*Revision: December 2014*

# CONDOR:
# CONceptual Design Of Rotorcraft

## Software User's Manual

**Integrated Product Lifecycle Engineering Laboratory**
**Georgia Institute of Technology, Atlanta, GA**


Georgia Institute of Technology



The CONDOR user manual and software were developed by:

Benjamin England, bengland3@gatech.edu,
Mike Roberts, mroberts@gatech.edu, and
Sylvester Ashok, sylvester_ashok@gatech.edu

# Contents

# Summary

CONDOR is a set of programs and scripts developed to evaluate the performance of single main rotor (SMR) helicopter and size a SMR for mission and performance requirements. CONDOR is written in Python 2.7, and requires editing of several input files in order to customize its use to a particular scenario and output the desired data. In order to better understand the process and check that your results are valid, a basic knowledge of Python is recommended, but not required.

CONDOR works using blade element momentum theory (BEMT) for performance calculations and the ratio of fuel ($R_f$) method for the calculation of fuel requirements and gross weight.

If the user would like to learn more about python we suggest using the following resources.

Additional Python resources:

https://docs.python.org/2.7/
http://www.learnpython.org/
https://wiki.python.org/moin/BeginnersGuide

# 1: Installation Instructions

The necessary program files as well as some example input files are included with the distribution zip file. This can be extracted to any desired location. It requires Python as well as five additional libraries, all of which are available online. For a Windows installation, these can be downloaded from the following websites and installed.

- Python 2.7: http://www.python.org/
- NumPy: http://www.numpy.org/
- SciPy: http://www.scipy.org/
- MatPlotLib: http://matplotlib.org/
- Configuration/ Validate Dependency: https://pypi.python.org/pypi/configobj/5.0.6

If using Linux or MacOS, NumPy, SciPy, and MatPlotLib should all be available through your distribution's package manager. For example, to install all the dependencies under Debian or Ubuntu, open a console and type:

```
sudo apt-get install python python-numpy python-scipy python-matplotlib
```

The Configuration and Validate Dependencies can be installed either in your Python "libs" folder in the python 2.7 installation location or in the root <CONDOR/> location after unzipping. There is no advantage to either method.

# 2: File Architecture

The executables and python scripts needed to run CONDOR are all located in the root <*CONDOR/*> directory.

The configuration files (missions, vehicles, and c81 tables) are specified in various '.cfg' files, located in the <*CONDOR/Config/*> directory. Vehicle description files are located in <*./Config/Vehicles*>. Mission files are located in <*./Config/Missions/*>. The C81 tables are located in <*./Config/C81/*>. The configspec files are located in <*./Config/Configspec/*>. The standard atmosphere is located in <*./Config/Standard_Atmosphere/*>. The particular configuration to use is specified in the master input file (also stored in the <*CONDOR/Config/*> directory). It is necessary to edit the input files to switch configuration. Editing this file is explained further in the tutorial section of this Manual.

All vehicle output is written to <*CONDOR/Output/*>. Specifically, vehicle output is written to <*CONDOR/Output/vehicleName_VehOutput.cfg*>. Graphical output is stored in <*CONDOR/Output/Figures/*>.

There are three layers of code. The first level, CONDOR.py, reads the master input file and decides which codes to run. The next layer, indicated by <*name_Run.py*>, executes the necessary sequence of functions to run the specific function desired. The final layer is the class definition that hold the actual function definitions that perform evaluations. The following sections will take you through the code dependencies and the descriptions of each code and there interactions. All codes are stored in the root *CONDOR* directory.

### CONDOR.py

This file contains the master script, which runs all the other scripts. The functions contained in this file read the master input configuration file and accordingly direct the other files to run using the specified data input files.

### Rf_Run.py

This code runs the ratio of fuel method on the specified vehicle by validating the input file and then calling the necessary function definitions from the <*rf.py*> file.

### rf.py

This is an implementation of the Ratio of Fuel ($R_f$) method. The output is the final gross weight of the vehicle based on the mission requirements and set vehicle characteristics in the output folder. The code takes in the vehicle specs from the vehicle configuration file then runs <*vehicle.py*> to get performance data at each subsequent gross weight. It iterates through gross weight values specified in the vehicle configuration file in the "Simulation" section to find the gross weight that satisfies the mission based on the expected weight build up and mission requirements.

