NDARC

NASA Design and Analysis of Rotorcraft

User Training

March 2014

PREPARED BY

Christopher Silva
Jeffrey Sinsay
U.S. Army Aviation Development Directorate (AFDD)
Aviation & Missile Research, Development & Engineering Center
Research, Development and Engineering Command

Wayne Johnson
National Aeronautics and Space Administration

Ames Research Center, Moffett Field, California 94035

Training Objectives

Purpose:

- Provide overview of how NDARC conducts design and analysis tasks
- Introduce NDARC input/output construct and syntax
- Explain steps necessary to accomplish common design / analysis tasks
- Illuminate common pitfalls and methods for debugging input

Assumptions:

- Familiar with the preliminary design process
- Understand classic helicopter and fixed wing theory that underpins NDARC analysis (see Theory manual for details)
- Familiar with FORTRAN namelist input

Outcomes:

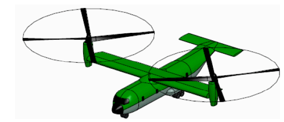
- User can execute and parse input/output of NDARC examples
- User can create a new design by modifying an example
- User can use Theory manual and Input manual to set up more complex cases

Outline

- Introduction
- Documentation
- Overview
  - Tasks
  - Aircraft
- NDARC Job
  - Input
  - Organization
  - Output
- Solution Procedures
  - Debugging
- Input Manual
  - Aircraft
  - Tasks
- Tutorial

Rotorcraft System Design and Analysis

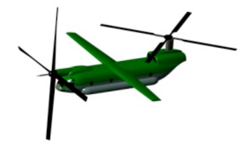
Rotorcraft design work in government laboratory supports research and acquisition



NASA Rotary Wing Project requires system analysis tool

- For technology impact assessments, and to support research investments
- To define context for research and development, and support design efforts

Department of Defense acquisition phases require rotorcraft design work



- For concept exploration, decision, and refinement, and for technology development
- Perform quantitative evaluation and independent synthesis of wide array of aircraft configurations and concepts
- Provide foundation for specification and requirement development

Tools previously available to government have out-of-date software and limited capabilities

Proprietary tools of helicopter industry not available to government

Rotorcraft system analysis developed by NASA and US Army AFDD

- NDARC — NASA Design and Analysis of Rotorcraft

NDARC
NASA Design and Analysis of Rotorcraft

Principal Tasks:

- **Facilitate design of new rotorcraft concepts**
  - Synthesis: Create new concepts from library of components
  - Size: Parametrically vary components to meet specified requirements
- **Analyze rotorcraft air vehicle systems**
  - Point Performance: Calculate performance at specified flight conditions
  - Mission Performance: Fuel burn and time to accomplish variety of missions

Critical Attributes:

- **Rapid Turnaround** – Execute case on order of minutes
- **Flexible Sizing Constraints** – Sized based on multiple missions and performance points
- **Configuration Generality** – Goverment activities require ability to model broad array of rotorcraft concepts
- **Capture Technology Impact** – Must be able to consider new technology at a system and component level
- **Extensible** – Analysis capability and code architecture should not inhibit creativity, easy modification for individual projects
- **Documentation** – Complete and thorough documentation of theory and code

Software

Design and development of NDARC started in January 2007

- Release 1.0 in May 2009, coding effort about 1.5 man-years
- Release 1.8 in February 2014

Distribution controlled by the Software Release Authority at Ames Research Center

- Source code and documentation available to users, subject to Software Usage Agreement

Input: namelist-based text format

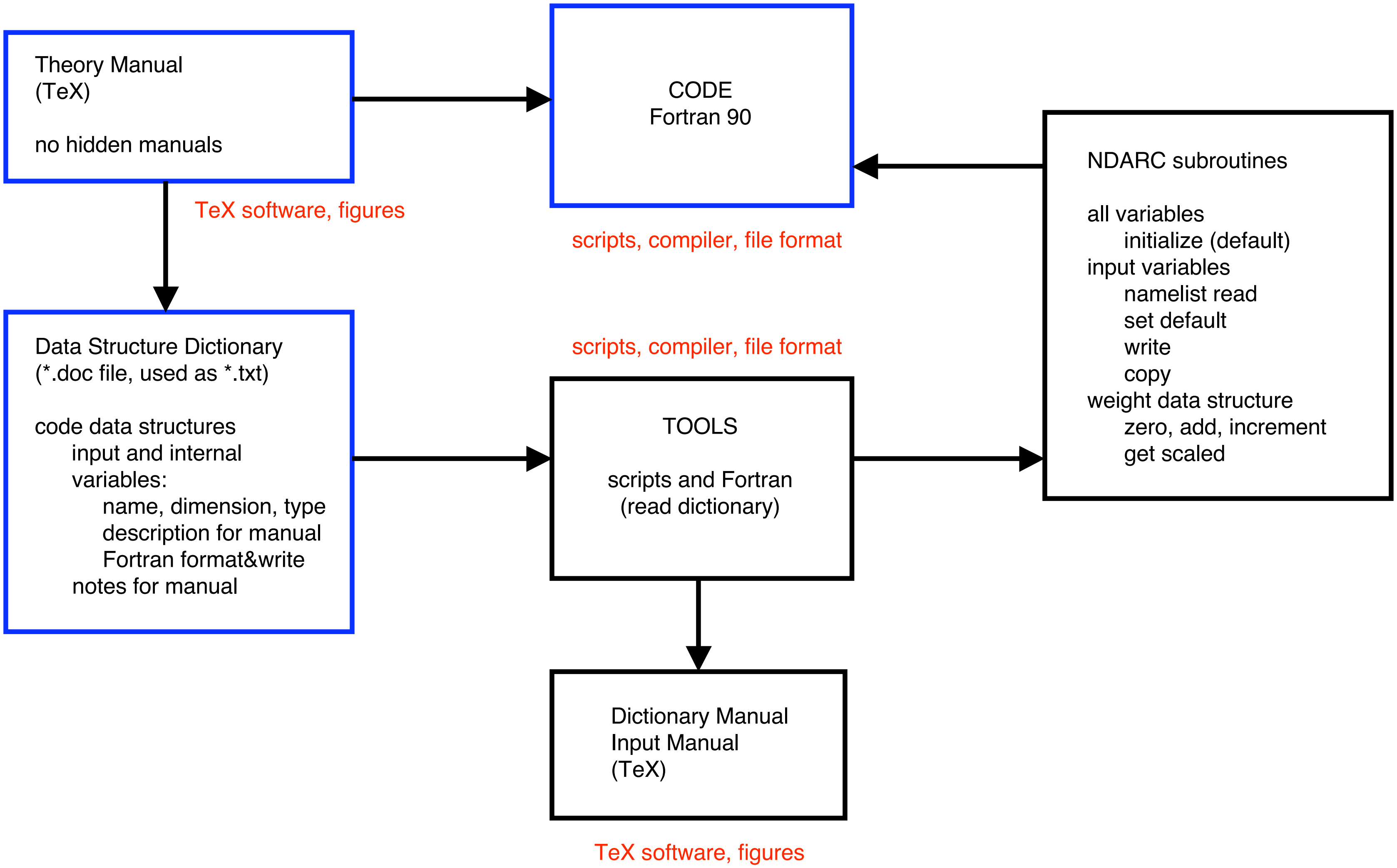
Output: text files formatted for printing and for spreadsheets, and special files (e.g. for preparation of layout drawings)

Execution times: seconds for job with just few analysis tasks; minutes for job that sizes aircraft based on multiple flight conditions and missions

NDARC written in Fortran 95

- Using special-purpose software tool
  - Tool is Fortran code utilizing character string manipulation
  - Manage data structures (in module), generate input manual, generate code to read and print input
- Compiled on several platforms and operating systems

NDARC Development Process



Validation and Demonstration

Validation: exercise ANALYSIS tasks

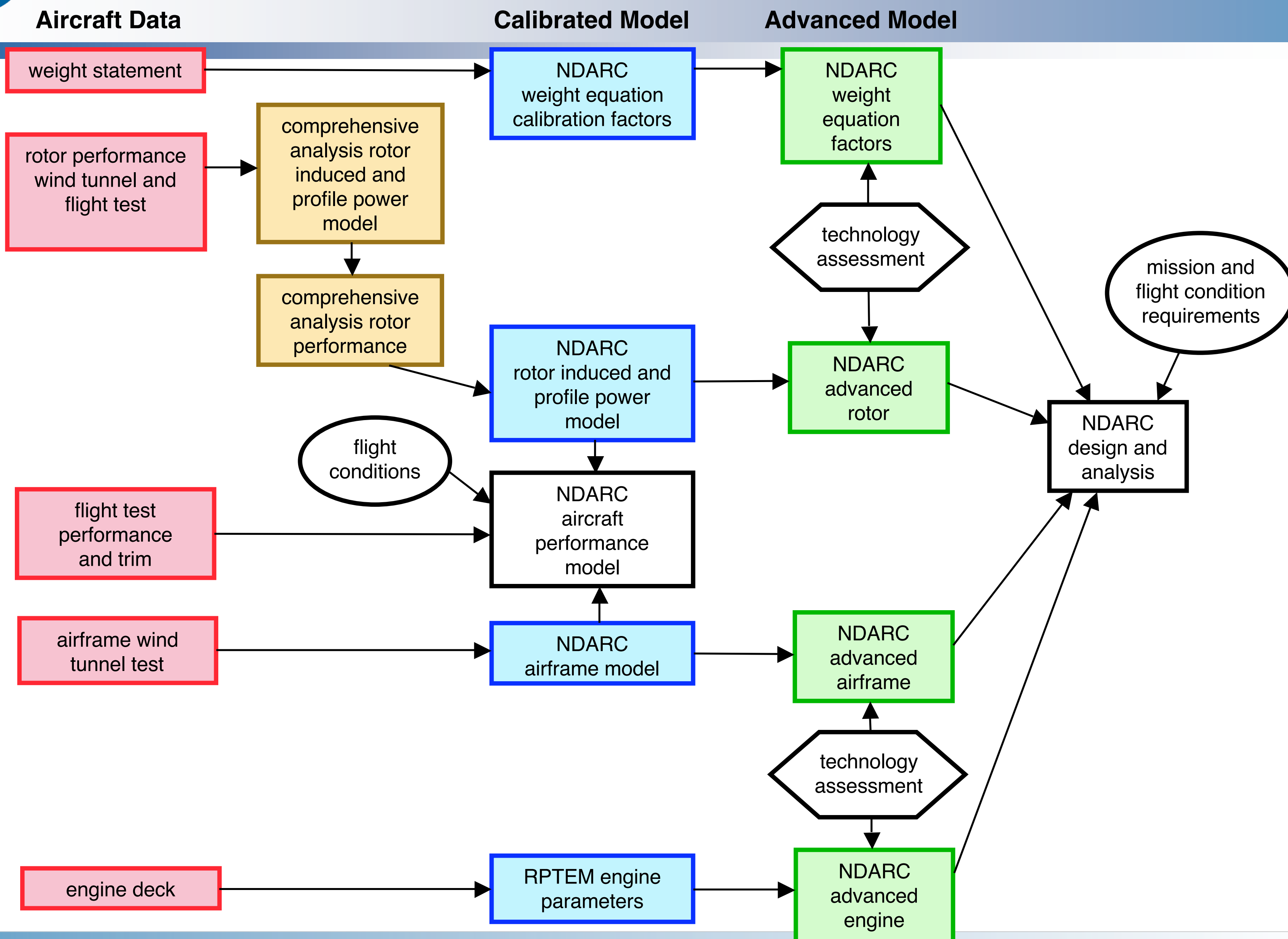
- **Develop NDARC models by using geometry and weight information, airframe wind tunnel test data, engine decks, rotor tests, and comprehensive analysis results**
  - **Correlate performance calculations from comprehensive analysis with wind tunnel or flight test data**
  - **Develop parameters of NDARC rotor performance model based on calculated induced power factor κ and mean drag coefficient $c_{d\,\mathrm{mean}}$**
- **Compare NDARC results for aircraft and component performance with flight test data**

Rotorcraft comprehensive analysis used: CAMRAD II

Demonstration: exercise DESIGN task

- **Based on the calibrated models, capability to size rotorcraft explored**
- **Part of code development, not design project**

Validation Process



Development Test Cases

Aircraft

- **UH-60A (single main-rotor and tail-rotor helicopter)**
- **CH-47D (tandem helicopter)**
- **XH-59A (coaxial lift-offset helicopter)**
- **XV-15 (tiltrotor)**

Selected because for each aircraft have:

- **Flight performance data**
- **Weight statement**
- **Detailed geometry**
- **Correlated comprehensive analysis model**

"NDARC — NASA Design and Analysis of Rotorcraft. Validation and Demonstration." American Helicopter Society Specialists' Conference on Aeromechanics, San Francisco, CA, January 2010

NDARC Configurations

Turboshaft Engine

Other Propulsion Systems

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

- Debugging

Input Manual

- Aircraft
- Tasks

Tutorial

Documentation

Complete and thorough documentation of theory and code

- Theory manual documents NDARC component methods and solution procedures
  - NASA TP 2009-215402 (release 1.0 theory)
  - "NDARC — NASA Design and Analysis of Rotorcraft. Theoretical Basis and Architecture." American Helicopter Society Specialists' Conference on Aeromechanics, San Francisco, CA, January 2010
  - "Propulsion System Models for Rotorcraft Conceptual Design." Fifth Decennial AHS Aeromechanics Specialists' Conference, San Francisco, CA, January 2014
- Input manual documents input syntax and variable names, provides guidance on the selection of component and solution methods
- Dictionary documents all data structures (including input parameters)

Documentation available in NDARC distribution package

Sample cases

- **Provide examples of common rotorcraft configurations**
- **Serve as point of departure for creating new models**

NDARC Wiki available for collaboration and community use

https://wiki.nasa.gov/ndarc-nasa-design-and-analysis-of-rotorcraft

Sample Cases in Distribution Package

aircraft	description	engine	jobs	size	notes
helicopter	helicopter.list	gen2000.list	helicopter.run	no	
			size_eng.run	engine	payload-range
			size_rotor.run	rotor	
			takeoff_hel.run	no	takeoff performance
tandem	tandem.list	gen4000.list	tandem.run	no	
coaxial	coaxial.list	gen2000.list	coaxial.run	no	
tiltrotor	tiltrotor.list	gen2000.list	tiltrotor.run	yes	
			tiltrotor_ext.run	no	wing extensions
			takeoff_tr.run	no	takeoff performance
compound	compound.list	gen4000.list	compound.run	yes	based on helicopter model
autogyro	autogyro.list	turbojet	autogyro.run	yes	

Data Dictionary (Input Manual) Layout

Common: Job ← Structure name

Variable	Type		Description	Default
			NDARC	
			Version (set by main program)	
version	c*6		number n.n	
modification	c*32		modification	
versionout	c*64		string for headers (`Version n.n, modification "xxx"`)	
		+	Initialization	
INIT_input	int	+	input parameters (0 default, 1 last case input, 2 last case solution)	1
INIT_data	int	+	other parameters (0 default, 1 start of last case, 2 end of last case)	0

Valid flag values

\+ identifies input value

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

Clarifying comments on input usage

Variable	Type		Description	Default
		+	Errors	
ACT_error	int	+	action on error (0 none, 1 exit)	1
ACT_version	int	+	action on version mismatch in input (0 none, 1 exit)	0
		+	File open	
OPEN_status	int	+	status keyword for write (0 unknown, 1 replace, 2 new, 3 old)	2

default value

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

- Debugging

Input Manual

- Aircraft
- Tasks

Tutorial

NDARC Tasks

Job consists of one or more cases, each case optionally performing design and analysis tasks

- Design task: Size rotorcraft to satisfy specified design conditions and missions

- Analysis tasks: Off-design mission performance, flight performance for point operating conditions
- Source of aircraft description for analysis task:
  - From sizing task
  - From previous case or previous NDARC job
  - Or independently generated (such as description of existing aircraft)

- Maps: Generate performance maps for airframe aerodynamics or engine

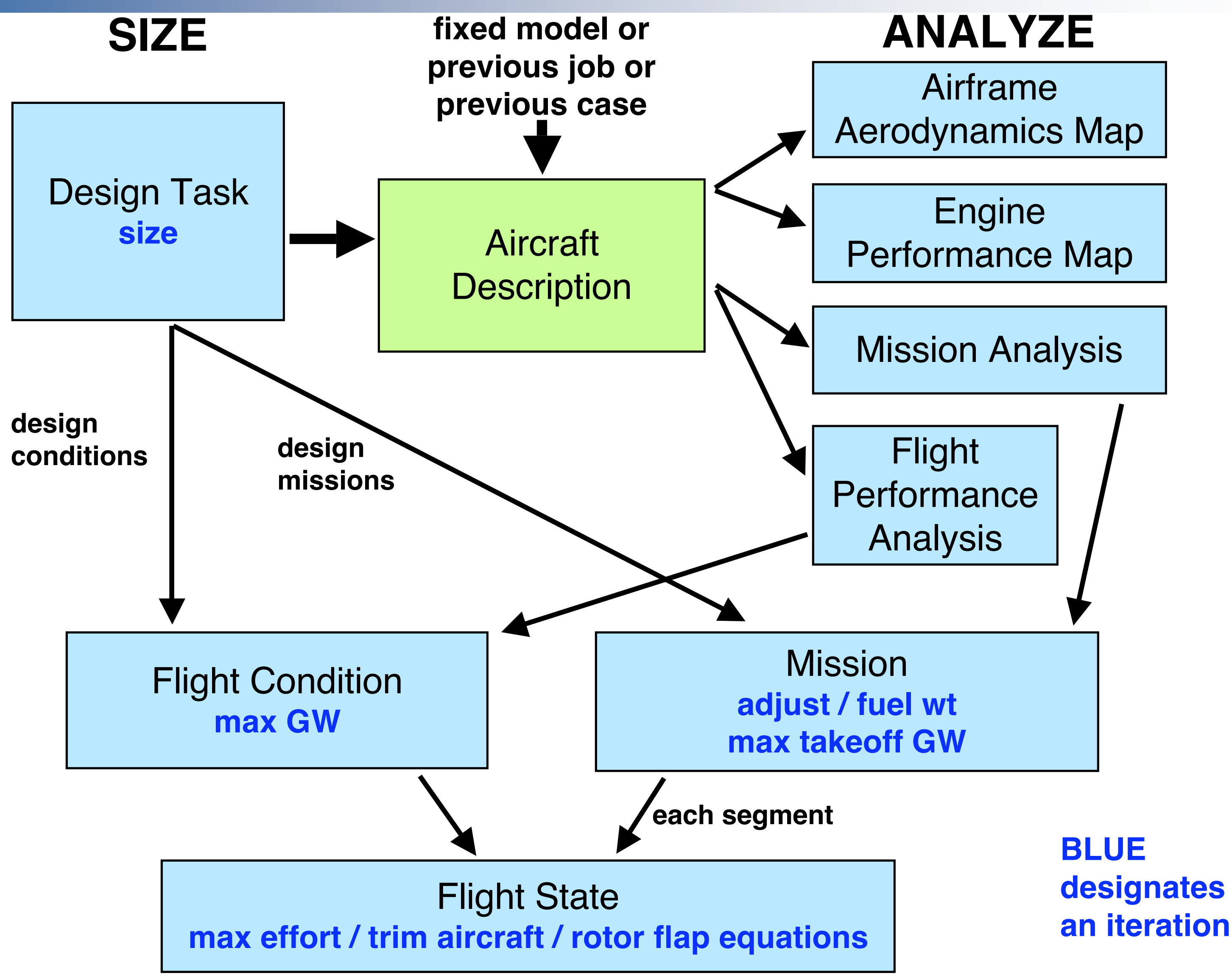
Description and analysis of conventional rotorcraft configurations facilitated

- single main rotor and tail rotor helicopter
- tandem helicopter
- coaxial helicopter
- tiltrotor

Retaining capability to model novel and advanced concepts

- For example, compound rotorcraft: add wings and propellers to aircraft

NDARC Tasks


SIZE
fixed model or
previous job or
previous case
ANALYZE
Design Task
size
Aircraft
Description
Airframe
Aerodynamics Map
Engine
Performance Map
Mission Analysis
Flight
Performance
Analysis
design
conditions
design
missions
Flight Condition
max GW
Mission
adjust / fuel wt
max takeoff GW
each segment
Flight State
max effort / trim aircraft / rotor flap equations
BLUE
designates
an iteration

NDARC Terminology

Job: An execution instance of NDARC, composed of one or more cases.

Case: A collection of tasks that form a cohesive process. The type and number of tasks executed in a case are completely flexible, no one type of task is required in a given case. Typically a case consists of one sizing task and a collection of mission and performance analysis tasks on the aircraft resulting from the sizing task.

Task: A specific procedure NDARC can perform. Valid tasks are: sizing, mission analysis, performance analysis, engine mapping, aerodynamic mapping.

Flight State: Defined for each flight condition or mission segment. The collection of variables that describe speed, motion, atmospheric condition, configuration state (rotor speed, control positions, engine rating, etc) and trim approach.

Flight Condition: A single event for which aircraft performance is calculated.

Sizing Flight Condition: A flight condition event for which the aircraft configuration is parametrically altered (sized) until the aircraft can meet the input performance requested.

Mission: An ordered collection of mission segments (flight states) over which fuel burn and flight time/distance is integrated. Gross weight is adjusted based on fuel burn throughout the mission, and the resulting breakdown of takeoff gross weight into fuel, payload, fixed useful load and operating weight empty can be determined.

Design Mission: A mission analyzed as part of the sizing task, the aircraft configuration is parametrically adjusted (sized) until it can meet the input mission requirements.

Off-Design Mission: A mission analyzed on a fixed aircraft configuration, mission parameters (range and/or payload) must be adjusted until the fuel available at takeoff is converged with the fuel required for the mission.

Component: A discrete element of an aircraft which performs functions and has intrinsic properties (dimensions, weight, aerodynamics loads, etc.).

Size: The process of parametrically adjusting key aircraft design variables (rotor radius, engine size, takeoff gross weight, etc) until all design mission and sizing flight conditions are meet.

Synthesis: The process of assembling components to meet a specified set of requirements. In the context of NDARC this process is performed prior to execution in the construction of the aircraft input file.

Solution Procedure

Sizing Task

Size Iteration

method: successive substitution

- Missions
- Flight Conditions

↓

Mission Analysis

- Missions

↓

Flight Perf Analysis

- Flight Conditions

Flight Condition

Maximum GW

method: secant or false position

- Flight State

Mission

Mission Iteration
adjust, fuel weight

method: successive substitution

Segments

Maximum GW

method: secant or false position

- Flight State

Flight State

Maximum Effort

method: golden section for maximum endurance, range, or climb;
otherwise secant or false position

Trim

method: Newton-Raphson

Component Performance Evaluation

Blade Flapping

method: Newton-Raphson

Sizing Task

"Size the aircraft" means develop consistent description of system

- Sizing task determines dimensions, power, and weight of rotorcraft
  - That can perform specified set of design conditions and missions
- Relations between dimensions, power, and weight generally require iterative solution

From design flight conditions and missions, can determine

- Total engine power or rotor radius
  - Or both power and radius can be fixed
- Design gross weight
- Maximum takeoff weight
- Drive system torque limit
- Fuel tank capacity
- Antitorque or auxiliary-thrust design thrust

Component sizing options

- Choices for independent/dependent design parameters of all components
- Code facilitates parameter variation by automating dependencies

Component Sizing Options

Available choices for independent/dependent design parameters

- Parameters not fixed are dependent (fallout)
- Facilitates parameter variation by automating dependencies

Rotor: fix subset of

- Radius or disk loading, thrust-weighted σ, hover tip speed, blade loading C_W/σ
- Tail rotor radius can be scaled from main rotor radius
- Rotor radius can also be fallout of sizing task

Wing: fix subset of

- Area or wing loading, span, chord, aspect ratio
- Or span obtained from aircraft or rotor geometry, or span of another wing

Tail: fix subset of

- Area or tail volume, span, chord, aspect ratio

From designated sizing conditions and/or missions (or fixed input)

- Fuel tank (missions)
- Maximum takeoff weight (conditions)
- Drive system rating (conditions and missions)

Engine size: power rating scaled (or fixed input)

Missions and Flight Conditions

Missions defined for sizing task, and for mission analysis

- Mission consists of number of mission segments
  - For each segment the time, distance, and fuel burn are evaluated
- Mission parameters include mission takeoff gross weight and useful load
  - Takeoff gross weight can be input, fallout, or maximized (for power required = power available, at designated segment)
- With specified takeoff fuel weight:
  - Mission time or distance adjusted so fuel required for mission (burned plus reserve) = takeoff fuel weight
- Mission iteration is on time/distance (if adjustable), or on fuel weight

Flight conditions specified for sizing task, and for flight performance analysis

- Flight condition parameters include gross weight and useful load
- Gross weight can be input, fallout, or maximized (for power required = power available)

Flight State

Flight state is defined for each mission segment and each flight condition

- Aircraft performance analyzed for specified state, or for maximum effort performance
- Maximum effort can be
  - –speed, rate of climb, altitude (etc.) for
  - –best endurance or best range, or power required = power available (etc.)

Aircraft must be trimmed in specified operating condition

- Solving for controls and motion that produce aircraft force and moment equilibrium (and/or designated quantities ⇒ target value)
- Various flight states can require different trim strategies

Solution of blade flap equations of motion may be required

- To evaluate rotor inplane forces or blade pitch angles

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

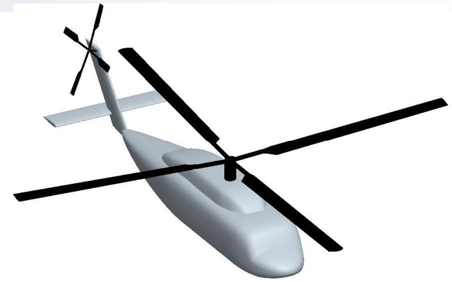
- Debugging

Input Manual

- Aircraft
- Tasks

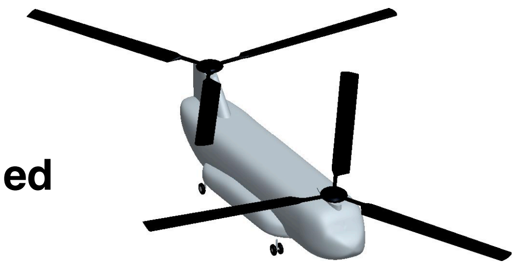
Tutorial

Rotorcraft Configuration

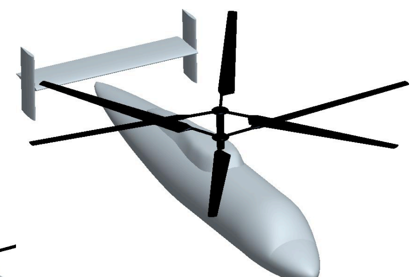


Description and analysis of conventional rotorcraft configurations are facilitated

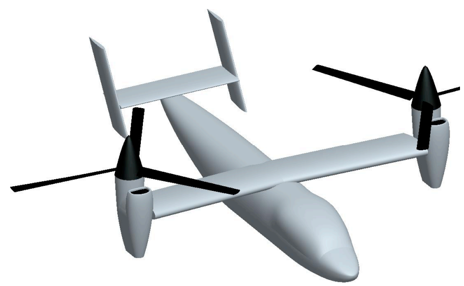
- **Single main rotor and tail rotor helicopter**
- **Tandem helicopter**
- **Coaxial helicopter**
- **Tiltrotor**



Novel and advanced concepts typically modeled by starting with one of these conventional configurations



For example, compound rotorcraft can be constructed by adding wings and propellers



Aircraft Description

Critical to achieving capability to model wide array of rotorcraft concepts is decomposition of aircraft into set of fundamental components

Aircraft consists of set of components

- **Fuselage, Landing Gear, Systems**
- **Rotors**
  - main rotor, tail rotor, propeller
  - tilting, ducted, antitorque, auxiliary-thrust, variable diameter, reaction drive
  - twin rotors
- **Wings**
- **Tails**
  - horizontal or vertical
- **Fuel tanks**
  - fuel quantity measured as either weight or energy
- **Propulsion groups**
  - set of rotors and engine groups, connected by drive system
  - components define power required, engine groups define power available
- **Engine Groups (turboshaft, compressor, electric motor, generator)**
  - transfers power by shaft torque
  - one or more engines of same type
- **Jet Groups (turbojet, turbofan)**
  - produces force on aircraft
- **Charge Groups (fuel cell, solar cell)**
  - Generates energy for the aircraft

Aircraft Component

Control

- Control definition key feature for configuration generality

- Aircraft controls (including pilot's controls) connected to component controls
  - Aircraft controls used for trim
  - Aircraft and component controls: zero, constant, or function of flight speed
  - Tilt control: nacelle tilt or conversion control

- One or more control states
  - Different connections (matrix T), with defaults for each configuration
  - For example: tiltrotor in helicopter mode and airplane mode flight

- Optional conversion schedule (based on flight speed)
  - Defines nacelle tilt, tip speed, control state, drive system state

