcoll	int	collective stick	
klatcyc	int	lateral cyclic stick	
kIngcyc	int	longitudinal stick	
kpedal	int	pedal	
ktilt	int	tilt	
		+ control values (function speed)	
nVcont(ncontmax)	int	+ number of speeds (0 zero value; 1 constant; $\geq 2$ piecewise linear, maximum nvelmax)	
nVcoll	int	+ collective stick	0
nVlatcyc	int	+ lateral cyclic stick	0
nVIngcyc	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 $\alpha_{tilt}$	
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 $\theta_F$	
nVpitch	int	+	number of speeds (0 zero value; 1 constant; $\geq 2$ piecewise linear, maximum nvelmax)	
pitch(nvelmax)	real	+	values	
Vpitch(nvelmax)	real	+	speeds (CAS or TAS)	
			+ aircraft roll angle $\phi_F$	
nVroll	int	+	number of speeds (0 zero value; 1 constant; $\geq 2$ piecewise linear, maximum nvelmax)	
roll(nvelmax)	real	+	values	
Vroll(nvelmax)	real	+	speeds (CAS or TAS)	
<hr/>				
			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	
<hr/>				
			+ 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 ( $V \leq V_{\text{conv-hover}}$ )	1
kcont_conv	int	+	conversion mode ( $V_{\text{conv-hover}} < V < V_{\text{conv-cruise}}$ )	1
kcont_cruise	int	+	cruise mode ( $V \geq V_{\text{conv-cruise}}$ )	1
			+ drive system state (each propulsion group)	
kgear_hover(npropmax)	int	+	hover and helicopter mode ( $V \leq V_{\text{conv-hover}}$ )	1
kgear_conv(npropmax)	int	+	conversion mode ( $V_{\text{conv-hover}} < V < V_{\text{conv-cruise}}$ )	1
kgear_cruise(npropmax)	int	+	cruise mode ( $V \geq V_{\text{conv-cruise}}$ )	1
<hr/>				
			conversion control: use depends on STATE_control, SET_tilt, SET_Vtip of FltState	
			hover and helicopter mode ( $V \leq V_{\text{conv-hover}}$ ): use tilt=90, Vtip_hover, kgear_hover, kcont_hover	
			cruise mode ( $V \geq V_{\text{conv-cruise}}$ ): 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	
<hr/>				

SET_Vschedule	int	+	Velocity schedules (1 CAS, 2 TAS)	1
<hr/>				
velocity schedules: all described as function CAS or TAS conversion, controls and motion, rotor tip speed, landing gear retraction, trim targets				
<hr/>				
		+	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
Derived				
itrim_quant(mtrimmax,ntrimstatemax)	int		trim quantity name (TRIM_QUANT_xxx)	
itrim_quantn(mtrimmax,ntrimstatemax)	int		trim quantity structure number	
itrim_quantk(mtrimmax,ntrimstatemax)	int		trim quantity kind (0 other, 1 rotor, 2 rotor lift, 3 rotor prop, 4 wing, 5 wing lift)	
itrim_var(mtrimmax,ntrimstatemax)	int		trim variable name (TRIM_VAR_xxx, or control number)	
itrim_varn(mtrimmax,ntrimstatemax)	int		trim variable structure number	
<hr/>				
trim state: one or more set of quantities and variables for trim iteration FltState identifies trim state (STATE_trim match IDENT_trim),				

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	'PEG margin n'	FltState%trim_target
jet group thrust	'jet n'	FltState%trim_target
jet thrust margin	'J 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', 'T margin tran n'	FltState%trim_target
rotor flapping	'betac n', 'Ingflap 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



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'	

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\_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	+	buttlines	
WL_Ref	real	+	waterline	

		calculated reference point (input or component)
SLref	real	stationline
BLref	real	buttline
WLref	real	waterline
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; arbitrary 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 $\tau$	
dtemp	real	+ temperature increment $\Delta T$	0.
density	real	+ density $\rho$	
csound	real	+ speed of sound $c_s$	
viscosity	real	+ viscosity $\mu$	
altitude	real	+ altitude	
		Derived	
iSET_atmos	int	atmosphere (SET_atmos_XXX)	
density_to	real	density $\rho$	
sigma_to	real	density ratio $\rho/\rho_0$	
theta_to	real	temperature ratio $T/T_0$	
delta_to	real	pressure ratio $p/p_0$	

---

takeoff condition (density) used for  $C_T/\sigma$  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)  
 see FltState%SET\_atmos for other options (polar, tropical, and hot days)

---

		Size
diskload	real	aircraft disk loading
Aref	real	reference rotor area
wingload	real	aircraft wing loading
Sref	real	reference wing area
Pav	real	total takeoff power available
powerload	real	aircraft power loading
Tav	real	total takeoff thrust available
thrustload	real	aircraft weight-to-thrust

---

aircraft disk loading =  $W_D/A_{\text{ref}}$ ,  $A_{\text{ref}} = \sum f_A A$ ; rotor disk loading =  $f_W W_D/A$   
 aircraft wing loading =  $W_D/S_{\text{ref}}$ ,  $S_{\text{ref}} = \sum S$ ; individual wing loading =  $f_W W_D/S$   
 aircraft power loading =  $W_D/P_{\text{av}}$ ,  $P_{\text{av}} = \sum N_{\text{eng}} P_{\text{eng}}$  (each engine group at takeoff rating)  
 aircraft thrust-to-weight =  $W_D/T_{\text{av}}$ ,  $T_{\text{av}} = \sum N_{\text{jet}} T_{\text{jet}}$  (each jet group at takeoff rating)

---

		Configuration
nWingExt	int	wing extensions (0 for none)
nWingExtKit	int	wing extension kits (0 for none)
nWingKit	int	wing kits (0 for none)
nWotherkit	int	other kit (0 for none)
SET_fold	int	folding (0 none, 1 fold weights, 2 with kit) (from Systems)

		Neutral point
SLna	real	stationline $SL_{na}$
		Operating size (hover; controls = 0 except tilt = 90)
length_op	real	length
width_op	real	width
area_op	real	area
		Fuel tank system
burnweight	int	first fuel tank that burns weight (0 none)
eref	real	reference specific energy (MJ/kg)
		Cost
CAC	real	aircraft $C_{AC}$
CAC_nokit	real	aircraft $C_{AC}$ , folding kit not installed
Cmaint	real	maintenance $C_{maint}$
Cmaint_nokit	real	maintenance $C_{maint}$ , folding kit not installed
factor_inf	real	inflation factor $F_i$ (year_inf relative 1994, including factor inflation)
factor_inf2011	real	inflation factor $F_i$ (2011 relative 1994, CPI)
Ccomp	real	composite cost increment $C_{comp}$
CMEP	real	mission equipment package cost $C_{MEP}$
CFCE	real	flight control electronics cost $C_{FCE}$
Wcomp	real	composite weight increment $W_{comp}$
WMEP	real	mission equipment package weight $W_{MEP}$
WFCE	real	flight control electronics weight $W_{FCE}$
Kconfig	real	configuration factor, $K_{ET}K_{EN}K_{LG}K_R$
rAF	real	airframe $C_{AF}/W_{AF}$ (\$/lb or \$/kg)
rAC	real	total aircraft $C_{AC}/W_{EK}$ (\$/lb or \$/kg)
WAFcost	real	airframe weight $W_{AF}$
WEKcost	real	$W_{EK}$ = weight empty + airframe kits = $W_{AF} + W_{MEP} + W_{FCE}$
Pcost	real	rated takeoff power $P$
Clabor	real	labor cost $C_{labor}$
Cparts	real	parts cost $C_{parts}$
Engine	real	engine overhaul cost $C_{engine}$
Cmajor	real	major periodic maintenance cost $C_{major}$
MMHperFH	real	maintenance man hours per flight hour

		+	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	
		+	structural design gross weight	
SDGW	real	+	structural design gross weight $W_{SD}$	
dSDGW	real	+	gross weight increment	0.
fSDGW	real	+	gross 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	+	gross weight increment	0.
fWMTO	real	+	gross weight factor	1.
nz_ult	real	+	design ultimate flight load factor $n_{zult}$ 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

if calculated, then input parameter is initial value

fixed weight empty (FIX\_WE=1): adjust contingency weight to achieve

Wfuel\_DGW and Wpay\_DGW usually calculated (identified as input so inherited by next case)

SET\_SDGW, structural design gross weight:

'input' = input

'f(DGW)' = based on DGW;  $W_{SD} = dSDGW + fSDGW * W_D$

'maxfuel' = based on fuel state;  $W_{SD} = dSDGW + fSDGW * W_G$ ,  $W_G = W_D - W_{fuel\_DGW} + fFuelSDGW * W_{fuel-cap}$

'perf' = calculated from maximum gross weight at SDGW sizing conditions (DESIGN\_sdgw)

SET\_WMTO, maximum takeoff weight:

'input' = input

'f(DGW)' = based on DGW;  $W_{MTO} = dWMTO + fWMTO * W_D$

'maxfuel' = based on maximum fuel;  $W_{MTO} = dWMTO + fWMTO * W_G$ ,  $W_G = W_D - W_{fuel\_DGW} + W_{fuel-cap}$

'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	
Weight	Weight		aircraft weight statement (operating weight, without payload and usable fuel)	
WO	real		operating weight $W_O$	
WE	real	+	weight empty $W_E$	
growth_factor	real		growth factor = $W_D / (W_D - W_{scaled} - W_{fuel})$	
			+ 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 $k_z / L$	
			Derived moments of inertia (corresponding to aircraft weight statement)	
lxx	real		$I_{xx}$	
lyy	real		$I_{yy}$	
lzz	real		$I_{zz}$	
lxy	real		$I_{xy}$	
lyz	real		$I_{yz}$	
lxx	real		$I_{xz}$	
<hr/>				
			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	
<hr/>				
			+ 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, $D/q = A_{ref} C_D$ )	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	+	$k_{DL} = (D/q)_V / A_{ref}$	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  $\text{ft}^2/\text{klb}^{2/3}$  or  $\text{m}^2/\text{Mg}^{2/3}$   
 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)

---

KIND_alpha	int	+	Aerodynamics 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

---

			Derived	
DoQC_comp	real		sum component cruise drag, area $(D/q)_{comp}$ (without contingency)	
DoQH_comp	real		sum component helicopter drag, area $(D/q)_{comp}$ (without contingency)	
DoQV_comp	real		sum component vertical drag, area $(D/q)_{comp}$ (without contingency)	
DoQC_AC	real		total cruise drag, area $(D/q)_{AC}$	
DoQH_AC	real		total helicopter drag, area $(D/q)_{AC}$	
DoQV_AC	real		total vertical drag, area $(D/q)_{AC}$	
CDC_AC	real		total cruise $(D/q)_{AC} / A_{ref}$	
CDH_AC	real		total helicopter $(D/q)_{AC} / A_{ref}$	
kDragC_AC	real		total cruise $(D/q) / (W_{MTO} / 1000)^{2/3}$	

kDragH_AC	real		total helicopter $(D/q)/(W_{MTO}/1000)^{2/3}$	
kDL_AC	real		total vertical $(D/q)_V/A_{ref}$	
DoQwet_AC	real		total cruise wetted drag, area $(D/q)_{wet}$	
Swet_AC	real		total wetted area $S_{wet}$	
CD_AC	real		total cruise $(D/q)_{wet}/S_{wet}$	
		+	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
nEngineTable	int	+	engine tables (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 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 batter, used in one or more fuel tanks

---



		Aircraft Input for case
inAircraft	int	Aircraft
inSystems	int	Systems
inFuselage	int	Fuselage
inLandingGear	int	LandingGear
inRotor(nrotormax)	int	Rotor
inWing(nwingmax)	int	Wing
inTail(ntailmax)	int	Tail
inFuelTank(ntankmax)	int	FuelTank
inPropulsion(npropmax)	int	Propulsion
inEngineGroup(nengmax)	int	EngineGroup
inJetGroup(njetmax)	int	JetGroup
inChargeGroup(nchrgmax)	int	ChargeGroup
inEngineModel(nengmax)	int	EngineModel
inEngineParamN(nspeedmax,nengmax)	int	ParamN
inEngineTable(nengmax)	int	EngineTable
inCompressorModel(nengmax)	int	CompressorModel
inMotorModel(nengmax)	int	MotorModel
inJetModel(njetmax)	int	JetModel
inFuelCellModel(nchrgmax)	int	FuelCellModel
inSolarCellModel(nchrgmax)	int	SolarCellModel
inBatteryModel(ntankmax)	int	BatteryModel
inCost	int	Cost
Design specification (from Size)		
iSIZE_perf(npropmax)	int	performance (SIZE_perf_engine, rotor, param, none)
iSIZE_jet(njetmax)	int	performance (SIZE_jet_jet, param, none)
iSIZE_charge(nchrgmax)	int	performance (SIZE_charge_chrg, param, none)
iSIZE_rotor(nrotormax)	int	rotor sized (SIZE_rotor_radius, thrust, none)
iSET_rotor_radius(nrotormax)	int	rotor radius (SET_rotor_radius, DL, ratio, scale, not_radius)
FIX_rotor_CWs(nrotormax)	int	rotor $C_W/\sigma$ (1 fixed, 0 not)
FIX_rotor_Vtip(nrotormax)	int	rotor $V_{tip}$ (1 fixed, 0 not)
FIX_rotor_sigma(nrotormax)	int	rotor $\sigma$ (1 fixed, 0 not)

iSET_wing_area(nwingmax)	int	wing area (SET_wing_area, WL, not_area)
iSET_wing_span(nwingmax)	int	wing span (SET_wing_span, ratio, radius, width, hub, panel, not_span)
FIX_wing_chord(nwingmax)	int	wing chord (1 fixed, 0 not)
FIX_wing_AR(nwingmax)	int	wing aspect ratio (1 fixed, 0 not)
FIX_DGW	int	design gross weight (0 calculated, 1 fixed)
FIX_WE	int	weight empty (0 calculated, 1 fixed)
iSET_tank(ntankmax)	int	fuel tank (SET_tank_input, miss, misspower)
iSET_SDGW	int	SDGW (SET_SDGW_input, fDGW, maxfuel, perf)
iSET_WMTO	int	WMTO (SET_WMTO_input, fDGW, maxfuel, perf)
iSET_limit_ds(npropmax)	int	drive system torque limit (SET_limit_input, ratio, Pav, Preq)
kind_iter_size	int	kind iteration, performance (0 none, 1 size engine or radius or jet group or charge group)
kind_iter_param	int	kind iteration, parameters (0 none, 1 calculate parameters)
nSIZE_engine(npropmax)	int	conditions and missions for size engine or rotor
nSIZE_jet(njetmax)	int	conditions and missions for size jet group
nSIZE_charge(nchrgmax)	int	conditions and missions for size charge group
nDESIGN_GW	int	design conditions and missions for DGW
nDESIGN_xmsn(npropmax)	int	design conditions and missions for transmission
nDESIGN_wmto	int	design conditions for WMTO
nDESIGN_tank	int	design missions for fuel tank
nDESIGN_thrust	int	design conditions and missions for antitorque or aux thrust rotor
		Design data (from sizing)
DGW_source	int	design gross weight source (1 condition, 2 mission)
DGW_kState	int	design gross weight source number
DGW_kSeg	int	design gross weight segment number
nDesignState	int	number design of conditions and missions (maximum ndesignmax)
XAircraft(ndesignmax)	XAircraft	design data

**Structure: XAircraft**

Variable	Type	Description	Default
Design Data			
source	int	source (1 condition, 2 mission)	
kState	int	source number	
kSeg	int	segment number	
title	c*100	title	
kind	c*12	kind (condition or mission)	
number	c*12	number (condition or mission/segment)	
label	c*12	label	
setgw	c*12	Set Gross Weight	
setul	c*12	Set Useful Load	
design	c*12	design	
Nauxtank(nauxtankmax,ntankmax)	int	number of auxiliary fuel tanks $N_{\text{auxtank}}$ (each aux tank size)	
Ncrew	int	number of crew	
Npass	int	number of passengers	
Ncrew_seat	int	number of crew seats	
Npass_seat	int	number of passenger seats	
kits	c*12	kits	
Weights (from FltAircraft)			
GW	real	gross weight $W_G$ ; at segment start	
Wpayload	real	payload weight $W_{\text{pay}}$	
Wpay_pass	real	passengers $W_{\text{pass}}$	
Wpay_cargo	real	cargo $W_{\text{cargo}}$	
Wpay_extload	real	external load $W_{\text{ext-load}}$	
Wpay_ammunition	real	ammunition $W_{\text{ammo}}$	
Wpay_weapons	real	weapons $W_{\text{weapons}}$	
Wpay_other	real	other $W_{\text{other}}$	
Wfuel_total	real	usable fuel weight $W_{\text{fuel}}$ ; at segment start	

Wfuel(ntankmax)	real	usable fuel weight
Wfuel_std(ntankmax)	real	standard tanks
Wfuel_aux(ntankmax)	real	auxiliary tanks
WO	real	operating weight $W_O$
WE	real	weight empty $W_E$ (from Aircraft)
WFixUL	real	fixed useful load $W_{FUL}$
Wcrew	real	crew
W_fixUL_fluid	real	fluids (from Aircraft%Weight)
Wauxtank	real	auxiliary fuel tanks
W_fixUL_other	real	other fixed useful load
Woful(10)	real	catagories
Wequip	real	equipment increment
Wfoldkit	real	folding kit
Wextkit	real	wing extension kit
Wwingkit	real	wing kit
Wotherkit	real	other kit
WUL	real	useful load $W_{UL}$
WML	real	military load
Efuel_total	real	Energy (from FltAircraft)
Efuel(ntankmax)	real	usable fuel energy $E_{fuel}$ ; at segment start
Efuel_std(ntankmax)	real	usable fuel energy
Efuel_aux(ntankmax)	real	standard tanks
		auxiliary tanks

## Chapter 36

**Structure: Systems**

Variable	Type	Description	Default
		+ Systems	
title	c*100	+ title	
notes	c*1000	+ notes	
		+ Weight	
Weight	Weight	weight statement (systems)	
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

---

Structure: Systems

142

SET_fold	int	+	folding (0 none, 1 fold weights, 2 with kit)	0
		+	folding weight in kit $f_{\text{foldkit}}$ (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 $W_{\text{vib}}$	
fWvib	real	+	fraction weight empty $f_{\text{vib}}$	
SET_Wcont	int	+	contingency weight (1 fraction weight empty, 2 input)	1
Wcont	real	+	weight $W_{\text{cont}}$	
fWcont	real	+	fraction weight empty $f_{\text{cont}}$	

---


$$W_E = (\text{structure} + \text{propulsion group} + \text{systems and equipment}) + W_{\text{vib}} + W_{\text{cont}}$$

$$\text{SET\_Wvib: } W_{\text{vib}} \text{ input or } W_{\text{vib}} = f_{\text{vib}} W_E$$

$$\text{SET\_Wcont: } W_{\text{cont}} \text{ input or } W_{\text{cont}} = f_{\text{cont}} W_E, \text{ or adjust } W_{\text{cont}} \text{ for input } W_E \text{ (FIX\_WE=1)}$$

SET\_fold, folding:

set component  $dW_{\text{xxfold}}=0$  and  $fW_{\text{xxfold}}=0$  for no rotor/wing/tail/body fold weight

fraction  $fW_{\text{foldkit}}$  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+WDlect

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):

$U_{crew\_seat\_inc} = U_{seat\_crew} + U_{accom\_crew} + U_{ox\_crew} + U_{armor\_crew}$

$U_{pass\_seat\_inc} = U_{seat\_pass} + U_{accom\_pass} + U_{ox\_pass}$

---



		Derived	
		fixed useful load, fold kit	
W_fixUL_foldkit_fus	real	fuselage	
W_fixUL_foldkit_roto	real	rotors	
W_fixUL_foldkit_wing	real	wings	
W_fixUL_foldkit_tail	real	tails	
		armament group	
Warmor_floor	real	cabin floor armor weight	
Warmor_wall	real	cabin wall armor weight	
Warmor_crew	real	crew armor weight	
		furnishings and equipment group	
Wseat	real	seats	
Wacom	real	miscellaneous accommodation	
Wox	real	oxygen system	
Winsulation	real	acoustic and thermal insulation weight	
Whandling_cargo	real	cargo handling weight	
Ucrewseatinc	real	equipment weight increment per crew seat	
Upasseatinc	real	equipment weight increment per passenger seat	
Wtip(nrotormax)	real	weight on wing tip	
		+ Weight	
		+ systems and equipment	
		+ flight control group and hydraulic group	
MODEL_fc	int	+ model (0 input, 1 AFDD, 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.

