Gas HALE Model

```
#inPDF:skip

from numpy import pi
from gpkit import VectorVariable, Variable, Model, units
from gpkit.tools import te_exp_minus1
import gpkit
exec gpkit.mdmake("gas_hale_rebuild5.md")
import numpy as np
gpkit.settings['latex_modelname'] = False

class GasPoweredHALE(Model):
    def setup(self):
        constraints = []
```

Flight segment definitions

```
        # define number of segments
        NSeg = 9 # number of flight segments
        NCruise = 2 # number of cruise segments
        NClimb = 2 # number of climb segments
        NLoiter = NSeg - NCruise - NClimb# number of loiter segments
        iCruise = [1,-1] # cuise index
        iLoiter = [3] # loiter index
        for i in range(4,NSeg-1): iLoiter.append(i)
        iClimb = [0,2] # climb index
```

Fuel weight model

- end of first segment weight + first segment fuel weight must be greater
- than MTOW. Each end of segment weight must be greater than the next end
- of segment weight + the next segment fuel weight. The last end segment
- weight must be greater than the zero fuel weight

```
        MTOW = Variable('MTOW', 'lbf', 'max take off weight')
        W_end = VectorVariable(NSeg, 'W_{end}', 'lbf', 'segment-end weight')
        W_fuel = VectorVariable(NSeg, 'W_{fuel}', 'lbf',
                                'segment-fuel weight')
```

```
W_zfw = Variable('W_{zfw}', 'lbf', 'Zero fuel weight')
W_pay = Variable('W_{pay}',10,'lbf', 'Payload weight')
W_avionics = Variable('W_{avionics}', 2, 'lbf', 'Avionics weight')
f_airframe = Variable('f_{airframe}', 0.25, '-', 'airframe weight fraction')
W_airframe = Variable('W_{airframe}', 'lbf', 'airframe weight')
W_begin = W_end.left # define beginning of segment weight
W_begin[0] = MTOW

constraints.extend([MTOW >= W_end[0] + W_fuel[0],
                    W_end[:-1] >= W_end[1:] + W_fuel[1:],
                    W_end[-1] >= W_zfw])
```

Steady level flight model

```
CD = VectorVariable(NSeg, 'C_D', '-', 'Drag coefficient')
CL = VectorVariable(NSeg, 'C_L', '-', 'Lift coefficient')
V = VectorVariable(NSeg, 'V', 'm/s','cruise speed')
rho = VectorVariable(NSeg, r'\rho', 'kg/m^3', 'air density')
S = Variable('S', 16, 'ft^2', 'wing area')
eta_prop = VectorVariable(NSeg, r'\eta_{prop}', np.linspace(0.8,0.8,NSeg), '-',
                          'propulsive efficiency')
P_shaft = VectorVariable(NSeg, 'P_{shaft}', 'hp', 'Shaft power')

constraints.extend([P_shaft >= V*(W_end+W_begin)/2*CD/CL/eta_prop, # + W_begin*h_do
                    0.5*rho*CL*S*V**2 >= (W_end+W_begin)/2])
```

Climb model

```
h_dot = Variable('h_{dot}', 500, 'ft/min', 'Climb rate')

constraints.extend([P_shaft[iClimb]*eta_prop[iClimb]/V[iClimb] >= h_dot*W_begin[iCli
                                                    0.5*rho[iClimb]*V[iClimb]**2*S*CD[iCli
```

Engine Model

```
W_eng = Variable('W_{eng}', 'lbf', 'Engine weight')
W_engtot = Variable('W_{eng-tot}', 'lbf', 'Installed engine weight')
W_engref = Variable('W_{eng-ref}', 4.4107, 'lbf', 'Reference engine weight')
P_shaftref = Variable('P_{shaft-ref}', 2.295, 'hp', 'reference shaft power')
```

```
constraints.extend([W_eng/W_engref >= 0.5538*(P_shaft/P_shaftref)**1.075,
                    W_engtot >= 2.572*W_eng**0.922*units('lbf')**0.078])
```

Weight breakdown

```
constraints.extend([W_airframe >= f_airframe*MTOW,
                    W_zfw >= W_pay + W_avionics + W_airframe + W_engtot])
```

Breguet Range

```
z_bre = VectorVariable(NSeg, 'z_{bre}', '-', 'breguet coefficient')
BSFC = VectorVariable(NSeg, 'BSFC', 'lbf/hr/hp', 'brake specific fuel consumption')
t = VectorVariable(NSeg, 't', 'days', 'time per flight segment')
t_cruise = Variable('t_{cruise}', 0.5, 'days', 'time to station')
t_station = Variable('t_{station}', 'days', 'time on station')
R = Variable('R', 200, 'nautical_miles', 'range to station')
g = Variable('g', 9.81, 'm/s^2', 'Gravitational acceleration')

constraints.extend([z_bre >= V*t*BSFC*CD/CL/eta_prop,
                    R <= V[iCruise]*t[iCruise],
                    t[iLoiter] >= t_station/NLoiter,
                    t[iCruise[0]] <= t_cruise,
                    W_fuel/W_end >= te_exp_minus1(z_bre, 3)])
```

BSFC model

- BSFC data was taken from http://www.3w-international.com/Drone_Engines_Sale/engine-details-test-data/engine-data-3W-28i-HFE-FI.php

```
J = Variable('J', 0.127, '1/radians', 'advance ratio on propellors')
RPM = VectorVariable(NSeg, 'RPM', 'rpm', 'rotational speed')
R_prop = Variable('R_{prop}', 9, 'inches', 'Prop radius')
RPM_ref = Variable('RPM_{ref}', 3000, 'rpm', 'reference rotational speed')
rho_fuel = Variable(r'\rho_{fuel}', 6.01, 'lbf/gallon', 'density of 100LL')
mdot_fuel = VectorVariable(NSeg, r'\dot{m}_{fuel}', 'cm^3/hr', 'fuel consumption')
mdot_fuelref = Variable(r'\dot{m}_{fuel-ref}', 360, 'cm^3/hr',
                        'fuel consumption reference')
Tau = VectorVariable(NSeg, r'\Tau', 'N*m', 'torque')
Tau_ref = Variable(r'\Tau_{ref}', 0.55, 'N*m', 'reference torque')
```

```
constraints.extend([#J == V/RPM/R_prop,
                    Tau <= 3*units('N*m'),
                    Tau/Tau_ref >= 0.9143*(RPM/RPM_ref)**1.5663,
                    P_shaft <= Tau*RPM,
                    mdot_fuel/mdot_fuelref >= 0.9164*(RPM/RPM_ref)**1.5462,
                    BSFC >= mdot_fuel*rho_fuel/P_shaft,
                    RPM >= 2000*units('rpm'),
                    ])
```

Aerodynamics model

```
Cd0 = Variable('C_{d0}', 0.02, '-', 'Non-wing drag coefficient')
CLmax = Variable('C_{L-max}', 1.5, '-', 'Maximum lift coefficient')
e = Variable('e', 0.9, '-', 'Spanwise efficiency')
AR = Variable('AR', 20, '-', 'Aspect ratio')
b = Variable('b', 'ft', 'Span')
mu = Variable(r'\mu', 1.5e-5, 'N*s/m^2', 'Dynamic viscosity')
Re = VectorVariable(NSeg, 'Re', '-', 'Reynolds number')
Cf = VectorVariable(NSeg, 'C_f', '-', 'wing skin friction coefficient')
Kwing = Variable('K_{wing}', 1.3, '-', 'wing form factor')
cl_16 = Variable('cl_{16}', 0.0001, '-', 'profile stall coefficient')

constraints.extend([CD >= Cd0 + 2*Cf*Kwing + CL**2/(pi*e*AR) + cl_16*CL**16,
                    b**2 == S*AR,
                    CL <= CLmax,
                    Re == rho*V/mu*(S/AR)**0.5,
                    Cf >= 0.074/Re**0.2])
```

Atmosphere model

- http://en.wikipedia.org/wiki/Density_of_air#Altitude

```
h = VectorVariable(NSeg, 'h', 'ft', 'Altitude')
gamma = Variable(r'\gamma',1.4,'-', 'Heat capacity ratio of air')
p_sl = Variable('p_{sl}', 101325, 'Pa', 'Pressure at sea level')
T_sl = Variable('T_{sl}', 288.15, 'K', 'Temperature at sea level')
L_atm = Variable('L_{atm}', 0.0065, 'K/m', 'Temperature lapse rate')
T_atm = VectorVariable(NSeg, 'T_{atm}', 'K', 'Air temperature')
a_atm = VectorVariable(NSeg, 'a_{atm}','m/s', 'Speed of sound at altitude')
R_spec = Variable('R_{spec}', 287.058,'J/kg/K', 'Specific gas constant of air')
TH = (g/R_spec/L_atm).value.magnitude  # dimensionless
```

```
        constraints.extend([#h <= [20000, 20000, 20000]*units.m,  # Model valid to top of t
                            T_sl >= T_atm + L_atm*h,     # Temp decreases w/ altitude
                            rho == p_sl*T_atm**(TH-1)/R_spec/(T_sl**TH)])
```

altitude constraints

```
        h_station = Variable('h_{station}', 15000, 'ft', 'minimum altitude at station')
        h_min = Variable('h_{min}', 5000, 'ft', 'minimum cruise altitude')

        constraints.extend([h[iLoiter] >= h_station,
                            h[iCruise] >= h_min,
                            h[iClimb] >= h_min,
                            t[iClimb[0]]*h_dot == h_min,
                            t[iClimb[1]]*h_dot == 10000*units('ft'),
                            ])
```

wind speed model

```
        V_wind = VectorVariable(NLoiter, 'V_{wind}', [20,25,30,10,15], 'm/s', 'wind speed')

        constraints.extend([V[iLoiter] >= V_wind])
```

Conclusion

```
        objective = 1/t_station
        return objective, constraints

if __name__ == '__main__':
    M = GasPoweredHALE()
    M.solve()
```

Solution

```
#inPDF: replace with sol.generate.tex
    with open("sol.generated.tex", "w") as f:
        f.write(M.solution.table(latex=True))
```