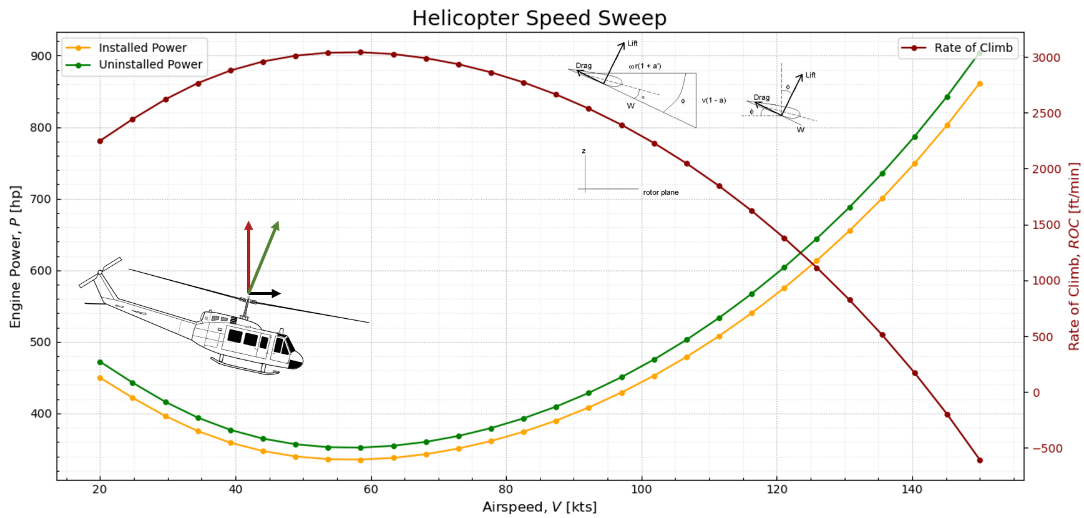
heliPypter Documentation

Release 0.0.7

Benjamin Crews

Apr 20, 2021

Contents:



heliPypter is a package for rotorcraft performance evaluation. Rotorcraft attributes are provided as input, and performance characteristics such as Engine Horsepower, Specific Range, and Fuel consumption are evaluated.

heliPypter has an object oriented philosophy, so different rotorcraft configurations can be built, modified, and evaluated quickly, with the same methods. The classes have methods for Hover in and out of Ground Effect (HOGE and HIGE), as well as forward flight.

Under the hood, briefly speaking, the code applies Momentum Theory assuming constant chord ideal twist. Correction factors can be supplied to align results with tapered, linearly twisted blades. For forward flight, Glauert's Model is used.

CHAPTER 1

Installation

For most sytems, the easiest way to install this package is through the Python Package Index:

` $ pip install helipypter `

heliPypter requires pandas and scikit-aero, which should be automatically installed by pip, if you need them.

The project homepage is *here <https://github.com/Czarified/helipypter>*.

CHAPTER 2

Analysis With heliPypter

2.1 Basic Flight Performance

Using heliPypter, performance for a traditional helicopter with single main and tail rotors can be evaluated. The first step is defining all the inputs (there are many). The details of all inputs are fully documented on the API page.

Warning: Units are important, so make sure they are all Imperial! Metric and automated units with Pint may be supported in a future release. If you want it, post on the issues page.

The Helicopter class takes numeric weight values for fuel, and a single lumped value for all other masses. It then adds the remaining fuel weight and empty mass whenver you call the heli.GW property. Let's use an empty weight fraction to generate this helicopter.

```
import helipypter.vehicles as vh

# Empty weight fraction
EW_frac = 0.528
# Total Gross Weight
GW_total = 5000
# Crew Weight
w_crew = 200
# Trapped Fluids
w_fluids = 13

w_empty = EW_frac*GW_total + w_crew + w_fluids
# Our payload is 6 people @ 213 lbs each
w_payload = 6*213
w_fuel = GW_total - w_empty - w_payload

doc_chopper = vh.Helicopter(name='Documentation Helicopter Spec',
                            MR_dia = 35,
                              MR_b = 4,
```

(continues on next page)

(continued from previous page)

```
                            MR_ce = 10.4,
                         MR_Omega = 43.2,
                           MR_cd0 = 0.0080,
                           TR_dia = 5.42,
                             TR_b = 4,
                            TR_ce = 7,
                         TR_Omega = 239.85,
                           TR_cd0 = 0.015,
                         GW_empty = w_empty,
                          GW_fuel = w_fuel,
                       GW_payload = w_payload,
                         download = 0.03,
                               fe = 12.9,
                           l_tail = 21.21,
                             S_vt = 20.92,
                            cl_vt = 0.22,
                            AR_vt = 3
                                 )
```

Note: The Main Rotor Blade incompressible minimum drag, *MR_cd0*, is a vehicle characteristic. If we could clean up this blade drag term, it would logically affect all flight performance, so it's included here in the base definition of the vehicle.

