# Liouville and Boltzmann

Sandra Bustamante

## 1 Collisionless Boltzmann Equation

The continuity equation is

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial w} \cdot (f\dot{w}) = 0 \tag{1}$$

where we can consider $w = (p, q)$ as any arbitrary system of canonical coordinates [1] and $\dot{w}$is$(\dot{p}, \dot{q})$.

In a rotating frame, the canonical coordinates are $w = (x, v)$ and $\dot{w} = (\dot{x}, \dot{v})$ so the continuity equation is given by

$$\begin{aligned}
&\frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{w}} \cdot (f\dot{\mathbf{w}}) = 0 \\
&\frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{x}} \cdot (f\dot{\mathbf{x}}) + \frac{\partial}{\partial \mathbf{v}} \cdot (f\dot{\mathbf{v}}) = 0 \\
&\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot \dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}} \cdot f + \frac{\partial f}{\partial \mathbf{v}} \cdot \dot{\mathbf{v}} + \frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{v}} \cdot f = 0 \\
&\frac{\partial f}{\partial t} + \nabla f \cdot \dot{\mathbf{x}} + \frac{\partial f}{\partial \mathbf{v}} \cdot \dot{\mathbf{v}} = 0.
\end{aligned} \tag{2}$$

where

$$\begin{aligned}
\mathbf{v} &= \dot{\mathbf{x}} \\
\dot{\mathbf{v}} = \ddot{\mathbf{x}} &= -\nabla \mathbf{\Phi} - 2\mathbf{\Omega} \times \dot{\mathbf{x}} - \mathbf{\Omega} \times (\mathbf{\Omega} \times \mathbf{x}) \\
&= -\nabla \mathbf{\Phi}_{eff} - 2\mathbf{\Omega} \times \dot{\mathbf{v}}
\end{aligned}$$

Substituting the last equations into the continuity equation 2 and we get

$$\begin{aligned}
&\frac{\partial f}{\partial t} + \nabla f \cdot \mathbf{v} + \frac{\partial f}{\partial \mathbf{v}} \cdot (-\nabla \Phi_{eff} - 2\mathbf{\Omega} \times \dot{\mathbf{x}}) = 0 \\
&\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f - (2\mathbf{\Omega} \times \mathbf{v} + \nabla \mathbf{\Phi}_{eff}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0
\end{aligned}$$

## 2 Liouville equation and the pendulum

Consider the pendulum of length $l$ and mass $m$ in a gravitational field $g$. The kinetic energy and potential energy are

$$T = \frac{1}{2} m l^2 \dot{\theta}^2 \tag{3}$$

and

$$V = mgl(1 - \cos\theta) \tag{4}$$

### 2.a

Starting from the Lagrangian,

$$\mathcal{L}(\theta, \dot{\theta}, t) \equiv T - V = \frac{1}{2} m l^2 \dot{\theta}^2 - mgl(1 - \cos\theta),$$

we can find the generalized momentum by

$$p \equiv \frac{\partial \mathcal{L}}{\partial \dot{q}} = ml^2 \dot{\theta}.$$

Then the Hamiltonian is

$$\begin{aligned}
\mathcal{H} &\equiv T + V \\
\mathcal{H} &= \frac{1}{2} m l^2 \dot{\theta}^2 + mgl(1 - \cos\theta) \\
\mathcal{H} &= \frac{1}{2} \frac{p^2}{ml^2} + mgl(1 - \cos\theta).
\end{aligned}$$

Hamilton's equations are:

$$\begin{aligned}
\dot{p} &\equiv -\frac{\partial \mathcal{H}}{\partial \theta} = -mgl \sin\theta \\
\dot{q} &\equiv \frac{\partial \mathcal{H}}{\partial p} = \frac{p}{ml^2} = \dot{\theta}
\end{aligned}$$

### 2.b

From the dimensionless energy $\varepsilon = E/mgl$

$$\begin{aligned}
\varepsilon &= \frac{1}{mgl} \left[ \frac{1}{2} m l^2 \dot{\theta}^2 + mgl(1 - \cos\theta) \right] \\
&= \frac{1}{2} \frac{l}{g} \dot{\theta} + (1 - \cos\theta),
\end{aligned}$$

we can find the dimensionless variables as

$$z_1 \equiv \theta$$
$$z_2 \equiv \frac{l}{g}\dot{\theta} = \frac{\dot{\theta}}{\omega^2}$$

where $\omega^2 = g/l$.

