Classical Mechanics Assignment #2

Johannes Byle

September 10, 2021

1. (a) Solving the first derivative to find the location of extrema:

$$V(x) = \frac{x^2}{2} - \frac{x_0\sqrt{1+gx^2}}{\sqrt{1+gx_0^2}}$$
$$\frac{\delta V(x)}{\delta x} = x\left[1 - \frac{gx_0}{\sqrt{gx^2+1}\sqrt{gx_0^2+1}}\right]$$

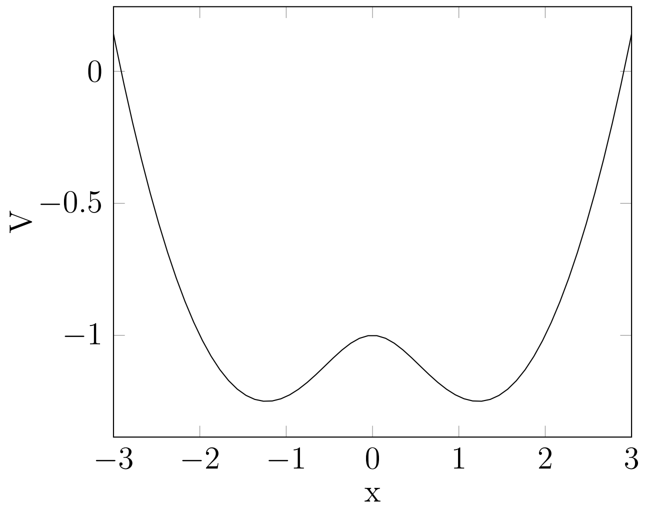
Solving for where $\frac{\delta V(x)}{\delta x} = 0$:

$$x = 0,\ \pm\frac{\sqrt{g^2x_0^2 - gx_0^2 - 1}}{\sqrt{g^2x_0^2 + g}}$$

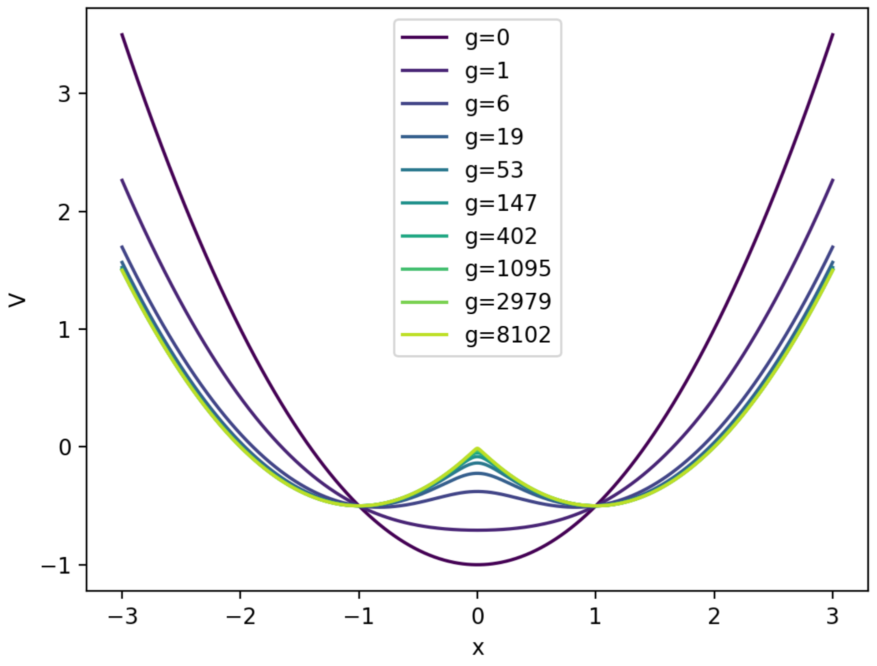
Taking the second derivative to evaluate each extremum:

$$\frac{\delta V(x)}{\delta x^2} = \frac{x_0\left(\frac{g^2x^2}{(gx^2+1)^{3/2}} - \frac{g}{\sqrt{gx^2+1}}\right)}{\sqrt{gx_0^2+1}} + 1$$
$$\frac{\delta V(0)}{\delta x^2} = 1$$