Structure: Systems

146

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.
WFltCont	WFltCont		AFDD model	
		+	anti-icing group	
MODEL_DI	int	+	model (0 input, 1 AFDD, 2 custom)	1
		+	weight increment	
dWDlelect	real	+	electrical system	0.
dWDlsys	real	+	anti-ice system	0.
WDelce	WDelce		AFDD model	

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = \text{TECH}_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use } \text{MODEL}_{xx}=0 \text{ or } \text{TECH}_{xx}=0.$$

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 $\chi_{RWb}$	1.0
TECH_RWfc_mb	real	+	control boost mechanisms $\chi_{RWmb}$	1.0
TECH_RWfc_nb	real	+	non-boosted $\chi_{RWnb}$	1.0

		+	fixed wing flight control weight	
TECH_FWfc_mb	real	+	control boost mechanisms $\chi_{FWmb}$	1.0
TECH_FWfc_nb	real	+	non-boosted $\chi_{FWnb}$	1.0
		+	conversion flight control weight	
TECH_CVfc_mb	real	+	control boost mechanisms $\chi_{CVmb}$	1.0
TECH_CVfc_nb	real	+	non-boosted $\chi_{CVnb}$	1.0
		+	flight control hydraulics	
TECH_RWhyd	real	+	rotary wing $\chi_{RWhyd}$	1.0
TECH_FWhyd	real	+	fixed wing $\chi_{FWhyd}$	1.0
TECH_CVhyd	real	+	conversion $\chi_{CVhyd}$	1.0
		+	anti-icing	
TECH_Dlect	real	+	electrical system $\chi_{Dlect}$	1.0
TECH_Dlsys	real	+	anti-ice system $\chi_{Dlsys}$	1.0

**Structure: WFltCont**

Variable	Type	Description	Default
		+ Flight Control Group, AFDD Weight Model	
		+ rotary wing flight controls	
		+ non-boosted controls	
MODEL_WRWfc	int	+ model (1 fraction, 2 parametric)	1
fRWfc_nb	real	+ weight $f_{RWnb}$ (fraction boost mechanisms weight)	0.6
xRWfc_red	real	+ hydraulic system redundancy/complexity factor $f_{RWred}$	3.0
KIND_WRWfc	int	+ survivability (1 baseline, 2 UTTAS/AAH level of survivability)	2
		+ fixed wing flight controls	
MODEL_WFWfc	int	+ model (1 full controls, 2 only on horizontal tail)	1
fFWfc_nb	real	+ non-boosted weight $f_{FWnb}$ (fraction total fixed wing flight control weight)	0.10
		+ conversion flight 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
		+ Hydraulic Group, AFDD Model	
		+ flight control hydraulics	
fRWhyd	real	+ rotary wing $f_{RWhyd}$ (fraction rotary wing boost mechanisms + hydraulic weight)	0.40
fFWhyd	real	+ fixed wing $f_{FWhyd}$ (fraction fixed wing boost mechanisms weight)	0.10
fCVhyd	real	+ conversion $f_{CVhyd}$ (fraction conversion boost mechanisms weight)	0.10
<hr/> only MODEL_WRWfc=fraction uses fRWfc_nb typically fRWfc_nb = 0.6 (data range 0.3 to 1.8), fRWhyd = 0.4 <hr/>			
WtParam_fc(8)	real	+ Custom Weight Model + parameters	0.

**Structure: WDeIce**

Variable	Type	Description	Default
		+ Anti-Icing Group, AFDD Weight Model	
kDelce_elec(nrotormax)	real	+ weight factor for electrical system $K_{elec}$ (lb/ft <sup>2</sup> or kg/ft <sup>2</sup> )	0.25
kDelce_rotor(nrotormax)	real	+ weight factor for main rotor $K_{rotor}$ (lb/ft <sup>2</sup> or kg/ft <sup>2</sup> )	0.25
kDelce_wing(nwingmax)	real	+ weight factor for wing $K_{wing}$ (lb/ft or kg/ft)	0.0
kDelce_air(nengmax)	real	+ weight factor for engine air intake $K_{air}$ (lb/lb or kg/kg)	0.006
kDelce_jet(njetmax)	real	+ weight factor for jet air intake $K_{jet}$ (lb/lb or kg/kg)	0.006
		+ Custom Weight Model	
WtParam_DI(8)	real	+ parameters	0.

**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 $\ell_{fus}$	
SET_nose	int	+ nose length (distance forward of hub; 1 input, 2 calculated)	1
Length_nose	real	+ nose length $\ell_{nose}$	
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 $\ell_{aft}$	
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)	2
Swet	real	+ wetted area $S_{wet}$	
Sproj	real	+ projected area $S_{proj}$	
fSwet	real	+ factor for wetted area $f_{wet}$	1.
fSproj	real	+ factor for projected area $f_{proj}$	1.
Height_fus	real	+ fuselage height $h_{fus}$	
Circum_boom	real	+ tail boom effective circumference $C_{boom}$	
Width_boom	real	+ tail boom effective width $w_{boom}$	
Swet_in	real	+ input wetted area $S_{wet}$	
Sproj_in	real	+ input projected area $S_{proj}$	
SET_Scabin	int	+ cabin area (1 input, 2 calculated)	2
Scabin	real	+ total cabin surface area $S_{cabin}$	

Structure: Fuselage

151

Scabin_floor	real	+	cabin floor area $S_{cabin-floor}$	
Scabin_wall	real	+	cabin wall area $S_{cabin-wall}$	
fScabin	real	+	factor for total cabin surface area $f_{cabin}$	0.6
fScabin_floor	real	+	factor for cabin floor area $f_{cabin-floor}$	0.6
fScabin_wall	real	+	factor for cabin wall area $f_{cabin-wall}$	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)  
 boom circumference and width used if SET\_Swet not input (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
		+	Aerodynamics	
MODEL_aero	int	+	model (0 none, 1 standard)	1
AFuse	AFuse		standard model	
DoQ_cont	real	+	contingency drag, area $(D/q)_{cont}$	0.
DoQV_cont	real	+	contingency vertical drag, area $(D/q)V_{cont}$	0.

			Derived	
DoQ_fus	real		fuselage drag, area $(D/q)_{fus}$	
DoQV_fus	real		fuselage vertical drag, area $(D/q)_{V_{fus}}$	
DoQ_fit	real		fittings and fixtures drag, area $(D/q)_{fit}$	
DoQ_rb	real		rotor-body interference drag, area $(D/q)_{rb}$	
<hr/>				
			DoQ_cont calculated if total drag fixed (Aircraft FIX_drag); otherwise input	
			DoQV_cont calculated if total download fixed (Aircraft FIX_DL); otherwise input	
<hr/>				
			+ Weight	
Weight	Weight		weight statement (component)	
			+ fuselage group	
MODEL_weight	int		fuselage group model (0 input, 1 AFDD, 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.
WFuse	WFuse		AFFD model	
			+ Technology Factors	
TECH_body	real		basic body $\chi_{basic}$	1.0
TECH_mar	real		body marinization $\chi_{mar}$	1.0
TECH_press	real		pressurization $\chi_{press}$	1.0
TECH_crash	real		body crashworthiness $\chi_{cw}$	1.0
TECH_ftfold	real		tail fold $\chi_{ftfold}$	1.0
TECH_fwfold	real		wing fold $\chi_{fwfold}$	1.0

weight model result multiplied by technology factor and increment added:

$$W_{xx} = TECH_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use } MODEL_{xx}=0 \text{ or } TECH_{xx}=0.$$



**Structure: AFuse**

Variable	Type	Description	Default
		+ Aerodynamics, Standard Model	
AoA_zl	real	+ zero lift angle of attack $\alpha_{zl}$ (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, $(M/q)_0$	0.0
CM0	real	+ moment at zero lift, $C_{M0}$ (based on wetted area and fuselage length, $M/q = SlC_M$ )	0.0
dMoQda	real	+ moment slope, $d(M/q)/d\alpha$ (per rad)	0.0
dCMda	real	+ moment slope, $C_{M\alpha} = dC_M/d\alpha$ (per rad; based on wetted area and fuselage length, $M/q = SlC_M$ )	0.0
SS_zy	real	+ sideslip angle for zero side force $\beta_{zy}$ (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, $(N/q)_0$	0.0
CN0	real	+ moment at zero lift, $C_{N0}$ (based on wetted area and fuselage length, $N/q = SlC_N$ )	0.0
dNoQdb	real	+ moment slope, $d(N/q)/d\beta$ (per rad)	0.0
dCNdb	real	+ moment slope, $C_{N\beta} = dC_N/d\beta$ (per rad; based on wetted area and fuselage length, $N/q = SlC_N$ )	0.0

---

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 $(D/q)_{fit}$	
CD_fit	real	+	coefficient $C_{Dfit}$ (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.
CD_rb_total	real		total rotor-body interference drag, $C_{Drb}$	
		+	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, $D/q = S_{proj}C_D$ )	0.
CDVs	real		$C_{DV}S_{proj}/S_{wet}$	
		+	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 $C_{Dmin}$ (deg)	0.0
Kdrag	real	+	drag increment $K_d$ , $\Delta C_D = C_{D0}K_d \alpha_e ^{X_d}$	0.0
Xdrag	real	+	drag increment $X_d$ , $\Delta C_D = C_{D0}K_d \alpha_e ^{X_d}$	2.

Structure: AFuse

155

MODEL_trans	int	+	transition from forward flight drag to vertical drag	
AoA_tran	real	+	model (1 input transition angle of attack, 2 calculate for quadratic)	1
at	real	+	angle of attack for transition $\alpha_t$ (deg)	25.
Xd	real		angle of attack for transition $\alpha_t$ (deg) (derived)	
			exponent $X_d$ (derived)	

## Chapter 41

**Structure: WFuse**

Variable	Type	Description	Default
		+ Fuselage Group, AFDD Weight Model	
MODEL_body	int	+ model (1 AFDD84 (UNIV), 2 AFDD82 (HELO))	1
fWbody_mar	real	+ body weight for marinization $f_{mar}$ (fraction basic body weight)	0.0
fWbody_press	real	+ body weight for pressurization $f_{press}$ (fraction basic body weight)	0.0
fWbody_crash	real	+ body weight for crashworthiness $f_{cw}$ (fraction body weight)	0.0
fWbody_tfold	real	+ tail fold weight $f_{tfold}$ (fraction tail (UNIV) or body (HELO) weight)	0.0
fWbody_wfold	real	+ wing fold weight $f_{wfold}$ (fraction wing+tip (UNIV) or body+tailfold (HELO) weight)	0.0
KIND_ramp	int	+ rear cargo ramp (0 none, 1 yes)	0
<hr/> <p>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  typically <math>f_{Wbody\_crash} = 0.06</math>  typically <math>f_{Wbody\_tfold} = 0.30</math> (UNIV) or <math>0.05</math> (HELO) for folding tail</p> <hr/>			
WtParam_fuse(8)	real	+ Custom Weight Model + parameters	0.



		+	Weight		
Weight	Weight		weight statement (component)		
		+	alighting gear group		
MODEL_weight	int	+	alighting gear group model (0 input, 1 AFDD, 2 custom)		1
		+	weight increment		
dWLG	real	+	basic landing gear		0.
dWLGret	real	+	retraction		0.
dWLGcrash	real	+	crashworthiness		0.
WGear	WGear		AFDD model		
		+	Technology Factors		
TECH_LG	real	+	basic landing gear $\chi_{LG}$		1.0
TECH_LGret	real	+	retraction $\chi_{LGret}$		1.0
TECH_LGcrash	real	+	crashworthiness $\chi_{LGcw}$		1.0

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = \text{TECH}_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use MODEL}_{xx}=0 \text{ or TECH}_{xx}=0.$$


---

---

**Structure: AGear**

Variable	Type	Description	Default
DoQ	real	+ Drag, Standard Model + drag area extended, $D/q$	

**Structure: WGear**

Variable	Type	Description	Default
		+ Landing Gear Group, AFDD 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 $f_{LGret}$ (fraction basic weight)	0.08
fWLG_crash	real	+ landing gear weight for crashworthiness $f_{LGcw}$ (fraction basic+retraction weight)	0.14
<hr/> only MODEL_LG=fraction uses fWLG_basic typically fWLG_basic = 0.0325 (fraction method) typically fWLG_ret = 0.08, fWLG_crash = 0.14 <hr/>			
WtParam_gear(8)	real	+ Custom Weight Model + parameters	0.



**Structure: Rotor**

Variable	Type	Description	Default
		+ Rotor	
title	c*100	+ title	
notes	c*1000	+ notes	
config	c*32	+ Configuration	'main'
rotorconfig	int	configuration (ROTORCONFIG_main, tail, prop)	
isMainRotor	int	main rotor (0 not)	
isAntiQRotor	int	antitorque rotor (0 not)	
isAuxTRotor	int	auxiliary thrust rotor (0 not)	
isVariableDiam	int	variable diameter rotor (0 not)	
isDuctedFan	int	ducted fan (0 not)	
isReactionDrive	int	reaction drive (0 not)	
twinrotor	int	configuration (ROTORCONFIG_tandem, coaxial, tiltrotor, not_twin)	

---

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', 'ducted fan', 'reaction drive' (keyword = var, duct, react)

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

---

kRotor	int	rotor number	
		+ 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	
rVtip_ref(ngearmax)	real	ratio to state #1	
Omega_ref	real	reference rotational speed (state #1)	
INPUT_gear	int	+ gear ratio input for dependent branch (1 Vtip_ref, 2 gear)	1
gear(ngearmax)	real	+ gear ratio $r = \Omega_{dep}/\Omega_{prim}$ (ratio rpm to rpm of primary rotor)	1.0
		+ Reaction drive	
r_react	real	+ effective radial station of force (fraction Radius)	1.0
prop_react(3)	int	propulsion for reaction drive (group (1 engine, 2 jet), number, model)	

---

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;  $V_{tip-hover} = V_{tip-ref}(1)$   
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; $\geq 2$ 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 ( $V_{tip}/V_{tip-hover}$ )	
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 $P_{RSlimit}$	
fPlimit_rs	real	+	rotor shaft power limit factor	1.
Qlimit_rs	real		rotor shaft torque limit ( $P_{RSlimit}$ at $\Omega_{ref}$ )	

---

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 ×  $P_{DSlimit}$   
 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	
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 $C_W/\sigma$ (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
iSET_thrust	int	+	rotor thrust for disk loading and blade loading (1 from DGW, 2 from Tdesign)	

---

rotor disk loading =  $T/A$ ; aircraft disk loading =  $W_D/A_{ref}$ ,  $A_{ref} = \sum(f_A A)$

$W = f_W W_D$  (main rotor) or  $f_{Thrust} * T_{design}$  (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 =  $T/\text{diskload}$

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  $C_W/\sigma$  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, tail rotor)	'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	

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
			derived	
iSET_geom	int		position (SET_geom_standard, tiltrotor, coaxial, tandem, tailrotor)	
clearance_calc	real		clearance between rotor and fuselage $d_{fus}$	
Hsep_twin	real		horizontal separation $\ell$ (fraction Diameter)	
Vsep_twin	real		vertical separation $s$ (fraction Diameter)	
overlap_twin	real		overlap $o$ (1 – separation/Diameter)	
m_twin	real		overlap area fraction $m$	
		+	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
		+	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)	

---

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

same WingForRotor for otherRotor, first rotor is right

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, first rotor is lower

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, first rotor is front  
 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

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:  $R = \text{Radius in hover and helicopter mode } (V \leq V_{\text{conv-hover}})$

$R = \text{Radius} * f_{\text{Rcruise}}$  in cruise mode ( $V \geq V_{\text{conv-cruise}}$ ); linear with  $V$  in conversion mode

function of speed: use nVdiam, fdiam, Vdiam to calculate  $R$

		+	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 $\sigma_t/\sigma_g$	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 $\theta_L$ (deg, root to tip)	-10.
nprop	int	+	number of radial stations (maximum nrmax)	2
rprop(nrmax)	real	+	radial stations ( $r_{\text{root}}/R$ )	
fchord(nrmax)	real	+	chord distribution $c(r)/c_{\text{ref}}$	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 $\nu$ (per-rev at hover tip speed)	1.04

Structure: Rotor

167

conefreq	real	+	coning natural frequency $\nu$ (0. to use flapfreq)	0.
gamma	real	+	blade Lock number $\gamma$	8.
precone	real	+	precone $\beta_p$ (deg)	0.
delta3	real	+	pitch-flap coupling $\delta_3$ (deg)	0.
		+	aerodynamics	
dclda	real	+	blade section 2D lift-curve slope $a = c_{l\alpha}$ (per-rad)	5.7
tiploss	real	+	tip loss factor $B$ (lift zero from $BR$ to tip)	0.97
xroot	real	+	root cutout ( $r_{root}/R$ )	0.1
fBlockage	real	+	blockage factor (force increment $f_{BT}$ )	0.0

---

SET\_chord: use one of fTWsigma, taper, or fchord(r); others calculated  
 for nonlinear distribution, scale input fchord to unit thrust-weighted chord  
 $f_{TWsigma} = \sigma_{tw}/\sigma_{geom}$ ; for linear taper  $f = c(.75R)/c(.5R) = (1 + 3taper)/(2 + 2taper)$   
 equivalent linear taper =  $c(tip)/c(root) = (2f - 1)/(3 - 2f)$   
 for linear taper  $f_c = c/c_{ref} = 1 + (r - 0.75)4(taper - 1)/(1 + 3taper)$

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

blockage factor: force acting on aircraft is  $(1-fBlockage)*thrust$

---

		+	Geometry (for graphics)	
thick	real	+	blade thickness-to-chord ratio	0.12
			Geometry (derived)	
frotate	real		direction of rotation (1 counter-clockwise, -1 clockwise)	

Arotor	real	rotor area ( $\pi R^2$ )	
chord	real	thrust-weighted chord	
sigma_geom	real	solidity $\sigma = Nc/\pi R$ ; mean geometric chord	
chord_geom	real	mean geometric chord	
AspectRatio	real	aspect ratio, $R/\text{chord\_geom}$	
Ablade	real	thrust-weighted blade area	
KP	real	$\tan(\delta_3)$	
fc(nrmax)	real	chord distribution $f_c = c(r)/c_{\text{ref}}$ (scaled to unit thrust-weighted chord)	
tw(nrmax)	real	twist $\theta_{tw}(r)$ (relative $0.75R$ )	
gamma_calc	real	blade Lock number $\gamma$	
AI_calc	real	autorotation index $KE/P$	
lblade	real	blade moment of inertia $I_{\text{blade}}$	
Kflap	real	flap stiffness $K_{\text{flap}}$ (KIND_hub = hingeless)	
eflap	real	flap hinge offset $e$ (KIND_hub = articulated)	
Kcone	real	cone stiffness $K_{\text{cone}}$ (conefreq input)	
Khub	real	hub moment spring $K_{\text{hub}}$	
		+ 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
dr	real	radial increment $dr = (1 - \text{xroot})/\text{mr}$	
cspsi(mpsimax)	real	$\cos(\psi_j)$ , $\psi_j = j \Delta\psi$ , $j = 1$ to mpsi ( $\Delta\psi = 2\pi/\text{mpsi}$ )	
snspsi(mpsimax)	real	$\sin(\psi_j)$ , $\psi_j = j \Delta\psi$ , $j = 1$ to mpsi ( $\Delta\psi = 2\pi/\text{mpsi}$ )	
		+ 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 $\theta_h$ (deg)	0.
cant_hub	real	+ cant angle $\phi_h$ (deg)	0.



