HW1.

1. **[1+1+1+1pt]** Let $f(x) = e^x - x^2 - 2x - 1$ and $g(x) = 2\ln(x+1)$, where $x \in (-1, \infty)$.

   (a) Verify that the roots of $f(x)$ are the same as the fixed points of $g(x)$.

(a) The domain of g is $(-1, +\infty)$

if $g(x) = 2\ln(x+1) = x$

$$\ln(x+1)^2 = x$$

$$e^{\ln(x+1)^2} = e^x$$

$$(x+1)^2 = e^x$$

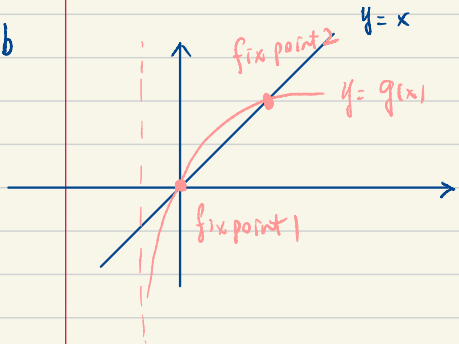
$$\Rightarrow \quad e^x - x^2 - 2x - 1 = 0$$

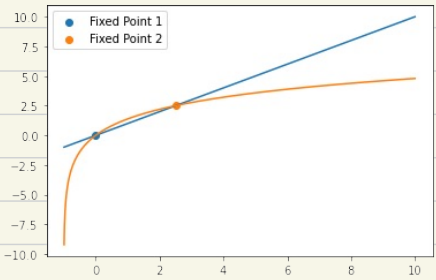
which indicates $f(x) = 0$

Thus, the roots of $f(x)$ are the same as the fixed points of g on $(-1, +\infty)$

(b) Sketch $y = g(x)$, $y = x$ and indicate all fixed points (sketch by hand or plot with a computer). You don't need to calculate them. (Hint for the sketch: Note that $g'(0) > 1$).

b





(c) Use Brouwer's fixed point theorem to argue the existence of a fixed point ξ in the interval $[a, b] = [e - 1, e^2 - 1]$.

Proof.

As: $g(e-1) = 2\ln(e-1+1) = 2 \geq e-1 = a$

$g(e^2-1) = 2\ln(e^2-1+1) = 4 \leq e^2-1 = b$

$\Leftrightarrow \mathrm{Im}(g) \subset [a,b]$

and g is Continuous on $[a, b]$

By Brouwer fixed point Thm.

$\exists$ a fixed point in the interval $[e-1, e^2-1]$

(d) Use the contraction mapping theorem to show that ξ is the only fixed point in the interval $[e-1, e^2-1]$.

Proof. According to the contraction mapping theorem.

It suffices to verify g is contraction on $[e-1, e^2-1]$

$g'(x) = \frac{2}{x+1}$ which is

1° decreasing on $[e-1, e^2-1]$

2° remains positive.

Therefore, as $g'(e-1) = \frac{2}{e} < 1$. we have $|g'(x)| < 1$ on $[e-1, e^2-1]$

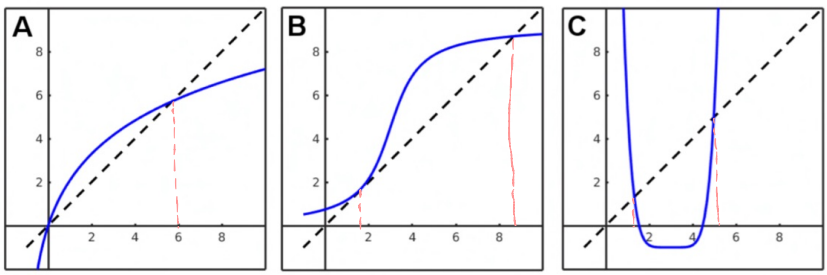
which implies g is contraction on $[e-1, e^2-1]$

2. **[2+2pt]** Stability of fixed points.

(a) For each of the three functions (solid lines) depicted below,

(i) Write down the approximate values of the fixed points (as estimated by eye).

(ii) State for each fixed point, whether it is stable, unstable or neither of the two.



