NDARC
NASA Design and Analysis of Rotorcraft

Input

Release 1.6
February 2012

Wayne Johnson
NASA Ames Research Center, Moffett Field, CA

Contents

Chapter 1

Data Structures and Input

1–1 Overview

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

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

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

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

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.

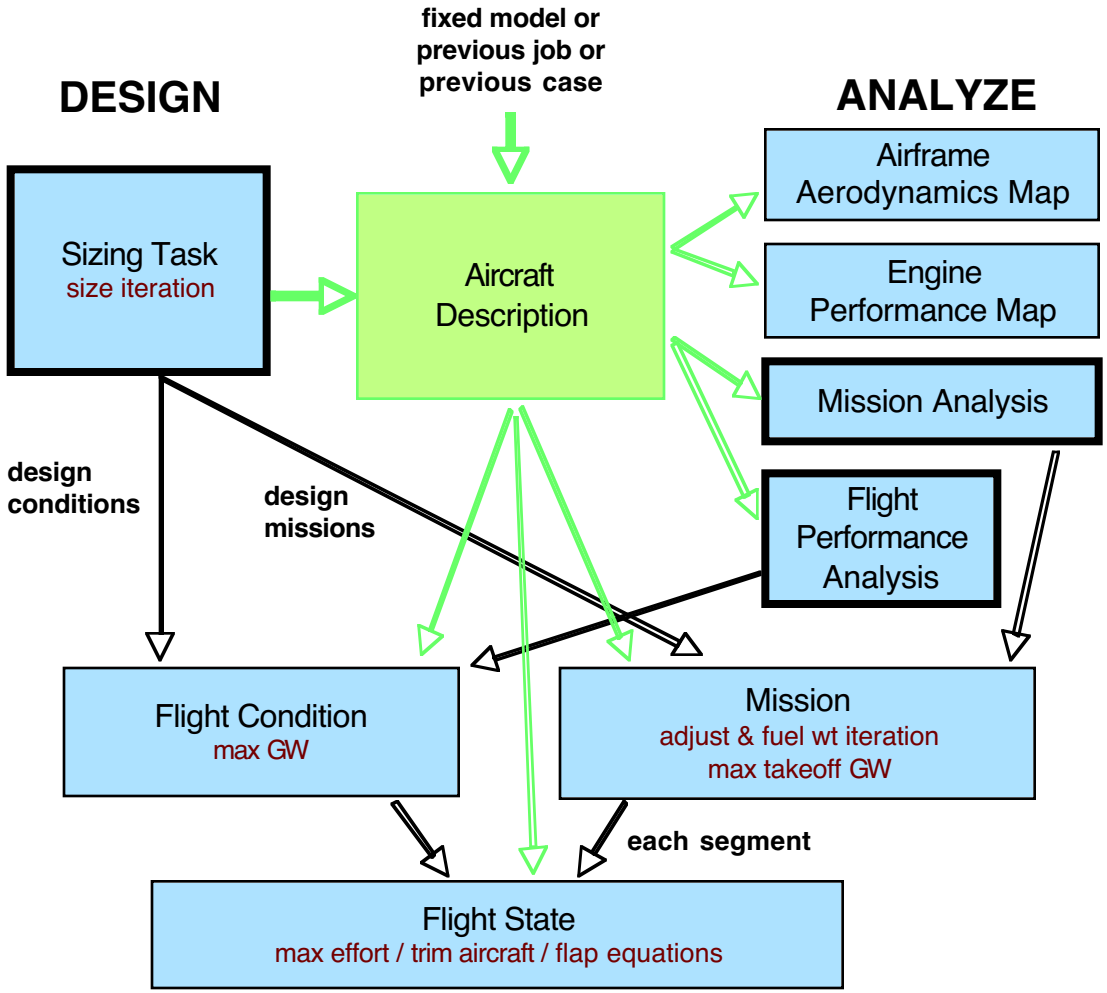

Figure 1-1 Outline of NDARC tasks.