		+	orientation of pivot axes	
dihedral_pivot	real	+	pivot dihedral angle $\phi_p$ (deg)	
pitch_pivot	real	+	pivot pitch angle $\theta_p$ (deg)	
sweep_pivot	real	+	pivot sweep angle $\psi_p$ (deg)	
		+	reference shaft control	
incid_ref	real	+	incidence $i_{ref}$ (deg)	0.
cant_ref	real	+	cant angle $c_{ref}$ (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.
			Derived	
iDirection	int		nominal orientation (1, -1, 2, -2, 3, -3, -3, r2, 1)	
axis_incid	int		axis incidence ( $\pm 123$ )	
axis_cant	int		axis cant ( $\pm 123$ )	
KIND_incid	int		incidence (0 fixed, 1 controlled)	
KIND_cant	int		cant angle (0 fixed, 1 controlled)	
CPF(3,3)	real		pivot axes relative airframe, $C^{PF}$	
CFP(3,3)	real		pivot axes relative airframe, $C^{FP}$	
WCHF(3,3)	real		$WC^{HF}$ ( $C^{SF}$ for reference control)	
CSF(3,3)	real		rotor shaft relative airframe, $C^{SF}$ (zero shaft control)	
<hr/>				
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 = $W_{tip\_total}/nRotorOnWing$ or $w/N_{rotor}$				
$w = W_{gbrs}$ (drive system) or $W_{gbrs} + \sum(W_{ES})$ (drive system and engine system)				
<hr/>				
		+	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 $C_T/\sigma$ )	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; $\geq 2$ 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; $\geq 2$ 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; $\geq 2$ 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; $\geq 2$ 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; $\geq 2$ piecewise linear, maximum nvelmax)	0
cant(nvelmax)	real	+	values	
Vcant(nvelmax)	real	+	speeds (CAS or TAS)	

		+	diameter $f_{\text{diam}}$ (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; $\geq 2$ piecewise linear, maximum nvelmax)	0
fdiam(nvelmax)	real	+	values	
Vdiam(nvelmax)	real	+	speeds (CAS or TAS)	
		+	gear ratio factor $f_{\text{gear}}$ (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 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,  $C_L/\sigma$ , or  $C_T/\sigma$   
 Kprop can be fraction total aircraft drag, propulsive force  $-X$ ,  $-C_X/\sigma$ , or  $-X/q$

---

			+ Rotor Thrust Capability ( $C_T/\sigma$ vs $\mu$ )	
			+ sustained	
nsteady	int	+	number of points (maximum 20)	16
mu_steady(20)	real	+	advance ratio	
CTs_steady(20)	real	+	$C_T/\sigma$	
			+ transient	
ntran	int	+	number of points (maximum 20)	16
mu_tran(20)	real	+	advance ratio	
CTs_tran(20)	real	+	$C_T/\sigma$	

---

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
PRotorInd	PRotorInd		standard model, induced power	
PRotorPro	PRotorPro		standard model, profile power	
PRotorTab	PRotorTab		table model	
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)	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.
IRotor	IRotor		standard model	

---

Kint=0 to suppress interference at component; MODEL\_int=0 for no interference at all  
with transition: interference factors linearly vary from Kint at  $V \leq V_{int\_low}$  to 0 at  $V \geq V_{int\_high}$

---

		+	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 $\theta_{ref}$ , $C^{BS} = Y_{-\theta_{ref}}$	0.0
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 $S_{pylon}$	0.0
kSwet_pylon	real	+	factor, $k = S_{pylon}/(w/N_{rotor})^{2/3}$ (Units_Dscale)	1.0
SET_Sduct	int	+	duct area (1 fixed, input S_duct; 2 scaled, from fLength_duct)	2
S_duct	real	+	area $S_{duct}$	0.0
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 $S_{spin}$	0.0
fSwet_spin	real	+	factor, $k = S_{spin}/A_{spin}$	1.0
fRadius_spin	real	+	spinner radius (fraction rotor radius)	0.
			Derived	
CBS(3,3)	real		pylon axes relative shaft, $C^{BS}$	
CBF(3,3)	real		pylon axes relative airframe, $C^{BF}$ (zero shaft control)	
Radius_spin	real		spinner radius $R_{spin}$	

---

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  $\text{ft}^2/\text{lb}^{2/3}$  or  $\text{m}^2/\text{kg}^{2/3}$

$w = W_{gbrs}$  (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)

$S_{\text{duct}} = (2\pi R)\ell_{\text{duct}}$ ,  $\ell_{\text{duct}} = \text{fLength\_duct} * R$ ; used for drag (wetted area  $2S_{\text{duct}}$ ) and weight

SET\_Sspin, spinner wetted area: (use Swet\_spin) or calculated (from fSwet\_spin)

$A_{\text{spin}} = \pi R_{\text{spin}}^2 = \text{spinner frontal area (from fRadius\_spin} * R)$ ; spinner radius used for drag and weight

---

		+ Drag	
MODEL_drag	int	+ model (0 none, 1 standard)	1
ldrag	real	+ incidence angle for helicopter nominal drag (deg; 0 for not tilt)	0.
DRotor	DRotor	standard model	
		Derived	
DoQC_hub	real	hub cruise drag, area $(D/q)_{\text{hub}}$	
DoQH_hub	real	hub helicopter drag, area $(D/q)_{\text{hub}}$	
DoQV_hub	real	hub vertical drag, area $(D/q)_{\text{hub}}$	
DoQC_pylon	real	pylon cruise drag, area $(D/q)_{\text{pylon}}$	
DoQH_pylon	real	pylon helicopter drag, area $(D/q)_{\text{pylon}}$	
DoQV_pylon	real	pylon vertical drag, area $(D/q)_{\text{pylon}}$	
DoQC_duct	real	duct cruise drag, area $(D/q)_{\text{duct}}$	
DoQH_duct	real	duct helicopter drag, area $(D/q)_{\text{duct}}$	
DoQV_duct	real	duct vertical drag, area $(D/q)_{\text{duct}}$	
DoQ_spin	real	spinner drag, area $(D/q)_{\text{spin}}$	
Swet_rotor	real	total wetted area $S_{\text{wet}}$	

		+	Weight		
Weight	Weight		weight statement (component)		
		+	rotor group (or empennage or propulsion group)		
MODEL_weight	int	+	model (0 input, 1 AFDD, 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.
WRotor	WRotor		AFDD model		
SET_lblade	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 $KE/P = \frac{1}{2}N_{blade}I_{blade}\Omega^2/P$ (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
Wblade	real		blade weight (all blades; required for drive system weight)		
Wtip	real		weight on wing tip (required for tiltrotor wing weight)		
		+	Technology Factors		
TECH_blade	real	+	blade weight $\chi_{blade}$		1.0
TECH_hub	real	+	hub and hinge weight $\chi_{hub}$		1.0
TECH_shaft	real	+	inter-rotor shaft $\chi_{shaft}$		1.0
TECH_spin	real	+	fairing/spinner weight $\chi_{spin}$		1.0
TECH_rfold	real	+	blade fold weight $\chi_{fold}$		1.0
TECH_tr	real	+	tail rotor weight $\chi_{tr}$		1.0
TECH_aux	real	+	auxiliary thrust weight $\chi_{at}$		1.0
TECH_rsupt	real	+	rotor support structure weight $\chi_{supt}$		1.0
TECH_duct	real	+	duct weight $\chi_{duct}$		1.0

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = \text{TECH}_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use MODEL}_{xx}=0 \text{ or TECH}_{xx}=0.$$

blade weight:  $W_{blade} = \chi_{blade} w_{blade} + dW_{blade} + (1 + f) W_{tip} N_{blade}$

SET\_lblade: calculate blade moment of inertia lblade

0 from Lock number gamma, independent of blade weight

1 from blade weight

2 from Lock number gamma, tip weight Wblade\_tip calculated from lblade

3 from autorotation index AI, tip weight Wblade\_tip calculated from lblade

for tail rotor or auxiliary thrust weight model (MODEL\_config = 2 or 3), Wblade\_tip not used and SET\_lblade = 0

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)

---



**Structure: PRotorInd**

Variable	Type	Description	Default
		+ Rotor Induced Power, Standard Energy Performance Method	
MODEL_ind	int	+ model (1 constant, 2 standard)	2
		+ induced velocity factors (ratio to momentum theory induced velocity)	
Ki_hover	real	+ hover $\kappa_{\text{hover}}$	1.12
Ki_climb	real	+ axial climb $\kappa_{\text{climb}}$	1.08
Ki_prop	real	+ axial cruise (propeller) $\kappa_{\text{prop}}$	2.0
Ki_edge	real	+ edgewise flight (helicopter) $\kappa_{\text{edge}}$	2.0
		+ variation with thrust	
CTs_Hind	real	+ $(C_T/\sigma)_{\text{ind}}$ for $\kappa_h$ variation	0.08
kh1	real	+ coefficient $k_{h1}$ for $\kappa_h$	0.
kh2	real	+ coefficient $k_{h2}$ for $\kappa_h$	0.
Xh2	real	+ exponent $X_{h2}$ for $\kappa_h$	2.
CTs_Pind	real	+ $(C_T/\sigma)_{\text{ind}}$ for $\kappa_p$ variation	0.08
kp1	real	+ coefficient $k_{p1}$ for $\kappa_p$	0.
kp2	real	+ coefficient $k_{p2}$ for $\kappa_p$	0.
Xp2	real	+ exponent $k_{p2}$ for $\kappa_p$	2.
		+ variation with shaft angle	
kpa	real	+ coefficient $k_{h\alpha}$ for $\kappa_p$	0.
Xpa	real	+ exponent $X_{h\alpha}$ for $\kappa_p$	2.
Maxial	real	+ constant $M_{\text{axial}}$ in transition from hover to climb	1.176
Xaxial	real	+ exponent $X_{\text{axial}}$ in transition from hover to climb	0.65
		+ variation with axial velocity	
mu_prop	real	+ advance ratio $\mu_{z\text{prop}}$ 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	
mu_edge	real	+	advance ratio $\mu_{\text{edge}}$ for $K_{i\_edge}$	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 $f_{\text{off}}$	0.
ko2	real	+	factor $k_{o2}$ for $f_{\text{off}}$	8.
Ki_min	real	+	minimum $\kappa_{\text{min}}$	1.
Ki_max	real	+	maximum $\kappa_{\text{max}}$	10.
fedge	real		edgewise scale factor $S$	
fprop	real		axial scale factor $S$	

---

MODEL\_ind=constant uses only Ki\_hover, Ki\_prop, Ki\_edge  
nonzero values of Ki in FltState supersede calculated value

---

		+	Momentum theory	
MODEL_GE	int	+	ground effect (0 none, 1 Cheeseman and Bennett, 2 BE Cheeseman and Bennett, 3 Law, 4 Hayden, 5 Zbrozek)	3
Cge	real	+	effective height correction $C_g$	1.
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.

---

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 $f_{V_x}$ (fan edgewise velocity/free stream velocity)	1.
fDuctVz	real	+ velocity ratio $f_{V_z}$ (fan axial velocity/free stream velocity)	1.
<hr/>			
ducted fan model used only if config='duct'			
<hr/>			
		+ Twin rotors	
MODEL_twin	c*12	+ model (based on config, none, side-by-side, coaxial, tandem)	'config'
Kh_twin	real	+ ideal induced velocity correction $\kappa_{twin}$ for hover	1.00
Kf_twin	real	+ ideal induced velocity correction $\kappa_{twin}$ for forward flight	0.85
Cind_twin	real	+ constant $C$ in hover to forward flight transition	1.0
A_coaxial	real	+ coaxial rotor nonuniform disk loading factor $\bar{\alpha}$	1.05
		Derived	
iMODEL_twin	int	model (MODEL_twin_none, sidebyside, coaxial, tandem)	
xh	real	thrust factor $x_h$ , hover	
xf1	real	thrust factor $x_{f1}$ , forward flight, this rotor	
xf2	real	thrust factor $x_{f2}$ , forward flight, other rotor	
<hr/>			
MODEL_twin: 'config', 'none', 'side-by-side' or 'tiltrotor', 'coaxial', or 'tandem'			
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			
<hr/>			

**Structure: PRotorPro**

Variable	Type	Description	Default
		+ Rotor Profile Power, Standard Energy Performance Method	
		+ Technology factor	
TECH_drag	real	+ profile power $\chi$	1.0
Re_ref	real	+ Reference Reynolds number $Re_{ref}$ (0. for no correction)	0.0
MODEL_basic	int	+ Basic model $c_{dbasic}$ (1 array, 2 equation)	2
		+ array ( $c_d$ vs thrust-weighted $C_T/\sigma$ )	
ncd	int	+ number of points (maximum 24)	24
CTs_cd(24)	real	+ blade loading	
cd(24)	real	+ drag coefficient	
		+ equation	
CTs_Dmin	real	+ $(C_T/\sigma)_{Dmin}$ for minimum profile drag ( $\Delta =  C_T/\sigma - (C_T/\sigma)_{Dmin} $ )	0.07
d0_hel	real	+ coefficient $d_{0hel}$ in drag, $c_{dh} = d_{0hel} + d_{1hel}\Delta + d_{2hel}\Delta^2 + \Delta c_{dsep}$ (hover/edgewise)	0.009
d1_hel	real	+ coefficient $d_{1hel}$ in drag (hover/edgewise)	0.0
d2_hel	real	+ coefficient $d_{2hel}$ in drag (hover/edgewise)	0.5
d0_prop	real	+ coefficient $d_{0prop}$ in drag, $c_{dp} = d_{0prop} + d_{1prop}\Delta + d_{2prop}\Delta^2 + \Delta c_{dsep}$ (axial)	0.009
d1_prop	real	+ coefficient $d_{1prop}$ in drag (axial)	0.0
d2_prop	real	+ coefficient $d_{2prop}$ 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	+ $(C_T/\sigma)_{sep}$ for separation ( $\Delta c_{dsep} = d_{sep}( C_T/\sigma  - (C_T/\sigma)_{sep})^{X_{sep}}$ )	0.07
dsep	real	+ factor $d_{sep}$ in drag increment	4.0
Xsep	real	+ exponent $X_{sep}$ in drag increment	3.0
df1	real	+ variation with edgewise velocity, coefficient $d_{f1}$	0.0
df2	real	+ variation with edgewise velocity, coefficient $d_{f2}$	0.0
Xf	real	+ variation with edgewise velocity, exponent $X_f$	2.