A. Fixed point

$x_1 = 0$ unstable

$x_2 = 6$ stable

B Fixed Point

$x_1 = 1.8$ neither of 2

$x_2 = 8.3$ stable

C. Fixed Point

$x_1 = 1.6$ unstable

$x_2 = 4.6$ unstable.

(b) You are given the first ten iterates of two sequences, x_k and y_k, both of which converge to zero:

k	x_k	y_k
0	1.0000000000000	1.0000000000000
1	0.3000000000000	0.6648383611734
2	0.0900000000000	0.4404850619261
3	0.0270000000000	0.2895527955097
4	0.0081000000000	0.1869046766665
5	0.0024300000000	0.1155100169867
6	0.0007290000000	0.0638472856062
7	0.0002187000000	0.0254178900244
8	0.0000656100000	0.0032236709627
9	0.0000196830000	0.0000080907744
10	0.0000059049000	0.0000000000001

(i) What is (most likely) the order of convergence of x_k? Explain your answer.
(ii) What is (most likely) the order of convergence of y_k? Explain your answer[1].

(i) x_k Linear Order of Convergence.

As $\frac{|x_{k+1} - 0|}{|x_k - 0|} \doteq 0.3 < 1$ which indicates Linear Growth.

(ii) y_k Quadratic Order of Convergence.

Every iteration, it is doubling its digit. (for latter iterations)

which indicates $\frac{|y_{k+1} - 0|}{|y_k - 0|^2} = \lambda \in \mathbb{R}^+$

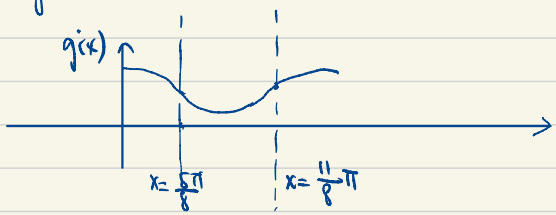
3. **[2+1+1pt]** Let g be defined on $[5\pi/8, 11\pi/8]$.

$$g(x) = x + 0.8\sin x.$$

(a) Determine the (smallest possible) Lipschitz constant L.

(a) proof. According to mean value Thm, it suffices to compute $\max |g'|$ at $[5\pi/8, 11\pi/8]$

$g'(x) = 1 + 0.8\cos x$



Here we can see $|g'(x)|_{\max} = \max\{g'(\frac{5\pi}{8}), g'(\frac{11}{8}\pi)\}$

$= 1 + 0.8\cos(\frac{5\pi}{8}) \approx 0.6938$

(b) How many iterations are required to increase the accuracy by a factor of 100, i.e., given some $x_0 \in [1, 2]$, when what is k such that you can guarantee that $|x_k - \xi| \le 10^{-2}|x_0 - \xi|$?

Sol Assme we are to compute fixed point.

$$\frac{|x_{k+1} - \xi|}{|x_k - \xi|} = \frac{|g(x_k) - \xi|}{|x_k - \xi|} = g'(\delta) \le L = 0.6938$$

$\delta \in [x_k, x_{k+1}]$ ← Mean Value Thm.

Therefore. iteration required is

$$k = \lceil \log_L 10^{-2} \rceil = \lceil 12.59 \rceil = 13$$

(c) Starting with initial guess $x_0 = 5\pi/8$, compute the first fixed point iterate x_1 and use it, together with the Lipschitz constant you found, to compute after how many fixed point iterations k you can be certain that $|\xi - x_k| < 10^{-10}$.

$x_0 = \frac{5\pi}{8}$

$x_1 = g(x_0) = x_0 + 0.8 \sin(\frac{5}{8}\pi) = 2.7026$

By the formula given in the lecture,

$$k_0(\varepsilon) = \left[\frac{\ln|x_1 - x_0| - \ln|\varepsilon(1-L)|}{\ln(\frac{1}{L})} \right] + 1$$

plug in $L = 0.6958$ $\varepsilon = 10^{-10}$

we get

$$k_0(10^{-10}) = 66$$

4. **[1+1+1+1pt]** The equation

$$f(x) := x^2 - 5 = 0,$$

has a single root $\xi = \sqrt{5} \approx 2.2361\ldots$ in the interval $[1, 3]$. Consider the fixed point iteration $x_{k+1} = g(x_k)$, where g is defined as one of the following options:

- $g_1(x) = 5 + x - x^2$,
- $g_2(x) = 1 + x - \frac{1}{5}x^2$,
- $g_3(x) = \frac{1}{2}x + \frac{5}{2x^2}$.