### vehicle_Run.py

This code runs the vehicle analysis on the specified vehicle; this code is for pure performance analysis only. With the correct master inputs it will run on multiple similar gross

weights as is in the baseline gross weight given in the vehicle specification. If the vehicle name matches names hardcoded into the files it will also give power comparisons from the actual vehicle power data. Given this ability it is important to make sure the name only matches one of the following names if the vehicle is actually one of the following, otherwise incorrect data will be output.

| |
|---|
| 's92' |
| 'XH59' |
| 'MD_530FF' |
| 'CH53E' |

### vehicle.py

This code contains all the function definitions used by the RF codes and the vehicle run code. The definitions contained are used to evaluate the performance data of the aircraft. In each flight regime it uses the BEMT code to evaluate power requirements and whether or not the rotor can trim in forward flight for each mission segment.

### BEMT_Run.py

This code runs the BEMT.py code which will print the power required to trim at the altitude and delta temperature specified in the vehicle engine sizing section at a default speed of 150 knots: (this value can be changed by going into the *<BEMT_Run.py>* code and changing the value in the *run_BEMT()* definition; the value that needs to be changed is after the *<else:>* statement)

### BEMT.py

This code sets up the rotor parameters then trims the rotor and evaluates the power required in the specified configuration. It is not dependent on any of the codes included in CONDOR. This code is designed to use the C81 table that is specified in the vehicle configuration file. If the rotor is unable to trim, the Vehicle.py code will output a trim failure message to the output file. The rotor will not trim if it has crossed the aerodynamic capability limit, i.e. it has exceeded its bladed loading limit.

# 3: Tutorial

Once the dependencies are installed, the various scripts included in CONDOR can be run by opening a command prompt or terminal, and navigating to the containing directory to run <*python CONDOR.py Config/input_file.cfg*>, where *input_file.cfg* can be any name you desire, but is associated with the master input file and follows the format laid out in "*Master_Input.configspec*," which is explained in the next section.

## Master Input Configuration File

The master configuration file requires a series of 'True' or 'False' statements as well as the specification of the Aircraft and Mission configuration file locations. The sections and possible inputs are laid out in the table below. The text in the column called "Section Name" and the text in the column "Sub Section Name" cannot be changed.

If "Rf_Run" is set to 'True' the $R_f$ method will be used to size a helicopter to the input mission.

If "Vehicle_Run" is set to 'True' the code will run the input vehicle using the input Gross Weight (GW) and vehicle characteristics, and attempt to perform the mission, and output the vehicle performance data: the "Vehicle_Run" setting is how vehicle performance characteristics can be calculated.

If "BEMT_Run" is set to 'True' along with "BEMT_debug" and "BEMT_plot", rotor trim plots and BEMT data will be output for the aircraft operating at 150 knots: the "BEMT_debug" code is mainly used for debug purposes and it is also the way to output rotor trim plots. The speed at which the debug is run at is hardcoded in but this is a simple change.

If "Vehicle_showPower" is set to 'True' power curves will be output at altitude specified in the Vehicle input file. These are found in the output folder. All sections must have values assigned in order for the code to run.

| **Section Name** | **Sub-Section Name** | **Input Value Type** | **Description** |
|---|---|---|---|
| [Code Selection] | | | define codes to run |
| | BEMT_Run | boolean() | true/false- run BEMT alone |
| | Rf_Run | boolean() | true/false- run Rf alone |
| | Vehicle_Run | boolean() | true/false- run Vehicle alone |
| [Veh Mission Config] | | | define aircraft and mission file |
| | Aircraft_Config | string() | "Config/Vehicles/<file name>" |
| | Mission_Config | string() | "Config/Missions/<file name>" |
| [BEMT Options] | | | BEMT run options |
| | BEMT_debug | boolean() | debug options- true/false; outputs to cmd window all trim data |
| | BEMT_plot | boolean() | plot BEMT output; create trim plots at current velocity; requires |