---

default array (cd(1)=0):  $C_T/\sigma = 0.0$  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 $c_{dstall}$ (0 none)	1
		+	$C_T/\sigma$ at stall ( $\Delta_s =  C_T/\sigma  - (f_s/f_\alpha f_{off})(C_T/\sigma)_s$ , $\Delta c_d = d_{s1}\Delta_s^{X_{s1}} + d_{s2}\Delta_s^{X_{s2}}$ )	
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 $c_{dcomp}$ (0 none, 1 drag divergence, 2 similarity)	1
		+	similarity model	
fSim	real	+	factor $f$	1.0
thick_tip	real	+	blade tip thickness-to-chord ratio $\tau$	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 ( $M_{dd} = M_{dd0} - M_{ddcl} c_\ell$ )	
Mdd0	real	+	$M_{dd0}$ at zero lift	0.88
Mddcl	real	+	derivative with lift $\kappa = \partial M_{dd} / \partial c_\ell$	0.16

**Structure: PRotorTab**

Variable	Type	Description	Default
		+ 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	' '
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 $c_{dmean}$ , 2 table $c_{dmean}^F = 8C_{Po}/\sigma$ )	1
nVar_pro	int	+ number independent variables (1 to 3)	0
var_pro(3)	c*12	+ variables	' '
		+ independent variables	
nmu	int	+ number of $\mu_{HP}$ or $\mu_{TPP}$ values (maximum ntablemax)	0
nmuz	int	+ number of $\mu_{zHP}$ or $\mu_{zTPP}$ values (maximum ntablemax)	0
nalpha	int	+ number of $\alpha_{HP}$ or $\alpha_{TPP}$ values (maximum ntablemax)	0
nCTs	int	+ number of $C_T/\sigma$ values (maximum ntablemax)	0
nMx	int	+ number of lift offset $M_x/TR$ values (maximum ntablemax)	0
nMat	int	+ number of $M_{at}$ values (maximum ntablemax)	0
mu(ntablemax)	real	+ advance ratio $\mu_{HP}$ or $\mu_{TPP}$	
muz(ntablemax)	real	+ axial advance ratio $\mu_{zHP}$ or $\mu_{zTPP}$	
alpha(ntablemax)	real	+ shaft angle-of-attack $\alpha_{HP}$ or $\alpha_{TPP}$	
CTs(ntablemax)	real	+ blade loading $C_T/\sigma$	
Mx(ntablemax)	real	+ lift offset $M_x/TR$	
Mat(ntablemax)	real	+ advancing tip Mach number $M_{at}$	
		+ table	
Ki(ntablemax,ntablemax,ntablemax)	real	+ induced power factor $\kappa$	
cdo(ntablemax,ntablemax,ntablemax)	real	+ profile power mean $c_d$	

ivar_ind(3)	int	induced power variables (tablevar_mu, muz, alpha, muTPP, muzTPP, alphaTPP, CTs, Mx, Mat)
nvi(3)	int	number of variables
vi(ntablemax,3)	real	variable values
ivar_pro(3)	int	profile power variables (tablevar_mu, muz, alpha, muTPP, muzTPP, alphaTPP, CTs, Mx, Mat)
nvo(3)	int	number of variables
vo(ntablemax,3)	real	variable values

## Derived

---

independent variables: var\_ind and var\_pro

'mu', 'muHP': edgewise advance ratio  $\mu$  (hub plane)

'muz', 'muzHP': axial velocity ratio  $\mu_z$  (hub plane)

'alpha', 'alphaHP': shaft angle-of-attack  $\alpha = \tan^{-1}(\mu_z/\mu)$  (hub plane)

'muTPP': edgewise advance ratio  $\mu$  (tip-path plane)

'muzTPP': axial velocity ratio  $\mu_z$  (tip-path plane)

'alphaTPP': shaft angle-of-attack  $\alpha = \tan^{-1}(\mu_z/\mu)$  (tip-path plane)

'CTs', 'CT/s': blade loading  $C_T/\sigma$

'Mx', 'offset': lift offset  $M_x/TR$

'Mat': advancing tip Mach number  $M_{at}$

arrays of independent variables used for both induced power and profile power

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

---



**Structure: DRotor**

Variable	Type	Description	Default
		+ 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 $(D/q)_{\text{hub}}$	
CD_hub	real	+ coefficient $C_{D\text{hub}}$ (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 $(D/q)_{\text{pylon}}$	
CD_pylon	real	+ coefficient $C_{D\text{pylon}}$ (based on pylon wetted area, $D/q = SC_D$ )	0.0
SET_Dduct	int	+ duct drag specification (1 fixed, $D/q$ ; 2 scaled, $C_D$ )	2
DoQ_duct	real	+ area $(D/q)_{\text{duct}}$	
CD_duct	real	+ coefficient $C_{D\text{duct}}$ (based on duct wetted area, $D/q = SC_D$ )	0.0
SET_Dspin	int	+ spinner drag specification (1 fixed, $D/q$ ; 2 scaled, $C_D$ )	1
DoQ_spin	real	+ area $(D/q)_{\text{spin}}$	0.0
CD_spin	real	+ coefficient $C_{D\text{spin}}$ (based on spinner wetted area, $D/q = SC_D$ )	0.0
		+ vertical drag	
SET_Vhub	int	+ hub drag specification (1 fixed, $D/q$ ; 2 scaled, $C_D$ )	2
DoQV_hub	real	+ area $(D/q)_{V\text{hub}}$	
CDV_hub	real	+ coefficient $C_{DV\text{hub}}$ (based on rotor area, $D/q = SC_D$ )	0.0
SET_Vpylon	int	+ pylon drag specification (1 fixed, $D/q$ ; 2 scaled, $C_D$ )	2
DoQV_pylon	real	+ area $(D/q)_{V\text{pylon}}$	
CDV_pylon	real	+ coefficient $C_{DV\text{pylon}}$ (based on pylon wetted area, $D/q = SC_D$ )	0.0
SET_Vduct	int	+ duct drag specification (1 fixed, $D/q$ ; 2 scaled, $C_D$ )	2
DoQV_duct	real	+ area $(D/q)_{V\text{duct}}$	
CDV_duct	real	+ coefficient $C_{DV\text{duct}}$ (based on duct wetted area, $D/q = SC_D$ )	0.0
		+ transition from forward flight drag to vertical drag	
MODEL_Dhub	int	+ hub drag model (1 general, 2 quadratic)	2
MODEL_Dpylon	int	+ pylon drag model (1 general, 2 quadratic)	2

MODEL_Dduct	int	+	duct drag model (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.
Xh	real		hub drag, transition exponent $X_d$ (derived)	
Xp	real		pylon drag, transition exponent $X_d$ (derived)	
Xd	real		duct drag, transition exponent $X_d$ (derived)	

---

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  $\text{ft}^2/\text{klb}^{2/3}$  or  $\text{m}^2/\text{Mg}^{2/3}$ ;  $\text{ft}^2/\text{lb}^{1/2}$  or  $\text{m}^2/\text{kg}^{1/2}$   
 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  $f_{\text{Thrust}} * T_{\text{design}}$  (antitorque or aux thrust rotor)

---

**Structure: IRotor**

Variable	Type	Description	Default
		+ 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 $K_{int}$ (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

---

$K_{int}=0$  to suppress interference at component; MODEL\_int=0 for no interference at all  
interference factor linearly transition from  $K_{int}$  at  $V \leq V_{int\_low}$  to 0 at  $V \geq V_{int\_high}$

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= $\infty$

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 $C_{int}$ (0. for no interference)	0.

---

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

---

**Structure: WRotor**

Variable	Type	Description	Default
		+ Rotor Group, AFDD 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)	1
MODEL_Whub	int	+ hub and hinge weight model (1 AFDD82, 2 AFDD00, 3 lift offset)	1
MODEL_Wshaft	int	+ inter-rotor shaft weight (from lift offset; 0 not included)	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)	1
		+ first flapwise natural frequency $\nu$ (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
		+ auxiliary thrust	
MODEL_aux	int	+ auxiliary thrust weight model (1 AFDD10, 2 AFDD82)	1
thrust_aux	real	+ design maximum thrust $T_{at}$ (for AFDD82 model)	0.
power_aux	real	+ design maximum power $P_{at}$ (for AFDD10 model)	0.
material_aux	real	+ material factor $f_m$ (for AFDD10 model)	1.
fWfold	real	+ blade fold weight $f_{fold}$ (fraction total blade weight)	0.
fWsupt	real	+ rotor support structure weight (fraction maximum takeoff weight)	0.
Uduct	real	+ duct weight per area $U_{duct}$ (lb/ft <sup>2</sup> or kg/m <sup>2</sup> )	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=0`, use design maximum thrust of rotor from sizing task

if `power_aux=0`, use 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  $\Omega_{prop}$  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

---

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

**Structure: Wing**

Variable	Type	Description	Default
		+ Wing	
title	c*100	+ title	
notes	c*1000	+ notes	
kWing	int	wing number	
		+ 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

---

Structure: Wing

192

		+ 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 $\tau_w$	.23
fWidth_box	real	+ wing torque box chord $w_{tb}$ (fraction wing chord)	0.45

---

RotorOnWing required for SET\_wing = 'width' or 'hub'; MODEL\_wing = tiltrotor; SET\_Vdrag = airfoil  $c_{d90}$

RotorOnPanel required for SET\_panel = '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.
taper	real	+ taper ratio	
sweep	real	+ sweep (+ aft, deg)	
dihedral	real	+ dihedral (+ up, deg)	
MAC	real	+ mean aerodynamic chord $\bar{c}_A$	
xAC	real	+ mean aerodynamic center chordwise offset from root aero center $\bar{x}_A$ (+ aft)	
zAC	real	+ mean aerodynamic center vertical offset from root aero center $\bar{z}_A$ (+ up)	



		+	Geometry	
loc_wing	Location	+	aerodynamic center location	
nPanel	int	+	number of wing panels (maximum npanelmax)	1
KIND_AOffset	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.
lambdal(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 $\Lambda_p$ (deg, + aft)	0.
dihedral_panel(npanelmax)	real	+	dihedral $\delta_p$ (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

---

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

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

#### Derived

iSET_panel_span(npanelmax)	int	span (SET_panel_span, bratio, free)
iSET_panel_edge(npanelmax)	int	edge (SET_panel_edge, station, radius, width, hub, adjust)
iSET_panel_area(npanelmax)	int	area (SET_panel_area, Sratio, chord, cratio, taper, free)
kind_area	int	kind area and chord solution (1 tapered panels, 2 free panel)
chordI(npanelmax)	real	inboard chord $c_{Ip}$

chordO(npanelmax)	real	outboard chord $c_{Op}$	
eAC_aileron(npanelmax)	real	aileron aerodynamic center lateral position $y$ (from centerline, fraction wing semispan)	
rArea_flap(npanelmax)	real	flap area/panel area	
rArea_flaperon(npanelmax)	real	flaperon-aileron area/panel area	
Ktef_flap(4,npanelmax)	real	trailing edge flap factors ( $L_f, X_f, M_f, D_f$ )	
Ktef_flaperon(4,npanelmax)	real	trailing edge flap factors ( $L_f, X_f, M_f, D_f$ )	
rArea_Wflap	real	total flap area/wing area	
rArea_Wflaperon	real	total flaperon-aileron area/wing area	
isConsistent	int	consistent geometry (0 if calculated geometry not consistent)	
		+ 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
areaX	real	extension area $S_X$ (both sides)	
spanX	real	extension span $b_X$ (one side)	
areal	real	inboard area ( $S - S_X$ )	
spanl	real	inboard span ( $b - 2b_X$ )	
area_flapl	real	inboard flap area	
area_flaperonl	real	inboard flaperon-aileron area	
AspectRatiol	real	inboard wing aspect ratio	
sweepl	real	inboard wing sweep	
taperl	real	inboard wing taper	
		+ 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 $\delta_{Fp}$		
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; $\geq 2$ piecewise linear, maximum nvelmax)		0
flap(nvelmax,npanelmax)	real	+	values		
Vflap(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)		
		+	flaperon $\delta_{fp}$		
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; $\geq 2$ piecewise linear, maximum nvelmax)		0
flaperon(nvelmax,npanelmax)	real	+	values		
Vflaperon(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)		
		+	aileron $\delta_{ap}$		
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; $\geq 2$ 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; $\geq 2$ piecewise linear, maximum nvelmax)		0
incid(nvelmax,npanelmax)	real	+	values		
Vincid(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; $\geq 2$ 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
ldrag	real	+	incidence angle $i$ for helicopter nominal drag (deg; 0 for not tilt)	0.
AWing	AWing		standard model	
			Derived	
DoQC_wing	real		wing cruise drag, area $(D/q)_{wing}$	
DoQH_wing	real		wing helicopter drag, area $(D/q)_{wing}$	
DoQV_wing	real		wing vertical drag, area $(D/q)_{wing}$	
DoQ_wb	real		wing-body interference drag, area $(D/q)_{wb}$	
Swet	real		total wetted area $S_{wet}$	

		+	Weight		
Weight	Weight		weight statement (component)		
		+	wing group		
MODEL_weight	int	+	model (0 input, 1 AFDD, 2 custom)		1
		+	weight increment		
dWprim	real	+	wing primary structure		0.0
dWext	real	+	wing extension		0.0
dWfair	real	+	fairing		0.0
dWfit	real	+	fittings		0.0
dWflap	real	+	flaps and control surfaces		0.0
dWwfold	real	+	wing fold		0.0
dWefold	real	+	wing extension fold		0.0
WWing	WWing		AFDD model (except tiltrotor)		
WWingTR	WWingTR		AFDD tiltrotor model		
		+	tiltrotor model		
xWtip	real	+	increment for weight on wing tips		0.0
Wwing_total	real		wing weight		
Wwing_ext	real		wing extension weight		
Wwing_kit	real		wing kit weight		
Wtip_total	real		weight on wing tips		
		+	Technology Factors		
TECH_prim	real	+	wing primary structure (torque box) weight $\chi_{prim}$		1.0
TECH_ext	real	+	wing extension weight $\chi_{ext}$		1.0
TECH_fair	real	+	fairing weight $\chi_{fair}$		1.0
TECH_fit	real	+	fittings weight $\chi_{fit}$		1.0
TECH_flap	real	+	flaps and control surfaces weight $\chi_{flap}$		1.0
TECH_wfold	real	+	wing fold weight $\chi_{fold}$		1.0
TECH_efold	real	+	wing extension fold weight $\chi_{efold}$		1.0

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = TECH_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use } MODEL_{xx}=0 \text{ or } TECH_{xx}=0.$$

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  
xWtip adjusts Wtip\_total, without changing weight statements  
negative increment required when engine and transmission not at tip location with rotor

---

**Structure: AWing**

Variable	Type	Description	Default
		+ Wing Aerodynamics, Standard Model	
AoA_zl	real	+ zero lift angle of attack $\alpha_{zl}$ (deg)	0.
CLmax	real	+ maximum lift coefficient $C_{Lmax}$	1.5
		+ 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 $\tau$	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 $C_{L0}$	0.0
dCLda3D	real	+ 3D lift curve slope $C_{L\alpha}$ (derived)	
fDind	real	+ $1/(\pi e AR)$	
AoA_max	real	+ $\alpha_{max} = C_{Lmax} / (dC_L/d\alpha_{3D})$ (deg)	
eta0	real	+ control effectiveness factor $\eta_0, \eta_0 - \eta_1  \delta $	0.85
eta1	real	+ control effectiveness factor $\eta_1, \eta_0 - \eta_1  \delta $	0.43
		+ 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



---

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) $\Delta C_D = C_{D0} K_d  \alpha_e ^{X_d}$	2
AoA_Dmin	real	+	angle of attack for wing minimum drag $\alpha_{Dmin}$ (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) $\Delta C_D = C_{D0} K_s ( \alpha_e  - \alpha_s)^{X_s}$	3
AoA_sep	real	+	angle of attack for separation $\alpha_s$ (deg)	10.
Ksep	real	+	drag increment $K_s$	0.
Xsep	real	+	drag increment $X_s$	2.
Xd	real		drag exponent $X_d$ (derived)	
Xs	real		drag exponent $X_s$ (derived)	
		+	transition from forward flight drag to vertical drag	
AoA_tran	real	+	angle of attack for transition $\alpha_t$ (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 $K_{int}$ at other wings (0. for no interference)	0.
		+	interference power factor $K_{int}$ at rotors (0. for no interference)	
Kintn_rotor(nrotormax)	real	+	normal (helicopter)	0.
Kintp_rotor(nrotormax)	real	+	inplane (propeller)	0.

---

for tandem wings, typically

$K_{int\_wing}(aftwing)=2.$  for front-on-aft interference

$K_{int\_wing}(frontwing)=0.$  for aft-on-front interference

for biplane wings, typically  $K_{int\_wing}(otherwing)=0.7$

with mutual interference (as for biplane), require trim or other iteration for convergence

interference power: inplane (propeller) factor  $K_{intp\_rotor}$  negative for favorable

---

**Structure: WWing**

Variable	Type	Description	Default
		+ Wing Group, AFDD Weight Model	
MODEL_wing	int	+ model (1 area, 2 parametric, 3 tiltrotor)	2
		+ parametric method	
bFold	real	+ fraction wing span that folds $b_{\text{fold}}$ (0 to 1)	0.0
		+ area method	
Uprim	real	+ weight per area $U_{\text{prim}}$ , wing primary structure (lb/ft <sup>2</sup> or kg/m <sup>2</sup> )	5.
Uext	real	+ weight per area $U_{\text{ext}}$ , wing extension (lb/ft <sup>2</sup> or kg/m <sup>2</sup> )	3.
		+ weight factors (fraction total wing weight)	
fWfair	real	+ fairing $f_{\text{fair}}$	0.10
fWfit	real	+ fittings $f_{\text{fit}}$	0.12
fWflap	real	+ flaps and control surfaces $f_{\text{flap}}$	0.10
fWfold	real	+ wing fold $f_{\text{fold}}$	0.0
fWefold	real	+ wing extension fold $f_{\text{efold}}$ (fraction wing extension weight)	0.0
		+ Custom Weight Model	
WtParam_wing(8)	real	+ parameters	0.

**Structure: WWingTR**

Variable	Type	Description	Default
		+ Wing Group, AFDD Tiltrotor Weight Model	
		+ jump takeoff condition	
CTs_jump	real	+ rotor maximum blade loading $C_T/\sigma$	0.20
n_jump	real	+ load factor $n_{\text{jump}}$ at SDGW	2.0
Vtip_jump	real	+ rotor tip speed (0. to use hover $V_{\text{tip}}$ )	750.0
thickTR	real	+ wing airfoil thickness-to-chord ratio $\tau_w$	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 $w_{\text{attach}}/w_{\text{fus}}$	1.
wAttach	real	+ width $w_{\text{attach}}$ (ft or m)	0.
fRG_pylon	real	+ pylon radius of gyration $r_{\text{pylon}}/R$ (fraction rotor radius)	0.30
		+ wing mode frequencies (per rev, fraction rotor speed)	
freq_beam	real	+ beam bending frequency $\omega_B$	0.5
freq_chord	real	+ chord bending frequency $\omega_C$	0.8
freq_tors	real	+ torsion frequency $\omega_T$	0.9
SET_refrpm	int	+ reference rotor speed (0 from input Vtip_freq, 1 hover $V_{\text{tip}}$ , 2 cruise $V_{\text{tip}}$ )	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 <sup>2</sup> , in/in, lb/in <sup>3</sup> ; or N/m <sup>2</sup> , m/m, kg/m <sup>3</sup> )	
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 $\epsilon_U$	0.01
StrainU_box	real	+	torque box ultimate strain allowable $\epsilon_U$	0.01
density_spar	real	+	density spar cap $\rho_{sp}$	0.06
density_box	real	+	density torque box $\rho_{tb}$	0.06
		+	weight per area (lb/ft <sup>2</sup> or kg/m <sup>2</sup> )	
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)	0.0
fWefoldTR	real	+	wing extension fold $f_{efold}$ (fraction wing extension weight)	0.0
<hr/>				
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 $V_{tip\_ref}(1)$ , cruise $V_{tip\_cruise}$				
thickTR only used for tiltrotor wing weight				
SET_Attach: attachment width used for both torsion stiffness and fairing area				
<hr/>				
WtParam_wingtr(8)	real	+	Custom Weight Model parameters	0.