Aircraft Controls

Control definition key feature for configuration generality

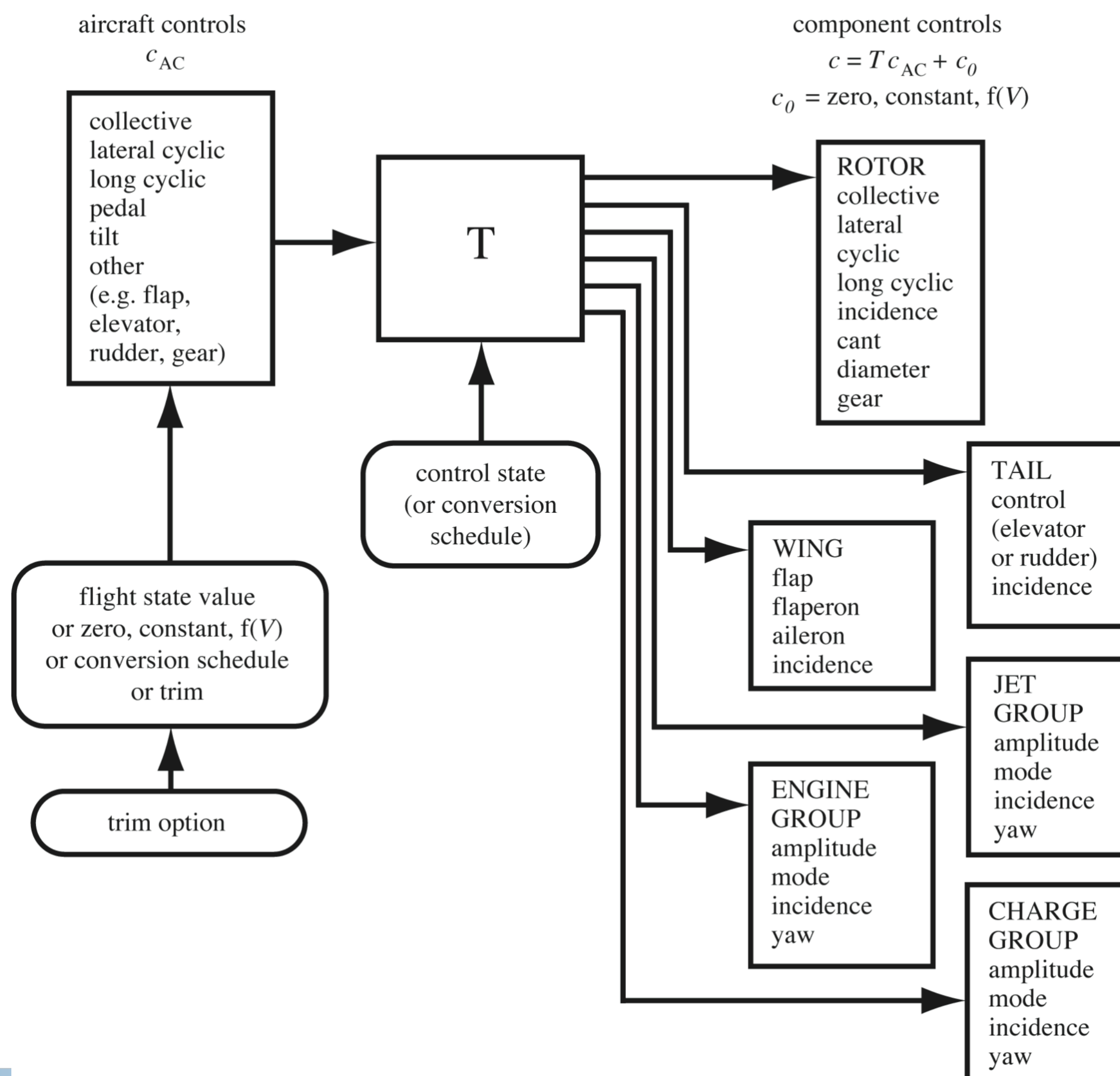
Aircraft controls connected to component controls

Aircraft controls include:

- **Pilot's controls**
- **Configuration variables (e.g. tilt of nacelle/pylon, engine, rotor shaft)**
- **Connections to component controls**

Only pilot's controls set / adjusted for flight condition or mission segment

- **Access to component controls only through matrix T**



Aircraft Controls

Specification of flight state (including trim) uses aircraft controls

Controls defined in *Aircraft* structure

		+	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

Connected to component controls

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

For example, tail aerodynamic control:

		+	Controls	
		+	elevator δ_e or rudder δ_r	
INPUT_cont	int	+	connection to aircraft controls (0 none, 1 input T matrix)	1
T_cont(ncontmax,nstatemax)	real	+	control matrix	
nVcont	int	+	number of speeds (0 zero value; 1 constant; ≥ 2 piecewise linear, maximum nvelmax)	0
cont(nvelmax)	real	+	values	
Vcont(nvelmax)	real	+	speeds (CAS or TAS)	

Aircraft Controls Example

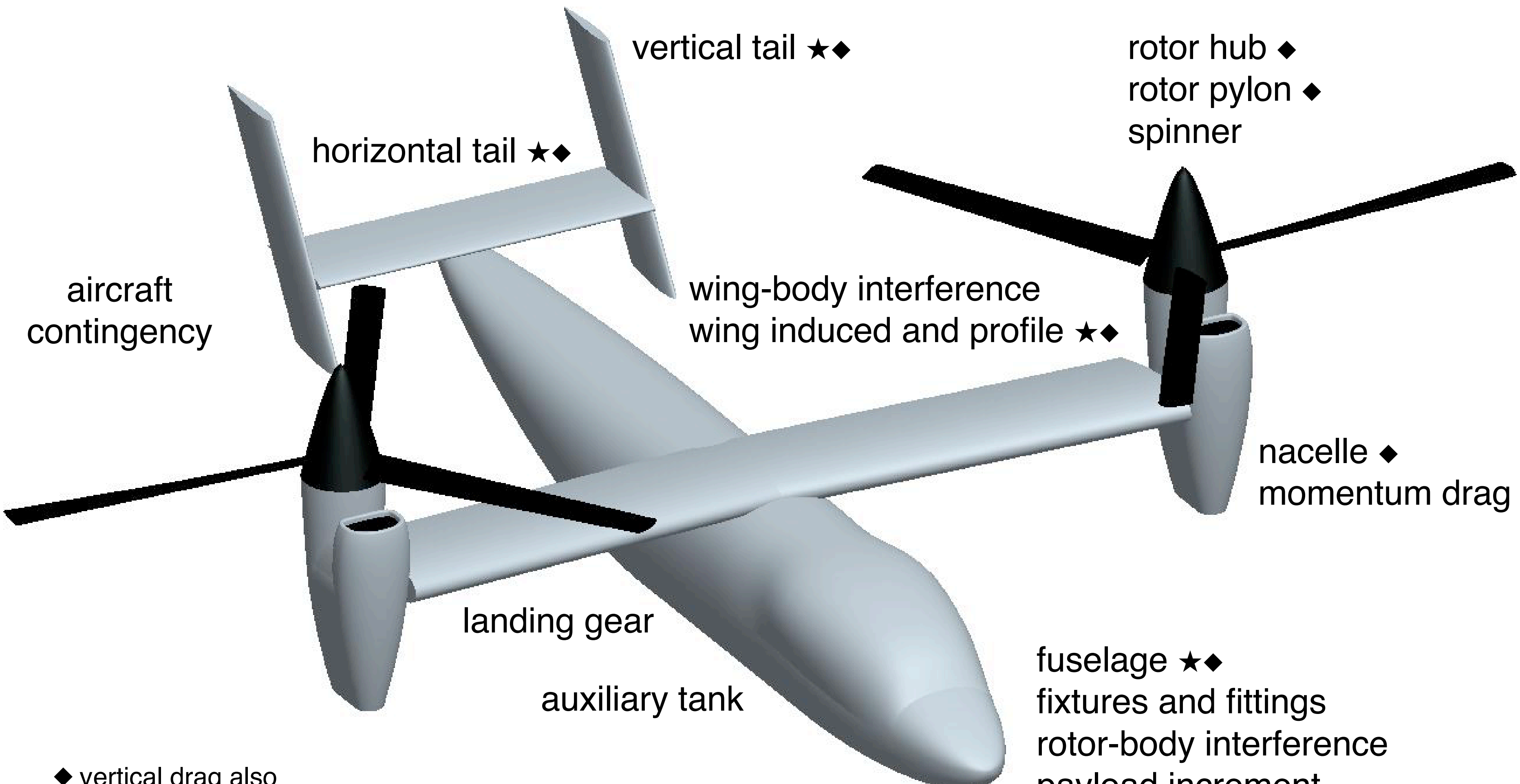
Control Matrix (State 1)

Component			Rotor 1 (RH)				Rotor 2 (LH)				Wing			Tail
Component Ctrl														
	idx	IDENT_control												
Pilot Controls	1	coll	1.0				1.0							
	2	latcyc	-1.0				1.0							
	3	lngcyc		-1.0				-1.0						
	4	pedal		1.0				-1.0						
	5	tilt				1.0				1.0				
	6	flap									1.0			
	7	flaperon										1.0		
	8	elevator												1.0
	9	aileron											1.0	
	10	rudder												

Control Matrix (State 2)

Component			Rotor 1 (RH)				Rotor 2 (LH)				Wing			Tail
Component Ctrl														
	idx	IDENT_control												
Pilot Controls	1	coll	1.0				1.0							
	2	latcyc											-1.0	
	3	lngcyc												1.0
	4	pedal	0.1				-0.1							
	5	tilt				1.0				1.0				
	6	flap									1.0			
	7	flaperon										1.0		
	8	elevator												
	9	aileron											1.0	
	10	rudder												

Drag



◆ vertical drag also
★ function angle of attack

Aerodynamics and Drag

Aerodynamic models define loads either fixed or scaled

- **Fixed:** D/q, L/q, **etc.**
- **Scaled:** CD, CL, **etc.**

Scaled coefficients based on appropriate area S **(and length for moments)**

- CD = D/qS, CL = L/qS
- **Reference area can be input or scaled with size**

Aerodynamic loads vary with angle of attack

- **Stall models for large angle**
- **Vertical drag**

Some components (including fuselage and wing) see aerodynamic interference velocities from rotors

Geometry

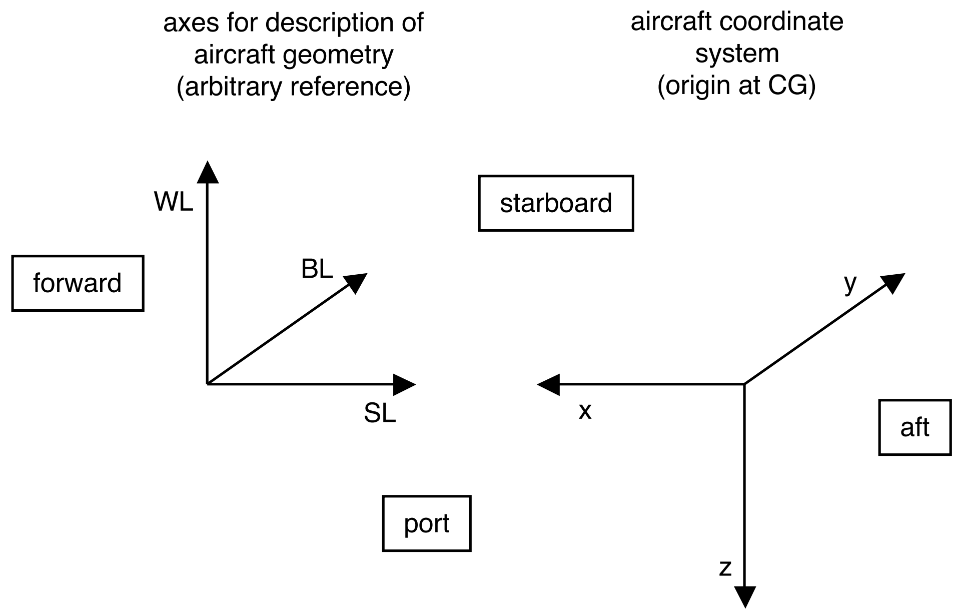
Component position input: fixed or scaled

Fixed:

- Station line, butt line, water line (SL, BL, WL)

Scaled

- x/L, y/L, z/L
- reference length L = rotor radius, wing span, or fuselage length
- relative reference point = input, rotor, wing, fuselage, or center of gravity



Rotor

One or more rotors (or none)

- Designated main, tail, or propeller: weight model, where in weight statement
- Designated antitorque or auxiliary-thrust: special sizing options
- Other configuration features: tilting, ducted, variable diameter, reaction drive

Connected to propulsion group (drive train)

- Set tip speed, drive losses (even if no shaft power source)

Energy method for power: induced + profile + parasite

- In terms of induced power factor and mean drag coefficient
- Including induced power for twin rotors

Inplane forces relative TPP: calculate with blade element theory, or neglect
Profile inplane forces: calculate with blade element theory, or simplified

Rotor interference at other components: fuselage, wings, tails

- Wake-induced velocity at component estimated based on inflow at rotor

Rotor drag (hub, pylon, spinner)

Rotor Control

Rotor Controls: collective, lateral cyclic, longitudinal cyclic

- **Shaft incidence angle (tilt) and cant angle can be controls**

Collective control:

- **Direct command of rotor thrust or C_T/σ (shaft axes)**
  - **simpler solution of rotor equations (inflow determined by thrust)**
  - **calculate collective pitch angle from thrust (BE theory)**
    - May encounter flight states where commanded thrust can not be produced by rotor, so solution for collective not possible
- **Command blade collective pitch $\theta_{.75}$**
  - **calculate thrust from collective pitch angle (BE theory)**

Cyclic control: tip-path plane command or no-feathering plane command

- **Tip-path plane (TPP) command**
  - **Cyclic control tilts TPP (command cyclic flapping β_c, β_s) hence tilts thrust vector**
    - appropriate for main rotors
    - solve for blade cyclic pitch angles from flapping (BE theory)
    - flap angles or hub moment or lift offset
- **No-feathering plane (NFP) command**
  - **Cyclic control tilts swashplate (command blade cyclic pitch θ_c, θ_s)**
    - required for rotors without swashplate (commanded cyclic is zero)
    - solve for flapping from blade cyclic pitch angle (BE theory)

Rotor Forces

Rotor hub loads

$$F = \begin{pmatrix} H \\ Y \\ T \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -f_B T \end{pmatrix}$$

shaft axis thrust
T = control
(f_B for blockage)

$$M = \begin{pmatrix} M_x \\ M_y \\ -Q \end{pmatrix} = \begin{pmatrix} K_{\text{hub}}\beta_s \\ -K_{\text{hub}}\beta_c \\ -P_{\text{shaft}}/\Omega \end{pmatrix}$$

hub moment proportional to TPP tilt (control)

torque from rotor shaft power

Inplane forces (H and Y) = tilt T with TPP + profile + force relative TPP

profile force: $C_{Ho} = \dfrac{\sigma}{8} c_{d\,\text{mean}} F_H(\mu, \mu_z)$

Inplane forces relative TPP calculated with blade element theory

- **No stall, inflow and drag from power model, only flap degree of freedom**
- **With TPP command and neglect inplane forces relative TPP, collective and cyclic pitch angles not required**
- **But rotor force not perpendicular to TPP in general**

Rotor Power

Power required: induced + profile + parasite

$$P = P_i + P_o + P_p$$

Parasite power = propulsive force x **flight speed (**$P_p = -XV$**)**

Induced power factor κ **and mean drag coefficient** $c_{d\,\mathrm{mean}}$

$$P_i = \kappa P_{\mathrm{ideal}} = \kappa T v_{\mathrm{ideal}}$$

$$P_o = \rho A(\Omega R)^3 C_{Po} = \rho A(\Omega R)^3 (\sigma/8) c_{d\,\mathrm{mean}} F_P(\mu, \mu_z)$$

Models account for influence of speed, thrust, compressibility, stall, lift offset, and induced interference between twin rotors

Spreadsheet for development of model from higher-fidelity calculations

- rotor_perf_template.xls

Calibration of model reflects level of technology

Rotor Induced Power

hover and propeller:

$$\kappa_h = \kappa_{hover} + \text{quadratic}(C_T/\sigma)$$
$$\kappa_p = \kappa_{prop} + \text{quadratic}(C_T/\sigma) + f(\mu)$$

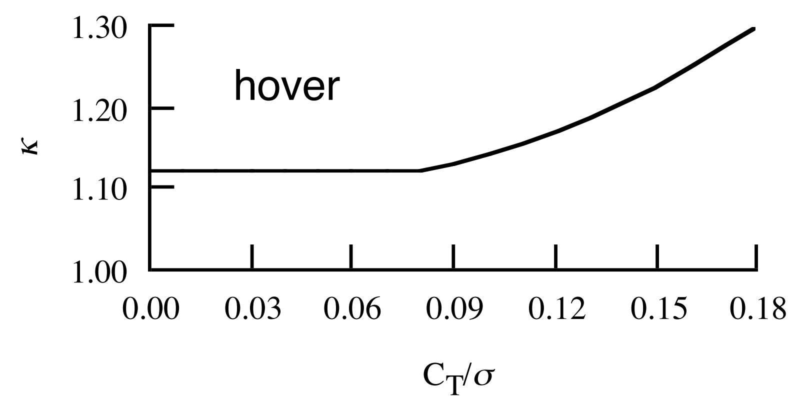
axial flight: scaled so

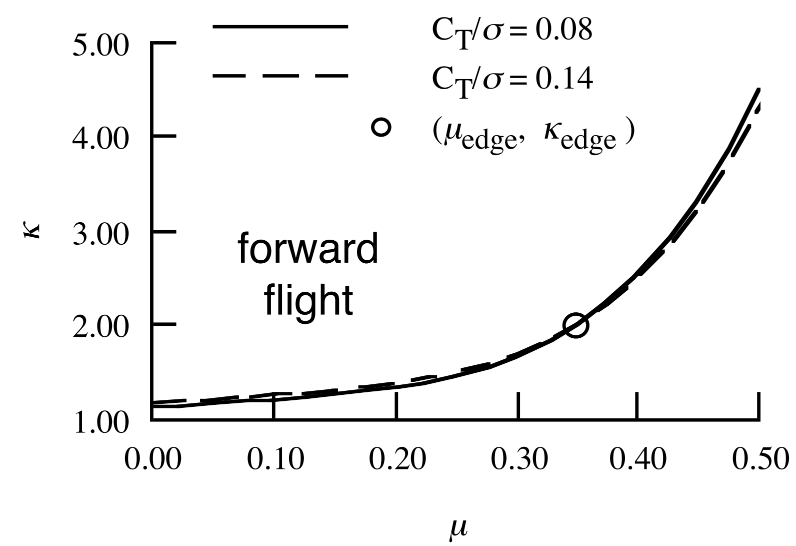
$$\kappa = \kappa_h \text{ at } \mu_z = 0,\ \kappa = \kappa_p \text{ at } \mu_z = \mu_{z\,prop}$$
$$\kappa_{axial} = \kappa_h + \text{polynomial}(\mu_z)$$

edgewise flight: scaled so

$$\kappa = \kappa_{axial} \text{ at } \mu = 0,\ \kappa = f_{off}\,\kappa_{edge} \text{ at } \mu = \mu_{edge}$$
$$\kappa = \kappa_{axial} + \text{polynomial}(\mu)$$

f_{off} = influence of lift offset





hover and axial flight

thrust variation through $\Delta = C_T/\sigma - (C_T/\sigma)_{\text{ind}}$

$$\kappa_h = \kappa_{\text{hover}} + k_{h1}\Delta_h + k_{h2}|\Delta_h|^{X_{h2}}$$

$$\kappa_p = \kappa_{\text{prop}} + k_{p1}\Delta_p + k_{p2}|\Delta_p|^{X_{p2}} + k_{p\alpha}|\mu|^{X_{p\alpha}}$$

polynomial for variation with axial velocity

scaled so $\kappa = \kappa_h$ at $\mu_z = 0$ and $\kappa = \kappa_p$ at $\mu_z = \mu_{z\,\text{prop}}$

$$\kappa_{\text{axial}} = \kappa_h + k_{a1}\mu_z + S_a(k_{a2}\mu_z^2 + k_{a3}\mu_z^{X_a})$$

polynomial for variation with edgewise velocity

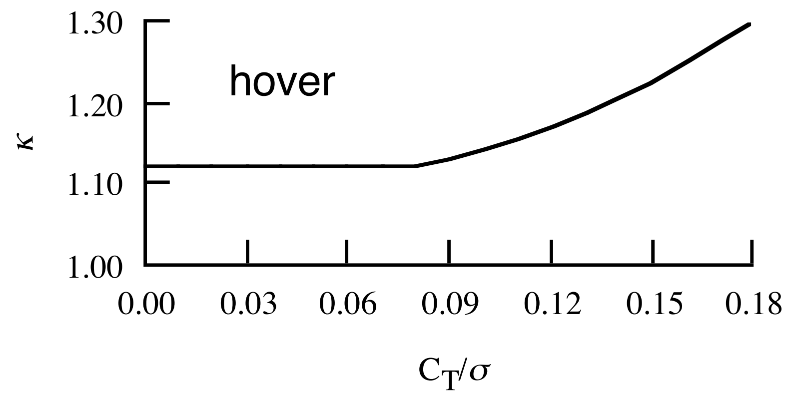
scaled so $\kappa = f_{\text{off}}\kappa_{\text{edge}}$ at $\mu = \mu_{\text{edge}}$

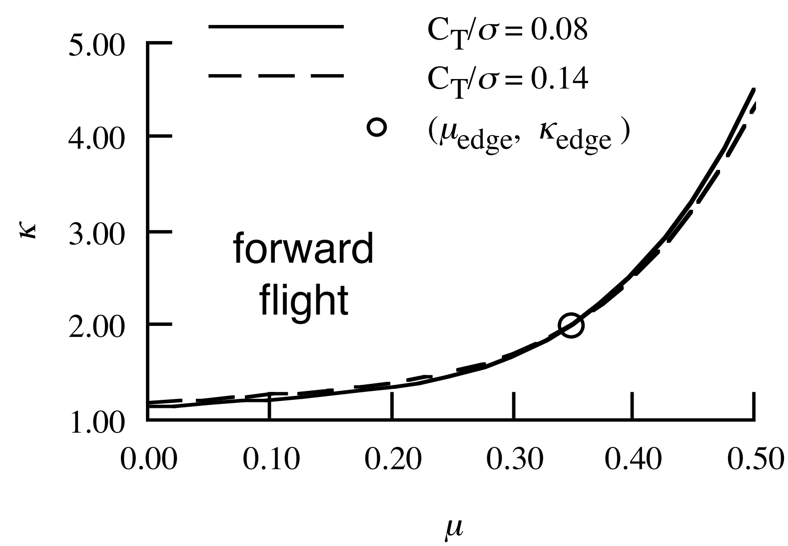
$$\kappa = \kappa_{\text{axial}} + k_{e1}\mu + S_e(k_{e2}\mu^2 + k_{e3}\mu^{X_e})$$

influence of lift offset:

$$f_{\text{off}} = 1 - k_{o1}(1 - e^{-k_{o2}o_x})$$

$$o_x = rM_x/TR = (K_{\text{hub}}/TR)\beta_s$$





Rotor Profile Power

drag = basic + stall + compressibility

$$c_{d\ \text{mean}} = \chi S(c_{d\ \text{basic}} + c_{d\ \text{stall}} + c_{d\ \text{comp}})$$

χ = technology factor

***S* accounts for Reynolds number**

basic drag $c_{d\ \text{basic}}$:

quadratic function C_T/σ plus faster growth at high (sub-stall) loading

hover and propeller:

$c_{dh} = d_{0hel} + q(C_T/\sigma)$ + separation

$c_{dp} = d_{0prop} + q(C_T/\sigma)$ + separation + $f(\mu)$

interpolated with μ_z:

$$c_{d\ \text{basic}} = c_{dh} + (c_{dp} - c_{dh})\frac{2}{\pi}\tan^{-1}\left(|\mu_z|/\lambda_h\right)$$

stall drag $c_{d\ \text{stall}}$:

occurrence of significant stall

$$\Delta_s = |C_T/\sigma| - (f_s/f_{\text{off}})(C_T/\sigma)_s$$

$$c_{d\ \text{stall}} = d_{s1}\Delta_s^{X_{s1}} + d_{s2}\Delta_s^{X_{s2}}$$

stall loading $(C_T/\sigma)_s$ function of

$V = (\mu^2 + \mu_z^2)^{1/2}$

f_{off} = influence of lift offset

compressibility drag $c_{d\ \text{comp}}$:

from advancing tip Mach number and drag divergence Mach number

$$\Delta M = M_{at} - M_{dd}$$

$$c_{d\ \text{comp}} = d_{m1}\Delta M + d_{m2}\Delta M^{X_m}$$

Rotor Profile Power

drag = basic + stall + compressibility

$$c_{d\,\text{mean}} = \chi S(c_{d\,\text{basic}} + c_{d\,\text{stall}} + c_{d\,\text{comp}})$$

χ is technology factor
$S = (Re_{\text{ref}} / Re)^{0.2}$ accounts for Reynolds number effects

basic drag $c_{d\,\text{basic}}$ = quadratic function of C_T/σ
plus faster growth at high (sub-stall) loading

$$\Delta = |C_T/\sigma - (C_T/\sigma)_{D\,\text{min}}|$$

$$\Delta_{\text{sep}} = |C_T/\sigma| - (C_T/\sigma)_{\text{sep}}$$

for helicopter and propeller operation

$$c_{dh} = d_{0\,\text{hel}} + d_{1\,\text{hel}}\Delta + d_{2\,\text{hel}}\Delta^2 + d_{\text{sep}}\Delta_{\text{sep}}^{X_{\text{sep}}}$$

$$c_{dp} = d_{0\,\text{prop}} + d_{1\,\text{prop}}\Delta + d_{2\,\text{prop}}\Delta^2 + d_{\text{sep}}\Delta_{\text{sep}}^{X_{\text{sep}}} + d_{p\alpha}|\mu|^{X_{p\alpha}}$$

interpolated with axial velocity μ_z

$$c_{d\,\text{basic}} = c_{dh} + (c_{dp} - c_{dh})\tfrac{2}{\pi}\tan^{-1}\left(|\mu_z|/\lambda_h\right)$$

stall drag $c_{d\,\text{stall}}$ from occurrence of significant stall

$$\Delta_s = |C_T/\sigma| - (f_s/f_{\text{off}})(C_T/\sigma)_s$$

$$c_{d\,\text{stall}} = d_{s1}\Delta_s^{X_{s1}} + d_{s2}\Delta_s^{X_{s2}}$$

f_s is input factor
stall loading $(C_T/\sigma)_s$ is function of $V = (\mu^2 + \mu_z^2)^{1/2}$

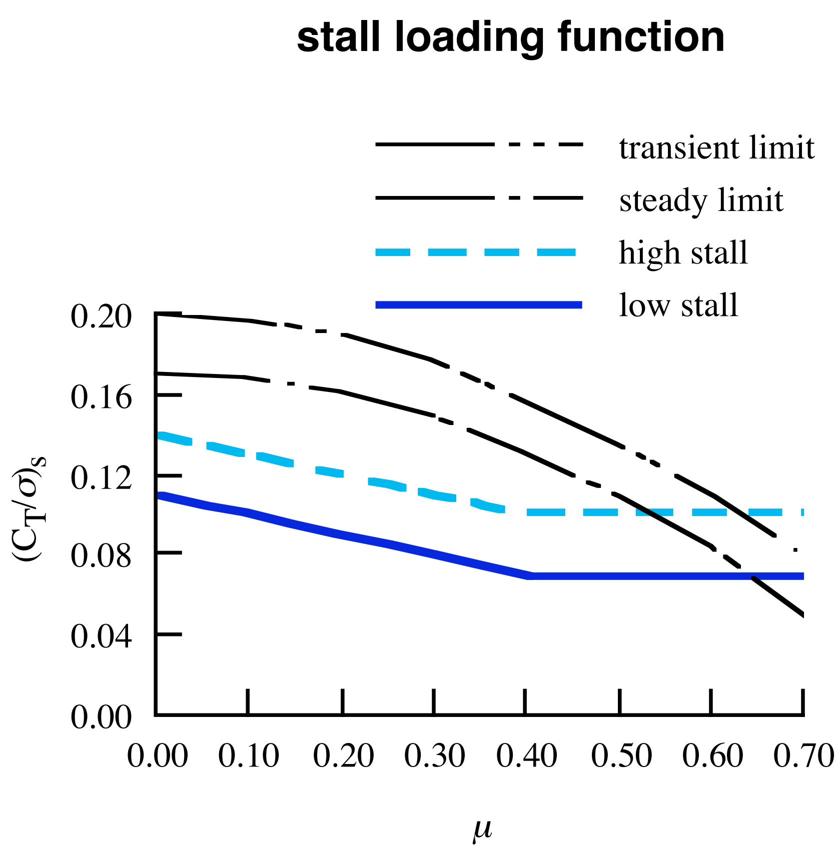
influence of lift offset:

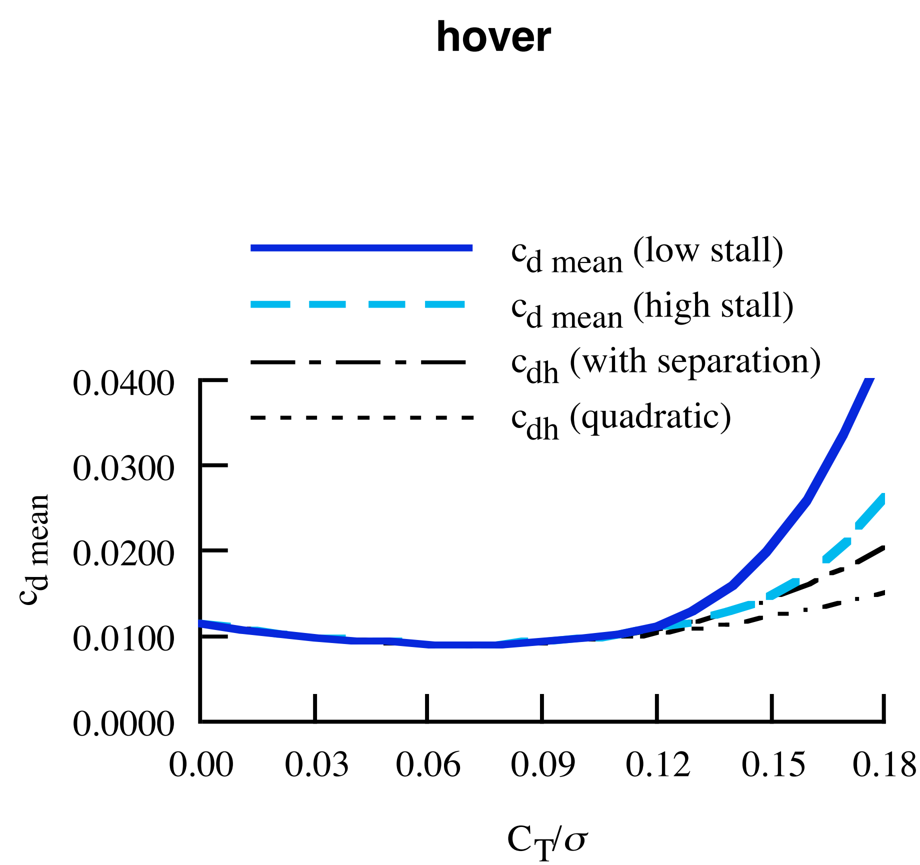
$$f_{\text{off}} = 1 - d_{o1}(1 - e^{-d_{o2}o_x})$$
$$o_x = rM_x/TR = (K_{\text{hub}}/TR)\beta_s$$

compressibility drag $c_{d\,\text{comp}}$ depends on advancing tip Mach number and drag divergence Mach number

$$\Delta M = M_{at} - M_{dd}$$
$$c_{d\,\text{comp}} = d_{m1}\Delta M + d_{m2}\Delta M^{X_m}$$

