Examining a Nonlinear Differential Equation

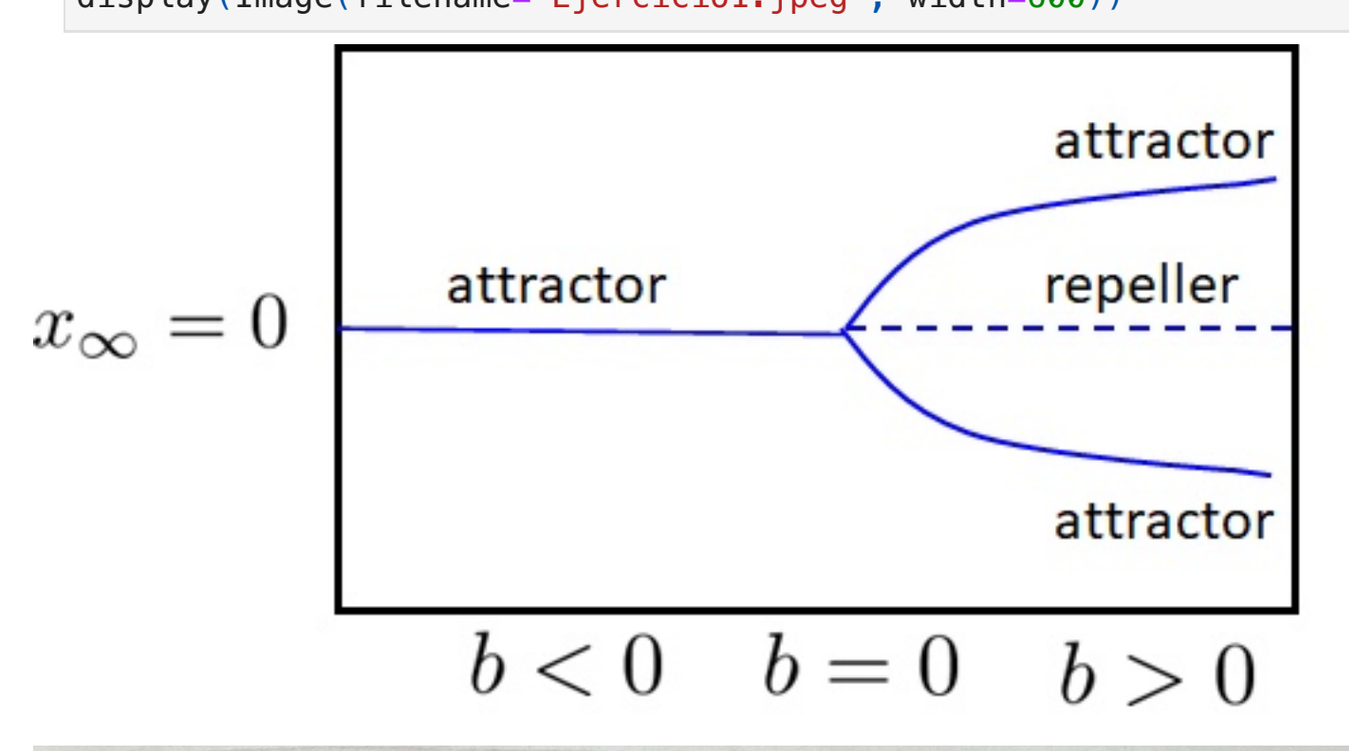
Consider the following homogeneous nonlinear differential equation of order 1:

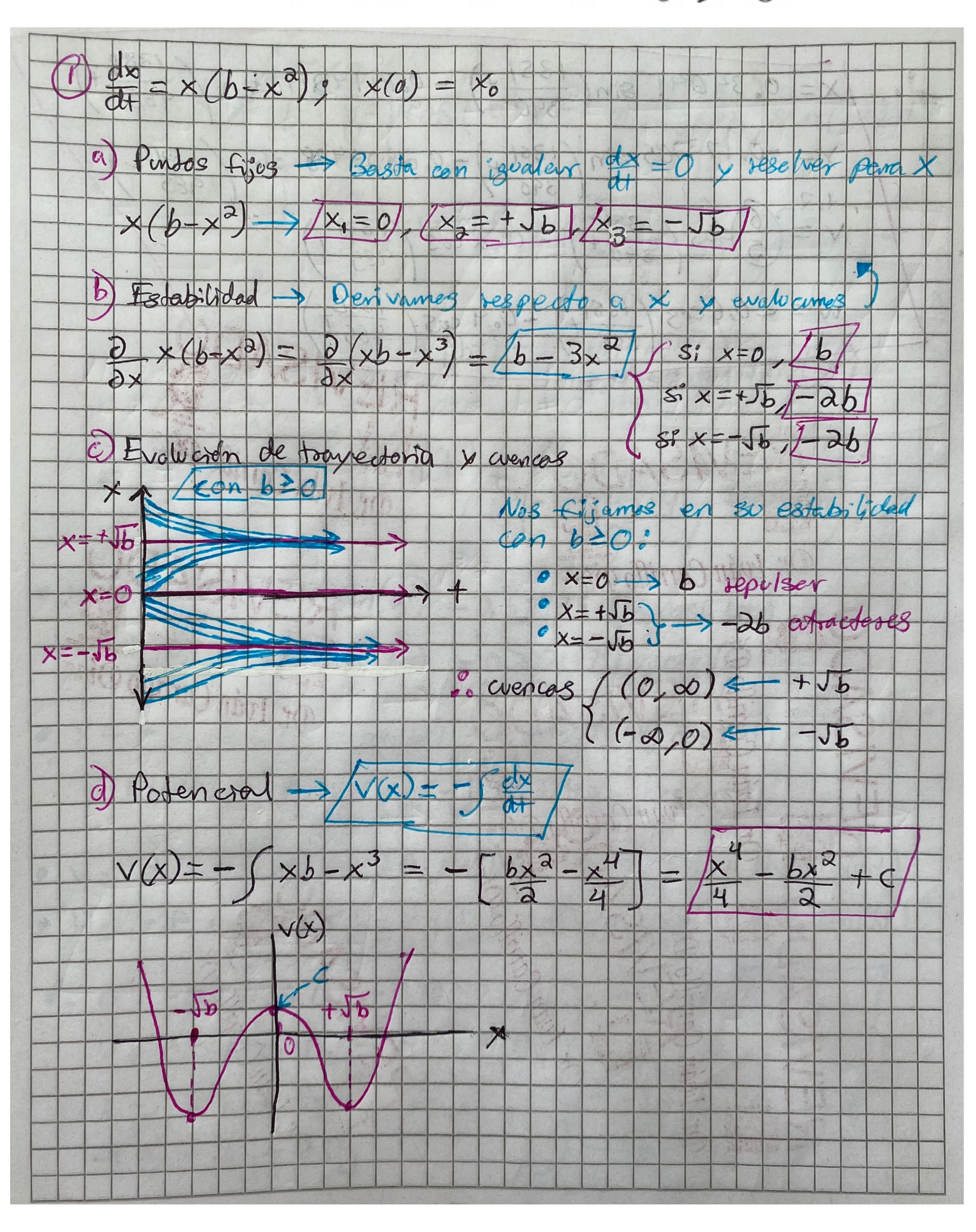
$$\frac{dx}{dt} = x(b - x^2); x(0) = x_0$$

a) Find fixed points (remember that the sign of b was crucial)
b) Evaluate its stability
c) Draw the trajectory evolution and identify basins
d) Calculate the potential and draw it

A very important result was the following branch diagram.

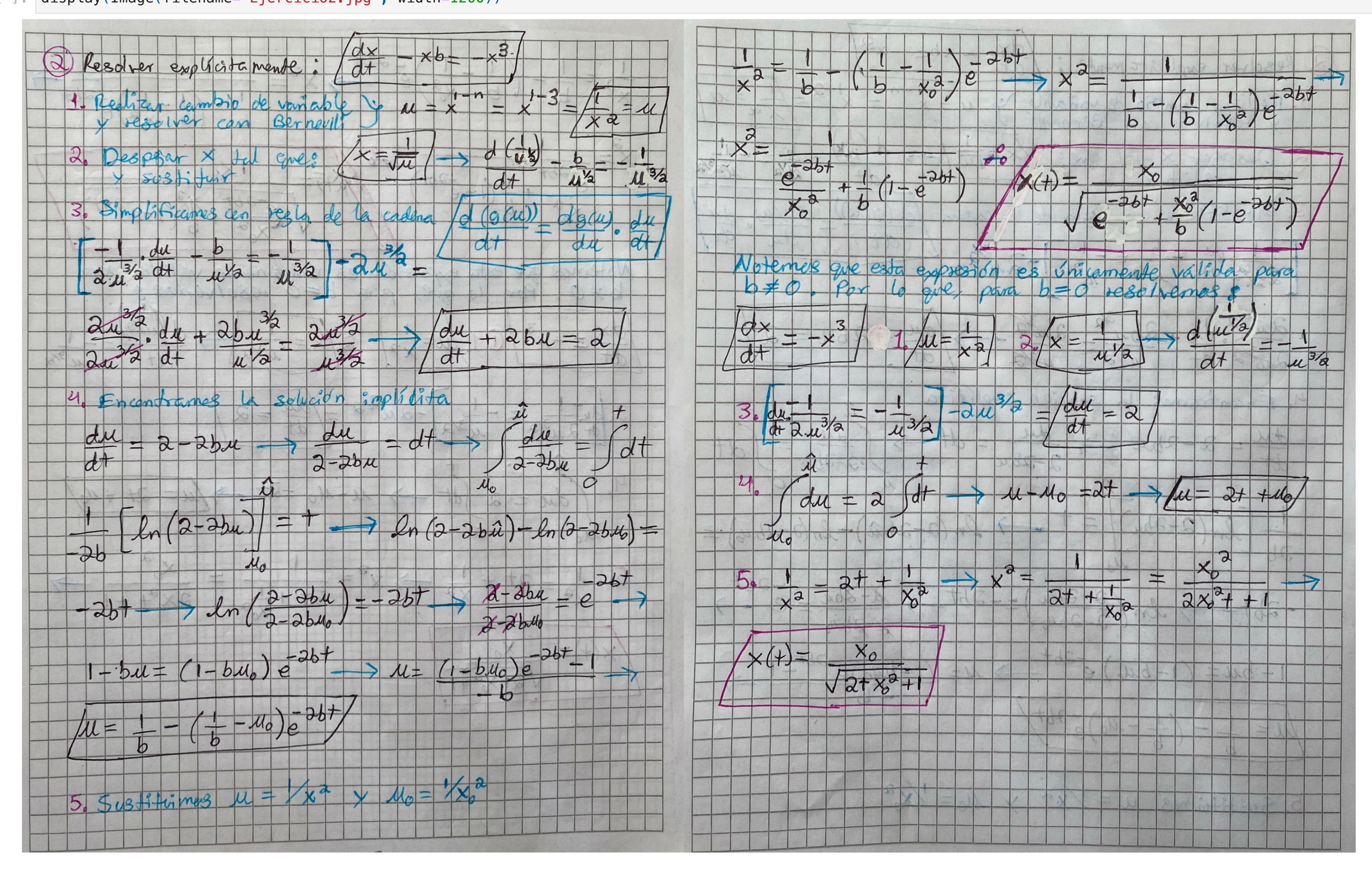
In [ ]:
```
from IPython.display import display, Image
display(Image(filename="Imagen1.jpg"))
display(Image(filename="Ejercicio1.jpeg", width=600))
```