**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
isHortail	int	horizontal tail (0 vertical)	
isVtail	int	V-tail (0 not)	
kTail	int	tail number	
		+ 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)	
taper	real	+	taper ratio	1.0
sweep	real	+	sweep (+ aft, deg)	0.
dihedral	real	+	dihedral (deg)	0.
thick	real	+	thickness ratio	.12
			Derived	
iSet_tail_area	int		area (SET_tail_area, vol)	
iSet_tail_len	int		length (SET_tail_span, AR, chord)	
Length_tail	real		tail length $\ell$	
rArea_control	real		control surface area/tail area	
Ktef_cont(4)	real		trailing edge flap factors ( $L_f, X_f, M_f, D_f$ )	
CBF(3,3)	real		tail axes relative airframe, $C^{BF}$	
areaVtail	real		V-tail area $S_V$	
spanVtail	real		V-tail span $b_V$	
AspectRatioVtail	real		V-tail aspect ratio	
		+	Geometry	
loc_tail	Location	+	aerodynamic center location	
cant	real	+	cant angle $\phi$ (deg)	0.0
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 $\delta_e$ or rudder $\delta_r$	
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; $\geq 2$ piecewise linear, maximum nvelmax)	0
cont(nvelmax)	real	+	values	
Vcont(nvelmax)	real	+	speeds (CAS or TAS)	

Structure: Tail

208

		+	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; $\geq 2$ 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
ATail	ATail		standard model	
			Derived	
DoQ_tail	real		tail drag, area $(D/q)_{tail}$	
DoQV_tail	real		tail vertical drag, area $(D/q)_{Vtail}$	
Swet	real		total wetted area	
		+	Weight	
Weight	Weight		weight statement (component)	
		+	tail (empennage group)	
MODEL_weight	int	+	model (0 input, 1 AFDD, 2 custom)	1
		+	weight increment	
dWtail	real	+	basic	0.0



Structure: Tail

209

dWfold	real	+	fold	0.0
WTail	WTail		AFDD model	
Vdive	real	+	design dive speed $V_{dive}$ (TAS)	200.
Wtail_total	real		tail weight	
		+	Technology Factors	
TECH_tail	real	+	tail weight $\chi_{ht}$ or $\chi_{vt}$	1.0
TECH_tfold	real	+	fold weight $\chi_{fold}$	1.0

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = \text{TECH}_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use } \text{MODEL}_{xx}=0 \text{ or } \text{TECH}_{xx}=0.$$

dive speed:  $V_{max} = \text{SLS max speed}, V_{dive} = 1.25V_{max}$

---

**Structure: ATail**

Variable	Type	Description	Default
		+ Tail Aerodynamics, Standard Model	
AoA_zl	real	+ zero lift angle of attack $\alpha_{zl}$ (deg)	0.
CLmax	real	+ maximum lift coefficient $C_{Lmax}$	1.
		+ 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 $\tau$	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 $C_{L0}$	0.0
dCLda3D	real	+ 3D lift curve slope $C_{L\alpha}$ (derived)	
fDind	real	+ $1/(\pi e AR)$	
AoA_max	real	+ $\alpha_{max} = C_{Lmax} / (dC_L/d\alpha_{3D})$ (deg)	
eta0	real	+ control effectiveness factor $\eta_0, \eta_0 - \eta_1  \delta $	0.85
eta1	real	+ control effectiveness factor $\eta_1, \eta_0 - \eta_1  \delta $	0.43
		+ 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.

---

SET\_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

---

Structure: ATail

211

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

**Structure: WTail**

Variable	Type	Description	Default
		+ Tail, AFDD 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)	1
		+ vertical tail	
MODEL_Vtail	int	+ model (1 helicopter or compound, 2 tiltrotor or tiltwing, 3 area)	1
place_AntiQ	int	+ antitorque placement (0 none, 1 on tail boom, 2 on vertical tail)	1
		+ area method	
Utail	real	+ weight per area $U_{tail}$ (lb/ft <sup>2</sup> or kg/m <sup>2</sup> )	3.
fTfold	real	+ fold weight factor $f_{fold}$ (fraction total tail weight excluding fold)	0.0
		+ Custom Weight Model	
WtParam_tail(8)	real	+ parameters	0.

**Structure: FuelTank**

Variable	Type	Description	Default
		+ Fuel Tank System	
title	c*100	+ title	
notes	c*1000	+ notes	
kTank	int	+ tank number	
		+ Configuration	
SET_burn	int	+ fuel quantity stored and used (1 weight, 2 energy)	1
		+ fuel weight properties	
fuel_density	real	+ fuel weight per volume $\rho_{\text{fuel}}$ (lb/gallon or kg/liter)	6.5
specific_energy	real	+ fuel energy per weight $e_{\text{fuel}}$ (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 $W_{\text{fuel-cap}}$ (weight, lb or kg)	
Efuel_cap	real	+ fuel capacity $E_{\text{fuel-cap}}$ (energy, MJ)	
fFuel_cap	real	+ ratio capacity to mission fuel $f_{\text{fuel-cap}}$	1.0
IDENT_battery	c*16	+ battery identification	' '

---

store and use weight: energy calculated from weight

use Wfuel\_cap, Waux\_cap, fuel\_density, specific\_energy, fFuelWing; fWtank, fWauxtank, other weight parameters

store and use energy: fuel weight zero

use Efuel\_cap, Eaux\_cap, IDENT\_battery; eWtank, eWauxtank, energy\_density, other weight parameters

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(fFuelCap\*(maximum mission fuel), (maximum mission fuel)+(reserve fuel))

'miss+power' = calculate from mission fuel used and mission battery discharge power

battery identification: energy storage only, match ident of BatteryModel

---

		+	Geometry (for graphics)		
place	int	+	placement (1 internal, 2 sponson, 3 wing, 4 combination)		1
		+	Auxiliary Fuel Tank		
Mauxtanksize	int	+	number of auxiliary tank sizes (minimum 1, maximum nauxtankmax)		1
Waux_cap(nauxtankmax)	real	+	fuel capacity $W_{aux-cap}$ (weight)		1000.
Eaux_cap(nauxtankmax)	real	+	fuel capacity $E_{aux-cap}$ (energy)		20000.
fWauxtank(nauxtankmax)	real	+	tank weight $f_{auxtank}$ (fraction auxiliary fuel weight)		0.
eWauxtank(nauxtankmax)	real	+	tank weight $e_{auxtank}$ (MJ/kg)		0.
DoQ_auxtank(nauxtankmax)	real	+	drag $(D/q)_{auxtank}$ (each tank)		
loc_auxtank(nauxtankmax)	Location	+	location		
			Derived		
Vfuel_cap	real		fuel capacity $V_{fuel-cap}$ (volume)		
Wfuel_wing	real		wing fuel capacity $W_{fuel-wing}$		
rWfuel_wing	real		wing fuel capacity (fraction Wfuel_cap)		
ncomp_in_tank	int		number of components in fuel tank system		
kBatteryModel	int		battery identification (BatteryModel, from IDENT_battery)		
specific_power	real		specific power $\pi_{tank} = x_{mbd}e_{tank}/3.6$ (kW/kg)		
		+	Weight		
Weight	Weight		weight statement (component, not including auxiliary tanks)		
		+	fuel system (propulsion group)		
MODEL_weight	int	+	model (0 input, 1 AFDD, 2 custom)		1
		+	weight increment		
dWtank	real	+	tanks and support; battery management system		0.
dWplumb	real	+	plumbing; power distribution (wiring)		0.
WTank	WTank		AFDD model		
Neng	int		number of main engines		
fuelflow	real		total fuel flow $F$ at DGW takeoff conditions (lb/hr or kg/hr)		

		+	Technology Factors	
TECH_tank	real	+	fuel tank weight $\chi_{\text{tank}}$	1.0
TECH_plumb	real	+	plumbing weight $\chi_{\text{plumb}}$	1.0

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = \text{TECH}_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use } \text{MODEL}_{xx}=0 \text{ or } \text{TECH}_{xx}=0.$$

---

**Structure: WTank**

Variable	Type	Description	Default
		+ Fuel System, AFDD Weight Model	
		+ weight storage	
		+ fuel tank	
MODEL_tank	int	+ model (1 fraction, 2 parametric)	2
fWtank	real	+ tank weight $f_{\text{tank}}$ (fraction fuel capacity weight)	
ntank_int	int	+ number of internal tanks $N_{\text{int}}$	4
Ktoler	real	+ ballistic tolerance factor $f_{bt}$ (1.0 to 2.5)	2.5
KIND_crash	int	+ survivability (1 baseline, 2 UTTAS/AAH level of survivability)	2
		+ plumbing	
nplumb	int	+ total number of fuel tanks (internal and auxiliary) for plumbing $N_{\text{plumb}}$	4
K0_plumb	real	+ weight increment $K_{0\text{plumb}}$	150.
K1_plumb	real	+ weight factor $K_{1\text{plumb}}$	2.0
		+ energy storage	
eWtank	real	+ tank weight $e_{\text{tank}}$ (MJ/kg)	
energy_density	real	+ tank volume density $\rho_{\text{tank}}$ (MJ/liter)	
fBMS	real	+ battery management system (fraction basic tank weight)	0.2
fwire	real	+ power distribution (wiring) weight (fraction basic tank weight)	0.2
<p>MODEL_tank: fraction method uses fWtank; parametric method uses ntank_int, Ktoler, KIND_crash</p> <p>K1_plumb is a crashworthiness and survivability factor; typically K1_plumb = 2.</p> <p>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</p>			
		+ Custom Weight Model	
WtParam_tank(8)	real	+ parameters	0.



## Chapter 61

**Structure: Propulsion**

Variable	Type	Description	Default
		+ Propulsion Group	
title	c*100	+ title	
notes	c*1000	+ notes	
<hr/> <p>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)</p> <hr/>			
kPropulsion	int	propulsion group number	
Specification			
kRotor_prim	int	primary rotor	
rotor_in_group(nrotormax)	int	rotors in group (0 no, 1 main rotor, 2 other)	
nRotor	int	number of rotors in group	
nRotor_main	int	number of main rotors	
kEngine_prim	int	primary engine group	
engine_in_group(nengmax)	int	engine groups in propulsion group (0 no, 1 only produce power, 2 can consume power)	
nEngineGroup	int	number of engine groups	
firstEngineGroup	int	first engine group	
canConsumePower	int	engine group generator or compressor, can consume shaft power (0 only produce power)	
+ 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  $V_{tip}$ ), 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  $f_{gear}$  (control) included  
when evaluate rotational speed of dependent rotors and engines

---

		+	Transmission losses	
MODEL_Xloss	int	+	model (1 fraction component power required; 2 with function drive shaft limit)	2
fPloss_xmsn	real	+	gear box loss $\ell_{xmsn}$ (fraction total component power required)	0.04
Ploss_windage	real	+	power loss due to windage $P_{windage}$	0.0
		+	Accessory losses	
Pacc_0	real	+	power loss $P_{acc0}$ , constant	0.0
Pacc_d	real	+	power loss $P_{accd}$ , scale with density	0.0
Pacc_n	real	+	power loss $P_{accn}$ , scale with density and rpm	0.0
Pacc_deice	real	+	deice power loss $P_{acci}$	0.0
fPacc_ECU	real	+	ECU (etc.) power loss $\ell_{acc}$ (fraction component+transmission power)	0.0
fPacc_IRfan	real	+	IRS fan loss $\ell_{IRfan}$ (fraction total engine power)	0.0
		+	Geometry	
SET_length	int	+	drive shaft length (1 input; 2 calculated)	2
Length_ds	real	+	length $\ell_{DS}$	
fLength_ds	real	+	factor	0.9
<hr/>				
SET_length: input (use Length_ds) or calculated (from fLength_ds)				
<hr/>				
		+	Drive system torque limit	
Plimit_ds	real	+	drive system power limit $P_{DSlimit}$	
fPlimit_ds	real	+	drive system power limit factor	1.0

		+ Drive system ratings	
nrate_ds	int	+ number of ratings (1 for only continuous, maximum nratemax)	1
rating_ds(nratemax)	c*12	+ drive system rating designation	' '
frating_ds(nratemax)	real	+ torque limit factor	1.0
		Derived	
Qlimit_ds	real	drive system torque limit ( $P_{DSlimit}$ at primary rotor reference speed)	
krateC_ds	int	MCQ or MCP rating number	
arating_ds(nratemax)	c*12	drive system rating designation	
xrating_ds(nratemax)	real	torque limit factor ( $f/f_{MCQ}$ )	

---

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,  $fPlimit\_ds \sum(N_{eng} P_{eng})$

SET\_limit\_ds='Pav': from engine power available at transmission sizing conditions and missions (DESIGN\_xmsn)

$fPlimit\_ds(\Omega_{ref}/\Omega_{prim}) \sum(N_{eng} P_{av})$

SET\_limit\_ds='Preq': from engine power required at transmission sizing conditions and missions (DESIGN\_xmsn)

$fPlimit\_ds(\Omega_{ref}/\Omega_{prim}) \sum(N_{eng} P_{req})$

engine shaft: options for SET\_limit\_ds ≠ 'input'

SET\_limit\_es=0: Plimit\_es

SET\_limit\_es=1:  $fPlimit\_es \times$  (engine group  $P_{eng}$  or  $P_{av}$  or  $P_{req}$ , depending on SET\_limit\_ds)

SET\_limit\_es=2:  $fPlimit\_es \times P_{DSlimit}(P_{engEG}/P_{engPG})$

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 ( $nEngine \times P_{eng}$ )

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%Power`  
next torque limits are applied (unless `FltState%SET_Plimit=off`), first engine shaft limit and then drive system limit  
drive system ratings: must include maximum continuous ('MCQ' or 'MCP')  
blank to use engine ratings of first engine group  
limit at flight state is  $rxP_{limit}$ , where  $r$  is the rotor speed ratio and  $x$  is the rating factor  
if `nrate_ds` ≤ 1, drive system rating not used

---

			+ Control	
			+ rotational speed increment $\Delta N$ , 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 <code>nvelmax</code> )	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	
Weight	Weight		+ weight statement (component, not including EngineGroup)	
			+ drive system (propulsion group)	
MODEL_DS	int		+ model (0 input, 1 AFDD, 2 custom)	1
			+ weight increment	
dWgb	real		+ gear box	0.

Structure: Propulsion

221

dWrs	real	+	rotor shaft	0.
dWds	real	+	drive shaft	0.
dWrb	real	+	rotor brake	0.
dWcl	real	+	clutch	0.
dWgd	real	+	gas drive	0.
WDrive	WDrive		AFDD model	
STATE_gear_wt	int	+	drive system state for weight	1
kEngineGroup_wt	int	+	EngineGroup for weight	1
Wtip	real		weight on wing tip	
Wgbrs	real		weight gear box and rotor shaft	
		+	Technology Factors	
TECH_gb	real	+	gear box weight $\chi_{gb}$	1.0
TECH_rs	real	+	rotor shaft weight $\chi_{rs}$	1.0
TECH_ds	real	+	drive shaft weight $\chi_{ds}$	1.0
TECH_rb	real	+	rotor brake weight $\chi_{rb}$	1.0
TECH_cl	real	+	clutch weight $\chi_{cl}$	1.0
TECH_gd	real	+	gas drive weight $\chi_{gd}$	1.0

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = \text{TECH}_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use MODEL}_{xx}=0 \text{ or TECH}_{xx}=0.$$

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)

---

**Structure: WDrive**

Variable	Type	Description	Default
		+ Drive System, AFDD Weight Model	
MODEL_gbrs	int	+ gear box and rotor shaft model (1 AFDD83, 2 AFDD00)	1
		+ gear box (including rotor shafts)	
fShaft	real	+ rotor shaft weight $f_{rs}$ (fraction gear box and rotor shaft weight)	0.13
ngearbox	int	+ number of gear boxes $N_{gb}$	7
fTorque	real	+ second (main or tail) rotor rated torque $f_Q$ (fraction total drive system rated torque)	0.03
		+ drive shaft	
ndriveshaft	int	+ number of intermediate drive shafts $N_{ds}$ (excluding rotor shafts)	6
fPower	real	+ second (main or tail) rotor rated power $f_P$ (fraction total drive system rated power)	0.15
<hr/>			
only MODEL_gbrs=AFDD83 uses ngearbox and fTorque			
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)			
<hr/>			
		+ Custom Weight Model	
WtParam_drive(8)	real	+ parameters	0.

## Chapter 63

**Structure: EngineGroup**

Variable	Type	Description	Default
		+ Engine Group	
title	c*100	+ title	
notes	c*1000	+ notes	
kEngineGroup	int	engine group number	
		+ Description	
MODEL_engine	c*32	+ engine model	'RPTM'
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	
		+ 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 $r = \Omega_{spec}/\Omega_{prim}$ (ratio rpm to rpm of primary rotor in propulsion group)	1.0
		Derived	
iMODEL_engine	int	engine model (MODEL_engine_XXX)	
KIND_engine	int	engine model (MODEL_engine_RPTM, table, comp, motor)	
canConsumePower	int	can consume shaft power (0 only produce power), generator or compressor	
canProducePower	int	can produce shaft power (0 only consume power)	
isConvertReact	int	convertible engine, reaction drive (0 not)	
isConvertJet	int	convertible engine, turbojet/fan (0 not)	

kModel_eng	int	identification (EngineModel or EngineTable or CompressorModel or MotorModel, from IDENT_engine)
kModel_sys2	int	identification (EngineModel, from IDENT_system2)
kBattery	int	battery model, from kFuelTank (0 for none)
nrate	int	number of ratings
rating(nratemax)	c*12	rating designations (lowercase)
krateC	int	MCP rating number
krate_to	int	takeoff power rating number
WOneEng	real	weight one engine $W_{\text{one eng}}$
Nref	real	reference engine turbine speed (at drive state #1)

---

MODEL\_engine: engine model

'RPTEM', 'shaft' = turboshaft engine (RPTEM); IDENT\_engine → EngineModel; fuel is weight

'table' = turboshaft engine (table); IDENT\_engine → EngineTable; fuel is weight

'recip' = reciprocating engine (RPTEM); IDENT\_engine → EngineModel; fuel is weight

'comp' = compressor; IDENT\_engine → CompressorModel; not use fuel

'comp+react' = compressor for reaction drive; 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

'motor+cell' = motor + fuel cell; IDENT\_engine → MotorModel; fuel is weight

MODEL\_engine: convertible engine; only with turboshaft or reciprocating

'+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 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  $P_{\text{eng}} = 0$ . 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, fuel cell)  
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  $f_{gear}$  (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\_power: used if SIZE\_perf='engine', to distribute power required among engine groups  
 must size at least first engine group; SET\_power and fPsize values not used for first group  
 calculate fPsize for first engine group, must be  $> 0$ .  
 FltState%SET\_Preq specifies distribution of power required for flight state

---

		Engine model performance parameters (one engine)
P0(nratemax)	real	power ( $P_0$ )
SP0(nratemax)	real	specific power ( $SP_0$ )
Pmech(nratemax)	real	mechanical limit of power ( $P_{mech}$ or $P_{peak}$ )
sfc0C	real	specific fuel consumption at MCP ( $sfc_{0C}$ )
Fg0C	real	gross jet thrust at MCP ( $F_{g0C} = ST_{0C}\dot{m}_{0C}$ )
Nspec	real	specification turbine speed ( $N_{spec}$ )
Nopt0C	real	optimum turbine speed at MCP ( $N_{opt0C}$ )
mdot0C	real	mass flow at MCP ( $\dot{m}_{0C} = P_{0C}/SP_{0C}$ )
wdot0C	real	fuel flow at MCP ( $\dot{w}_{0C} = sfc_{0C}P_{0C}$ )
		Engine model performance parameters (one engine), ratio converted to base
rsfc0C_conv	real	specific fuel consumption at MCP
rFg0C_conv	real	gross jet thrust at MCP, jet/fan only
rwdot0C_conv	real	fuel flow at MCP

