## נספח ב – תדפיס של קובץ ההרצות rocket.py

# coding: utf-8

"""

Graphs for the work

"""

from rocket\_project import \* # import everything from rocket\_project.py

from matplotlib import pyplot as plt

import numpy as np

# %% Rho(z) - fig 3

plt.figure()

z = np.arange(0, 20e3)

plt.plot(z/1e3, list(map(Rho, z)))

plt.grid(True)

plt.xlabel("Altitude [km]")

plt.ylabel("Air Density [kg/m^3]")

# %% G(z) - not in the work

plt.figure()

z = np.arange(0, 1e7, 1e4)

plt.plot(z/1e3, list(map(Gravity, z)))

plt.grid(True)

plt.xlabel("Altitude [km]")

plt.ylabel("Gravity [N]")

# %% Comparing Angles - Figure 4

plt.figure()

theta0\_range = [22.5, 45, 45+22.5]

for theta0 in theta0\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=1,

Cd=0, Cl=0, Sd=0, Sl=0, dmdt=0)

values = set\_initial\_values(x=0, z=0, velocity=10,

theta\_velocity\_degree=theta0,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0, t\_final=3,

dt=1e-4)

plt.plot(results.x, results.z, 'o', label=str(theta0))

t = results.index.values

plt.plot(values['vx'] \* t, values['vz'] \* t - 0.5 \* 9.80665 \* (t\*\*2),'k-',

linewidth=2,

label=str(theta0) + " theory")

plt.legend()

plt.grid(True)

plt.xlabel("X [m]")

plt.ylabel("Z [m]")

# %% Comparing dt - not in the work

plt.figure()

dt\_range = np.logspace(-1, -3, 3)

for dt in dt\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=10, m0=1,

Cd=0, Cl=0, Sd=0, Sl=0, dmdt=0)

values = set\_initial\_values(x=0, z=0, velocity=10, theta\_velocity\_degree=45,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0, t\_final=3000,

dt=dt)

plt.figure(num='rocket simulation')

plt.subplot(2, 2, 1)

results.x.plot()

plt.subplot(2, 2, 2)

results.z.plot()

plt.subplot(2, 1, 2)

plt.plot(results.x, results.z)

plt.legend(dt\_range)

# %% Tsiolkovsky rocket equation - Figure 5

plt.figure()

res = []

gas\_velocity = 10

fuel\_mass = 1

body\_mass = 1

dmdt\_range = np.arange(2, 100)

for dmdt in dmdt\_range:

parameters = set\_parameters(theta\_rocket\_degree=90,

gas\_velocity=gas\_velocity, m0=1,

Cd=0, Cl=0, Sd=0, Sl=0, dmdt=dmdt)

values = set\_initial\_values(x=0, z=10, velocity=0, theta\_velocity\_degree=0,

fuel\_mass=1, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0,

t\_final=2, dt=1e-4)

res.append(results.vz.max())

time\_till\_end\_of\_fuel = fuel\_mass/dmdt\_range

maximal\_velocity = gas\_velocity \* np.log(2/1) - 9.8 \* time\_till\_end\_of\_fuel

plt.plot(dmdt\_range, res, 'o', label="simulation")

plt.plot(dmdt\_range, maximal\_velocity, label="theory")

plt.xlabel("dm/dt [kg/sec]")

plt.ylabel("maximal velocity [m/s]")

plt.grid()

plt.legend()

# %% escape velocity - Figure 6

plt.figure()

res = []

for v in [10e3, 50e3]:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=1,

Cd=0, Cl=0, Sd=0, Sl=0, dmdt=0)

values = set\_initial\_values(x=0, z=1, velocity=v, theta\_velocity\_degree=90,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0,

t\_final=20000, dt=1e-2)

res.append(results)

plt.subplot(1, 2, 1)

plt.semilogy(res[0].z)

plt.semilogy(res[1].z)

plt.grid(True)

plt.legend(["v=10,000 m/s", "v=50,000 m/s"])

plt.subplot(1, 2, 2)

plt.plot(res[0].vz)

plt.plot(res[1].vz)

plt.grid(True)

plt.legend(["v=10,000 m/s", "v=50,000 m/s"])