This shows us that $x = 0$ is a local minimum. Not wanting to solve the algebra, simply by looking at the plot of $V(x)$ below we can see that as long as the other extrema exist they will be local minima:



These minimum will only be real (i.e. exist) if $(g^2x_0^2 - gx_0^2) \geq 1$

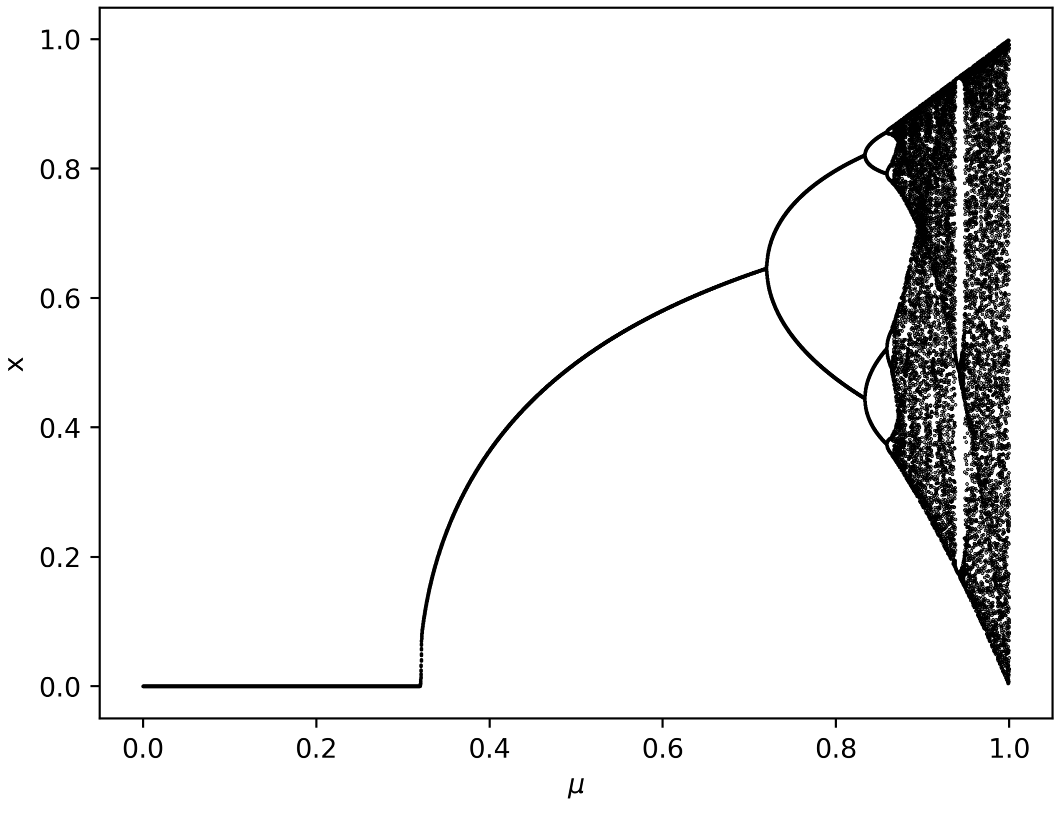


(b)

(c) The behaviour of the particle will depend on the exact value of g and x_0. If $g <= \frac{x_0+\sqrt{x_0^2+4}}{2x_0}$, or if $x_0 \gg 0$, then the particle will oscillate around $x = 0$. Otherwise, the particle will oscillate around the other extrema: $\pm\frac{\sqrt{g^2x_0^2-gx_0^2-1}}{\sqrt{g^2x_0^2+g}}$.