---

motor or generator: only P0, Pmech, Nspec used  
 motor + fuel cell: and mdot0C, wdot0C, sfc0C, with Fg0C=0

---

		+ 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 $P_{ESlimit}$	
fPlimit_es	real	+ engine shaft power limit factor	1.0
		Derived	
Qlimit_es	real	engine shaft torque limit ( $P_{ESlimit}$ at engine turbine reference speed)	
<hr/>			
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, $fPlimit\_ds \sum(N_{eng} P_{eng})$			
SET_limit_ds='Pav': from engine power available at transmission sizing conditions and missions (DESIGN_xmsn)			
$fPlimit\_ds(\Omega_{ref}/\Omega_{prim}) \sum(N_{eng} P_{av})$			
SET_limit_ds='Preq': from engine power required at transmission sizing conditions and missions (DESIGN_xmsn)			
$fPlimit\_ds(\Omega_{ref}/\Omega_{prim}) \sum(N_{eng} P_{req})$			
engine shaft: options for SET_limit_ds ≠ 'input'			
SET_limit_es=0: Plimit_es			
SET_limit_es=1: $fPlimit\_es \times$ (engine group $P_{eng}$ or $P_{av}$ or $P_{req}$ , depending on SET_limit_ds)			
SET_limit_es=2: $fPlimit\_es \times P_{DSlimit}(P_{engEG}/P_{engPG})$			
engine shaft power limit: corresponds to all engines of engine group ( $n_{Engine} \times P_{eng}$ )			
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			
<hr/>			
		+ Installation	
Kffd	real	+ deterioration factor on engine fuel flow or performance $K_{ffd}$	1.05
eta_d	real	+ engine inlet efficiency $\eta_d$ (fraction, for $\delta_M$ )	0.98
		+ power losses (fraction power available, $P_{loss}/P_a$ )	
fPloss_inlet	real	+ engine inlet loss $\ell_{in}$	0.0
fPloss_ps	real	+ inlet particle separator loss $\ell_{in}$	0.0
fPloss_exh	real	+ engine exhaust loss $\ell_{ex}$ (IRS off)	0.015

		+	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 $\eta_{aux}$	0.75
		+	IR suppressor	
		+	power losses (IRS on)	
fPloss_exh_IRon	real	+	engine exhaust loss $\ell_{ex}$	0.030
		+	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 $\eta_{aux}$	0.75
		+	Convertible	
Kffd_conv	real	+	deterioration factor on engine fuel flow or performance $K_{ffd}$	1.05
		+	power losses (fraction power available, $P_{loss}/P_a$ )	
fPloss_exh_conv	real	+	engine exhaust loss $\ell_{ex}$	0.015

---

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; motor + fuel cell: Kffd, fMF\_auxair, eta\_auxair

---

		+	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)	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)	0.0
kSwet	real	+	factor, $k = S_{wet}/(w/N_{eng})^{2/3}$ (Units_Dscale)	0.8
Snac	real		nacelle/cowling area $S_{nac}$	
Swet_nac	real		total wetted area	

---

SET\_geom: calculation override part of location input

SET\_geom=tiltrotor: calculate lateral position (BL) from RotorForEngine

SET\_Swet, wetted area: input (use Swet) or calculated (from kSwet)  
 units of kSwet are  $\text{ft}^2/\text{lb}^{2/3}$  or  $\text{m}^2/\text{kg}^{2/3}$   
 $w = W_{ES}$  (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

---

		Derived	
iDirection	int	nominal orientation (1, -1, 2, -2, 3, -3)	
axis_incid	int	axis incidence ( $\pm 123$ )	
axis_yaw	int	axis yaw ( $\pm 123$ )	
isFixed	int	orientation (1 fixed)	
CBF(3,3)	real	engine axes relative airframe, $C^{BF}$ (fixed)	
ef0(3)	real	engine direction, $e_{f0}$	
ef(3)	real	engine direction, $e_f$ (fixed)	
		+ 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; $\geq 2$ 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 $\psi$	
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
ldrag	real	+	incidence angle $i$ for helicopter nominal drag (deg; 0 for not tilt)	0.
DEngSys	DEngSys		standard model	

			Derived	
DoQC_nac	real		nacelle cruise drag, area $(D/q)_{nac}$	
DoQH_nac	real		nacelle helicopter drag, area $(D/q)_{nac}$	
DoQV_nac	real		nacelle vertical drag, area $(D/q)_{nac}$	
<hr/>				
			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	
<hr/>				
		+ Weight		
Weight	Weight		weight statement (component, including engine weight)	
		+ engine weight		
MODEL_weight	int	+ model (0 input, 1 RPTEM or NASA, 2 custom)		1
dWEng	real	+ weight increment (all engines)		0.0
		+ engine system (except engine), engine section or nacelle group, air induction group		
		+ model (0 input, 1 AFDD, 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		0.0
dWacc	real	+ accessories		0.0
dWsupt	real	+ engine support		0.0
dWcowl	real	+ engine cowling		0.0
dWpylon	real	+ pylon support		0.0
dWair	real	+ air induction		0.0
WEngSys	WEngSys		AFDD model	
Weng_total	real		engine weight	
WES	real		engine system weight $W_{ES}$ (engine, exhaust, accessories)	
Wtip	real		weight on wing tip	

+ Technology Factors				
TECH_eng	real	+	engine weight $\chi_{eng}$	1.0
TECH_cowl	real	+	engine cowling weight $\chi_{cowl}$	1.0
TECH_pylon	real	+	pylon structure weight $\chi_{pylon}$	1.0
TECH_supt	real	+	engine support structure weight $\chi_{supt}$	1.0
TECH_air	real	+	air induction system weight $\chi_{airind}$	1.0
TECH_exh	real	+	exhaust system weight $\chi_{exh}$	1.0
TECH_acc	real	+	engine accessories weight $\chi_{acc}$	1.0

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = \text{TECH}_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use } \text{MODEL}_{xx}=0 \text{ or } \text{TECH}_{xx}=0.$$

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

---

**Structure: DEngSys**

Variable	Type	Description	Default
		+ 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	
Xdrag	real	+ exponent $X_d$	2.0

---

SET\_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

---



## Chapter 65

**Structure: WEngSys**

Variable	Type	Description	Default
fWpylon	real	+ Engine Section or Nacelle Group, Air Induction Group, AFDD Weight Model + pylon support structure weight $f_{pylon}$ (fraction maximum takeoff weight)	0.0
fWair	real	+ air induction weight $f_{airind}$ (fraction engine support plus air induction weight) + Engine System, AFDD Model + exhaust system weight, per engine, including IR suppressor; $W_{exh}$ vs $P_{0C}$	0.3
Kwt0_exh	real	+ $K_{0exh}$	0.0
Kwt1_exh	real	+ $K_{1exh}$ + engine accessories	0.002
MODEL_lub	int	+ lubrication system weight (1 in engine weight, 2 in accessory weight)	1
<hr/> 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 <hr/>			
WtParam_engsys(8)	real	+ Custom Weight Model + parameters	0.

## Chapter 66

**Structure: JetGroup**

Variable	Type	Description	Default
		+ Jet Group	
title	c*100	+ title	
notes	c*1000	+ notes	
kJetGroup	int	jet group number	
		+ 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	
		Derived	
iMODEL_jet	int	jet model (MODEL_jet_XXX)	
KIND_jet	int	jet model (MODEL_jet_RPJEM, simple)	
isConvertReact	int	convertible engine (0 not)	
kModel_jet	int	identification (JetModel, from IDENT_jet)	
kModel_sys2	int	identification (JetModel, from IDENT_system2)	
nrate	int	number of ratings	
rating(nratemax)	c*12	rating designations (lowercase)	
krateC	int	MCT rating number	
krate_to	int	takeoff thrust rating number	
WOneJet	real	weight one jet $W_{one\ jet}$	

---

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  
 '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  $T_{jet} = 0$ . and no size task (or jet thrust not sized)

---

		Jet model performance parameters (one jet)	
T0(nratemax)	real	thrust ( $T_0$ )	
ST0(nratemax)	real	specific thrust ( $ST_0$ )	
Tmech(nratemax)	real	mechanical limit of thrust ( $T_{mech}$ )	
sfc0C	real	specific fuel consumption at MCT ( $sfc_{0C}$ )	
mdot0C	real	mass flow at MCT ( $\dot{m}_{0C} = T_{0C}/ST_{0C}$ )	
wdot0C	real	fuel flow at MCT ( $\dot{w}_{0C} = sfc_{0C}T_{0C}$ )	
Edot0C	real	energy flow at MCT ( $\dot{w}_{0C} = sfc_{0C}T_{0C}$ )	
		Jet model performance parameters (one jet), ratio converted to base	
rsfc0C_conv	real	specific fuel consumption at MCT	
rwdot0C_conv	real	fuel flow at MCT	
		+ Installation	
Kffd	real	+ deterioration factor on jet fuel flow $K_{ffd}$	1.05
eta_d	real	+ jet inlet efficiency $\eta_d$ (fraction, for $\delta_M$ )	0.98
		+ power losses (fraction thrust available, $T_{loss}/T_a$ )	
fTloss_inlet	real	+ engine inlet loss $\ell_{in}$	0.0
fTloss_exh	real	+ engine exhaust loss $\ell_{ex}$ (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 $\eta_{aux}$	0.75

		+ IR suppressor	
		+ power losses (IRS on)	
fTloss_exh_IRon	real	+ engine exhaust loss $\ell_{ex}$	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 $\eta_{aux}$	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 $\ell_{ex}$	0.01
<hr/>			
installation power losses = inlet + exhaust (including IRS)			
IR suppressor state specified by STATE_IRS_jet in operating condition			
<hr/>			
		+ Simple force generator	
Tmax	real	+ design maximum thrust $T_{max}$	0.0
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
<hr/>			
fuel tank system identified must be consistent with SET_burn			
<hr/>			
		+ 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)	0.0
kSwet	real	+ factor, $k = S_{wet}/(W_{ES}/N_{jet})^{2/3}$ (Units_Dscale)	0.8

Snac	real	nacelle/cowling area $S_{nac}$	
Swet_nac	real	total wetted area	
<hr/>			
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			
<hr/>			
Derived			
iDirection	int	nominal orientation (1, -1, 2, -2, 3, -3)	
axis_incid	int	axis incidence ( $\pm 123$ )	
axis_yaw	int	axis yaw ( $\pm 123$ )	
isFixed	int	orientation (1 fixed)	
CBF(3,3)	real	jet relative airframe, $C^{BF}$ (fixed)	
ef0(3)	real	jet direction, $e_{f0}$	
ef(3)	real	jet direction, $e_f$ (fixed)	
+ 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 $\psi$	
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
ldrag	real	+	incidence angle $i$ for helicopter nominal drag (deg; 0 for not tilt)	0.
DJetSys	DJetSys		standard model	
			Derived	
DoQC_nac	real		nacelle cruise drag, area $(D/q)_{nac}$	
DoQH_nac	real		nacelle helicopter drag, area $(D/q)_{nac}$	
DoQV_nac	real		nacelle vertical drag, area $(D/q)_{nac}$	

		+	Weight		
Weight	Weight		weight statement (component, including jet weight)		
		+	jet weight		
MODEL_weight	int	+	model (0 input, 1 RPJEM, 2 custom)		1
dWJet	real	+	weight increment (all jets)		0.0
		+	engine system (except jet), engine section or nacelle group, air induction group		
		+	model (0 input, 1 AFDD, 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		0.0
dWacc	real	+	accessories		0.0
dWsupt	real	+	engine support		0.0
dWcowl	real	+	engine cowling		0.0
dWpylon	real	+	pylon support		0.0
dWair	real	+	air induction		0.0
WJetSys	WJetSys		AFDD model		
Wjet_total	real		jet weight		
WES	real		engine system weight $W_{ES}$ (engine, exhaust, accessories)		
		+	Technology Factors		
TECH_jet	real	+	jet weight $\chi_{jet}$		1.0
TECH_jetcowl	real	+	engine cowling weight $\chi_{cowl}$		1.0
TECH_jetpylon	real	+	pylon structure weight $\chi_{pylon}$		1.0
TECH_jetsupt	real	+	engine support structure weight $\chi_{supt}$		1.0
TECH_jetair	real	+	air induction system weight $\chi_{airind}$		1.0
TECH_jetexh	real	+	exhaust system weight $\chi_{exh}$		1.0
TECH_jetacc	real	+	engine accessories weight $\chi_{acc}$		1.0

---

weight model result multiplied by technology factor and increment added:

$W_{xx} = TECH_{xx} * W_{xx\_model} + dW_{xx}$ ; 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

---

**Structure: DJetSys**

Variable	Type	Description	Default
		+ 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	
Xdrag	real	+ exponent $X_d$	2.0

---

SET\_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

---



**Structure: WJetSys**

Variable	Type	Description	Default
		+ Engine Section or Nacelle Group, Air Induction Group, AFDD Weight Model	
fWpylon	real	+ pylon support structure weight $f_{pylon}$ (fraction maximum takeoff weight)	0.0
fWair	real	+ air induction weight $f_{airind}$ (fraction engine support plus air induction weight)	0.3
		+ Engine System, AFDD Model	
		+ exhaust system weight, per jet; $W_{exh}$ vs $T_{0C}$	
Kwt0_exh	real	+ $K_{0exh}$	0.0
Kwt1_exh	real	+ $K_{1exh}$	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.

**Structure: ChargeGroup**

Variable	Type	Description	Default
		+ Charge Group	
title	c*100	+ title	
notes	c*1000	+ notes	
kChargeGroup	int	charge group number	
		+ Description	
MODEL_charge	c*32	+ charger model	' '
IDENT_charge	c*16	+ charger identification	'Charge'
nCharge	int	+ number of chargers $N_{\text{chrg}}$	1
Pchrg	real	+ charger power $P_{\text{chrg}}$ (SLS static at takeoff rating, 0. for P0_ref(rating_to))	0.
rating_to	c*12	+ takeoff thrust rating	'MCP'
rating_idle	c*12	+ idle thrust rating	'MCP'
kFuelTank	int	+ fuel tank system number (generated)	1
kFuelTank_burn	int	+ fuel tank system number (burned)	
		Derived	
iMODEL_charge	int	charger model (MODEL_charge_XXX)	
KIND_charge	int	charger model (MODEL_charge_fuelcell, solarcell)	
kModel_chrg	int	identification (FuelCellModel or SolarCellModel, from IDENT_charge)	
kBattery	int	battery model, from kFuelTank (0 for none)	
nrate	int	number of ratings	
rating(nratemax)	c*12	rating designations (lowercase)	
krateC	int	MCP rating number	
krate_to	int	takeoff thrust rating number	
WOneChrg	real	weight one charger $W_{\text{one chrg}}$	

---

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

charger identification: match ident of FuelCellModel or SolarCellModel

for fixed charger: use  $P_{\text{chrg}} = 0$ . 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

---

#### Charger model performance parameters (one charger)

P0(nratemax)	real		power ( $P_0$ )	
sfc0C	real		specific fuel consumption at MCP ( $\text{sfc}_{0C}$ )	
mdot0C	real		mass flow at MCP ( $\dot{m}_{0C}$ )	
wdot0C	real		fuel flow at MCP ( $\dot{w}_{0C} = \text{sfc}_{0C} P_{0C}$ )	
solararea	real		solar cell total area	
		+	<b>Installation</b>	
Kffd	real	+	deterioration factor on charger fuel flow or performance $K_{ffd}$	1.05
		+	auxiliary air momentum drag	
fMF_auxair	real	+	mass flow $f_{\text{aux}}$ (fraction charger mass flow)	0.007
eta_auxair	real	+	ram recovery efficiency $\eta_{\text{aux}}$	0.75
		+	<b>Geometry</b>	
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 $S_{\text{wet}}$ (per charger)	0.0
kSwet	real	+	factor, $k = S_{\text{wet}} / (W / N_{\text{chrg}})^{2/3}$ (Units_Dscale)	0.8
Snac	real		nacelle/cowling area $S_{\text{nac}}$	
Swet_nac	real		total wetted area	

---

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

---

		Derived	
iDirection	int	nominal orientation (1, -1, 2, -2, 3, -3)	
axis_incid	int	axis incidence ( $\pm 123$ )	
axis_yaw	int	axis yaw ( $\pm 123$ )	
isFixed	int	orientation (1 fixed)	
CBF(3,3)	real	charger relative airframe, $C^{BF}$ (fixed)	
ef0(3)	real	charger direction, $e_{f0}$	
ef(3)	real	charger direction, $e_f$ (fixed)	
		+ 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 $\psi$	
INPUT_yaw	int	+	connection to aircraft controls (0 none, 1 input $T$ matrix)	1
T_yaw(ncontmax,nstatemax)	real	+	control matrix	
nV_yaw	int	+	number of speeds (0 zero value; 1 constant; $\geq 2$ piecewise linear, maximum nvelmax)	0
yaw(nvelmax)	real	+	values	
V_yaw(nvelmax)	real	+	speeds (CAS or TAS)	
<hr/>				
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				
<hr/>				
		+	Nacelle Drag	
MODEL_drag	int	+	model (0 none, 1 standard)	1
ldrag	real	+	incidence angle $i$ for helicopter nominal drag (deg; 0 for not tilt)	0.
DChrgSys	DChrgSys		standard model	
			Derived	
DoQC_nac	real		nacelle cruise drag, area $(D/q)_{nac}$	
DoQH_nac	real		nacelle helicopter drag, area $(D/q)_{nac}$	
DoQV_nac	real		nacelle vertical drag, area $(D/q)_{nac}$	
		+	Weight	
Weight	Weight		weight statement (component, including charger weight)	
		+	charger weight	
MODEL_weight	int	+	model (0 input, 1 AFDD, 2 custom)	1
dWChrg	real	+	weight increment (all chargers)	0.0
WChrgSys	WChrgSys		AFDD model	
Wchrg_total	real		charge group weight	
WES	real		engine system weight $W_{ES}$ (engine, exhaust, accessories)	
		+	Technology Factors	
TECH_chrg	real	+	charger weight $\chi_{chrg}$	1.0

---

weight model result multiplied by technology factor and increment added:

$$W_{xx} = \text{TECH}_{xx} * W_{xx\_model} + dW_{xx}; \text{ for fixed (input) weight use MODEL}_{xx}=0 \text{ or TECH}_{xx}=0.$$

engine system weight = engine + exhaust + accessory = charge group weight (WES used for nacelle wetted area)

charger weight parameters in FuelCellModel or SolarCellModel

---

**Structure: DChrgSys**

Variable	Type	Description	Default
		+ 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	
Xdrag	real	+ exponent $X_d$	2.0

---

SET\_xxx: fixed (use DoQ) or scaled (use CD); other parameter calculated

---

---

**Structure: WChrgSys**

Variable	Type	Description	Default
WtParam_chrgsys(8)	real	+ Custom Weight Model + parameters	0.