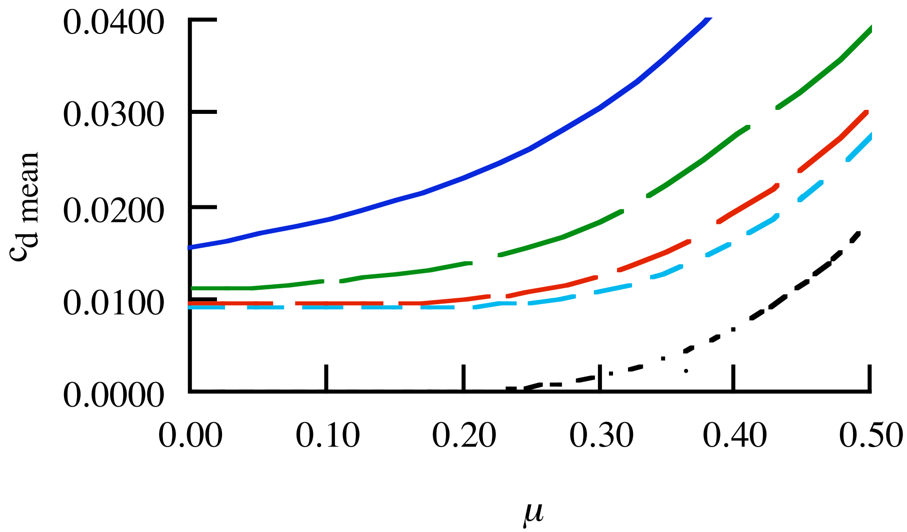
Rotor Profile Power



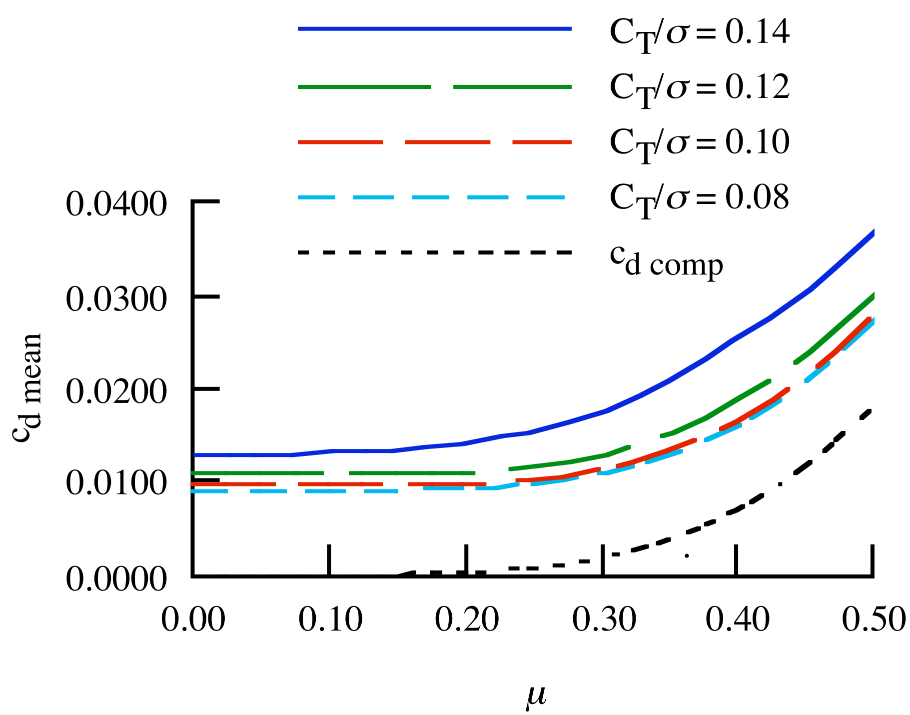


Rotor Profile Power

low stall function



high stall function



Rotor Interference

Rotor aerodynamic interference at other components: fuselage, wings, tails

Wake-induced velocity at component estimated based on inflow at rotor

- **Induced velocity at the rotor disk known, acting in direction opposite thrust**
- **Interference velocity proportional to induced velocity, in the same direction**

$$v_{\text{int}}^{F} = K_{\text{int}} f_W f_z f_r f_t v_{\text{ind}}^{F}$$

$f_W f_z$ **accounts for axial development of wake velocity**

- **–step function, or nominal rate of change, or input rate of change**

f_r **accounts for immersion in wake**

- **–contracted or uncontracted wake radius**
- **–step function, or always immersed, or input transition distance**

f_t **accounts for twin rotors**

K_{int} **is input empirical factor**

- **–can be reduced to zero at high speed**

Rotor Interference

To account for extent of wing or tail area immersed in wake, interference velocity calculated at several points along span and averaged

Need rotor interference on fuselage and wing for hover download

May need rotor-wing and wing-rotor interference for cruise performance

Often turn off rotor interference above 10-20 knots

Wing

Geometry defined in terms of wing panels

- Symmetric
- Each panel has straight aerodynamic center and linear taper
  - Sweep, dihedral, offsets of aerodynamic center
- Set of outboard panels can be considered wing extension

Controls: flap, flaperon, aileron, incidence

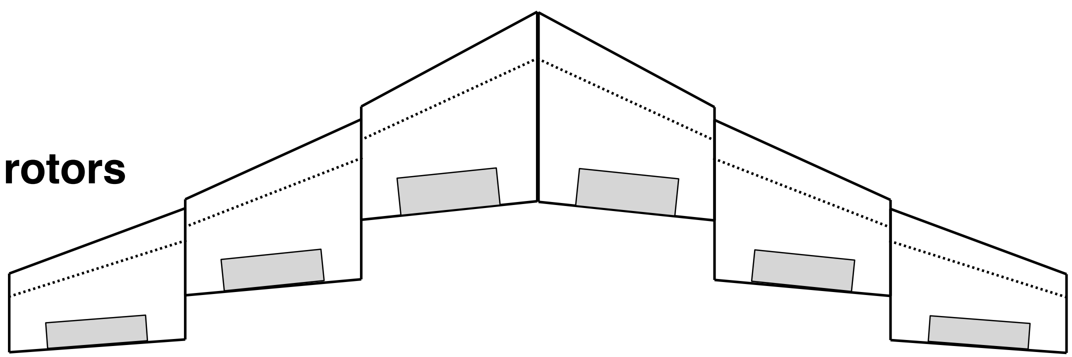
- Controls for each panel
- Flaperon and aileron are same surface

Wing interference on other wings (biplane or tandem)

Wing interference on tail

Wing interference on rotors

Induced-drag interference from rotors



Propulsion

Propulsion Group

- Set of rotors and engine groups, connected by drive system
  - One or more drive states, with different gear ratios
  - Tip speed: input, reference, function speed or conversion schedule, or various defaults
- Power required = component power + transmission losses + accessory losses
- Drive system limit (torque), rotor and engine shaft limits
- Drive system weight

Engine Group

- Each engine group has one or more engines of same type
- Performance at required power: mass flow, fuel flow, jet thrust, momentum drag
- Controls: yaw, incidence
- Drag, weight

Referred Parameter Turboshaft Engine Model

- Enables aircraft performance analysis to cover entire spectrum of operation
  - Curve fits of referred performance from engine deck, including effect of turbine speed
- Effects of size (scaling model, based on mass flow) and technology (specific power and specific fuel consumption)

Propulsion

Propulsion Groups

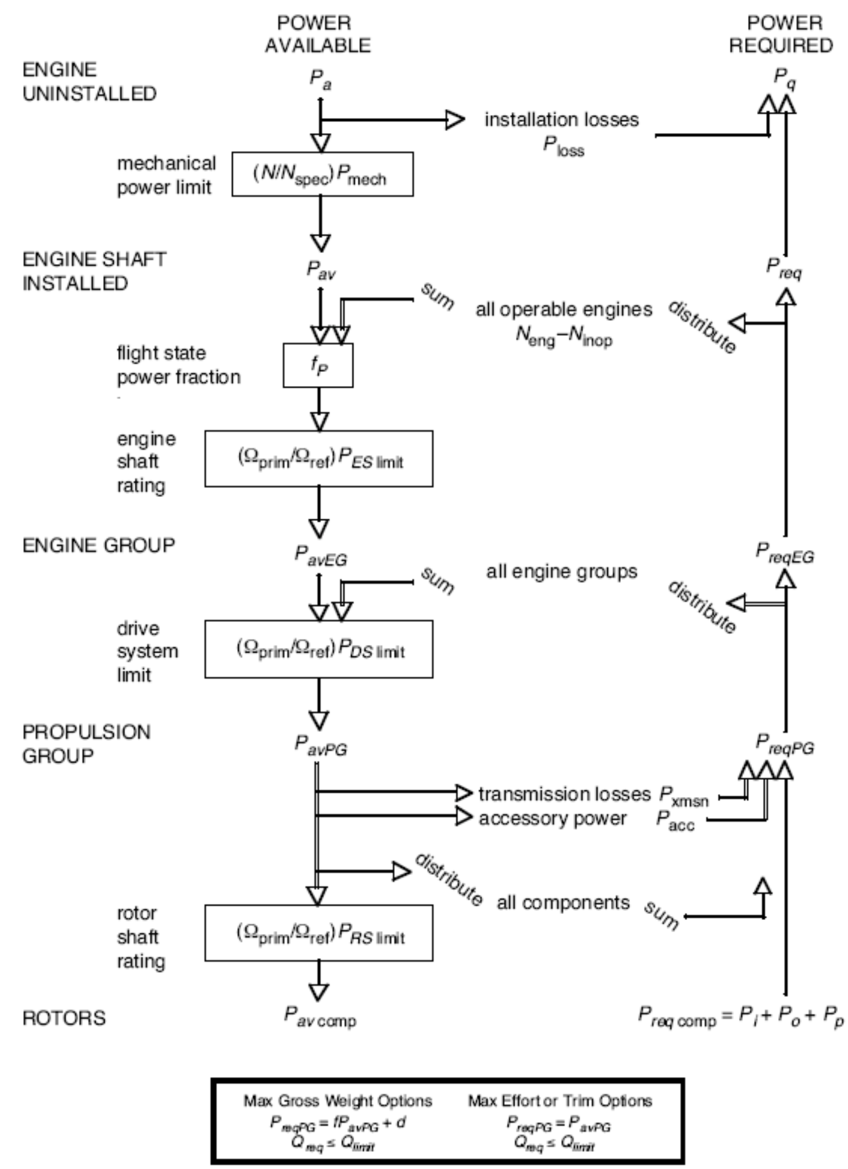
- Set of rotors and engine groups, connected by drive system
  - One or more drive states, with different gear ratios
- Power required = $P_{comp}+P_{xmsn}+P_{acc}$
- Drive system limit (torque), rotor and engine shaft limits

Engine Group

- Each engine group has one or more engines of same type
- Performance: mass flow, fuel flow, jet thrust, momentum drag
- Controls: yaw, incidence

Referred Parameter Turboshaft Engine Model

- Enables aircraft performance analysis to cover entire spectrum of operation
  - Curve fits of referred performance from engine deck, including effect of turbine speed
- Effects of size (scaling model) and technology (specific power and sfc)



Original Propulsion Representation

Mechanical drive train, connecting engine groups and rotors

Engine group, consisting of one or more turboshaft engines

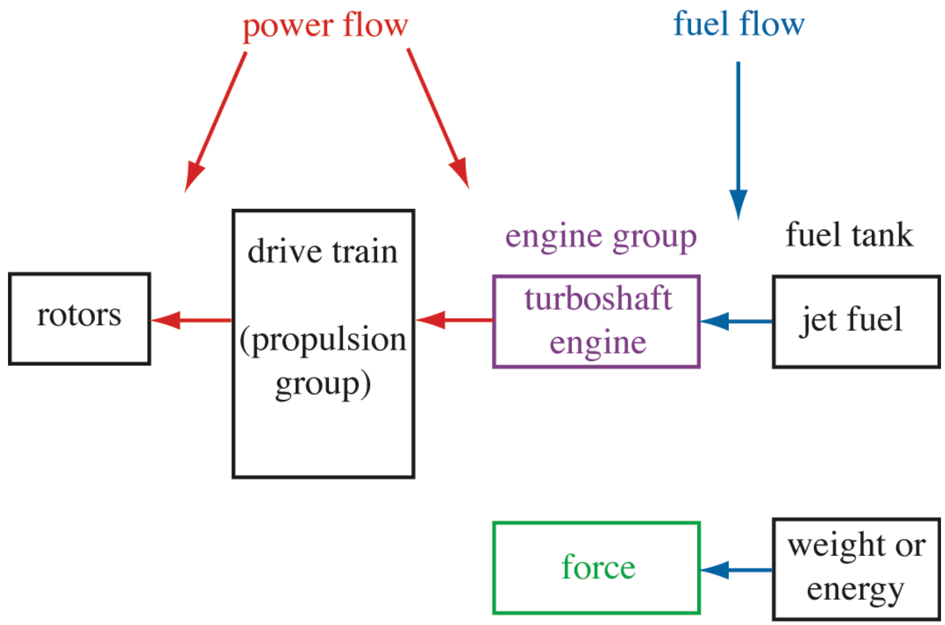
- **Referred Parameter Turboshaft Engine Model**

Fuel tank system (main and aux tanks)

- **Weight changes as fuel used, fuel is measured in weight**

Force generation by simple model

- **Fuel used is measured as weight or energy**



Extended Propulsion Representation

Engine Group
Turboshaft engine
Reciprocating engine
Compressor
Motor
Generator
Generator-Motor

Transfers power by shaft torque

Connected to drive train

Propulsion group includes rotors

Jet Group
Turbojet / turbofan
Reaction drive
Simple force

Produces force on aircraft

Charge Group
Fuel cell
Solar cell

Generates energy for aircraft

Fuel Tank
Weight
- jet fuel
- gasoline
- diesel
- hydrogen

Energy
- battery
- flywheel
- capacitor

Associated with components that use fuel

Extended Propulsion Representation

Engine Group

Turboshaft engine
- convertible — jet
- convertible — reaction

Reciprocating engine
Compressor
Compressor-reaction
Motor
Motor + fuel cell
Generator
Generator-Motor

Transfers power by shaft torque

Connected to drive train

Propulsion group includes rotors

Jet Group

Turbojet / turbofan
- convertible — reaction

Reaction drive
Simple force

Produces force on aircraft

Charge Group

Fuel cell
Solar cell

Generates energy for aircraft

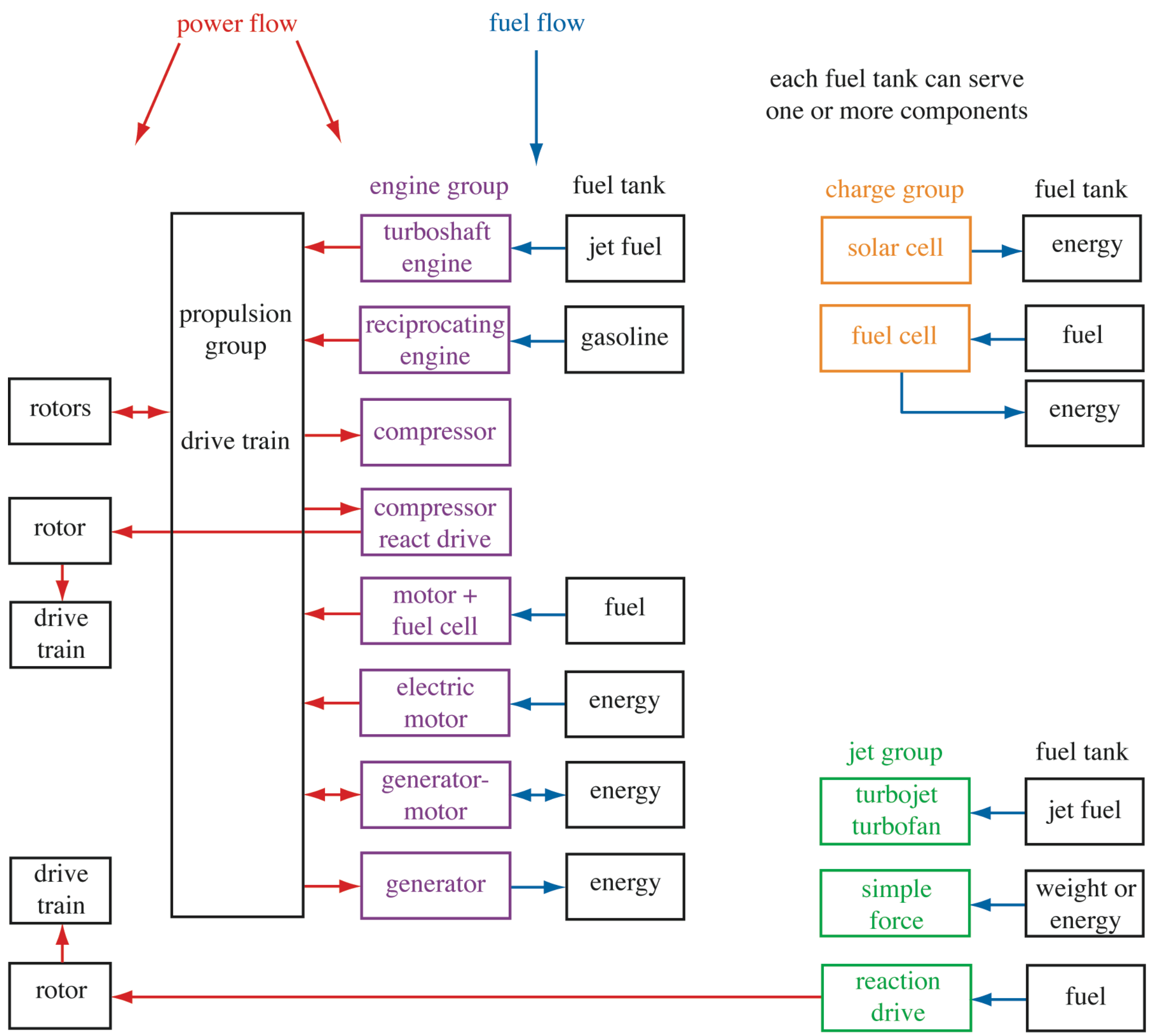
Fuel Tank

Weight
- jet fuel
- gasoline
- diesel
- hydrogen

Energy
- battery
- flywheel
- capacitor

Associated with components that use fuel

Extended Propulsion Representation



Propulsion Component Models

Engine group, jet group, and charge group provide general framework for theory and code

Performance and weight evaluated using engine, jet, or charger model

- Typically need power or thrust available, mass flow, fuel flow, jet force
- Functions of independent parameters (power or thrust, atmosphere, speed, etc.)

Need parameterized, surrogate representation of component performance and weight

- Applicable to wide range of operating conditions and component size

Component Models Implemented

Engine group models

- Referred Parameter Turboshaft Engine Model (RPTEM)
- compressor
- motor/generator

Jet group models

- Referred Parameter Jet Engine Model (RPJEM)
- simple force

Charge group models

- fuel cell
- solar cell

Battery model for fuel tanks

Referred Parameter Turboshaft Engine Model

Power available

$$\text{power} \quad \frac{P_a}{\delta\sqrt{\theta}} = P_0 g_p(\theta, M, n)$$

Performance at power required

$$\text{mass flow} \quad \frac{\dot{m}_{req}}{\delta/\sqrt{\theta}} = \dot{m}_{0C}\, g_m(q, \theta, M, n)$$

$$\text{fuel flow} \quad \frac{\dot{w}_{req}}{\delta\sqrt{\theta}} = \dot{w}_{0C}\, g_w(q, \theta, M, n)$$

$$\text{gross thrust} \quad \frac{F_g}{\delta} = F_{g0C}\, g_f(q, \theta, M, n)$$

Scale with pressure ($\delta = p/p_0$) and temperature ($\theta = T/T_0$)

Functions of power required ($q = P_q/(P_{0C}\delta\sqrt{\theta})$), temperature ratio, Mach number, turbine speed ($n = N/\sqrt{\theta}$)

Referred Parameter Turboshaft Engine Model

AMRDEC

Power available

power $$\frac{P_a}{\delta\sqrt{\theta}} = P_0 g_p(\theta, M, n)$$

g_p from constants that are piecewise linear functions of θ

Performance at power required

mass flow $$\frac{\dot{m}_{req}}{\delta/\sqrt{\theta}} = \dot{m}_{0C}\, g_m(q, \theta, M, n)$$

fuel flow $$\frac{\dot{w}_{req}}{\delta\sqrt{\theta}} = \dot{w}_{0C}\, g_w(q, \theta, M, n)$$

gross thrust $$\frac{F_g}{\delta} = F_{g0C}\, g_f(q, \theta, M, n)$$

g_m, g_w, g_f proportional to cubic polynomials in q

Good representation for design code NDARC

Scale with pressure ($\delta = p/p_0$) and temperature ($\theta = T/T_0$)

Functions of power required ($q = P_q/(P_{0C}\delta\sqrt{\theta})$), temperature ratio, Mach number, turbine speed ($n = N/\sqrt{\theta}$)

Fuel Tank Systems

Each system consists of main tank(s) and auxiliary tank(s)

- Engines, jets, chargers associated with a fuel tank system
- Fuel container has weight
- Fuel quantity stored and burned is measured in weight or energy

Weight changes as fuel used

- Jet fuel, gasoline, diesel, hydrogen
- Characteristics: density (lb/gal or kg/liter), specific energy (MJ/kg), tank weight

Energy changes as fuel used, weight does not change

- Battery, flywheel, capacitor
- Characteristics: tank density (MJ/liter), tank specific energy (MJ/kg)

Battery model

- Characteristics: efficiency (varies with power, state-of-charge), power density (kW/kg)

Fuel Properties

fuel	specification	density		specific energy			energy density
		lb/gal	kg/L	MJ/kg	BTU/lb	lb/hp-hr	MJ/L
gasoline	MIL-STD-3013A	6.0*	0.719	43.50	18700*	0.136	31.3
diesel	nominal	7.0	0.839	43.03	18500	0.138	36.1
	range	6.84–7.05	0.820–0.845	43.0	18487	0.138	35.8
JetA/A-1	MIL-STD-3013A	6.7*	0.803	42.80	18400*	0.138	34.4
	range	6.84/6.71	0.820/0.804	42.8	18401	0.138	34.8
JP-4	nominal	6.5	0.779	42.80	18400	0.138	33.3
	MIL-DTL-5624U	6.23–6.69	0.751*–0.802*	42.8*	18401	0.138	32.2
JP-5	MIL-STD-3013A	6.6*	0.791	42.57	18300*	0.139	33.7
	alternate design	6.8*	0.815	42.91	18450*	0.138	35.0
	MIL-DTL-5624U	6.58–7.05	0.788*–0.845*	42.6*	18315	0.139	34.8
JP-8	MIL-STD-3013A	6.5*	0.779	42.80	18400*	0.138	33.3
	alternate design	6.8*	0.815	43.19	18570*	0.137	35.2
	MIL-DTL-83133H	6.45–7.01	0.775*–0.840*	42.8*	18401	0.138	34.6
hydrogen (700 bar)		0.328	0.03930	120.	51591	0.0493	4.72
hydrogen (liquid)		0.592	0.07099	120.	51591	0.0493	8.52

*specification value

Energy Storage Properties

		tank specific energy MJ/kg	kW-hr/kg	tank energy density MJ/L	kW-hr/m^3	efficiency	power kW/kg
lead-acid battery		0.11–0.14	0.03–0.04	0.22–0.27	60–75	70–90%	0.18
nickel-cadmium battery		0.14–0.20	0.04–0.06	0.18–0.54	50–150	70–90%	0.15
lithium-ion	state-of-art	0.54–0.90	0.15–0.25	0.90–1.30	250–360	~99%	1.80
	+5 years	1.26	0.35	1.80	500		
	+10 years	2.34	0.65	2.25	625		
ultracapacitor		0.01–0.11	0.004-0.03	0.02–0.16	6–45		1.00
flywheel	steel	0.11	0.03			~90%	
	graphite	0.90	0.25				

Cost

CTM (Harris and Scully) Rotorcraft Cost Model

- Aircraft purchase price
- Maintenance cost
- Direct operating cost

Inflation factors

- DoD: deflators for Total Obligational Authority and Procurement
- Consumer price index: all urban consumers, U.S city average, all items

Weights

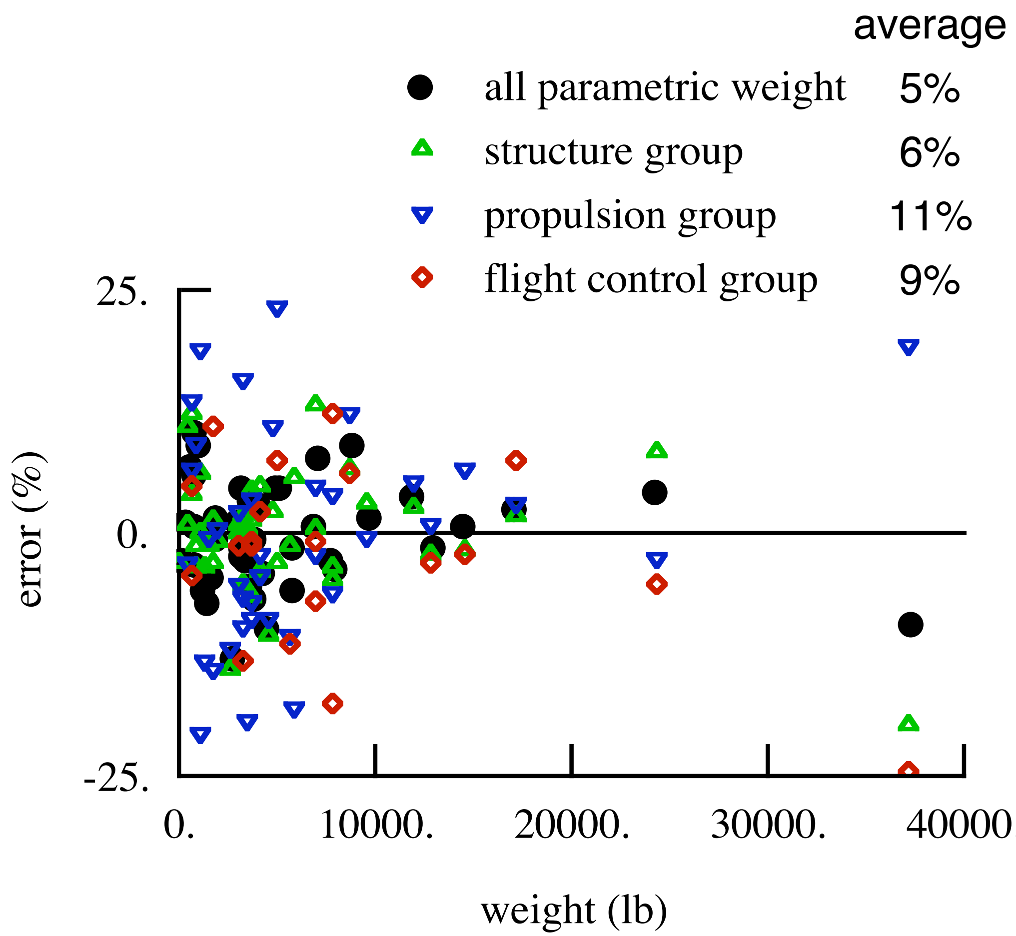
Parametric equations based on weight of existing turbine powered helicopters and tiltrotors (and some fixed wing aircraft component weights)

- **Parametric equation average error:**
  - **rotor blades and hub 9%**
  - **fuselage 7%**
  - **wing 3%, tail 23%**
  - **drive system 8%**
  - **flight controls 9%**

Include weight increments and calibration/technology factors

Weight breakdown based on extended SAWE RP8A weight statement

accuracy of sum of all parametric weights (42 aircraft, 15-75% of empty weight)



Weights

Component weights:

parametric weight model, with technology factor χ

plus input increment

$$W = \chi W_{\text{model}} + dW$$

or fixed (input) value (*dW*)

typical options: *WEIGHT_zzzz* = 0 input, 1 AFDD, 2 custom

Technology factor *TECH_zzzz* = (calibration) * (technology)

Baseline technology factor values from calibration to existing aircraft

- Match parametric equation (for variation with size) to weight of most-similar design
- Spreadsheets to develop factors (weight_eq_helicopter.xls, weight_eq_tiltrotor.xls)

Weight Definitions

Design gross weight (typically from sizing task)

Structural design gross weight, maximum takeoff weight (influence weight estimates)

gross weight	$W_G = W_E + W_{UL} = W_O + W_{\text{pay}} + W_{\text{fuel}}$
operating weight	$W_O = W_E + W_{FUL}$
useful load	$W_{UL} = W_{FUL} + W_{\text{pay}} + W_{\text{fuel}}$

$W_E, W_{FUL}, W_{\text{pay}}, W_{\text{fuel}}$ = weight empty, fixed useful load, payload, fuel

Payload, operating weight, empty weight definitions from SAWE RP7D

For NDARC, weight empty is a parameter of aircraft (not of flight state)

- **Operating weight variations with flight state (flight condition or mission segment) accounted for in fixed useful load**

Weight information follows SAWE RP8A Group Weight Statement format

- **With extensions that reflect parametric weight estimation**

Weight Definitions from SAWE RP7D

Payload

- Payload is any item which is being transported and is directly related to the purpose of the flight as opposed to items that are necessary for the flight operation. Payload can include, but is not limited to, passengers, cargo, passenger baggage, ammo, internal and external stores, and fuel which is to be delivered to another aircraft or site. Payload may or may not be expended in flight.