# %% ballistic throw with drag - Figure 7

plt.figure()

res = []

cd\_range = np.linspace(0, 1e-1, 3)

for cd in cd\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=1,

Cd=cd, Cl=0, Sd=1, Sl=0, dmdt=0)

values = set\_initial\_values(x=0, z=0, velocity=10,

theta\_velocity\_degree=45,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0,

t\_final=5, dt=1e-4)

res.append(results)

for cd, r in zip(cd\_range, res):

plt.plot(r.x, r.z, label=f"Cd={cd}")

plt.grid(True)

plt.legend()

plt.xlabel("X [m]")

plt.ylabel("Z [m]")

plt.figure()

for cd, r in zip(cd\_range, res):

plt.plot(r.index, r.z, label=f"Cd={cd}")

plt.grid(True)

plt.legend()

plt.xlabel("time [sec]")

plt.ylabel("Z [m]")

# %% ballistic throw with lift range vs Cd - Figure 8

plt.figure()

res = []

cd\_range = np.arange(0.25, 2, 0.25)

for cd in cd\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=1,

Cd=cd, Cl=0, Sd=1, Sl=0, dmdt=0)

values = set\_initial\_values(x=0, z=0, velocity=136,

theta\_velocity\_degree=60,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0,

t\_final=5, dt=1e-3)

res.append(results)

for cd, r in zip(cd\_range, res):

plt.plot(r.x, r.z, label=f"Cd={cd}")

plt.grid(True)

plt.legend()

plt.xlabel("X [m]")

plt.ylabel("Z [m]")

# %% ballistic throw with lift - Figure 9

plt.figure()

res = []

cl\_range = [0, 0.05, 0.1]

for cl in cl\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=1,

Cd=0, Cl=cl, Sd=0, Sl=1, dmdt=0)

values = set\_initial\_values(x=0, z=0, velocity=10,

theta\_velocity\_degree=45,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0,

t\_final=5, dt=1e-3)

res.append(results)

for cl, r in zip(cl\_range, res):

plt.plot(r.x, r.z, label=f"Cl={cl}")

plt.grid(True)

plt.legend()

plt.xlabel("X [m]")

plt.ylabel("Z [m]")

plt.figure()

for cl, r in zip(cd\_range, res):

plt.plot(r.index, r.z, label=f"Cl={cl}")

plt.grid(True)

plt.legend()

plt.xlabel("time [sec]")

plt.ylabel("Z [m]")

# %% ballistic throw with drag range vs Cd - Figure 10

plt.figure()

res = []

cd\_range = np.arange(0, 1, 0.01)

plt.close('all')

for cd in cd\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=1,

Cd=cd, Cl=0, Sd=1, Sl=0, dmdt=0)

values = set\_initial\_values(x=0, z=0, velocity=10,

theta\_velocity\_degree=45,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0,

t\_final=5, dt=1e-3)

res.append(results.x.tolist()[-1])

plt.plot(cd\_range, res)

plt.xlabel("Cd ")

plt.ylabel("X [m]")

plt.grid(True)

# %% ballistic throw with lift range vs Cl - Figure 11

plt.figure()

res = []

cl\_range = np.linspace(0, 1, 100)

for cl in cl\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=1,

Cd=0, Cl=cl, Sd=0, Sl=1, dmdt=0)

values = set\_initial\_values(x=0, z=0, velocity=10,

theta\_velocity\_degree=45,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0,

t\_final=5, dt=1e-4)

res.append(results.tail(1))

plt.plot(cl\_range, [r.x.tolist()[0] for r in res])

plt.xlabel("Cl ")

plt.ylabel("X [m]")

plt.grid(True)

plt.figure()

plt.plot(results.index, results.x, label="X")

plt.plot(results.index, results.z, label="X")

plt.xlabel("Cl ")

plt.ylabel("meter")

plt.grid(True)

# %% aircraft - Figure 12

plt.figure()

res = []

v\_range = np.linspace(200, 300, 3) \* 1000 / 3600

for v in v\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=78e3,

Cd=0, Cl=2, Sd=0, Sl=122.6, dmdt=0)

values = set\_initial\_values(x=0, z=5, velocity=v,

theta\_velocity\_degree=0,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0, t\_final=2, dt=1e-3)

res.append(results)

for v, r in zip(v\_range, res):

plt.plot(r.index, r.z, label=f"v = {v\*3600/1000} km/h")

plt.xlim([0, 2])

plt.ylim([0, 30])

plt.grid(True)

plt.legend()

plt.xlabel("X [m]")

plt.ylabel("Z [m]")