- Let's solve the equation explicitly. This is a Bernoulli equation, so a simple change of variable will allow you to find the solution.

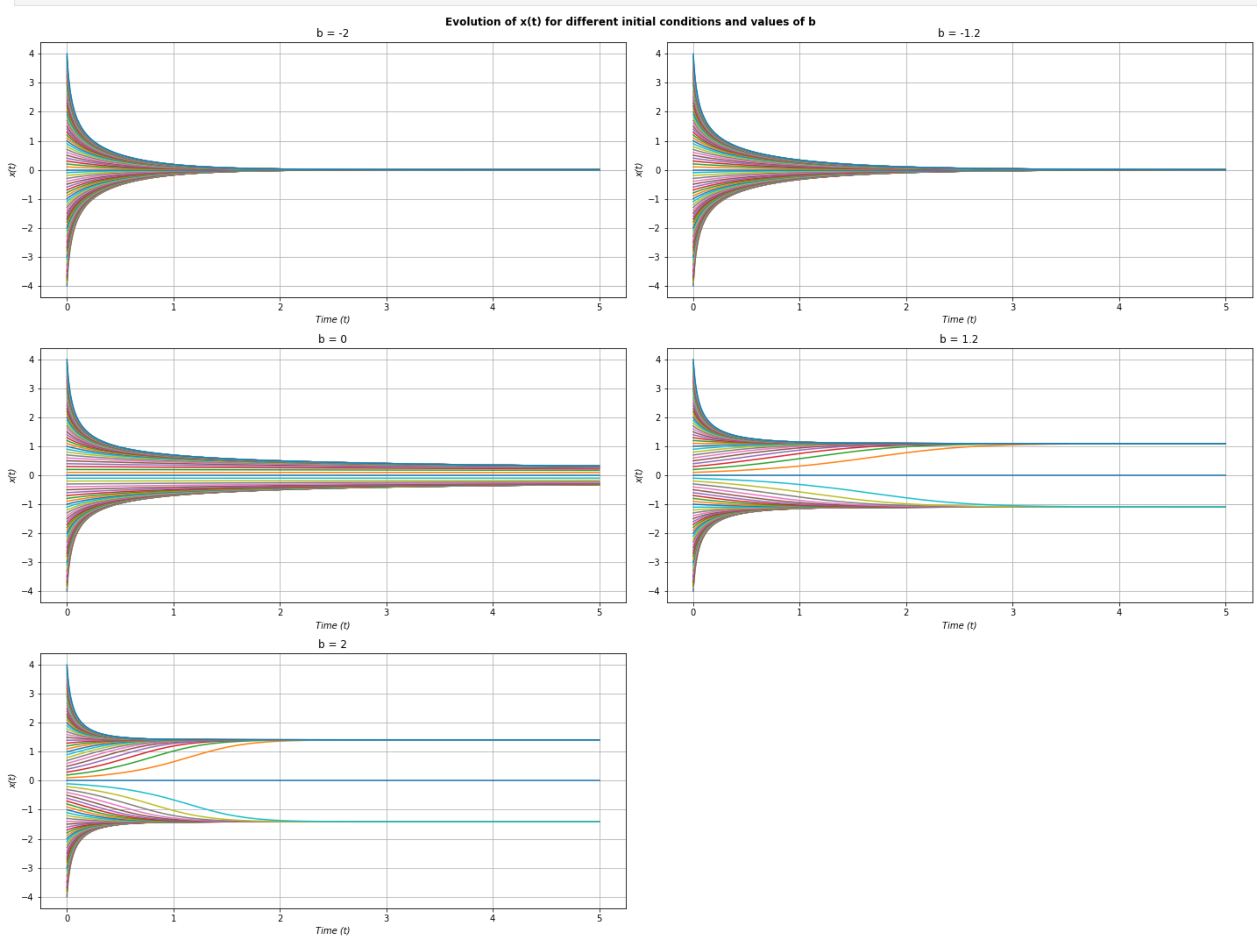
In [ ]:
```
display(Image(filename="Ejercicio2.jpg", width=1200))
```