2. (a)
```
import numpy as np
import matplotlib.pyplot as plt
from math import pi, sin
from tqdm import tqdm


def f(x_, mu_=0.5):
    return mu_ * sin(pi * x_)


def f2(x_, a=0.5):
    if x_ < 0.5:
        return 2 * a * x_
    else:
        return 2 * a * (1 - x_)


num_points = 1000
mu_array = []
x_array = []
```

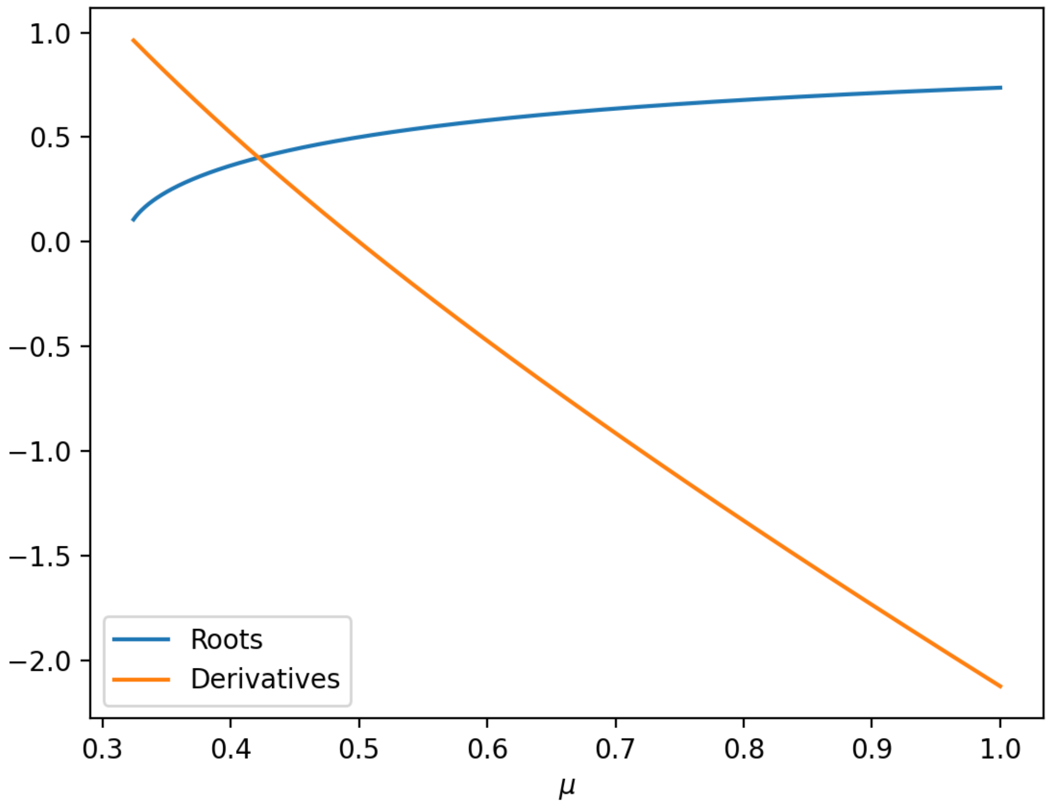
```
for mu in tqdm(np.linspace(0, 1, num_points * 10)):
    x0 = 10 ** (-5)
    x = [x0]
    for _ in range(num_points):
        x0 = f(x0, mu_=mu)
        x.append(x0)
    final_slice = int(num_points / 100)
    mu_array += [mu] * final_slice
    x_array += x[-final_slice:]
plt.scatter(mu_array, x_array, c="black", s=0.1)
plt.ylabel("x")
plt.xlabel(r"$\mu$")
plt.savefig("bifurcation.png", dpi=500)
plt.show()
```