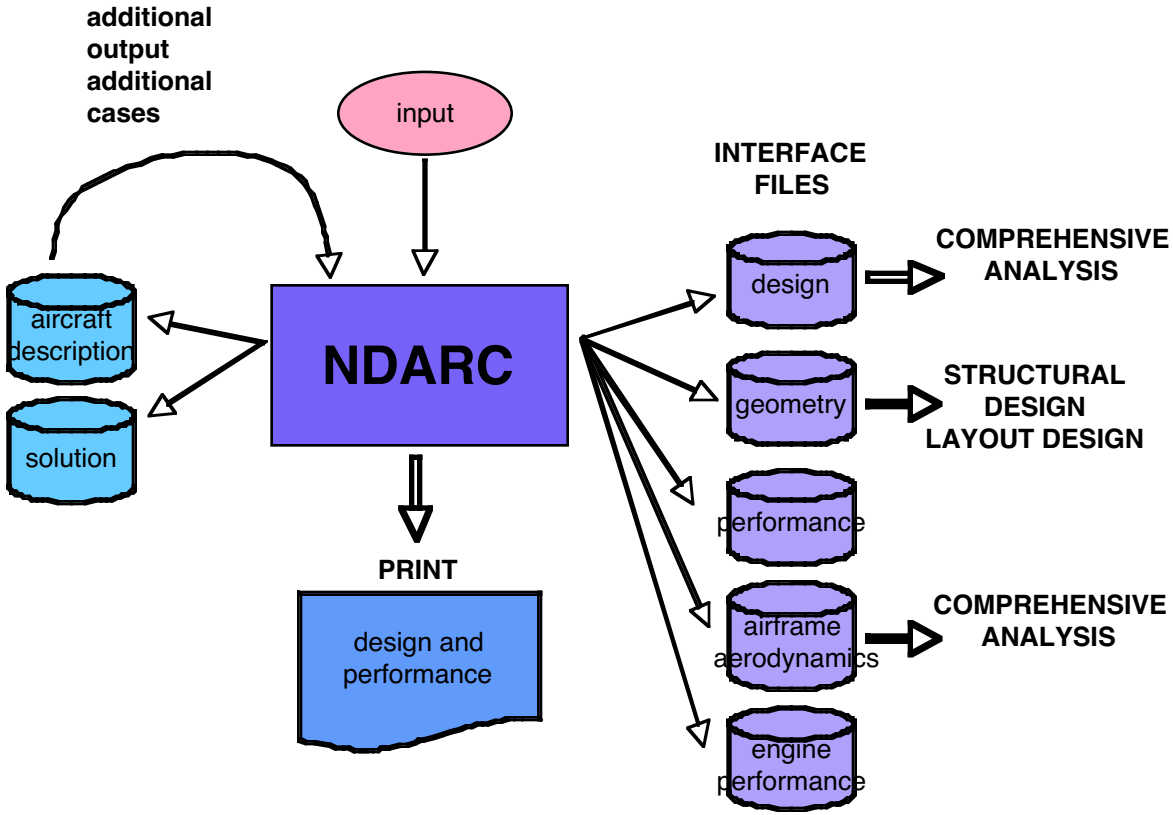

Figure 1-2 NDARC Interfaces.

```
&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

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; propulsion group number for QUANT='EngineGroup', engine model number for QUANT='EngineParam'; value is 1 if absent; input variable can be PARENT, PROPULSION, or ENGINEGROUP.

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 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: propulsion group number for QUANT='EngineGroup', 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).

Table 1-2. ACTION options.

ACTION	keyword	QUANT	function
Primary Input Only			
blank	—	blank	open and read secondary input file, name = FILE
'open file'	file, open		open and read secondary input file, name = FILE
'load aircraft'	aircraft, desc		load aircraft description file, name = FILE
'read solution'	solution	'text'	read complete solution file, name = FILE (text)
'read solution'	solution	not 'text'	read complete solution file, name = FILE (binary)
'end of case'	end+case		stop case input, execute case
'end of job'	end+job, quit		stop job input, execute case, exit code
Primary or Secondary Input			
blank	—	'structure'	read VALUE namelist
'read namelist'	list	'structure'	read VALUE namelist
'copy input'	copy	'structure'	copy input from source (same structure), SOURCE=SRCnumber
'initialize'	init	'structure'	set structure variables to default values
'delete all'	del+all	'structure'	delete all conditions or missions
'delete one'	del+one	'structure'	delete one condition or mission
'delete last'	del+last	'structure'	delete last conditions or missions
'configuration'	config		set input based on aircraft configuration
'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	
'Force n'	Force	nForce	
'Wing n'	Wing, AWing, WWing, WWingTR	nWing	
'Tail n'	Tail, ATail, WTail	nTail	
'FuelTank'	FuelTank, WTank		
'Propulsion n'	PropGroup, WDrive	nPropulsion	
'EngineGroup n'	EngineGroup, DEngSys, WEngSys	nEngineGroup	Propulsion number
'EngineModel n'	EngineModel, EngineParam	nEngine	
'EngineParamN n'	EngineParam	nspeed	EngineModel number
'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',create='time-date',title='xxx',version='0.0',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',create='time-date',title='xxx',version='0.0',modification='xxx',&END
&VALUE_ADIMEN nrotor=m,force=m,nwing=m,ntail=m,npropulsion=m,nenginemodel=m,nenginegroup=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',create='time-date',title='xxx',version='0.0',modification='xxx',&END
&VALUE_ADIMEN nrotor=m,force=m,nwing=m,ntail=m,npropulsion=m,nenginemodel=m,nenginegroup=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`, `Force`, `Wing`, or `Tail` 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). A technology factor or geometry variable in the `EngineGroup` input becomes a two-dimensional array in the `TechFactors` or `Geometry` input, the first argument being the engine group number and the second argument the propulsion group number.

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.

Switches in the case input control print of input parameters. Optionally the input parameters can be printed only if not the default value, or only if changed from the last case.

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
  - Cases
  - Size
    - SizeParam
    - FltCond(nfltmax)
    - FltState(nfltmax)
    - Mission(nmissmax)
      - MissParam
      - MissSeg(nsegmax)
      - FltState(nsegmax)
  - OffDesign
    - OffParam
    - Mission(nmissmax)
      - MissParam
      - MissSeg(nsegmax)
      - FltState(nsegmax)
  - Performance
    - PerfParam
    - FltCond(nfltmax)
    - FltState(nfltmax)
  - MapEngine
  - MapAero
  - Solution
  - Cost
    - CostCTM
  - Aircraft
    - [Location]loc_cg
    - Weight
  - Systems
    - Weight
    - WFltCont
    - WDeIce
- Fuselage
  - [Location]loc_fuselage
  - AFuse
  - Weight
  - WFuse
- LandingGear
  - [Location]loc_gear
  - AGear
  - Weight
  - WGear
- Rotor(nrotormax)
  - [Location]loc_rotor
  - [Location]loc_pylon
  - [Location]loc_pivot
  - [Location]loc_nac
  - PRotorInd
  - PRotorPro
  - PRotorTab
  - IRotor
  - DRotor
  - Weight
  - WRotor
- Force(nforcemax)
  - [Location]loc_force
  - Weight
- Wing(nwingmax)
  - [Location]loc_wing
  - AWing
  - Weight
  - WWing
  - WWingTR
- Tail(ntailmax)
  - [Location]loc_tail
  - ATail
  - Weight
  - WTail
- FuelTank
  - [Location]loc_auxtank(ntankmax)
  - Weight
  - WTank
- Propulsion(npropmax)
  - PropGroup
    - Weight
    - WDrive
  - EngineGroup(nengmax)
    - [Location]loc_engine
    - DEngSys
    - Weight
    - WEngSys
- EngineModel(nengmax)
  - [EngineParam]Param
  - [EngineParam]ParamN(nspeedmax)

- FltState(nfltmax)
  - FltAircraft
  - FltFuse
  - FltGear
  - FltRotor(nrotormax)
  - FltForce(nforcemax)
  - FltWing(nwingmax)
  - FltTail(ntailmax)
  - FltTank
  - FltPropulsion(npropmax)
    - FltProp, FltEngn(nengmax)

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, nForce=0, nWing=0, nTail=2; nPropulsion=1, nEngineGroup=1; nEngineModel=1

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

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

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

e) Fuel Tank: place=1 (internal tank), Mauxtanksize=1, WTank%ntank=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), Idrag=0. (not tilt); DRotor%SET_Dspin=1, DRotor%DoQ_spin=0. (no spinner drag)
Weight: WRotor%MODEL_config=1 (rotor), WRotor%KIND_rotor=2 (not tilting)
Control:
INPUT_coll=0, INPUT_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/σ)
KIND_lngcyc=1, KIND_latcyc=1 (1 for TPP tilt, 2 for hub moment, 3 for lift offset)
KIND_tilt=0 (fixed shaft)

g) Force
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)

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

i) 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)

j) Propulsion: nGear=1, STATE_gear_wt=1

k) Engine Group
Configuration: INPUT_gear=1 (gear ratio from N_spec), SET_power=0 (sized), fPsize=1., direction='x', SET_geom=0 (standard position)
Drag: MODEL_drag=1, Idrag=0. (not tilt)
Control:
INPUT_incid=0, INPUT_yaw=0 (no connection to aircraft controls)
T_incid=0., T_yaw=0. (all controls, all states)

2–2 Helicopter

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

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

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

2–3 Tandem

a) Components: nTail=0 (no tail)

b) Fuel Tank: place=2 (sponson)

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

d) 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_lngcyc=1 (rotor control connection to aircraft controls)
control: T_coll(1,1)=1., T_coll(4,1)=r, T_latcyc(2,1)= $-\,r$, T_lngcyc(3,1)=-1.
Second rotor is upper rotor: otherRotor=1, rotate=-Rotor(1)%rotate
rotation: $r = 1$, if (Rotor(1)%rotate < 0) $r = -1$; $r = -r$
control: INPUT_coll=1, INPUT_latcyc=1, INPUT_lngcyc=1 (rotor control connection to aircraft controls)
control: T_coll(1,1)=1., T_coll(4,1)=r, T_latcyc(2,1)= $-\,r$, T_lngcyc(3,1)=-1.
Performance: PRotorInd%MODEL_twin='coaxial', PRotorInd%Kh_twin=1., PRotorInd%Kf_twin=0.85, IRotor%MODEL_int_twin=2
Drag: SET_Sspin=1, Swet_spin=0., DRotor%SET_Dspin=1, DRotor%DoQ_spin=0., DRotor%CD_spin=0. (no spinner drag)

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

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

2–5 Tiltrotor

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

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

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

d) Landing Gear: KIND_LG=1 (retractable)

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

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

g) Wing
Configuration: fDGW=1., nRotorOnWing=2, RotorOnWing(1)=1, RotorOnWing(2)=2, SET_ext=0
Control: KIND_flaperon=3 (independent), nVincid=1
    INPUT_flap=1, INPUT_flaperon=1, INPUT_aileron=1 (wing control connection to aircraft controls)
    T_aileron(2,2)=-1. (airplane mode aileron connection to aircraft control 'latcyc')
    T_flap(6,1)=1., T_flap(6,2)=1. (flap direct control)
    T_flaperon(7,1)=1., T_flaperon(7,2)=1. (flaperon direct control)
    T_aileron(9,1)=1., T_aileron(9,2)=1. (aileron direct control)
Weight: WWing%MODEL_wing=3 (tiltrotor)

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

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

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

Parameters

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

Common: Job

Variable	Type		Description	Default
			NDARC	
		+	Initialization	
INIT_input	int	+	input parameters (0 default, 1 last case input, 2 last case solution)	1
INIT_data	int	+	other parameters (0 default, 1 start of last case, 2 end of last case)	0
			INIT_input: if default, all input variables set to default values if last-case-input, then case inherits input at beginning of previous case if last-case-solution, then case inherits input at end of previous case use INIT_input=2 to analyze case #1 design in subsequent cases INIT_data: if always start-last-case, then case starts from default if default, all other variables set to default values	
		+	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

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

Structure: Cases

Variable	Type		Description	Default
		+	Case Description	
title	c*100	+	title	
subtitle1	c*100	+	subtitle	
subtitle2	c*100	+	subtitle	
subtitle3	c*100	+	subtitle	
notes	c*1000	+	notes	
ident	c*32	+	identification	
		+	Case Tasks (0 for none)	
TASK_Size	int	+	size aircraft for design conditions	1
TASK_Mission	int	+	mission analysis	1
TASK_Perf	int	+	flight performance analysis	1
TASK_Map_engine	int	+	map of engine performance	0
TASK_Map_aero	int	+	map of airframe aerodynamics	0
		+	Write Input Parameters	
WRITE_input	int	+	selection (0 none, 1 all, 2 not default, 3 not last case, 4 group selection)	3
		+	group selection	
WRITE_input_Job	int	+	Job	0
WRITE_input_Case	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_Force(nforcemax)	int	+	Force	0
WRITE_input_Wing(nwingmax)	int	+	Wing	0
WRITE_input_Tail(ntailmax)	int	+	Tail	0
WRITE_input_FuelTank	int	+	FuelTank	0
WRITE_input_Propulsion(npropmax)	int	+	Propulsion	0
WRITE_input_EngineModel(nengmax)	int	+	EngineModel	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
		+	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`)	' '

		+	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_perf	int	+	performance (0 standard format, n for special)	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

---

		+	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^2 or m^2; 1 ft^2, 2 m^2)	0
Units_ROC	int	+	rate of climb units (0 ft/min; 1 ft/sec, 2 m/sec)	0
		+	units for parameters	
Units_Dscale	int	+	input D/q scaled with gross weight (0 analysis default, 1 English, 2 SI)	0

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)

Default units for flight condition and mission: override with Units_xxx
- speed in knots, time in minutes, distance in nm, ROC in ft/min

Structure: Size

Variable	Type		Description	Default
		+	Size Aircraft for Design Conditions and Missions	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Sizing Method	
SIZE_perf(npropmax)	c*16	+	quantity sized from performance	'engine'
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	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

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)
'input' = input Wfuel_cap
'miss' = calculate from mission fuel burned
Wfuel_cap = max(fFuelCap*(maximum mission fuel), (maximum mission fuel)+(reserve fuel))

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$\sum(N_{\rm eng}P_{\rm eng})$
'Pav' = from engine power available at transmission sizing conditions and missions (DESIGN_xmsn)
fPlimit_ds$(\Omega_{\rm ref}/\Omega_{\rm prim})\sum(N_{\rm eng}P_{av})$
'Preq' = from engine power required at transmission sizing conditions and missions (DESIGN_xmsn)
fPlimit_ds$(\Omega_{\rm ref}/\Omega_{\rm prim})\sum(N_{\rm eng}P_{req})$
engine shaft rating also uses PropGroup%SET_limit_es
rotor shaft rating 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 (FltAircraft%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: SIZE_perf='none', SET_rotor=2*'radius+Vtip+sigma'
FIX_DGW=1, SET_tank='input', SET_limit_ds='input', SET_SDGW='input', SET_WMTO='input'
and perhaps Rotor%SET_limit_rs=0, PropGroup%SET_limit_es=2*0
with wing panels: SET_wing='WL+panel', Wing%SET_panel='width+taper','span+taper'

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

---

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

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

---

Structure: OffDesign

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

mission analysis input required if Cases%TASK_Mission=1

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

Structure: Performance

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

flight performance analysis input required if Cases%TASK_Perf=1

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

Structure: MapEngine

Variable	Type		Description	Default
		+	Map of Engine Performance	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Identification	
kPropulsion	int	+	propulsion group	1
kEngineGroup	int	+	engine group	1
KIND_map	int	+	Kind (1 performance, 2 model)	1
			engine map input required if Cases%TASK_Map_engine=1 only performance parameters or model parameters used	