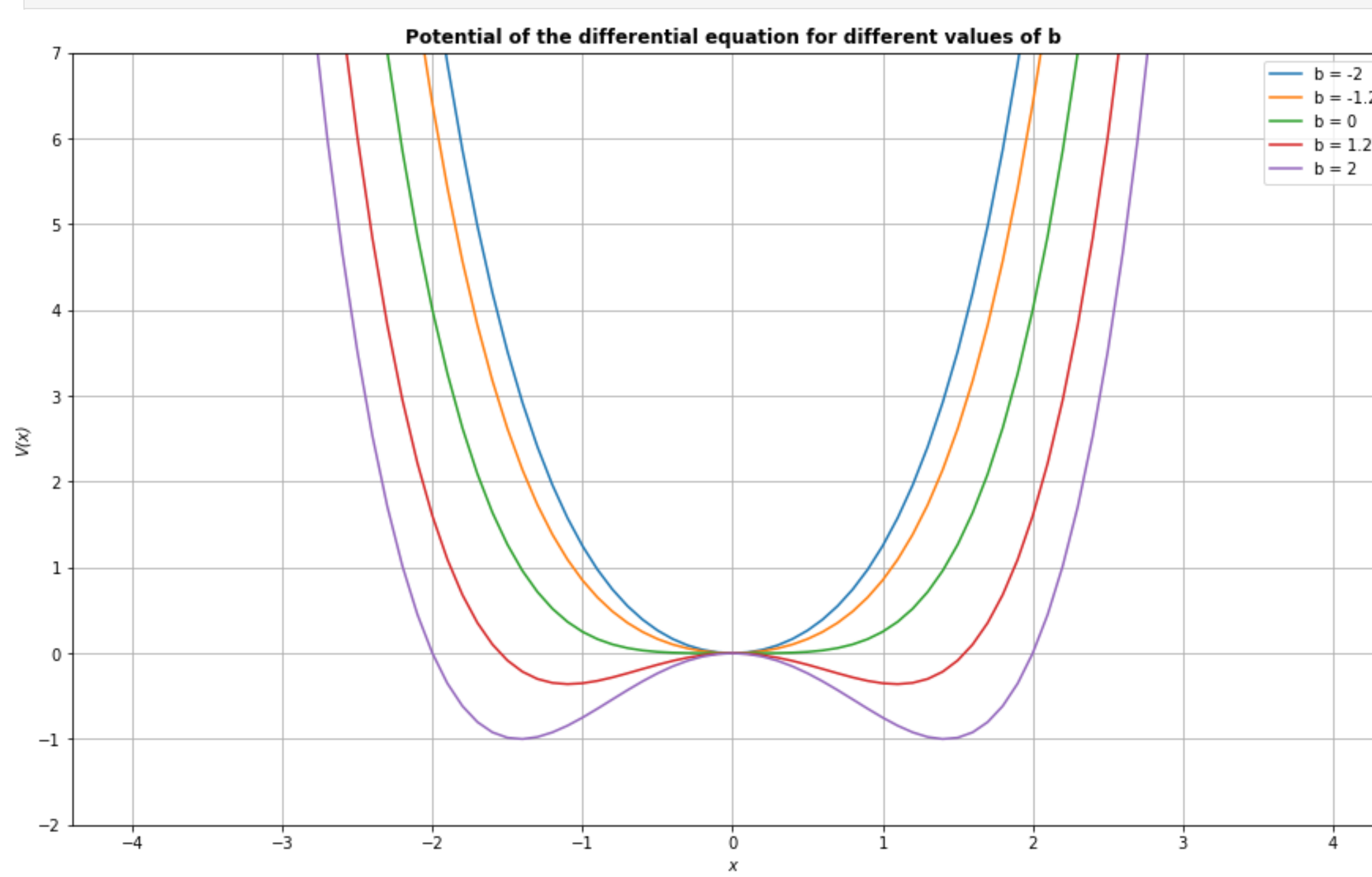
- Now that you have the solution, let's graph it and explore behaviors.
  a) Let's vary b.
  b) For each value b, produce a graph with the evolution from different initial conditions.
  c) Graph the potential functions for the different values of b that you chose. You can choose the integration constant C = 0.