| | | | |
|---|---|---|---|
| | | | BEMT_debug=true |
| [RF Options] | | | RF Run options |
| | RF_debug | boolean() | debug options- true/false; prints debug information to cmd window; set to 'false' in normal use |
| | RF_writeOutput | boolean() | write output to file- true/false; when 'false' runs slightly faster but no displayed results; mainly used for debug procedures; set 'true' in normal use. |
| [Vehicle Options] | | | vehicle run options |
| | Vehicle_showPower | boolean() | shows power curves if Vehicle_Run is true |
| | Power_Breakdown | Boolean() | shows the power curve breakdown into Parasite, Induced and Profile as well as the total power and Max continuous power |
| | vehicle_debug | boolean() | vehicle debug option; debug information printed to cmd window; set to 'false' for normal use |
| | Vehicle_debugFine | boolean() | vehicle fine debug option; more debug information |
| | Vehicle_size_compare | boolean() | will run a series of GWs for vehicle power curves if desired |
| | vehicle_Alt_Power_Curve | Boolean() | select whether or not to plot altitude power curve |
| | vehicle_CT_Sigma_Curve | Boolean() | select whether or not to create a CT/sigma vs advance ratio plot; increases run time significantly |
| | Trim_Velocity_Plots | Boolean() | Select whether or not 5 trim versus velocity plots will be created. Plots are: Inflow, Collective, Beta0, Theta_1c, Theta_1s |

### Vehicle Configuration File

The vehicle configuration has many inputs whose types vary extensively. The following table explains each section.

This file is used for both design and performance calculations. In the event of performance calculations weights, engine values and drag values are taken directly from the file for use. For design procedures the values are scaled with respect to gross weight and power required at that gross weight. Values with a star next to the m in the first column are scaled when the RF method is run but set to their baseline values when the vehicle performance code is run.

| **Section Name** | **Sub-Section Name** | **Type** | **Description** |
|---|---|---|---|
| [Sizing Results] | no input required; section used for output | | |
| [Economics] | no input required; section used for output | | |
| [Veh_Name] | | | |
| | name | String() | name of the vehicle- used for output file naming |
| [Performance] | no input required; section used for output | | |



| | | | |
|---|---|---|---|
| [Main Rotor] | | | |
| | NumRotors | integer() | # of rotors |
| | DiskLoading | float() | lb/ft$^2$ |
| | NumBlades | integer() | # of blades |
| | Solidity | float() | solidity ratio |
| | TipSpeed | float() | ft/sec |
| | Kint | float() | rotor interference fraction |
| | Kov | float() | rotor overlap fraction |
| | AirfoilFile | string() | C81 table file name |
| | DragDivergenceMachNumber | float() | drag divergence mach number for the blade airfoil |
| | AverageChord | float() | blade average chord |
| | TaperRatio | float() | ratio of tip to root taper |
| | TipTwist | float() | twist of tip relative to root, degrees |
| | RootCutout | float() | non-dimensional root cutout, fraction of radius |
| [Weights] | | | |
| * | BaselineGrossWeight | float() | lbs |
| * | BaselineEmptyWeight | float() | lbs |
| | NumEngines | float() | number of engines |
| * | BaselineWeightPerEngine | float() | lbs |
| | BaselineDriveSystemWeightScalingFactor | float() | scaling factor multiplied by engine weight to empirically estimate drive system weight |
| | StructureWeightTechImprovementFactor | float() | technology factor for structural improvement |
| | EngineWeightTechImprovementFactor | float() | technology factor for engine weight improvement |
| | DriveSystemWeightTechImprovementFactor | float() | technology factor for transmission improvement |
| | UsefulLoad | float() | minimum enforced payload capacity, lbs |
| [Powerplant] | | | |
| * | BaselineMRP | float() | baseline maximum rated power, hp |
| * | BaselineMCP | Float() | Baseline maximum continuous power, hp |
| * | BaselineSFC | float() | baseline specific fuel consumption lbs/hp-hr |