**Structure: EngineModel**

Variable	Type	Description	Default
		+ Engine Model	
title	c*100	+ title	'Default'
notes	c*1000	+ notes	
ident	c*16	+ identification	'Engine'
<hr/> <p>engine identification: used by IDENT_engine of EngineGroup input (eg 'T800')</p> <p>installed: power available <math>P_{av}</math>, power required <math>P_{req}</math>, gross jet thrust <math>F_G</math>, net jet thrust <math>F_N</math>  uninstalled: power available <math>P_a</math>, power required <math>P_q</math>, gross jet thrust <math>F_g</math>, net jet thrust <math>F_n</math>  "0" = SLS static; "C" = MCP  mass flow = power / specific power (<math>SP = P/\dot{m}</math>); fuel flow = specific fuel consumption * power (<math>sfc = \dot{w}/P</math>)</p> <p>engine model can be used by more than one engine group, so all parameters fixed</p> <p>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</p> <hr/>			
kEngineModel	int	engine model number	
		+ Weight	
MODEL_weight	int	+ RPTM model (0 fixed, 1 $W(P)$ , 2 $SW(\dot{m})$ )	1
Weng	real	+ engine weight (fixed)	0.
		+ engine weight, $W_{eng}$ vs $P_{0C}$ model ( $W = K_{0eng} + K_{1eng}P + K_{2eng}P^{X_{eng}}$ )	
Kwt0_eng	real	+ constant $K_{0eng}$	0.
Kwt1_eng	real	+ constant $K_{1eng}$	0.25
Kwt2_eng	real	+ constant $K_{2eng}$	0.

Structure: EngineModel				250
Xwt_eng	real	+	exponent $X_{eng}$	0.
		+	engine weight, $SW = P/W_{eng}$ vs $\dot{m}_{0C}$ model	
SW_ref	real	+	specific weight reference $SW_{ref}$ ( $\dot{m} = \dot{m}_{tech}$ )	4.
SW_limit	real	+	specific weight limit $SW_{lim}$ ( $\dot{m} = \dot{m}_{lim}$ )	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'
krateC	int		MCP rating number	
		+	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 ( $P_{mech}$ )	2500.
sfc0C_ref	real	+	specific fuel consumption at MCP ( $sfc_{0C}$ )	0.45
SF0C_ref	real	+	specific jet thrust ( $F_{g0C} = SF_{0C}\dot{m}_{0C}$ )	10.
Nspec_ref	real	+	specification turbine speed ( $N_{spec}$ )	20000.
Nopt0C_ref	real	+	optimum turbine speed at MCP ( $N_{opt0C}$ )	20000.
			Derived ratios	
rP0(nratemax)	real		power ( $P_{0R}/P_{0C}$ )	
rSP0(nratemax)	real		specific power ( $SP_{0R}/SP_{0C}$ )	
rPmech(nratemax)	real		mechanical limit of power ( $P_{mechR}/P_{0C}$ )	

---

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' = Contingency 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

reciprocating engine (MODEL\_engine):

mass flow proportional engine speed ( $\dot{m} \sim N$ ; Kmfa and Kmfg not used)

no other influence of engine speed (MODEL\_OptN=0, INPUT\_param=1)

suggest fuel flow proportional to power ( $\dot{w} \sim P$ ; Kffq1=1, Xffq=0) and zero gross thrust (SF0C\_ref=0.)

---

		+ Technology	
SP0C_tech	real	+ specific power at MCP $SP_{tech}$ (0. for SP0_ref(MCP))	0.
sfc0C_tech	real	+ specific fuel consumption at MCP $sfc_{tech}$ (0. for sfc0C_ref)	0.
Nspec_tech	real	+ specification turbine speed $N_{tech}$ (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 $sfc$ ( $\dot{m}_{lim}$ )	30.
SP0C_limit	real	+ specific power limit $SP_{lim}$	200.
sfc0C_limit	real	+ specific fuel consumption limit $sfc_{lim}$	0.34
KNspec	real	+ specification turbine speed variation ( $K_{Ns2}$ )	0.
		Derived	
		specific power available (SLS static, MCP, $N_{spec}$ ), $SP_{0C}$ vs $\dot{m}_{0C}$	
P0C_limit	real	power limit	
Ksp0	real	$K_{sp0}$	
Ksp1	real	$K_{sp1}$	
		specific fuel consumption (SLS static, MCP, $N_{spec}$ ), $sfc_{0C}$ vs $\dot{m}_{0C}$	
Ksfc0	real	$K_{sfc0}$	
Ksfc1	real	$K_{sfc1}$	
		specification turbine speed, $N_{spec}$ vs $\dot{m}_{0C}$	
KNs1	real	$K_{Ns1}$	
KNs2	real	$K_{Ns2}$	
		optimum turbine speed, $N_{opt0C}$	
KNo	real	$K_{No}$	
		engine weight, $SW = P/W_{eng}$ vs $\dot{m}_{0C}$	
Ksw0	real	$K_{sw0}$	
Ksw1	real	$K_{sw1}$	

---

$SP$  and  $sfc$  functions are defined by values  $SP_{0C\_tech}$ ,  $sfc_{0C\_tech}$ ,  $\dot{m}_{tech}=P_{0C\_ref}/SP_{0C\_tech}$  and limits  $SP_{0C\_limit}$ ,  $sfc_{0C\_limit}$ ,  $MF\_limit$

defaults  $SP_{0C\_tech}=SP_{0\_ref}(MCP)$ ,  $sfc_{0C\_tech}=sfc_{0C\_ref}$ ,  $N_{spec\_tech}=N_{spec\_ref}$   
 require  $\dot{m}_{tech} < \dot{m}_{lim}$  (otherwise get  $SP_{0C} = SP_{0C\_tech}$  and  $sfc_{0C} = sfc_{0C\_tech}$ )

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, $N_{opt}/N_{spec}$ vs $P_q/P_0$	
KNoptA	real	+	constant $K_{NoptA}$	1.
KNoptB	real	+	constant $K_{NoptB}$	0.
		+	cubic, $N_{opt}/N_{opt0C}$ vs $P_q/P_{0C}$	
KNopt0	real	+	constant $K_{Nopt0}$	1.
KNopt1	real	+	constant $K_{Nopt1}$	0.
KNopt2	real	+	constant $K_{Nopt2}$	0.
KNopt3	real	+	constant $K_{Nopt3}$	0.
XNopt	real	+	exponent $X_{Nopt}$	0.
		+	power turbine efficiency function, $\eta_t(N)/\eta_t(N_{spec})$	
XNeta	real	+	exponent $X_{N\eta}$	2.0

---

engine power and performance variation with power turbine speed determined by  $N_{opt}$  and  $X_{N\eta}$   
 used only for  $INPUT\_param = \text{single set}$ ; no variation if  $MODEL\_OptN=0$

---

			+ Power Available and Power Required Parameters	
INPUT_param	int	+	parameter input form (1 single set; 2 interpolated)	1
Param	EngineParam		single set	
			+ interpolated	
nspeed	int	+	number of engine speeds (maximum nspeedmax)	1
rNeng(nspeedmax)	real	+	engine speed ratio, $N/N_{\text{spec}}$	1.
ParamN(nspeedmax)	EngineParam		parameter sets	

---

interpolated: rNeng unique and sequential

---

**Structure: EngineParam**

Variable	Type	Description	Default
		+ Parameters	
parent	int	parent (1 Param, 2 ParamN)	
kEngineModel	int	engine model number	
kEngineParam	int	engine param number (ParamN)	
		+ Power Available	
INPUT_lin	int	+ input form (1 coefficients $K_0, K_1$ ; 2 values $\theta_b, K_b$ ) + referred specific power available, $SP_a/SP_0$ vs temperature	1
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	+ $\theta_b$	
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	+ $\theta_b$	
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	+ $\theta_b$	
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	+ $\theta_b$	
Xmfab(nengkmax,nratemax)	real	+ $X_{mfa-b}$	

---

piecewise linear function:

input form = coefficients  $K_0, K_1$  (N sets) or values  $\theta_b, K_b$  (N+1 values)

form not input is calculated (N-1  $\theta_b, K_b$  or N  $K_0, K_1$ )

input  $K_0, K_1$ : adjacent  $K_1$  different, resulting  $\theta_b$  unique and sequential

input  $\theta_b, K_b$ :  $\theta_b$  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

$\eta_t$  = 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 $\ell_{ex}$	
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: EngineTable**

Variable	Type	Description	Default
		+ Engine Table	
title	c*100	+ title	'Default'
notes	c*1000	+ notes	
ident	c*16	+ identification	'Engine'
<hr/> engine identification: used by IDENT_engine of EngineGroup input engine table can be used by more than one engine group, so all parameters fixed <hr/>			
kEngineTable	int	engine table number	
		+ Engine ratings	
nrate	int	+ number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+ rating designations	'MCP'
krateC	int	MCP rating number	
Nspec	real	+ Specification turbine speed ( $N_{spec}$ )	
		+ Table	
nalt	int	+ number of altitudes (maximum nengtmax)	0
nspeed	int	+ number of speeds (maximum nengtmax)	0
alt(nengtmax)	real	+ altitude $h$	
speed(nengtmax)	real	+ speed $V$ (TAS)	
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 $F_N(h, V, R)$	

			Factors	
Kp	real	+	power available	1.0
Kw	real	+	fuel flow	1.0
Kf	real	+	net thrust	1.0

---

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, so expect fPloss\_inlet=0, fPloss\_ps=0, fPloss\_exh=0

fuel flow multiplied by Kffd, accounting for deterioration of engine efficiency

---

**Structure: CompressorModel**

Variable	Type	Description	Default
		+ Compressor Model	
title	c*100	+ title	'Default'
notes	c*1000	+ notes	
ident	c*16	+ identification	'Comp'
<hr/> 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 <hr/>			
kCompressorModel	int	compressor model number	
		+ Weight	
MODEL_weight	int	+ model (0 fixed, 1 $W(P)$ )	1
Wcomp	real	+ compressor weight (fixed)	0.
		+ compressor weight, $W_{comp}$ vs $P_{0C}$ model ( $W = K_{0comp} + K_{1comp}P + K_{2comp}P^{X_{comp}}$ )	
Kwt0_comp	real	+ constant $K_{0comp}$	0.
Kwt1_comp	real	+ constant $K_{1comp}$	0.2
Kwt2_comp	real	+ constant $K_{2comp}$	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'
krateC	int	+ MCP rating number	
		+ Reference	
P0_ref(nratemax)	real	+ power ( $P_0$ )	
SP0_ref(nratemax)	real	+ specific power ( $SP_0$ )	
Pmech_ref(nratemax)	real	+ mechanical limit of power ( $P_{mech}$ )	
ST0C_ref	real	+ specific jet thrust ( $F_{g0C} = ST_{0C}\dot{m}_{0C}$ )	
Nspec_ref	real	+ specification compressor speed ( $N_{spec}$ )	
		Derived ratios	
rP0(nratemax)	real	+ power ( $P_{0R}/P_{0C}$ )	
rSP0(nratemax)	real	+ specific power ( $SP_{0R}/SP_{0C}$ )	
rPmech(nratemax)	real	+ mechanical limit of power ( $P_{mechR}/P_{0C}$ )	
<hr/>			
Reference Compressor Rating: SLS, static			
if MCP scaled, ratios to MCP values kept constant			
compressor rating: match rating designation in FltState			
<hr/>			
		+ 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.
		+ referred specific thrust at power required, $ST_{req}/ST_0$	
Xstq	real	+ exponent $X_{stq}$	1.

**Structure: MotorModel**

Variable	Type	Description	Default
		+ Motor Model	
title	c*100	+ title	'Default'
notes	c*1000	+ notes	
ident	c*16	+ identification	'Motor'
<hr/> 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 <hr/>			
kMotorModel	int	motor model number	
		+ Weight	
MODEL_weight	int	+ NASA model (0 fixed, 1 $W(P)$ , 2 $W(Q)$ )	2
Wmotor	real	+ motor weight (fixed)	0.
		+ motor weight, $W_{\text{motor}}$ vs $P_{0C}$ model ( $W = K_{0\text{motor}} + K_{1\text{motor}}P + K_{2\text{motor}}P^{X_{\text{motor}}}$ )	
Kwt0_motor	real	+ constant $K_{0\text{motor}}$	0.
Kwt1_motor	real	+ constant $K_{1\text{motor}}$	0.
Kwt2_motor	real	+ constant $K_{2\text{motor}}$	0.
Xwt_motor	real	+ exponent $X_{\text{motor}}$	0.
		+ motor weight, $W_{\text{motor}}$ vs $Q_{\text{peak}}$ model	
KIND_design	int	+ torque-to-weight design (0 only high $Q/W$ ; 1 high $Q/W$ , 2 low $Q/W$ factor)	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'
krateC	int	+ MCP rating number	
		+ Reference	
P0_ref(nratemax)	real	+ power ( $P_0$ )	0.
Ppeak_ref(nratemax)	real	+ mechanical limit of power ( $P_{peak}$ )	
Nspec_ref	real	+ specification motor speed ( $N_{spec}$ )	
		Derived ratios	
rP0(nratemax)	real	+ power ( $P_{0R}/P_{0C}$ )	
rPpeak(nratemax)	real	+ mechanical limit of power ( $P_{peakR}/P_{0C}$ )	
<hr/>			
Reference Motor Rating: SLS, static			
if MCP scaled, ratios to MCP values kept constant			
motor rating: match rating designation in FltState			
<hr/>			
		+ 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 $P_{eng}$ )	1.00
loss_motor	real	+ power loss (fraction $P_{eng}$ )	0.00
		+ efficiency map ( $P_{loss} = P_{eng} f_{loss} \sum_{i=0}^3 \sum_{j=0}^3 C_{ij} t^i n^j$ )	
Closs(4,4)	real	+ loss coefficients $C_{loss}(i+1,j+1) = C_{ij}$	0.00
floss	real	+ factor $f_{loss}$	1.00
eta_cont	real	+ controller efficiency	1.00
		+ Fuel Cell	
sfc0C	real	+ specific fuel consumption at MCP ( $sfc_{0C}$ )	0.
Kmf	real	+ mass flow ratio ( $\dot{m}/\dot{w}$ )	86.
KIND_eff_cell	int	+ efficiency kind (1 fixed, 2 function power)	2

## Structure: MotorModel

263

eta_cell	real	+	reference efficiency (at $P_{eng}$ )	1.00
loss_cell	real	+	power loss (fraction $P_{eng}$ )	0.00
		+	Scaling	
KNspec	real	+	specification motor speed variation ( $K_{Ns}$ )	0.
KNbase	real	+	base motor speed variation ( $K_{Nb}$ )	0.

---

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

---

**Structure: JetModel**

Variable	Type	Description	Default
		+ Jet Model	
title	c*100	+ title	'Default'
notes	c*1000	+ notes	
ident	c*16	+ identification	'Jet'
<hr/> <p>jet identification: used by IDENT_jet of JetGroup input</p> <p>installed: thrust available <math>T_{av}</math>, thrust required <math>T_{req}</math>  uninstalled: thrust available <math>T_a</math>, thrust required <math>T_q</math>  “0” = SLS static; “C” = MCT  mass flow = thrust / specific thrust (<math>ST = T/\dot{m}</math>); fuel flow = specific fuel consumption * thrust (<math>sfc = \dot{w}/T</math>)</p> <p>jet model can be used by more than one jet group, so all parameters fixed</p> <p>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</p> <hr/>			
kJetModel	int	jet model number	
		+ Weight	
MODEL_weight	int	+ RPJEM model (0 fixed, 1 $W(T)$ )	1
Wjet	real	+ jet weight (fixed)	0.
		+ jet weight, $W_{jet}$ vs $T_{0C}$ model ( $W = K_{0jet} + K_{1jet}T + K_{2jet}T^{X_{jet}}$ )	
Kwt0_jet	real	+ constant $K_{0jet}$	0.
Kwt1_jet	real	+ constant $K_{1jet}$	0.2



Kwt2_jet	real	+	constant $K_{2jet}$	0.
Xwt_jet	real	+	exponent $X_{jet}$	0.
WtParam_jet(8)	real	+	Custom Weight Model parameters	0.
		+	Parameters	
		+	Jet Ratings	
nrate	int	+	number of ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	rating designations	'MCT'
krateC	int	+	MCT rating number	
		+	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 ( $T_{mech}$ )	
sfc0C_ref	real	+	specific fuel consumption at MCT ( $sfc_{0C}$ )	
			Derived ratios	
rT0(nratemax)	real		thrust ( $T_{0R}/T_{0C}$ )	
rST0(nratemax)	real		specific thrust ( $ST_{0R}/ST_{0C}$ )	
rTmech(nratemax)	real		mechanical limit of thrust ( $T_{mechR}/T_{0C}$ )	
<hr/>				
Reference Jet Rating: SLS, static				
if MCT scaled, ratios to MCT values kept constant				
jet rating: match rating designation in FltState				
<hr/>				
		+	Technology	
ST0C_tech	real	+	specific thrust at MCT $ST_{tech}$ (0. for ST0_ref(MCT))	0.
sfc0C_tech	real	+	specific fuel consumption at MCT $sfc_{tech}$ (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 ( $\dot{m}_{lim}$ )	0.
ST0C_limit	real	+	specific thrust limit $ST_{lim}$	0.
sfc0C_limit	real	+	specific fuel consumption limit $sfc_{lim}$	

		Derived		
			specific thrust available (SLS static, MCT), $ST_{0C}$ vs $\dot{m}_{0C}$	
T0C_limit	real		thrust limit	
Kst0	real		$K_{st0}$	
Kst1	real		$K_{st1}$	
			specific fuel consumption (SLS static, MCT), $sfc_{0C}$ vs $\dot{m}_{0C}$	
Ksfc0	real		$K_{sfc0}$	
Ksfc1	real		$K_{sfc1}$	
<hr/>				
<p><math>ST</math> and <math>sfc</math> functions are defined by values <math>ST_{0C\_tech}</math>, <math>sfc_{0C\_tech}</math>, <math>\dot{m}_{tech}=T_{0C\_ref}/ST_{0C\_tech}</math>  and limits <math>ST_{0C\_limit}</math>, <math>sfc_{0C\_limit}</math>, <math>MF\_limit</math>  defaults <math>ST_{0C\_tech}=ST_{0\_ref}(MCT)</math>, <math>sfc_{0C\_tech}=sfc_{0C\_ref}</math>  require <math>\dot{m}_{tech} &lt; \dot{m}_{lim}</math> (otherwise get <math>ST_{0C} = ST_{0C\_tech}</math> and <math>sfc_{0C} = sfc_{0C\_tech}</math>)  for no variation of <math>ST</math> and <math>sfc</math> with scale, use <math>FIX\_size=1</math> or <math>MF\_limit=0</math>.</p> <hr/>				
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.
Xffq	real	+	exponent $X_{ffq}$	1.
		+	referred mass flow at thrust required, $\dot{m}_{req}/\dot{m}_{0C}$ vs $T_q/T_{0C}$	
Kmfq	real	+	constant $K_{mfq}$ (0, 1, or 1/2)	1.
Xmfq	real	+	exponent $X_{mfq}$	1.