(a) Identify the fixed point functions for which the fixed point is also a root of f.

(a)

$g_1(x) = x \iff 5 + x - x^2 = x \iff x^2 - 5 = 0 \iff f(x) = 0$

$g_2(x) = x \iff 1 + x - \frac{1}{5}x^2 = x \iff 1 - \frac{1}{5}x^2 = 0 \iff x^2 - 5 = 0 \iff f(x) = 0$

$g_3(x) = \frac{1}{2}x + \frac{5}{2x^2} \overset{=x}{\iff} \frac{5}{2x^2} - \frac{1}{2}x = 0 \iff 5 - x^3 = 0 \iff f(x) = 0$

g_1 and g_2 work. g_3 doesn't work.

(b)

(b) For the cases where computing the fixed point is equivalent with solving $f(x) = 0$, discuss whether the fixed point iteration is guaranteed to converge in some neighborhood of ξ.

It suffices to check if $|g'(\xi)| < 1$ or not

$g_1'(x) = 1 - 2x \qquad |g_1'(\xi)| = |1 - 2\sqrt{5}| > 1$

$g_2'(x) = 1 - \frac{2}{5}x \qquad |g_2'(\xi)| = |1 - \frac{2\sqrt{5}}{5}| < 1$

Therefore. g_1 is Not guaranteed to Converge in a neighbourhood of ξ

g_2 is guaranteed to Converge in a neighbourhood of ξ.

(c) If the iteration in b) is guaranteed to converge, compute the value of

$$\lim_{k\to\infty} \frac{|x_{k+1}-\xi|}{|x_k-\xi|}$$

Hint: Everything is easier with the mean value theorem!

proof.

g_2 is guaranteed to Converge.

$$\frac{|x_{k+1}-\xi|}{|x_k-\xi|} = \frac{|g(x_k)-g(\xi)|}{|x_k-\xi|}$$

Given g_2 Continuously differentiable.

According to mean value Thm. $\exists\, \eta_k \in [x_k, \xi]$ s.t. $g'(\eta_k) = \dfrac{g(x_k)-g(\xi)}{x_k-\xi}$

$$\Rightarrow \lim_{k\to\infty} \frac{|x_{k+1}-\xi|}{|x_k-\xi|} = \lim_{k\to\infty} |g'(\eta_k)|$$

As $x_k \to \xi$ as $k \to \xi$.
so does $\eta_k \to \xi$

$$\lim_{k\to\infty} |g'(\eta_k)| = |g'(\lim_{k\to\infty} \eta_k)| = |g'(\xi)|$$

$$= |1 - \tfrac{2}{\xi} x| = 1 - \tfrac{2}{\sqrt{5}}$$

(d) Give Newton's method for solving $f(x) = 0$ and, for given starting value $x_0 = 2$ compute the first Newton iterate x_1.

$f(x) = x^2 - 5$

$f'(x) = 2x$

$$f(x_0) + f'(x_0)(x - x_0) = 0$$

$$f'(x_0)(x - x_0) = -f(x_0)$$

$$(x - x_0) = -\frac{f(x_0)}{f'(x_0)}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 2 - \frac{-1}{4} = \frac{9}{4}$$

5. **[3pt]** Define the function g by $g(0) = 0$, $g(x) = -x\sin^2(1/x)$ for $0 < x \leq 1$. Show that g is continuous, and that 0 is the only fixed point of g in the interval $[0, 1]$. By considering the iteration $x_{n+1} = g(x_n)$, with $x_0 = 1/(k\pi)$ and $x_0 = 2/((2k+1)\pi)$, where k is an integer, show that using the definition of stability provided in class, $\xi = 0$ is neither stable nor unstable.

proof

Continuity: g is obviously continuous on $(0,1]$. It suffices to prove that

$\lim_{x \to 0} -x\sin^2(1/x) = 0$

we'll show it by ε-δ argument.