| | | | |
|---|---|---|---|
| | SFCTechImprovementFactor | float() | tech factor is expect investment to improve SFC |
| | TransmissionEfficiency | float() | transmission power losses (0 for no losses) |
| | IdlePower | Float() | Power required in Idle Segments |
| [Body] | | | |
| | DragTechImprovementFactor | float() | improvement factor for anticipated improvements in body aerodynamics through investment (0 if no improvements) |
| | DownwashFactor | float() | downwash factor for rotor flow impinging on body |
| * | BaselineFlatPlate | Float() | baseline flat plate drag area, ft^2 |
| [Antitorque] | | | |
| | AntitorquePowerFactor | float() | proportion of total power that goes to anti-torque |
| | TailLength_RotorRadiusRatio | float() | ratio of tail length to the rotor radius; must be greater than 1 to prevent rotor tail interference |
| | NumBladesTail | integer() | number of blades on tail |
| | TailSolidity | float() | tail rotor solidity |
| | TailDiskLoading | float() | disk loading for tail rotor, $lb/ft^2$ |
| | TipSpeed | float() | tail tip speed, ft/sec |
| | CD0 | float() | tail blade section $CD_0$ |
| [Simulation] | | | |
| | TimeStep | float() | mission time step, minutes |
| | MaxSteps | float() | max number of simulation steps |
| | StartGW | float() | Starting GW for search bounds |
| | GWMin | float() | minimum gross weight search bound, pounds |
| | Gwmax | float() | maximum gross weight search bound, pounds |
| | GWTolerance | float() | Tolerance for accuracy of the Gross weight search |
| | TrimAccuracyPercentage | float() | accuracy of the trim solution in percent |
| | numBladeElementSegments | integer() | number of segments in the radian and circumferential direction to divide the rotor disk into |
| | PowerCurveResolution | float() | spacing of power curve steps, knots |
| | HoverCeilingMax | float() | maximum hover ceiling search bound, ft |
| | HoverCeilingMin | float() | minimum hover ceiling search bound, ft |
| | HoverCeilingTolerance | float() | percentage accuracy of the hover ceiling search |

| | | | |
|---|---|---|---|
| | CT_SigmaTolerance | Float() | Tolerance used for creating max $C_T/\sigma$ curve. Smaller=more accurate & greater time |
| | CT_SigmaCurveResolution | Float() | steps for $C_T/\sigma$ curve, knots |
| | Curve_Altitude | Float() | altitude at which powers curves are calculated at, ft |
| [Engine Scaling] | | | |
| | RequiredHoverHeight | float() | If can't hover at this height force execution failure |
| | DeltaTemp | float() | for all engine power analysis and engine scaling ISA +\- temp (different from mission +\-) |
| | CruiseAltitude | float() | altitude for scaling engine performance to, ft |
| [Power Curve] | no input required; section used for output | | |
| [Condition] | no required input; Section used in execution | | |
| [Trim Failure] | no input required; section used for output | | |

### Mission Configuration File

The mission configuration file defines the mission as many mission segments as desired for analysis at each condition. If it is a climb segment assumes the initial altitude is the previous segments final altitude. The segments should be named following the format [Segment #] where # is the segment number.

| Section Name | Sub-Section Name | Input Value Type | Description |
|---|---|---|---|
| [__many__] | Means can have as many mission sections as desired | | |
| | Distance | float() | mission distance, nmi |
| | Speed | float() | mission speed, knots |
| | ClimbSegment | boolean() | true/false- whether or not this is a climb segment |
| | ClimbRate | Float() | Ft/min climb rate only used in Climb Segment |
| | IdleSegment | Float() | (min) Minutes of idle power warmup |
| | IGE | Boolean() | (True/ False) In ground effect hover at $\frac{z}{R} = .75$. * |
| | Altitude | float() | mission altitude or altitude climbing to, ft |
| | DeltaTemp | float() | ISA +/- |
| | CrewWeight | float() | crew weight, lbs |
| | PayloadWeight | float() | payload weight, lbs |
| | MiscWeight | float() | additional weight, lbs |

* Do not set to true in forward flight as it will give erroneous result. There is an effect for flying close to the ground but the models are much more complex and not well understood and most importantly not included in the code. However, do to how the code is written a value will be output. In order to prevent the code from stopping, no error is thrown in this case.

### C81 File

The C81 file format consists of 4 columns in a file with the designation "*<name>.c81*". The first column contains the angle of attack for $\alpha$ from -180° to 180°, the second column contains the coefficient of lift, the third column is the coefficient of drag, and the last column is the moment coefficient.

### ISA Standard Atmosphere

The ISA Standard atmosphere is provided as text file and is titled "ISA_Standard_Atmosphere.txt"; this title is hardcoded into the software so if a different table is desired, the title of the file needs to be replaced with this, ensuring that the format body is identical. The first two lines are column headers and the data begins on the third line. If a different format is desired, the source code would have to be modified.