		+	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 τ or temperature increment ΔT (Units_temp)	
temp_min	real	+	minimum	0.
temp_max	real	+	maximum	100.
temp_inc	real	+	increment	10.

		+	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 FltAircraft%SET_atmos for other options (polar, tropical, and hot days)

		+	Model	
		+	flight speeds V(TAS, Units_vel)	
nV_model	int	+	number (maximum 10)	1
V_model(10)	real	+	values	0.
V_min	real	+	minimum	0.
V_max	real	+	maximum	400.
V_inc	real	+	increment	50.

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

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

Structure: MapAero

Variable	Type		Description	Default
		+	Map of Airframe Aerodynamics	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Tables	
KIND_table	int	+	kind (1 one-dimensional, 2 multi-dimensional)	1
		+	aerodynamic loads (0 for components off)	
SET_fuselage	int	+	fuselage and landing gear	1
SET_tail	int	+	tails	1
SET_wing	int	+	wings	1
SET_rotor	int	+	rotors	1
SET_engine	int	+	engines and fuel tank	1
			airframe aerodynamics map input required if Cases%TASK_Map_aero=1	
			multi-dimensional: generate 6 files of three-dimensional tables one file for each `load=DRAG, SIDE, LIFT, ROLL, PITCH, YAW` filename=`FILE_aero//load` or `AEROn//load` one-dimensional: generate 1 file of all six loads function of single independent variable = var_lift(1)	
		+	Operating Condition	
STATE_control	int	+	aircraft control state	1
STATE_LG	c*12	+	landing gear state	'retract'
Nauxtank(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
control(ncontmax)	real	+	aircraft controls	0.
tilt	real	+	tilt	0.
alpha	real	+	angle of attack α	0.
beta	real	+	sideslip angle β	0.

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

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

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

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

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.
SET_alt	int	+	altitude (0 input, 1 from KIND_source)	0
		+	source for gross weight and altitude	
KIND_source	int	+	kind (1 size mission, 2 size condition, 3 off design mission, 4 performance condition)	1
kSource	int	+	mission or condition number	0
kSegment	int	+	segment number	0
seg_source	int	+	segment (1 start, 2 midpoint)	1
SET_UL	c*12	+	useful load	'pay'
Wpay	real	+	input payload weight $W_{\rm pay}$ (Units_pay)	0.
dFuel	real	+	fuel weight increment	0.
fFuel	real	+	fuel capacity factor	1.
SET_auxtank	int	+	auxiliary fuel tanks (1 adjust Nauxtank, 2 only increase)	1
mauxtank	int	+	tank size changed (–1 first, –2 first size already used, m for m-th size)	-1
dNauxtank	int	+	number tanks added or dropped	1
Nauxtank(ntankmax)	int	+	number of auxiliary fuel tanks $N_{\rm auxtank}$ (each aux tank size)	
		+	fixed useful load	
dWcrew	real	+	crew weight increment	0.
dWequip	real	+	equipment weight increment	0.

SET_foldkit	int	+	folding kit on aircraft (0 none, 1 present)	1
SET_extkit(nwingmax)	int	+	wing extension kit on aircraft (0 none, 1 present)	1
SET_wingkit(nwingmax)	int	+	wing kit on aircraft (0 none, 1 present)	1
SET_otherkit	int	+	other kit on aircraft (0 none, 1 present)	0
DESIGN_engine	int	+	design condition for power (1 to use for engine 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: $P_{req} = \text{fPav}P_{av} + \text{dPav}$
$\min((fP_{avPG} + d) - P_{reqPG}) = 0$, over all propulsion groups
'maxQ' = maximum GW for transmission torque equal limit: zero torque margin
$\min(P_{\text{limit}} - P_{req}) = 0$, over all propulsion groups, engine groups, and rotors

'maxPQ', 'maxQP' = maximum GW for power required equal specified power and transmission torque equal limit
most restrictive of power and torque margins
'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 (Wpay); fuel weight fallout
'fuel' = input fuel weight (dFuel, fFuel, Nauxtank); payload weight fallout
'pay+fuel' = input payload and fuel weights; gross weight fallout

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

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

auxiliary fuel tanks: SET_auxtank used for fallout fuel weight (SET_UL='pay')
otherwise number of auxiliary fuel tanks fixed at input value

---

		+	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, Wpay, dFuel, fFuel, 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, fPower, DoQ_pay, fDoQ_pay, DoQV_pay, dSLcg, dBLcg, dWLcg, trim_targetn
n = propulsion group (Vtip, dPav, fPav), control number, or trim quantity; 1 if absent
for fPower, value is factor on input fPower for all engine groups, all propulsion groups

Structure: Mission

Variable	Type		Description	Default
		+	Mission Profile	
title	c*100	+	title	
label	c*8	+	label	
		+	Specification	
SET_GW	c*16	+	mission takeoff gross weight W_G	'pay+miss'
GW	real	+	input gross weight	0.
dGW	real	+	gross weight increment	0.
fGW	real	+	gross weight factor	1.
SET_UL	c*16	+	useful load	'pay+miss'
Wpay	real	+	input takeoff payload weight W_{pay} (Units_pay)	0.
SET_pay	c*16	+	payload changes	'delta'
dFuel	real	+	fuel weight increment	0.
fFuel	real	+	fuel capacity factor	1.
SET_auxtank	int	+	auxiliary fuel tanks (1 adjust Nauxtank, 2 only increase, 3 increase at start and drop)	1
mauxtank	int	+	tank size changed (–1 first, –2 first size already used, m for m-th size)	-1
dNauxtank	int	+	number tanks added or dropped	1
Nauxtank(ntankmax)	int	+	number of auxiliary fuel tanks N_{auxtank} (each aux tank size)	
		+	fixed useful load	
SET_foldkit	int	+	folding kit on aircraft (0 none, 1 present)	1
SET_reserve	int	+	fuel reserve (1 fraction mission fuel, 2 fraction fuel capacity, 3 only mission segments)	1
fReserve	real	+	fuel reserve fraction f_{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_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)
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
        $\min(P_{\text{limit}} - P_{req}) = 0$, over all propulsion groups, engine groups, and rotors
    'maxPQ', 'maxQP' = maximum GW for power required equal specified power and transmission torque equal limit
        at mission segment MaxGW, minimum gross weight of designated segments
        most restrictive of power and torque margins
    '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)

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))

auxiliary fuel tanks:
SET_auxtank options: 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} at start
for fallout (SET_UL = 'pay'), adjust W_{fuel} with change in Nauxtank (fixed $W_G - W_{\mathrm{pay}} = W_O + W_{\mathrm{fuel}}$)
for all SET_UL, adjust W_O with change in Nauxtank
fuel tank design mission: Nauxtank=0, allow W_{fuel} to exceed tank capacity

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

		+	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

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_fuel	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	real	+	fuel weight change	0.
rWfuel	real	+	threshold fraction	0.
mWfuel	real	+	threshold weight	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.
		+	useful load	
xWpay	real	+	payload weight change (Units_pay)	0.
		+	fixed useful load	
dWcrew	real	+	crew weight increment	0.
dWequip	real	+	equipment weight increment	0.
SET_extkit(nwingmax)	int	+	wing extension kit on aircraft (0 none, 1 present)	1
SET_wingkit(nwingmax)	int	+	wing kit on aircraft (0 none, 1 present)	1
SET_otherkit	int	+	other kit on aircraft (0 none, 1 present)	0
SET_alt	int	+	altitude at start of segment (0 input, 1 from previous segment)	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.

---

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='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_fuel, refuel: change at start of segment
- SET_fuel = 1: fill all tanks (including any auxiliary tanks installed)
- SET_fuel = 2: add fuel weight xWfuel
- SET_fuel = 3: drop fuel weight xWfuel
- SET_fuel = 4: if below fraction rWfuel of fuel capacity (including auxiliary tanks), fill all tanks
- SET_fuel = 5: if below fraction rWfuel of fuel capacity (including auxiliary tanks), add xWfuel
- SET_fuel = 6: if below weight mWfuel, fill all tanks

SET_fuel = 7: if below weight mWfuel, add xWfuel
added fuel limited by capacity; not used for first segment
xWfuel positive (add or drop determined by SET_fuel)

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)

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

---

takeoff distance calculation: set of consecutive kind='takeoff' segments
first segment identified by SET_takeoff='start' ($V = 0$)
last segment if next segment is not kind='takeoff', or is SET_takeoff='start'
takeoff segment kind
SET_takeoff='start', 'ground run' (keyword = ground or run), 'engine fail' (keyword = eng or fail)
SET_takeoff='liftoff', 'rotation', 'transition', 'climb', 'brake'
each segment requires appropriate configuration, trim option, max effort specification
not use dist, time, reserve, adjust, range_credit, SET_fuel, 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, friction 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

Structure: FltState

Variable	Type		Description	Default
		+	Flight State	
		+	Specification	
SET_max	int	+	maximum effort performance (maximum 2, 0 to analyze specified condition)	0
max_quant(2)	c*12	+	quantity	' '
max_var(2)	c*12	+	variable	' '
max_limit(2)	int	+	switch quantity if exceed limit (0 not, 1 power margin, 2 torque margin, 3 both)	0
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 θ_V (deg)	0.
side	real	+	sideslip angle ψ_V (deg)	0.
		+	aircraft motion	
SET_pitch	int	+	pitch motion specification (0 Aircraft value, 1 FltState input)	1
SET_roll	int	+	roll motion specification (0 Aircraft value, 1 FltState input)	1
pitch	real	+	pitch θ_F	0.
roll	real	+	roll ϕ_F	0.
SET_turn	int	+	turn specification (0 zero, 1 turn rate, 2 load factor, 3 bank angle)	0
rate_turn	real	+	turn rate $\dot{\psi}_F$ (deg/sec)	0.
nz_turn	real	+	load factor n (g)	1.
bank_turn	real	+	bank angle ϕ_F (deg)	0.
SET_pullup	int	+	pullup specification (0 zero, 1 pitch rate, 2 load factor)	0
rate_pullup	real	+	pitch rate $\dot{\theta}_F$ (deg/sec)	0.
nz_pullup	real	+	load factor n (g)	1.
SET_acc	int	+	linear acceleration specification (0 zero, 1 acceleration, 2 load factor)	0
ax_linear	real	+	x-acceleration a_{ACx} (ft/sec^2 or m/sec^2)	0.
ay_linear	real	+	y-acceleration a_{ACy} (ft/sec^2 or m/sec^2)	0.

az_linear	real	+	z-acceleration a_{ACz} (ft/sec^2 or m/sec^2)	0.
nx_linear	real	+	x-load factor increment n_{Lx} (g)	0.
ny_linear	real	+	y-load factor increment n_{Ly} (g)	0.
nz_linear	real	+	z-load factor increment n_{Lz} (g)	0.
altitude	real	+	altitude h (Units_alt)	0.
SET_atmos	c*12	+	atmosphere specification	'std'
temp	real	+	temperature τ (Units_temp)	
dtemp	real	+	temperature increment ΔT (Units_temp)	0.
density	real	+	density ρ	
csound	real	+	speed of sound c_s	
viscosity	real	+	viscosity μ	
SET_GE	int	+	ground effect (0 OGE, 1 IGE)	0
HAGL	real	+	height of landing gear above ground level h_{LG}	999.
STATE_LG	c*12	+	landing gear state	'default'
STATE_control	int	+	aircraft control state	1
SET_control(ncontmax)	int	+	control specification (0 Aircraft value, 1 FltState input)	1
SET_coll	int	+	collective stick	1
SET_latcyc	int	+	lateral cyclic stick	1
SET_lngcyc	int	+	longitudinal cyclic stick	1
SET_pedal	int	+	pedal	1
SET_tilt	int	+	tilt (0 Aircraft value, 1 FltState input, 2 Aircraft conversion schedule)	1
control(ncontmax)	real	+	aircraft controls	
coll	real	+	collective stick c_{AC0}	0.
latcyc	real	+	lateral cyclic stick c_{ACc}	0.
lngcyc	real	+	longitudinal cyclic stick c_{ACs}	0.
pedal	real	+	pedal c_{ACp}	0.
tilt	real	+	tilt α_{tilt}	0.
SET_comp_control	int	+	use component control (0 for $c = Tc_{AC}$; 1 for $c = Tc_{AC} + c_0$)	1
SET_cg	int	+	center of gravity specification (0 baseline plus increment, 1 input)	0
dSLcg	real	+	stationline	0.
dBLcg	real	+	buttline	0.
dWLcg	real	+	waterline	0.
		+	Specification, each propulsion group	
SET_Vtip(npropmax)	c*12	+	rotor tip speed specification	'hover'

Vtip(npropmax)	real	+	tip speed	
c0_Vtip(npropmax)	real	+	$C_T/\sigma = c_0 - \mu c_1$, or μ, or M_{at}	
c1_Vtip(npropmax)	real	+	$C_T/\sigma = c_0 - \mu c_1$	
STATE_gear(npropmax)	int	+	drive system state	1
SET_Plimit(npropmax)	int	+	drive system limit (0 not applied to power available)	1
SET_Qlimit(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 $dP_{\rm acc}$	0.
		+	each engine group	
rating(nengmax,npropmax)	c*12	+	engine rating	'MCP'
fPower(nengmax,npropmax)	real	+	fraction of rated engine power available f_P (0. to 1.+)	1.
nEngInop(nengmax,npropmax)	int	+	number of inoperative engines $N_{\rm inop}$	0
SET_Preq(nengmax,npropmax)	int	+	power required distribution (0 P_{reqEG} fixed with fPower fraction, 1 proportional $P_{\rm eng}$)	1
STATE_IRS(nengmax,npropmax)	int	+	IR suppressor system state (0 off, hot exhaust; 1 on, suppressed exhaust)	0
		+	Performance	
DoQ_pay	real	+	payload forward flight drag increment D/q (Units_drag)	0.
fDoQ_pay	real	+	payload drag increment scaling with weight $\Delta(D/q)/W_{\rm pay}$ (Units_drag, Units_pay)	0.
DoQV_pay	real	+	payload vertical drag increment D/q (Units_drag)	0.
		+	Rotor (nonzero to supersede rotor model)	
Ki(nrotormax)	real	+	induced power factor κ	0.
cdo(nrotormax)	real	+	profile power mean c_d	0.
MODEL_Ftpp(nrotormax)	int	+	inplane forces, tip-path plane axes (1 neglect, 2 blade-element theory)	0
