# Analysis of Van der Pol Oscillators

Zinnun Malikov, Alexis Solorzano

Math 2552 Differential Equations

Abstract: This research paper presents an analysis of Van der Pol oscillators, a class of nonlinear oscillatory systems. They are used to represent non-conservative, oscillating systems with non-linear damping. We will investigate the behavior of Van der Pol oscillators across different damping strengths, focusing on cases where the parameter $\mu$ is equal to zero, small ($0 < \mu \ll 1$), and large ($1 \ll \mu$). Numerical solutions to these equations will provide insights into the dynamic behavior of Van der Pol oscillators and offer comparisons with other predictions.

Keywords: Van der Pol oscillator, RLC circuit, non-linear damping

## 1. Introduction

The Van der Pol (VDP) oscillator equation describes the behavior of a nonlinear oscillator, which exhibits self-sustained oscillations. Dutch physicist Balthasar van der Pol, introduced the equation in the 1920s while studying electronic circuits to describe oscillations in a vacuum tube electrical circuit. The oscillator evolves according a second-order ordinary differential equation:

$$\boldsymbol{x'' - \mu(1 - x^2)x' + x = 0, \qquad \mu > 0} \tag{1}$$

This equation stems from Kirchoff's Voltage Law, used to describe the behavior of current in an RLC circuit shown below. This circuit consists of an inductor $L$, a resistor $R$, and a capacitor $C$, and a voltage source $E$ (constant). The behavior is described in equation (2). Note that $Q$ is the charge on the capacitor at a given point (Kanamaru 2007).

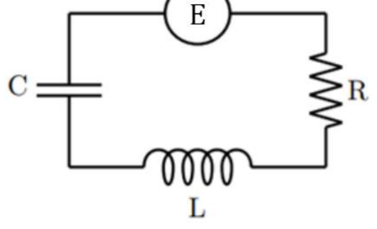


Figure 1: RLC Circuit

$$\boldsymbol{LI' + RI + \frac{Q}{C} = E} \tag{2}$$

However, a VDP oscillator involves an active element (semiconductor) instead of a passive resistor. The semiconductor pumps energy into the system when the current is low, and dampens energy when the current is too high. This action is modelled by $I^2 - \alpha$, yielding:

$$LI' + (I^2 - \alpha)I + \frac{Q}{C} = E$$

Taking the derivative of both sides results in the VDP equation (the constants can be rescaled to be in standard form):

$$LI'' + 3(I^2 - \frac{\alpha}{3})I' + \frac{I}{C} = 0$$

Also note that $Q' = I$ since current is the amount of charge flowing through a point over a period of time (Brown 1963).

## 2. Case $\mu = 0$: Harmonic Oscillator

When $\mu = 0$, the system is analogous to an undamped oscillator and the equation becomes the harmonic oscillator:

$$x'' + x = 0$$

For this system, orbits in the phase-plane are circular around the origin. Focus will be placed on the cases when $\mu > 0$, which will result in VDP behavior.

## 3. Case $0 < \mu \ll 1$: Small non-linearity

Define equation (1) as a system of first-order equations:

$$\begin{cases} \boldsymbol{x' = v} \\ \boldsymbol{v' = \mu(1 - x^2)v - x} \end{cases} \tag{3}$$

$$x'' - \underbrace{\mu(1 - x^2)x'}_{\text{damping}} + \underbrace{x}_{\text{restoring}} = 0 \tag{4}$$

For small values of $\mu$, the VDP oscillator system will be weakly non-linear. Analysis of the behavior of the VDP oscillator reveals convergence to a limit cycle. Since the non-linear case stems from the harmonic oscillator, all orbits are circular, with period $2\pi$.

$$x(t) = A\cos(t)$$

Observing the change in total energy over one cycle can help reveal the behavior of this VDP system. On a limit cycle, the amount of energy pumped in by the non-linear damping term from (4) equals zero: $\Delta E = 0$. However, other trajectories not on the limit cycle will result in a non-zero change in energy: $\Delta E > 0$ or $\Delta E < 0$. Energy can be represented as:

$$E = \tfrac{1}{2}x'^2 + \tfrac{1}{2}x^2$$

$$E' = x'x'' + xx' = x'(x'' + x) = \mu x'^2(1 - x^2)$$

Suppose trajectories have the form: $x = A\cos(t) + O(\mu)$, where $O(\mu)$ is an error term of order $\mu$. It is possible to now calculate the change in energy over one cycle:

$$\Delta E = \int_0^T \frac{dE}{dt} dt = -\mu \int_0^{2\pi + O(\mu)} A^2 \sin^2(t)(A^2\cos^2(t) - 1) + O(\mu^2) dt$$

$$= -2\mu\pi[A^4\{\sin^2(t)\cos^2(t)\} - A^2\{\sin^2(t)\}]$$

where {} denotes an average over the cycle. This method of averaging stems from the assumption that since the variables are slowly varying in time, they are acting like constants *on average*. This calculation yields:

$$\Delta E = -2\mu\pi A^2(\frac{A^2}{8} - \frac{1}{2})$$

Since the change in energy is zero, $A = 2$. Thus, the limit cycle is $x(t) = 2\cos(t) + O(\mu)$. This calculation confirms the converging behavior of these types of VDP systems; convergence to an orbit of radius approximately 2. Moreover, trajectories starting further away will still exhibit convergence behavior. Since the non-linearity is small, it can be argued that the phase plane will be a slight distortion of the phase plane of the harmonic oscillator. Because of the circular behavior, consider the use of polar coordinates $(r, \theta)$. The energy analysis above yields the conclusion:

$$\lim_{t\to\infty} r(t) = 2$$

## 4. Case $1 \ll \mu$: Strong non-linearity

In the case for large values of $\mu$, the VDP system can be analyzed using Lienard's Theorem. Begin by rewriting the VDP equation (1) as:

$$x'' - \mu(1-x^2)x' + x = \frac{d}{dt}\left(x' + \mu\underbrace{\left(\frac{x^3}{3} - x\right)}_{F(x)}\right) + x$$

Define the new variable $w = x' + \mu F(x)$. Note that $w' = -x$. The VDP is now equal to:

$$\begin{cases} \boldsymbol{x' = w - \mu F(x)} \\ \boldsymbol{w' = -x} \end{cases} \tag{5}$$

Looking at the nullclines, $x' = 0$ when $w = \mu F(x)$, and $w' = 0$ when $x = 0$. Critical points (t) occur at $x = \pm 1$, $w = \mp\frac{2}{3}\mu$ and $x = \pm 2$, $w = \pm\frac{2}{3}\mu$. Note that $\mu$ is very large, and $w$ will be much larger than $y$. Thus, another change of variables is introduced: $y = \frac{w}{\mu}$, yielding (6):

$$\begin{cases} \boldsymbol{x' = \mu(y - F(x))} \\ \boldsymbol{y' = -\frac{x}{\mu}} \end{cases} \tag{6}$$

The critical points now occur at $x = \pm 1$, $y = \mp\frac{2}{3}$ and $x = \pm 2$, $y = \pm\frac{2}{3}$. However, off the $x' = 0$ nullcline, $x'$ is very large, while $y'$ is close to zero everywhere. If the initial condition is off the nullcline, the point will quickly move horizontally to the nullcline and then slowly change vertically, eventually converging to the stable limit cycle.

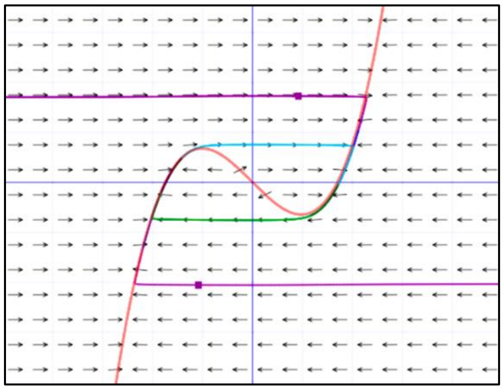


*Figure 2 Trajectories of the system with μ = 10*

In Figure 2, observe how initial points off the nullcline move horizontally toward the nullcline and then gradually approach the stable limit cycle.

## 5. Numerical Analysis

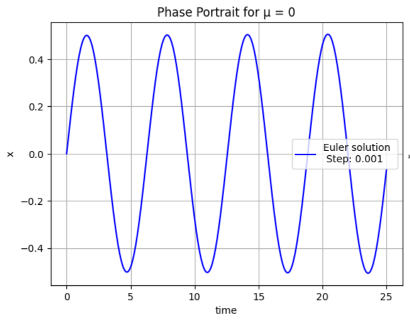


*Figure 3 Harmonic t vs. x*

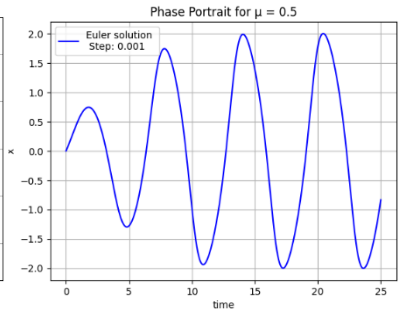


*Figure 4 Weak t vs. x*

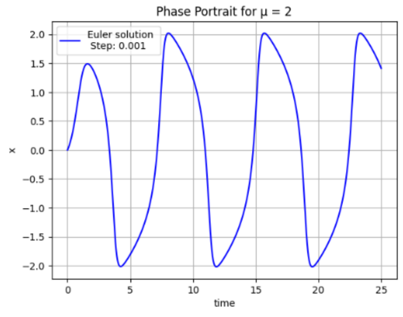


*Figure 5 Strong t vs. x*

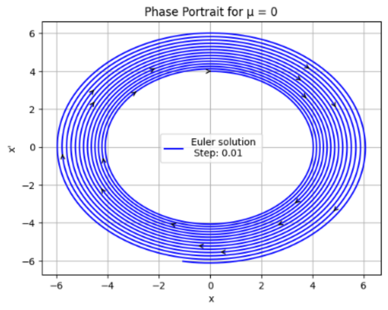


*Figure 6 Harmonic x vs. x′*

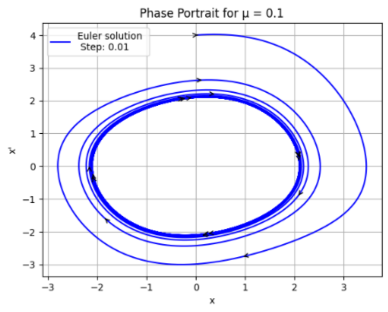


*Figure 7 Weak x vs. x′*

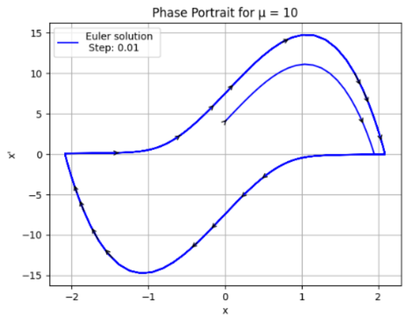


*Figure 8 Strong x vs. x′*

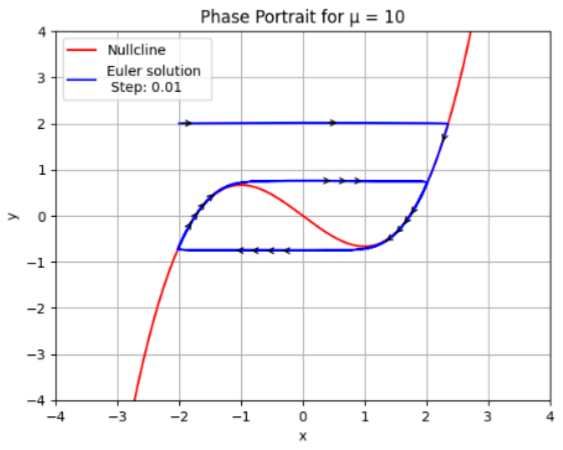


*Figure 9 Strong x vs. y*

Note the behavior of the VDP oscillator under each case. As previously mentioned, the undamped case is simply the harmonic oscillator and is stable, as seen in *Figures 3 & 6*. In the case of weak non-linearity, as seen in *Figures 4* & *7*, the system is asymptotically stable. Moreover, *Figure 7* reveals a convergence to a spiral of radius two, which supports the theoretical prediction from Section 3. Finally, the strong non-linear case, as seen in *Figures 5, 8 & 9,* exhibits a pattern that deviates from the harmonic and weak cases. These oscillations are sharper. Moreover, *Figure 9* shows that points off the nullcline will quickly move horizontally to the nullcline and then slowly change vertically, eventually converging to the stable limit cycle. Step sizes are indicated in the graphs above (0.001 with time as a variable, 0.01 otherwise).

## 6. Reality Check

In electronic circuits, particularly those containing vacuum tubes or transistors, the behavior of the circuit can exhibit oscillatory dynamics similar to van der Pol oscillators. One real-world example is the Colpitts oscillator, commonly used in radio frequency (RF) circuits (Feng 2014). The Colpitts oscillator utilizes active components such as transistors or vacuum tubes to generate oscillations at a specific frequency. The damping factor in such circuits, determined by component values and operating conditions, directly influences the stability and frequency of oscillation (Harwood 2011). When the damping factor is small ($0 < \mu \ll 1$), the circuit can exhibit sustained oscillations with a phase portrait resembling a spiral, as described in the van der Pol oscillator analysis. This aligns with the observation that for small damping factors, the phase portrait approaches a spiral of radius 2. Moreover, the numerical solution generated by the code exhibits this behavior; it approaches a spiral of radius two. Moreover, the sharper phase portrait of the case when $1 \ll \mu$ corresponds to behavior observed in real-world conditions (Harwood 2011).

## 7. Conclusion

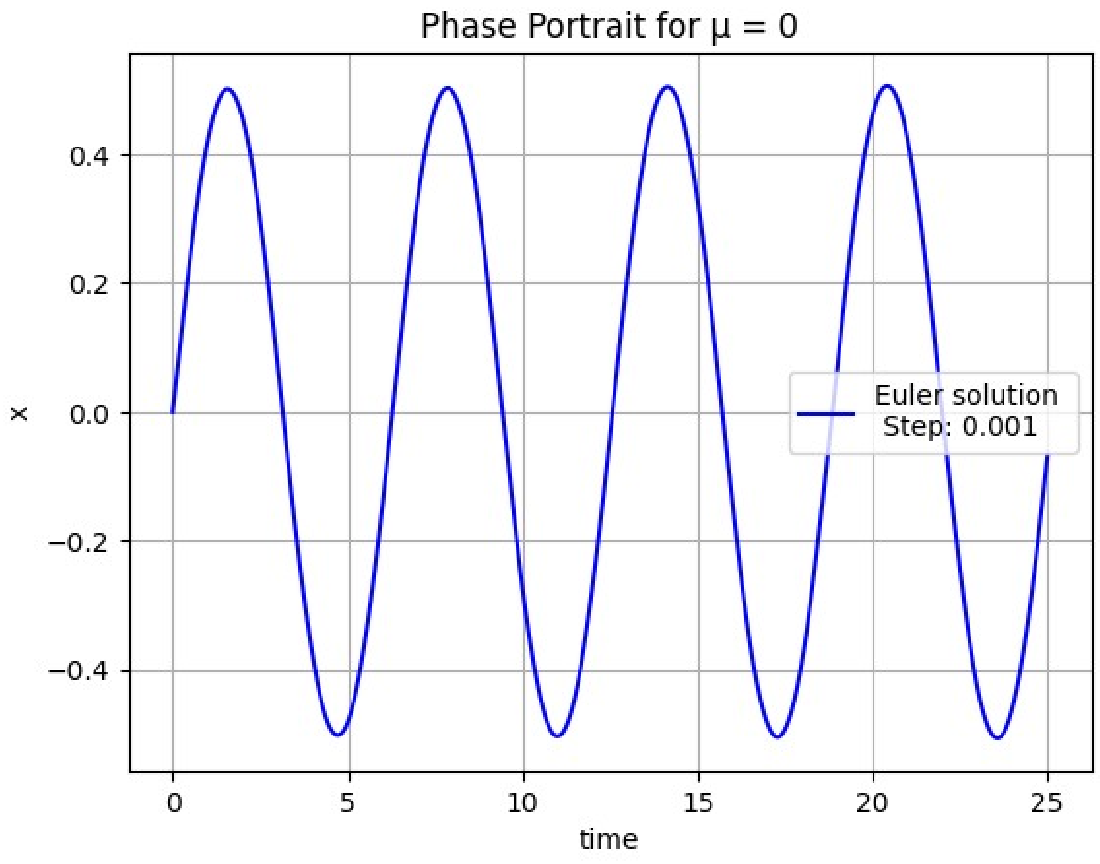
This research paper analyzed Van der Pol oscillators, a class of nonlinear oscillatory systems. used to represent non-conservative, oscillating systems with non-linear damping. We investigated the behavior of Van der Pol oscillators across different damping strengths, focusing on cases where the parameter $\mu$ is equal to zero, small ($0 < \mu \ll 1$), and large ($1 \ll \mu$). In the case $\mu = 0$, the system was described as a harmonic oscillator. In the case $0 < \mu \ll 1$, the VDP system was described as converging to an orbit of radius 2. Finally, in the case $1 \ll \mu$, the VDP system was manipulated with a variable transformation. The result showed convergence toward a stable limit cycle. Furthermore, numerical solutions to the VDP equations provided insights into the dynamic behavior of Van der Pol oscillators.

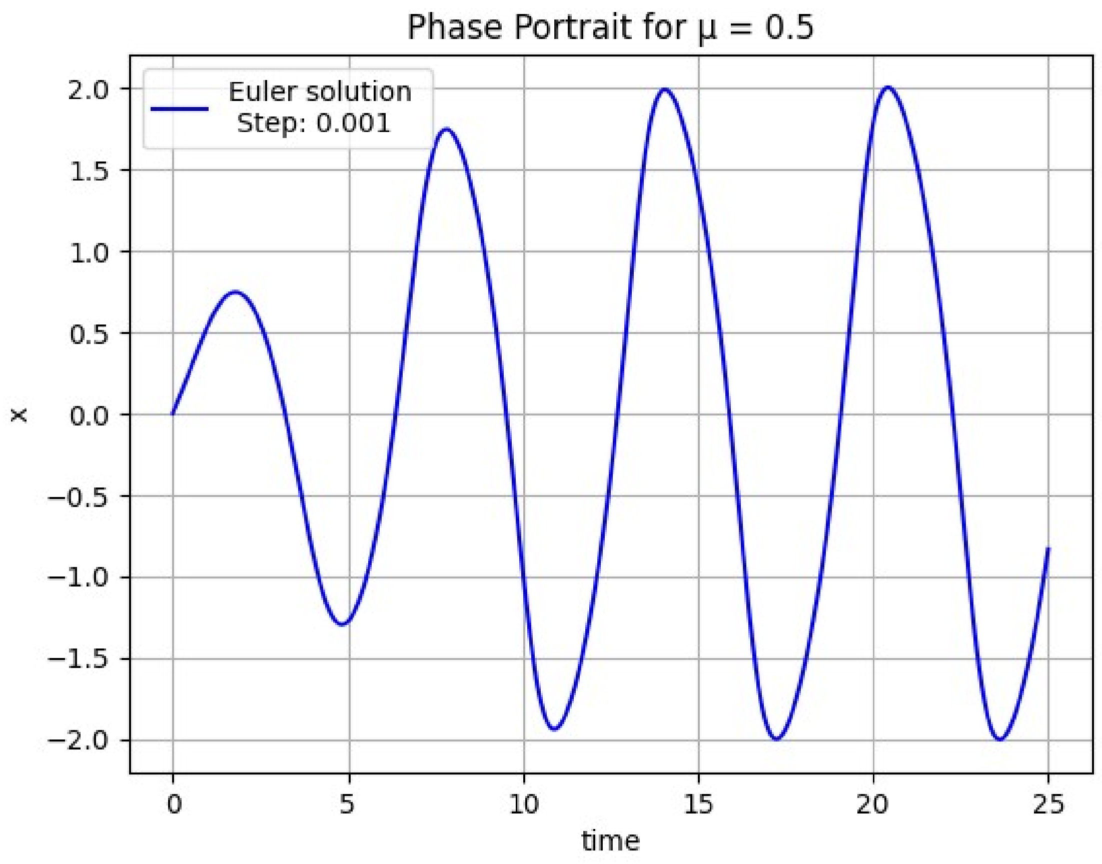
## Appendix

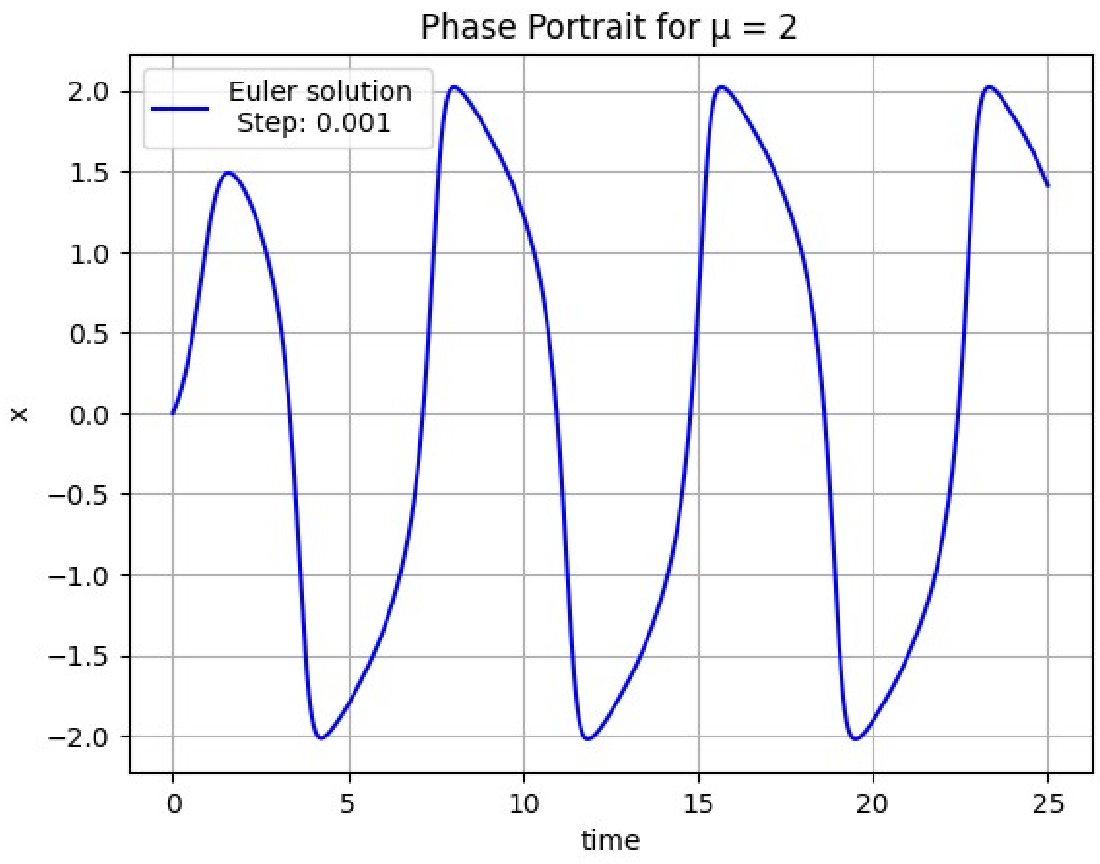
The code for creating the numerical analysis can be found in the GitHub folder below. It is also displayed in the next pages:

https://github.com/ZinnunMalikov/ZM-VDP-Oscillator

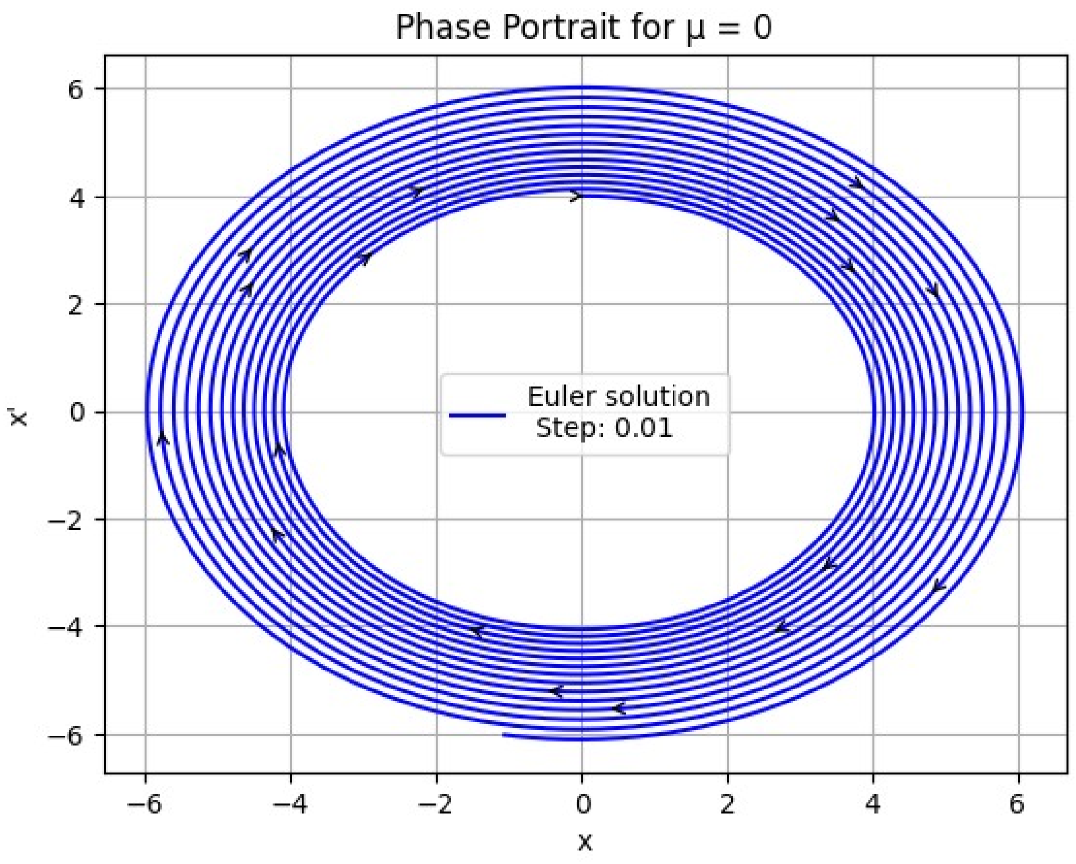
```
# import libraries for numerical functions and plotting
import numpy as np
import matplotlib.pyplot as plt


### VDP  time vs position
# Mu Constant

def run(u):
  f = lambda t,z1,z2: z2
  g = lambda t,z1,z2: u*(1-z1**2)*z2-z1

  #Step Size
  dt = 0.001
  time = np.arange(0, 25.001, dt)

  z1 = np.zeros_like(time)
  z2 = np.zeros_like(time)
  # initial conditions
  z1[0] = 0
  z2[0] = 0.5

  # Forward Euler iterations
  for idx, t in enumerate(time[:-1]):
      z1[idx+1] = z1[idx] + dt*f(t, z1[idx], z2[idx])
      z2[idx+1] = z2[idx] + dt*g(t, z1[idx], z2[idx])

  pz1 = []
  pz2 = []

  # Cosmetic Points

  for i in range(len(z1)):
    if i % 20 == 0:
      pz1.append(z1[i])

  for i in range(len(z2)):
    if i % 20 == 0:
      pz2.append(z2[i])

  # Plots
  plt.plot(time, z1, '-', label='Euler solution \n Step: ' + str(dt),
color = 'blue')
  # plt.plot(pz1, pz2, 'o', color = 'blue')
  plt.xlabel('time')
  plt.ylabel('x')
  plt.legend()
  plt.grid(True)


  plt.title('Phase Portrait for μ = ' + str(u))
  plt.show()

for u in [0, 0.5, 2]:
  run(u)
```

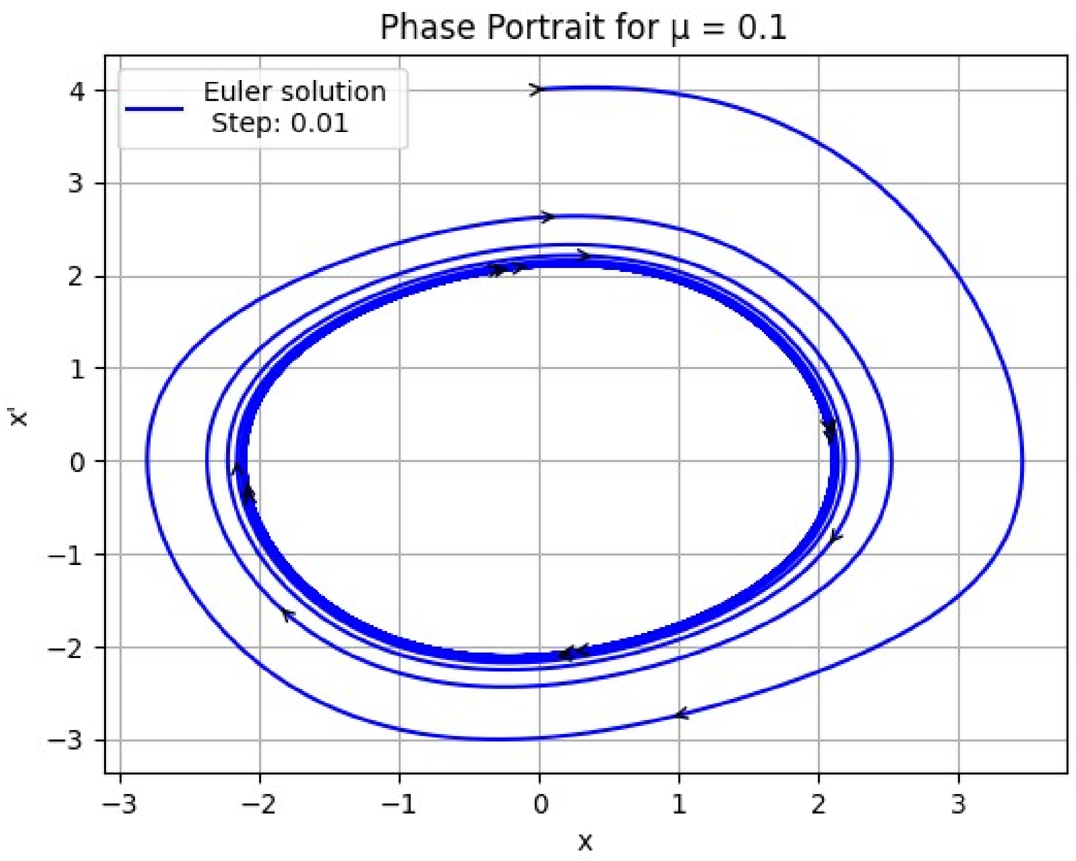


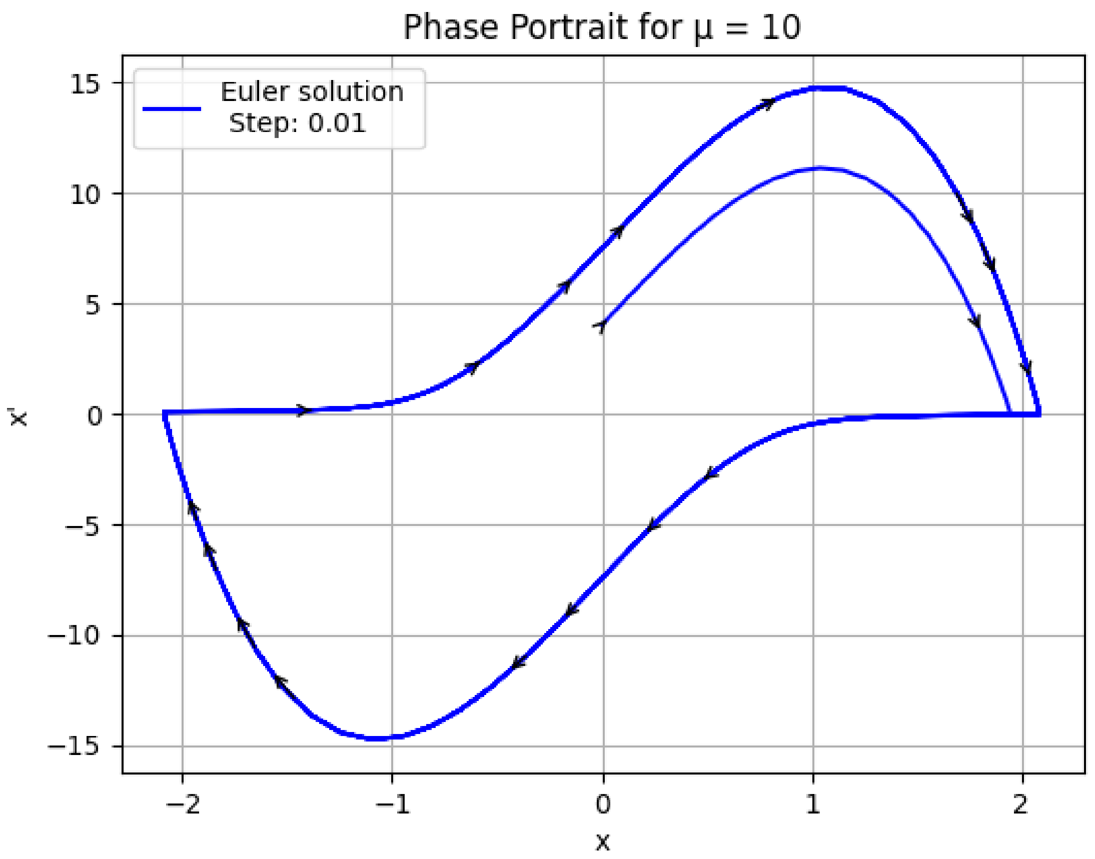

Phase Portrait for μ = 0.5
Euler solution
Step: 0.001
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
x
0
5
10
15
20
25
time



```
### VDP  position vs velocity
# Mu Constant

def run2(u):
  f = lambda t,z1,z2: z2
  g = lambda t,z1,z2: u*(1-z1**2)*z2-z1

  #Step Size
  dt = 0.01
  time = np.arange(0, 85.001, dt)

  z1 = np.zeros_like(time)
  z2 = np.zeros_like(time)
  # initial conditions
  z1[0] = 0
  z2[0] = 4

  # Forward Euler iterations
  for idx, t in enumerate(time[:-1]):
      z1[idx+1] = z1[idx] + dt*f(t, z1[idx], z2[idx])
      z2[idx+1] = z2[idx] + dt*g(t, z1[idx], z2[idx])

  pz1 = []
  pz2 = []

  # Cosmetic Points

  for i in range(len(z1)):
    if i % 20 == 0:
      pz1.append(z1[i])

  for i in range(len(z2)):
    if i % 20 == 0:
      pz2.append(z2[i])

  # Plots
  plt.plot(z1, z2, '-', label='Euler solution \n Step: ' + str(dt),
color = 'blue')
  # plt.plot(pz1, pz2, 'o', color = 'blue')
  plt.xlabel('x')
  plt.ylabel("x'")
  plt.legend()
  plt.grid(True)

  # compute the distances, ds, between points
  dx, dy = z1[+1:]-z1[:-1],  z2[+1:]-z2[:-1]
  ds = np.array((0, *np.sqrt(dx*dx+dy*dy)))

  # compute the total distance from the 1st point, measured on the
curve
  s = np.cumsum(ds)

  # interpolate
  xinter = np.interp(np.linspace(0,s[-1], 20), s, z1)
  yinter = np.interp(np.linspace(0,s[-1], 20), s, z2)

  # plot the interpolated points
  for i in range(len(xinter) - 2):
      xc, yc = xinter[i], yinter[i]
      plt.annotate('', xy=(xc + dt*f(t, xc, yc), yc + dt*g(t, xc,
yc)), xytext=(xc, yc),
                  arrowprops=dict(facecolor='blue',  arrowstyle = '-
>'))

  plt.title('Phase Portrait for μ = ' + str(u))

  plt.show()

for u in [0, 0.1, 10]:
  run2(u)
```

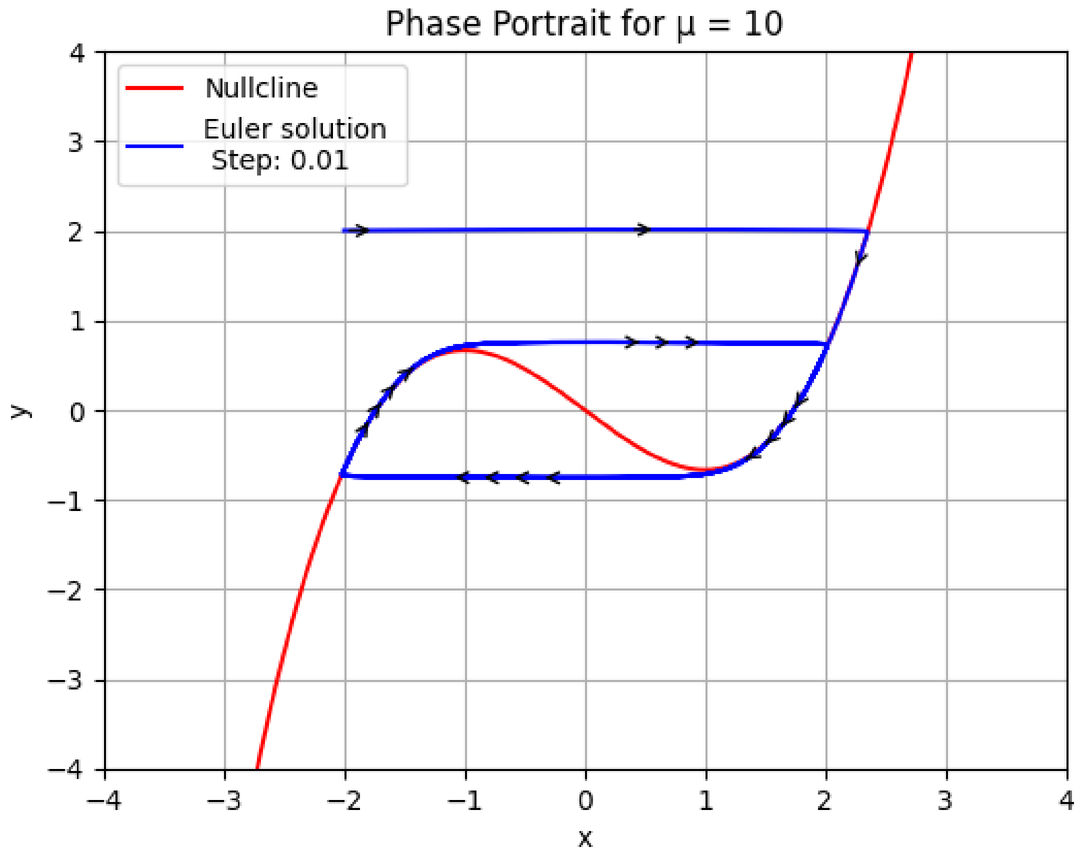

Phase Portrait for μ = 0
Euler solution
Step: 0.01
x
x'
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


Phase Portrait for μ = 0.1
Euler solution
Step: 0.01
x
x'
4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3



```
### Strong VDP position vs transform
# Mu Constant
u = 10
f = lambda t,z1,z2: u*(z2-((z1**3)/3-z1))
g = lambda t,z1,z2: -z1/u
xl, xu = -4, 4
yl, yu = -4, 4


# Step Size
dt = 0.01
time = np.arange(0, 85.001, dt)
nullc = np.arange(xl, xu, dt)

z1 = np.zeros_like(time)
z2 = np.zeros_like(time)
z3 = []

# initial conditions
z1[0] = -2
z2[0] = 2

# Forward Euler iterations
```

```python
for idx, t in enumerate(time[:-1]):
    z1[idx+1] = z1[idx] + dt*f(t, z1[idx], z2[idx])
    z2[idx+1] = z2[idx] + dt*g(t, z1[idx], z2[idx])

for x in (nullc):
  z3.append((1/3)*x**3 - x)

pz1 = []
pz2 = []

# Cosmetic Points

for i in range(len(z1)):
  if i % 20 == 0:
    pz1.append(z1[i])

for i in range(len(z2)):
  if i % 20 == 0:
    pz2.append(z2[i])

# Plots
plt.plot(nullc, z3, '-', label='Nullcline', color = 'red')
line = plt.plot(z1, z2, '-', label='Euler solution \n Step: ' +
str(dt), color = 'blue')[0]
plt.xlabel('x')
plt.ylabel("y")
plt.xlim(-4, 4)
plt.ylim(-4, 4)

# #Arrows
# for i in range(0, len(z1) - 10, 550):  # Add arrow every 10 data
points
#     plt.annotate('', xy=(z1[i+5], z2[i+5]), xytext=(z1[i], z2[i]),
#                  arrowprops=dict(facecolor='blue',  width = 1,
shrink = 0.5))

# compute the distances, ds, between points
dx, dy = z1[+1:]-z1[:-1],  z2[+1:]-z2[:-1]
ds = np.array((0, *np.sqrt(dx*dx+dy*dy)))

# compute the total distance from the 1st point, measured on the curve
s = np.cumsum(ds)

# interpolate
xinter = np.interp(np.linspace(0,s[-1], 20), s, z1)
yinter = np.interp(np.linspace(0,s[-1], 20), s, z2)

# plot the interpolated points
for i in range(len(xinter) - 2):
    xc, yc = xinter[i], yinter[i]
    plt.annotate('', xy=(xc + dt*f(t, xc, yc), yc + dt*g(t, xc, yc)),
xytext=(xc, yc),
                    arrowprops=dict(facecolor='blue',  arrowstyle = '-
>'))

plt.legend()
plt.grid(True)
plt.title('Phase Portrait for μ = ' + str(u))
plt.show()
```



## References

Brown, D. P. (1963). Derivative-explicit differential equations for RLC graphs. Journal of the Franklin Institute, 275(6), 503-514.

Feng, Z. (2014). Duffing-van der Pol-type oscillator systems. Discrete Contin. Dyn. Syst. Ser. S, 7(6), 1231-1257.

Harwood, K. (2011). Modeling a RLC Circuit's Current with Differential Equations.

Kanamaru, T. (2007). Van der Pol oscillator. Scholarpedia, 2(1), 2202.