Knowing that the equation of motion for this system is

$$\ddot{\theta} = -\frac{g}{l}\sin\theta \tag{5}$$

we can find that the solution for this differential equation will be given by

$$\theta = \theta_0 \cos\omega t$$

By analogy we can find that the solution of the dimensionless pendulum is

$$z_1 = z_{10}\cos\omega\tau$$

From these equation we know that the period can be found out when the inside of the cosine is $2\pi = \omega\tau$ so $\tau_0 = 2\pi/\omega$ where $\tau_0$ is the period in the dimensionless pendulum.

## 2.c

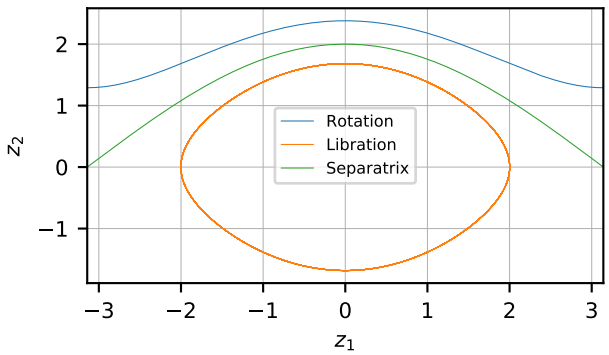


Figure 1: Phase Space of a pendulum in rotation ($\varepsilon > 2$), libration ($\varepsilon < 2$) and separatrix ($\varepsilon = 2$).

| | $z_1$ | $z_2$ | Energy |
|---|---|---|---|
| Rotation | -4.71 | 1.91 | 2.83 |
| Libration | -1.57 | 0.91 | 1.42 |
| Separatrix | -4.71 | 1.41 | 2.00 |

Table 1: Initial conditions used for obtaining the phase-space of the pendulum for rotation, libration and separatrix.

We can integrate the pendulum's equation of motion 5 using a 4th order Runge-Kutta and the initial conditions listed in table 1.

Figure 1 shows the phase space in the three domains: rotation, libration and separatrix.

## 2.d

Considering a disk-like of initial conditions enclosed within a circle centered at $(q, p) = (z_1, z_2) = (0, 1)$ with radius of $1/2$ as shown in Figure 2, we can compute the evolution of the disk from $\tau = 0$ to each of $\tau = 0.25\tau_0, 0.5\tau_0,$ and$\tau_0$.

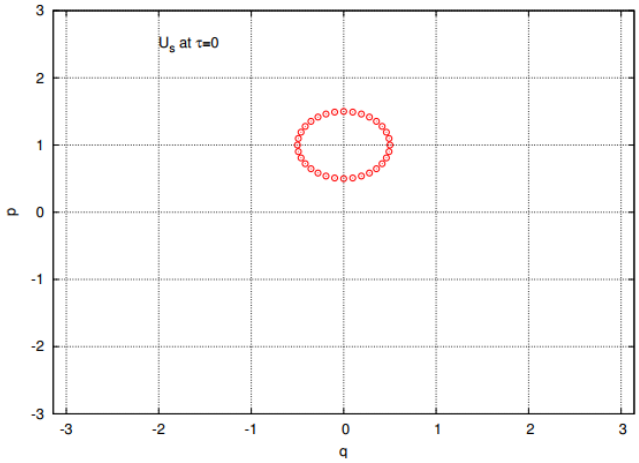


Figure 2: Initial conditions for $U_s$

Figure 3 shows the evolution of 32 orbits equally spaced at different times obtained from 4th or-