In [ ]:
```
# We import the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import matplotlib.lines as mlines

# Function that returns the solution of the differential equation for b ≠ 0
def solution_b(t, x0, b):
    return x0 / np.sqrt(np.exp(-2*b*t) + (x0**2 / b) * (1 - np.exp(-2*b*t)))

# Function that returns the solution of the differential equation for b = 0
def solution(t, x0):
    return x0 / np.sqrt(2 * t * (x0**2) + 1)

# Function to calculate potential
def potential(x, b):
    return (x**4 / 4) - (b * x**2 / 2)

# Define parameters
b_values = [-2, -1.2, 0, 1.2, 2]
x0_values = np.arange(-4, 4.1, 0.1)
t_values = np.linspace(0, 5, 500)
```

In [ ]:
```
# Graph evolution for different initial conditions and values of b
plt.figure(figsize=(20, 15))

for i, b in enumerate(b_values):
    plt.subplot(3, 2, i + 1)
    plt.title(f'b = {b}')
    plt.xlabel('Time (t)', fontstyle='italic')
    plt.ylabel('x(t)', fontstyle='italic')

    # Solve for x with each initial condition x_0 and graph
    for x0 in x0_values:
        if b != 0:
            x_values = solution_b(t_values, x0, b)
        else:
            x_values = solution(t_values, x0)
        plt.plot(t_values, x_values)
        plt.grid(True)

plt.suptitle('Evolution of x(t) for different initial conditions and values of b', fontweight='bold')
plt.tight_layout()
plt.show()
```

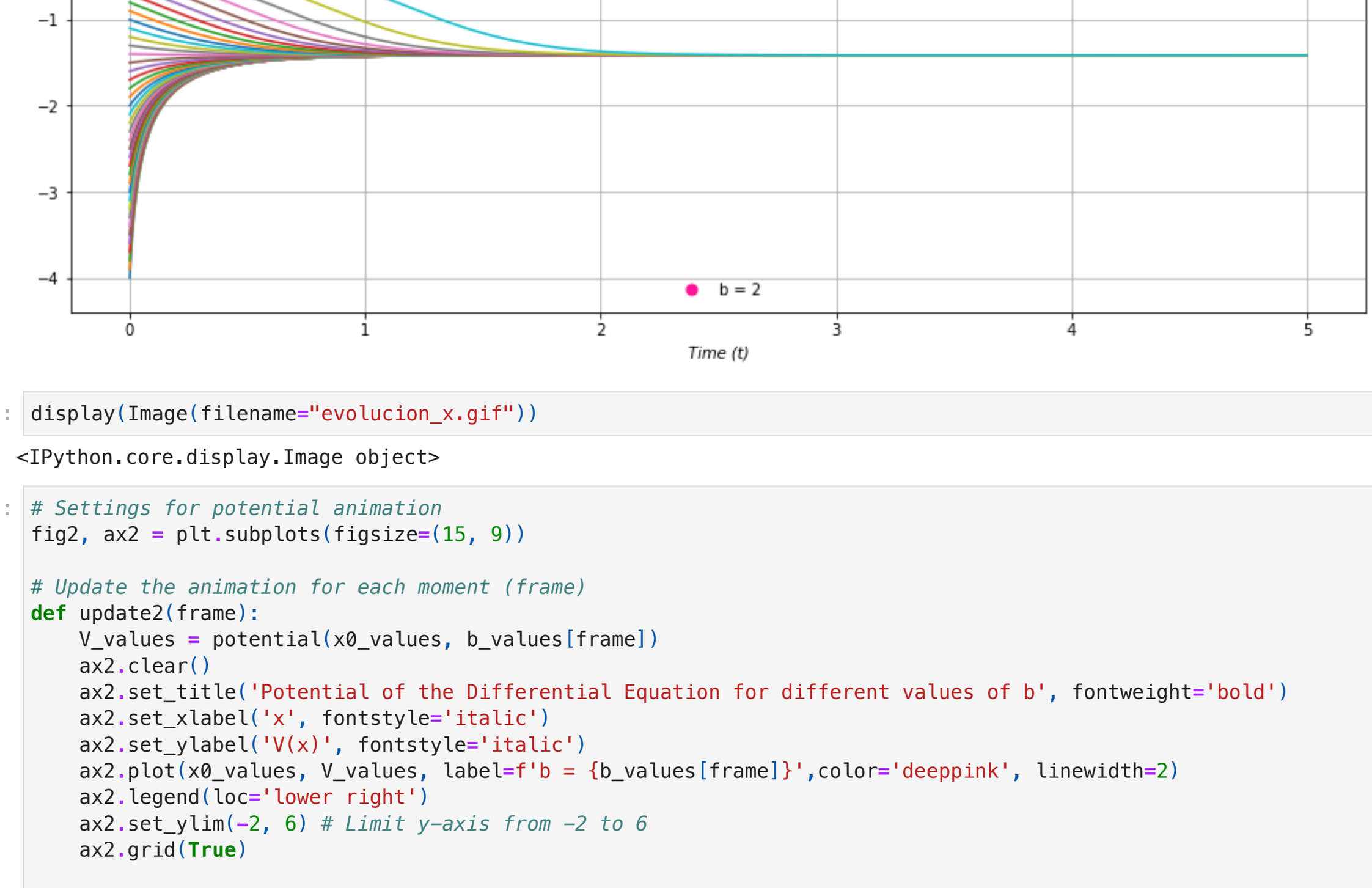


In [ ]:
```
# Graph potential function for different values of b
plt.figure(figsize=(15, 9))

# We solve for V with each value of b and graph
for b in b_values:
    V_values = potential(x0_values, b)
    plt.plot(x0_values, V_values, label=f'b = {b}')

plt.title('Potential of the differential equation for different values of b', fontweight='bold')
plt.xlabel('x', fontstyle='italic')
plt.ylabel('V(x)', fontstyle='italic')
plt.ylim(-2, 7) # Limit y-axis from -2 to 7
plt.legend()
plt.grid(True)
plt.show()
```



- Optional challenge: Produce 2 gifs, one that shows how the trajectories of (b) and another that shows the potentials of (c) changing when the parameter b varies

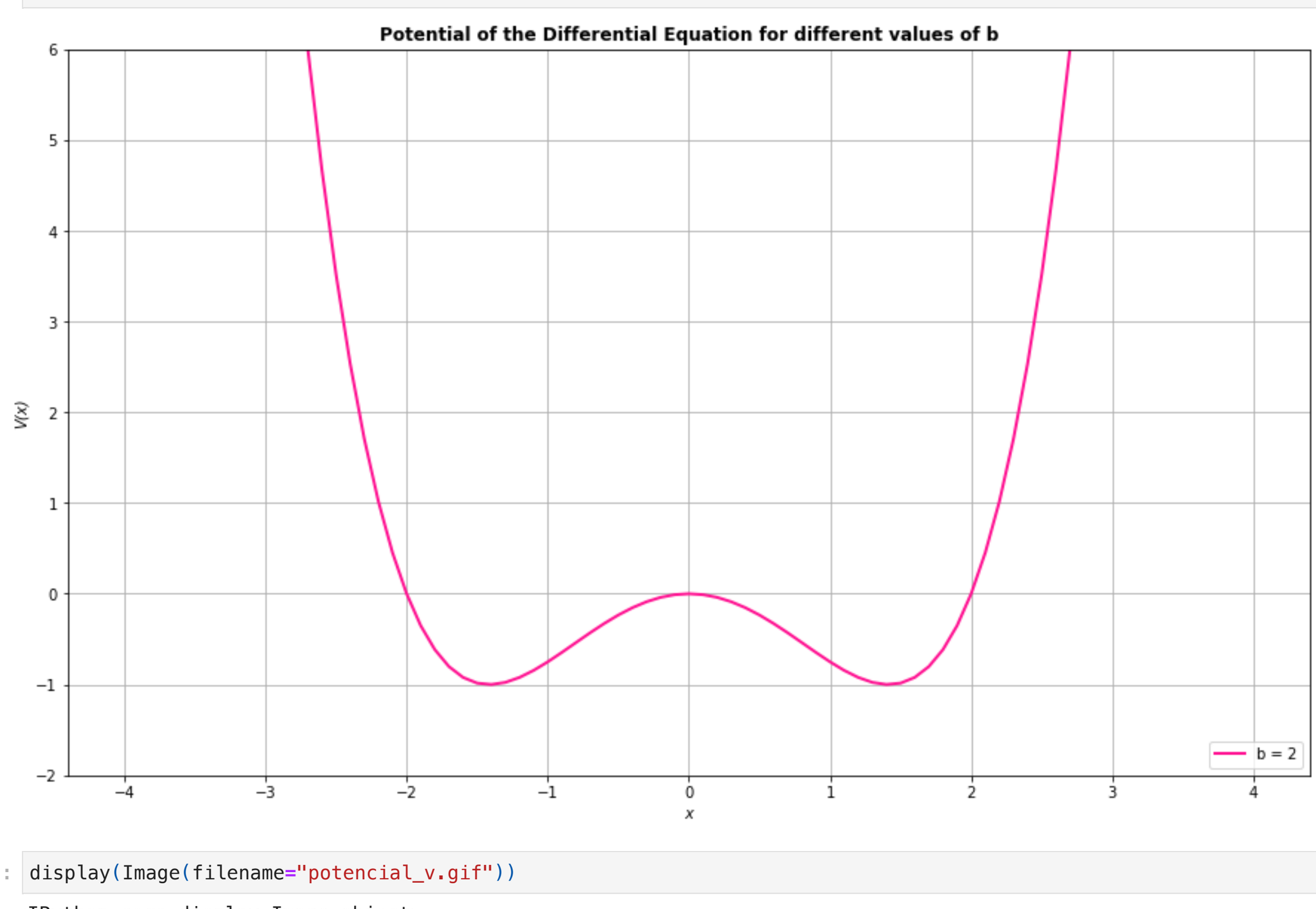
In [ ]:
```
# Settings for x(t) evolution animation
fig1, ax1 = plt.subplots(figsize=(15, 9))

# Update the animation for each moment (frame)
def update1(frame):
    ax1.clear()
    ax1.set_title('Evolution of x(t) for different initial conditions and values of b', fontweight='bold')
    ax1.set_xlabel('Time (t)', fontstyle='italic')
    ax1.set_ylabel('x(t)', fontstyle='italic')
    ax1.grid(True)

    # Solve for x with each initial condition x_0
    for x0 in x0_values:
        if b_values[frame] != 0:
            x_values = solution_b(t_values, x0, b_values[frame])
        else:
            x_values = solution(t_values, x0)
        ax1.plot(t_values, x_values, label=f'b = {b_values[frame]}')

    # Place custom legend
    pink_dot = mlines.Line2D([], [], color='deeppink', marker='o', linestyle='None', markersize=7, label=f'b = {b_values[frame]}')
    ax1.legend(handles = [pink_dot], frameon=False, loc='lower center')

# Save animation to gif file using Pillow as writer
ani1 = FuncAnimation(fig1, update1, frames=len(b_values), interval=1000, repeat=True)
ani1.save('evolucion_x.gif', writer='pillow')
```