**Structure: FuelCellModel**

Variable	Type	Description	Default
		+ Fuel Cell Model	
title	c*100	+ title	'Default'
notes	c*1000	+ notes	
ident	c*16	+ identification	'Cell'
<hr/> 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 <hr/>			
kFuelCellModel	int	fuel cell model number	
		+ Weight	
MODEL_weight	int	+ model (0 fixed, 1 $W(P)$ )	1
Wcell	real	+ fuel cell weight (fixed)	0.
		+ fuel cell weight, $W_{\text{cell}}$ vs $P_0C$ 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 $X_{\text{cell}}$	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'
krateC	int	+ MCP rating number	
		+ Reference	
P0_ref(nratemax)	real	+ power ( $P_0$ )	0.
sfc0C_ref	real	+ specific fuel consumption at MCP ( $sfc_{0C}$ )	0.
		Derived ratios	
rP0(nratemax)	real	+ power ( $P_{0R}/P_{0C}$ )	
<hr/>			
Reference Fuel Cell Rating: SLS, static			
if MCP scaled, ratios to MCP values kept constant			
fuel cell rating: match rating designation in FltState			
<hr/>			
		+ Performance	
Kmf	real	+ mass flow ratio ( $\dot{m}/\dot{w}$ )	86.
		+ Efficiency	
KIND_eff	int	+ kind (1 fixed, 2 function power)	2
eta_cell	real	+ reference efficiency (at $P_{\text{chrg}}$ )	1.00
loss_cell	real	+ power loss (fraction $P_{\text{chrg}}$ )	0.00

**Structure: SolarCellModel**

Variable	Type	Description	Default
		+ Solar Cell Model	
title	c*100	+ title	'Default'
notes	c*1000	+ notes	
ident	c*16	+ identification	'Cell'
<hr/> 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 <hr/>			
kSolarCellModel	int	solar cell model number	
		+ Weight	
MODEL_weight	int	+ model (0 fixed, 1 $W(A)$ )	1
Wsolar	real	+ solar cell weight (fixed)	0.
ssolar	real	+ weight density ( $\text{kg/m}^2$ )	
		+ 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'
krateC	int	MCP rating number	

Structure: SolarCellModel

270

P0_ref(nratemax)	real	+ Reference	
		+ power ( $P_0$ )	0.
rP0(nratemax)	real	Derived ratios	
		power ( $P_{0R}/P_{0C}$ )	

---

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

---

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

**Structure: BatteryModel**

Variable	Type	Description	Default
		+ Battery Model	
title	c*100	+ title	'Default'
notes	c*1000	+ notes	
ident	c*16	+ identification	'Battery'
<hr/> 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 <hr/>			
kBatteryModel	int	battery model number	
		+ 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 $x_{CCmax}$ (1/hr)	4.
		Derived	
CfromE	real	charge capacity $C$ (A-hr) from energy capacity (MJ); $(10^6/3600)/V_{ref}$	
PfromE	real	power capacity $P$ (hp or kW) from energy capacity (MJ); $x_{mbd}/E_{conv\_dE}$	
		+ 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 $P_{\text{ref}}$ )	1.00
loss_chrg	real	+	power loss (fraction $P_{\text{ref}}$ )	0.00
		+	Lithium-Ion Model	
		+	discharge	
fcrit	real	+	critical voltage factor ( $F_V = f_{\text{crit}}$ is capacity)	0.6
fd	real	+	nominal discharge voltage ( $V_d = f_d V_{\text{ref}}$ )	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 $T_{\text{ref}}$ (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 $x_{mbd} C R / V_{\text{ref}}$	0.1
kdl	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 ( $V_c = f_c V_{\text{ref}}$ )	1.0
kcV	real	+	CC phase starting voltage decrement $k_{cV}$	0.1
ks	real	+	CV phase parameter $k_{\sigma}$	0.2
			Derived	
DoDrev(40)	real		reversed DoD	
FVrev(40)	real		reversed FV	

---

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; arbitrary 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)

---

		Derived
		input, from Aircraft%INPUT_geom and FIX_geom (1 fixed; 2 scaled)
INPUT_geom_x	int	$x$
INPUT_geom_y	int	$y$
INPUT_geom_z	int	$z$
		from Aircraft%INPUT_geom and FIX_geom (0 calculated, 1 fixed, 2 scaled)
FIX_x	int	$x$
FIX_y	int	$y$
FIX_z	int	$z$
isFixed	int	all fixed (0 not, some scaled or calculated)
		fixed (dimensional, arbitrary origin)
SLloc	real	stationline
BLloc	real	butline
WLloc	real	waterline
		scaled (based on reference length, from reference point)
XoLloc	real	$x/L$
YoLloc	real	$y/L$
ZoLloc	real	$z/L$
		reference length
KIND_scale_loc	int	from Aircraft%KIND_scale and KIND_scale (1 rotor radius, 2 wing span, 3 fuselage length)
kScale_loc	int	from Aircraft%kScale and kScale (component number)
scale	real	reference length

---

FIX = 0:  $x$  calculation depends on component/configuraton; calc SLloc and XoLloc

FIX = 1:  $x$  from SLloc; calc XoLloc

FIX = 2:  $x$  from XoLloc; calc SLloc

---

		Geometry (dimensional, body axes, relative reference point)
x	real	$x$ (+ forward)
y	real	$y$ (+ right)
z	real	$z$ (+ down)

## Chapter 82

**Structure: Weight**

Variable	Type	Description	Default
WE	real	WEIGHT EMPTY	
W_structure	real	STRUCTURE	
W_wing	real	wing group	
W_wing_basic	real	basic structure	
W_wing_secondary	real	secondary structure	
W_wing_fair	real	fairings (not RP8A)	
W_wing_fit	real	fittings (not RP8A)	
W_wing_fold	real	fold/tilt (not RP8A)	
W_wing_control	real	control surfaces	
W_rotor	real	rotor group	
W_rotor_blade	real	blade assembly	
W_rotor_hub	real	hub & hinge	
W_rotor_basic	real	basic (not RP8A)	
W_rotor_shaft	real	inter-rotor shaft (not RP8A)	
W_rotor_fair	real	fairing/spinner (not RP8A)	
W_rotor_fold	real	blade fold (not RP8A)	
W_rotor_supt	real	rotor support structure (not RP8A)	
W_rotor_duct	real	duct (not RP8A)	
W_tail	real	empennage group	
W_Htail	real	horizontal tail (not RP8A)	
W_Htail_basic	real	basic (not RP8A)	
W_Htail_fold	real	fold (not RP8A)	
W_Vtail	real	vertical tail (not RP8A)	
W_Vtail_basic	real	basic (not RP8A)	
W_Vtail_fold	real	fold (not RP8A)	
W_tailrotor	real	tail rotor (not RP8A)	
W_tr_blade	real	blades	
W_tr_hub	real	hub & hinge	

W_tr_supt	real	rotor supports
W_tr_duct	real	rotor/fan duct
W_fuselage	real	fuselage group
W_fus_basic	real	basic (not RP8A)
W_fus_wingfold	real	wing & rotor fold/retraction (not RP8A)
W_fus_tailfold	real	tail fold/tilt (not RP8A)
W_fus_mar	real	marinization (not RP8A)
W_fus_press	real	pressurization (not RP8A)
W_fus_crash	real	crashworthiness (not RP8A)
W_gear	real	alighting gear group
W_gear_basic	real	basic (not RP8A)
W_gear_retract	real	retraction (not RP8A)
W_gear_crash	real	crashworthiness (not RP8A)
W_nacelle	real	engine section or nacelle group
W_nac_engsupt	real	engine support (not RP8A)
W_nac_cowling	real	engine cowling (not RP8A)
W_nac_pylon	real	pylon support (not RP8A)
W_airind	real	air induction group
W_propulsion	real	<b>PROPULSION GROUP</b>
W_engsys	real	engine system
W_engine	real	engine
W_exhaust	real	exhaust system
W_acc	real	accessories (not RP8A)
W_propeller	real	propeller/fan installation
W_prop_blade	real	blades (not RP8A)
W_prop_hub	real	hub & hinge (not RP8A)
W_prop_supt	real	rotor supports (not RP8A)
W_prop_duct	real	rotor/fan duct (not RP8A)
W_fuelsys	real	fuel system
W_fuel_tank	real	tanks and support
W_fuel_plumb	real	plumbing
W_drive	real	drive system
W_drive_box	real	gear boxes
W_drive_xmsn	real	transmission drive

W_drive_rtrsft	real	rotor shaft
W_drive_brake	real	rotor brake (not RP8A)
W_drive_clutch	real	clutch (not RP8A)
W_drive_gas	real	gas drive
W equip	real	SYSTEMS AND EQUIPMENT
Wfltcont	real	flight controls group
W_fc_cockpit	real	cockpit controls
W_fc_afcs	real	automatic flight control system
W_fc_system	real	system controls
W_fc_fw	real	fixed wing systems
W_fc_fw_nonboost	real	non-boosted (not RP8A)
W_fc_fw_mech	real	boost mechanisms (not RP8A)
W_fc_rw	real	rotary wing systems
W_fc_rw_nonboost	real	non-boosted (not RP8A)
W_fc_rw_mech	real	boost mechanisms (not RP8A)
W_fc_rw_boost	real	boosted (not RP8A)
W_fc_cv	real	conversion systems
W_fc_cv_nonboost	real	non-boosted (not RP8A)
W_fc_cv_mech	real	boost mechanisms (not RP8A)
W_auxpower	real	auxiliary power group
W_instrument	real	instruments group
W_hydraulic	real	hydraulic group
W_hyd_fw	real	fixed wing (not RP8A)
W_hyd_rw	real	rotary wing (not RP8A)
W_hyd_cv	real	conversion (not RP8A)
W_hyd_eq	real	equipment (not RP8A)
W_pneumatic	real	pneumatic group
W_electrical	real	electrical group
W_elect_aircraft	real	aircraft (not RP8A)
W_elect_deice	real	anti-icing (not RP8A)
W_avionics	real	avionics group (mission equipment)
W_arm	real	armament group
W_armprov	real	armament provisions (not RP8A)
W_armor	real	armor (not RP8A)

W_furnish	real	furnishings & equipment group
W_environ	real	environmental control group
W_deice	real	anti-icing group
W_load	real	load & handling group
W_vib	real	VIBRATION (not RP8A)
W_cont	real	CONTINGENCY
W_fixUL	real	FIXED USEFUL LOAD
W_fixUL_crew	real	crew
W_fixUL_fluid	real	fluids (oil, unusable fuel) (not RP8A)
W_fixUL_auxtank	real	auxiliary fuel tanks
W_fixUL_other	real	other fixed useful load (not RP8A)
W_fixUL_equip	real	equipment increment (not RP8A)
W_fixUL_foldkit	real	folding kit (not RP8A)
W_fixUL_extkit	real	wing extension kit (not RP8A)
W_fixUL_wingkit	real	wing kit (not RP8A)
W_fixUL_otherkit	real	other kit (not RP8A)
Wpayload	real	PAYLOAD
Wfuel	real	USABLE FUEL
Wfuel_std	real	standard tanks (not RP8A)
Wfuel_aux	real	auxiliary tanks (not RP8A)
Wscaled	real	scaled weight (sum all K=3 in operating weight)
Wfixed	real	fixed weight (sum all K=2 in operating weight)
Wfeature	real	military features in empty weight
WO	real	OPERATING WEIGHT = weight empty + fixed useful load
WUL	real	USEFUL LOAD = fixed useful load + payload + usable fuel
GW	real	GROSS WEIGHT = weight empty + useful load = operating weight + payload + usable fuel

---

follows SAWE RP8A Group Weight Statement, except as noted  
 typical only lowest elements of hierarchy specified, others obtained by summation

set status flag when define weight

can define weights (k=2 or 3) at any level, ignore child weights if not lowest level  
 when print weight statement, designate all fixed (ie input) quantities

usage:

set all W=K=0; put W, with K=2 or 3

then fill structure: if K=0 and some child defined/sum, then  $W = \sum(\text{child})$  and K=1

addition or increment sums all elements, with status Kt of total as follows

	Ka =	0	1	2	3
Kb = 0		0	1	2	3
Kb = 1		1	1	3	3
Kb = 2		2	3	2	3
Kb = 3		3	3	3	3

Status (0 none; 1 sum of child; 2 defined, fixed (input); 3 defined, not fixed (scaled, wt eq; or composite))

KE	int	WEIGHT EMPTY
K_structure	int	STRUCTURE
K_wing	int	wing group
K_wing_basic	int	basic structure
K_wing_secondary	int	secondary structure
K_wing_fair	int	fairings (not RP8A)
K_wing_fit	int	fittings (not RP8A)
K_wing_fold	int	fold/tilt (not RP8A)
K_wing_control	int	control surfaces
K_rotor	int	rotor group
K_rotor_blade	int	blade assembly
K_rotor_hub	int	hub & hinge
K_rotor_basic	int	basic (not RP8A)
K_rotor_shaft	int	inter-rotor shaft (not RP8A)
K_rotor_fair	int	fairing/spinner (not RP8A)
K_rotor_fold	int	blade fold (not RP8A)
K_rotor_supt	int	rotor support structure (not RP8A)
K_rotor_duct	int	duct (not RP8A)
K_tail	int	empennage group
K_Htail	int	horizontal tail (not RP8A)

Structure: Weight

K_Htail_basic	int	basic (not RP8A)
K_Htail_fold	int	fold (not RP8A)
K_Vtail	int	vertical tail (not RP8A)
K_Vtail_basic	int	basic (not RP8A)
K_Vtail_fold	int	fold (not RP8A)
K_tailrotor	int	tail rotor (not RP8A)
K_tr_blade	int	blades
K_tr_hub	int	hub & hinge
K_tr_supt	int	rotor supports
K_tr_duct	int	rotor/fan duct
K_fuselage	int	fuselage group
K_fus_basic	int	basic (not RP8A)
K_fus_wingfold	int	wing & rotor fold/retraction (not RP8A)
K_fus_tailfold	int	tail fold/tilt (not RP8A)
K_fus_mar	int	marinization (not RP8A)
K_fus_press	int	pressurization (not RP8A)
K_fus_crash	int	crashworthiness (not RP8A)
K_gear	int	alighting gear group
K_gear_basic	int	basic (not RP8A)
K_gear_retract	int	retraction (not RP8A)
K_gear_crash	int	crashworthiness (not RP8A)
K_nacelle	int	engine section or nacelle group
K_nac_engsupt	int	engine support (not RP8A)
K_nac_cowling	int	engine cowling (not RP8A)
K_nac_pylon	int	pylon support (not RP8A)
K_airind	int	air induction group
K_propulsion	int	PROPULSION GROUP
K_engsys	int	engine system
K_engine	int	engine
K_exhaust	int	exhaust system
K_acc	int	accessories (not RP8A)
K_propeller	int	propeller/fan installation
K_prop_blade	int	blades (not RP8A)
K_prop_hub	int	hub & hinge (not RP8A)



K_prop_supt	int	rotor supports (not RP8A)
K_prop_duct	int	rotor/fan duct (not RP8A)
K_fuelsys	int	fuel system
K_fuel_tank	int	tanks and support
K_fuel_plumb	int	plumbing
K_drive	int	drive system
K_drive_box	int	gear boxes
K_drive_xmsn	int	transmission drive
K_drive_rtrsft	int	rotor shaft
K_drive_brake	int	rotor brake (not RP8A)
K_drive_clutch	int	clutch (not RP8A)
K_drive_gas	int	gas drive
K equip	int	SYSTEMS AND EQUIPMENT
Kfltcont	int	flight controls group
K_fc_cockpit	int	cockpit controls
K_fc_afcs	int	automatic flight control system
K_fc_system	int	system controls
K_fc_fw	int	fixed wing systems
K_fc_fw_nonboost	int	non-boosted (not RP8A)
K_fc_fw_mech	int	boost mechanisms (not RP8A)
K_fc_rw	int	rotary wing systems
K_fc_rw_nonboost	int	non-boosted (not RP8A)
K_fc_rw_mech	int	boost mechanisms (not RP8A)
K_fc_rw_boost	int	boosted (not RP8A)
K_fc_cv	int	conversion systems
K_fc_cv_nonboost	int	non-boosted (not RP8A)
K_fc_cv_mech	int	boost mechanisms (not RP8A)
K_auxpower	int	auxiliary power group
K_instrument	int	instruments group
K_hydraulic	int	hydraulic group
K_hyd_fw	int	fixed wing (not RP8A)
K_hyd_rw	int	rotary wing (not RP8A)
K_hyd_cv	int	conversion (not RP8A)
K_hyd_eq	int	equipment (not RP8A)

Structure: Weight

K_pneumatic	int	pneumatic group
K_electrical	int	electrical group
K_elect_aircraft	int	aircraft (not RP8A)
K_elect_deice	int	anti-icing (not RP8A)
K_avionics	int	avionics group (mission equipment)
K_arm	int	armament group
K_armprov	int	armament provisions (not RP8A)
K_armor	int	armor (not RP8A)
K_furnish	int	furnishings & equipment group
K_environ	int	environmental control group
K_deice	int	anti-icing group
K_load	int	load & handling group
K_vib	int	VIBRATION (not RP8A)
K_cont	int	CONTINGENCY
K_fixUL	int	FIXED USEFUL LOAD
K_fixUL_crew	int	crew
K_fixUL_fluid	int	fluids (oil, unusable fuel) (not RP8A)
K_fixUL_auxtank	int	auxiliary fuel tanks
K_fixUL_other	int	other fixed useful load (not RP8A)
K_fixUL_equip	int	equipment increment (not RP8A)
K_fixUL_foldkit	int	folding kit (not RP8A)
K_fixUL_extkit	int	wing extension kit (not RP8A)
K_fixUL_wingkit	int	wing kit (not RP8A)
K_fixUL_otherkit	int	other kit (not RP8A)
Kpayload	int	PAYLOAD
Kfuel	int	USABLE FUEL
Kfuel_std	int	standard tanks (not RP8A)
Kfuel_aux	int	auxiliary tanks (not RP8A)
KO	int	OPERATING WEIGHT = weight empty + fixed useful load
KUL	int	USEFUL LOAD = fixed useful load + payload + usable fuel
KGW	int	GROSS WEIGHT = weight empty + useful load = operating weight + payload + usable fuel