MODEL_Fpro(nrotormax)	int	+	inplane forces, profile (1 simplified, 2 blade element theory)	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_limit: switch quantity to power and/or torque margin if margin negative; useful for best range

max_quant='rotor(s) n' uses Rotor%CTs_steady, max_quant='rotor(t) n' uses Rotor%CTs_tran

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)
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',	$\min(Q_{\text{limit}} - Q_{req}) = 0$ (all limits)
power and torque margin	'PQ margin',	most restrictive
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'	$C_{L\max} - C_L = 0$

description	max_var	
horizontal velocity	'speed'	times fVel
vertical rate of climb	'ROC'	times fVel
altitude	'alt'	
aircraft angular rate	'pullup', 'turn'	Euler angle rates
aircraft acceleration	'xacc', 'yacc', 'zacc'	linear, airframe axes
aircraft acceleration	'xaccI', 'yaccI', 'zaccI'	linear, inertial axes
aircraft acceleration	'xaccG', 'yaccG', 'zaccG'	linear, ground axes

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 (θ_V, ψ_V)
  - sideslip positive aircraft moving to right, climb positive aircraft moving up
  - specify horizontal velocity, vertical rate of climb, and sideslip angle
- orientation body relative inertial axes defined by Euler angles, yaw/pitch/roll $(\psi_F, \theta_F, \phi_F)$
  - 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
'CT/s', 'mu', 'Mat' = use tip speed from input C_T/σ or μ or M_{at} (c0_Vtip, c1_Vtip)
'xxx+diam' = for variable diameter rotor, scale $V_{\rm tip}$ with radius ratio
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
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 rating and then drive system rating
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
nEnglnop, number inoperative engines: 1 for one engine inoperative (OEI), maximum nEngine
SET_Preq: for distribution of propulsion group power required among engine groups (use fPower)

STATE_trim, aircraft trim state: match IDENT_trim, 'none' for no trim
identfies 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

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 quadradic, 3 cubic)	3
init_fly	int	+	reinitialize aircraft controls (0 no, 1 force retrim)	0
trace_fly	int	+	trace iteration (0 for none)	0
		+	Maximum gross weight (flight condition or mission takeoff)	
method_maxgw	int	+	method (1 secant, 2 false position)	1
		+	convergence control	
niter_maxgw	int	+	maximum number of iterations	40
toler_maxgw	real	+	tolerance (fraction reference)	.002
relax_maxgw	real	+	relaxation factor	.5
perturb_maxgw	real	+	variable perturbation amplitude (fraction reference)	.02
maxderiv_maxgw	real	+	maximum derivative amplitude (0. for no limit)	0.
maxinc_maxgw	real	+	maximum increment fraction (0. for no limit)	0.
trace_maxgw	int	+	trace iteration (0 for none)	0
		+	Mission	
		+	convergence control	
niter_miss	int	+	maximum number of iterations	40
toler_miss	real	+	tolerance (fraction reference)	.01
relax_miss	real	+	relaxation factor (mission fuel)	1.
relax_range	real	+	relaxation factor (range credit)	1.
relax_gw	real	+	relaxation factor (max takeoff GW)	1.
trace_miss	int	+	trace iteration (0 for none)	0
		+	Size aircraft	
		+	convergence control	
niter_size	int	+	maximum number of iterations (performance loop)	40
niter_param	int	+	maximum number of iterations (parameter loop)	40
toler_size	real	+	tolerance (fraction reference)	.01

		+	relaxation factors	
relax_size	real	+	power or radius	1.
relax_DGW	real	+	gross weight	1.
relax_xmsn	real	+	drive system limit	1.
relax_wmto	real	+	WMTO and SDGW	1.
relax_tank	real	+	fuel tank capacity	1.
relax_thrust	real	+	rotor thrust	1.
		+	maximum increment fraction (0. for no limit)	
maxinc_size	real	+	power or radius	0.
maxinc_DGW	real	+	gross weight	0.
maxinc_xmsn	real	+	drive system limit	0.
maxinc_wmto	real	+	WMTO and SDGW	0.
maxinc_tank	real	+	fuel tank capacity	0.
maxinc_thrust	real	+	rotor thrust	0.
trace_size	int	+	trace iteration (0 for none, 2 for power)	0

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

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

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

		+	Flight condition and mission segment	
toler_check	real	+	check Preq, Qlimit, Wfuel (fraction reference)	.005

		+	Tolerance and perturbation scales	
KIND_Wscale	int	+	weight scale (1 design gross weight, 2 nominal C_T/σ)	1
KIND_Pscale	int	+	power scale (1 aircraft power, 2 derived from weight scale)	1
KIND_Lscale	int	+	length scale (1 rotor radius, 2 wing span, 3 fuselage length)	1
scaleRotor	int	+	rotor number	1
scaleWing	int	+	wing number	1

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

inflation: factor always used, even with internal table
CPI or DoD table, relative year_inf: $F_i = \text{inflation}\big(F_{\text{table}}(\text{year\_inf})/F_{\text{table}}(1994)\big)$
input factor, relative 1994: $F_i = \text{inflation}$

		+	Cost	
MODEL_cost	int	+	model (1 CTM)	1
FuelPrice	real	+	fuel price G ($/gallon or $/liter)	5.0
Npass	int	+	number of passengers N_{pass}	100
		+	DOC+I	
BlockHours	real	+	available block hours per year B	3751.
NonFlightTime	real	+	non-flight time per trip T_{NF} (min)	12.
Kcrew	real	+	crew cost factor K_{crew}	1.0
DepPeriod	real	+	depreciation period D (years)	15.
LoanPeriod	real	+	loan period L (years)	15.
IntRate	real	+	interest rate i (%)	8.

ResidValue	real	+	residual value V (%)	10.
Spares	real	+	spares per aircraft S (% purchase price)	25.
		+	Technology Factors	
TECH_cost_af	real	+	airframe χ_{AF}	0.87
TECH_cost_maint	real	+	maintenance χ_{maint}	1.0
		+	CTM rotorcraft cost model	
		+	Flyaway	
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.

rComp negative for cost reduction
cost factors correspond to year_inf

Structure: Aircraft

Variable	Type		Description	Default
		+	Aircraft	
title	c*100	+	title	
notes	c*1000	+	notes	
config	c*16	+	Configuration	'helicopter'
			config: identifies rotorcraft configuration config = 'rotorcraft', 'helicopter', 'tandem', 'coaxial', 'tiltrotor'	
		+	Aircraft Controls	
ncontrol	int	+	number of aircraft controls (maximum ncontmax)	4
IDENT_control(ncontmax)	c*16	+	labels of aircraft controls	
nstate_control	int	+	number of control states (maximum nstatemax)	1
		+	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
nVlngcyc	int	+	longitudinal stick	0
nVpedal	int	+	pedal	0
nVtilt	int	+	tilt	0
cont(nvelmax,ncontmax)	real	+	values	
coll(nvelmax)	real	+	collective stick c_{AC0}	
latcyc(nvelmax)	real	+	lateral cyclic stick c_{ACc}	
lngcyc(nvelmax)	real	+	longitudinal cyclic stick c_{ACs}	
pedal(nvelmax)	real	+	pedal c_{ACp}	
tilt(nvelmax)	real	+	tilt α_{tilt}	

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

---

control system: set of aircraft controls c_{AC} defined
- aircraft controls connected to individual controls of each component, $c = Tc_{AC} + c_0$
- for each component control, define matrix T (for each control state) and value c_0
  - flight state specifies control state, or that control state obtained from conversion schedule
- c_0 can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
  - use of component control c_0 can be suppressed for flight state using SET_comp_control

aircraft controls: identified by IDENT_control
- typical aircraft controls are pilot's controls; default IDENT_control='coll','latcyc','lngcyc','pedal','tilt'
- available for trim (flight state specifies trim option)
  - initial values specified if control is trim variable; otherwise fixed for flight state
- each aircraft control can be zero, constant, or function of flight speed (CAS or TAS, piecewise linear input)
  - coll/latcyc/lngcyc/pedal/tilt input put in appropriate nVcont-cont-Vcont, based on IDENT_control
- flight state input can override
- by connecting aircraft control to component control, flight state can specify component control value

---

		+	Aircraft Motion
		+	aircraft pitch angle θ_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 ϕ_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)

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

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

conversion control: use depends on STATE_control, SET_tilt, SET_Vtip of FltState
hover and helicopter mode ($V \leq V_{\mathrm{conv-hover}}$): use tilt=90, Vtip_hover, kgear_hover, kcont_hover
cruise mode ($V \geq V_{\mathrm{conv-cruise}}$): 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

SET_Vschedule	int	+	Velocity schedules (1 CAS, 2 TAS)	1

velocity schedules: all described as function CAS or TAS
conversion, controls and motion, rotor tip speed, landing gear retraction, trim targets

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

---

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

description	trim_var	
aircraft 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
aircraft control	match IDENT_control	

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
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', 'lngflap n'	FltState%trim_target
rotor flapping	'betas n', 'latflap n'	FltState%trim_target
rotor hub moment	'hub Mx n', 'roll n'	FltState%trim_target
rotor hub moment	'hub My n', 'pitch n'	FltState%trim_target
rotor torque	'hub Mz n', 'torque n'	FltState%trim_target
wing lift	'lift wing n', 'flift wing n'	FltState%trim_target, Wing%Klift
wing lift coefficient	'CL wing n'	FltState%trim_target, Wing%Klift
wing lift margin	'L margin n'	FltState%trim_target
tail lift	'lift tail n'	FltState%trim_target

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

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

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

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

center of gravity: baseline is for nacelle angle = 90
- flight state has calculated or input actual cg location

		+	Takeoff flight condition	
SET_atmos	c*12	+	atmosphere specification	'std'
temp	real	+	temperature τ	
dtemp	real	+	temperature increment ΔT	0.
density	real	+	density ρ	

csound	real	+	speed of sound c_s	
viscosity	real	+	viscosity μ	
altitude	real	+	altitude	

takeoff condition (density) used for C_T/σ in rotor sizing
SET_atmos, atmosphere specification:
'std' = standard day at specified altitude (use altitude)
'dtemp' = standard day at specified altitude, plus temperature increment (use altitude, dtemp)
'temp' = standard day at specified altitude, and specified temperature (use altitude, temp)
'dens' = input density and temperature (use density, temp)
'input' = input density, speed of sound, and viscosity (use density, csound, viscosity)
see FltAircraft%SET_atmos for other options (polar, tropical, and hot days)

		+	Weight	
DGW	real	+	design gross weight W_D	
Wfuel_DGW	real	+	mission fuel $W_{\rm fuel}$ corresponding to DGW	
Wpay_DGW	real	+	payload $W_{\rm pay}$ 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 $n_{z\rm ult}$ at SDGW	6.0

input or calculated: design gross weight W_D (FIX_DGW), structural design gross weight W_{SD} (SET_SDGW), maximum takeoff weight W_{MTO} (SET_WMTO), weight
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, $W_G = W_D$–Wfuel_DGW+fFuelSDGW*$W_{\mathrm{fuel-cap}}$
'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, $W_G = W_D$–Wfuel_DGW+$W_{\mathrm{fuel-cap}}$
'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
WE	real	+	weight empty W_E
		+	moments of inertia (based on design gross weight, scaled with reference length)
kx	real	+	roll radius of gyration k_x/L
ky	real	+	pitch radius of gyration k_y/L
kz	real	+	yaw radius of gyration k_z/L

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

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

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

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

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

		+	Number of Components	
nRotor	int	+	rotors (maximum nrotormax)	2
nForce	int	+	forces (maximum nforcemax)	0
nWing	int	+	wings (maximum nwingmax)	0
nTail	int	+	tails (maximum ntailmax)	1

nPropulsion	int	+	propulsion groups (maximum npropmax)	1
nEngineModel	int	+	engine models (maximum nengmax)	1

propulsion group is set of components and engine groups, connected by drive system
engine model describes particular engine, used in one or more engine group

Structure: Systems

Variable	Type		Description	Default
		+	Systems	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Weight	
		+	fixed useful load	
Wcrew	real	+	crew weight	
Wtrap	real	+	trapped fluids and engine oil weight	
Woful	real	+	other fixed useful load	
Wotherkit	real	+	other kit	
SET_fold	int	+	folding (0 none, 1 fold weights, 2 with kit)	0
		+	folding weight in kit f_{foldkit} (fraction wing/rotor/tail/body fold weight)	
fWfoldkitW(nwingmax)	real	+	wing	0.5
fWfoldkitR(nrotormax)	real	+	rotor	0.5
fWfoldkitT(ntailmax)	real	+	tail	0.5
fWfoldkitFt	real	+	body (wing and rotor fold)	0.5
fWfoldkitFw	real	+	body (tail fold)	0.5
SET_Wvib	int	+	vibration treatment weight (1 fraction weight empty, 2 input)	1
Wvib	real	+	weight W_{vib}	
fWvib	real	+	fraction weight empty f_{vib}	
SET_Wcont	int	+	contingency weight (1 fraction weight empty, 2 input)	1
Wcont	real	+	weight W_{cont}	
fWcont	real	+	fraction weight empty f_{cont}	

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

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

		+	systems and equipment	
		+	flight control group and hydraulic group	
MODEL_fc	int	+	model (0 input, 1 AFDD, 2 custom)	1
MODEL_RWfc	int	+	rotary wing flight controls (1 global, 2 for each rotor)	1
refRotor	int	+	reference rotor number for global	1
