import math

import matplotlib.pyplot as plt

import numpy as np

from scipy.interpolate import interp1d

l=0.5

g=0.2

def f(x,t):

return l\*x\*(1-x)+g\*math.sin(t)

def g\_der(x):

return l-2\*l\*x

def k(z,x):

return z\*g\_der(x)

def RKx(x,t,tau):

h = (tau/2)\*f(x,t)

x\_new = x+tau\*f(x+h,t+tau/2)

t\_new = t+tau

return x\_new, t\_new

def RKz(x,z,t,tau):

h = (tau/2)\*k(z,x)

z\_new = z+tau\*k(z+h,x)

t\_new = t+tau

return z\_new, t\_new

tau = 1/100

tmax = 2\*math.pi

N = int(round(tmax/tau))

z=[]

Ndati = 2000

x0=np.linspace(-0.15,1.85,Ndati)

x=[]

t=[0]

#risolvo il sistema di equazioni differenziali

for xp in x0:

x.append([xp])

z.append([0.001])

t=[0]

for i in range(N):

x[-1].append(RKx(x[-1][i],t[i],tau)[0])

a,b = RKz(x[-1][i-1],z[-1][i],t[i],tau)

z[-1].append(a)

t.append(b)

phi2pi = [item[-1] for item in x]

#calcolo la derivata come rapporto di zT/z0

phi2pi\_der = [item[-1]/item[0] for item in z]

#calcolo la derivata come rapporto incrementale

phi2pi\_der\_ri = [(phi2pi[i+1]-phi2pi[i])/0.001 for i in range(1999)]

fig, axs = plt.subplots(2, figsize=(8,8))

axs[0].plot(x0,phi2pi\_der, 'r')

axs[0].title.set\_text('Metodo derivato a lezione')

axs[0].grid()

axs[1].plot(x0[:1999],phi2pi\_der\_ri)

axs[1].title.set\_text('Incremento finito')

axs[1].grid()

plt.savefig('confonto\_metodi\_derivate.png')

plt.show()

#trovo i punti fissi con Newton

F = [y-x for x,y in zip(x0,phi2pi)]

Fint = interp1d(x0, F)

Fder = [y-1 for y in phi2pi\_der]

Fderint = interp1d(x0, Fder)

zer = []

for xst in x0:

#il try catch è per non considerare i punti che non convergono

#non è ottimizzato perché gli scarta quando esplode

#andrebbe calcolata la derivata seconda

try:

for i in range(100):

xp = xst-(Fint(xst)/Fderint(xst))

xst = xp

zer.append(xp)

except:

continue

zer = list(set(zer))

print(zer)

plt.plot(x0,phi2pi)

plt.plot([-0.4,1],[-0.4,1])

plt.plot([-0.4,max(x0)],[0,0], 'k--')

plt.plot([0,0],[-0.4,1], 'k--')

plt.xlim([-0.4,max(x0)])

plt.ylim([-0.4,1])

plt.title('Punti fissi mappa di Poincaré\nx0='+str(zer[0])+' x1='+str(zer[1]))

plt.grid()

plt.savefig('punti\_fissi\_poincare.png')

plt.show()