# %% aircraft - Figure 14 - takeoff velocity vs. lift coefficient

"""

0.5 Cl Sl rho(0) v^2 = m g ==> v = sqrt( 2g/rho(0) ) \* sqrt( m / (Cl Sl) )

"""

plt.clf()

res = []

m0 = 1

sl = 1

z0 = 5

cl\_range = np.arange(1, 10, .1)

for cl in cl\_range:

v\_theory = np.sqrt( 2\*Gravity(0)/Rho(0) ) \* np.sqrt( m0 / (cl\*sl) )

v\_range = np.arange(v\_theory\*0.98, 1.02\*v\_theory, v\_theory/100)

for v in v\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=m0,

Cd=0, Cl=cl, Sd=0, Sl=sl, dmdt=0)

values = set\_initial\_values(x=0, z=z0, velocity=v,

theta\_velocity\_degree=0,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0, t\_final=1e-3, dt=1e-4)

if results.z.tolist()[-1] > z0: # takeoff

res.append(v)

break # no need to check larger v values

plt.plot(cl\_range, res, 'o', label="simulation")

plt.plot(cl\_range, np.sqrt( 2\*Gravity(0)/Rho(0) ) \* np.sqrt( m0/(cl\_range\*sl)),

label="theory")

plt.grid(True)

plt.xlabel("Cl \* Sl")

plt.ylabel("takeoff velocity [m/s]")

plt.legend()

# %% aircraft - Figure 15 - takeoff velocity vs. lift mass

"""

0.5 Cl Sl rho(0) v^2 = m g ==> v = sqrt( 2g/rho(0) ) \* sqrt( m / (Cl Sl) )

"""

plt.clf()

res = []

m0 = 1

cl = 1

sl = 1

z0 = 5

m\_range = np.arange(1, 10, .1)

for m0 in m\_range:

v\_theory = np.sqrt( 2\*Gravity(0)/Rho(0) ) \* np.sqrt( m0 / (cl\*sl) )

v\_range = np.arange(v\_theory\*0.98, 1.1\*v\_theory, v\_theory/100)

for v in v\_range:

parameters = set\_parameters(theta\_rocket\_degree=0, gas\_velocity=0, m0=m0,

Cd=0, Cl=cl, Sd=0, Sl=sl, dmdt=0)

values = set\_initial\_values(x=0, z=z0, velocity=v,

theta\_velocity\_degree=0,

fuel\_mass=0, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0, t\_final=1e-3, dt=1e-4)

if results.z.tolist()[-1] > z0: # takeoff

res.append(v)

break # no need to check larger v values

plt.plot(m\_range, res, 'o', label="simulation")

plt.plot(m\_range, np.sqrt( 2\*Gravity(0)/Rho(0) ) \* np.sqrt( m\_range/(cl\*sl)),

label="theory")

plt.grid(True)

plt.xlabel("mass [kg]")

plt.ylabel("takeoff velocity [m/s]")

plt.legend()

# %% Comparing dmdt

plt.figure()

dmdt\_range = np.logspace(0, 1, 2)

for dmdt in dmdt\_range:

parameters = set\_parameters(theta\_rocket\_degree=90, gas\_velocity=10, m0=1,

Cd=0, Cl=0, Sd=0, Sl=0, dmdt=dmdt)

values = set\_initial\_values(x=0, z=10, velocity=0, theta\_velocity\_degree=0,

fuel\_mass=1, parameters=parameters)

results = Euler\_Cromer(values, parameters, t\_init=0, t\_final=50,

dt=1e-4)

print(results.tail(1))

plt.figure(num='rocket simulation')

plt.subplot(2, 2, 1)

results.x.plot()

plt.subplot(2, 2, 2)

results.z.plot()

plt.subplot(2, 1, 2)

plt.plot(results.x, results.z)

plt.legend(dmdt\_range)

plt.figure()

v = np.sqrt(results.vx\*\*2 + results.vz\*\*2)

v.plot()

results.vx.plot()

results.vz.plot()

RocketSimulation (c) by Roi Dvir

RocketSimulation is licensed under a Creative Commons Attribution NonCommercial-NoDerivatives 4.0 International License.

You should have received a copy of the license along with this work. If not, see <http://creativecommons.org/licenses/by-nc-nd/4.0/>.