MODEL_FWfc	int	+	fixed wing flight controls (0 for not present)	1
MODEL_CVfc	int	+	conversion controls (0 for not present)	1
		+	flight control weight increment	
dWRWfc_b	real	+	rotary wing, boosted	0.
dWRWfc_mb	real	+	rotary wing, control boost mechanisms	0.
dWRWfc_nb	real	+	rotary wing, non-boosted	0.
dWFWfc_mb	real	+	fixed wing, control boost mechanisms	0.
dWFWfc_nb	real	+	fixed wing, non-boosted	0.
dWCVfc_mb	real	+	conversion, boosted	0.
dWCVfc_nb	real	+	conversion, control boost mechanisms	0.
		+	fixed flight controls	
Wfc_cc	real	+	cockpit controls	0.
Wfc_afcs	real	+	automatic flight control system	0.
		+	hydraulic weight increment	
dWRWhyd	real	+	rotary wing	0.
dWFWhyd	real	+	fixed wing	0.
dWCVhyd	real	+	conversion	0.
WEQhyd	real	+	equipment hydraulics	0.
Wauxpower	real	+	auxiliary power group (APU)	0.
Winstrument	real	+	instruments group	0.
Wpneumatic	real	+	pneumatic group	0.
Welectrical	real	+	electrical group (aircraft)	0.

WMEQ	real	+	avionics group (mission equipment)	0.
		+	armament group	
Warmor	real	+	armor	0.
Warmprov	real	+	armament provisions	0.
Wfurnish	real	+	furnishings and equipment group	0.
Wenviron	real	+	environmental control group	0.
		+	anti-icing group	
MODEL_DI	int	+	model (0 input, 1 AFDD, 2 custom)	1
		+	weight increment	
dWDIelect	real	+	electrical system	0.
dWDIsys	real	+	anti-ice system	0.
Wload	real	+	load and handling group	0.

---

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

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

---

		+	Technology Factors	
		+	rotary wing flight control weight	
TECH_RWfc_b	real	+	boosted χ_{RWb}	1.0
TECH_RWfc_mb	real	+	control boost mechanisms χ_{RWmb}	1.0
TECH_RWfc_nb	real	+	non-boosted χ_{RWnb}	1.0
		+	fixed wing flight control weight	
TECH_FWfc_mb	real	+	control boost mechanisms χ_{FWmb}	1.0
TECH_FWfc_nb	real	+	non-boosted χ_{FWnb}	1.0
		+	conversion flight control weight	
TECH_CVfc_mb	real	+	control boost mechanisms χ_{CVmb}	1.0
TECH_CVfc_nb	real	+	non-boosted χ_{CVnb}	1.0
		+	flight control hydraulics	
TECH_RWhyd	real	+	rotary wing $\chi_{RW\mathrm{hyd}}$	1.0

TECH_FWhyd	real	+	fixed wing $\chi_{FW\mathrm{hyd}}$	1.0
TECH_CVhyd	real	+	conversion $\chi_{CV\mathrm{hyd}}$	1.0
		+	anti-icing	
TECH_DIelect	real	+	electrical system $\chi_{DI\mathrm{elect}}$	1.0
TECH_DIsys	real	+	anti-ice system $\chi_{DI\mathrm{sys}}$	1.0
		+	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 $f_{RW\mathrm{red}}$	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 $f_{RW\mathrm{hyd}}$ (fraction rotary wing boost mechanisms + hydraulic weight)	0.40
fFWhyd	real	+	fixed wing $f_{FW\mathrm{hyd}}$ (fraction fixed wing boost mechanisms weight)	0.10
fCVhyd	real	+	conversion $f_{CV\mathrm{hyd}}$ (fraction conversion boost mechanisms weight)	0.10

only MODEL_WRWfc=fraction uses fRWfc_nb
typically fRWfc_nb = 0.6 (data range 0.3 to 1.8), fRWhyd = 0.4

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

		+	Anti-Icing Group, AFDD Weight Model	
kDeIce_elec(nrotormax)	real	+	weight factor for electrical system K_{elec} (lb/ft^2 or kg/ft^2)	0.25
kDeIce_rotor(nrotormax)	real	+	weight factor for main rotor K_{rotor} (lb/ft^2 or kg/ft^2)	0.25
kDeIce_wing(nwingmax)	real	+	weight factor for wing K_{wing} (lb/ft or kg/ft)	0.0
kDeIce_air	real	+	weight factor for engine air intake K_{air} (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 ℓ_{fus}	
SET_nose	int	+	nose length (distance forward of hub; 1 input, 2 calculated)	1
Length_nose	real	+	nose length ℓ_{nose}	
fLength_nose	real	+	nose length (fraction rotor radius)	
SET_aft	int	+	aft length (distance aft of hub; 1 input, 2 calculated)	1
Length_aft	real	+	aft length ℓ_{aft}	
fLength_aft	real	+	aft length (fraction rotor radius)	
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}	
refRotor	int	+	reference rotor number (for rotor radius)	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)
SET_aft: input (use Length_aft) or calculated (from fLength_aft)
fRef_fus=(SL_fuselage–SL_nose)/Length_fus; used to calculate operating length

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)

		+	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
DoQ_cont	real	+	contingency drag, area $(D/q)_{\text{cont}}$	0.
DoQV_cont	real	+	contingency vertical drag, area $(D/q)_{V\text{cont}}$	0.

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

		+	Weight	
		+	fuselage group	
MODEL_weight	int	+	fuselage group model (0 input, 1 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.
		+	Technology Factors	
TECH_body	real	+	basic body χ_{basic}	1.0
TECH_mar	real	+	body marinization χ_{mar}	1.0
TECH_press	real	+	pressurization χ_{press}	1.0
TECH_crash	real	+	body crashworthiness χ_{cw}	1.0
TECH_ftfold	real	+	tail fold χ_{tfold}	1.0
TECH_fwfold	real	+	wing fold χ_{wfold}	1.0

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

		+	Aerodynamics, Standard Model	
AoA_zl	real	+	zero lift angle of attack α_{zl} (deg)	0.
AoA_max	real	+	angle of attack for maximum lift $\alpha_{\max}$ (deg)	10.
		+	lift	
SET_lift	int	+	specification (1 fixed, L/q; 2 scaled, C_L)	2
dLoQda	real	+	lift slope, $d(L/q)/d\alpha$ (per rad)	0.
dCLda	real	+	lift slope, $C_{L\alpha} = dC_L/d\alpha$ (per rad; based on wetted area, $L/q = SC_L$)	0.
		+	pitch moment	
SET_moment	int	+	specification (1 fixed, M/q; 2 scaled, C_M)	2
MoQ0	real	+	moment at zero lift, $(M/q)_0$	0.0
CM0	real	+	moment at zero lift, C_{M0} (based on wetted area and fuselage length, $M/q = S\ell C_M$)	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, $M/q = S\ell C_M$)	0.0
SS_zy	real	+	sideslip angle for zero side force β_{zy} (deg)	0.
SS_max	real	+	sideslip angle for maximum side force $\beta_{\max}$ (deg)	10.
		+	side force	
SET_side	int	+	specification (1 fixed, Y/q; 2 scaled, C_Y)	2

dYoQdb	real	+	side force slope, $d(Y/q)/d\beta$ (per rad)	0.
dCYdb	real	+	side force slope, $C_{Y\beta} = dC_Y/d\beta$ (per rad; based on wetted area, $Y/q = SC_Y$)	0.
		+	yaw moment	
SET_yaw	int	+	specification (1 fixed, N/q; 2 scaled, C_N)	2
NoQ0	real	+	moment at zero lift, $(N/q)_0$	0.0
CN0	real	+	moment at zero lift, C_{N0} (based on wetted area and fuselage length, $N/q = S\ell C_N$)	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, $N/q = S\ell C_N$)	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 $(D/q)_{\rm fit}$	
CD_fit	real	+	coefficient $C_{D{\rm fit}}$ (based on wetted area, $D/q = SC_D$)	0.
		+	rotor-body interference	
SET_Drb	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ_rb(nrotormax)	real	+	area $(D/q)_{rb}$	
CD_rb(nrotormax)	real	+	coefficient C_{Drb} (based on wetted area, $D/q = SC_D$)	0.
		+	vertical drag	
SET_Vdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQV	real	+	area $(D/q)_V$	
CDV	real	+	coefficient C_{DV} (based on projected area, $D/q = S_{\rm proj}C_D$)	0.
		+	sideward drag	
SET_Sdrag	int	+	specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQS	real	+	area $(D/q)_S$	
CDS	real	+	coefficient C_{DS} (based on wetted area, $D/q = SC_D$)	0.

		+	drag variation with angle of attack	
MODEL_drag	int	+	model (0 none, 1 general, 2 quadratic)	2
AoA_Dmin	real	+	angle of attack for fuselage minimum drag $C_{D\min}$ (deg)	0.0
Kdrag	real	+	drag increment K_d, $\Delta C_D = C_{D0} K_d \lvert\alpha_e\rvert^{X_d}$	0.0
Xdrag	real	+	drag increment X_d, $\Delta C_D = C_{D0} K_d \lvert\alpha_e\rvert^{X_d}$	2.
		+	transition from forward flight drag to vertical drag	
MODEL_trans	int	+	model (1 input transition angle of attack, 2 calculate for quadratic)	1
AoA_tran	real	+	angle of attack for transition α_t (deg)	25.

		+	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

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 fWbody_crash = 0.06
typically fWbody_tfold = 0.30 (UNIV) or 0.05 (HELO) for folding tail

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

Structure: LandingGear

Variable	Type		Description	Default
		+	Landing Gear	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Geometry	
loc_gear	Location	+	landing gear location	
d_gear	real	+	distance from bottom of landing gear to WL_gear d_{LG}	0.
place	int	+	placement (1 located on body, 2 located on wing)	1
KIND_LG	int	+	retraction (0 fixed, 1 retracts)	1
speed	real	+	retraction speed (CAS or TAS, knots)	
			landing gear location: with HAGL (FltState) determines rotor height above ground level height rotor = landing gear above ground + hub above landing gear = HAGL + (WL_hub–WL_gear+d_gear) place: used for weight (fuselage and wing)	
		+	Aerodynamics	
MODEL_aero	int	+	model (0 none, 1 standard)	1
		+	Weight	
		+	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.

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

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

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

		+	Landing Gear Group, 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 $f_{LG\mathrm{ret}}$ (fraction basic weight)	0.08
fWLG_crash	real	+	landing gear weight for crashworthiness $f_{LG\mathrm{cw}}$ (fraction basic+retraction weight)	0.14

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

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

Structure: Rotor

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

configuration designation: principal designation required, rest identify special characteristics
- principal designation = 'main', 'tail', 'prop'
- antitorque = 'antiQ', 'auxT'
- twin rotor = 'coaxial', 'tandem', 'tiltrotor' (keyword = tan, coax, tilt)
- others = 'variable diameter', 'ducted fan' (keyword = var, duct)

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

		+	Propulsion group	
kPropulsion	int	+	group number	1
KIND_xmsn	int	+	drive system branch (1 primary, 0 dependent)	1
Vtip_ref(ngearmax)	real	+	reference tip speed	
INPUT_gear	int	+	gear ratio input for dependent branch (1 Vtip_ref, 2 gear)	1
gear(ngearmax)	real	+	gear ratio $r = \Omega_{\rm dep}/\Omega_{\rm prim}$ (ratio rpm to rpm of primary rotor)	1.0

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

		+	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 ($V_{\mathrm{tip}}/V_{\mathrm{tip-hover}}$)	
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 rating) 1
Plimit_rs	real	+	rotor shaft power rating $P_{RS\text{limit}}$
fPlimit_rs	real	+	rotor shaft power rating factor 1.

drive system torque limit: Size%SET_limit_ds = input (use Plimit_rs) or calculated (from fPlimit_rs)
- SET_limit_ds='input': Plimit_rs input
- SET_limit_ds≠'input': from rotor power required at transmission sizing flight conditions (DESIGN_xmsn)

rotor shaft: options for SET_limit_ds≠'input'
- SET_limit_rs=0: Plimit_rs
- SET_limit_rs=1: fPlimit_rs × (rotor P_{req})
- SET_limit_rs=2: fPlimit_rs $\times P_{DS\text{limit}}$

rotor shaft power rating: 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/σ (thrust-weighted)
sigma	real	+	solidity $\sigma = Nc/\pi R$ (thrust-weighted)
Tdesign	real	+	thrust for antitorque or aux thrust rotor
Pdesign	real	+	power for antitorque or aux thrust rotor

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

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

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
		+	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 =Radius in hover and helicopter mode ($V \leq V_{\mathrm{conv-hover}}$)
R =Radius*fRcruise in cruise mode ($V \geq V_{\mathrm{conv-cruise}}$); linear with V in conversion mode
function of speed: use nVdiam, fdiam, Vdiam to calculate R

		+	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 σ_t/σ_g	1.
taper	real	+	taper ratio t (tip chord/root chord)	1.
SET_twist	int	+	twist distribution (1 linear from twistL, 2 nonlinear from twist)	1
twistL	real	+	linear twist θ_L (deg, root to tip)	-10.
nprop	int	+	number of radial stations (maximum nrmax)	2

rprop(nrmax)	real	+	radial stations ($r_{\rm root}/R$)	
fchord(nrmax)	real	+	chord distribution $c(r)/c_{\rm ref}$	1.
twist(nrmax)	real	+	twist $\theta_{tw}(r)$ (deg)	
		+	flap dynamics	
KIND_hub	int	+	hub type (1 articulated, 2 hingeless)	1
flapfreq	real	+	first flapwise natural frequency ν (per-rev at hover tip speed)	1.04
conefreq	real	+	coning natural frequency ν (0. to use flapfreq)	0.
gamma	real	+	blade Lock number γ	8.
precone	real	+	precone β_p (deg)	0.
delta3	real	+	pitch-flap coupling δ_3 (deg)	0.
		+	aerodynamics	
dclda	real	+	blade section 2D lift-curve slope $a = c_{\ell\alpha}$ (per-rad)	5.7
tiploss	real	+	tip loss factor B (lift zero from BR to tip)	0.97
xroot	real	+	root cutout ($r_{\rm root}/R$)	0.1
fBlockage	real	+	blockage factor (force increment $f_B T$)	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
fTWsigma = sigma_tw/sigma_geom; for linear taper $f = c(.75R)/c(.5R) = (1 + 3\text{taper})/(2 + 2\text{taper})$
equivalent linear taper $= c(\text{tip})/c(\text{root}) = (2f - 1)/(3 - 2f)$
for linear taper $f_c = c/c_{\rm ref} = 1 + (r - 0.75)4(\text{taper} - 1)/(1 + 3\text{taper})$

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

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

---

		+	Geometry (for graphics)	
thick	real	+	blade thickness-to-chord ratio	0.12

		+	Blade element theory solution	
		+	integration	
mr	int	+	number of radial stations (xroot to 1; maximum mrmax)	4
mpsi	int	+	number of azimuth angles (maximum mpsimax)	8
		+	Geometry	
loc_rotor	Location	+	hub location	
loc_pylon	Location	+	pylon location	
loc_pivot	Location	+	pivot location	
loc_naccg	Location	+	nacelle cg location	
direction	c*16	+	nominal orientation ('+x', '–x', '+y', '–y', '+z', '–z'; 'main' (–z), 'tail' (ry), 'prop' (x))	'main'