- Important Debug/Troubleshoot Information -

The code is setup to output specific error codes if the BEMT code is unable to trim the rotor. These error codes are found in the list of 'power required' outputs as the last value, which are found in vehicle outputs "*xxx_VehOutput.cfg*" *file*.

The values and their associated meanings are as follows:

| | |
|---|---|
| 1. | The rotor failed to produce enough lift, aerodynamic limit reached |
| 2. | The power failed to converge |
| 3. | A very high collective pitch is required to trim |
| 4. | The power required has exceed limits (greater than 40000 hp) |
| 5. | The trim routine was unable to balance the rotor longitudinally |
| 6. | The trim routine was unable to balance the rotor laterally |
| 7. | Some other unknown issue |

While conduction performance evaluations and flying the mission, the vehicle is not scaled, and hence if it is unable to perform this mission, the output file will display one of a few different options.

- If the required power is higher than the available power in any mission segment, the code will continue to execute. However, there will be a warning message in the Mission output for that segment that says: '*Warning: More Power Required than available*'
- If the code is unable to trim the aircraft at any speed an output value is produced as described in the previous section labeled "Important Debug/ Troubleshoot Information".
- At the top of the mission output file there is a value called "Mission Fuel Left." This value represents the difference between the amount of fuel at takeoff and fuel used. If the value is negative then the output is indicating that more fuel is required than is available.
- The event of a trim failure also produces a description in the vehicle output file.

During the Ratio of Fuel method there are times when no feasible solution is found. When this occurs a message will be output in the vehicle output file. Under the section "Sizing Results" the message after "Stop Reason" will explain whether or not the code found a solution.

# 4: Example Input Files

There are several example input files provided in the "CONDOR\Config\"directory. These example files represent data from several existing aircraft and notional missions. Also contained are several C81 files used by the code and the standard atmosphere.

For example to run the MD 530 performance analysis, navigate the command prompt to the CONDOR root directory and enter the following command:

*Python CONDOR.py Config/4_Master_Input.cfg*

This input file is setup to run the MD530 vehicle (defined in: *<Config/Vehicles/vehicle_MD_530FF.cfg>*) performance analysis and will output the power curve with a comparison to the published data and a power component breakdown.

It will also attempt to perform the mission defined in: *<Config/Missions/MD_530FF_Mission1.cfg>*

Output data for this analysis is generated and stored in the "MD_530FF_MissionOut.cfg" and "MD530FF_VehOutput.cfg" files

# 5: Theoretical Background

The CONDOR program consists of two main components, Analysis and Design. The Design is done through the editing of the configuration files as well as the RF method which calculates total fuel required. The Analysis is done through several performance calculations for a given configuration. The backbone of these calculations is the determination of power required and power available. Calculating power required is a function of the aircraft configuration, the altitude, temperature and flight speed. Power available is a function of altitude and temperature. The power required calculation is based on the equation below.

$$HP_{Total} = (ihp + Rhp + php + HP_{TR} + HP_{ACC})\left(\frac{1}{\eta_{XMSN}}\right)$$

Where:

$ihp$ = rotor induced power
$Rhp$ = rotor profile power
$php$ = *aircraft parasite power*
$HP_{TR}$ = tail rotor power
$HP_{ACC}$ = accessory power
$\eta_{XMSN}$ = transmission efficiency

The power available for a turboshaft engine is determined from the equation:

$$HP_{AV} = HP_{ISA/SLP}\frac{\rho_{altitude}}{\rho_{ISA/SLP}}$$

The analysis calculates the power required and available for a given gross weight and airspeed. The calculation process for the analysis is shown below.

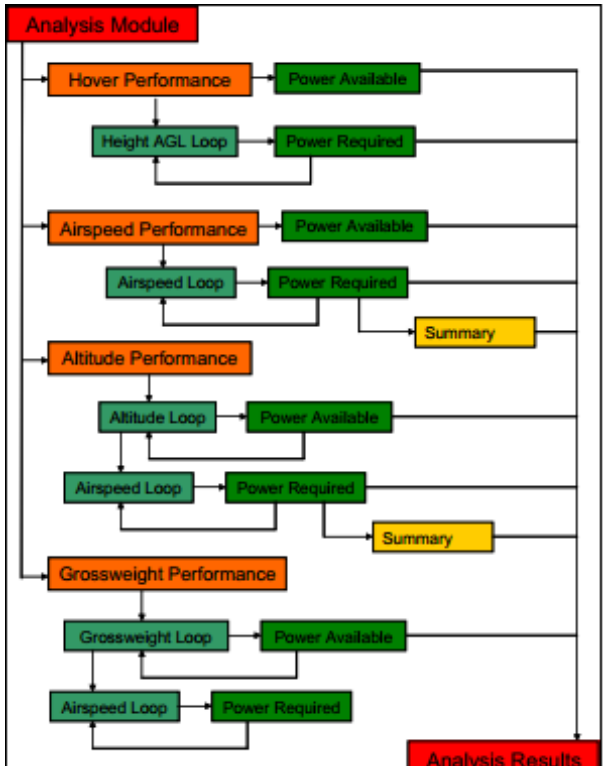


*Figure 1. Calculation process for analysis of given condition.*

The design module uses the power required and available to determine the fuel required to complete a mission as well as the fuel available for that gross weight. The gross weight is then adjusted until the fuel required matches the fuel available. This process is called the Ratio of Fuel ($R_F$) method. The iteration scheme for the $R_F$ method is shown below.

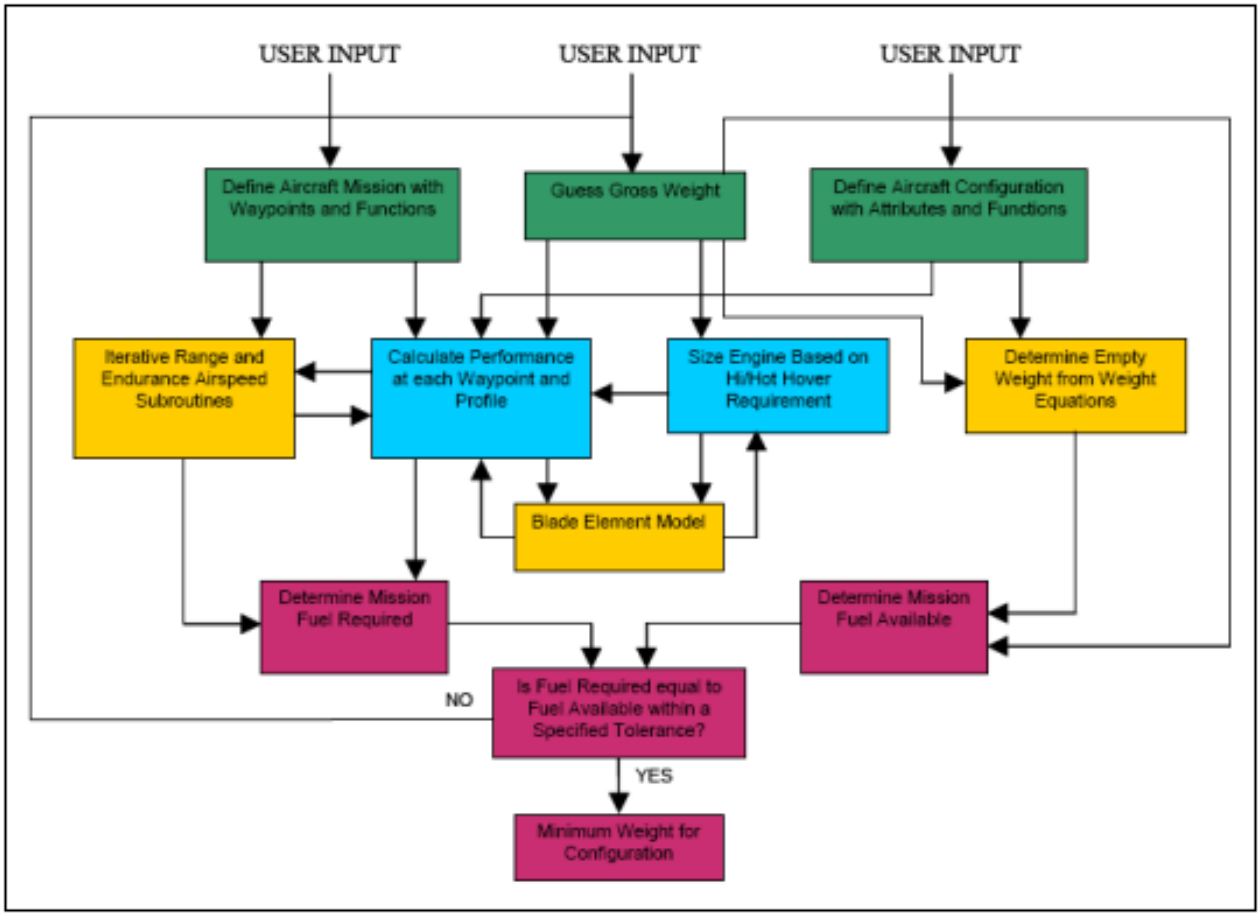


*Figure 2. Ratio of Fuel method process.*

## Blade Element Momentum Theory

CONDOR uses combined Blade Element Momentum Theory in order to calculate the rotor induced power and rotor profile power. Below is the inflow model that is used in the BEMT code and is based on the Prandtl Root/ Tip loss model.

$$\lambda_i = \frac{T}{2\rho A\sqrt{(V\cos(\alpha_{TPP}))^2 + (V\sin(\alpha_{TPP}) + F * \lambda_i)^2}}$$

$$f_{root/\,tip} = \frac{N_b}{2}\left(\frac{1-r}{r\phi}\right)$$

$$F_{root/\,tip} = \left(\frac{2}{\pi}\right)\cos^{-1}(\exp(-f))$$

Where:

- $\lambda_i$ = inflow
- $T$ = Thrust

- $A$ = Blade Area
- $V$ = Forward speed
- $\alpha_{TPP}$ = the angle between the vertical force and horizontal force being supplied by the rotor tip-path-plane
- $F$ = Loss correction factor
- $\phi$ = induced inflow angle
- $r$ = local radius along the rotor

This solution for this inflow requires an iterative process that converges on the inflow solution. $F_{root}$ is multiplied by $F_{tip}$ to get the $F$ used in the inflow equation.

The $C_l$ and $C_d$ at each location is then retrieved from the C81 table assigned to the rotor and the Thrust and Torque calculated. For forward flight the rotor is trimmed by introducing a tilt in the rotor tip-path-plane using $\alpha_{TPP}$, collective and cyclic pitch.

### Tail Rotor Forward Flight Power Calculations

For the calculation of anti-torque power, CONDOR uses a corrected momentum theory for forward flight. Using the torque calculated for the main rotor, the anti-torque requirement is estimated and the power required calculated to produce the thrust required. The following are the power calculations for the tail rotor:

$$T_{tr} = \frac{\tau_{MainRotor}}{l_{tr}}\ lbs$$

Where:

$l_{tr} = tail\ rotor\ distance\ from\ c.g\ in\ ft$

$$IHP_{tr} = \frac{1.13 T_{tr} v_{tr} K_u}{550}\ hp$$

$$v_{tr} = induced\ velocity = \sqrt{T_{tr}/2\rho A}$$

$$A_{tr} = Tail\ Rotor\ disk\ area$$

$$K_u = \sqrt{\frac{-(V_\infty/v)^2 + \sqrt{(V_\infty/v)^4 + 4}}{2}} = Correction\ factor\ (1\ in\ hover)$$

1.13 empirical correction to account for losses

550 converts lbfts$^{-1}$ to hp

$$RHP_{tr} = \frac{C_{D0}\rho A \sigma V_T^3}{550 * 8}(1 + 4.65\mu^2)\ hp$$

$$V_{Ttr} = Tip\ speed = \Omega Rtr$$

$$\sigma_{tr} = rotor\ solidity = \frac{tail\ rotor\ blade\ area}{tail\ rotor\ disk\ area}$$

$$\mu = Advance\ Ratio = \frac{V_\infty}{V_{Ttr}}\ (0\ in\ hover)$$

$$550\ covnerts\ lbfts^{-1}\ to\ hp$$

# 6: Validation

CONDOR is validated using available flight and power data for various single main rotor helicopters with different missions. The missions listed below were set with available knowledge of suggested cruise altitude and velocity to compare gross weight estimates in the RF method as well as power characteristics over the flight regime to check max velocity calculations gotten through the BEMT code.

- Mission 1 (Airline Configuration): Take off and climb → cruise to destination (Max Range Speed, max payload (5000 lbs) → Descend and Land
  - S92: Calculated= 28,974 lbs, Actual = 26,500 lbs, %diff = 9.3%
- Mission 2 (Search and Rescue): Takeoff, 6 min SL hover → Cruise (best Range Speed) → Hover 10 Min (zero altitude + pick up 6 survivors) → Cruise (Best Range Speed) → 6 min hover and land
  - S92: Calculated = 25,688 lbs, Actual = 26,500, %diff = 3.0%

Further Validation is planned.

In order to validate the blade element code several power curves were created to compare the power required at different conditions. The following power curves were created with comparison values of the actual vehicles.

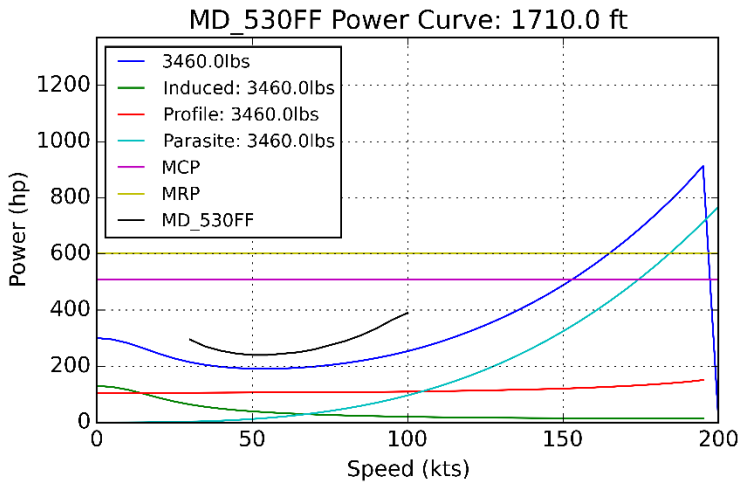


*Figure 3. MD 530FF Power Curve Comparison*

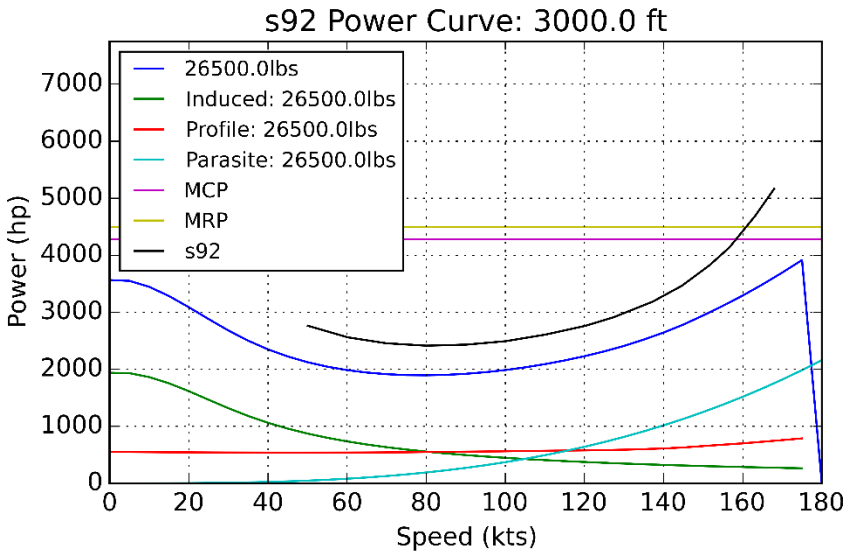


*Figure 4. S92 Power curve Comparison*

## 7: Limitations

There are several known limitations and areas for potential improvement to the code. They are as follows.

1. The look up tables do not take into account changes in Reynolds number over the blade span. There is a compressibility correction built in, however, this is inherently less accurate than using a better lookup table which varies over Reynolds numbers.
2. Weight build up is done through simple models which can be seen in the *<vehicle.py>* code in the *<scaleWeights>* definition.
3. Due to the fact that no hinge offset is provided in the inputs and in order to simplify the code, there is not a moment trim only a force trim.
4. CONDOR does not handle compound helicopter design. Compounding in terms of multiple rotors, propellers, and wings can be introduced using lift and thrust sharing. Considerable modifications to the trim codes would be required. If multiple rotors such as intermeshing and coaxial are used, advanced inflow models with rotor interference would be required.