Operating Weight

- Operating weight is the sum of aircraft weight empty and operating items. Operating weight is equivalent to takeoff gross weight less usable fuel, payload, and any item to be expended in flight.

Weight Empty

- Weight empty is an engineering term which is defined as the weight of the complete aircraft as defined in the aircraft specifications, dry, clean, and empty except for fluids in closed systems such as a hydraulic system.

Weight Statement

WEIGHT EMPTY
- STRUCTURE
  - wing group
    - basic structure
    - secondary structure
      - fairings (*), fittings (*), fold/tilt (*)
    - control surfaces
  - rotor group
    - blade assembly
    - hub & hinge
      - basic (*), fairing/spinner (*), blade fold (*), shaft (*)
    - rotor support structure (*), duct (*)
  - empennage group
    - horizontal tail (*)
      - basic (*), fold (*)
    - vertical tail (*)
      - basic (*), fold (*)
    - tail rotor (*)
      - blades, hub & hinge, rotor supports, rotor/fan duct
  - fuselage group
    - basic (*)
    - wing & rotor fold/retraction (*)
    - tail fold/tilt (*)
    - marinization (*)
    - pressurization (*)
    - crashworthiness (*)
  - alighting gear group
    - basic (*), retraction (*), crashworthiness (*)
  - engine section or nacelle group
    - engine support (*), engine cowling (*), pylon support (*)
  - air induction group
- PROPULSION GROUP
  - engine system
    - engine
    - exhaust system
    - accessories (*)
  - propeller/fan installation
    - blades (*), hub & hinge (*), rotor supports (*), rotor/fan duct (*)
  - fuel system
    - tanks and support
    - plumbing
  - drive system
    - gear boxes
    - transmission drive
    - rotor shaft
    - rotor brake (*)
    - clutch (*)
    - gas drive

* = RP8A extension, to accommodate weight equations

Weight Statement

SYSTEMS AND EQUIPMENT
- flight controls group
  - cockpit controls
  - automatic flight control system
  - system controls
    - fixed wing systems
      - non-boosted (*), boost mechanisms (*)
    - rotary wing systems
      - non-boosted (*), boost mechanisms (*), booste(
    - conversion systems
      - non-boosted (*), boost mechanisms (*)
- auxiliary power group
- instruments group
- hydraulic group
  - fixed wing (*), rotary wing (*), conversion (*)
  - equipment (*)
- pneumatic group
- electrical group
  - aircraft (*), anti-icing (*)
- avionics group (mission equipment)
- armament group
  - armament provisions (*), armor (*)
- furnishings & equipment group
- environmental control group
- anti-icing group
- load & handling group

VIBRATION (*)
CONTINGENCY

FIXED USEFUL LOAD
- crew
- fluids (oil, unusable fuel) (*)
- auxiliary fuel tanks
- other fixed useful load (*)
- equipment increment (*)
- folding kit (*)
- wing extension kit (*)
- wing kit (*)
- other kit (*)

PAYLOAD

USABLE FUEL
- standard tanks (*)
- auxiliary tanks (*)

OPERATING WEIGHT = weight empty + fixed useful load
USEFUL LOAD = fixed useful load + payload + usable fuel

GROSS WEIGHT = weight empty + useful load
GROSS WEIGHT = operating weight + payload + usable fuel

* = RP8A extension, to accommodate weight equations

Military Load (AFDD Definitions)

Aircraft operating weight can be divided into core vehicle weight and military load

- **Core vehicle weight:** weight in minimum airworthy state, with aircraft capable of normal flight throughout envelope, but not mission capable
- **Military load:** sum of fixed useful load and military features in weight empty

Thus

weight empty = core vehicle weight + military features

military load = fixed useful load + military features in weight empty

operating weight = $W_E + W_{FUL}$ = core vehicle weight + military load

In terms of NDARC weight breakdown, military features in weight empty includes:

- **folding weight (wing, rotor, tail, fuselage terms)**
- **crashworthiness weight (fuselage, landing gear terms)**
- **marinization weight (fuselage)**
- **rotor brake (drive system)**
- **avionics group (mission equipment)**
- **armament group**
- **furnishings and equipment group**
- **anti-icing group (including electrical group term)**
- **load and handling group**

Systems Component

Weight information

- **Vibration**
  - –Input, or fraction of weight empty

- **Contingency**
  - –Input, or fraction of weight empty

- **Systems and equipment group**
  - –By group, or details

- **Fixed useful load**

Details of Weight Description (RP8A)

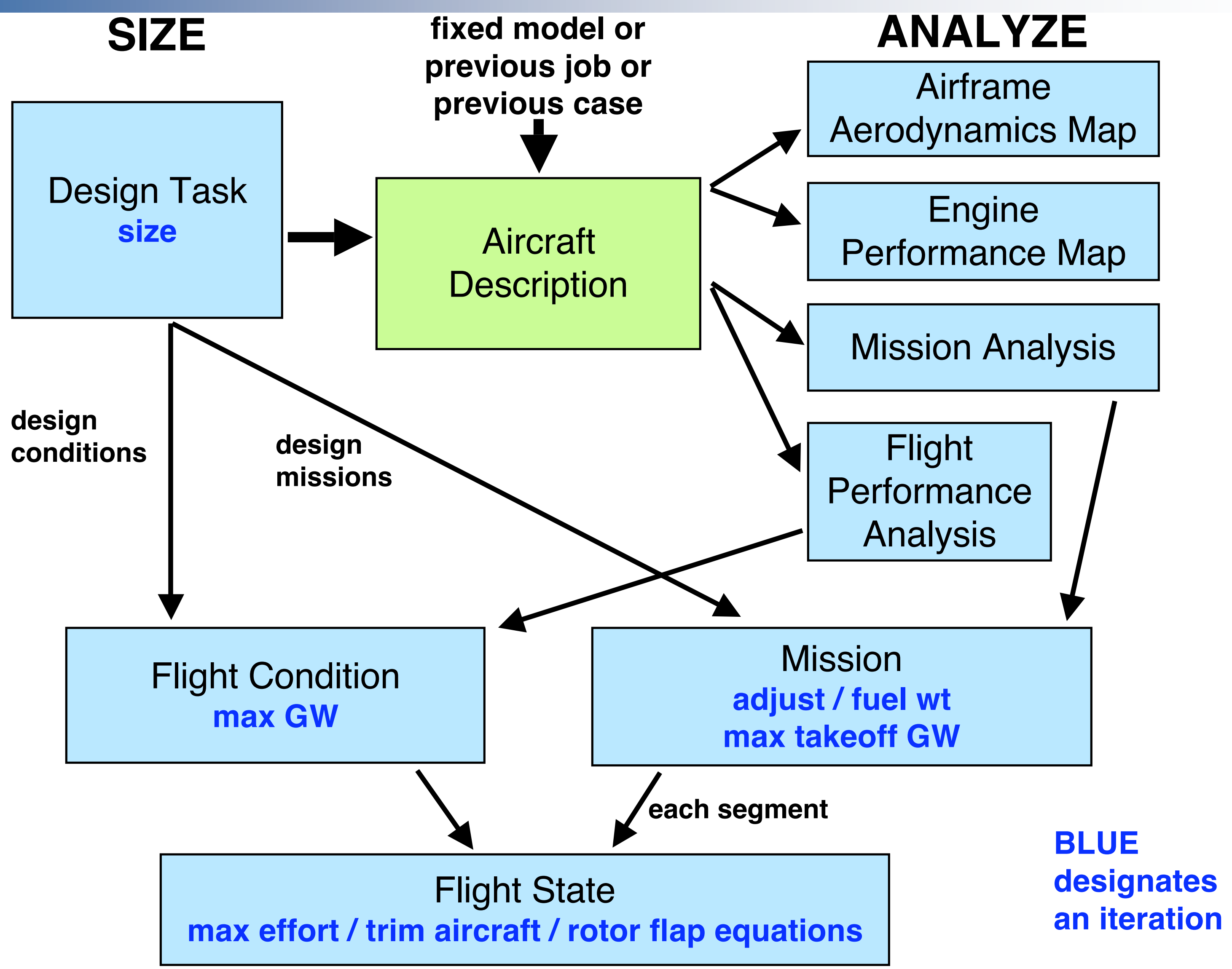
WEIGHT EMPTY
- SYSTEMS AND EQUIPMENT
  - electrical group
    - aircraft
      - power supply
      - power conversion
      - power distribution and controls
      - lights and signal devices
      - equipment supports
    - anti-icing
  - avionics group (mission equipment)
    - equipment
    - installation
  - armament group
    - armament provisions
    - armor
  - furnishings & equipment group
    - accommodation for personnel
      - seats
      - miscellaneous accommodation
      - oxygen system
    - miscellaneous equipment
    - furnishings
    - emergency equipment
      - fire detection and extinguishing
      - other emergency equipment
  - load & handling group
    - aircraft handling
    - load handling

USEFUL LOAD
- FIXED USEFUL LOAD
  - crew
  - other fixed useful load
    - various categories
  - equipment increment
- PAYLOAD
  - passengers/troops
  - cargo
  - ammunition
  - weapons

Outline

- Introduction
- Documentation
- Overview
  - Tasks
  - Aircraft
- NDARC Job
  - Input
  - Organization
  - Output
- Solution Procedures
  - Debugging
- Input Manual
  - Aircraft
  - Tasks
- Tutorial

NDARC Tasks



SIZE
fixed model or previous job or previous case
ANALYZE
Design Task
size
Aircraft Description
Airframe Aerodynamics Map
Engine Performance Map
Mission Analysis
Flight Performance Analysis
design conditions
design missions
Flight Condition
max GW
Mission
adjust / fuel wt
max takeoff GW
each segment
Flight State
max effort / trim aircraft / rotor flap equations
BLUE designates an iteration

Solution Procedure

Sizing Task

Size Iteration
method: successive substitution
- Missions
- Flight Conditions

↓

Mission Analysis

- Missions

↓

Flight Perf Analysis

- Flight Conditions

Flight Condition

Maximum GW
method: secant or false position
- Flight State

Mission

Mission Iteration
adjust, fuel weight
method: successive substitution

Segments

Maximum GW
method: secant or false position
- Flight State

Flight State

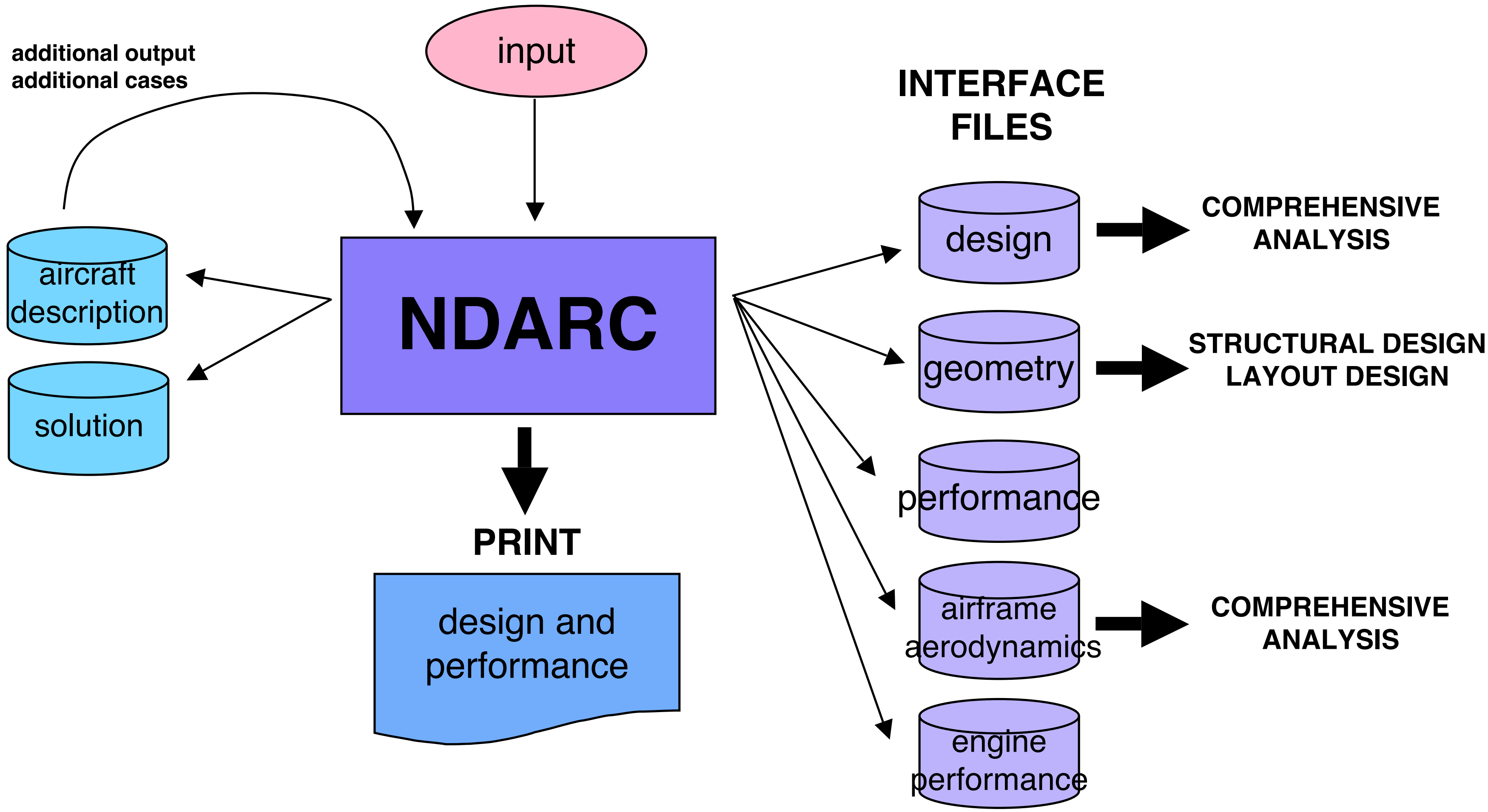
Maximum Effort
method: golden section for maximum endurance, range, or climb;
otherwise secant or false position

Trim
method: Newton-Raphson

Component Performance Evaluation

Blade Flapping
method: Newton-Raphson

NDARC Interfaces



Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

- Debugging

Input Manual

- Aircraft
- Tasks

Tutorial

Sample Input and Output

Helicopter (fixed)

- **Shell script:** *helicopter.bat*
- **Primary input:** *helicopter.njob*
- **Secondary input: aircraft description** *helicopter.airc*, **engine model** *gen2000.ts*

Helicopter, size engine

- **Shell script:** *size_eng.bat*
- **Primary input:** *size_eng.njob*

Output

- **Standard output:** *size_eng.out* **(text);** *size_eng.pdf*
  - *Size_eng_dsgn.doc* **(cut from** *size_eng.out*, **formatted);** *size_eng_dsgn.pdf*
- **Geometry for CAD:** *size_eng.geom*
- **Design and performance:** *size_eng.dsgn* **and** *size_eng.perf* **=>** *size_eng.xls*

Tutorial: execute *helicopter* **and** *size_eng* **jobs**

Line Endings

NDARC files in distribution package generally have mix of line endings

- Unix: LF
- Mac: CR
- Windows: CR-LF

Often source of problems on PCs, when try to compile or run without checking line endings

- Change to Windows line endings with editor such as Notepad++

NDARC Input Format

file “helicopter.bat” — shell script — run NDARC, redirect input and output

```
..\..\bin\ndarc.exe < helicopter.njob > Output\helicopter.out
```

file “helicopter.njob” — primary input — case control, load additional input

```
&JOB &END
&DEFN action='ident',created='today',title='standard input',&END
!###############################################################################
&DEFN action='read file',file='gen2000.ts',&END
&DEFN action='read file',file='helicopter.airc',&END
!===============================================================================
&DEFN quant='Cases',&END
&VALUE
   title='Helicopter',
   TASK_size=0,TASK_mission=1,TASK_perf=1,
&END
&DEFN quant='Size',&END
&VALUE
   param=value,   ! comments
&END
!===============================================================================
&DEFN action='read file',file='helicopter.miss',&END
&DEFN action='read file',file='helicopter.cond',&END
!===============================================================================
&DEFN action='endofcase',&END
&DEFN action='endofjob',&END
```

start with JOB namelist
identify file/data

read engine description
read aircraft description

case control
(organized as appropriate)

read mission definition
read flight condition definition

input finished, run case
job finished, exit code

NDARC Input Format

file “helicopter.airc” — secondary input — aircraft description

```
! Single Main Rotor and Tail Rotor Helicopter
&DEFN action='ident',created='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 param=value, &END   ! comments
&DEFN quant='Size',&END
&VALUE param=value, &END   ! comments
!==============================================================================
&DEFN quant='Aircraft',&END
&VALUE param=value, &END   ! comments
&DEFN quant='Rotor 1',&END
&VALUE param=value, &END   ! comments
!==============================================================================
&DEFN quant='Geometry',&END
&VALUE
    loc_rotor(1)%XoL=0.00,loc_rotor(1)%YoL=0.00,loc_rotor(1)%ZoL=0.00,
    . . .
&END
!==============================================================================
&DEFN quant='TechFactors',&END
&VALUE TECH_xxxx=value,. . .,&END
!##############################################################################
&DEFN action='endoffile',&END
```

identify file/data

configuration defaults

aircraft data

(organized as appropriate)

input for all geometry

input for all technology factors

secondary input finished, exit to primary file

Input

Job reads input from files

- Primary from standard input (perhaps redirected)
  - –Primary can direct to read other files (by name or logical name)
- Namelist format

Primary input starts with JOB namelist, then DEFN namelists to define action and contents

- Can read secondary input files
- Can read aircraft description file: complete description from previous job (but not solution)
- Can read solution file (text or binary): restores solution to state when file created

Secondary input file has DEFN namelists to define action and contents

Input organized as appropriate

- Within files and within namelists

NDARC Input Overview

Input is through namelists

- Primary input is via STDIN, typically redirected to a primary input file
- Primary input file can open and read secondary input files
- Files identified by filename (or logical name, if supported by OS; not supported by Windows)

Input read via *DEFN / VALUE* namelist pair

- *DEFN* defines an action for NDARC to take
- *VALUE* defines input parameters, when input read is commanded
- Read one structure type and instance at a time (e.g cases, rotor, engine, wing, etc)

DEFN namelist may contain the following parameters:

- *ACTION:* character string which defines an action for the code to take (case independent)
- *QUANT*: character string of name of data structure to be input
  - can include numeric character which indexes the data structure when multiple structures of the same type are present. (case independent)
- *SOURCE:* integer for *ACTION='copy'*; identifies index number of data structure of the same type to copy
- *PARENT:* integer which identifies associated parent structure (used for *EngineParam*)
- *FILE:* file name or logical name

When DEFN contains ACTION = 'ident', then the following parameters may be specified:

- *CREATED:* character string of creation time and date (length = 20)
- *TITLE:* character string of title identifying input file (length = 80)
- *VERSION*: code version that input file is intended to be used with (length = 6)
- *MODIFICATION:* character string of code modification identifier (length = 32)

NDARC Input Overview

Options for *ACTION* parameter, NDARC searches string for keyword

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`
'read solution'	*solution*	not *'text'*	read complete solution file, **NAME** = `FILE`
'end'	*end* (or EOF)		Same as **ACTION** = '*endofjob'*
'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 of type structure
'read namelist'	*list*	'structure'	read **VALUE** namelist of type structure
'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*		setup defaults based on aircraft configuration
'identification'	*ident*		identify file
'end'	*end* (or EOF)		close file, return to primary file

NDARC Input Overview

Options for *QUANT* parameter related to case execution and sizing

QUANT	data structures read	Maximum n	Purpose
'Job'	Job		modify job parameters for next case
'Cases'	Cases		controls case execution and input/output
'Size'	SizeParam		controls sizing process
*'SizeCondition **n**'*	one FltCond+FltState	nFltCond	input one of n point sizing conditions, number of conditions set with Size%nFltCond
*'SizeMission **n**'*	one MissParam, MissSeg+FltState as array	nMission	input one of n sizing missions; number of missions set with Size%nMission
'OffDesign'	OffParam		controls off-design analysis
*'OffMission **n**'*	one MissParam, MissSeg+FltState as array	nMission	input one of n off-design missions; number of mission set with OffDesign%nMission
'Performance'	PerfParam		controls point performance analysis
*'PerfCondition **n**'*	one FltCond+FltState	nFltCond	input one of n point performance conditions; number of conditions set with PerfParam%nFltCond
'MapEngine'	MapEngine		controls generation of engine maps as function of altitude, flight speed, turbine speed, and power factor
'MapAero'	MapAero		controls generation of aerodynamic performance maps as function of alpha, beta and aircraft controls
'Solution'	Solution		Solution procedure parameters

NDARC Input Overview

Options for *QUANT* parameter related to aircraft synthesis/description

QUANT	Data structures read	Maximum n	PARENT
'Cost'	Cost, CostCTM		
'Aircraft'	Aircraft		
'Systems'	Systems, WFltCont, WDeIce		
'Fuselage'	Fuselage, AFuse, WFuse		
'LandingGear'	LandingGear, AGear, WGear		
'Rotor ***n****'*	Rotor, PRotorInd, PRotorPro, PRotorTab, IRotor, DRotor, WRotor	nRotor	
'Wing ***n****'*	Wing, Awing, WWing, WWingTR	nWing	
'Tail ***n****'*	Tail, ATail, WTail	nTail	
'FuelTank'	FuelTank, WTank		
'Propulsion ***n****'*	Propulsion, WDrive	nPropulsion	
'EngineGroup ***n****'*	EngineGroup, DEngSys, WEngSys	nEngineGroup	
'JetGroup ***n****'*	JetGroup, DJetSys, WJetSys	nJetGroup	
'ChargeGroup ***n'***	ChargeGroup, DChrgSys, WChrgSys	nChargeGroup	
'TechFactors'	All variables of form TECH_xxxx		
'Geometry'	All derived-type Location variables		

NDARC Input Overview

Options for *QUANT* parameter related to aircraft synthesis/description

QUANT	Data structures read	Maximum n	PARENT
*'EngineModel **n**'*	EngineModel, EngineParam	nEngineGroup	
*'EngineParamN **n**'*	EngineParam	nspeed	EngineModel number
*'EngineTable **n**'*	EngineTable	nEngineTable	
*'CompressorModel **n**'*	CompressorModel	nCompressorModel	
*'MotorModel **n**'*	MotorModel	nMotorModel	
*'JetModel **n**'*	JetModel	nJetModel	
*'FuelCellModel **n**'*	FuelCellModel	nFuelCellModel	
*'SolarCellModel **n**'*	SolarCellModel	nSolarCellModel	
*'BatteryModel **n**'*	BatteryModel	nBatteryModel	

NDARC Data Structures

Internal Location	QUANT
Design	*No Inputs*
Cases	Cases
Size	Size
SizeParam	
FltCond(nFltCond)	SizeCondition n
FltState(nFltCond)	
Mission(nMission)	SizeMission n
MissParam	
MissSeg(nSeg)	
FltState(nSeg)	
OffDesign	OffDesign
OffParam	
Mission(nMission)	OffMission n
MissParam	
MissSeg(nSeg)	
FltState(nSeg)	
Performance	Performance
PerfParam	
FltCond(nFltCond)	PerfCondition n
FltState(nFltCond)	
MapEngine	MapEngine
MapAero	MapAero
Solution	Solution
Cost	Cost
CostCTM	
Aircraft	Aircraft
[Location]loc_cg	Geometry
Weight	*No Inputs*
XAircraft	*No Inputs*
Systems	Systems
Weight	*No Inputs*
WFltCont	Systems
WDeIce	

Internal Location	QUANT
Fuselage	Fuselage
[Location]loc_fuselage	Geometry
AFuse	Fuselage
Weight	*No Inputs*
WFuse	Fuselage
LandingGear	LandingGear
[Location]loc_gear	Geometry
AGear	Landing Gear
Weight	*No Inputs*
WGear	Landing Gear
Rotor(nRotor)	Rotor n
[Location]loc_rotor	Geometry
[Location]loc_pylon	
[Location]loc_pivot	
[Location]loc_nac	
PRotorInd	Rotor n
PRotorPro	
PRotorTab	
IRotor	
DRotor	
Weight	*No Inputs*
WRotor	Rotor n
Wing(nWing)	Wing n
[Location]loc_wing	Geometry
Awing	Wing n
Weight	*No Inputs*
WWing	Wing n
WWingTR	
Tail(nTail)	Tail n
[Location]loc_tail	Geometry
ATail	Tail n
Weight	*No Inputs*
WTail	Tail n

Internal Location	QUANT
FuelTank(nTank)	FuelTank n
[Location]loc_auxtank	Geometry
Weight	*No Inputs*
WTank	FuelTank n
Propulsion(nPropulsion)	Propulsion n
Weight	*No Inputs*
WDrive	Propulsion n
EngineGroup(nEngineGroup)	EngineGroup n
[Location]loc_engine	Geometry
DEngSys	EngineGroup n
Weight	*No Inputs*
WEngSys	EngineGroup n
JetGroup(nJetGroup)	JetGroup n
[Location]loc_jet	Geometry
DJetSys	JetGroup n
Weight	*No Inputs*
WJetSys	JetGroup n
ChargeGroup(nChargeGroup)	ChargeGroup n
[Location]loc_charge	Geometry
DChrgSys	ChargeGroup n
Weight	*No Inputs*
WChrgSys	ChargeGroup n
EngineModel(nEngineModel)	EngineModel n
[EngineParam]Param	
[EngineParam]ParamN(nspeed)	EngineParamN n
EngineTable(nEngineTable)	EngineTable n
CompressorModel(nCompressorModel)	CompressorModel n
MotorModel(nMotorModel)	MotorModel n
JetModel(nJetModel)	JetModel n
FuelCellModel(nFuelCellModel)	FuelCellModel n
SolarCellModel(nSolarCellModel)	SolarCellModel n
BatteryModel(nBatteryModel)	BatteryModel n

Conventions

Case not important in character string input

- Character string input consists of keywords
- Code searches for keywords in string

Default values specified in dictionary

- Blank implies a default of zero
- All elements of arrays have the same default value

Tasks, aircraft, and components have title variables

- And notes variables (long character string)

Conventions

QUANT **string includes structure number**

- **If absent for component, number = 1**

  QUANT='rotor' same as *QUANT='rotor 1'*

- **Or next condition or next mission**

  QUANT='SizeMission' same as *QUANT='SizeMission n+1'*

  where *n* is last mission already defined

Case inherits input for flight conditions and missions from previous case (default ***INIT_input*** **)**

- **May need to delete flight conditions or missions**

ACTION='delete one',QUANT='structure n'	delete n-th
ACTION='delete all',QUANT='structure'	delete all
ACTION='delete last',QUANT='structure n'	delete n-th to last

Conventions

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

Each mission (*MissParam*, *MissSeg*, and *FltState* variables) input in 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 first array index always segment number**
  - **So columns of input correspond to mission segments**

    param = segment1, segment2, segment3, . . .

    param(1,k) = segment1, segment2, segment3, . . .

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

Separate namelists for all technology factors (all *TECH_xxx* variables), and all geometry (all *Location* variables

- **Scalar in the *Rotor*, *Wing*, *Tail*, *Propulsion*, *EngineGroup*, *JetGroup*, or *ChargeGroup* input becomes array in *TechFactors* or *Geometry* input**
- **Note *Location* variable is the array (for example, *loc_rotor(1)%SL*)**

JOB Namelist

Start of NDARC input, only read once

Default values always used to start first case (before input read)

Set initialization of parameters from previous case

- **Inherit only input (default)**

  INIT_input=1,INIT_data=0
- **Inherit design and solution: all parameters (input, input-modified, derived)**

  INIT_input=2,INIT_data=2

File write behavior

- **Default:** *Open_status=2*,
  - **STATUS='NEW' behavior in FORTRAN**
  - **On some platforms will cause NDARC to exit with error if file already exists**
- **For Windows machines use** *Open_status=1*
  - **STATUS='REPLACE' behavior**
  - **Automatically overwrites existing file**

JOB Namelist

Common: Job

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

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

Variable	Type		Description	Default
		+	Errors	
ACT_error	int	+	action on error (0 none, 1 exit)	1
ACT_version	int	+	action on version mismatch in input (0 none, 1 exit)	0
		+	File open	
OPEN_status	int	+	status keyword for write (0 unknown, 1 replace, 2 new, 3 old)	2

Cases Structure

Chapter 5

Structure: Cases

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

case documentation

tasks: 0 none, 1 execute

Data Structure Sizes

Parameters in NDARC_structures module

Revise and re-compile code if encounter limit

Chapter 3

Parameters

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

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

- Debugging

Input Manual

- Aircraft
- Tasks

Tutorial

Job Setup

Inputs
Shell script ([task].bat, [task].com, etc)
- **Operating System-dependent**

Job file ([task].njob)
- **Controls Execution**
- **Defines solution and output options**
- **Loads other files**

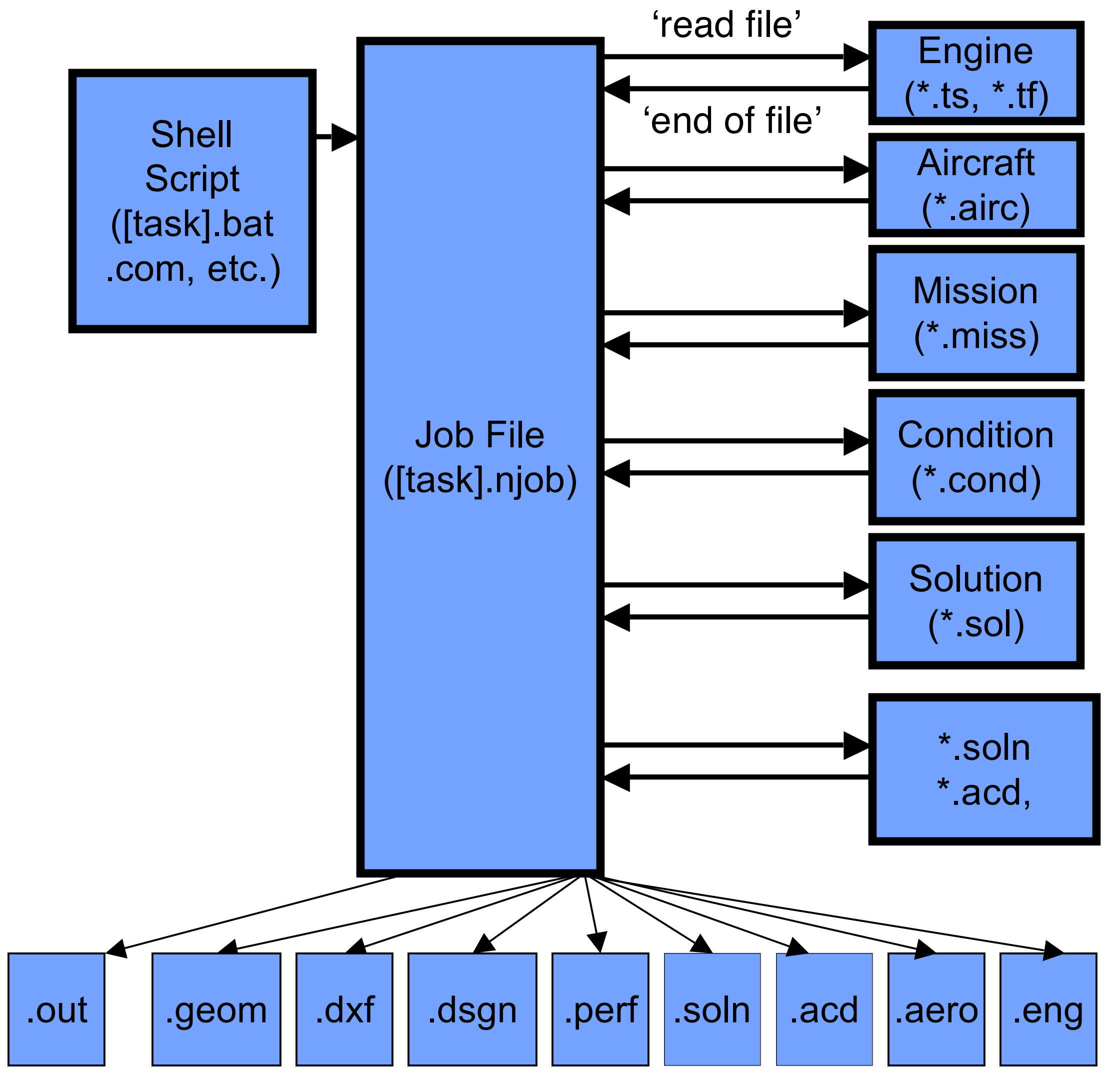
Engine file (*.ts, *.tf)
- **Engine properties and scaling parameters**

Aircraft file ([run].airc)
- **Aircraft configuration,geometry, trim strategies**
- **Tech factors**

Mission file (*.miss)
- **Sizing or off-design missions**

Condition file (*.cond)
- **Sizing or off-design point analysis**

Aero/Engine maps (*.maero, *.meng)



Outputs
OUT file ([task].out)
Geometry files
- **Pro/E compatible format ([task].geom)**
- **AutoCad 3D format ([task].dxf)**

Design summary ([task].dsgn)
Cond/Miss performance summary ([task].perf)
Solution dump ([task].soln)
Aircraft dump ([task].acd)
Aerodynamic map ([task].aero)
Engine map ([task].eng)

Input: *.njob file

Proxy for command prompt input

- **Files may only be loaded at this level**

Structures in the Case Control section of the NDARC file

- `Size`
- `(Solution)`
- `(Performance)`
- `(OffDesign)`

.njob file

Job Namelist
    Identify

Case Control

Cases
    Load Files
    (Solution Control)
    Sizing
    Adjust namelists as necessary

Cases
    Load Files
    (Solution Control)
    Sizing
    Adjust namelists as necessary

Namelists	Actions	Quants
&JOB &DEFN &VALUE	‘ident’, ‘read file’, ‘end of case’, ‘end of job’	Cases, Size, (Solution), (Performance), (OffDesign)

Input: *.ts file

Engine inputs

- **Mapping from engine deck (beyond scope of tutorial)**
- **Referred Parameter Turboshaft Engine Model**

Structures

- **EngineModel**
- **EngineParam**

.ts file

Header Identify Change log
General information Describe ratings, ref params
By rating Define curves and breakpoints for each rating defined above

Namelists	Actions	Quants
&DEFN &VALUE	‘ident’	EngineModel

Input: *.airc file

Aircraft Definition

- **Design parameters**
  - **–Disk Loading, Wing loading**
- **Components**
- **Trim controls and mapping**

Structures

- **Aircraft**

.airc file

Header
Identify
Change log

Required resources, assumptions

Various components and quants listed below

Namelists	Actions	Quants
&DEFN &VALUE	'ident'	Aircraft, Cost, Systems, Fuselage, LandingGear, Rotor, Force, Wing, Tail, FuelTank, Propulsion, EngineGroup, TechFactors, Geometry

Inputs: *.cond file

Point design condition(s)

Point/sweep off-design condition(s)

Structures

.cond file

Header Identify Applicabe aircraft
Performance (optional) Number of conditions
PerfCondition or OffDesign

Namelists	Actions	Quants
&DEFN &VALUE	‘ident’	(Performance), SizeCondition or OffDesign

Inputs: *.miss file

May be the location of `nMission`

Sizing mission(s)

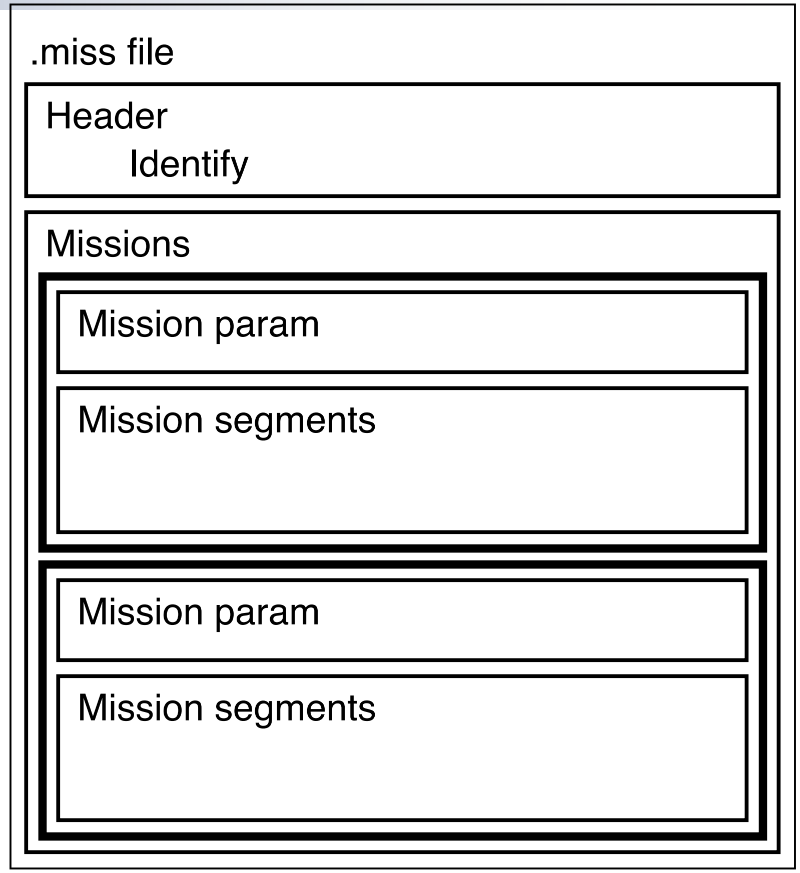
- **Segments may be referenced**

Off-design mission(s)

- **Each point on a PL vs. R curve**

Structures

- `OffParam`
- `MissParam`
- `MissSeg` **as array**
- `FltState` **as array**



Namelists	Actions	Quants
&DEFN &VALUE	‘ident’	(OffDesign), SizeMission or OffMission

Input: *.sol file

Solution methodology control

- **Optionally located in** `[task].njob`

```
&DEFN quant='Solution', &END
&VALUE
  trace_xxx = #,
  niter_xxx = #,
  relax_xxx = #,
  perturb_trim = 0.01,
&END
```

.sol file

Header Identify
Solution parameters

Namelists	Actions	Quants
&DEFN &VALUE	‘ident’	Solution

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

- Debugging

Input Manual

- Aircraft
- Tasks

Tutorial

STDOUT Overview

```
######################################################################################################################
NDARC -- NASA Design and Analysis of Rotorcraft
Version 1.8
######################################################################################################################
```
Report reading inputs
```
######################################################################################################################
```
Print out input values (controlled by *Cases%WRITE_input* variable)
```
######################################################################################################################
```
Execution messages (including debug info, set debug level with *Solution%trace_xxxx* variables)
```
######################################################################################################################
Case number  1, Time-Date = hh:mm:ss dd-mmm-yyyy, Identification = aaaaaaaa
Case Convergence
```
Convergence information
```
######################################################################################################################
```
Design summary (also optionally output to *FILE_design*)
```
######################################################################################################################
```
Design weight information (also optionally output to *FILE_design*)
```
######################################################################################################################
```
Performance information (also optionally output to *FILE_perf*)

- Point Performance Summary (Design / Off-Design)
- Mission Summary (Design / Off-Design)
- Point Performance Flight State Summary (Design / Off-Design)
- Mission Segments Flight State Summary (Design / Off-Design)
- Point Performance Flight State Loads & Aero (Design / Off-Design)
- Mission Segment Flight State Loads & Aero (Design / Off-Design)

```
######################################################################################################################
NDARC end of case number  1 (CPU time =   n.nnn min, elapsed time =   n.nnn min)
######################################################################################################################
Read input parameters, case number  2
End of job
######################################################################################################################
NDARC end of job (CPU time =   n.nnn min, elapsed time =   n.nnn min)
```

NDARC Output Overview

Primary output stream to STDOUT, typically redirected to file
Additional files can be written to aid in post-processing / data transfer

- Output set in *Cases* structure
  - *OUT_design*: controls writing of tab delimited file summarizing design (name = *FILE_design*)
  - *OUT_perf*: controls writing of tab delimited file of mission and performance output (name = *FILE_perf*)
    - Same information as in STDOUT; for spreadsheets, full-precision numbers
  - *OUT_geometry*: controls writing of separate file of geometry parameters for use with CAD software (name = *FILE_geometry*)

Output formatting controlled with *Cases%WRITE_xxxx* variables

Structure: Cases 25

		+	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)	' '

Output: *.out file

File is a redirection of STDOUT

- **This would all go to the console otherwise**

Making use of `[task].out`

- **Simple pagination**
  - **Page break character**
- **Segments wrap**

[task].out file

- Responses to [task.njob] (page 1)
- Initial values (page 2)
- Cases (multiple)
  - Sizing (page 3)
    - Size
      - Param
  - Off Design
    - Param
  - Convergence Summary (page 4)
    - Param
  - Design (page 5)
  - Weight details (page 6)
  - Performance Summaries
  - Performance Details (multiple)
  - Aero (multiple, last in sweep only)

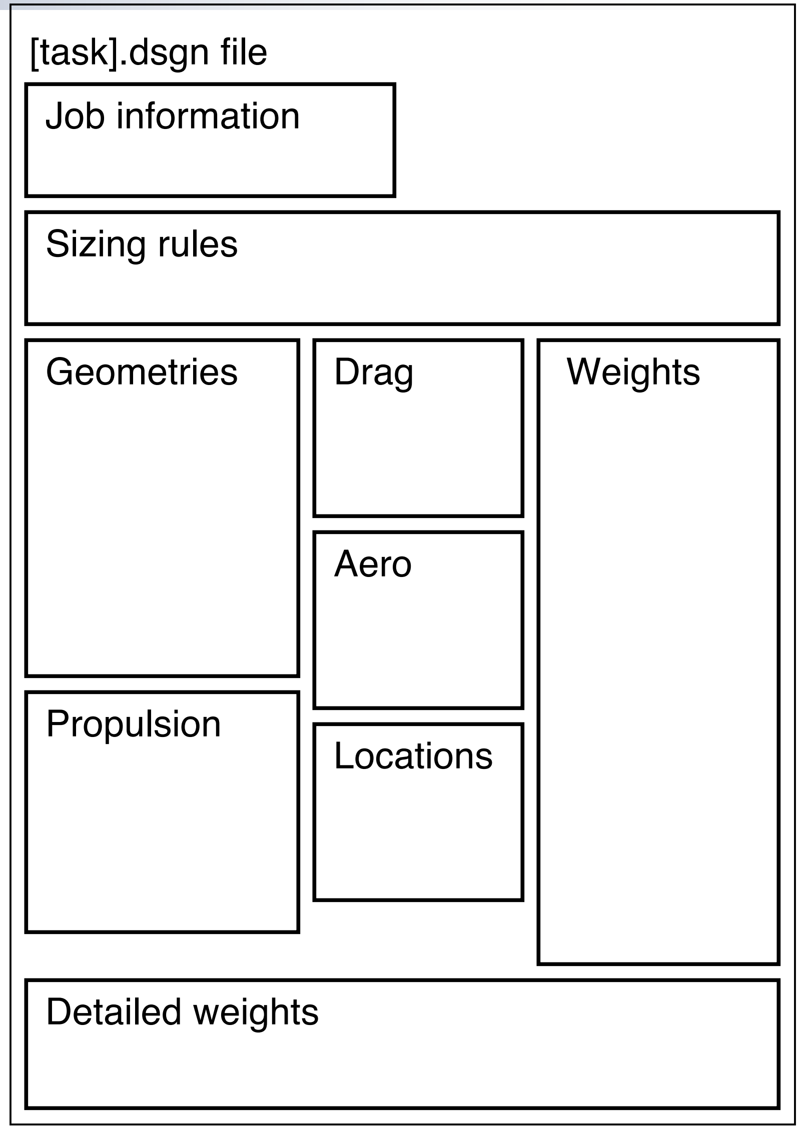
Output: *.dsgn file

Flags in `[job].njob` **quant='Cases'**

```
OUT_design=1,
FILE_design='[task].dsgn',
```

Contents of `[task].dsgn`

- **Tab-deliminated, blocks of data**
- **Aircraft descriptive summary**
  - –**Weights & tech factors**
  - –**Sizing rules**
  - –**Dimensions, power, drag**
  - –**Some reference conditions**



Output: *.perf file

Flags in `[job].njob` **quant='Cases'**

```
          OUT_perf=1,
 FILE_perf='[task].perf',
```

Contents of `[task].perf`

- **Tab-delimited, columnar data**
- **Summaries**
- **Detailed breakdowns**
- **Component Loads and Aerodynamics**
  - **For each segment**
  - **Only last segment in a sweep**

[task].perf file

Summary
Performance Summary
Mission Summary
Performance Details
Mission Details
Component Loads and Aerodynamics

Output: *.geom file

Flags in `[job].njob` **quant='Cases'**

```
OUT_geom=1,
FILE_geom='[task].geom',
```

Contents of `[task].geom`

- **Variable = value list**
- **Can drive Pro/Engineer models**

[task].geom file

- Aircraft top-level info
- Fuselage
- Landing Gear
- Fuel Tank
- Rotors
- Tails
- Wings
- Propulsion
- Engines
- Location

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

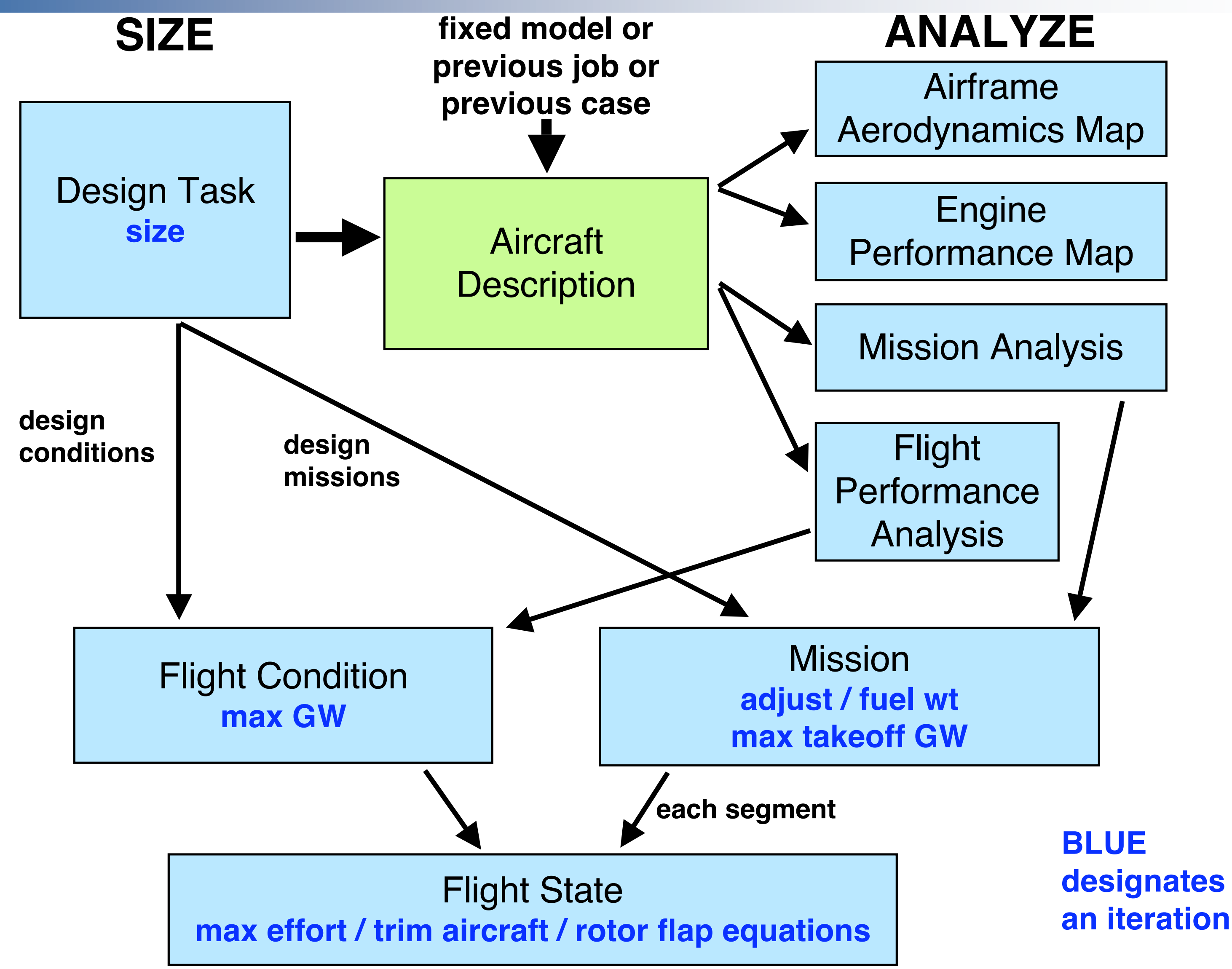
- Debugging

Input Manual

- Aircraft
- Tasks

Tutorial

NDARC Tasks


SIZE
ANALYZE
fixed model or previous job or previous case
Design Task
size
Aircraft Description
Airframe Aerodynamics Map
Engine Performance Map
Mission Analysis
Flight Performance Analysis
design conditions
design missions
Flight Condition
max GW
Mission
adjust / fuel wt
max takeoff GW
each segment
Flight State
max effort / trim aircraft / rotor flap equations
BLUE designates an iteration

Solution Procedure

Sizing Task

Size Iteration

method: successive substitution

- Missions
- Flight Conditions

↓

Mission Analysis

- Missions

↓

Flight Perf Analysis

- Flight Conditions

Flight Condition

Maximum GW

method: secant or false position

- Flight State

Mission

Mission Iteration
adjust, fuel weight

method: successive substitution

Segments

Maximum GW

method: secant or false position

- Flight State

Flight State

Maximum Effort

method: golden section for maximum endurance, range, or climb;
otherwise secant or false position

Trim

method: Newton-Raphson

Component Performance Evaluation

Blade Flapping

method: Newton-Raphson

NDARC Solution Procedure

Details in Theory Manual

Namelist *Solution* controls solution procedure globally

- Key parameters controlling solution can be over-ridden for each flight condition and mission segment

Procedure	Objective(s)	Method(s)	Notes
Sizing	Size A/C: Rotor, Wing, DGW, WMTO, Peng, XMSN, Fuel Capacity	Successive Substitution	Setup using *SizeParam* variables
Mission	TOGW, Wfuel, Wpay, Range/Time	Successive Substitution	
Max GW	Max GW for P available	Secant False Position	
Max Effort	Set for flight state using *max_quant* & *max_var*	Secant False Position Golden-Section Curve-Fit	Selected by *method_fly* & *method_flymax*
Trim	A/C controls & motion	Newton-Raphson	*STATE_trim* controls trim type

NDARC Solution Procedure

Nested solution iterations: rotor, trim, max effort (fly), max GW, mission, size

Parameters for each iteration:

toler_zzzz: **solution tolerance**

- **Small enough for accuracy (smaller wastes computation time)**
- **Accuracy required of inner loop may be driven by convergence of outer loops**

relax_zzzz: **relaxation factor**

- **Reduce to achieve convergence;** *relax=1.0* **works for many loops**

niter_zzzz: **maximum number of iterations**

trace_zzzz: **produce trace of iteration in output**

- **Needed to diagnose run failure**

Rotor Flapping

Solves the rotor flapping equations with Newton-Raphson technique

- **Blade element method implemented for collective & cyclic pitch angles (or flap angles) and inplane hub forces, not rotor performance**

Rotor%KIND_control **determines control mode**

- **Feathering plane or tip path plane command**
- **Collective commands thrust or blade pitch**
- **Rotor solution procedure solves for unknowns**

Solution control: *niter_rotor*, *toler_rotor*, *relax_rotor*, *deriv_rotor*, *maxinc_rotor*

- **Set for each rotor**
- *relax_rotor* **&** *toler_rotor* **can be adjusted for each flight state**

Tighter than default toler_trim may improve solution stability

Linear flap equations with no stall in aerodynamics — usually converges

- **Divergence of iteration usually caused by problems with an outer loop**

Trim

Solves pilot controls and motion that are required to reach equilibrium

- Process of trimming is part of solution procedure, trim schemes are part of aircraft specification
  - Up to *mtrimmax* (16) degrees of freedom can be trimmed at each state
  - Up to *ntrimstatemax* (20) trim schemes can be defined
- *Aircraft%IDENT_trim*: list of labels of each scheme
  - When initial input is setup with `action='config'`, 9 default schemes created:

	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'

  - Schemes can be altered or created using inputs in *Aircraft* structure
    - › Typical to modify *trim_var* to reflect control effectors of aircraft configuration
  - Independent variables set with *trim_var,* some basic parameters+aircraft controls
  - Dependent variables set with *trim_quant*, see input manual for complete listing
- Trim matrix must be square, number of trim quantities = number of trim variables
- *Solution%init_trim*: When set to `1` forces controls to globally reinitialize to flight state input for each iteration
  - *FltAircraft%init_trim*: can force reinitialize for specific flight state

Newton-Raphson Trim Iteration

Successive values of trim variables solving *f(x)=0* calculated using:

$$\vec{x}_{n+1} = \vec{x}_n - \lambda D^{-1} \vec{f}\left(\vec{x}_n\right)$$

where *D* is the derivative matrix (Jacobian)

- Default is calculation of *D* once at start, control with *Solution%mpid_trim* for periodic recalc (expensive)
- 1st or 2nd order difference for estimates of derivatives, *Solution%deriv_trim*
  - *deriv_trim=2* often helps convergence
- Derivative step size set with *Solution%perturb_trim*

Convergence good if initial guess for variables is close to final solution

- Improving guess often more important than adjusting solution parameters
- Relaxation factor (λ) to improve convergence robustness set with *relax_trim*
- Convergence checked by testing if function evaluation within specified tolerance (*toler_trim*)
- Number of iterations also limited with *niter_trim*

relax_trim, toler_trim, init_trim, perturb_trim **can be adjusted for each flight state**

Maximum Effort

Solution loop which adjusts variable (*max_var*) to maximize quantity (*max_quant*)

- Two loop levels may be specified
- Typical max_quant: `range`, `end`, `Pmargin`, `climb`, `alt`

Solution procedures:

- *method_fly*: `1`: secant, `2`: false position
  - Used to find specified power margin or thrust margin
- *method_flymax*: `1`: secant, `2`: false position, `3`: golden section, `4`: curve fit
  - Search for absolute maximum range, endurance, climb

Parameters:

- *maxderiv_fly*: limits the derivative value for better convergence (default is no limit)
- *maxinc_fly*: limits the incremental change in variable size between iterations
- *relax_fly*: relaxation factor
- *perturb_fly*: perturbation increment (faction of ref. value) for derivative calc
- initial value of adjusted variable can be important for convergence

relax_fly, toler_fly, init_fly, perturb_fly, maxderiv_fly, maxinc_fly, method_flymax **can be adjusted for each flight state**

Maximum Effort

Secant & False Position:

- Based on Newton-Raphson method, updates derivative info each step
- Secant derivative based on current and previous aircraft trim evaluations
- False Position derivative based on trim evaluations which bracket solution

Golden Section:

- Assumes unimodal function in region of interest
- Selection of new interior search point based on the golden ratio
- Subsequent interior search points selected to continuously bracket minimum
- Search ends when search region < tolerance
- Secant method to find 99% max specific range using G-S velocity from max SR

Curve Fit:

- Useful for flat maximums where tight trim tolerances are required to get G-S behavior
- Least-squares curve fit to region of interest selected to bracket maximum
- Domain of curve fit selected such that function is *rfit_fly**max at boundaries
    - Default value: `rfit_fly=0.98`
- *nfit_fly* sets order of polynomial (`2`: quadratic, `3`: cubic)
- Both max and 99% of max can be found (best range)

Example Flight State Input

Best Range Example:

```
&DEFN quant='PerfCondition', &END     !  off-design performance condition
&VALUE
  title = 'Example Vbr Calculation'
  . . .
  SET_max=1,
  max_quant='range',
  max_var='speed',
  Vkts=200.,                         ! speed will be adjusted, just an initial guess
  . . .
&END
```

Best Climb Example:

```
&DEFN quant='PerfCondition', &END     !  off-design performance condition
&VALUE
  title = 'Example Best Climb Calculation'
  . . .
  SET_max=2,
  max_quant='Pmarg','climb',
  max_var='ROC','speed',
  Vkts=120.,ROC=500.,                ! speed and ROC adjusted, initial guesses
  . . .
&END
```

Maximum Gross Weight

Adjusts gross weight such that power required = power available from propulsion group

- Use *fPav* and *dPav* to set power available

$$P_{req} = fPav\, P_{av} + dPav$$

- Allows for calculation of gross weight at arbitrary power (`fPav=0, dPav`=power)

Solved using secant or false position method (*method_maxGW*)

- Solution control with *niter_maxgw*, *toler_maxgw*, *relax_maxgw*, *perturb_maxgw*, *maxderiv_maxgw*, *maxinc_maxgw* as with max effort solution procedure.
- Convergence based on magnitude of gross weight increment, with initial weight used as the scaling term for tolerance

For missions, *MissSeg%MaxGW* identifies segments where max gross weight is evaluated

relax_maxgw, toler_maxgw, perturb_maxgw, maxderiv_fly, maxinc_fly **can be adjusted for each flight state**

Example Flight State Input

Max Gross Weight Case that sizes the transmission:

```
&DEFN quant='SizeCondition', &END  ! sizing performance condition
&VALUE
  DESIGN_xmsn=1,DESIGN_wmto=1,     ! set what parameters can be designed
  . . .
  SET_GW='max',                    ! maximize Gross Weight (default fPav=1.,dPav=0.)
  rating='MRP',fPower=0.95,        ! define power available
  SET_Plimit=0,                    ! calculating drive limits, so turn off limit
  . . .
&END
```

Mission Iteration

Collection of segments (sequential flight states)

Successive Substitution method used to solve mission performance

- Iteration performed on take-off fuel weight (*toler_miss)*
- Segments with range credit require an inner iteration
- For fixed take-off weight missions delta fuel is used to adjust segments
- Relaxation factor used in parameter update
  - *relax_miss*: fuel weight relaxation factor
  - *relax_range*: range credit relaxation factor
  - *relax_gw*: max take-off gross weight relaxation factor
- Maximum number of iterations set with *niter_miss*

Adjustment of *relax_miss*, *relax_range*, *toler_miss* **for each mission can be required**

Sizing Solution Procedure

Iteration to find engine power or rotor size, weight and dimensions of aircraft

- **Two nested sizing loops**
  - Inner loop on parameters: *DGW, WMTO, Plimit_ds, Wfuel_cap*
    - –*niter_param*: sets number of inner parameter loops
    - –*niter_param=1*: parameters updated as part of performance loop
  - Outer loop on performance: engine size or rotor diameter
    - –*niter_size*: sets number of outer performance loops
- **Method of successive substitution used**
  - Convergence based on sizing parameters and weight empty
  - *toler_size*: sets tolerance for convergence; smaller than default typically required to get repeatable results when running multiple cases.
  - *relax_size*: relaxation factor for parameter update; may need to adjust with tolerence
  - Should adjust mission and max effort tolerances along with sizing tolerance

Take care to avoid defining unsolvable design problem

- For sizing of a new concept, should start with simple set of missions/parameters
- Convergence best when aircraft variables initialized with reasonable values

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

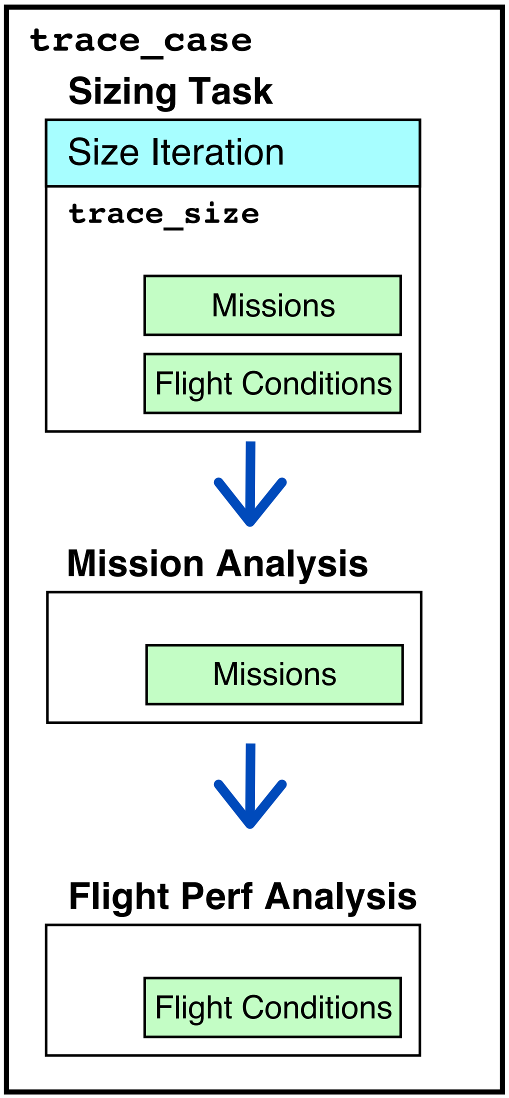
- Debugging

Input Manual

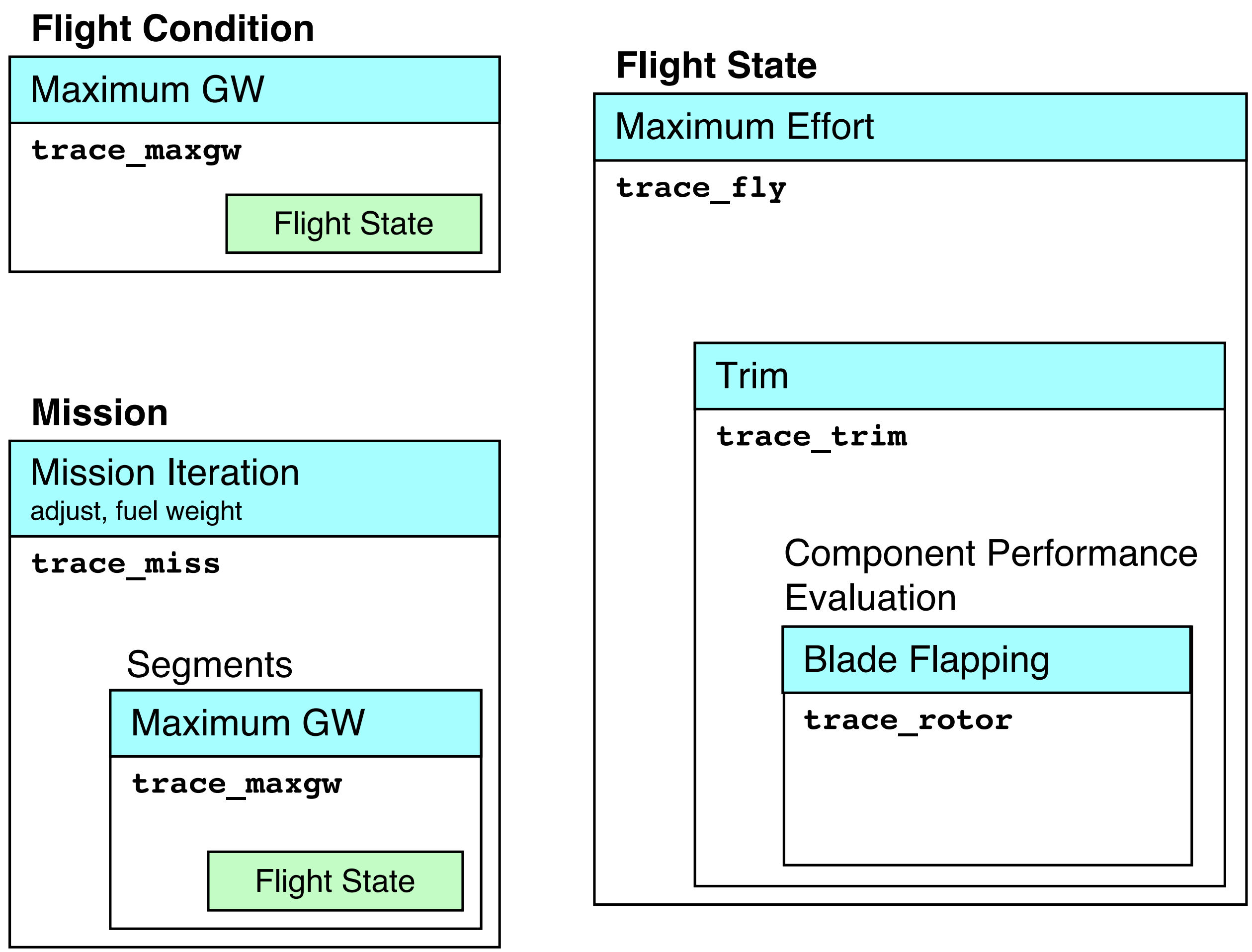
- Aircraft
- Tasks

Tutorial

Solution Tracing (Debug) Overview



Trace variables can be set in the *Solution* quant to report details of the solution procedure to the STDOUT stream



Debugging

Fatal NDARC errors

- Depending on compiler, verbose output only seen at shell, so use the “pause” command in cases where NDARC crashes without much useful information in the output file

Input Errors

- Turn off *ACT_error* at your own peril!
  - NDARC provides warnings for input errors, at end of output file

Logic Errors

- NDARC catches many, but not all

Convergence

- Using the various trace options provides insight to what is failing

Solution parameters

- Relaxation factors
- Perturbation
- Tolerance
- Maximum step size
- Solution methodology (golden section, …)

Importance of initial guesses

- Sometimes, convergence improved if guess is close, high, or low
  - Can run off-design (non-sizing) case to find starting guesses

Check Convergence

Search for "case convergence" in output file:

```
###################################################################################################################
Case number  1, Time-Date =  8:44:37 10-Feb-2014, Identification =
Case Convergence

Iteration          Status (entire case)  MaxIter  Tolerance  Perturb  Relaxation          Method
-------------------------------------------------------------------------------------------------------------------
Size performance: converged              100     0.01000             0.500
Size parameters:  converged                1     0.01000             1.000 DGW 1.000 xmsn 1.000 wmto/sdgw 1.000 tank 1.000 thrust
Mission:          converged               40     0.01000             1.000 fuel 1.000 range 1.000 maxTOGW
Maximum GW:       converged               40     0.00200    0.0200   0.500              maxderiv 0.000  method = secant
Maximum effort:   converged               80     0.00200    0.0500   0.500              maxderiv 0.000  method = secant
Trim:             converged               40     0.00100    0.0020   0.400              deriv = first order     mpid =   0
Rotor 1:          converged               40     0.01000             0.500              deriv = first order
Rotor 2:          converged               40     0.01000             0.500              deriv = first order
Check Preq, Qlimit, Wfuel:  tolerance = 0.00500
-------------------------------------------------------------------------------------------------------------------
```

followed by convergence details for size iteration, each mission, and each flight condition and mission segment

If any iteration not converged, results are not reliable

Serious convergence problems can produce floating-point overflow

- **Solution behavior depends on compiler and operating condition**
- **Best if job exits on overflow**

Solution Tracing

If job fails, use convergence information or trace of solution (*trace_case=2* **) to identify solution/iteration that did not converge**

The use *trace_size, trace_miss, trace_maxgw, trace_fly, trace_trim, trace_rotor* **to view details of appropriate solution procedure**

- **Typically run jobs with** *trace_size=2* **(or** *trace_miss=1* **if no sizing task)**
  - –Routinely running jobs with other trace variables set generates lot of output and slows execution
- **Turn trace variables on one at at time**
  - –Trace output for nested iterations is difficult to interpret
- **Focus trace on iteration that is problem**
  - –Can turn on trace globally, or for individual missions or mission segments or flight conditions
  - –Use *trace_start* to delay start of detailed output

Solution Tracing

Solution iterations are uniquely identified internally with a running counter

- *trace_case*: flag to show counter information in STDOUT (0 – none ,1 – top level (outer loop) tracing, 2 – debug level of tracing)
- *trace_start*: suppress trace information in STDOUT until internal counter is greater than *trace_start*
- Output format: `Fly`(TYPE)(n) `* counter`
  - (TYPE): The name of the subroutine for the procedure being executed
    - `Aircraft` – a maximum effort solution procedure
    - `Mission` – mission loop solution procedure
    - `MissionSol, segment n` – mission flight state procedure or the nth segment
    - `Mission_MaxGW` – max gross weight solution procedure in a mission
    - `Condition_MaxGW` – max gross weight solution procedure
    - `Condition` – fight state for both sizing and off-design analysis
  - (n): nth input condition, separate number for sizing and off-design analysis

```
FlyMission 1 *          700
  FlyMissionSol, segment 1 *          705
  FlyMissionSol, segment 2 *          710
  FlyMissionSol, segment 3 *          713
FlyCondition 1 *          715
  FlyCondition_MaxGW *          718
FlyCondition 2 *          725
  FlyCondition_MaxGW *          730
…
Flight performance analysis (point operating conditions)
FlyCondition 1 *       1507
FlyCondition 1 *       1510
```

Sizing task with 1 mission and 2 performance conditions

Trace output suppressed with `trace_start=699`

`trace_case=2` in red

trace_size

Provides information on how aircraft sizing is progressing

- Sizing trace output denoted with `S` on right side
- First line identifies solution parameter values
- Power data (red) shown only when `trace_size=2`
- Asterisk (*) denotes aircraft parameters that are being considered in sizing
- When *Solution%niter_param*=1 design parameters all updated simultaneously (no inner sizing loop for *DGW, WMTO, Plimit_ds, Wfuel_cap* and *Tdesign*)
- Unless `trace_case=2`, solution counter output is suppressed

Solution%toler_size — *Solution%relax_size* — *Solution%niter_size* — *Solution%niter_param*

```
Size (tolerance = 0.01000, relaxation = 1.000, maximum iterations =  40 performance  1 parameter)                  S
start                                                                                                              S
          aircraft weight         design GW     =    142881.0*              max TO weight =    190031.7            S
          aircraft weight         fuel tank cap =     34280.0               weight empty  =     87620.9*           S
          prop 1 engine group 1   engine power  =     20000.0*              Plimit_es     =     20000.0            S
          prop 1 engine group 2   engine power  =     20000.0*              Plimit_es     =     20000.0            S
          propulsion 1                                                      Plimit_ds     =     30096.0            S
          rotor 1                 radius        =      37.500               Plimit_rs     =     40000.0*           S
          rotor 2                 radius        =      37.500               Plimit_rs     =     40000.0*           S
performance iteration  1 (quantity and error ratio)                                                               S
          aircraft weight         design GW     =    147731.4*   339.47     max TO weight =    196482.8    0.0000  S
          aircraft weight         fuel tank cap =     34280.0    0.0000     weight empty  =     87620.9*   0.0000  S
          prop 1 engine group 1   engine power  =     19168.7*   20.782     Plimit_es     =     20000.0    927.20  S
          prop 1 engine group 2   engine power  =     19168.7*   20.782     Plimit_es     =     20000.0    927.20  S
          propulsion 1            Pratio        =      0.9584               Plimit_ds     =     30096.0    927.20  S
          rotor 1                 radius        =      37.500    0.0000     Plimit_rs     =     13774.8*   927.20  S
          rotor 2                 radius        =      37.500    0.0000     Plimit_rs     =     13774.8*   0.0000  S
              design conditions:         Preq =     28868.4     25058.6
              mission 1, segments:       Preq =     24347.6     24009.4     11777.1     11706.4     22645.9     11633.8
                                                    11566.6     11501.2
              design conditions:     Preq/Pav =      1.0002      1.0003
              mission 1, segments:   Preq/Pav =      0.9991      0.9584*     0.8161*     0.8112*     0.9040*     0.8062*
                                                     0.8015*     0.7970*
```

trace_miss

Provides information on how mission solution is progressing

- Mission trace output denoted with `M` on right side
- Often used in conjunction with *trace_size*
- Counter label provides information on which mission is being calculated
- Use `trace_case=2` to see segment identification; *trace_fly, trace_trim, trace_maxgw* to view details of segment solution
- Global values/tracing set in *Solution* quant
- Local values/tracing for a particular mission set in *SizeMission* or *OffMission* input

Solution%toler_miss *Solution%relax_miss* *Solution%relax_range* *Solution%relax_gw* *Solution%niter_miss*

```
FlyMission 1 *        658
    Mission (tolerance = 0.01000, relaxation = 1.000 (fuel) 1.000 (range) 1.000 (max TO GW), maximum iterations =  40)        M
    iteration  1                                                                                                               M
              mission fuel =   14454.4*    takeoff GW =  164590.9                                                              M
    iteration  2                                                                                                               M
              mission fuel =   13765.0*    takeoff GW =  144765.3     error ratio =    48.251       0.0000        0.0000       M
    iteration  3                                                                                                               M
              mission fuel =   13742.2*    takeoff GW =  144075.8     error ratio =    1.5924       0.0000        0.0000       M
    iteration  4                                                                                                               M
              mission fuel =   13741.6*    takeoff GW =  144053.1     error ratio =  0.42786E-01    0.0000        0.0000       M
```

trace_maxgw

Provides information on how an solution to maximize gross weight proceeds

- Can be set globally with *Solution%trace_maxgw* or for specific instance using *FltAircraft%trace_maxgw*
- Max Gross Weight trace output denoted with `W` on right side
- Counter label (*trace_case*) provides info on where in execution a solution attempt is occurring
- Use `trace_case=2` to see detailed location when gross weight maximum occurs inside a mission segment or is part of a flight condition solution.

toler_maxgw *relax_maxgw* *Solution%niter_maxgw (Global Only)* *maxderiv_maxgw*

```
FlyCondition 1 *            98
  FlyCondition_MaxGW *         101

   Maximum Gross Weight (tolerance = 0.00200, relaxation = 0.500, maximum iterations =  40, maximum derivative =    0.00    ) W
   quantity = power margin (min(Pav-Preq)=0); variable = gross weight                                                          W
                  gross weight    power margin      derivative       gain              error ratio                               W
   start            0.14288E+06     7397.3           0.0000          0.0000             0.0000                                   W
   perturb 1        0.14574E+06     6782.2          -0.21525        -2.3229             1000.0                                   W
   iteration  1     0.16149E+06     3242.3          -0.22469        -2.2252             5513.0                                   W
   iteration  2     0.16871E+06     1528.0          -0.23761        -2.1043             2524.8                                   W
…
   iteration 11     0.17488E+06     -4.7715          1.8849         -1.2629            0.62333                                   W
   Maximum Gross Weight (tolerance = 0.00200, relaxation = 0.500, maximum iterations =  40, maximum derivative =    0.00    ) W
   quantity = power margin (min(Pav-Preq)=0); variable = gross weight                                                          W
                  gross weight    power margin      derivative       gain              error ratio                               W
   start            0.14288E+06     7397.3           0.0000          0.0000             0.0000                                   W
   perturb 1        0.14574E+06     6782.2          -0.21525        -2.3229             1000.0                                   W
   perturb 2        0.14860E+06     6158.6          -0.21822        -2.2913             1000.0                                   W
   iteration  1     0.16271E+06     2954.9          -0.22703        -2.2023             4938.1                                   W
   iteration  2     0.16922E+06     1402.9          -0.23850        -2.0965             2277.4                                   W
…
   iteration 11     0.17488E+06     -4.3906         -1.9080         -1.2633            0.57412                                   W
```

`method_maxgw=1` secant solution method

`method_maxgw=2` false position solution method

trace_fly

Provides information on how a solution of maximum effort flight state proceeds

- Maximum effort trace output denoted with `E` (inner loop) `E2` (outer loop) on right side
- Set for global trace with *Solution%trace_fly*
- Specific condition trace using *FltAircraft%trace_fly*= Inner, Outer
  - *FltAircraft%trace_fly* is 2 element vector; inner and outer loop printout individually controlled
  - Different than *Solution%trace_fly*!
- Counter label (*trace_case*) provides info on where in execution a solution attempt is occurring
- Use `trace_case=2` to see detailed location of where occurs in solution procedure
  - Typically set *trace_start* to suppress conditions prior to occurrence of interest when using *Solution%trace_fly*

- Two line title provides information on solution procedure parameters & targets of maximum effort

toler_fly   *relax_fly*   *Solution%niter_fly*   *maxderiv_fly(1)*

```
 Maximum Effort (tolerance = 0.00200, relaxation = 0.500, maximum iterations =  80, maximum derivative =    0.00     )        E
   quantity = range (99% maximum V/fuelflow); variable = speed *fVel =   1.0000)                                              E
```

max_quant(1)   *max_var(1)*

- Secant & False Position solution procedure trace:

```
                 speed           V/wdot          slope             derivative       gain             error ratio               E
   start                                           0.0000            0.0000           0.0000           0.0000                  E
   perturb 1      104.00          0.14676        0.60401E-03       0.30201E-03       1655.6            0.0000                  E
   perturb 2      106.00          0.14772        0.47693E-03      -0.63542E-04      -7868.8            0.0000                  E
   iteration  1 109.75          0.14914        0.37977E-03      -0.25888E-04      -19314.            46.910                  E
```

(speed = *max_var*; V/wdot = *max_quant*)

- Golden Section Search trace:

```
                 speed           V/wdot        error ratio     x1-x0     x1       x2-x1       f1-f0      f1         f2-f1      E
 start                                           0.0000          0.00     0.00      0.00       0.00       0.00       0.00      E
 perturb 1       104.00          0.14676         0.0000          2.00     104.     -2.00      0.121E-02  0.147     -0.147      E
 perturb 2       106.00          0.14772         0.0000          2.00     104.      2.00      0.121E-02  0.147      0.954E-03 E
 search    1     108.00          0.14855         0.0000          2.00     106.      2.00      0.954E-03  0.148      0.833E-03 E
 ...
 iteration  1    124.76          0.15134         15.451          2.00     124.      0.764     0.325E-04  0.151     -0.375E-04 E
```

(speed = *max_var*; V/wdot = *max_quant*)

trace_trim

Provides information on how an solution of trim proceeds

- Trim trace output denoted with `T` on right side
- Title block shows type of trim attempted (*IDENT_trim*) and solution procedure parameters

STATE_trim *mtrim* *toler_trim* *relax_trim* *Solution%niter_trim*

```
Trim (state free , 6 variables; tolerance = 0.00100, relaxation = 0.500, maximum iterations = 40)     T
```

- Next two lines show independent variables (*trim_quant*) and dependant quantities *(trim_var)* of trim matrix

```
variables  = coll      latcyc    lngcyc    pedal      pitch      roll        T
quantities = force x   force y   force z   moment x   moment y   moment z    T
```

- Two levels of tracing available:
  - 1: Trim variable values and targets displayed along with derivative matrix
  - 2: Show all control positions for each component at every perturbation/iteration

"Pilot" control values [independent variables] (e.g. *coll*, *pedal*)

Trim quantities [dependent variables]

Component Controls

Trace_trim=2

```
start         5.00  0.00  2.00  0.00  0.00  0.00   -801.      -651.     0.279E+04 -0.523E+04  0.431E+05  0.303E+05 T
controls: rotor 1     coll =  0.8333E-01 lngcyc  =  2.00 latcyc  =  0.00  incid =   0.00  cant =   0.00         T
controls: rotor 2     coll =  0.000      lngcyc  =  0.00 latcyc  =  0.00  incid =   0.00  cant =   0.00         T
controls: tail 1      cont =  0.00                                        incid =  -2.50                        T
controls: tail 2      cont =  0.00                                        incid =   0.00                        T
controls: prop 1 engn 1                                                   incid =   0.00  yaw  =   0.00         T
perturb  1    6.00  0.00  2.00  0.00  0.00   0.00   -323.      -949.    -306.     -0.740E+04  0.514E+05  0.395E+05 T
. . .
derivative matrix                                                                                               T
     row 1 =   478.       10.3       366.      -2.33      -506.      0.251                                      T
     row 2 =  -298.       277.      -17.6      -286.       0.166     314.                                       T
     row 3 = -0.310E+04  0.558      19.3       101.      -0.101E+04  -2.59                                      T
     row 4 = -0.217E+04  0.382E+04  -251.     -0.136E+04   167.      -1.74                                      T
     row 5 =  0.831E+04  -58.6     -0.416E+04  0.300E+04 -0.234E+05  -1.69                                      T
     row 6 =  0.929E+04   867.      0.289E+04  0.856E+04 -0.283E+04   1.04                                      T
iteration  1  5.24  0.05  3.40  -2.35  0.45  -1.35   -353.     -486.     0.173E+04 -0.267E+04  0.310E+05  0.153E+05 T
. . .
```

"Pilot" Controls

Trim quantities

"Pilot" control values [independent variables]

Trim quantities [dependent variables]

trace_rotor

Provides information on how an individual rotor flap solution proceeds

- Rotor trace output denoted with R on right side
- Rotor flapping solved for each rotor, each trim perturbation/iteration step

```
Rotor 1 equations (tolerance = 0.01000, relaxation = 0.500, maximum iterations =  40) CT/s =  0.10000                R
perturb  0    t75,tc,ts =  11.56   3.56  -9.66   b0,bc,bs =   3.66   2.00   0.00  Eqt,Eqc,Eqs =   1.438   0.349   0.194  R
perturb  1    t75,tc,ts =  12.56   3.56  -9.66   b0,bc,bs =   3.66   2.00   0.00  Eqt,Eqc,Eqs =   0.301   0.344  -0.723  R
perturb  2    t75,tc,ts =  11.56   4.56  -9.66   b0,bc,bs =   3.66   2.00   0.00  Eqt,Eqc,Eqs =   1.435  -0.667   0.194  R
perturb  3    t75,tc,ts =  11.56   3.56  -8.66   b0,bc,bs =   3.66   2.00   0.00  Eqt,Eqc,Eqs =   0.910   0.350  -0.941  R
derivative matrix columns = (   1.14     0.01     0.92) / (   0.00     1.02     0.00) / (   0.53     0.00     1.13)      R
gain matrix columns       = (   0.70     0.00    -0.57) / (   0.00     0.49     0.00) / (  -0.33     0.00     0.71)      R
iteration  1  t75,tc,ts =  12.51   3.72 -10.34   b0,bc,bs =   4.56   2.00   0.00  Eqt,Eqc,Eqs =   1.438   0.349   0.194  R
. . .
rotor 1 *** not converged ***
```

Rotor convergence failure message (printed regardless of *trace_rotor* value)

Solution Tracing Best Practices

Effective solution debugging requires methodical approach

- **Work from outer loop to inner loop**
  - –Selectively turn on tracing
  - –Problem is often result of conditions set in loop at higher level than where failure occurs
  - –Warning printed of inner loop failures at each step

- **Use *trace_start* to keep output file size manageable**
  - –*trace_case*=2 is useful in zeroing in on specific condition in mission
  - –Alternatively can set trace variables locally in input namelist

- **Modify solution procedure locally**
  - –Improve first "guess" as contained in input (pilot controls; speeds, etc)
    - › Avoid 0 values for pilot controls unless expected final trim value
    - › Use smallest trim DOF as practical for condition (i.e. 3 DOF in Fwd Flt)
  - –Modification of relaxation factor often fixes trim for otherwise well formulated conditions; *deriv_trim=2* often helps
  - –If inputs other than solution procedure parameters modified to get successful solution, reset parameters to defaults and determine if trim is well behaved

Solution Tracing Best Practices

Running simplified conditions can be helpful

- Power sweep to bracket Vmax or Vbr calculation
- Mission profiles with input speeds
- Fewer performance point sizing conditions & design missions in sizing procedure
- Single loop maximum effort instead of double loop
- Ignore some mission segments

If size or mission loop diverges, reduce number of iterations

- Set *niter_size* /*niter_param* or *niter_miss* to last iteration that worked
  - –So can examine full output for conditions and mission segments

Often try several solution parameter changes in search for converged solution

- When convergence achieved, re-consider whether need all these changes

Tutorial — Trace and Convergence

hel-db0: **comment out solution parameters in** *helicopter.njob*, **run with just defaults**

- *Solution*: *perturb_trim=.01,deriv_trim=2*
- *PerfCondition 2*: *relax_fly=.2*
- **overflow at FlyCondition 2 (***trace_case=1*

hel-db1: **trace solution**

- *Solution*: *trace_case=2*
- **mission 1 converges, condition 1 converges; condition 2 (hover ceiling) diverges**

hel-db2: **trace max effort iteration**

- *Solution*: *trace_fly=1*
- **max effort stops after perturbation**

hel-db3: **trace trim iteration**

- *Solution*: *trace_trim=1* **— lot of output**
- *PerfCondition 2*: *trace_trim=1*
- **trim diverges in first max effort step after perturbation**

hel-db4: **max effort relaxation factor**

- *PerfCondition 2*: *relax_fly=.2* **(default** *.5***)**
- **case executes without overflow**
- **"Case Convergence" indicates trim not converged for condition 4 (Vbr)**

hel-db5: **trim derivative**

- *Solution*: *deriv_trim=2*
- **condition 4 still diverge**

hel-db6: **trim derivative**

- *Solution*: *perturb_trim=.01* **(default** *.002***)**
- **case converged ("Case Convergence")**

helicopter.njob:

- **remove** *trace_fly*, *trace_trim*; *trace_case=0* **or** *1*
- **keep** *trace_size* **(size job) or** *trace_miss* **(no sizing) to observe convergence of outermost loop**

Outline

- Introduction
- Documentation
- Overview
  - Tasks
  - Aircraft
- NDARC Job
  - Input
  - Organization
  - Output
- Solution Procedures
  - Debugging
- Input Manual
  - Aircraft
  - Tasks
- Tutorial

Aircraft Synthesis

Aircraft consists of set of components

- **Aircraft, Fuselage, Landing Gear, Systems, Cost**
- **Rotors**
  - main rotor, tail rotor, propeller
  - tilting, ducted, antitorque, auxiliary-thrust, variable diameter, reaction drive
  - twin rotors
- **Wings**
- **Tails**
  - horizontal or vertical
- **Fuel tanks**
  - fuel quantity measured as either weight or energy
- **Propulsion groups**
  - set of rotors and engine groups, connected by drive system
  - components define power required, engine groups define power available
- **Engine Groups (turboshaft, compressor, electric motor, generator)**
  - transfers power by shaft torque
  - one or more engines of same type
- **Jet Groups (turbojet, turbofan)**
  - produces force on aircraft
- **Charge Groups (fuel cell, solar cell)**
  - Generates energy for the aircraft

Input Manual describes parameters that define each of these components

Aircraft — Configuration

Conventional configurations can be rapidly modeled in NDARC

- *Aircraft%config = 'helicopter', 'tandem', 'coaxial', 'tiltrotor'*
- Code can automatically configure input parameters to have expected behavior
  - –Default set of components
  - –Default controls and trim approach
- Typically synthesize new designs by modification to relevant default configuration

Configuration-dependent initialization (ch.2 of manual)

```
!==============================================================================
! default helicopter
&DEFN quant='Aircraft',&END
&VALUE config='helicopter',&END
&DEFN quant='Rotor 1',&END
&VALUE rotate=1,&END
&DEFN action='configuration',&END
!==============================================================================
```

twin main rotors: set *rotate* for each

tandem rotors: set *overlap_tandem*

first rotor = main, front, lower, right rotor

Default Configurations

Documented in Input Manual (ch.2)

Number of default components:

- *nRotor=2, nWing=0, nTail=2, nPropulsion=1, nEngineGroup=1, nEngineModel=1,*

Seven default aircraft controls (except tiltrotor):

- *IDENT_control='coll','latcyc','lngcyc','pedal','tailinc','elevator','rudder'*
- **Connected to rotor and tail controls as appropriate for configuration**
- **Tiltrotor: add** *'tilt', 'flap', 'flaperon', 'aileron'* **controls**
  - **2 control states (helicopter and airplane)**

Nine trim states (trim schemes):

	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'

Aircraft — Controls

Aircraft controls: *ncontrol*, *IDENT_control(ncontrol)*

Number of control states: *nstate_control*

Control values as function of speed: *nVcont*, *cont(nVcont)*, *Vcont(nVcont)*

- **Select use for each condition and segment:**
  - *FltState%SET_control(ncontrol) = 0* use *Aircraft%cont*
  - *FltState%SET_control(ncontrol) = 1* use *FltState%control* (default)

---

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 Controls

Control definition key feature for configuration generality

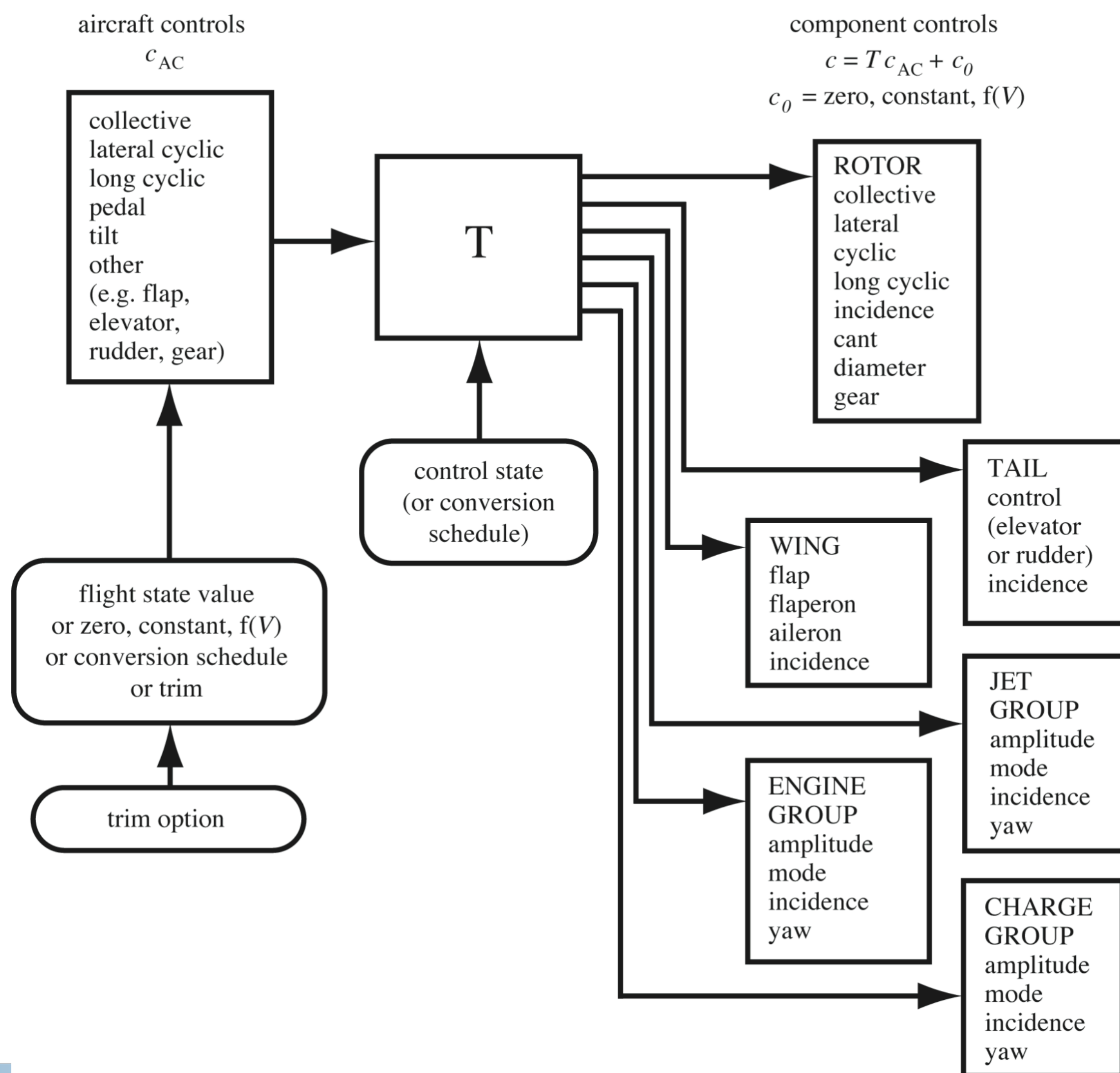
Aircraft controls connected to component controls

Aircraft controls include:

- **Pilot's controls**
- **Configuration variables (e.g. tilt of nacelle/pylon, engine, rotor shaft)**
- **Connections to component controls**

Only pilot's controls set / adjusted for flight condition or mission segment

- **Access to component controls only through matrix T**



Aircraft

Aircraft motion

- **Pitch and roll motion as function of speed**
- **Select use for each condition and segment:** *Aircraft* **or** *FltState*
  - *FltState%SET_pitch*, *FltState%SET_roll*

Conversion: schedule as function of speed

- **Tilt, control state, drive system state, tip speed**
- **Select use for each condition and segment:** *Aircraft* **or** *FltState*
  - *FltState%SET_tilt*, *FltState%STATE_control*, *FltState%STATE_gear*, *FltState%SET_Vtip*

Velocity schedules: *SET_Vschedule = 1* **CAS,** *2* **TAS**

Aircraft — Trim

Number of trim states: *nstate_trim*

Trim state definition:

- **Label:** *IDENT_trim(nstate_trim)*
  - *FltState%STATE_trim* **identifies trim method by label**
- **Number of trim variables:** *mtrim(nstate_trim)*
- **Trim quantities:** *trim_quant(mtrim,nstate_trim)*
- **Trim variables:** *trim_var(mtrim,nstate_trim)*
- **Target source:** *trim_target (mtrim,nstate_trim)* **=** *FltState* **or component**

Aircraft — Geometry

Fixed or scaled input

- *INPUT_geom=1*: **fixed, SL / BL / WL**
- *INPUT_geom=2*: **scaled, x/L / y/L / z/L**

Reference length for scaled geometry: *KIND_scale*, *kScale*

Reference point: *KIND_Ref*, *kRef*, *SL_Ref*, *BL_Ref*, *WL_Ref*

Geometry

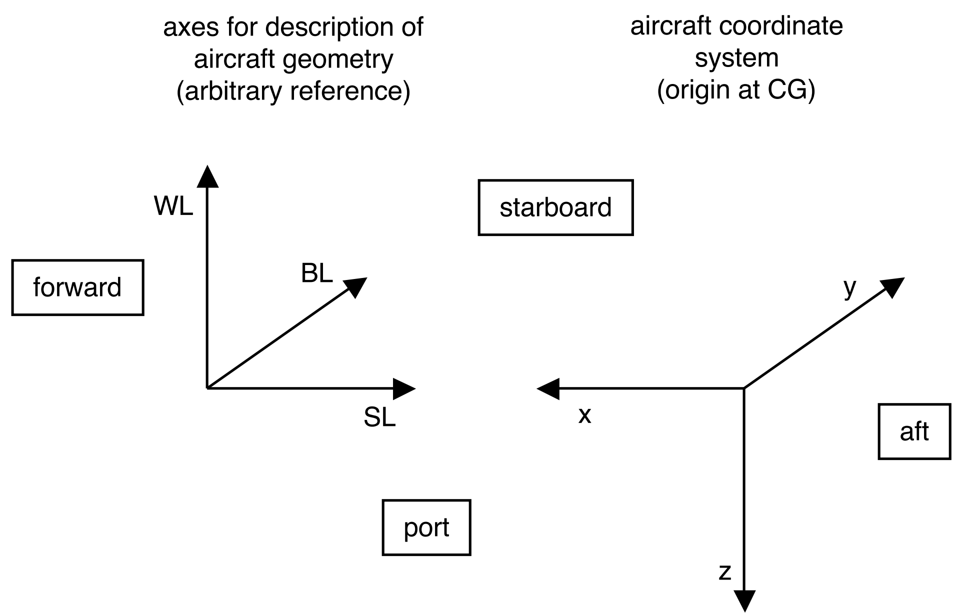
Component position input: fixed or scaled

Fixed:

- Station line, butt line, water line (SL, BL, WL)

Scaled

- x/L, y/L, z/L
- reference length L = rotor radius, wing span, or fuselage length
- relative reference point = input, rotor, wing, fuselage, or center of gravity



Location

Aircraft: **define how geometry is input**

- *INPUT_geom*: **Fixed (***SL, BL, WL***) or scaled (***XoL, YoL, ZoL***)**
- *KIND_scale*: **Global reference length for scaled geometry**
  –rotor radius (*kScale*), wing span (*kScale*), or fuselage length
- *KIND_Ref*: **Location reference point set**
  –input SL/WL/BL, rotor center, wing, fuselage, or center-of-gravity

Component or *Geometry***:**

- **Input either**
  loc_zzz%SL, loc_zzz%BL, loc_zzz%WL
- **or**
  loc_zzz%XoL, loc_zzz%YoL, loc_zzz%ZoL

- **Option to fix some geometry (***loc_zzz%FIX_geom* **override** *INPUT_geom***)**
- **Option to specifiy reference length (***loc_zzz% KIND_scale* **override** *KIND_scale***)**

Aircraft

Takeoff flight condition: *SET_Atmos*
- *Temp, dtemp, density, csound, viscosity, altitude*

Weight
- Design gross weight, structural design gross weight, maximum takeoff weight
- Weight empty
- Moments of inertia

Drag
- **Total aircraft drag:** *FIX_drag*
- **Total aircraft download:** *FIX_DL*

Number of components: set for default configuration
- *nRotor, nWing, nTail*
- *nTank, nPropulsion, nEngineGroup, nJetGroup, nChargeGroup*
- *nEngineModel, nEngineTable, nCompressorModel, nMotorModel*
- *nJetModel*
- *nFuelCellModel, nSolarCellModel*
- *nBatteryModel*

Cost

Inflation factors: *MODEL_inf, year_infl, inflation, EXTRAP_inf*

- **Input inflation always used**
  - **Default** *inflation*=**100.%, with CPI or DoD for** *year_infl*
- **DoD: deflators for Total Obligational Authority and Procurement**
- **Consumer price index: all urban consumers, U.S city average, all items**

CTM (Harris and Scully) Rotorcraft Cost Model

- **Aircraft purchase price**
- **Maintenance cost**
- **Direct operating cost**

Systems

Weight information

- **Payload, fixed useful load**
- **Folding**
  - –Optionally fraction of fold weights in kit, which can be removed when not required
- **Vibration treatment**
- **Contingency**

Systems and equipment

Weight models

- **Flight control group**
  - –non-boosted controls: do not see aerodynamic surface or rotor loads
  - –boost mechanisms: actuators
  - –boosted controls: affected by aerodynamic surface or rotor loads
- **Anti-icing group**

Fuselage

Geometry

- **Location (*loc_fuselage*)**
- **Length, width, wetted area, cabin area, reference length**
- **Geometry for graphics**

Aerodynamics

- **Model, contingency drag**

Weight

Aerodynamics and drag model

Weight models

LandingGear

Geometry

- **Location (*loc_gear*)**
- **Height rotor above ground**
- **Retraction**

Aerodynamics

Weight

Drag model

Weight models

Rotor

One or more rotors (or none)
- Designated main, tail, or propeller: weight model, where in weight statement
- Designated antitorque or auxiliary-thrust: special sizing options
- Other configuration features: tilting, ducted, variable diameter, reaction drive

Connected to propulsion group (drive train)
- Set tip speed, drive losses (even if no shaft power source)

Energy method for power: induced + profile + parasite
- In terms of induced power factor and mean drag coefficient
- Including induced power for twin rotors

Inplane forces relative TPP: calculate with blade element theory, or neglect
Profile inplane forces: calculate with blade element theory, or simplified

Rotor interference at other components: fuselage, wings, tails
- Wake-induced velocity at component estimated based on inflow at rotor

Rotor drag (hub, pylon, spinner)

Rotor

Configuration

- **Principal designation (main, tail, prop): rotor weight in weight statement**
- **Antitorque, auxiliary thrust: special options for sizing**
- **Twin rotors (coaxial, tandem, tiltrotor): special options for geometry and performance**
- **Variable diameter, ducted fan**
- **Reaction drive**

Propulsion group: identified by *kPropulsion* **(which can have several drive states)**

- **Drive system branch:** *KIND_xmsn=1* **primary,** *0* **dependent**
- **Primary: specify reference tip speed** *Vtip_ref*, **and default tip speeds**
- **Dependent: specify gear ratios, or calculate from** *Vtip_ref* **of this and primary rotor**

Reaction drive

Default rotor tip speeds: only for primary rotor; select by *FltState%SET_Vtip*

Drive system torque limit

Parameters: use depends on *Size%SET_rotor*

- **Rotor disk loading T/A (***diskload***), from** *fDGW****DGW or** *fThrust*** *Tdesign*
- **Aircraft disk loading from** *fArea*
- **Radius, CWs =** C_W/σ**, sigma =** σ
- *Tdesign, Pdesign, Ndesign* **for antitorque or aux thrust rotor**

Rotor

Geometry

- **Position** *SET_geom*, **twin rotor parameters, tail rotor, variable geometry**
- **Direction of rotation (***rotate=1* **CCW,** *-1* **CW), number of blades**
- **Planform and twist**
- **Flap dynamics**
- **Aerodynamics**
- **Blockage factor**

Geometry for graphics

Blade element theory solution

Geometry

- **Locations:** *loc_rotor, loc_pylon*; *loc_pivot, loc_naccg* **with shaft tilt**
- **Nominal orientation:** ±x, ±y, ±z, main (-z), tail (ry), prop (x)
- **Shaft control (***KIND_tilt***): fixed shaft, or incidence and/or cant control**
- **Orientation of rotor shaft**
- **Orientation of pivot axis**

Controls

- **Rotor control mode:** *KIND_control, KIND_cyclic, KIND_coll*
- **Collective, longitudinal cyclic, lateral cyclic, incidence, cant, diameter, gear ratio**
  - **Parameters** *INPUT_zzz, T_zzz, nVzzz, zzz, Vzzz*

Trim targets
Rotor thrust capability

Rotor

Performance

- **Power model:** *MODEL_perf=1* **standard,** *2* **table**
- **Inplane forces:** *MODEL_Ftpp, MODEL_Fpro*

Interference

- **Model, transition**

Geometry

- **Hub/pylon aerodynamic axes**
- **Pylon wetted area, duct area, spinner area**

Drag

Weight

- **Model, increments**
- **Blade moment of inertia**
- **Technology factors**

Rotor induced and profile power models, table model
Drag model
Interference model
Weight models

Wing

Geometry defined in terms of wing panels

- Symmetric
- Each panel has straight aerodynamic center and linear taper
  - Sweep, dihedral, offsets of aerodynamic center
- Set of outboard panels can be considered wing extension

Controls: flap, flaperon, aileron, incidence

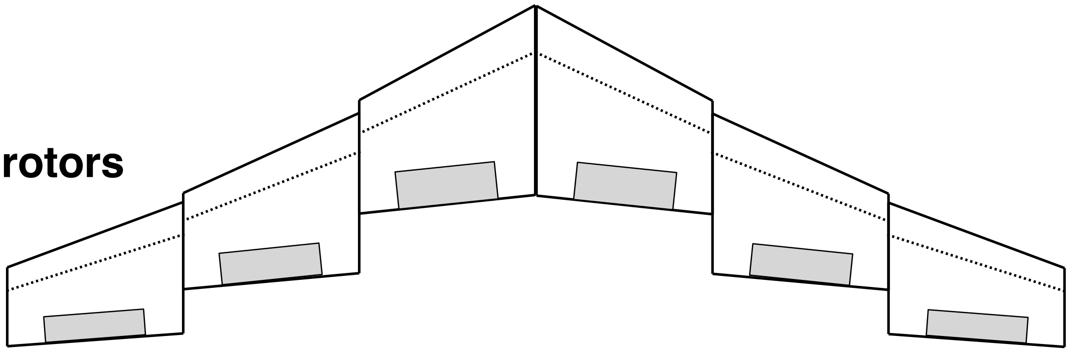
- Controls for each panel
- Flaperon and aileron are same surface

Wing interference on other wings (biplane or tandem)

Wing interference on tail

Wing interference on rotors

Induced-drag interference from rotors



Wing

Geometry: Use depends on *Size%SET_wing*

- **Wing loading W/S (***wingload***), from** *fDGW****DGW**
- **Area, span, chord, aspect ratio**

Geometry

- **Rotors, span calculation, thickness, torque box**

Geometry for graphics

Geometry

- **Location (loc_wing)**
- **Number of panels**

Wing panels

Wing extensions

Wing kit

Controls (each panel)

- **Kind:** *KIND_flap, KIND_aileron, KIND_incid, KIND_flaperon*
- **Flap, flaperon, aileron, incidence**
  - **Parameters** *INPUT_zzz, T_zzz, nVzzz, zzz, Vzzz*

Wing

Trim targets

Aerodynamics
Weight

Aerodynamics and drag model
Weight models
Tiltrotor wing weight model

Tail

Kind: *KIND_tail=1* **horizontal,** *2* **vertical**

Geometry: use depends on *SET_tail*

- **Area, span, chord, aspect ratio, tail volume**

Geometry for graphics

Geometry

- **location (***loc_tail***)**
- **Cant angle, control surface**

Controls

- **Elevator or rudder, incidence**
    - **Parameters** *INPUT_zzz, T_zzz, nVzzz, zzz, Vzzz*

Aerodynamics

Weight

Aerodynamics and drag model

Weight models

FuelTank

Each system consists of main tank(s) and auxiliary tank(s)

- Engines, jets, chargers associated with a fuel tank system
- Fuel container has weight
- Fuel quantity stored and burned is measured in weight or energy

Weight changes as fuel used

- Jet fuel, gasoline, diesel, hydrogen
- Characteristics: density (lb/gal or kg/liter), specific energy (MJ/kg), tank weight

Energy changes as fuel used, weight does not change

- Battery, flywheel, capacitor
- Characteristics: tank density (MJ/liter), tank specific energy (MJ/kg)

Battery model

- Characteristics: efficiency (varies with power, state-of-charge), power density (kW/kg)

FuelTank

Configuration

- **Fuel quantity stored and used:** *SET_burn=1* **weight,** *2* **energy**
- **Weight properties:** *fuel_density*, *specific_energy*
- **Sizing: fuel capacity (***Wfuel_cap* **or** *Efuel_cap***),** *fFuel_cap*
- **Battery identification:** *IDENT_battery* **points to BatteryModel**

Geometry for graphics

Auxiliary fuel tank

- **Number tank sizes**
- **Capacity, tank weight, drag**
- **Location (***loc_auxtank***)**

Weight

Weight models

- **Weight storage: tank, plumbing**
- **Energy storage: tank weight and volume density**

Propulsion

Propulsion Group

- Set of rotors and engine groups, connected by drive system
  - One or more drive states, with different gear ratios
  - Tip speed: input, reference, function speed or conversion schedule, or various defaults
- Power required = component power + transmission losses + accessory losses
- Drive system limit (torque), rotor and engine shaft limits
- Drive system weight

Engine Group

- Each engine group has one or more engines of same type
- Performance at required power: mass flow, fuel flow, jet thrust, momentum drag
- Controls: yaw, incidence
- Drag, weight

Referred Parameter Turboshaft Engine Model

- Enables aircraft performance analysis to cover entire spectrum of operation
  - Curve fits of referred performance from engine deck, including effect of turbine speed
- Effects of size (scaling model, based on mass flow) and technology (specific power and specific fuel consumption)

Propulsion

Propulsion Groups

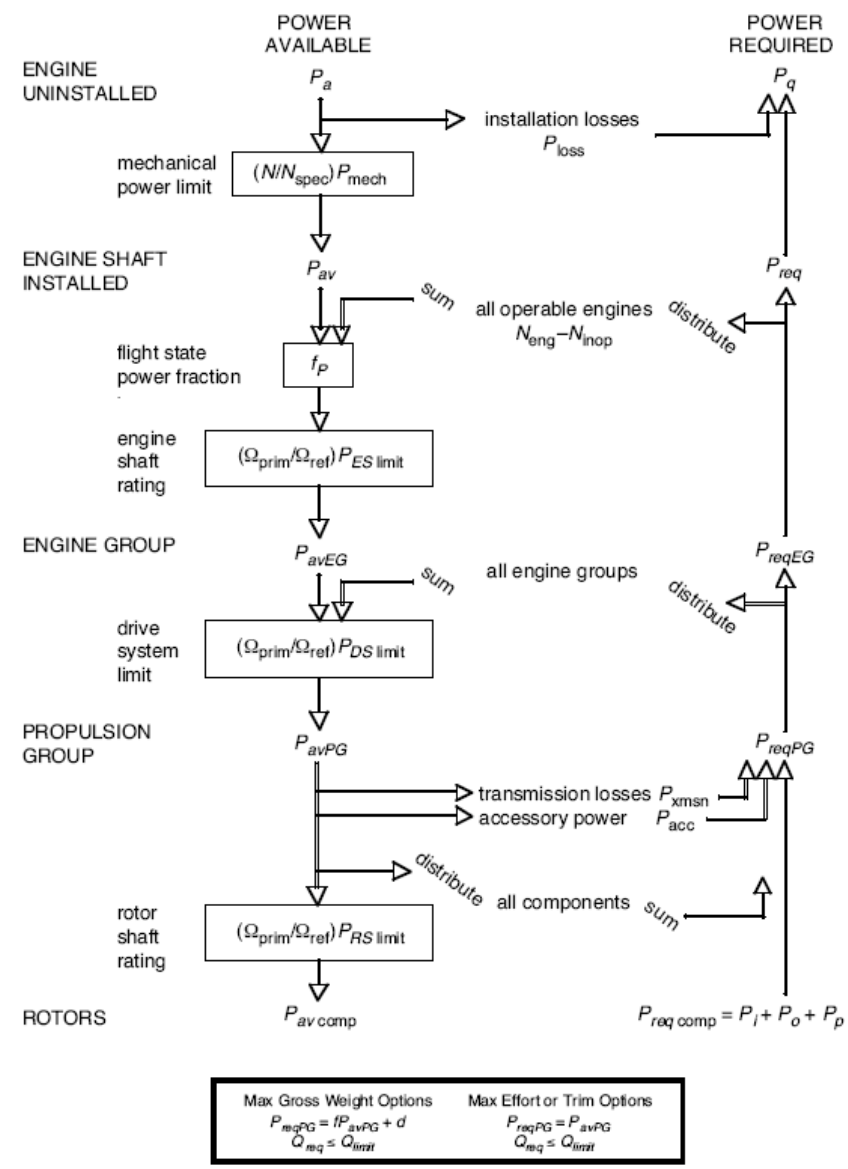
- Set of rotors and engine groups, connected by drive system
  - One or more **drive states**, with different gear ratios
- Power required = $P_{comp}+P_{xmsn}+P_{acc}$
- Drive system limit (torque), rotor and engine shaft limits

Engine Group

- Each engine group has one or more engines of same type
- Performance: mass flow, fuel flow, jet thrust, momentum drag
- Controls: yaw, incidence

Referred Parameter Turboshaft Engine Model

- Enables aircraft performance analysis to cover entire spectrum of operation
  - Curve fits of referred performance from engine deck, including effect of turbine speed
- Effects of size (scaling model) and technology (specific power and sfc)



Original Propulsion Representation

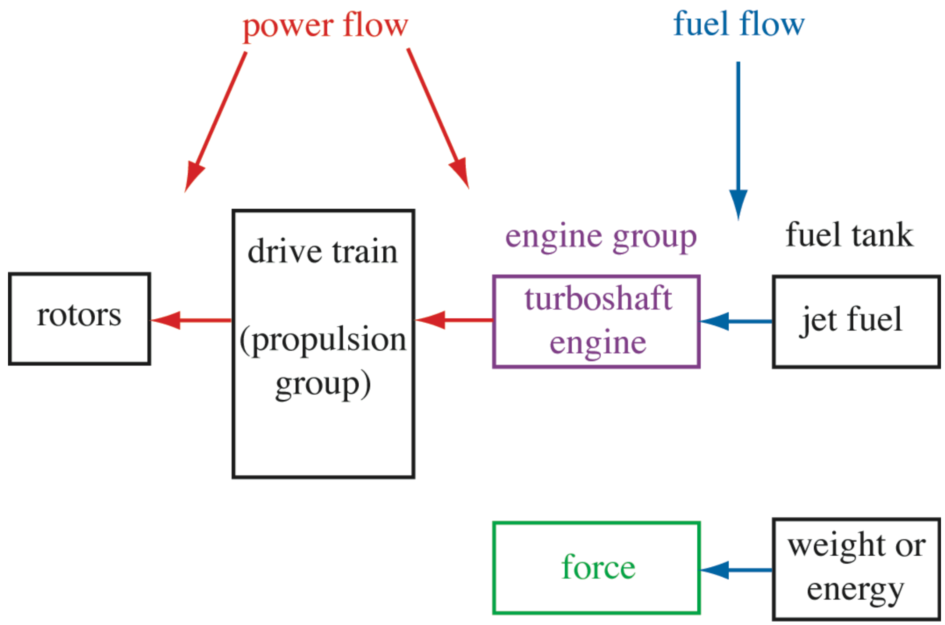
Mechanical drive train, connecting engine groups and rotors

Engine group, consisting of one or more turboshaft engines

- **Referred Parameter Turboshaft Engine Model**

Fuel tank system (main and aux tanks)

- **Weight changes as fuel used, fuel is measured in weight**



Force generation by simple model

- **Fuel used is measured as weight or energy**

Connecting components of propulsion system

rotor-drive-turboshaft-fuel

```
!==============================================================================
&DEFN quant='Aircraft',&END
&VALUE nTank=1,nPropulsion=1,nEngineGroup=1,nEngineModel=1,&END  ! defaults
!------------------------------------------------------------------------------
&DEFN quant='Rotor 1',&END
&VALUE kPropulsion=1,&END                                        ! default
!------------------------------------------------------------------------------
&DEFN quant='FuelTank',&END
&VALUE &END
!------------------------------------------------------------------------------
&DEFN quant='Propulsion',&END
&VALUE &END
!------------------------------------------------------------------------------
&DEFN quant='EngineGroup',&END
&VALUE
    MODEL_engine='RPTEM',                                        ! default
    IDENT_engine='enginemodel',
    kPropulsion=1,kFuelTank=1,                                   ! default
&END
!------------------------------------------------------------------------------
&DEFN quant='EngineModel',&END
&VALUE ident='enginemodel',&END
!==============================================================================
```

connect to drive

turboshaft engine
identify model
connect to drive and fuel tank

rotor-drive-motor-battery (using defaults)

```
!==============================================================================
&DEFN quant='Aircraft',&END
&VALUE nEngineModel=0,nMotorModel=1,nBatteryModel=1,&END
!------------------------------------------------------------------------------
&DEFN quant='FuelTank',&END
&VALUE SET_burn=2,IDENT_battery='battmodel',&END
!------------------------------------------------------------------------------
&DEFN quant='EngineGroup',&END
&VALUE
    MODEL_engine='motor',
    IDENT_engine='motormodel',
&END
!------------------------------------------------------------------------------
&DEFN quant='MotorModel',&END
&VALUE ident='motormodel',&END
!------------------------------------------------------------------------------
&DEFN quant='BatteryModel',&END
&VALUE ident='battmodel',&END
!==============================================================================
```

number of components

burn energy, connect to battery model

electric model

identify model

Propulsion

Drive system

- **Number of states**

Transmission losses

Accessory losses

Geometry: drive shaft length

Drive system torque limit

Drive system ratings

Weight

Weight models

EngineGroup

Description

- *MODEL_engine*
- *IDENT_engine*, *IDENT_system2*: **point to EngineModel, EngineTable, CompressorModel, or MotorModel**
- **Number of engines:** *nEngine*
- **Power:** *Peng*, *rating_to*
- *kFueltank*, *kRotor_react*

Propulsion group

- *kPropulsion*: **point to propulsion group**
- **Drive system branch (no rotors):** *KIND_xmsn=1* **primary,** *0* **dependent**
- **Gear ratio**

Sizing: distribute power required among engine groups (*Size%SIZE_perf*)

Drive system torque limit

EngineGroup

Installation

- **Deterioration factor on fuel flow or performance:** *Kffd*
- **Aerodynamic: efficiency and losses, auxiliary air momentum drag**
- **Electric: efficiency**

IR suppressor

Convertible

Geometry

- **Location (***loc_engine***)**
- **Nominal orientation, position**
- **Nacelle/cowling wetted area**

Controls

- **Amplitude, mode; incidence, yaw; gear ratio**
  - **Parameters** *INPUT_zzz, T_zzz, nVzzz, zzz, Vzzz*

Nacelle drag

Weight

Drag model

Weight models

- **Engine section or nacelle group, air induction group**
- **Engine system, exhaust and accessories**

EngineModel

Identification

- *ident*: **match** *IDENT_engine* **of EngineGroup**

Weight

Engine ratings

Reference

- **Power, specific power, mechanical limit**
- **Specific fuel consumption, specific jet thrust**
- **Turbine speed**

Technology

Scaling

Optimum power turbine speed

Power available and power required parameters

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

- Debugging

Input Manual

- Aircraft
- Tasks

Tutorial

NDARC Job

Job consists of tasks and requirements

- **Job, Cases, Solution input**

Tasks

- **Size** aircraft for design conditions and missions
- **OffDesign**: mission analysis
- **Performance**: flight performance analysis

Requirements

- **FltCond**: flight condition (size and performance)
- **Mission**: mission analysis (size and off-design)
  - **MissSeg**: mission segment
- **FltState** (or FltAircraft): for each flight condition and mission segment

Input Manual describes parameters for each of these input blocks

Job Input

Start of NDARC input, only read once

Default values always used to start first case (before input read)

Set initialization of parameters from previous case

- **Inherit only input (default)**

  INIT_input=1,INIT_data=0
- **Inherit design and solution: all parameters (input, input-modified, derived)**

  INIT_input=2,INIT_data=2

File write behavior

- **Default:** *Open_status=2*,
  - **STATUS='NEW' behavior in FORTRAN**
  - **On some platforms will cause NDARC to exit with error if file already exists**
- **For Windows machines use** *Open_status=1*
  - **STATUS='REPLACE' behavior**
  - **Automatically overwrites existing file**

Cases Input

Description

- **Title, subtitles, notes, identification**
- **Analysis generates time-date identification string**

Tasks

- **Size aircraft for design condition:** *TASK_Size*
  - –Sizing task not required if aircraft or solution file read-in
  - –Variables set in *Size* input structure — required even for fixed design
- **Mission analysis:** *TASK_Mission*
  - –Number of missions in *OffDesign* input structure
- **Flight performance analysis:** *TASK_perf*
  - –Number of flight conditions in *Performance* input structure
- **Map of engine performance or airframe aerodynamics**

Cases Input

Write input parameters

- **Default: only first case; including tech factors and geometry**

Output

- **Select files:** *OUT_zzzz* **(0 for none), file name** *FILE_zzzz*
  - **Design and performance files: same information as standard output**
    - Tab delimited, full precision numbers
  - **Geometry file: for graphics**
  - **Sketch file: to check geometry and solution (DXF format)**
  - **Aircraft and solution files: read by subsequent job**
- **Formats:** *WRITE_zzzz*
  - *WRITE_wt_long=0* omit zero lines in weight statement