In [ ]:
```
display(Image(filename="evolucion_x.gif"))
```

<IPython.core.display.Image object>

In [ ]:
```
# Settings for potential animation
fig2, ax2 = plt.subplots(figsize=(15, 9))

# Update the animation for each moment (frame)
def update2(frame):
    V_values = potential(x0_values, b_values[frame])
    ax2.clear()
    ax2.set_title('Potential of the Differential Equation for different values of b', fontweight='bold')
    ax2.set_xlabel('x', fontstyle='italic')
    ax2.set_ylabel('V(x)', fontstyle='italic')
    ax2.plot(x0_values, V_values, label=f'b = {b_values[frame]}',color='deeppink', linewidth=2)
    ax2.legend(loc='lower right')
    ax2.set_ylim(-2, 6) # Limit y-axis from -2 to 6
    ax2.grid(True)

# Save animation to gif file using Pillow as writer
ani2 = FuncAnimation(fig2, update2, frames=len(b_values), interval=1000, repeat=True)
ani2.save('potencial_v.gif', writer='pillow')
```

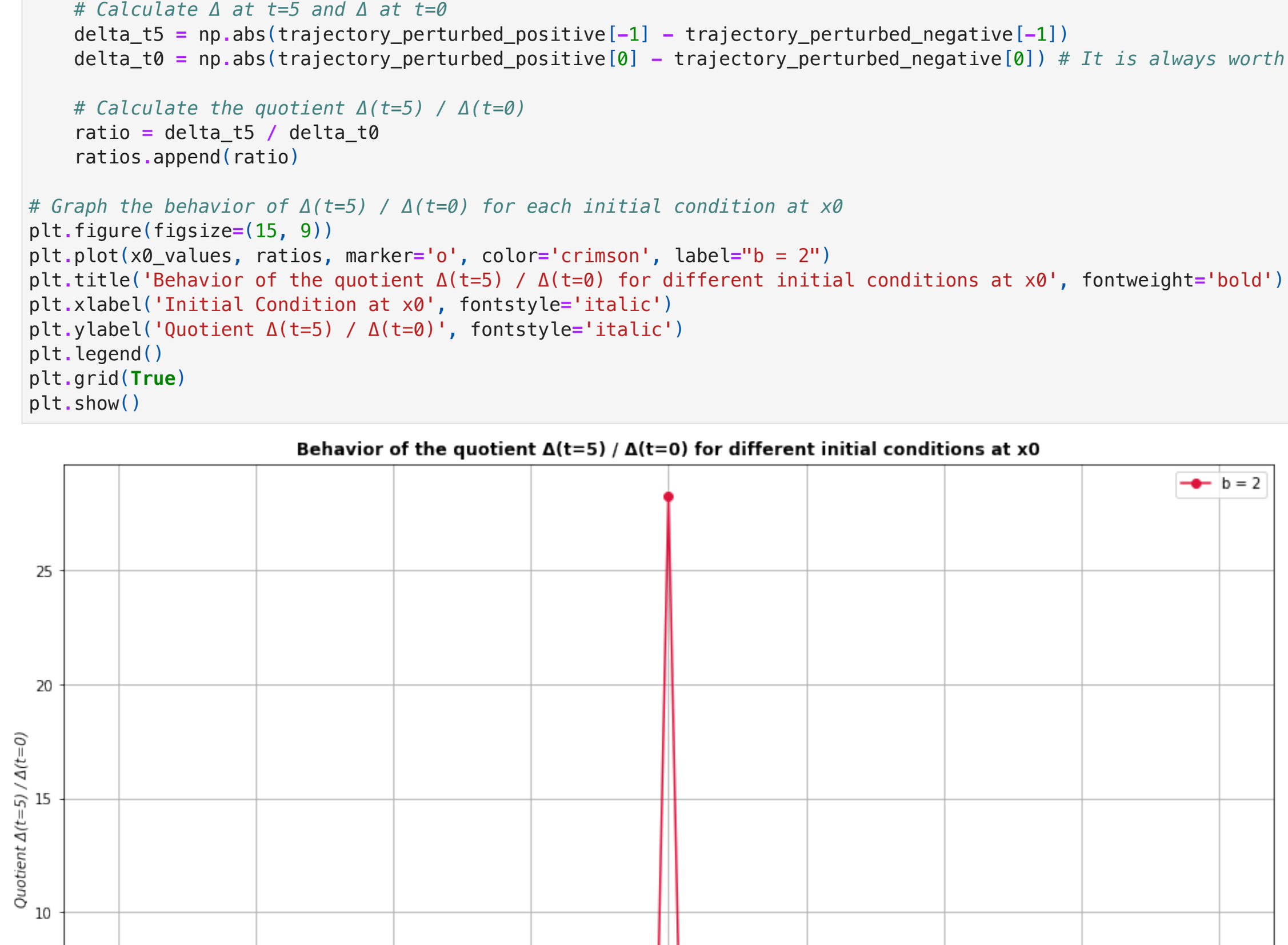


In [ ]:
```
display(Image(filename="potencial_v.gif"))
```

<IPython.core.display.Image object>

- In real life, we often do not know precisely the initial conditions of our equations. For each initial condition you defined in (b), and a value b = 2, consider 2 new initial values: $x_0 \pm \epsilon$ (a small perturbation, for example ϵ = 0.05). The initial distance between the two perturbed initial conditions is 2ϵ. Calculate the trajectories from these points and find the two corresponding values at t = 5. Calculate the quotient: $\frac{\Delta(t=5)}{\Delta(t=0)}$. Repeat this for all the initial conditions in (b) and make a graph. How does the graph behave? In what 'region' of the initial values is precision very important?

In [ ]:
```
# Define parameters
b = 2
epsilon = 0.05
t_values = np.array([0, 5]) # we are only interested in time 0 and 5

# List that stores the quotient Δ(t=5) / Δ(t=0) for each initial condition at x0
ratios = []

for x0 in x0_values:
    perturbed_x0_positive = x0 + epsilon
    perturbed_x0_negative = x0 - epsilon

    # Calculate trajectories for perturbed initial conditions
    trajectory_perturbed_positive = solution_b(t_values, perturbed_x0_positive, b)
    trajectory_perturbed_negative = solution_b(t_values, perturbed_x0_negative, b)

    # Calculate Δ at t=5 and Δ at t=0
    delta_t5 = np.abs(trajectory_perturbed_positive[-1] - trajectory_perturbed_negative[-1])
    delta_t0 = np.abs(trajectory_perturbed_positive[0] - trajectory_perturbed_negative[0]) # It is always worth 2*epsilon

    # Calculate the quotient Δ(t=5) / Δ(t=0)
    ratio = delta_t5 / delta_t0
    ratios.append(ratio)

# Graph the behavior of Δ(t=5) / Δ(t=0) for each initial condition at x0
plt.figure(figsize=(15, 9))
plt.plot(x0_values, ratios, marker='o', color='crimson', label="b = 2")
plt.title('Behavior of the quotient Δ(t=5) / Δ(t=0) for different initial conditions at x0', fontweight='bold')
plt.xlabel('Initial Condition at x0', fontstyle='italic')
plt.ylabel('Quotient Δ(t=5) / Δ(t=0)', fontstyle='italic')
plt.legend()
plt.grid(True)
plt.show()
```



1. **Graph behavior:**
   - The quotient $\frac{\Delta(t=5)}{\Delta(t=0)}$ is quite small for most initial conditions at x_0, except for $x_0 = 0$.
   - For $x_0 = 0$, $\frac{\Delta(t=5)}{\Delta(t=0)} \approx 28.284$, indicating that, small variations in the initial conditions around $x_0 = 0$ lead to more notable differences in the trajectories over time.
2. **Region of importance in the initial values:**
   - Accuracy is most critical around the initial condition $x_0 = 0$. In this region, initial perturbations have a more significant impact on the evolution of the system over time. In other words, the trajectories are more sensitive to small perturbations around this point.
   - For other initial conditions, the small perturbation (ϵ=0.05) does not have a significant impact on the relationship $\frac{\Delta(t=5)}{\Delta(t=0)}$.