KIND_tilt	int	+	shaft control (0 fixed shaft, 1 incidence, 2 cant, 3 both controls)	0
		+	orientation of rotor shaft	
incid_hub	real	+	incidence i (deg)	0.
cant_hub	real	+	cant angle c (deg)	0.
		+	orientation of pivot axes	
dihedral_pivot	real	+	pivot dihedral angle ϕ_h (deg)	
pitch_pivot	real	+	pivot pitch angle θ_h (deg)	
sweep_pivot	real	+	pivot sweep angle ψ_h (deg)	
		+	reference shaft control	
incid_ref	real	+	incidence $i_{\rm ref}$ (deg)	0.
cant_ref	real	+	cant angle $c_{\rm ref}$ (deg)	0.
		+	moving weight for cg shift	
SET_Wmove	int	+	weight (1 wing tip weight, 2 W_{gbrs}, 3 W_{gbrs} and W_{ES})	1
fWmove	real	+	fraction moving weight	1.

---

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

		+	Controls	
KIND_control	int	+	rotor control mode (1 thrust and TPP, 2 thrust and NFP, 3 pitch and TPP, 4 pitch and NFP)	1
KIND_cyclic	int	+	cyclic input (1 tip-path-plane tilt, 2 hub moment, 3 lift offset)	1
KIND_coll	int	+	collective input (1 thrust, 2 C_T/σ)	2
SCALE_coll	int	+	scale collective T matrix (0 for none)	1
		+	collective (magnitude of thrust vector)	
INPUT_coll	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_coll(ncontmax,nstatemax)	real	+	control matrix	
nVcoll	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 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_{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_{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; ≥ 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; ≥ 2 piecewise linear, maximum nvelmax)	
Kprop(nvelmax)	real	+	target	
Vprop(nvelmax)	real	+	speeds (CAS or TAS)	

target definition determined by Aircraft%trim_quant
Klift can be fraction total aircraft lift, lift, C_L/σ, or C_T/σ
Kprop can be fraction total aircraft drag, propulsive force $-X$, $-C_X/\sigma$, or $-X/q$)

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

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

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

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

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

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

		+	Geometry
SET_aeroaxes	int	+	hub/pylon aerodynamic axes (0 input pitch, 1 helicopter, 2 propeller or tiltrotor) 1
pitch_aero	real	+	pitch relative shaft axes θ_{ref}, $C^{BS} = Y_{-\theta_{\text{ref}}}$ 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, $k = S_{\text{pylon}}/(w/N_{\text{rotor}})^{2/3}$ (Units_Dscale) 1.0
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, $k = S_{\text{spin}}/A_{\text{spin}}$ 1.0
fRadius_spin	real	+	spinner radius (fraction rotor radius) 0.

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

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

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

		+	Drag	
MODEL_drag	int	+	model (0 none, 1 standard)	1
Idrag	real	+	incidence angle for helicopter nominal drag (deg; 0 for not tilt)	0.
		+	Weight	
		+	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.
Wduct	real	+	rotor/fan duct & rotor supports	0.
SET_Iblade	int	+	blade moment of inertia (0 from Lock number, 1 from blade wt, 2 tip wt from Lock number, 3 tip wt from AI)	1
AI	real	+	autorotation index $KE/P = \frac{1}{2} N_{\text{blade}} I_{\text{blade}} \Omega^2 / P$ (sec)	3.0
Wblade_tip	real	+	tip weight (per blade)	0.
rWblade_tip	real	+	location tip weight (fraction blade radius)	0.9
fWblade_tip	real	+	distributed weight for centrifugal force (fraction Wblade_tip)	1.0
rblade	real	+	radius of gyration for distributed mass (fraction blade radius)	0.6

		+	Technology Factors	
TECH_blade	real	+	blade weight χ_{blade}	1.0
TECH_hub	real	+	hub and hinge weight χ_{hub}	1.0
TECH_shaft	real	+	inter-rotor shaft χ_{shaft}	1.0
TECH_spin	real	+	fairing/spinner weight χ_{spin}	1.0
TECH_rfold	real	+	blade fold weight χ_{fold}	1.0
TECH_tr	real	+	tail rotor weight χ_{tr}	1.0
TECH_aux	real	+	auxiliary thrust weight χ_{at}	1.0

---

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

blade weight: $W_{\text{blade}} = \chi_{\text{blade}} w_{\text{blade}} + dW_{\text{blade}} + (1+f) W_{\text{tip}} N_{\text{blade}}$
SET_Iblade: calculate blade moment of inertia Iblade
0 from Lock number gamma, independent of blade weight
1 from blade weight
2 from Lock number gamma, tip weight Wblade_tip calculated from Iblade
3 from autorotation index AI, tip weight Wblade_tip calculated from Iblade
for tail rotor or auxiliary thrust weight model (MODEL_config = 2 or 3), Wblade_tip not used and SET_Iblade = 0

rotor weight = blade + hub + spinner + fold + shaft
rotor config determines where weight put in weight statement
main rotor: rotor group
tail rotor: empennage group (tail rotor)
propeller: propulsion group (propeller/fan installation)
rotor/fan duct & rotor supports weight not used for main rotor

---

		+	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 κ_{hover}	1.12
Ki_climb	real	+	axial climb κ_{climb}	1.08
Ki_prop	real	+	axial cruise (propeller) κ_{prop}	2.0
Ki_edge	real	+	edgewise flight (helicopter) κ_{edge}	2.0
		+	variation with thrust	
CTs_Hind	real	+	$(C_T/\sigma)_{\text{ind}}$ for κ_h variation	0.08
kh1	real	+	coefficient k_{h1} for κ_h	0.
kh2	real	+	coefficient k_{h2} for κ_h	0.
Xh2	real	+	exponent X_{h2} for κ_h	2.
CTs_Pind	real	+	$(C_T/\sigma)_{\text{ind}}$ for κ_p variation	0.08
kp1	real	+	coefficient k_{p1} for κ_p	0.
kp2	real	+	coefficient k_{p2} for κ_p	0.
Xp2	real	+	exponent k_{p2} for κ_p	2.
		+	variation with shaft angle	
kpa	real	+	coefficient $k_{h\alpha}$ for κ_p	0.
Xpa	real	+	exponent $X_{h\alpha}$ for κ_p	2.
Maxial	real	+	constant M_{axial} in transition from hover to climb	1.176
Xaxial	real	+	exponent X_{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 μ_{edge} for Ki_edge	0.35
ke1	real	+	coefficient k_{e1} for $\kappa(\mu)$ (linear)	0.8
ke2	real	+	coefficient k_{e2} for $\kappa(\mu)$ (quadratic)	0.
ke3	real	+	coefficient k_{e3} for $\kappa(\mu)$	1.
Xe	real	+	exponent X_e for $\kappa(\mu)$	4.5
kea	real	+	variation with rotor drag $k_{e\alpha}$	0.

		+	variation with lift offset	
ko1	real	+	coefficient k_{o1} for f_{off}	0.
ko2	real	+	factor k_{o2} for f_{off}	8.
Ki_min	real	+	minimum $\kappa_{\min}$	1.
Ki_max	real	+	maximum $\kappa_{\max}$	10.

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

		+	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
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.
		+	Ducted fan	
MODEL_duct	int	+	model (1 specify area ratio, 2 specify thrust ratio)	1
fDuctA	real	+	area ratio f_A (fan area/far wake area)	1.
fDuctT	real	+	thrust ratio f_T (rotor thrust/total thrust)	0.5
fDuctVx	real	+	velocity ratio f_{Vx} (fan edgewise velocity/free stream velocity)	1.
fDuctVz	real	+	velocity ratio f_{Vz} (fan axial velocity/free stream velocity)	1.

ducted fan model used only if config='duct'

		+	Twin rotors	
MODEL_twin	c*12	+	model (based on config, none, side-by-side, coaxial, tandem)	'config'
Kh_twin	real	+	ideal induced velocity correction κ_{twin} for hover	1.00
Kf_twin	real	+	ideal induced velocity correction κ_{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

MODEL_twin: 'config', 'none', 'side-by-side' or 'tiltrotor', 'coaxial', or 'tandem'
coaxial: MODEL_twin='coaxial' (with 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 $0.12D$

		+	Rotor Profile Power, Standard Energy Performance Method	
		+	Technology factor	
TECH_drag	real	+	profile power χ	1.0
Re_ref	real	+	Reference Reynolds number $Re_{\rm ref}$ (0. for no correction)	0.0
MODEL_basic	int	+	Basic model $c_{d\rm basic}$ (1 array, 2 equation)	2
		+	array (c_d vs thrust-weighted C_T/σ)	
ncd	int	+	number of points (maximum 24)	24
CTs_cd(24)	real	+	blade loading	
cd(24)	real	+	drag coefficient	
		+	equation	
CTs_Dmin	real	+	$(C_T/\sigma)_{D\rm min}$ for minimum profile drag ($\Delta = \lvert C_T/\sigma - (C_T/\sigma)_{D\rm min}\rvert$)	0.07
d0_hel	real	+	coefficient $d_{0\rm hel}$ in drag, $c_{dh} = d_{0\rm hel} + d_{1\rm hel}\Delta + d_{2\rm hel}\Delta^2 + \Delta c_{d\rm sep}$ (hover/edgewise)	0.009
d1_hel	real	+	coefficient $d_{1\rm hel}$ in drag (hover/edgewise)	0.0
d2_hel	real	+	coefficient $d_{2\rm hel}$ in drag (hover/edgewise)	0.5
d0_prop	real	+	coefficient $d_{0\rm prop}$ in drag, $c_{dp} = d_{0\rm prop} + d_{1\rm prop}\Delta + d_{2\rm prop}\Delta^2 + \Delta c_{d\rm sep}$ (axial)	0.009
d1_prop	real	+	coefficient $d_{1\rm prop}$ in drag (axial)	0.0
d2_prop	real	+	coefficient $d_{2\rm prop}$ in drag (axial)	0.5
dprop	real	+	variation with shaft angle, coefficient $d_{p\alpha}$ for c_{dp}	0.
Xprop	real	+	variation with shaft angle, exponent $X_{p\alpha}$ for c_{dp}	2.
CTs_sep	real	+	$(C_T/\sigma)_{\rm sep}$ for separation ($\Delta c_{d\rm sep} = d_{\rm sep}(\lvert C_T/\sigma\rvert - (C_T/\sigma)_{\rm sep})^{X_{\rm sep}}$)	0.07
dsep	real	+	factor $d_{\rm sep}$ in drag increment	4.0
Xsep	real	+	exponent $X_{\rm sep}$ in drag increment	3.0

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

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

MODEL_comp	int	+	Compressibility model $c_{d\text{comp}}$ (0 none, 1 drag divergence, 2 similarity)	1
		+	similarity model	
fSim	real	+	factor f	1.0
thick_tip	real	+	blade tip thickness-to-chord ratio τ	0.08

		+	drag divergence model ($\Delta_m = M_{at} - M_{dd}, \Delta c_d = d_{m1}\Delta_m + d_{m2}\Delta_m^{X_m}$)	
dm1	real	+	coefficient d_{m1} in drag increment	0.056
dm2	real	+	coefficient d_{m2} in drag increment	0.416
Xm	real	+	exponent X_m in drag increment	2.0
		+	drag divergence Mach number ($M_{dd} =$ Mdd0 $-$ Mddcl c_ℓ)	
Mdd0	real	+	M_{dd0} at zero lift	0.88
Mddcl	real	+	derivative with lift $\kappa = \partial M_{dd}/\partial c_\ell$	0.16

		+	Performance, Table Method	
MODEL_indTab	int	+	induced power model (0 standard, 1 table $\kappa(\mu)$, 2 table $\kappa(\mu_z)$)	1
MODEL_proTab	int	+	profile power model (0 standard, 1 table c_d, 2 table $c_d F = 8C_{Po}/\sigma$)	1
		+	table	
nmu	int	+	number of advance ratio values (maximum ntablemax)	0
nCTs	int	+	number of blade loading values (maximum ntablemax)	0
mu(ntablemax)	real	+	advance ratio μ	
muz(ntablemax)	real	+	axial advance ratio μ_z	
CTs(ntablemax)	real	+	blade loading C_T/σ	
Ki(ntablemax,ntablemax)	real	+	induced power factor $\kappa(\mu, C_T/\sigma)$ or $\kappa(\mu_z, C_T/\sigma)$	
cdo(ntablemax,ntablemax)	real	+	profile power mean $c_d(\mu, C_T/\sigma)$ or $c_d(\mu, C_T/\sigma)F(\mu, \mu_z)$	

nonzero values of Ki and/or cdo in FltState supersede table values

		+	Rotor Drag, Standard Model	
		+	forward flight drag	
SET_Dhub	int	+	hub drag specification (1 fixed, D/q; 2 scaled, C_D; 3 scaled, squared-cubed; 4 scaled, square-root)	2
DoQ_hub	real	+	area $(D/q)_{\rm hub}$	
CD_hub	real	+	coefficient $C_{D\rm hub}$ (based on rotor area, $D/q = SC_D$)	0.0024
kDrag_hub	real	+	$k = (D/q)/(W/1000)^{2/3}$ or $(D/q)/W^{1/2}$ (Units_Dscale)	0.8

SET_Dpylon	int	+	pylon drag specification (1 fixed, D/q; 2 scaled, C_D)	2
DoQ_pylon	real	+	area $(D/q)_{\text{pylon}}$	
CD_pylon	real	+	coefficient $C_{D\text{pylon}}$ (based on pylon 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
		+	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
X_hub	real	+	hub drag, transition exponent X_d	2.
X_pylon	real	+	pylon drag, transition exponent X_d	2.

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

component drag contributions must be consistent; pylon is rotor support, and nacelle is engine support
tiltrotor with tilting engines use pylon drag (and no nacelle drag), since pylon connected to rotor shaft axes
tiltrotor with nontilting engines: use nacelle drag as well
rotor with a spinner (such as on a tiltrotor aircraft) likely not have hub drag

SET_Dhub, hub drag: use one of DoQ_hub, CD_hub, kDrag_hub
units of kDrag are $\text{ft}^2/\text{klb}^{2/3}$ 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 fThrust*Tdesign (antitorque or aux thrust rotor)

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

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

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

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

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

		+	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 ν (per-rev at hover tip speed)	
flapfreq_blade	real	+	blade (0. to use flapfreq)	0.
flapfreq_hub	real	+	hub (0. to use flapfreq_blade)	0.
		+	lift offset rotor	
MODEL_offset	int	+	rotor tip clearance (for blade weight; 1 scaled, 2 fixed)	1
offset	real	+	design lift offset L (roll moment$/TR$)	0.3
thick20	real	+	blade airfoil thickness-to-chord ratio $\tau_{.2R}$ (at 20%R)	0.21
clearance_tip	real	+	tip clearance, scaled s/R or fixed s (ft or m)	0.05
		+	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.

---

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.

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

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

Structure: Force

Variable	Type		Description	Default
		+	Force	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Geometry	
loc_force	Location	+	location	
direction	c*16	+	nominal orientation ('+x', '–x', '+y', '–y', '+z', '–z')	'x'
		+	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)	
		+	incidence i (tilt)	
INPUT_incid	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_incid(ncontmax,nstatemax)	real	+	control matrix	
nVincid	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
incid(nvelmax)	real	+	values	
Vincid(nvelmax)	real	+	speeds (CAS or TAS)	
		+	yaw ψ	