The same goes for the airframe equivalent flat-plate drag, *fe*. If we were to perform an airframe drag cleanup design cycle on our vehicle, we can reduce this term here, or scale it however you want.

The Helicopter class has many default values. Some aren't shown here, so it's always good idea to check the vehicle definition using a simple print function.

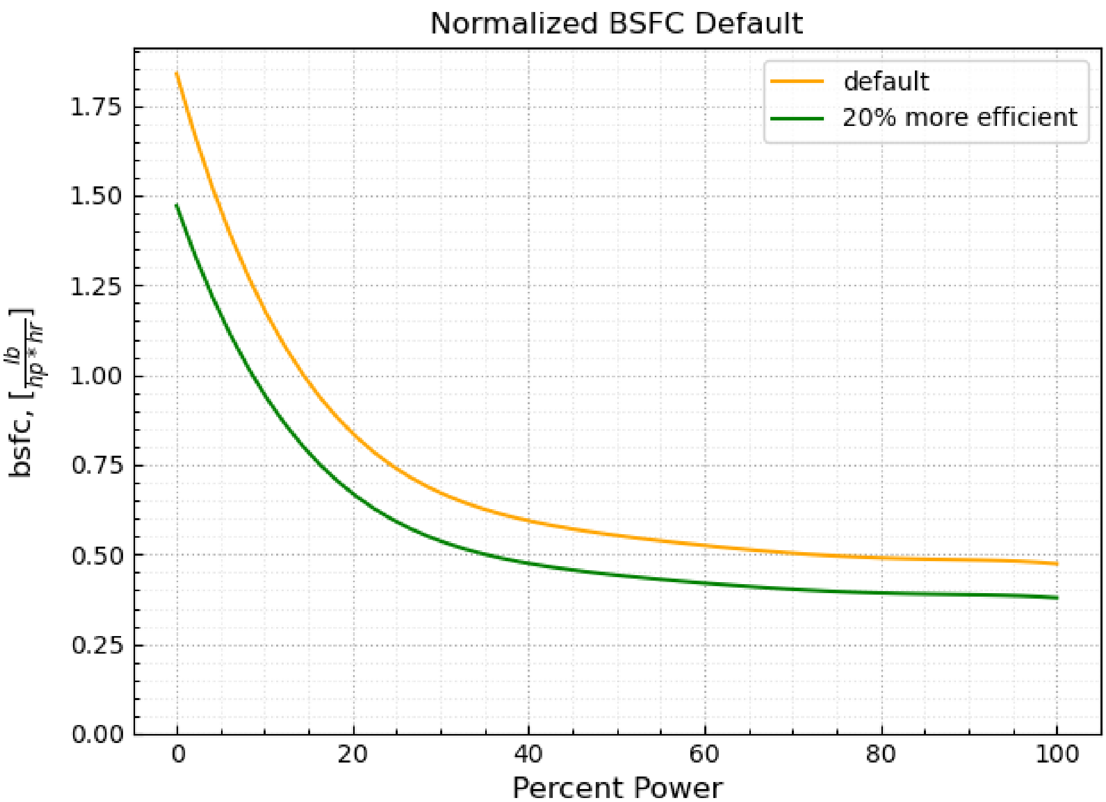
```
print(doc_chopper)

-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
       Documentation Helicopter Spec
Rotors: ('MR', 'TR')
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
Main Rotor Inputs:
           MR_dia:  35.000 [ft]
             MR_b:   4.000 []
            MR_ce:  10.400 [in]
         MR_Omega:  43.200 [rad/s]
           MR_cd0:   0.008 []
             MR_R:  17.500 []
             MR_A: 962.113 []
          MR_vtip: 756.000 []
           MR_sol:   0.063 []
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
Tail Rotor Inputs:
           TR_dia:   5.420 [ft]
             TR_b:   4.000 []
            TR_ce:   7.000 [in]
         TR_Omega: 239.850 [rad/s]
           TR_cd0:   0.015 []
             TR_R:   2.710 []
             TR_A:  23.072 []
          TR_vtip: 649.993 []
```

(continues on next page)

(continued from previous page)

```
          TR_sol:   0.274 []
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
Airframe Data:
        GW_empty: 2853.000 [lbs]
         GW_fuel: 869.000 [lbs]
      GW_payload: 1278.000 [lbs]
        download:   0.030 [.%]
     HIGE_factor:   1.200 []
              fe:  12.900 [ft2]
          l_tail:  21.210 [ft]
            S_vt:  20.920 [ft2]
           cl_vt:   0.220 []
           AR_vt:   3.000 []
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
Engine Data:
      eta_MRxsmn:   0.985 [.%]
      eta_TRxsmn:   0.971 [.%]
     eta_xsmn_co:   0.986 [.%]
        eta_inst:   0.950 [.%]
       xsmn_lim: 674.000 [hp]
        pwr_lim: 813.000 [hp]
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
```

Note: Not shown here are the engine Brake-Specific Fuel Consumption factors. Four factors can be provided, defining a polynomial function to return the bsfc, in *[lbs/(hp*hr)]*. See `helipypter.vehicles.Helicopter` method *Helicopter.bsfc*.