$\forall \varepsilon > 0$. $\exists\, \delta(\varepsilon) = \frac{\varepsilon}{2}$ s.t. for $x \in (-\frac{\varepsilon}{2}, \frac{\varepsilon}{2})$

$|-x\sin^2(1/x) - 0| \leq \frac{\varepsilon}{2} \cdot 1 < \frac{\varepsilon}{2}$

which shows $\lim_{x \to 0} -x\sin^2(1/x) = 0$, and continuity follows.

0 as fixed point: First of all we show that $x = 0$ is a fixed point as $g(0) = 0 = 0$

Then we show uniqueness:

If $\exists\, x \in [0,1]$ s.t. $g(x) = x$ and $x \neq 0$

$g(x) = -x\sin^2(\frac{1}{x}) = x \Rightarrow \sin^2(\frac{1}{x}) = -1$ which is impossible as $\sin^2(\frac{1}{x})$ is non-zero.

Stability: If $x_0 = \frac{1}{k\pi}$, $x_1 = g(x_0) = -x\sin^2(k\pi) = 0$ and $x_n = 0$ starting from $n \geq 2$

If $x_0 = \frac{2}{(2k+1)\pi}$ $x_1 = g(x_0) = -x_0\sin^2(k\pi + \frac{1}{2}\pi) = -x_0 = -\frac{2}{(2k+1)\pi}$ $x_2 = -x_1\sin^2(-(k\pi + \frac{1}{2}\pi))$

$\Rightarrow \begin{cases} x_{2n} = x_{2n-2} = \cdots = x_0 \\ x_{2n+1} = x_{2n-1} = \cdots = x_1 = -x_0 \end{cases}$

Here we can see, if $x_0 = \frac{1}{k\pi}$, then the iteration converges

if $x_0 = \frac{2}{(2k+1)\pi}$ the iteration doesn't converge.

From this conclusion, we can show 0 is neither stable nor unstable.

Not Unstable: $\exists$ points like $\{\frac{1}{k\pi}\}$ as initialization such that the iteration converges.
thus not unstable.

Not Stable: By definition, if it is stable $\exists$ a neighborhood of ξ such that every point in that interval as initialization makes iteration converge.
However, $\forall \xi > 0$, $\exists k(\xi)$ s.t. $0 < \frac{2}{2k(\xi)\pi+1} < \xi$ s.t.
the iteration diverges, thus being not stable.

6. **[2+2pt]** Raytracing is an algorithm that involves finding the point at which a ray (a line with a direction and an origin) intersects a curve or surface. We will consider a ray intersecting with an ellipse. The general equation for an ellipse is

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 - 1 = 0$$

and the equation for a ray starting from the point $P_0 = [x_0, y_0]$ in the direction $\mathbf{V}_0 = [u_0, v_0]$, is

$$\mathbf{R}(t) = [x_0 + tu_0, y_0 + tv_0]$$

where $t \in [0, \infty)$ parameterizes the ray. In this problem we will take $a = 3$, $b = 2$, $P_0 = [0, b]$, $\mathbf{V}_0 = [1, -0.3]$. Using your favorite root finding algorithm write a code which computes the intersection of the given ray and the ellipse and plot your results. .

(a) Plug the equation for the ray, $\mathbf{R}(t)$, into the equation for the ellipse and analytically (with pen and paper) solve for the value of t which gives the point of intersection, call it t_i.

(a)

$$\begin{cases} \left(\frac{x}{3}\right)^2 + \left(\frac{y}{2}\right)^2 - 1 = 0 \\ x = 0 + t \\ y = 2 - 0.3t \end{cases}$$

$$\frac{t^2}{9} + \frac{(2-0.3t)^2}{4} - 1 = 0$$

$$\frac{t^2}{9} + \frac{9}{400}t^2 - \frac{3}{10}t = 0$$

$$\frac{481}{3600}t^2 - \frac{3}{10}t = 0$$

$$\begin{cases} t_1 = 0 \\ t_2 = \frac{3}{10} \times \frac{3600}{481} = \frac{1080}{481} \quad (= 2.2453) \end{cases}$$

(b) Perform the same calculation numerically using your favorite root finder. Report your answer to within an error of 10^{-6} and justify how you found the minimum number of iterations required to achieve this tolerance. Also report the point of intersection $P_i = \mathbf{R}(t_i)$

For Question 6b & 7

See Attached PDF

7. **[4pt, extra credit]** If the walls of the ellipse are perfect mirrors, a ray of light will reflect infinitely around within the ellipse. We will write an algorithm to compute it's trajectory. Using the same parameters as the previous problem. Implement the following algorithm for 50 steps. At step k of the process, we are given a point on the ellipse $P_k = [x_k, y_k]$ and a ray direction $\mathbf{V}_k = [u_k, v_k]$

   (a) Using P_k and $\mathbf{V}_k$ and your favorite root finder, calculate the value of t corresponding to the point of intersection call it t_i^k.