Gravity: standard or input

Units

- **Analysis units:** *Units=1* **English,** *2* **SI**

Solution

All iterations: *niter_zzzz, toler_zzzz, relax_zzzz*

Rotor

Trim

- **Derivative:** *deriv_trim=1* **first order (default),** *2* **second order**

Maximum effort

- **Method:** *method_fly* **(default secant)**
- **Method for maximization:** *method_flymax* **(default golden section)**
  - Can use curve fit for difficult V-best-range convergence
- **Maximum derivative amplitude:** *maxderiv_fly*
- **Maximum increment fraction:** *maxinc_fly*

Maximum gross weight (flight condition or mission takeoff)

Mission

- **Relaxation:** *relax_miss*, *relax_range* **(range credit),** *relax_gw* **(max GW)**

Size aircraft

- **Number of iterations:** *niter_size*, *niter_param*
- **Relaxation: size (power or radius), DGW, drive system, WMTO and SDGW, fuel tank, design rotor thrust**

Trace: *trace_rotor, trace_trim, trace_fly, trace_maxgw, trace_miss, trace_size*

Case trace: *trace_case, trace_start*

Size

Sizing method

- **Performance:** *SIZE_perf*
  - **Engine power (***SIZE_perf='engine'***) or rotor radius (***SIZE_perf='rotor'***) from flight conditions and mission segments (if** *DESIGN_engine=1, SIZEengine=1***)**
    - › Ignore conditions and segments with zero power margin (max GW, max effort, or trim)
    - › *SIZE_param=1* to force parameter iteration when power not sized
- **Rotor:** *SET_rotor*
- **Wing:** *SET_wing*
- **Weight:** *FIX_DGW, FIX_WE, SET_SDGW, SET_WMTO*
- **Fuel tank capacity:** *SET_tank*
- **Drive system torque limit:** *SET_limit_ds*

Number of sizing flight conditions: *nFltCond* **(maximum** *nfltmax = 21***)**

- **Input one condition (***FltCond* **and** *FltState* **variables) in** *SizeCondition* **namelist**

Number of design missions: *nMission* **(maximum** *nmissmax = 20* **)**

- **Input one mission (***MissParam***,** *MissSeg***, and** *FltState* **variables) in** *SizeMission* **namelist**

Size

Fixed aircraft: input aircraft description, perhaps as aircraft file from previous job

- **turn off sizing:** *Cases%TASK_size=0*
- **fix aircraft:**
  - *SIZE_perf='none', SET_rotor=2*'radius+Vtip+sigma'*
  - *FIX_DGW=1, SET_SDGW='input', SET_WMTO='input'*
  - *SET_tank='input', SET_limit_ds='input'*
- **perhaps** *Rotor%SET_limit_rs=0, EngineGroup%SET_limit_es=0*
- **with wing panels:** *SET_wing='WL+panel', Wing%SET_panel='width+taper','span+taper'*

Flight State (FltState)

Fundamental operating state description

- **Defined for each mission segment and performance condition**
- **Input with** *SizeMission, SizeCondition, OffMission, PerfCondition*
- **Input values part of** *FltAircraft* **structure**

Maximum effort: one or two parameters maximized; on fixed operating state

- **One parameter:** *SET_max=1,max_quant=‘zzz’,max_var=‘zzz’*
- **Two parameters:** *SET_max=2,max_quant=‘zz1’,’zz2’,max_var=‘zz1’,’zz2’* **(first is inner loop)**

Examples

- **maximum speed:** *SET_max=1,max_quant='Pmarg',max_var='speed'*
- **best range speed (high):** *SET_max=1,max_quant='range',max_var='speed'*
- **best endurance speed:** *SET_max=1,max_quant='end',max_var='speed'*
- **max climb rate:** *SET_max=1,max_quant='Pmarg',max_var='ROC'*
- **best climb:** *SET_max=2,max_quant='Pmarg','climb',max_var='ROC','speed'*
- **ceiling:** *SET_max=1,max_quant='Pmarg',max_var='alt’* (ROC=100 for service ceiling)
- **ceiling:** *SET_max=2,max_quant='Pmarg','alt',max_var='alt','speed'*

Flight State (FltState)

Flight speed: *SET_vel*

- **horizontal velocity (***Vkts* **or** *Mach***), ROC, climb angle, sideslip angle**

Aircraft motion: pitch and roll; pullup, linear acceleration

Altitude

Atmosphere: *SET_atmos***;** *temp, dtemp, density, csound, viscosity*

- **Standard:** *SET_atmos='std',altitude=zzz.*
- **4k/95:** *SET_atmos='temp',altitude=4000.,temp=95.*

Ground effect: *SET_GE***,** *HAGL*

- **Height rotor =** *HAGL* **+ (***WL_hub* **-** *WL_gear* **+** *d_gear***)**

Landing gear state: *STATE_LG*

Aircraft control: *STATE_control* (0 use conversion schedule)

- **Specification:** *SET_control=0* **Aircraft or** *1* **FltState**
- **Control value:** *control*

Flight State (FltState)

Center of gravity

Each propulsion group

- **Rotor tip speed:** *SET_Vtip*
- **Drive system state:** *STATE_gear* **(select gear ratios)**
- **Drive system rating and limits:** *rating_ds, SET_plimit, SET_Qlimit*
- **Deice system state:** *STATE_deice*
- **Accessory power increment:** *dPacc*

Each engine group

- **Engine rating:** *rating*
  - **Match** *EngineModel%rating*, **typically MCP, IRP, MRP**
- **Fraction of rated power available:** *fPower*
- **Number inoperative engines:** *nEngInop*
- **Power required:** *SET_Preq* **(distributed or fixed)**
- **IR suppressor state:** *STATE_IRS*

Flight State (FltState)

Performance

- **Payload drag increments**

Rotor (supersede rotor model)

- **Induced power factor (*Ki*), profile power mean drag (*cdo*)**
- **Inplane forces calculation**
- **Control mode**

Trim: *STATE_trim*, **match** *Aircraft%IDENT_trim*, *'none'* **for no trim**

- **Configuration default defines trim states:** *IDENT_trim='free', 'symm', 'hover', 'thrust', 'rotor', 'windtunnel', 'power', 'ground'*
- **Requirement for** *trim_target* **depends on** *Aircraft%trim_quant*

Iterations: supersede *Solution* **input**

Performance

Number of flight conditions: *nFltCond*

- **Maximum number** *nfltmax = 21*
- **Input one condition (***FltCond* **and** *FltState* **variables) in** *PerfCondition* **namelist**

Flight Condition (FltCond)

Label: short description for column of output

Gross weight: *SET_GW*

- **Input** (*GW*)
- **DGW, SDGW, WMTO**
- **Maximize for** $P_{req} = fPav P_{av} + dPav$
- **Input payload and fuel weight, gross weight fallout**

Altitude: *SET_alt*

Source for gross weight and altitude: for *SET_GW='source'*

- *KIND_source* **= size mission, size condition, off design mission, performance condition**

Flight Condition (FltCond)

Useful load: *SET_UL*

- **Payload: input, fuel weight fallout**
- **Fuel tank: input, payload fallout**
  - **Auxiliary fuel tanks:** *SET_auxtank* **= adjust, only increase, fixed number**
- **Fixed useful load**
  - **Weight increments (baseline in** *Systems***)**
  - **Kits on aircraft**

Design condition: *DESIGN_zzzz* **(0 not)**

- **Power, DGW, transmission, SDGW, WMTO, rotor thrust**

Flight Condition (FltCond)

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

- **Simplifies input of performance conditions**
- **Input as one flight condition (*PerfCondition*), with sweep variable(s)**
- **Single data structure, so full information saved only for last value**
- **Maximum total number of values for all conditions is *nsweepmax=200***

Control: *SET_sweep = 0* **none,** *1* **from list,** *2* **from range**

Sweep variables: *nquant_sweep* **(maximum 3),** *quant_sweep*

- **For example:** *Vkts, altitude, GW* **(complete list in input manual)**

Range: *sweep_first, sweep_last, sweep_inc*

- **Sweeps executed from** *sweep_last* **to** *sweep_first*
- **Sign of** *sweep_inc* **ignored**

List: *nsweep, sweep, sweep2, sweep3*

OffDesign

Number of missions: *nMission*

- **Maximum number** *nmissmax = 20*
- **Maximum number of segments** *nsegmax = 20* **for each mission**
- **Input one mission (***MissParam***,** *MissSeg***, and** *FltState* **variables) in** *OffMission* **namelist**
  - –All mission segments are defined there, so *MissSeg* and *FltState* variables are arrays
  - –Each variable gets one more dimension, first array index is always segment number
    - › so row in namelist covers all segments

Mission

Label: short description for column of output

Gross weight: *SET_GW*

- **Input (***GW***)**
- **DGW, SDGW, WMTO**
- **Maximize for** P_{req} **=** *fPav* P_{av} **+** *dPav* **for designated segments (***MaxGW***)**
- **Input payload and fuel weight, gross weight fallout**
- **Input payload, fuel weight from mission, gross weight fallout**

Useful load: *SET_UL*

- **Payload: input, fuel weight fallout**
  - **Payload changes:** *SET_pay*
- **Fuel tank: input, initial payload fallout**
  - **Auxiliary fuel tanks:** *SET_auxtank* **= adjust, only increase, fixed number**
- **Mission fuel: fuel weight from mission, initial payload fallout**
- **Fixed useful load**
  - **Folding kit on aircraft**

Reserve: *SET_reserve, fReserve*

Mission

Split segments: increments

- **Distance, time, altitude, takeoff velocity, takeoff height**

Design mission: *DESIGN_zzzz* **(0 not)**

- **Power, DGW, transmission, fuel tank, rotor thrust**

Segment integration: *KIND_SegInt* **= start, midpoint (default), trapezoidal**

Mission iteration: supersede *Solution* **input**

Number of mission segments: *nSeg* **(maximum** *nsegmax = 20***)**

Mission Segment (MissSeg)

All *MissSeg* **and** *FltState* **variables in arrays of length** *nSeg*

Segment definition: *kind, dist* **(nm)**, *time* **(min)**

- *'taxi'*, *'idle'*: **taxi/warm-up (use** *time***)**
- *'dist'*: **fly specified distance (use** *dist***)**
- *'time'*: **fly specified time (use** *time***)**
- *'hold'*, *'loiter'*: **fly specified time (use** *time***), fuel burned but no distance added to range**
- *'climb'*: **climb/descend from present altitude to next segment altitude**
- *'spiral'*: **climb/descend from present altitude to next segment altitude, fuel burned but no distance added to range**
- *'takeoff'*, *'TO'*: **takeoff distance calculation**

Mission Segment (MissSeg)

Segment definition:

- **Reserve**
  - –**Time and distance not included in block time and range**
- **Adjustable**
  - –**Adjust time or distance segments, if** *SET_UL* **not mission fuel**
- **Range credit**
  - –**Range added to specified segment, to facilitate specification of range**
- **Ignore**
  - –**Removed from mission description, for flexible input**
- **Copy**
  - –**Duplicate specified segment**
- **Split**
  - –**Specify number, or calculate from increments**

Segment can be only one of reserve, adjustable, or range credit

Refuel: *SET_fuel*

Mission Segment (MissSeg)

Gross weight: *MaxGW*

- *SET_GW=* **maximize for** $P_{req} = fPav P_{av} + dPav$

Useful load

- **Payload weight change**
- **Fixed useful load**
  - –**Weight increments (baseline in** *Systems***)**
  - –**Kits on aircraft**

Altitude: *SET_alt*

Wind: *SET_wind*

- CA-HI, 85th Percentile, Winter Quartile: 9.59+0.00149*altitude(ft)

Design mission: *SizeZzzz* **(0 not)**

- **Power, DGW, transmission, rotor thrust**

Takeoff distance calculation

Example Mission Profile

```
&DEFN quant='SizeMission', &END
&VALUE
  title = 'Example Payload-Range Mission'
  label = 'Ex-Miss',
  SET_GW = 'max', SET_UL = 'miss', SET_pay = 'none',
  DESIGN_GW = 1, DESIGN_xmsn = 1, DESIGN_tank = 0,  !Mission sets DGW, XMSN Size
  Wpay = 4000.
  SET_reserve = 1, fReserve = 0.1, !Limit Fuel Reserve to 10% of Fuel Burned
  nSeg = 6,
  kind          =  'taxi',   'time',  'climb',   'dist',   'time',   'time',
  dist          =      0.,       0.,       0.,    250.,       0.,       0.,
  time          =      5.,       2.,       0.,      0.,       1.,      30.,
  reserve       =       0,        0,        0,       0,        0,        1,
  adjust        =       0,        0,        0,       0,        0,        0,
  range_credit  =       0,        0,        4,       0,        0,        0,
  split         =       0,        0,        0,       1,        0,        0,
  MaxGW         =       0,        1,        0,       0,        1,        0,
  Vkts          =      0.,       0.,     250.,    275.,       0.,     275.,
  ROC           =      0.,       0.,       0.,      0.,       0.,       0.,
  SET_atmos     =  'temp',   'temp',    'std',   'std',   'temp',   'temp',
  altitude      =      0.,       0.,    4000.,  12000.,    4000.,    4000.,
  temp          =  102.92,   102.92,     59.0,    95.0,     95.0,     95.0,
  rating        =   'IRP',    'MRP',    'IRP',   'MCP',    'MRP',    'MCP',
  fPower        =     1.0,     0.95,      1.0,     1.0,     0.95,      1.0,
  SET_max       =       0,        0,        2,       1,        0,        1,
  max_quant     =     ' ',      ' ',  'Pmarg', 'range',      ' ',  'range',
  max_var       =     ' ',      ' ',    'ROC', 'speed',      ' ',  'speed',
  max_quant(1,2)=     ' ',      ' ',  'climb',     ' ',      ' ',      ' ',
  max_var(1,2)  =     ' ',      ' ',  'speed',     ' ',      ' ',      ' ',
  STATE_control =       1,        1,        2,       2,        1,        2,
  coll          =      1.,       5.,       3.,      2.,       5.,       2.,
  lngcyc        =      0.,       0.,       2.,      4.,       0.,       4.,
  pitch         =      0.,       0.,       5.,      2.,       0.,       2.,
  pedal         =      0.,       4.,       0.,      0.,       4.,       0.,
  STATE_trim    = 'power',  'hover',   'free',  'free',  'hover',   'free',
&END
```

Using defaults for many parameters

climb seg #3 distance credited to seg #4

seg #4 to be split into multiple segments

Either hover segment may limit TOGW

Best range speed segment

Best rate of climb speed segment

Outline

Introduction

Documentation

Overview

- Tasks
- Aircraft

NDARC Job

- Input
- Organization
- Output

Solution Procedures

- Debugging

Input Manual

- Aircraft
- Tasks

Tutorial

Tutorial Tasks

Organization

Command file

Train01a: Size helicopter

- Organization
- Design flight profile
- Exercises for student

Train01b: Disk loading sweep

- Multiple-case jobs

Train01c: Analyze saved design

Train01d: Change technology

Train01e: Size with changed technology

Train02a: add fuel tank sizing mission

- Design flight profile

Train02b/c/d: Off-design conditions

Train02e/f: Off-design missions

Exercises for student

Organization of Tutorial Files

Folders

- **Aircraft: aircraft input**
- **Engine: engine model**
- **Size:** *size* **and** *no_size*
- **Solution: solution files**
- **Condition: point conditions and sweeps**
- **Mission: design and off-design missions**
- **Output: files generated by NDARC**

Command files

- *file.bat*: **execute NDARC, redirect input (to** *file.njob*) **and output**
- *file.njob*: *Cases* **and** *Size* **input, read secondary files**

Command File

File ***train01a.bat*****:**

```
..\..\bin\ndarc.exe < train01a.njob > Output\train01a.out
```

Change path as required for location of NDARC executable

Change format as appropriate for operating system

Double-click to run

Line Endings

NDARC files in distribution package generally have mix of line endings

- Unix: LF
- Mac: CR
- Windows: CR-LF

Often source of problems on PCs, when try to compile or run without checking line endings

- Change to Windows line endings with editor such as Notepad++

Train01a: Size Helicopter

Existing helicopter model

- **Size weight and power, 1 design mission and 5 design flight conditions**

Command: *train01a.bat* **=>** *train01a.njob*

- **Reads** *train01.airc, gen2000.ts, solution.sol, design01.cond, design01.miss*
- **Produces** *train01a.out, train01a.dsgn, train01a.perf, train01a.geom, train01a.dxf, train01a.acd, train01a.soln*

Input files

- **Examine aircraft model**
- **Examine design mission and flight conditions**

Output file *train01a.out*

- **Check convergence (search for "case conv")**
- **Examine design solution**
- **Examine performance results**

Organization

Order in **.njob* **file:**

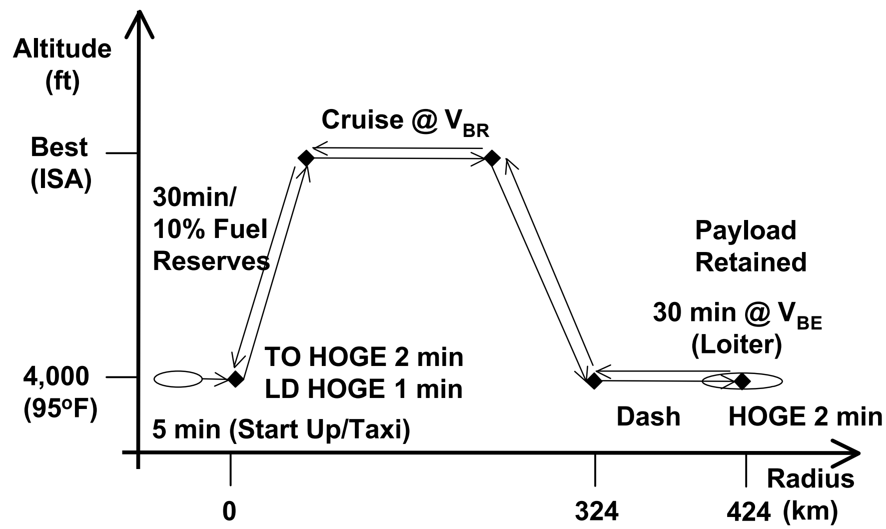
- **Read aircraft, engine, conditions, missions first**
  - **so can change later in file**
- **When change condition or mission in** **.njob*, **must identify by number**
  - **otherwise creating input for new condition/mission**

Input files

- **Organize the data**
  - **Many variables per line (shorter file)**
    - **Perhaps in columns**
  - **Or one variable per line (comments easier)**
- **Note default values**
  - **Make extensive use of defaults**
  - **Define even if has default value**
    - **Perhaps with comment describing variable options**

Utility Mission Flight Profile

Segment		Atm.		Time (min)	Dist. (km)	Speed (KTAS)	VROC Cap. (fpm)	Engine Rating
1	Taxi	4k	95°F	5	-	-	-	=100% MCP
2	Hover	4k	95°F	2	-	HOGE	500	≤95% MRP
3	Climb	-	ISA	-	Credit	~Vy	Fallout	≤100% IRP
4	Cruise	Best	ISA	-	324	V_{BR}	-	≤100% MCP
5	Dash	4k	95°F	-	100	V_{DASH}	-	=90% MCP
6	Loiter	4k	95°F	30	-	V_{BE}	-	≤100% MCP
7	Hover	4k	95°F	2	-	HOGE	500	≤95% MRP
8	Dash	4k	95°F	-	100	V_{DASH}	-	=90% MCP
9	Climb	-	ISA	-	Credit	~Vy	Fallout	≤100% IRP
10	Cruise	Best	ISA	-	324	V_{BR}	-	≤100% MCP
11	Hover	4k	95°F	1	-	HOGE	500	≤95% MRP
12	30min/ 10% Res.	Best	ISA	-	-	V_{BR}	-	≤100% MCP



Notes:
Sizes aircraft design gross weight and power
2500lb internal payload
HOGE: Hover out of ground effect; aircraft has capability for 500fpm VROC
VROC: Vertical rate of climb (purely vertical flight, no horizontal component to velocity)
Best: Selected for configuration's best performance
V_{BE}: Best endurance speed; minimum fuel flow
V_{BR}: Best range speed. May elect to use long range cruise speed; 99% of maximum specific range, high side
V_{DASH}: Dash or penetration speed
V_Y: Best rate of climb speed
Reserve fuel is that required for either 30 minutes at V_{BR} or 10% of mission fuel, whichever is greater

Exercises for Student

Add design flight condition to specify maximum speed (DGW, 100%MCP)

- **Define speed (***Vkts***), omit max effort (***SET_max=0***), size power (***DESIGN_engine=1***)**

Vary cruise altitude in mission

Vary take-off conditions

- *altitude, SET_atmos, temp*

Ignore climb segments in mission

- **Set** *ignore=1*

Activate best-altitude calculation in mission segments 4 and 10

Activate speed for best climb calculation in mission segments 3 and 9

- **Use** *SET_max=2*
- **If does not converge, debug with trace and revise solution parameters or initial conditions**

Train01b: Disk Loading Sweep

Vary disk loading in multiple-case job

&DEFN quant='Rotor 1',&END
&VALUE diskload=x.,&END
&DEFN action='endofcase',&END

Input for each case terminated by *action='endofcase'*

Exercises for student

- **Add disk loading values**
  - **If does not converge, debug with trace and revise solution parameters or initial conditions**

Multiple-Case Jobs

Set initialization of parameters from previous case in *Job* namelist

- **Inherit only input (default)**

  &JOB INIT_input=1,INIT_data=0,&END

- **Inherit design and solution: all parameters (input, input-modified, derived)**

  &JOB INIT_input=2,INIT_data=2,&END

Input for each case terminated by *action='endofcase'*

Separate design and performance files produced for each case (write_files=0)

- **Define file names for each case**

  &DEFN quant='Cases',&END

  &VALUE FILE_design='zzzz.dsgn',FILE_perf='zzzz.perf',&END

May be necessary to delete missions and conditions (*quant='delete'*)

Can sometimes solve difficult problems using multiple-case job with all input inherited from previous case

- **Sweep from easy-convergence to hard-convergence (for example, tech factors)**
- **Produce output files only for last case**

Train01c: Analyze Saved Design

Start with Train01a

Load aircraft description file (instead of engine and aircraft input)

&DEFN action='load aircraft',file='Output\train01a.acd',&END

Off-design mission and performance analysis:

&DEFN quant='Cases',&END

&VALUE TASK_size=0,TASK_mission=1,TASK_perf=1,&END

Change flight condition and mission defininition from design to off-design

- *Mission* **=>** *OffMission*, *SizeCondition* **=>** *PerfCondition*
  - **Flight condition** *KIND_source=1* **=>** *3*
- *DESIGN_zzzz* **not used**

Change Size parameters to no-sizing

```
&DEFN quant='Size',&END
&VALUE
  title='No Sizing',SIZE_perf='none',SET_rotor=2*'Radius+Vtip+sigma',
  FIX_DGW=1,FIX_WE=0,SET_SDGW='input',SET_WMTO='input',
    SET_tank='input',SET_limit_ds='input',
&END
```

Train01d: Change Technology

Start with Train01c

Revised technology in file *helicopter_tech.airc*

- **Induced power** *Ki_hover=1.125* **=>** *1.09*
- **Profile power** *cd_hel=0.0080* **=>** *0.0070*
- **Blade weight tech factor** *TECH_blade=1.0* **=>** *0.85*

Tech factors can be input in component structure, or in *quant='TechFactors'*

Read *helicopter_tech.airc* **after load** *training01a.acd*

Convergence requires *relax_maxgw=0.3* **(was** *0.5***)**

Train01e: Size with Changed Technology

Start with Train01a

Revised technology in file *helicopter_tech.airc*

- **Induced power** *Ki_hover=1.125* **=>** *1.09*
- **Profile power** *cd_hel=0.0080* **=>** *0.0070*
- **Blade weight tech factor** *TECH_blade=1.0* **=>** *0.85*

Tech factors can be input in component structure, or in *quant='TechFactors'*

Read *helicopter_tech.airc* **after read** *helicopter.airc*

Convergence requires

&DEFN quant='SizeMission 1',&END

&VALUE init_trim(8)=1,&END

Train02a: Add Fuel Tank Sizing Mission

Add second mission for fuel tank sizing
Start with Mission 1

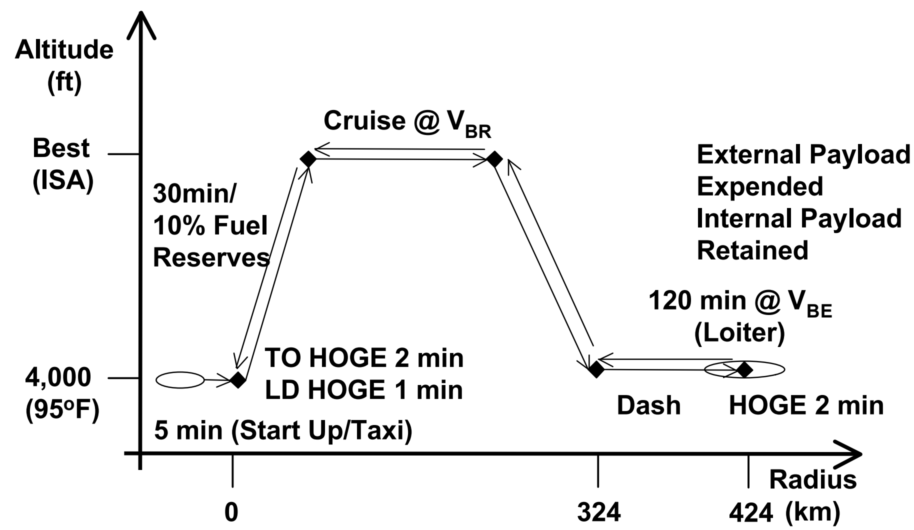
Midpoint loiter (segment 6): 120 min
Payload: 0 passenger, 1000 lb (and 5 ft^2 drag) dropped at segment 7
Mission equipment: additional 400 lb

Mission 1: *DESIGN_tank=0*
Mission 2: only *DESIGN_tank=1*

Convergence requires *init_trim=1*

Design Flight Profile 2 — Fuel Tank Sizing

Segment		Atm.		Time (min)	Dist. (km)	Speed (KTAS)	VROC Cap. (fpm)	Engine Rating
1	Taxi	4k	95°F	5	-	-	-	=100% MCP
2	Hover	4k	95°F	2	-	HOGE	500	≤95% MRP
3	Climb	-	ISA	-	Credit	~Vy	Fallout	≤100% IRP
4	Cruise	Best	ISA	-	324	V_{BR}	-	≤100% MCP
5	Dash	4k	95°F	-	100	V_{DASH}	-	=90% MCP
6	Loiter	4k	95°F	120	-	V_{BE}	-	≤100% MCP
7	Hover	4k	95°F	2	-	HOGE	500	≤95% MRP
8	Dash	4k	95°F	-	100	V_{DASH}	-	=90% MCP
9	Climb	-	ISA	-	Credit	~Vy	Fallout	≤100% MCP
10	Cruise	Best	ISA	-	324	V_{BR}	-	≤100% MCP
11	Hover	4k	95°F	1	-	HOGE	-	≤95% MRP
12	30min/ 10% Res.	Best	ISA	-	-	V_{BR}	-	≤100% MCP



Notes:
Sizes aircraft's fuel tank capacity
1400lb payload (1000lb expendable and 400lb MEP delta)
HOGE: Hover out of ground effect; aircraft has capability for 500fpm VROC
VROC: Vertical rate of climb (purely vertical flight, no horizontal component to velocity)
Best: Selected for configuration's best performance
V_{BE}: Best endurance speed; minimum fuel flow
V_{BR}: Best range speed. May elect to use long range cruise speed; 99% of maximum specific range, high side
V_{DASH}: Dash or penetration speed.
V_Y: Best rate of climb speed
Reserve fuel is that required for either 30 minutes at V_{BR} or 10% of mission fuel, whichever is greater
5 ft^2 external stores drag area

Train02b/c/d: Off-Design Conditions

Start with Train02a

Analyze saved design, aircraft description *train02a.acd* **(as for Train01c)**

Power vs speed (Train02b)

- **Sweep: speed = 0 to 180 knots**
- **Fix: DGW, altitude**
- **Atmosphere: SLS, 4k/95, 12k/ISA**

Best effort speeds vs altitude (Train02c)

- **Sweep: altitude = 0 to 20000 ft**
- **Fix: DGW, SLS**
- **Maximum effort: Vbe, Vbr, VMCP**

HOGE Ceiling vs GW (Train02d)

- **Sweep: altitude = 0 to 20000 ft**
- **Atmosphere: ISA, 95, 95 + 500fpm VROC**
- **Maximum GW**

Train02e/f: Off-Design Missions

Start with Train02a
Analyze saved design, aircraft description *train02a.acd* **(as for Train01c)**

Self-deploy (Train02e)

- **Takeoff weight: WMTO; payload = 0; aux tanks as required**
  - **Aux tank defined in** *helicopter.airc*: *Waux_cap=2000.,fWauxtank=0.11,*
- **Headwinds**
- **Adjust distance for available fuel**
- **Convergence requires** *relax_miss=0.5*

Payload-Range (Train02f)

- **Mission 1: Design mission, 4k/95 takeoff/midpoint/landing**
  - **subsequent missions copy mission 1**
  - **revise** *SET_GW, SET_UL, adjust, range_credit, ignore, Max_GW*
- **Mission 2: go nowhere**
  - **max TOGW, fallout payload; ignore climb and cruise segments**
- **Mission 3: dash only**
  - **max TOGW, fallout payload; ignore climb and long-range cruise**
- **Mission 4: payload corner (max internal fuel)**
  - **max TOGW, fallout payload; adjust distance for available fuel**
- **Mission 5: zero payload (max internal fuel)**
  - **zero payload, fallout gross weight; adjust distance for available fuel**

Excercises for Student

Change configuration

- Tandem helicopter
- Coaxial helicopter
- Tiltrotor

Autogyro

- Add propeller and drive train (or simple jet)
- Disconnect rotor from engine
- Define controls, trim strategies

Compound helicopter

- Add wing
- Add propeller, connect to drive train
- Define controls, trim strategies