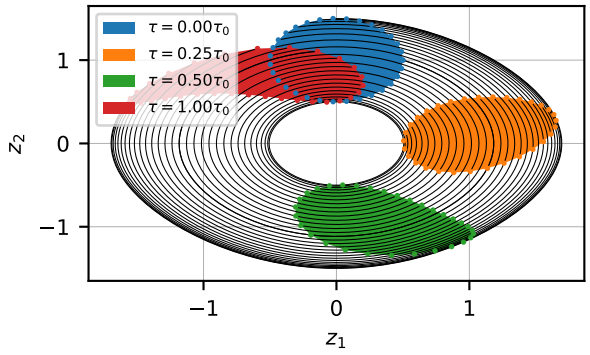


Figure 3: Evolution of the phase space of 32 orbits using a disk-like shape of initial conditions centered on $(z_1, z_2) = (0, 1)$ with a radius of $1/2$.

der Runge-Kutta integration. The color disks shows the final values of $(z_1, z_2)$ at the different stop times as shown. Note that the orbits are parallel through time and don't cross each other so we can argue that calculating the orbits at the edge of the disk puts a constraint on orbits at any point inside the disk. Additionally we can confirm that all of these orbits are in the libration regime since they go around in a circle and seem to reach the same initial point.

## 2.e

Figure 4 shows the evolution of 32 orbits using the same disk-like shape of initial conditions but now centered on $(z_1, z_2) = (0, 1.5)$. As you can note, we continue to have orbits in libration but since we are getting closer to the separatrix the orbits start to have a oblong shape.

## 2.f

Figure 5 shows the evolution of 32 orbits but now we centered the disk-like shape of initial conditions on $(z_1, z_2) = (0, 2)$. With this initial condi-

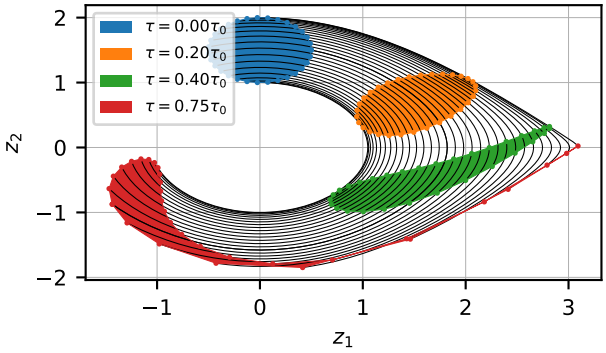


Figure 4: Evolution of the phase space of 32 orbits using a disk-like shape of initial conditions centered on $(z_1, z_2) = (0, 1.5)$ with a radius of $1/2$.

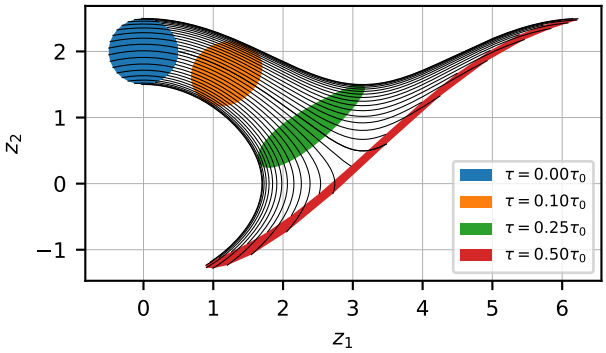


Figure 5: Evolution of the phase space of 32 orbits using a disk-like shape of initial conditions centered on $(z_1, z_2) = (0, 2)$ with a radius of $1/2$.

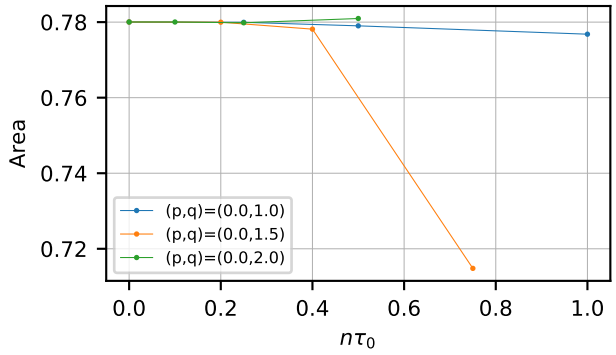


Figure 6: Plot showing the area calculated at the different stop times specified on each corresponding phase space plot using 32 points equally spaced.

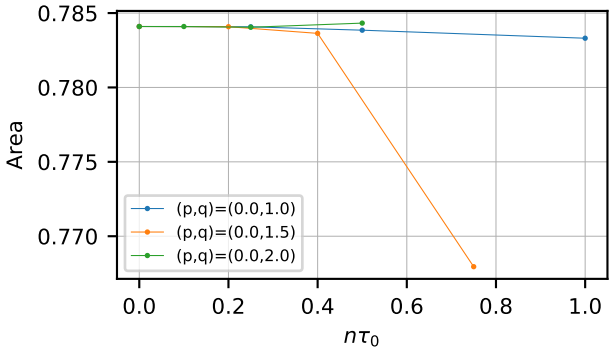


Figure 7: Plot showing the area calculated at the different stop times specified on each corresponding phase space plot using 64 points equally spaced.

tions we crossed the separatrix and have orbits in rotation.

## 2.g

We can represent the state of a system of N stars by a point in a 6N-dimensional phase space called Γ-space and the point described by the coordinates is a Γ-point. Liouville's theorem states that the flow of Γ-points through Γ-space is incompressible. We can calculate the area of our disk-like initial conditions and show that it remains constant through time.

To calculate the area I used the shoelace formula:

$$A = \frac{1}{2}\sum_{i=1}^{N}(x_i y_{i+1} - x_{i+1}y_i) \tag{6}$$

where, for this case, $x = z_1$ and $y = z_2$.

Since our disk-like of initial conditions all have a radius of 1/2 then we can calculate the analytic area to be $\pi r^2 = \pi(.5)^2 = 0.7854$.

Figure 6 shows the areas for each set of different initial conditions at the different $\tau$. We can note that they stay constant except for the last time step when the initial conditions are $(z_1, z_2) = (0, 1.5)$. This may be an effect of the disk being stretch out to the point that the calculated points don't sample the area good enough. This is confirmed with Figure 7 where instead of using 32 points, we used 64 points and the area difference is reduced from 0.06 to 0.01. Additionally, with more sample points the overall numeric area is closer in value to the estimated analytic area.

# References

[1] James Binney and Scott Tremaine. *Galactic Dynamics*. Princeton Series in Astrophysics, 2008.

# A Python code

Algorithm 1: Astro645HW05p2.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 11 12:51:02 2019

@author: Sandra Bustamante
"""

import numpy as np
import matplotlib.pyplot as plt
import NumericIntegrations as NI
import SetupPlots as SP
import pandas as pd
import time

#%% Definitions
def dvdt(t,z1,z2):
  #Pendulum equation of motion
  dvdt=-np.sin(z1)
  #print(dvdt)
  return dvdt

def energy(z1,z2):
  #Energy of pendulum
  E=(1/2)*z2**2+(1-np.cos(z1))
  return E

def calcUs(rU,q,p,bArray):
  start = time.time()
  thetaU=np.linspace(0,2*np.pi,64)
  q0=rU*np.cos(thetaU)+q
  p0=rU*np.sin(thetaU)+p
  lenq=len(q0)
  IVb=np.concatenate((q0.reshape(lenq,1),
                      p0.reshape(lenq,1)),
                     axis=1)
  qfinalArray=np.zeros((len(bArray),lenq))
  pfinalArray=np.zeros((len(bArray),lenq))
  qfinalArray[0]=q0
  pfinalArray[0]=p0

  for j in np.arange(1,len(bArray)):
    N=np.int(np.ceil((bArray[j]/h)))
    qArray=np.zeros((lenq,N+1))
    pArray=np.zeros((lenq,N+1))

    for i in range(lenq):
      t,q,p=NI.RK4(dvdt,a,bArray[j],h,IVb[i],dim=1)
      #saving values of all 32 orbits
      qArray[i]=q[:,0]
      pArray[i]=p[:,0]
#Array of the final values of each value of time bArray
    qfinalArray[j]=qArray[:,-1]
    pfinalArray[j]=pArray[:,-1]

  end = time.time()
  print('Time to run: %0.2f'%(end - start))
  return IVb,qArray,pArray,qfinalArray,pfinalArray

def calcArea(qArray,pArray):
  n=len(qArray)
  A=np.zeros(n)
  for i in np.arange(n):
    q=qArray[i]
    p=pArray[i]
    A[i]=.5*(q*np.roll(p,-1)-p*np.roll(q,-1)).sum()
  print(A)
  return A

#%%

Ecrit=2
indexNames=['Rotation','Libration','Separatrix']
z1=np.array([-3*np.pi/2,-np.pi/2,-3*np.pi/2])
z2=np.array([np.sqrt(2)+.5,np.sqrt(2)-.5,np.sqrt(2)])

IV=np.concatenate((z1.reshape(3,1),z2.reshape(3,1)),
                  axis=1)

a=0
b=100
h=0.01
N=round(b/h)

#%% 2.C
start = time.time()
t,thetaRot,thetaDotRot=NI.RK4(dvdt,a,b,h,IV[0],dim=1)
t,thetaLib,thetaDotLib=NI.RK4(dvdt,a,b,h,IV[1],dim=1)
t,thetaSep,thetaDotSep=NI.RK4(dvdt,a,b,h,IV[2],dim=1)
end = time.time()
print('Time to run: %0.2f'%(end - start))

#%% Plot 2.C
width,height=SP.setupPlot(singleColumn=False)
grid = plt.GridSpec(1,1)
fig1 = plt.figure(figsize=(width,height))

ax1 = fig1.add_subplot(grid[0,0])
ax1.plot(thetaRot,thetaDotRot,label=indexNames[0])
ax1.plot(thetaLib,thetaDotLib,label=indexNames[1])
ax1.plot(thetaSep,thetaDotSep,label=indexNames[2])
ax1.set_xlabel(r'$z_1$')
ax1.set_xlim(-np.pi,np.pi)
ax1.set_ylabel(r'$z_2$')
ax1.grid()
ax1.legend()

fig1.tight_layout()
#fig1.savefig('PendulumPhaseSpace.pdf')

#%% 2.D q,p=0,1
l=1
omega=1#constants.g/l
tau0=2*np.pi/omega
bArray2d=np.array([0,.25,.5,1])
tau2d=bArray2d*tau0
qp2d=(0,1)
P2d=calcUs(rU=1/2,q=qp2d[0],p=qp2d[1],bArray=tau2d)
IV2d,qA2d,pA2d,qfinalA2d,pfinalA2d=P2d

A2d=calcArea(qfinalA2d,pfinalA2d)

#%% 2.E q,p=0,1.5
bArray2e=np.array([0,.2,.4,.75])
tau2e=bArray2e*tau0
qp2e=(0,1.5)
P2e=calcUs(rU=1/2,q=qp2e[0],p=qp2e[1],bArray=tau2e)
IV2e,qA2e,pA2e,qfinalA2e,pfinalA2e=P2e
A2e=calcArea(qfinalA2e,pfinalA2e)

#%% 2.F q,p=0,2
bArray2f=np.array([0,.1,.25,.5])
tau2f=bArray2f*tau0
qp2f=(0,2)
P2f=calcUs(rU=1/2,q=qp2f[0],p=qp2f[1],bArray=tau2f)
IV2f,qA2f,pA2f,qfinalA2f,pfinalA2f=P2f
A2f=calcArea(qfinalA2f,pfinalA2f)

#%% Plot 2d
#width,height=SP.setupPlot(singleColumn=False)
#grid = plt.GridSpec(1,1)
fig2 = plt.figure(figsize=(width,height))

ax2 = fig2.add_subplot(grid[0,0])
ax2.plot(qA2d.T,pA2d.T,'k-')
for i in range(4):
  ax2.plot(qfinalA2d[i,:],pfinalA2d[i,:],'-o')
  ax2.fill(qfinalA2d[i,:],pfinalA2d[i,:],'-o',
           label=r'$\tau=%1.2f\tau_0$'%bArray2d[i])
ax2.set_xlabel(r'$z_1$')
#ax2.set_xlim(-np.pi,2.5)
ax2.set_ylabel(r'$z_2$')
ax2.grid()
ax2.legend()

fig2.tight_layout()
#fig2.savefig('PendulumPhaseSpaceUs2d.pdf')
fig2.savefig('PendulumPhaseSpaceUs2d64p.pdf')

#%% Plot 2e
#width,height=SP.setupPlot(singleColumn=False)
#grid = plt.GridSpec(1,1)
fig3 = plt.figure(figsize=(width,height))

ax3 = fig3.add_subplot(grid[0,0])
ax3.plot(qA2e.T,pA2e.T,'k-')
for i in range(4):
  ax3.plot(qfinalA2e[i,:],pfinalA2e[i,:],'-o')
  ax3.fill(qfinalA2e[i,:],pfinalA2e[i,:],'-o',
           label=r'$\tau=%1.2f\tau_0$'%bArray2e[i])
```

```
ax3.set_xlabel(r'$z_1$')
#ax3.set_xlim(-np.pi,np.pi)
ax3.set_ylabel(r'$z_2$')
ax3.grid()
ax3.legend()

fig3.tight_layout()
#fig3.savefig('PendulumPhaseSpaceUs2e.pdf')
fig3.savefig('PendulumPhaseSpaceUs2e64p.pdf')

#%% Plot 2f
#width,height=SP.setupPlot(singleColumn=False)
#grid = plt.GridSpec(1,1)
fig4 = plt.figure(figsize=(width,height))

ax4 = fig4.add_subplot(grid[0,0])
ax4.plot(qA2f.T,pA2f.T,'k-')
for i in range(4):
  ax4.plot(qfinalA2f[i,:],pfinalA2f[i,:],'-o')
  ax4.fill(qfinalA2f[i,:],pfinalA2f[i,:],'-o',
           label=r'$\tau=%1.2f\tau_0$'%bArray2f[i])
ax4.set_xlabel(r'$z_1$')
#ax4.set_xlim(-np.pi,np.pi)
ax4.set_ylabel(r'$z_2$')
ax4.grid()
ax4.legend()

fig4.tight_layout()
#fig4.savefig('PendulumPhaseSpaceUs2f.pdf')
fig4.savefig('PendulumPhaseSpaceUs2f64p.pdf')

#%% Plot 2g
#width,height=SP.setupPlot(singleColumn=False)
#grid = plt.GridSpec(1,1)
fig5 = plt.figure(figsize=(width,height))

ax5 = fig5.add_subplot(grid[0,0])
ax5.plot(bArray2d, A2d,'-o',
         label=r'(p,q)=(%1.1f,%1.1f)'%(qp2d[0],qp2d[1]))
ax5.plot(bArray2e,A2e,'-o',
         label=r'(p,q)=(%1.1f,%1.1f)'%(qp2e[0],qp2e[1]))
ax5.plot(bArray2f,A2f,'-o',
         label=r'(p,q)=(%1.1f,%1.1f)'%(qp2f[0],qp2f[1]))
ax5.set_ylabel('Area')
ax5.set_xlabel(r'$n\tau_0$')
ax5.grid()
ax5.legend()

fig5.tight_layout()
#fig5.savefig('PendulumPhaseSpaceAreas.pdf')
fig5.savefig('PendulumPhaseSpaceAreas64p.pdf')

##%% Save Data to csv file
#
#colNames=np.array(['$z_1$','$z_2$','Energy'])
#row1=np.array([z1[0],z2[0],energy(z1[0],z2[0])])
#row2=np.array([z1[1],z2[1],energy(z1[1],z2[1])])
#row3=np.array([z1[2],z2[2],energy(z1[2],z2[2])])
#
#rows=[row1,row2,row3]
#
#df = pd.DataFrame(rows,columns=colNames,
#                  index=indexNames)
#
#with open('PendulumIV.tex','w') as tf:
#    tf.write(df.to_latex(float_format='%2.2f',
#                         index=True,
#                         escape=False))
#
##%% Save Data to csv file
#
#colNames2=(['$%1.2f\tau_0$'%bArray2d[0],
#            '$%1.2f\tau_0$'%bArray2d[1],
#            '$%1.2f\tau_0$'%bArray2d[2],
#            '$%1.2f\tau_0$'%bArray2d[3]])
#row1b=qfinalA2d.T
#row2b=A2d
#indexNames2=([np.arange(32),'Areas'])
##row3=np.array([z1[2],z2[2],energy(z1[2],z2[2])])
#
#rows=row1b
#
#df = pd.DataFrame(rows,columns=colNames2)
#
#with open('Pendulum2d.tex','w') as tf:
#    tf.write(df.to_latex(float_format='%2.2f',
#                         index=True,
#                         escape=False))
```

## Algorithm 2: SetupPlots.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 28 17:24:55 2019

@author: sbustamanteg
"""
import numpy as np
import matplotlib.pyplot as plt

def setupPlot(singleColumn):

  if singleColumn:
    width=6.9
  else:
    width=3.39

  fontsize=8
  linewidth=0.4
  markersize=1

  height=width*(np.sqrt(5.)-1.)/2.
  params = {'axes.labelsize': fontsize,
            'axes.titlesize': fontsize,
            'font.size': fontsize,
            'legend.fontsize': fontsize-2,
            'xtick.labelsize': fontsize,
            'ytick.labelsize': fontsize,
            'lines.linewidth': linewidth,
            'grid.linewidth' : linewidth*.7,
            'axes.axisbelow' : True,
            'pgf.rcfonts' : False,
            'lines.markersize' : markersize,
           }
  plt.rcParams.update(params)
  return width,height

#def 3x1Plot():
```

Algorithm 3: NumericIntegrations.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 28 17:08:16 2019

@author: sbustamanteg
"""

import numpy as np

def leapfrog2D(dvdt,a,b,h,IV,dim=2):
  #h = (b-a)/float(N)
  t=np.arange(a,b+h,h)
  x=np.zeros((len(t),dim))
  v=np.zeros((len(t),dim))
  x[0], v[0] = IV
  for n in np.arange(1,len(t)):
    k1=x[n-1]+0.5*h*v[n-1]
    v[n]=v[n-1]+h*dvdt(t,k1,v)
    x[n]=k1+0.5*h*v[n]
  return t, x, v

def RK4(dvdt, a, b, h, IV,dim=2):
    t = np.arange(a,b+h,h)  # create time
    #N=int((b-a)/h)
    x = np.zeros((len(t),dim))        # initialize x
    v = np.zeros((len(t),dim))        # initialize x
    x[0], v[0] = IV        # set initial values
    # apply Fourth Order Runge-Kutta Method
    for i in np.arange(1,len(t)):
        k1= h*dvdt(t[i-1], x[i-1], v[i-1])
        j1= h*v[i-1]
        k2= h*dvdt(t[i-1]+h/2.0, x[i-1]+j1/2.0,
                   v[i-1]+k1/2.0)
        j2= h*(v[i-1]+k1/2.0)
        k3= h*dvdt(t[i-1]+h/2.0, x[i-1]+j2/2.0,
                   v[i-1]+k2/2.0)
        j3= h*(v[i-1]+k2/2.0)
        k4= h*dvdt( t[i], x[i-1] + j3, v[i-1] + k3)
        j4= h*(v[i-1] + k3)
        v[i] =v[i-1] + (k1 + 2.0*k2 + 2.0*k3 + k4)/6.0
        x[i] =x[i-1] + (j1 + 2.0*j2 + 2.0*j3 + j4)/6.0
    return t, x, v
```