   (b) Compute $P_{k+1} = \mathbf{R}(t_i^k)$.

Parameters Used:

$a = 3$

$b = 2$

$x_0 = 0$

$y_0 = 2$

$u_0 = 1$

$v_0 = -0.3$

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
```

Question 6 & Question 7

Since these two questions are very similar, so I implement all the code we need as a python class (**Rootfinder**), whose methods can do whatever we need.

In [2]:
```
class RootFinder:
    def __init__(self, a, b, u, v, x_0, y_0):
        self.a = a
        self.b = b

        self.u = u
        self.v = v
        self.x_0 = x_0
        self.y_0 = y_0


    def ray(self, t):
        """
        parameters: x_0, y_0, u_0, v_0 defualt (0, 2, 1, -0.3) as is given in the problem
        Input: t
        Output: (x, y) coordinates of the ray
        """


        return self.x_0 + t*self.u, self.y_0 + t*self.v

    def ellipse(self, x, y):
        """
        Input: x, y
        Output: the ellipse function = 0
        """

        return (x/self.a)**2 + (y/self.b)**2 - 1


    def f(self, t):
        """
        Goal: Composite the ellipse and the ray.
        Input: t
        Output: (actualy the error term), which is the objective function

        """
        x, y = self.ray(t)
        return self.ellipse(x, y)


    def fprime(self, t):
        """
        The derivative of the objective function, depending on a, b, u, v
        Input: t
        Output: f'(t)
        Using chain rule
        """
        x, y = self.ray(t)
        return self.u*(2*x)/(self.a**2) + self.v*(2*y)/(self.b**2)

    def newton_iteration(self, t_0, max_iteration=100):
        t = t_0
        t_history = [t]
        error_history = [abs(self.f(t))]
        for i in range(max_iteration):
            t = t - self.f(t)/self.fprime(t)
            error = abs(self.f(t))
            t_history.append(t)
            error_history.append(error)
            if error < 1e-6:
                break

        return t_history, error_history


    def solve(self, t_0):
        flag = True
        while flag:
            t_history, error_history = self.newton_iteration(t_0)
            if t_history[-1]>1e-4: # Make sure the sol is not trivial
                flag = False
            else:
                t_0 = random.choice(np.linspace(0, self.a, 100)) # Randomly choose a value to start


        return t_history[-1], t_history, error_history

    def intersection(self):
        # Compute new x and y

        new_x, new_y = self.ray(self.solve(t_0=1)[0])


        # calculate new u nad v

        normal_vector = np.array([(self.b * new_x)/self.a, (self.a * new_y)/self.b])
        n = normal_vector/np.linalg.norm(normal_vector)

        w = np.array([self.u, self.v])
        w = w/np.linalg.norm(w)

        new_u, new_v = w - 2 * np.dot(w, n)*n

        # Update the argument
        self.x_0 = new_x
        self.y_0 = new_y
        self.u = new_u
        self.v = new_v


        return np.array([new_x, new_y])
```

Question 6

Use **Rootfinder().solve()** (which calls **newton_iteration** internally), we are able to get the solution.

Note that, by excuding the solution $t = 0$, we are able to get the nontrivial solution.

Plugging back the value of t we get, we may get the coordinates of the intersection.

In [3]:
```
model_1 = RootFinder(a=3,b=2, u=1, v=-0.3, x_0=0, y_0=2)
t, _, _ = model_1.solve(t_0=4) # By setting the initial value big, we can avoid it converges to the root t=0
num_iter = len(_) - 1
x, y = model_1.ray(t)
print("The coordinates of the intersection we found is ({:.4f}, {:.4f}).".format(x, y))
```