The heli object can now be called to hover, burn fuel, idle, lookup engine power, or fly. However, before we can perform any flight maneuvers, atmospheric properties must be supplied. Here, we create an Environment class. For example, to create a Sea-level standard atmosphere and hover at it:

```
atm = vh.Environment(alt=0)

output = heli.HOGE(atm)
print('-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-')
print('{:^45}'.format('Results - HOGE'))
print('-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-')
for k,v in doc_chopper.HOGE(atm).items():
    print('{:>17}:   {:>7.4}'.format(k, v))
```

Hover Out of Ground Effect (HOGE) returns dictionary of the flight point predictions. Sometimes, dictionary output isn't the easiest to read, even though it's easy to lookup. So we created a simple loop to print the data.

```
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
            Results - HOGE
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
                a:     5.717
          delta_0:   0.009518
               Ct:   0.003937
        TR_thrust:     291.1
             Cq_i:   0.0001787
             Cq_v:       0.0
             Cq_0:   7.502e-05
             Cq_1:  -1.037e-05
```

(continues on next page)

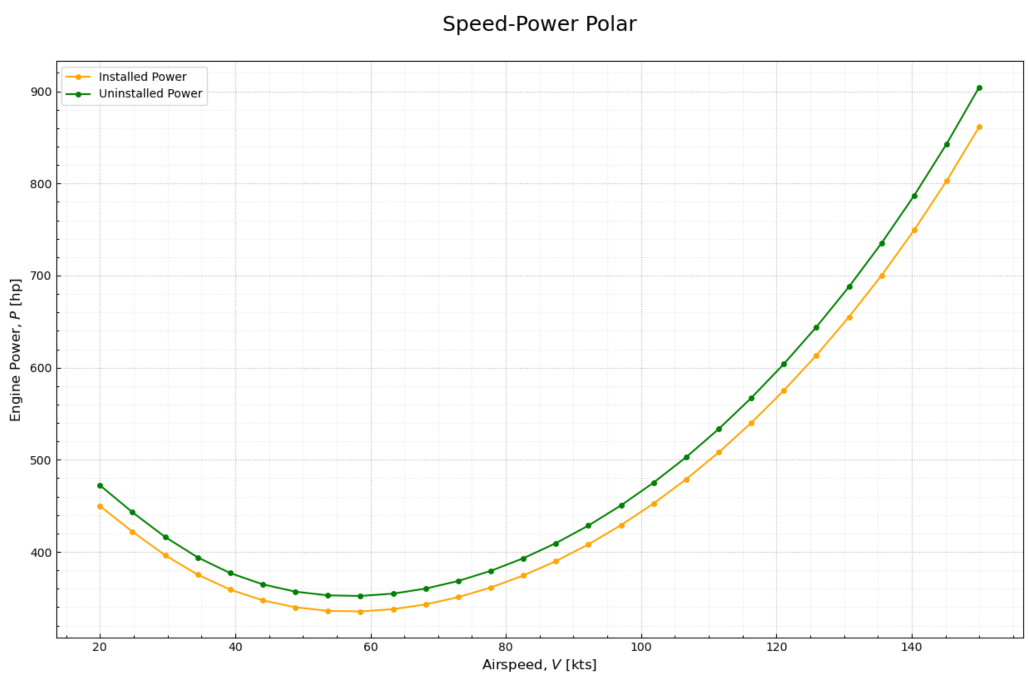
(continued from previous page)

```
          Cq_2:   1.317e-05
            Cq:   0.0002565
             Q:   6.174e+03
          P_MR:   2.425e+05
         HP_MR:     485.0
         HP_TR:      45.3
       SHP_ins:     566.0
     SHP_unins:     595.8
           sfc:    0.4982
```

Note: One of the optional inputs to the HOGE method is *k_i*. This number is the correction factor for non-uniform inflow, linear twist, and taper. It's defaulted to 1.1, and will typically be between 1 and 1.15.

Forward flight performance can be evaluated just as easily. Let's perform a speed sweep from 20 knots to 150 knots. The forward_flight method just takes an Environment for atmospheric properties, and either a single or list of airspeeds. This method returns a pandas dataframe that has several columns. It's sometimes hard to view this data, so heliPypter has convenient plotting functions.

```
import numpy as np
import helipypter.funcs as func

# Create an array of 14 equally spaced airspeed values
# This is just a little shorthand and not necessary.
speeds = np.linspace(20, 150, num=14)

data = doc_chopper.forward_flight(atm, speeds)

fig, ax = func.speed_power_polar(data)
```



There's lots of other data in this dataframe, and built-in functions exist to plot range and rate-of-climb. For now we'll stop here and move on to mission analysis.

2.2 Mission Analysis

The first step here is obviously to create a mission. Currently, there's no built-in classes representing a mission, because the contents of a mission are a simple collection of *mission points*, where each point has maneuver inputs. This data structure is very easily represented as a *namedtuple*. You can decide how you want to approach the specifics of mission analysis, this just one example. All helipypter classes should be flexible enough to fit your needs.

Note: In the future, this may change with some built-in missions, or a slightly different structure to make aircraft sizing straight-forward. At the time of creation, this was enough for me and I didn't need to bother with the overhead of a custom class.

```
from collections import namedtuple

Point = namedtuple('MissionPoint', ['maneuver', 'altitude', 'duration', 'speed'])
startup = Point(maneuver='idle', altitude=0, duration=1, speed=0)
takeoff = Point(maneuver='IRP', altitude=0, duration=1, speed=0)
climb_0 = Point('MCP', 0, 5, 1000)
cruise_0 = Point('flight', 5000, 160, 110)
hover_1 = Point('hover', 0, 1, 0)
loiter = Point('loiter', 5000, 10, 60)
unload = Point('unload', 0, 5, 1278)
ground = Point('idle', 0, 1, 0)

mission = (startup, takeoff,
       climb_0, cruise_0, loiter,
       hover_1, unload, hover_1,
       climb_0, cruise_0,
       hover_1, ground
      )
```

We've got a mission now, let's create a function to run the helicopter through the mission, burning fuel and changing weight as we go. We'll just use logging to print everything out to the console. If you have multiple missions and vehicles and you want to compare performance across them, you'll probably want to write all this data to anoter dataframe or dictionary.

This is a lot of clunky code. I'm sure it can be written to be more pythonic. Most of it is just our logging statements, though. Essentially, we step through the mission and evaluate each point, determining the fuel required, removing that fuel weight from the total fuel weight, and logging the results.

```
import logging

def mission_loop(heli, mission):
'''This temp function performs all the logic to simulate the fuel burn of a mission.''
↪'
    # Mission Loop
    logging.info('')
    logging.info('')
    logging.info('-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-')
    logging.info('{:^45}'.format('Project Spec Mission'))
    logging.info('-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-')
```

(continues on next page)

(continued from previous page)

```
    # Initialize the range tracker
    mission_range = 0
    for point in mission:
        if point.maneuver == 'idle':
            fuel = heli.idle()/60 * point.duration
            heli.burn(fuel)
            logging.info(f'Idled for {point.duration}[mins].')
            logging.info(f'   Burned {fuel:.2f}[lbs] of fuel.')
            logging.info(f'   New GW = {heli.GW:.2f}[lbs], fuel: {heli.GW_fuel:.2f}')
            logging.info('')

        elif point.maneuver == 'hover':
            # Actually calculate the fuel cost for
            # hovering at an exact weight and altitude
            data = heli.HOGE(vh.Environment(point.altitude))
            fuel = data['sfc']*data['SHP_unins']*point.duration/60
            heli.burn(fuel)
            logging.info(f'Hovered for {point.duration}[mins], burning {fuel:.2f}
↪[lbs] of fuel.')
            logging.info(f'   New GW = {heli.GW:.2f}[lbs], fuel: {heli.GW_fuel:.2f}')
            logging.info('')

        elif point.maneuver == 'loiter':
            data = heli.forward_flight(vh.Environment(point.altitude), point.speed)
            fuel = data.SHP_uninst[0]*data.bsfc[0]/60 * point.duration
            heli.burn(fuel)
            logging.info(f'Loitered at {point.speed}[kts] for {point.duration}[mins].
↪')
            logging.info(f'   Burned {fuel:.2f}[lbs] of fuel.')
            logging.info(f'   New GW {heli.GW:.2f}[lbs], fuel: {heli.GW_fuel:.2f}')
            logging.info('')

        elif point.maneuver == 'IRP':
            # IRP is the engine rated limit
            sfc = heli.bsfc(100)
            fuel = sfc*1*heli.pwr_lim/60 * point.duration
            heli.burn(fuel)
            logging.info(f'Ran at IRP for {point.duration}[mins].')
            logging.info(f'   Burned {fuel:.2f}[lbs] of fuel.')
            logging.info(f'   New GW = {heli.GW:.2f}[lbs], fuel: {heli.GW_fuel:.2f}')
            logging.info('')

        elif point.maneuver == 'MCP':
            # MCP is defined as 95% of IRP
            sfc = heli.bsfc(95)
            fuel = sfc*0.95*heli.pwr_lim/60 * point.duration
            heli.burn(fuel)
            logging.info(f'MCP Climb for {point.duration}[mins] @ {point.speed}[ft/
↪min].')
            logging.info(f'   Burned {fuel:.2f}[lbs] of fuel.')
            logging.info(f'   New GW = {heli.GW:.2f}[lbs], fuel: {heli.GW_fuel:.2f}')
            logging.info('')
            mission_range += 120*point.duration/60   # 120 kts has more ROC than 1000␣
↪TODO: Calculate this.

        elif point.maneuver == 'flight':
            data = heli.forward_flight(vh.Environment(point.altitude), point.speed)
```

(continues on next page)

(continued from previous page)

```
            fuel = point.duration/data.SR[0]
            heli.burn(fuel)
            logging.info(f'Forward flight for {point.duration}[nm] @ {point.speed}
↪[kts].')
            logging.info(f'    Burned {fuel:.2f}[lbs] of fuel.')
            logging.info(f'    New GW = {heli.GW:.2f}[lbs], fuel: {heli.GW_fuel:.2f}')
            logging.info('')
            mission_range += point.duration

        elif point.maneuver == 'climb':
            # Represents a hover climb/descent NOT @ MCP
            # There's no range credit for a "climb" maneuver instead of an "MCP"␣
↪maneuver.
            data = heli.HOGE(vh.Environment(point.altitude), Vroc=point.speed)
            fuel = data['sfc']*data['SHP_unins']*point.duration/60
            heli.burn(fuel)
            logging.info(f'Climb for {point.duration}[min] @ {point.speed}[ft/min]')
            logging.info(f'    Burned {fuel:.2f}[lbs] of fuel.')
            logging.info(f'    New GW = {heli.GW:.2f}[lbs], fuel: {heli.GW_fuel:.2f}')
            logging.info('')

        elif point.maneuver == 'unload':
            logging.info(f'Landed! Unloading {point.speed}[lbs] of cargo.')
            heli.unload(point.speed)
            fuel = heli.idle()/60 * point.duration
            heli.burn(fuel)
            logging.info(f'Idled for {point.duration}[mins], burning {fuel:.2f}[lbs]␣
↪of fuel.')
            logging.info(f'    New GW = {heli.GW:.2f}[lbs], fuel: {heli.GW_fuel:.2f}')
            logging.info('')

    logging.info('')
    logging.info(f'Mission Complete! {heli.GW_fuel:.2f} [lbs] of fuel remaining.')
    logging.info(f'Total Range = {mission_range:.2f}[nm]')
    logging.info('-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-')

mission_loop(doc_chopper, mission)
```

Results:

```
2020-05-02 21:16:07,914 -  INFO -  -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
2020-05-02 21:16:07,914 -  INFO -              Project Spec Mission
2020-05-02 21:16:07,914 -  INFO -  -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
2020-05-02 21:16:07,914 -  INFO -  Idled for 1[mins].
2020-05-02 21:16:07,914 -  INFO -     Burned 2.27[lbs] of fuel.
2020-05-02 21:16:07,914 -  INFO -     New GW = 4997.73[lbs], fuel: 866.73
2020-05-02 21:16:07,914 -  INFO -
2020-05-02 21:16:07,914 -  INFO -  Ran at IRP for 1[mins].
2020-05-02 21:16:07,914 -  INFO -     Burned 6.42[lbs] of fuel.
2020-05-02 21:16:07,914 -  INFO -     New GW = 4991.31[lbs], fuel: 860.31
2020-05-02 21:16:07,914 -  INFO -
2020-05-02 21:16:07,914 -  INFO -  MCP Climb for 5[mins] @ 1000[ft/min].
2020-05-02 21:16:07,914 -  INFO -     Burned 31.00[lbs] of fuel.
2020-05-02 21:16:07,914 -  INFO -     New GW = 4960.31[lbs], fuel: 829.31
2020-05-02 21:16:07,914 -  INFO -
2020-05-02 21:16:07,944 -  INFO -  Forward flight for 160[nm] @ 110[kts].
2020-05-02 21:16:07,944 -  INFO -     Burned 374.09[lbs] of fuel.
```

(continues on next page)

(continued from previous page)

```
2020-05-02 21:16:07,944 -  INFO -    New GW = 4586.22[lbs], fuel: 455.22
2020-05-02 21:16:07,945 -  INFO -
2020-05-02 21:16:07,974 -  INFO -  Loitered at 60[kts] for 10[mins].
2020-05-02 21:16:07,974 -  INFO -    Burned 32.14[lbs] of fuel.
2020-05-02 21:16:07,974 -  INFO -    New GW 4554.09[lbs], fuel: 423.09
2020-05-02 21:16:07,974 -  INFO -
2020-05-02 21:16:07,975 -  INFO -  Hovered for 1[mins], burning 4.60[lbs] of fuel.
2020-05-02 21:16:07,975 -  INFO -    New GW = 4549.49[lbs], fuel: 418.49
2020-05-02 21:16:07,975 -  INFO -
2020-05-02 21:16:07,975 -  INFO -  Landed! Unloading 1278[lbs] of cargo.
2020-05-02 21:16:07,975 -  INFO -  Idled for 5[mins], burning 11.34[lbs] of fuel.
2020-05-02 21:16:07,975 -  INFO -    New GW = 3260.14[lbs], fuel: 407.14
2020-05-02 21:16:07,975 -  INFO -
2020-05-02 21:16:07,975 -  INFO -  Hovered for 1[mins], burning 3.71[lbs] of fuel.
2020-05-02 21:16:07,975 -  INFO -    New GW = 3256.44[lbs], fuel: 403.44
2020-05-02 21:16:07,975 -  INFO -
2020-05-02 21:16:07,975 -  INFO -  MCP Climb for 5[mins] @ 1000[ft/min].
2020-05-02 21:16:07,975 -  INFO -    Burned 31.00[lbs] of fuel.
2020-05-02 21:16:07,975 -  INFO -    New GW = 3225.44[lbs], fuel: 372.44
2020-05-02 21:16:07,975 -  INFO -
2020-05-02 21:16:08,005 -  INFO -  Forward flight for 160[nm] @ 110[kts].
2020-05-02 21:16:08,005 -  INFO -    Burned 333.85[lbs] of fuel.
2020-05-02 21:16:08,005 -  INFO -    New GW = 2891.59[lbs], fuel: 38.59
2020-05-02 21:16:08,005 -  INFO -
2020-05-02 21:16:08,006 -  INFO -  Hovered for 1[mins], burning 3.48[lbs] of fuel.
2020-05-02 21:16:08,006 -  INFO -    New GW = 2888.11[lbs], fuel: 35.11
2020-05-02 21:16:08,006 -  INFO -
2020-05-02 21:16:08,006 -  INFO -  Idled for 1[mins].
2020-05-02 21:16:08,006 -  INFO -    Burned 2.27[lbs] of fuel.
2020-05-02 21:16:08,006 -  INFO -    New GW = 2885.84[lbs], fuel: 32.84
2020-05-02 21:16:08,006 -  INFO -
2020-05-02 21:16:08,006 -  INFO -
2020-05-02 21:16:08,006 -  INFO -  Mission Complete! 32.84 [lbs] of fuel remaining.
2020-05-02 21:16:08,006 -  INFO -  Total Range = 340.00[nm]
2020-05-02 21:16:08,006 -  INFO -  -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
```

2.3 Technology Factors

During design of an air vehicle it can be advantageous to explore the effects of different technology factors, represented as percent reductions, on the performance. Because of heliPypter's object-oriented approach, changing these inputs is relatively straight-forward.

Using our previously-created helicopter as a base, we can update these values one by one, or all at once, it's really up to you.

```python
import copy

## Reduce the empty weight fraction
EW_factor = 0.95

# Empty weight fraction
EW_frac = 0.528
# Total Gross Weight
GW_total = 5000
```

(continues on next page)

(continued from previous page)

```
# Crew Weight
w_crew = 200
# Trapped Fluids
w_fluids = 13

w_empty = EW_factor*EW_frac*GW_total + w_crew + w_fluids
# Our payload is still 6 people @ 213 lbs each
w_payload = 6*213
w_fuel = GW_total - w_empty - w_payload

lite_args = copy.copy(args)
lite_args[11] = w_empty
lite_args[12] = w_fuel
lite_args[13] = w_payload

# Generate the new vehicle, with all other characteristics the same
lightweight = chopper_gen(lite_args)
out = pd.DataFrame(data=func.missionSim(lightweight, mission), columns=['dist', 'fuel_
↪rem', 'fuel_used'])
print(f'Lite chopper range: {out.dist.sum()}')
print(f'Lite chopper remaining fuel: {out.fuel_rem.iat[-1]:.2f}')


## Reduce the MR_cd0
## Reduce the fe
cd0_factor = 0.95
fe_factor = 0.95

clean_args = copy.copy(args)
clean_args[5] = cd0_factor*clean_args[5]
clean_args[15] = fe_factor*clean_args[15]

clean_chopper = chopper_gen(clean_args)

out = pd.DataFrame(data=func.missionSim(clean_chopper, mission), columns=['dist',
↪'fuel_rem', 'fuel_used'])
print(f'Clean chopper range: {out.dist.sum()}')
print(f'Clean chopper remaining fuel: {out.fuel_rem.iat[-1]:.2f}')



## Reduce the Induced Power Factor
## Increase the fuel efficiency of the engine
eng_fac = 0.97

# Use this k_i when calling Helicopter.hover()
k_i = 1.05

efficient_chopper = copy.copy(doc_chopper)
efficient_chopper.bsfc_0 = eng_fac*efficient_chopper.bsfc_0
efficient_chopper.bsfc_1 = eng_fac*efficient_chopper.bsfc_1
efficient_chopper.bsfc_2 = eng_fac*efficient_chopper.bsfc_2
efficient_chopper.bsfc_3 = eng_fac*efficient_chopper.bsfc_3
efficient_chopper.bsfc_4 = eng_fac*efficient_chopper.bsfc_4
efficient_chopper.bsfc_5 = eng_fac*efficient_chopper.bsfc_5

# Since this one is a copy of the old one
```

(continues on next page)

(continued from previous page)

```
# We've already burned all the fuel and unloaded
# it, so we need to reset the weight values.
efficient_chopper.refuel()
efficient_chopper.reload()

out = pd.DataFrame(data=func.missionSim(efficient_chopper, mission), columns=['dist',
↪'fuel_rem', 'fuel_used'])
print(f'Efficient chopper range: {out.dist.sum()}')
print(f'Efficient chopper remaining fuel: {out.fuel_rem.iat[-1]:.2f}')
```

From here, we can evaluate each verion on the same set of missions, and observe the change in fuel consumption. Changes to the base class aren't limited to the above. A formulaic optimization procedure could be performed on any number of variables for design optimization. Programming this operation is beyond the scope of this analysis, however, and may be included at a later date.

Let's just look at the code output. The mission range was constant (although you could change it, but I would argue the the mission should design the vehicle and not vice versa), but the remaining fuel changes with each iteration.

```
Default chopper range: 340.0
Default chopper remaining fuel: 32.84
Lite chopper range: 340.0
Lite chopper remaining fuel: 164.84
Clean chopper range: 340.0
Clean chopper remaining fuel: 42.34
Efficient chopper range: 340.0
Efficient chopper remaining fuel: 57.55
```

We can see from the above that our design is very sensitive to weight and fuel efficiency. Improvements in these areas would be more effective uses of future design resources than in cleaner aerodynamics.

CHAPTER 3

Theoretical Background

Coming soon! Please be patient, I'm a busy grad student with a big-boy job and can only work on this in my free time.

3.1 References

[1] A. Gessow and G. Jr. Myers, Aerodynamics of the Helicopter, 8th ed. College Park Press, 1985. [2] C. N. Keys, Rotary-Wing Aerodynamics Performance Prediction of Helicopters, vol. 2, 2 vols. Philadelphia, Pennsylvania: NASA Scientific and Technical Information Office, 1979.

CHAPTER 4

Python API Documentation

4.1 Vehicles

The main class in helipypter is the Helicopter.

class helipypter.vehicles.**Helicopter**(*name: str = 'Tim the Enchanter', rotors: tuple = ('MR', 'TR'), MR_dia: float = 10, MR_b: int = 2, MR_ce: float = 10.4, MR_Omega: float = 43.2, MR_cd0: float = 0.008, TR_dia: float = 2, TR_b: int = 2, TR_ce: float = 6, TR_Omega: float = 20, TR_cd0: float = 0.015, GW_empty: float = 1000, GW_fuel: float = 0, GW_payload: float = 0, download: float = 0.03, HIGE_factor: float = 1.2, fe: float = 5, l_tail: float = 15, S_vt: float = 15, cl_vt: float = 0.1, AR_vt: float = 3, eta_MRxsmn: float = 0.985, eta_TRxsmn: float = 0.9712, eta_xsmn_co: float = 0.986, pwr_acc: float = 10, eta_inst: float = 0.95, xsmn_lim: float = 674, pwr_lim: float = 813, bsfc_0: float = 1.839, bsfc_1: float = -0.08754, bsfc_2: float = 0.00252, bsfc_3: float = -3.77e-05, bsfc_4: float = 2.822e-07, bsfc_5: float = -8.331e-10*)

This class represents a helicopter with typical design features. These features are:

- Single Main Rotor,
- Single Tail Rotor,
- No shared lift or forward thrusters

The basic Helicopter class has attributes and properties (according to Python definitions). Defaults are set for all values, so be careful with results before checking that all your input values (and units) are correct!

This class uses a constant equivalent chord for all blades, and an efficiency factor must be used to estimate real performance based on this. No root cut-out is available.

HIGE (*atm*, *Thrust=None*, *delta_1: float = -0.0216*, *delta_2: float = 0.4*, *k_i: float = 1.1*, *Vroc: float = 0*) → list
This method calculates the Hover In Ground Effect performance.

This is simply the HOGE, but with a factored thrust.

HOGE (*atm*, *Thrust=None*, *delta_1: float = -0.0216*, *delta_2: float = 0.4*, *k_i: float = 1.1*, *Vroc: float = 0*) → list
This method calculates Hover Out of Ground Effect performance.

Variables

- **atm** (*class*) – An Environment class object, which provides altitude and temperature.
- **delta_1** (*float*) – The second term in the 3-part drag equation (default to -0.0216 based on literature)
- **delta_2** (*float*) – The third term in the 3-part drag equation (default to 0.4 based on literature)
- **k_i** (*float*) – The "efficiency factor" which includes losses for non-uniform inflow, and non-ideal twist.
- **Vroc** (*float*) – The vertical rate of climb, in ft/min.

Returns a, delta_0, Ct, Cq_i, Cq_v, Cq_0, Cq_1, Cq_2, Q, P_MR, P_TR, SHP_ins, SHP_unins

Return type list(float, float, float, float, float, float, float, float, float, float, float, float, float)

A 3D lift coefficient [cl/rad]

Delta_0 corrected, compressible drag coefficient (1st term in 3-term drag equation)

Ct coefficient of thrust

Cq_i coefficient of torque, induced velocity contribution

Cq_v coefficient of torque, vroc contribution

Cq_0 coefficient of torque, 1st term drag

Cq_1 coefficient of torque, 2nd term drag

Cq_2 coefficient of torque, 3rd term drag

Q Main Rotor Torque

P_MR Main Rotor required Power

P_TR Tail Rotor required Power

SHP_ins Total shaft horsepower of the installed engine

SHP_unins Total shaft horsepower of an uninstalled engine (spec)

bsfc (*pwr*) → float
This method uses the normalized bsfc curve (engine specific).

Variables **pwr** (*float*) – Percent power (eg 47%)

Returns Brake specific fuel consumption (lbs/(hp*hr))

Return type float

burn (*fuel*)
This method burns an amount of fuel, reducing the fuel weight by the amount burned.

forward_flight (*atm*, *Airspeed*) → dict
This function evaluates performance in forward flight. Airspeed (in kts) input can be a single value, or a list of the desired speed sweep.

Performance metrics such as drag, MR power, TR power, Engine power, fuel consumption, and range are evaluated.

TODO, currently df is returned: A dictionary is returned with keys for each characteristic and a list of outputs as values.

get_units (*attribute_name*) → str
Returns the units from the metadata definition of the attribute.

idle (*pwr: float = 20*)
This method evaluates the fuel consumption for ground idle operations. Fuel consumption is returned in units of lb/hr. (This is the default bsfc curve)

A ground idle power setting of 20% is assumed by default.

refuel ()
This method refuels to the fuel capacity. Capacity is defined upon vehicle creation.

reload ()
This method reloads the payload to capacity. Capacity is defined upon vehicle creation.

unload (*weight*)
This method, similar to burn, removes weight from the aircraft by unloading a weight of payload.

Although the “Environment” class isn’t a vehicle, it’s temporarily stored in the vehicles module.

class helipypter.vehicles.**Environment** (*alt: float = 0*)
This class contains all the atmospheric data used in performance calculations. All atmospheric properties are attributes of this class.

Depends on sk-aero.coesa module. Note that only input is the altitude, in feet. All units returned are automatically converted from metric to Imperial.

4.2 Functions

helipypter.funcs.**speed_power_polar** (*data*)
This function generates a standard speed-power polar plot. Input data must have columns following the standard naming convention of the helicopter class.

ie. A dataframe output from the Helicopter.forward_flight method can be directly supplied.

helipypter.funcs.**specific_range** (*data*)
This function generates a standard specific range plot. Input data must have columns following the standard naming convention of the helicopter class.

ie. A dataframe output from the Helicopter.forward_flight method can be directly supplied.

helipypter.funcs.**roc** (*data*)
This function generates a standard rate of climb plot. Input data must have columns following the standard naming convention of the helicopter class.

ie. A dataframe output from the Helicopter.forward_flight method can be directly supplied.

helipypter.funcs.**missionSim** (*heli*, *mission*) → dict
This function runs a helicopter through a mission. For each point, the fuel consumption is evaluated, and the flight distance is evaluated.

Parameters

- **heli** (*Helicopter*) – Helicopter to be analyzed.
- **mission** (*tuple(nametuple)*) – Mission profile to be analyzed.

Returns Mission data table

Return type dict

CHAPTER 5

License

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for

software and other kinds of works.

The licenses for most software and other practical works are designed

to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not

price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you

these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether

gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run

modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.

States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and

modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of

works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this

License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work

in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based

on the Program.

To "propagate" a work means to do anything with it that, without

permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other

parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices"

to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official

standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other

than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all

the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users

can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that

same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of

copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not

convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under

the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological

measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you

receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,

and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to

produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

> a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
>
> b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
>
> c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
>
> d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent

works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms

of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

> a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
>
> b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as

long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded

from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any

tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods,

procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or

specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a

requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided,

in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this

License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option

remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you

add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

> a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
>
> b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
>
> c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
>
> d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
>
> e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
>
> f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further

restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you

must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the

form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly

provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your

license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is

reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the

licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or

run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically

receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an

organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the

rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this

License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free

patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express

agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license,

and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or

arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within

the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting

any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have

permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of

the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the

Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future

versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different

permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY

APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING

WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided

above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest

possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest

to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short

notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school,

if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <https://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program

into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <https://www.gnu.org/licenses/why-not-lgpl.html>.

CHAPTER 6

Attributions

Huey Graphic by Jetijones - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=15743299

Blade Element graphic by Smilesgiles89 - MS Paint, CC BY-SA 3.0, https://en.wikipedia.org/w/index.php?curid=38336902

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Index

B

E

F

G

H

I

M

R

S

U