INPUT_yaw	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_yaw(ncontmax,nstatemax)	real	+	control matrix	
nVyaw	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
yaw(nvelmax)	real	+	values	
Vyaw(nvelmax)	real	+	speeds (CAS or TAS)	

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

		+	Performance	
Fmax	real	+	design maximum force $F_{\max}$	0.0
sfc	real	+	thrust specific fuel consumption	1.0
		+	Weight	
KIND_weight	int	+	weight group (1 engine system, 2 propeller/fan installation, 3 tail rotor)	1
SW	real	+	specific weight S	
dWforce	real	+	weight increment	0.0

Structure: Wing

Variable	Type		Description	Default
		+	Wing	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Geometry	
wingload	real	+	wing loading $W/S = f_W W_D/S$	
fDGW	real	+	fraction DGW f_W (for wing loading)	1.0
area	real	+	area S	
span	real	+	span b	
chord	real	+	chord c	
AspectRatio	real	+	aspect ratio AR	
			wing parameters: for each wing; input two quantities, other two derived (SizeParam input) SET_wing = input two of ('area' or wing loading 'WL'), ('span' or 'ratio' or 'radius' or 'width' or 'hub' or 'panel'), 'chord', aspect ratio 'aspect' SET_wing = 'ratio+XX' to calculate span from span of another wing SET_wing = 'radius+XX' to calculate span from rotor radius SET_wing = 'width+XX' to calculate span from rotor radius, fuselage width, and clearance (tiltrotor) SET_wing = 'hub+XX' to calculate span from rotor hub position (tiltrotor) SET_wing = 'panel+XX' to calculate span from wing panel widths if wing sized from wing loading (SET_wing='WL+xx'), area = fDGW*DGW/wingload	
		+	Geometry	
		+	rotors	
nRotorOnWing	int	+	number of rotors mounted on wing	0
RotorOnWing(nrotormax)	int	+	rotor numbers	

		+	span calculation	
fSpan	real	+	ratio wing span to span of other wing, or to rotor radius	1.0
otherWing	int	+	other wing number	0
RotorForSpan	int	+	rotor number for span (if nRotorOnWing=0)	0
RotorOnPanel(npanelmax)	int	+	rotor at wing panel edge	
thick	real	+	thickness ratio τ_w	.23
fWidth_box	real	+	wing torque box chord w_{tb} (fraction wing chord)	0.45

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.
		+	Geometry	
loc_wing	Location	+	aerodynamic center location	
nPanel	int	+	number of wing panels (maximum npanelmax)	1
KIND_ACoffset	int	+	aero center offset (1 fixed, 2 fraction root chord, 3 fraction inboard chord)	1
		+	Wing Panels	
SET_panel(npanelmax)	c*24	+	panel parameters	'span+taper'
span_panel(npanelmax)	real	+	span (one side), b_p	
area_panel(npanelmax)	real	+	area (both sides), S_p	
chord_panel(npanelmax)	real	+	mean chord, c_p	
fspan_panel(npanelmax)	real	+	ratio span to wing span (one side), $b_p/(b/2)$	1.
farea_panel(npanelmax)	real	+	ratio area to wing area (both sides), S_p/S	1.
fchord_panel(npanelmax)	real	+	ratio mean chord to wing chord, c_p/c	1.

		+	panel edges	
edge_panel(npanelmax)	real	+	outboard edge, y_E	
fedge_panel(npanelmax)	real	+	outboard edge, $\eta_E = y/(b/2)$	1.
lambdaI(npanelmax)	real	+	inboard chord ratio, $c_I/c_{\rm ref}$	1.
lambdaO(npanelmax)	real	+	outboard chord ratio, $c_O/c_{\rm ref}$	1.
		+	aerodynamic center locus	
sweep_panel(npanelmax)	real	+	sweep Λ_p (deg, + aft)	0.
dihedral_panel(npanelmax)	real	+	dihedral δ_p (deg, + up)	0.
dxAC_panel(npanelmax)	real	+	chordwise offset at panel inboard edge x_{Ip} (+ aft)	0.
dzAC_panel(npanelmax)	real	+	vertical offset at panel inboard edge z_{Ip} (+ up)	0.
		+	control surfaces	
fchord_flap(npanelmax)	real	+	flap chord $\ell_F = c_F/c_p$ (fraction panel chord)	0.25
fchord_flaperon(npanelmax)	real	+	flaperon/aileron chord $\ell_f = c_f/c_p$ (fraction panel chord)	0.25
fspan_flap(npanelmax)	real	+	flap span $f_b = b_F/b_p$ (fraction panel span)	0.5
fspan_flaperon(npanelmax)	real	+	flaperon/aileron span $f_b = b_f/b_p$ (fraction panel span)	0.5
fAC_aileron(npanelmax)	real	+	aileron aerodynamic center lateral position y	0.7

---

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 area_chord= c_p
  - 'cratio' = input ratio to wing chord, fchord_panel= c_p/c

'taper' = from chord ratios lambda1 and lambda0

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 $c_{\rm ref}$ 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

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

vskip30pt

		+	Controls (each panel)	
		+	kind deflection	
KIND_flap(npanelmax)	int	+	flap (1 fraction root flap; 2 increment relative root flap; 3 independent)	3
KIND_aileron(npanelmax)	int	+	aileron (1 fraction root aileron; 2 increment relative root aileron; 3 independent)	3
KIND_incid(npanelmax)	int	+	incidence (1 fraction root incidence; 2 increment relative root incidence; 3 independent)	3
KIND_flaperon(npanelmax)	int	+	kind flaperon deflection (1 fraction flap; 2 increment relative flap; 3 independent)	1

		+	flap δ_{Fp}	
INPUT_flap(npanelmax)	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_flap(ncontmax,nstatemax,npanelmax)	real	+	control matrix	
nVflap(npanelmax)	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
flap(nvelmax,npanelmax)	real	+	values	
Vflap(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)	
		+	flaperon δ_{fp}	
INPUT_flaperon(npanelmax)	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_flaperon(ncontmax,nstatemax,npanelmax)	real	+	control matrix	
nVflaperon(npanelmax)	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
flaperon(nvelmax,npanelmax)	real	+	values	
Vflaperon(nvelmax,npanelmax)	real	+	speeds (CAS or TAS)	
		+	aileron δ_{ap}	
INPUT_aileron(npanelmax)	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_aileron(ncontmax,nstatemax,npanelmax)	real	+	control matrix	
nVaileron(npanelmax)	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 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; ≥ 2 piecewise linear, maximum nvelmax)	
Klift(nvelmax)	real	+	target	
Vlift(nvelmax)	real	+	speeds (CAS or TAS)	

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

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

		+	tiltrotor model	
xWtip	real	+	increment for weight on wing tips	0.0
		+	Technology Factors	
TECH_prim	real	+	wing primary structure (torque box) weight χ_{prim}	1.0
TECH_ext	real	+	wing extension weight χ_{ext}	1.0
TECH_fair	real	+	fairing weight χ_{fair}	1.0
TECH_fit	real	+	fittings weight χ_{fit}	1.0
TECH_flap	real	+	flaps and control surfaces weight χ_{flap}	1.0
TECH_wfold	real	+	wing fold weight χ_{fold}	1.0
TECH_efold	real	+	wing extension fold weight χ_{efold}	1.0

---

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

tiltrotor model requires weight on wing tips: both sides
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

		+	Wing Aerodynamics, Standard Model	
AoA_zl	real	+	zero lift angle of attack α_{zl} (deg)	0.
CLmax	real	+	maximum lift coefficient $C_{L\max}$	1.5
		+	lift	
SET_lift	int	+	specification (2 2D $dC_L/d\alpha$; 3 3D $dC_L/d\alpha$)	2
dCLda	real	+	lift curve slope $C_{L\alpha} = dC_L/d\alpha$ (per rad)	5.73
Tind	real	+	lift curve slope non-elliptical loading correction τ	0.25
Eind	real	+	Oswald or span efficiency e ($C_{Di} = (C_L - C_{L0})^2/(\pi e AR)$)	0.8
CL_Dmin	real	+	lift coefficient for minimum induced drag C_{L0}	0.0
eta0	real	+	control effectiveness factor η_0, $\eta_0 - \eta_1\lvert\delta\rvert$	0.85
eta1	real	+	control effectiveness factor η_1, $\eta_0 - \eta_1\lvert\delta\rvert$	0.43

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

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

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

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

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

		+	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 $b_{\rm fold}$ (0 to 1)	0.0
		+	area method	
Uprim	real	+	weight per area $U_{\rm prim}$, wing primary structure (lb/ft^2 or kg/m^2)	5.
Uext	real	+	weight per area $U_{\rm ext}$, wing extension (lb/ft^2 or kg/m^2)	3.
		+	weight factors (fraction total wing weight)	
fWfair	real	+	fairing $f_{\rm fair}$	0.10
fWfit	real	+	fittings $f_{\rm fit}$	0.12
fWflap	real	+	flaps and control surfaces $f_{\rm flap}$	0.10
fWfold	real	+	wing fold $f_{\rm fold}$	0.0
fWefold	real	+	wing extension fold $f_{\rm efold}$ (fraction wing extension weight)	0.0
		+	Custom Weight Model	
WtParam_wing(8)	real	+	parameters	0.
		+	Wing Group, AFDD Tiltrotor Weight Model	
		+	jump takeoff condition	
CTs_jump	real	+	rotor maximum blade loading C_T/σ	0.20
n_jump	real	+	load factor $n_{\rm jump}$ at SDGW	2.0
Vtip_jump	real	+	rotor tip speed (0. to use hover $V_{\rm tip}$)	750.0
thickTR	real	+	wing airfoil thickness-to-chord ratio τ_w	0.23
		+	width of wing structural attachments to body	
SET_Attach	int	+	definition (0 input wAttach, 1 fraction fuselage width, 2 fraction wing span)	1
fAttach	real	+	fraction width $w_{\rm attach}/w_{\rm fus}$	1.
wAttach	real	+	width $w_{\rm attach}$ (ft or m)	0.
fRG_pylon	real	+	pylon radius of gyration $r_{\rm pylon}/R$ (fraction rotor radius)	0.30
		+	wing mode frequencies (per rev, fraction rotor speed)	
freq_beam	real	+	beam bending frequency ω_B	0.5
freq_chord	real	+	chord bending frequency ω_C	0.8
freq_tors	real	+	torsion frequency ω_T	0.9
SET_refrpm	int	+	reference rotor speed (0 from input Vtip_freq, 1 hover $V_{\rm tip}$, 2 cruise $V_{\rm tip}$)	0

Vtip_freq	real	+	rotor tip speed	600.
MODEL_form	int	+	form factors (1 calculate, 2 input)	1
form_beam	real	+	torque box beam bending F_B	0.6048
form_chord	real	+	torque box chord bending F_C	0.4874
form_tors	real	+	torque box torsion F_T	1.6384
form_spar	real	+	spar caps vertical/horizontal bending F_{VH}	0.5018
eff_spar	real	+	spar structural efficiency e_{sp}	0.8
eff_box	real	+	torque box structural efficiency e_{tb}	0.8
		+	tapered spar cap correction factors	
C_t	real	+	weight correction C_t (equivalent stiffness)	0.75
C_j	real	+	weight correction C_j (equivalent strength)	0.50
C_m	real	+	strength correction C_m (equivalent stiffness)	1.5
		+	material (lb/in^2, in/in, lb/in^3; or N/m^2, m/m, kg/m^3)	
E_spar	real	+	spar modulus E_{sp}	10.E6
E_box	real	+	torque box modulus E_{tb}	10.E6
G_box	real	+	torque box shear modulus G_{tb}	4.0E6
StrainU_spar	real	+	spar ultimate strain allowable ϵ_U	0.01
StrainU_box	real	+	torque box ultimate strain allowable ϵ_U	0.01
density_spar	real	+	density spar cap ρ_{sp}	0.06
density_box	real	+	density torque box ρ_{tb}	0.06
		+	weight per area (lb/ft^2 or kg/m^2)	
Ufair	real	+	fairing U_{fair}	2.
Uflap	real	+	flaps and control surfaces U_{flap}	3.
UextTR	real	+	wing extension U_{ext}	3.
		+	weight factor	
fWfitTR	real	+	fittings f_{fit} (fraction maximum thrust of one rotor)	0.01
fWfoldTR	real	+	wing fold f_{fold} (fraction total wing weight excluding fold)	0.0
fWefoldTR	real	+	wing extension fold f_{efold} (fraction wing extension weight)	0.0

---

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

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

thickTR only used for tiltrotor wing weight

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

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

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)	1
		+	Geometry	
SET_tail	c*16	+	specification	'vol+aspect'
area	real	+	area S	
span	real	+	span b	
chord	real	+	chord c	
AspectRatio	real	+	aspect ratio AR	
TailVol	real	+	tail volume V	
TailVol2	real	+	second tail volume V	
KIND_TailVol	int	+	tail volume reference (1 wing, 2 rotor)	2
TailVolRef	int	+	wing or rotor number for tail volume	1

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' or both volumes 'both'), ('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))
for canted tail plane, can use both tail volumes (SET_tail='both+xxx')
TailVol based on area $S_t \cos^2 \phi$, horizontal or vertical depending on KIND_tail
TailVol2 based on area $S_t \sin^2 \phi$, vertical or horizontal
tail area from maximum of the two requirements

		+	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
		+	Geometry	
loc_tail	Location	+	aerodynamic center location	
cant	real	+	cant angle ϕ (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 δ_e or rudder δ_r	
INPUT_cont	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_cont(ncontmax,nstatemax)	real	+	control matrix	
nVcont	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 2 piecewise linear, maximum nvelmax)	0
cont(nvelmax)	real	+	values	
Vcont(nvelmax)	real	+	speeds (CAS or TAS)	
		+	incidence i	
INPUT_incid	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_incid(ncontmax,nstatemax)	real	+	control matrix	
nVincid	int	+	number of speeds (0 zero value; 1 constant; $\geq$ 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
		+	Weight	
		+	tail (empennage group)	
MODEL_weight	int	+	model (0 input, 1 AFDD, 2 custom)	1
		+	weight increment	
dWtail	real	+	basic	0.0
dWfold	real	+	fold	0.0
Vdive	real	+	design dive speed V_{dive} (TAS)	200.
		+	Technology Factors	
TECH_tail	real	+	tail weight χ_{ht} or χ_{vt}	1.0
TECH_tfold	real	+	fold weight χ_{fold}	1.0

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

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

		+	Tail Aerodynamics, Standard Model	
AoA_zl	real	+	zero lift angle of attack α_{zl} (deg)	0.
CLmax	real	+	maximum lift coefficient $C_{L\max}$	1.
		+	lift	
SET_lift	int	+	specification (2 2D $dC_L/d\alpha$; 3 3D $dC_L/d\alpha$)	2
dCLda	real	+	lift curve slope $C_{L\alpha} = dC_L/d\alpha$ (per rad)	5.73
Tind	real	+	lift curve slope non-elliptical loading correction τ	0.25

Eind	real	+	Oswald efficiency e ($C_{Di} = (C_L - C_{L0})^2/(\pi e AR)$)	0.8
CL_Dmin	real	+	lift coefficient for minimum induced drag C_{L0}	0.0
eta0	real	+	control effectiveness factor η_0, $\eta_0 - \eta_1\|\delta\|$	0.85
eta1	real	+	control effectiveness factor η_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

		+	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 $\alpha_{D\min}$ (deg)	0.
Kdrag	real	+	drag increment K_d	0.
Xdrag	real	+	drag increment X_d	2.
		+	transition from forward flight drag to vertical drag	
AoA_tran	real	+	angle of attack for transition α_t (deg)	25.

		+	Tail, 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)	1
		+	vertical tail	
MODEL_Vtail	int	+	model (1 helicopter or compound, 2 tiltrotor or tiltwing)	1
place_AntiQ	int	+	antitorque placement (0 none, 1 on tail boom, 2 on vertical tail)	1
fTfold	real	+	fold weight factor f_{fold} (fraction total tail weight excluding fold)	0.0
		+	Custom Weight Model	
WtParam_tail(8)	real	+	parameters	0.

Chapter 25

Structure: FuelTank

Variable	Type		Description	Default
		+	Fuel Tank	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Configuration	
		+	fuel tank sizing	
Wfuel_cap	real	+	fuel capacity $W_{\mathrm{fuel-cap}}$ (weight)	
fFuel_cap	real	+	ratio capacity to mission fuel $f_{\mathrm{fuel-cap}}$	1.0
fFuelWing(nwingmax)	real	+	fraction wing torque box filled by fuel tanks	1.0
fuel_density	real	+	fuel weight per volume ρ_{fuel} (lb/gallon or kg/liter)	6.5
			fuel tank sizing: usable fuel capacity Wfuel_cap (weight) SET_tank='input': input Wfuel_cap SET_tank='miss': calculate from mission fuel burned Wfuel_cap = max(fFuelCap*(maximum mission fuel), (maximum mission fuel)+(reserve fuel))	
		+	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 ntankmax)	1
Waux_cap(ntankmax)	real	+	fuel capacity $W_{\mathrm{aux-cap}}$ (weight)	1000.
fWauxtank(ntankmax)	real	+	tank weight f_{auxtank} (fraction auxiliary fuel weight)	0.
DoQ_auxtank(ntankmax)	real	+	drag $(D/q)_{\mathrm{auxtank}}$ (each tank)	
loc_auxtank(ntankmax)	Location	+	location	

		+	Weight	
		+	fuel system (propulsion group)	
MODEL_weight	int	+	model (0 input, 1 AFDD, 2 custom)	1
		+	weight increment	
dWtank	real	+	tanks and support	0.
dWplumb	real	+	plumbing	0.
		+	Technology Factors	
TECH_tank	real	+	fuel tank weight χ_{tank}	1.0
TECH_plumb	real	+	plumbing weight χ_{plumb}	1.0

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

		+	Fuel System, AFDD Weight Model	
		+	fuel tank	
ntank	int	+	number of internal tanks N_{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_{plumb}	4
K0_plumb	real	+	weight increment $K_{0\mathrm{plumb}}$	150.
K1_plumb	real	+	weight factor $K_{1\mathrm{plumb}}$	2.0

K1_plumb is a crashworthiness and survivability factor; typically K1_plumb = 2.

K0_plumb is the sum of weights for auxiliary fuel, in-flight refueling, pressure refueling, inerting system, etc.; typically K0_plumb = 50 to 250 lb.

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

Structure: Propulsion

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

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

		+	Engine groups	
nEngineGroup	int	+	number of engine groups (maximum nengmax)	1
		+	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)
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
first engine group source of parameters for sizing

		+	Transmission losses	
MODEL_Xloss	int	+	model (1 fraction component power required; 2 with function drive shaft rating)	2
fPloss_xmsn	real	+	gear box loss $\ell_{\rm xmsn}$ (fraction total component power required)	0.04
Ploss_windage	real	+	power loss due to windage $P_{\rm windage}$	0.0
		+	Accessory losses	
Pacc_0	real	+	power loss $P_{\rm acc0}$, constant	0.0
Pacc_d	real	+	power loss $P_{{\rm acc}d}$, scale with density	0.0
Pacc_n	real	+	power loss $P_{{\rm acc}n}$, scale with density and rpm	0.0
Pacc_deice	real	+	deice power loss $P_{{\rm acc}i}$	0.0
fPacc_ECU	real	+	ECU (etc.) power loss $\ell_{\rm acc}$ (fraction component+transmission power)	0.0
fPacc_IRfan	real	+	IRS fan loss $\ell_{\rm IRfan}$ (fraction total engine power)	0.0
		+	Geometry	
SET_length	int	+	drive shaft length (1 input; 2 calculated)	2
Length_ds	real	+	length ℓ_{DS}	
fLength_ds	real	+	factor	0.9

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

		+	Drive system torque limits	
SET_limit_es(nengmax)	int	+	engine shaft (0 input, 1 fraction power, 2 fraction drive system rating)	1
Plimit_ds	real	+	drive system power rating $P_{DS\rm limit}$	
Plimit_es(nengmax)	real	+	engine shaft power rating $P_{ES\rm limit}$	
fPlimit_ds	real	+	drive system power rating factor	1.0
fPlimit_es(nengmax)	real	+	engine shaft power rating factor	1.0

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$\sum(N_{\rm eng}P_{\rm eng})$
- SET_limit_ds='Pav': from engine power available at transmission sizing conditions and missions (DESIGN_xmsn)
  fPlimit_ds$(\Omega_{\rm ref}/\Omega_{\rm prim})\sum(N_{\rm eng}P_{av})$

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

drive system power rating: 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 FltAircraft%SET_Plimit=on)
engine shaft power rating: corresponds to all engines of engine group (nEngine $\times$ Peng)
limits engine group P_{av} (if FltAircraft%SET_Plimit=on)
rotor shaft power rating: corresponds to one rotor
all ratings
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 FltAircraft%fPower
next torque limits are applied (unless FltAircraft%SET_Plimit=off), first engine shaft rating and then drive system rating

		+	Weight	
		+	drive system (propulsion group)	
MODEL_DS	int	+	model (0 input, 1 AFDD, 2 custom)	1
		+	weight increment	
dWgb	real	+	gear box	0.
dWrs	real	+	rotor shaft	0.
dWds	real	+	drive shaft	0.
dWrb	real	+	rotor brake	0.
dWcl	real	+	clutch	0.
dWgd	real	+	gas drive	0.

STATE_gear_wt	int	+	drive system state for weight	1
kEngineGroup_wt	int	+	EngineGroup for weight	1
		+	Technology Factors	
TECH_gb	real	+	gear box weight χ_{gb}	1.0
TECH_rs	real	+	rotor shaft weight χ_{rs}	1.0
TECH_ds	real	+	drive shaft weight χ_{ds}	1.0
TECH_rb	real	+	rotor brake weight χ_{rb}	1.0
TECH_cl	real	+	clutch weight χ_{cl}	1.0
TECH_gd	real	+	gas drive weight χ_{gd}	1.0

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

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

		+	Drive System, 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

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

Structure: EngineGroup

Variable	Type		Description	Default
		+	Engine Group	
title	c*100	+	title	
notes	c*1000	+	notes	
		+	Description	
nEngine	int	+	number of engines $N_{\rm eng}$	1
nEngine_main	int	+	number of main engines	1
IDENT_engine	c*16	+	engine identification	'Engine'
MODEL_engine	int	+	engine model (1 RPTEM)	1
INPUT_gear	int	+	gear ratio input (1 from Nspec, 2 gear)	1
gear(ngearmax)	real	+	engine gear ratio $r = \Omega_{\rm spec}/\Omega_{\rm prim}$ (ratio rpm to rpm of primary rotor in propulsion group)	1.0
SET_power	int	+	specification (0 sized, 1 fixed)	0
fPsize	real	+	sized power ratio f_n	1.0
Peng	real	+	engine power $P_{\rm eng}$ (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'

---

engine identification: match ident of EngineModel input
number of main engines: for fuel tank weight
INPUT_gear: calculate gear from Nspec and Vtip_ref of primary rotor of propulsion group, or specify gear ratio
variable speed transmission: for drive system state STATE_gear_var, gear ratio factor $f_{\rm gear}$ (control) included when evaluate rotational speed of engine

takeoff power rating: for engine scaling; aircraft power loading; fuel tank weight
FltAircraft%rating can be set to 'idle' (rating_idle) or 'takeoff' (rating_to)

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.

		+	Installation	
eta_d	real	+	engine inlet efficiency η_d (fraction, for δ_M)	0.98
Kffd	real	+	deterioration factor on engine fuel flow K_{ffd}	1.05
		+	power losses (fraction power available, P_{loss}/P_a)	
fPloss_inlet	real	+	engine inlet loss ℓ_{in}	0.0
fPloss_ps	real	+	inlet particle separator loss ℓ_{in}	0.0
fPloss_exh	real	+	engine exhaust loss ℓ_{ex} (IRS off)	0.015
		+	auxiliary air momentum drag (IRS off)	
fMF_auxair	real	+	mass flow f_{aux} (fraction engine mass flow)	0.007
eta_auxair	real	+	ram recovery efficiency η_{aux}	0.75
		+	IR suppressor	
		+	power losses (fraction power available, P_{loss}/P_a)	
fPloss_exh_IRon	real	+	engine exhaust loss ℓ_{ex} (IRS on)	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 η_{aux}	0.75

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

		+	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, $k = S_{\text{wet}}/(w/N_{\text{eng}})^{2/3}$ (Units_Dscale)	0.8

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

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

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

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

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

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

		+	Weight	
		+	engine weight	
MODEL_weight	int	+	model (0 input, 1 RPTEM, 2 custom)	1
dWEng	real	+	weight increment	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
		+	Technology Factors	
TECH_eng	real	+	engine weight χ_{eng}	1.0
TECH_cowl	real	+	engine cowling weight χ_{cowl}	1.0
TECH_pylon	real	+	pylon structure weight χ_{pylon}	1.0
TECH_supt	real	+	engine support structure weight χ_{supt}	1.0
TECH_air	real	+	air induction system weight χ_{airind}	1.0
TECH_exh	real	+	exhaust system weight χ_{exh}	1.0
TECH_acc	real	+	engine accessories weight χ_{acc}	1.0

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

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

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

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

DoQV	real	+	area $(D/q)_V$	
CDV	real	+	coefficient C_{DV} (based on wetted area, $D/q = SC_D$)	
		+	transition from forward flight drag to vertical drag	
Xdrag	real	+	exponent X_d	2.0

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

		+	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 engine, including IR suppressor; W_{exh} vs P_{0C}	
Kwt0_exh	real	+	$K_{0\text{exh}}$	0.0
Kwt1_exh	real	+	$K_{1\text{exh}}$	0.002
		+	engine accessories	
MODEL_lub	int	+	lubrication system weight (1 in engine weight, 2 in accessory weight)	1

typically fWair = 0.3 (data range 0.1 to 0.6)

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

Structure: EngineModel

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

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

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

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

		+	Custom Weight Model	
WtParam_engine(8)	real	+	parameters	0.
		+	Parameters	
		+	Engine Ratings	
nrate	int	+	number of engine ratings (maximum nratemax)	1
rating(nratemax)	c*12	+	engine rating designations	'MCP'
		+	Reference	
P0_ref(nratemax)	real	+	power (P_0)	2000.
SP0_ref(nratemax)	real	+	specific power (SP_0)	150.
Pmech_ref(nratemax)	real	+	mechanical limit of power (P_{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 ($N_{\mathrm{opt0}C}$)	20000.

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

		+	Technology	
SP0C_tech	real	+	specific power at MCP SP_{tech} (0. for SP0_ref(MCP))	0.
sfc0C_tech	real	+	specific fuel consumption at MCP $\mathrm{sfc}_{\mathrm{tech}}$ (0. for sfc0C_ref)	0.
Nspec_tech	real	+	specification turbine speed N_{tech} (0. for Nspec_ref)	0.
		+	Scaling	
MF_limit	real	+	mass flow at limit SP and sfc ($\dot{m}_{\mathrm{lim}}$)	30.

SP0C_limit	real	+	specific power limit $SP_{\lim}$	200.
sfc0C_limit	real	+	specific fuel consumption limit $\mathrm{sfc}_{\lim}$	0.34
KNspec	real	+	specification turbine speed variation (K_{Ns2})	0.

SP and sfc functions are defined by values SP0C_tech, sfc0C_tech, $\dot{m}_{\mathrm{tech}}$=P0C_ref/SF0C_tech
and limits SP0C_limit, sfc0C_limit, MF_limit
defaults SP0C_tech=SP0_ref(MCP) and sfc0C_tech=sfc0C_ref
require $\dot{m}_{\mathrm{tech}} < \dot{m}_{\lim}$ (otherwise get SP_{0C} = SP0C_tech and sfc_{0C} = sfc0C_tech)
for no variation with scale, use MF_limit=0. (or SP0C_limit=SP0C_tech and sfc0C_limit=sfc0C_tech)

		+	Optimum Power Turbine Speed (used only for INPUT_param = single set)	
MODEL_OptN	int	+	model (1 linear, 2 cubic)	1
		+	linear, $N_{\mathrm{opt}}/N_{\mathrm{spec}}$ vs P_q/P_0	
KNoptA	real	+	constant $K_{N\mathrm{opt}A}$	1.
KNoptB	real	+	constant $K_{N\mathrm{opt}B}$	0.
		+	cubic, $N_{\mathrm{opt}}/N_{\mathrm{opt}0C}$ vs P_q/P_{0C}	
KNopt0	real	+	constant $K_{N\mathrm{opt}0}$	1.
KNopt1	real	+	constant $K_{N\mathrm{opt}1}$	0.
KNopt2	real	+	constant $K_{N\mathrm{opt}2}$	0.
KNopt3	real	+	constant $K_{N\mathrm{opt}3}$	0.
XNopt	real	+	exponent $X_{N\mathrm{opt}}$	0.
		+	power turbine efficiency function, $\eta_t(N)/\eta_t(N_{\mathrm{spec}})$	
XNeta	real	+	exponent $X_{N\eta}$	2.0
		+	Power Available and Power Required Parameters	
INPUT_param	int	+	parameter input form (1 single set; 2 interpolated)	1
		+	interpolated	
nspeed	int	+	number of engine speeds (maximum nspeedmax)	1
rNeng(nspeedmax)	real	+	engine speed ratio, N/N_{spec}	1.

interpolated: rNeng unique and sequential

Structure: EngineParam

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

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

input θ_b, K_b: θ_b unique and sequential

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

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

Structure: Location

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

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