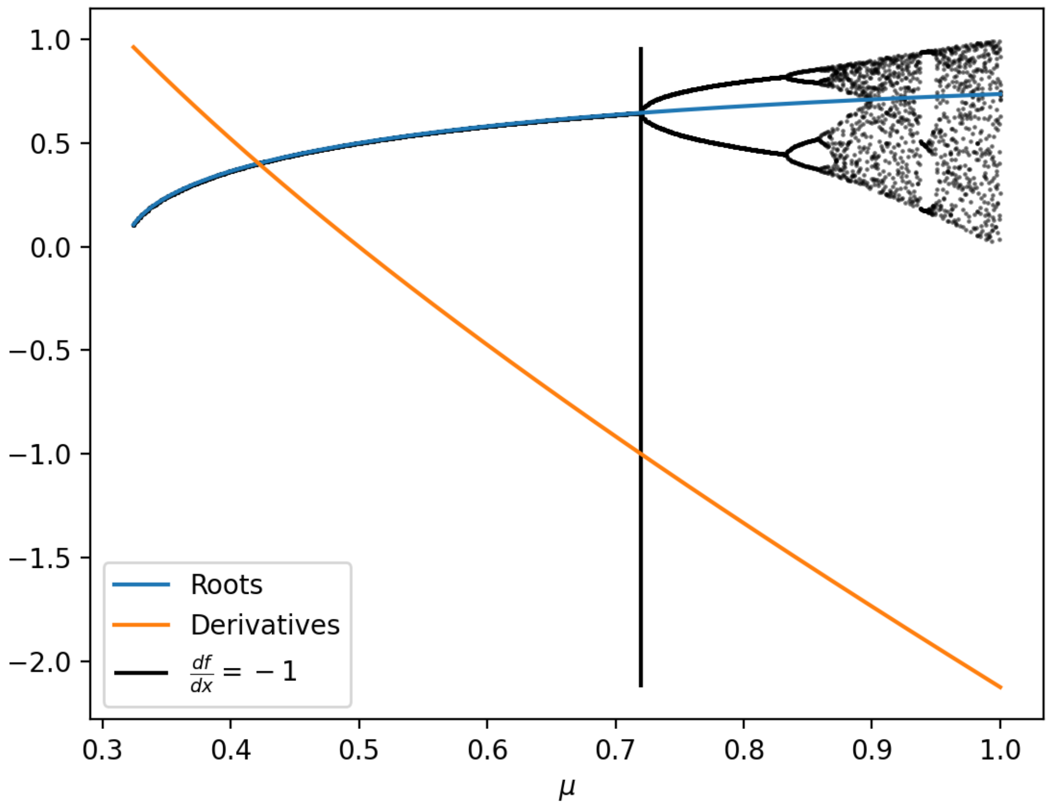
(b) Fixed points are where $x_{n+1} = x_n = \mu \sin(\pi x_n)$. The first trivial solution is where $x_n = 0$ as $\mu \sin(\pi \cdot 0) = 0$. The second solution is more complicated; taking a derivative in terms of x_n and solving for μ:

$$\frac{\delta}{\delta x_n}\left(\mu \sin\left(\pi x_n\right) - x_n\right) = \pi\mu\cos(\pi x) - 1$$
$$\mu = \frac{\sec(\pi x_n)}{pi}$$
$$\mu_0 = \frac{1}{\pi}$$

(c) The following plot illustrates the roots of the function $\mu\sin(\pi x) - x = 0$ plotted against the derivative of the function $\mu\sin(\pi x)$ evaluated at that root. The plot shows that the fixed point is stable until the derivative gets smaller than -1:



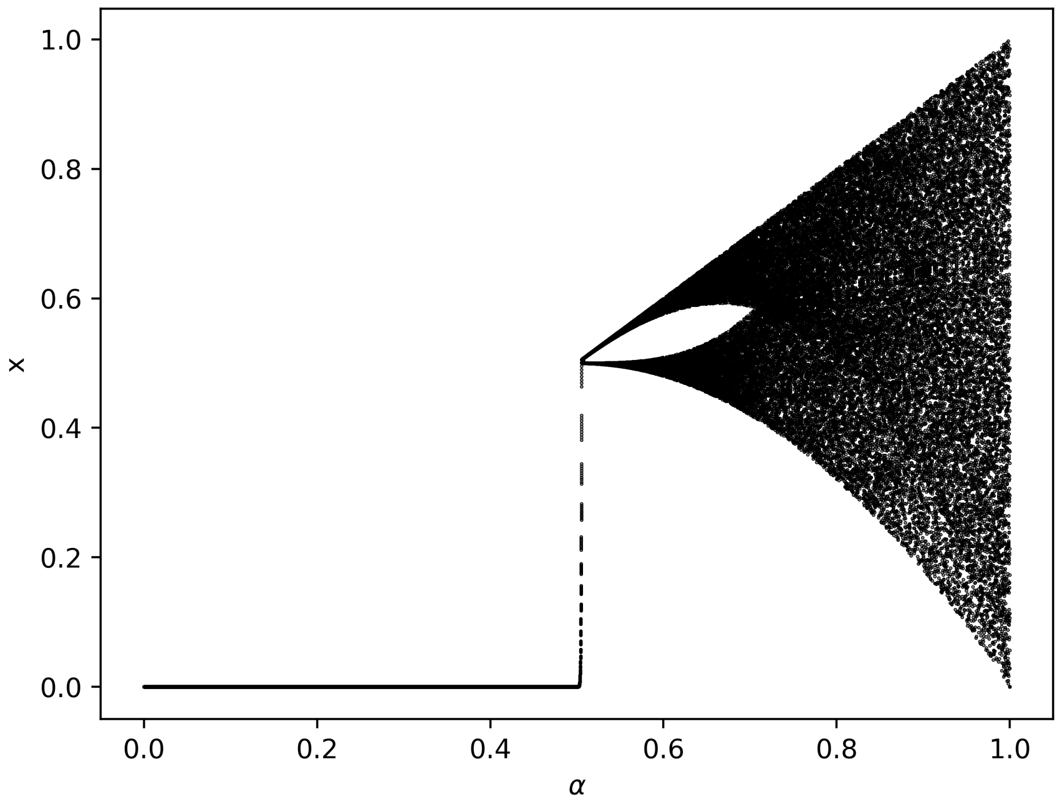
(d) This is illustrated even more clearly by superimposing the bifurcation diagram onto the plot, and highlighting where the derivative passes -1, as this corresponds to where the fixed point becomes unstable:



```
import numpy as np
from scipy.optimize import minimize_scalar, root_scalar
import matplotlib.pyplot as plt
from numpy import sin, cos, pi
from tqdm import tqdm


def f(x_, mu_=0.5):
    return mu_ * sin(pi * x_)


def d_f(x_, mu_=0.5):
    return pi * mu_ * cos(pi * x_)


def d_d_f(x_, mu_=0.5):
    return -pi ** 2 * mu_ * sin(pi * x_)


x = np.linspace(0, 1, 100)
roots = []
derivatives = []
mu_s = []
```

```
    diffs = []
    num_points = 1000
    final_slice = int(num_points / 100)
    x0 = 0.1
    for mu in tqdm(np.linspace(0, 1, 1000)):
        try:
            sol = root_scalar(lambda i: f(i, mu) - i, bracket=[0.1, 1])
            bifurcation_x = []
            for _ in range(num_points):
                x0 = f(x0, mu=mu)
                bifurcation_x.append(x0)
            plt.scatter([mu] * final_slice, bifurcation_x[-final_slice:], al
            diffs.append(sol.root - f(sol.root, mu=mu))
            roots.append(sol.root)
            derivatives.append(d_f(sol.root, mu=mu))
            mu_s.append(mu)
        except ValueError:
            pass
    derivatives = np.array(derivatives)
    d_root_1 = mu_s[np.abs(derivatives + 1).argmin()]
    plt.plot(mu_s, roots, label="Roots")
    plt.plot(mu_s, derivatives, label="Derivatives")
    plt.vlines(d_root_1, min(derivatives), max(derivatives), color="black",
    plt.xlabel(r"$\mu$")
    plt.axis("on")
    plt.legend()
    plt.savefig("stability_bifurcation.png", dpi=200)
    plt.show()
```

3. Based on the bifurcation diagram it is clear that it begins to be chaotic at $\alpha = \frac{1}{2}$. Thus it is chaotic between $\frac{1}{2} < \alpha \leq 1$



4. (a) The paper essentially used several methods to find patterns in the evolution of the dynamics of a spring-pendulum system. The methods they used included poincare maps at different energies, Lyapunov exponents, Correlation functions, and power spectra. The main result of the paper was that at higher energies the system displays greater regions of regularity, as demonstrated by the poincare maps and confirmed by most of the other methods.

   (b) The panels went from large negative energies to large positive energies, and showed that going in that direction increased the area of the phase space that showed more regularity. This can be seen by the fact that the negative energy panel had very dense areas of chaos, and that the largest reagions of regularity where found in the large energy poincare map of panel (c).)