```
The coordinates of the intersection we found is (2.2453, 1.3264).
```

Justification of # of iteration to converge

As we are using *Newtown's Method*, we have quadratic order for convergence. Namely, it is doubleing its zeros at every iterations.

As

$$error_0 = |f(t_0)| = |f(4)| = 0.9378$$

which is approximatly between 10^0 and 10^{-1}, and the iteration number n has the relation $1 \cdot 2^n = 6$, where 6 is given by the precision 10^{-6}. Therefore, we have

$$n = \log_2 6 = 2.58 \approx 3$$

Note that the actual tolerence will be somewhat bigger than 3, as the calculation of iterations only works when the initial value is very close to the root desired.

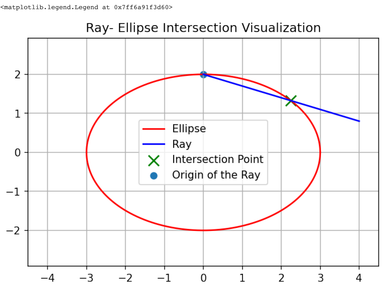
In [4]:
```
print("The actual # of iterations is ", num_iter)
```

```
The actual # of iterations is  5
```

Visualization

In [5]:
```
# Visualization

def ellipse_vis(x, a=3, b=2):
    """
    return half of the ellipse
    """
    return b*(1-x**2/a**2)**(1/2)

plt.rcParams['figure.dpi'] = 150

xspace = np.linspace(-3, 3, 10000)
plt.plot(xspace, ellipse_vis(xspace), c='{1,0,0}, label="Ellipse")
plt.plot(xspace, -ellipse_vis(xspace), c='{1,0,0}')


tspace = np.linspace(0, 4, 200)

ray_x = model_1.x_0 + model_1.u*tspace
ray_y = model_1.y_0 + model_1.v*tspace


plt.plot(ray_x, ray_y, c='{0,0,1}, label="Ray", linewidth=1.5)

plt.scatter([x], [y], label="Intersection Point", marker="x", s=100, c="g")
plt.scatter([0], [2], label="Origin of the Ray")



plt.title("Ray- Ellipse Intersection Visualization")

plt.axis("equal")
plt.xlim(-4.5, 4.5)
plt.grid()

plt.legend()
```

Out[5]:
```
<matplotlib.legend.Legend at 0x7ff6a91f3d60>
```



Question 7

As we have already implemented the **intersection** method in the class **RootFinder**, we can directly output the result.

The parameters we use is given in Question 6, which are

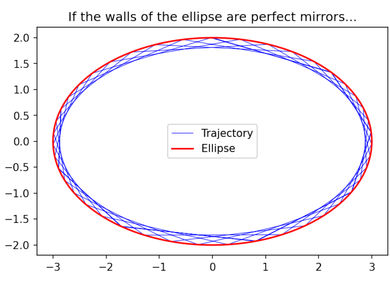
$$a = 3, b = 2$$
$$u = 1, v = -0.3$$
$$x_0 = 0, y_0 = 2$$

In [6]:
```
model_2 = RootFinder(a=3,b=2, u=1, v=-0.3, x_0=0, y_0=2)
output = np.zeros((51, 2))
output[0] = np.array([0, 2])
for i in range(1, 51):
    output[i] =  model_2.intersection()

output = output.T
```

Visualization

In [7]:
```
# Visualization

def ellipse_vis(x, a=3, b=2):
    """
    return half of the ellipse
    """
    return b*(1-x**2/a**2)**(1/2)


xspace = np.linspace(-3, 3, 10000)


plt.rcParams['figure.dpi'] = 150

plt.plot(output[0], output[1], c='{0,0,1}, label="Trajectory", linewidth=0.5)

plt.plot(xspace, ellipse_vis(xspace), c='{1,0,0}, label="Ellipse")
plt.plot(xspace, -ellipse_vis(xspace), c='{1,0,0}')
plt.title("If the walls of the ellipse are perfect mirrors...")
plt.legend()
```

Out[7]:
```
<matplotlib.legend.Legend at 0x7ff6a999fc70>
```



50^{th} intersection...

In [8]:
```
print("The coordinates of the 50th intersection is ({:.4f}, {:.4f}).".format(output[0][-1], output[1][-1]))
```

```
The coordinates of the 50th intersection is (-2.8530, 0.6185).
```

In [ ]: