01_01_Phugoid_Theory

Hal Snyder

8/31/2014

Contents


```
%auto
typeset_mode(true)
```

1 Phugoid Theory as SageMathCloud worksheet

Based on `numerical-mooc/lessons/01_phugoid/01_01_Phugoid_Theory.ipynb` from `MAE6286`.

Numbers in parentheses are equation numbers in the IPython notebook.

```
# (1) equation of lift

# L      lift force
# S      surface area
# rho    density of air
# v      forward velocity
# C_L    coefficient of lift

%var L,S,rho,v, C_L
eq_L = L == C_L * S * (1/2) * rho * v^2
eq_L
```

$$L = \frac{1}{2}\,C_L S \rho v^2$$

```
# (2) equation of drag


# D      drag force
# C_D    coefficient of drag

%var D,C_D
eq_D = D == C_D * S * (1/2) * rho * v^2
eq_D
```

$$D = \frac{1}{2}\,C_D S \rho v^2$$

```
# (3) equation of force perpendicular to the trajectory

# W     weight of airplane
# theta glide angle

%var W,theta
eq_fprp = L == W * cos(theta)
eq_fprp
```

$$L = W \cos(\theta)$$

```
# (3) equation of force parallel to the trajectory

eq_fpar = D == W * sin(theta)
eq_fpar
```

$$D = W \sin(\theta)$$

```
# (4) at trim velocity, lift matches weight

# v_t   trim velocity (when L==W)

%var v_t
eq_L2 = eq_L.subs(v = v_t,L = W)
eq_L2
```

$$W = \frac{1}{2} C_L S \rho v_t^2$$

```
# (5) lift ratio as function of flight velocity

eq_lr = eq_L / eq_L2
eq_lr
```

$$\frac{L}{W} = \frac{v^2}{v_t^2}$$

```
# (6) balance centripetal force from curve of plane's path and gravity

# g     acceleration of Earth's gravity
# R     radius of curvature of trajectory

%var g,R

eq_gbal = (L - W * cos(theta) == (W / g) * (v^2 / R)).add_to_both_sides(W\
    * cos(theta))
eq_gbal
```

$$L = W \cos(\theta) + \frac{W v^2}{R g}$$

```
# (7) phugoid equation in terms of velocity

eq_phv = (eq_gbal / W).subs_expr(eq_lr).factor().expand().\
   add_to_both_sides(- cos(theta))
```

```
eq_phv
```

$$\frac{v^2}{v_t^2} - \cos(\theta) = \frac{v^2}{Rg}$$

```
# (8) simplify - no friction, lift does no work, total energy is constant
# also set zero energy point at reference horizontal (z == 0)

# z     depth of plane below reference horizontal

%var z,z_t

eq_ze = (1/2) * v^2 - g * z == 0
eq_ze
eq_zt = eq_ze.subs(v = v_t, z = z_t)
eq_zt
```

$$\frac{1}{2}\,v^2 - gz = 0$$

$$\frac{1}{2}\,v_t^2 - gz_t = 0$$

```
# rearrange z equation

eq_ze2 = eq_ze.solve(v^2)[0]
eq_ze2
```

$$v^2 = 2\,gz$$

```
# rearrange z_t equation

eq_zt2 = eq_zt.solve(v_t^2)[0]
eq_zt2
```

$$v_t^2 = 2\,gz_t$$

```
# rewrite phugoid equation in terms of z, step 1

eq_p2 = eq_phv.subs_expr(eq_ze2)
eq_p2
```

$$\frac{2\,gz}{v_t^2} - \cos(\theta) = \frac{2\,z}{R}$$

```
# (9) rewrite phugoid equation in terms of z, step 2

eq_phz = (eq_p2 * eq_zt2).expand().subs_expr(eq_zt2).multiply_both_sides\
   (1/(2*g*z_t)).expand()
eq_phz
```

$$\frac{z}{z_t} - \cos(\theta) = \frac{2\,z}{R}$$

```
# treat infinitesimals naively

# (10) diff eq for glide angle vs trajectory length

# ds     tiny arc length of trajectory
# dth    tiny glide angle

%var ds
dth = var('dth',latex_name = "d\\theta")

eq_dthds = 1 / R == dth/ds
eq_dthds
```

$$\frac{1}{R} = \frac{d\theta}{ds}$$

```
# (10) diff eq for depth below horizontal vs trajectory length

# dz     tiny depth below horizontal

%var dz

eq_dzds = sin(theta) == - dz/ds
eq_dzds
```

$$\sin(\theta) = -\frac{dz}{ds}$$

```
# (11) diff eq for glide angle vs depth below horizontal

# chain rule is multiplication of infinitesimals

eq_dthdz = (eq_dthds / eq_dzds)
eq_dthdz
```

$$\frac{1}{R \sin(\theta)} = -\frac{d\theta}{dz}$$

```
# (12) multiply phugoid equation (9) by 1/(2*sqrt(z))

eq_phz2 = eq_phz.multiply_both_sides(1/(2*z^(1/2))).expand()
eq_phz2
```

$$-\frac{\cos(\theta)}{2\sqrt{z}} + \frac{\sqrt{z}}{2 z_t} = \frac{\sqrt{z}}{R}$$

```
# (13) substitute for 1/R in (12)

# split this step to avoid long line in worksheet
eq_phz3a = eq_phz2.subs(eq_dthdz.multiply_both_sides(sin(theta)))
eq_phz3b = eq_phz3a.add_to_both_sides((cos(theta)/(2*z^(1/2))))
eq_phz3b
```

$$\frac{\sqrt{z}}{2 z_t} = -\frac{d\theta \sqrt{z} \sin(\theta)}{dz} + \frac{\cos(\theta)}{2\sqrt{z}}$$

```
# (14) rewrite (13) as an exact derivative

# theta becomes a function of z instead of a variable

theta = function('theta',z)
theta
```

$$\theta(z)$$

```
# (14) continued

# g is the function whose exact derivative appeared in (13)

g = function('g',z)

eq_g = g == z ^ (1/2) * cos(theta)
eq_g
eq_g.derivative(z)

# cosmetics

%var dg
eq_g.derivative(z).subs(derivative(g,z) == dg / dz).subs(derivative(theta\
    ,z) == dth/dz)
```

$$g(z) = \sqrt{z}\cos(\theta(z))$$

$$D[0](g)(z) = -\sqrt{z}\sin(\theta(z))\,D[0](\theta)(z) + \frac{\cos(\theta(z))}{2\sqrt{z}}$$

$$\frac{dg}{dz} = -\frac{d\theta\sqrt{z}\sin(\theta(z))}{dz} + \frac{\cos(\theta(z))}{2\sqrt{z}}$$

```
### Next I want to redo steps (10)-(14) using differential equations
### instead of infinitesimals

### to be continued!
```