Math 6330 Homework 6

Jacob Hauck and Rajrani Gupta

March 12, 2024

3.2

Let $a \in \mathbf{R}$, and consider the difference equation

$$x_{n+1} = f(x_n) = \frac{2}{3}x_n + \frac{a}{3x_n^2}.$$

Then f has only one fixed point equal to $\sqrt[3]{a}$ because

$$f(x) = x \implies x = \frac{2}{3}x + \frac{a}{3x_n^2} \implies x^3 = a \implies x = \sqrt[3]{a}.$$

Furthermore, $\sqrt[3]{a}$ is an asymptotically stable fixed point because

$$f'(\sqrt[3]{a}) = \frac{2}{3} - \frac{2a}{3a} = 0.$$

Using the script in Listing 6 (see appendix), we see that it takes about 4 iterations starting from an initial guess of 1.5 for the first 5 digits to settle down – see the console output in Listing 1.

Listing 1: Console command and output

```
> python -m cbrt 2 --guess 1.5
n = 0    x_n = 1.5

n = 1    x_n = 1.2962962962962963

n = 2    x_n = 1.2609322247417485

n = 3    x_n = 1.2599218605659261

n = 4    x_n = 1.2599210498953948
```

Euler method for $\sqrt{2}$ with a large step size

We would like to consider the difference equation

$$x_{n+1} = x_n - h\frac{x_n^2 - 2}{2x_n},$$

which is Euler's method for approximating $\dot{x} = -\frac{x^2-2}{2x}$ with step size $h > 0$. For the right step size, this method should converge to $\sqrt{2}$. We have already seen that this works with $h = 1$. Using the script in Listing 7 (see appendix) we see that the method also appears to converge with $h = 1.5$ and $h = 1.9$ but fails to converge with $h = 2.5$ and $h = 2.1$ – see Listings 2, 3, 4, 5 for the corresponding console output.

Listing 2: Console command and output, $h = 1.5$ (abridged)

```
>python -m euler --start 1.6 --step 1.5
x_0 = 1.6

x_1 = 1.3375

x_2 = 1.4558703271028037

x_3 = 1.3942791226290783

...

x_11 = 1.414134600476956

x_12 = 1.4142530466279473

x_13 = 1.4141938210724339
```

Listing 3: Console command and output, $h = 1.9$ (abridged)

```
>python -m euler --start 1.6 --step 1.9
x_0 = 1.6

x_1 = 1.2674999999999998

x_2 = 1.5623888067061145

x_3 = 1.2942059815629339

...

x_97 = 1.4142072829470977

x_98 = 1.4142192138829808

x_99 = 1.4142084760356535
```

Listing 4: Console command and output, $h = 2.1$ (abridged)

```
>python -m euler --start 1.6 --step 2.1
x_0 = 1.6

x_1 = 1.2324999999999997

x_2 = 1.6422289553752538

x_3 = 1.1966383837491845

x_4 = 1.6950842096270753

...

x_18 = 3.0776930236495614

x_19 = 0.5284446077342535

x_20 = 3.9475042119776256

x_21 = 0.33460648903505996
```

Listing 5: Console command and output, $h = 2.5$ (abridged)

```
>python -m euler --start 1.6 --step 2.5
x_0 = 1.6

x_1 = 1.1624999999999996

x_2 = 1.8599126344086028

x_3 = 0.8791709985946473

x_4 = 2.623795134842801

...

x_66 = 0.08931453285309976

x_67 = 27.968636780129696

x_68 = -6.902773358330645

x_69 = 1.363520069474804
```

3.4 (a)

Consider the parametric map

$$f(\lambda, x) = \lambda x(1 - x)$$

for $\lambda > 1$. This map has fixed points when

$$x = \lambda x(1 - x) \implies x(\lambda x + 1 - \lambda) = 0,$$

so when $x = x_1 = 0$ or $x = x_2 = 1 - \frac{1}{\lambda}$. To find when these equilibrium points are non-hyperbolic, we need to find $\lambda > 1$ such that $|f_x(\lambda, x_i)| = 1$, where $i = 1, 2$.

Since $f_x(\lambda, x) = \lambda(1 - 2x)$, we see that x_1 is non-hyperbolic if $|\lambda| = 1$, which is impossible under the constraint $\lambda > 1$, so x_1 is always hyperbolic.

On the other hand, x_2 is non-hyperbolic if

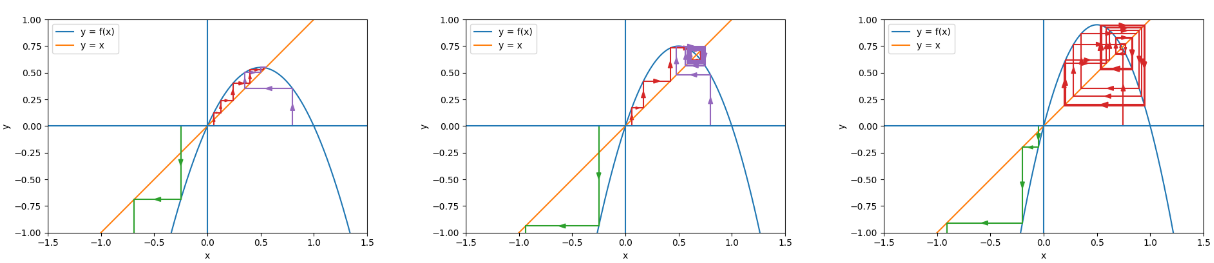
$$|\lambda(1 - 2x_2)| = \left|\lambda\left(1 - 2 + \frac{2}{\lambda}\right)\right| = 1 \iff |2 - \lambda| = 1,$$

that is, when $\lambda = 1$ or when $\lambda = 3$. Since we are considering $\lambda > 1$, we see that x_2 is non-hyperbolic only when $\lambda = 3$.

To determine the stability of x_1 and x_2, we observe that $|f_x(\lambda, x_1)| = |\lambda| > 1$, so x_1 is always unstable. For x_2, observe that $|f_x(\lambda, x_2)| = |2 - \lambda|$. Since $|2 - \lambda| < 1$ is equivalent to $1 < \lambda < 3$, we see that x_2 is asymptotically stable if $\lambda < 3$. Since $|2 - \lambda| > 1$ is equivalent to $\lambda < 1$ or $\lambda > 3$, we see that x_2 is unstable if $\lambda > 3$.

To investigate the stability of x_2 when $\lambda = 3$, we turn to the stair-step diagrams in Figure 1. Apparently, x_2 is asymptotically stable when $\lambda = 3$ (see Figure 1b).

Diagrams were generated using the code in Listing 8.



(a) $\lambda < 3$ (b) $\lambda = 3$ (c) $\lambda > 3$

Figure 1: 3.4 (a) typical stair-step diagrams

3.4(c)

Consider the parametric map
$$f(\lambda, x) = \lambda - x^2.$$
This map has fixed points when
$$\lambda - x^2 = x \iff x^2 + x - \lambda = 0,$$
so when
$$x = x_1 = \frac{-1 + \sqrt{1+4\lambda}}{2} \qquad \text{or} \qquad x = x_2 = \frac{-1 - \sqrt{1+4\lambda}}{2}.$$
This can only occur if $\lambda \geq -\frac{1}{4}$; otherwise, there are no fixed points. In the special case that $\lambda = -\frac{1}{4}$, we see that $x_1 = x_2 = -\frac{1}{2}$ and there is actually only one fixed point.

Noting that $f_x(\lambda, x) = -2x$, we If $\lambda = -\frac{1}{4}$, then the fixed point $-\frac{1}{2}$ is non-hyperbolic because
$$\left| f_x\left(-\frac{1}{4}, -\frac{1}{2}\right)\right| = 1.$$

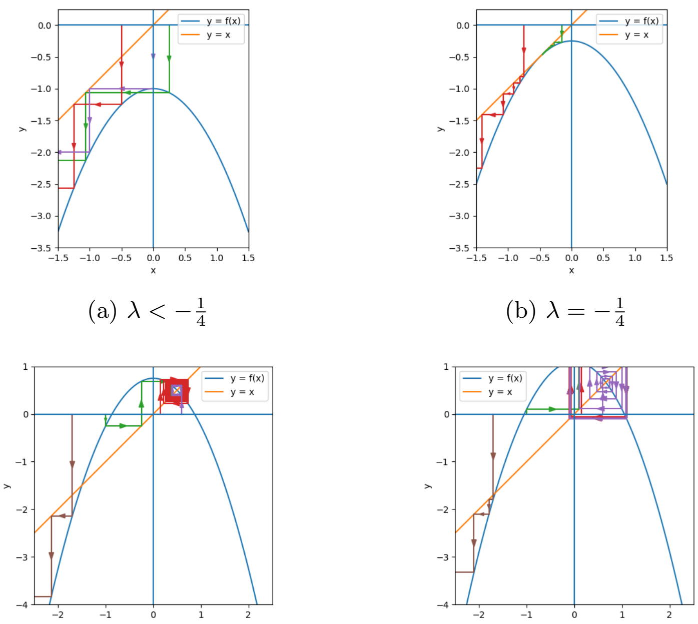
If $\lambda > -\frac{1}{4}$, then we can determine the values of λ such that the two fixed points x_1 and x_2 are non-hyperbolic by setting $|f_x(\lambda, x_i)| = 1$ for $i = 1, 2$. Starting with x_2, we see that
$$|f_x(\lambda, x_2)| = |-2x_2| = 1 + \sqrt{1+4\lambda} > 1,$$
so x_2 is always hyperbolic. For x_1, we see that
$$|f_x(\lambda, x_1)| = |-2x_1| = \left|1 - \sqrt{1+4\lambda}\right| = 1 \iff \sqrt{1+4\lambda} = 0 \quad \text{or} \quad \sqrt{1+4\lambda} = 2.$$
The former is impossible because we are considering the case that $\lambda > -\frac{1}{4}$, and $\sqrt{1+4\lambda} = 2$ if and only if $1 + 4\lambda = 4$, that is, $\lambda = \frac{3}{4}$. Thus, x_2 is non-hyperbolic only when $\lambda = \frac{3}{4}$.

To determine the stability of the fixed points, we first consider x_2 when $\lambda > -\frac{1}{4}$, as this fixed point is always hyperbolic. We see that
$$|f_x(\lambda, x_2)| = 1 + \sqrt{1+4\lambda} > 1$$
regardless of the value of $\lambda > -\frac{1}{4}$, so x_2 is always unstable.

Next, we consider x_1 when $\lambda > -\frac{1}{4}$ and $\lambda \neq \frac{3}{4}$. We have
$$|f_x(\lambda, x_1)| = \left|1 - \sqrt{1+4\lambda}\right| < 1 \iff 0 < \sqrt{1+4\lambda} < 2.$$
The last inequalities are equivalent to $-\frac{1}{4} < \lambda < \frac{3}{4}$, so x_1 is stable when $\lambda < \frac{3}{4}$. If $\lambda > \frac{3}{4}$, then certainly $|f_x(\lambda, x_1)| > 1$, so x_1 is unstable when $\lambda > \frac{3}{4}$.

To investigate the stability of the hyperbolic fixed points $-\frac{1}{2}$ when $\lambda = -\frac{1}{4}$ and x_2 when $\lambda = \frac{3}{4}$, we turn to the stair-step diagrams in Figure 2. We see that $-\frac{1}{2}$ when $\lambda = -\frac{1}{4}$ appears to be unstable (see Figure 2b), and x_2 when $\lambda = \frac{3}{4}$ appears to be asymptotically stable (see Figure 2d).

Diagrams were generated using the code in Listing 8.



(a) $\lambda < -\frac{1}{4}$

(b) $\lambda = -\frac{1}{4}$

(c) $-\frac{1}{4} < \lambda < \frac{3}{4}$

(d) $\lambda = \frac{3}{4}$

(e) $\lambda > \frac{3}{4}$

Figure 2: 3.4 (c) typical stair-step diagrams

3.4(d)

Consider the parametric map

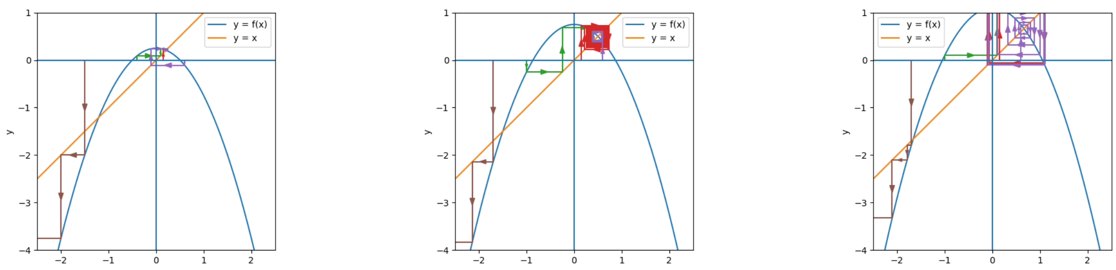
$$f(\lambda, x) = \lambda^2 - x^2.$$

This map is effectively the same as the one in 3.4(c) but reparametrized with λ^2 in place of λ. Since $\lambda^2 > -\frac{1}{4}$ for any value of λ, we can recycle the calculations from 3.4(c) to find that the fixed points of f are

$$x_1 = \frac{-1 + \sqrt{1 + 4\lambda^2}}{2} \qquad \text{and} \qquad x_2 = \frac{-1 - \sqrt{1 + 4\lambda^2}}{2}.$$

Further recycling calculations from 3.4 (c), the fixed point x_1 is always hyperbolic and unstable, and the fixed point x_2 is non-hyperbolic only when $\lambda^2 = \frac{3}{4}$, that is, when $\lambda = \pm\frac{\sqrt{3}}{2}$. Additionally, x_2 is asymptotically stable if $\lambda^2 < \frac{3}{4}$, that is, if $|\lambda| < \frac{\sqrt{3}}{2}$, and x_2 is unstable if $\lambda^2 > \frac{3}{4}$, that is, if $|\lambda| > \frac{\sqrt{3}}{2}$. Finally, x_2 is also asymptotically stable if $\lambda^2 = \frac{3}{4}$, that is, if $\lambda = \pm\frac{\sqrt{3}}{2}$.

Stair-step diagrams for these different cases are given in Figure 3. Diagrams were generated using the code in Listing 8.



(a) $|\lambda| < \frac{\sqrt{3}}{2}$ (b) $|\lambda| = \frac{\sqrt{3}}{2}$ (c) $|\lambda| > \frac{\sqrt{3}}{2}$

Figure 3: 3.4 (d) typical stair-step diagrams

3.6

Consider the map

$$x_{n+1} = f(x_n) = bx_n\left(\frac{1+b}{b} - x_n\right).$$

This map has fixed points when

$$x = f(x) = bx\left(\frac{1+b}{b} - x\right) \iff bx(1-x) = 0,$$

so when $x = 0$ or $x = 1$. Note that

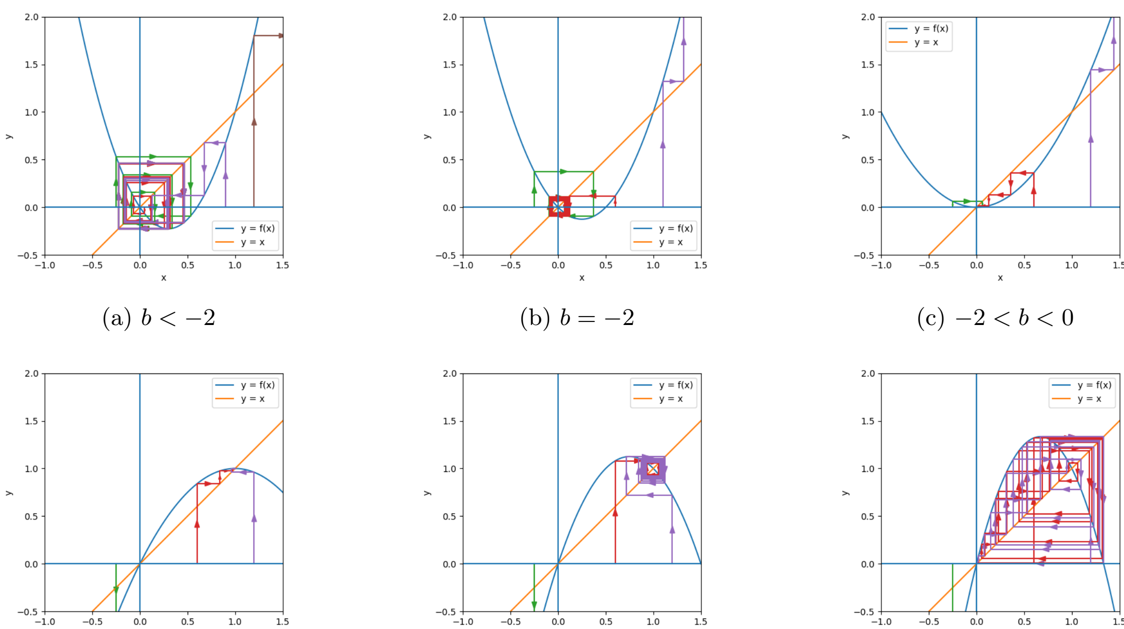
$$f'(x) = 1 + b - bx - bx = 1 + b - 2bx$$

Then 0 is asymptotically stable when $|1+b| < 1$, that is, when $-2 < b < 0$, and 0 is unstable when $|1+b| > 1$, that is, when $b < -2$ or $b > 0$.

Furthermore, 1 is asymptotically stable when $|1-b| < 1$, that is, when $0 < b < 2$, and 1 is unstable when $|1-b| > 1$, that is, when $b < 0$ or $b > 2$.

When $b = -2$ or 2, at least one of the fixed points is non-hyperbolic, and we turn to the stair-step diagrams in Figure 4 to determine stability (note that we don't mind about $b = 0$ because the map requires us to divide by b, so we must assume $b \neq 0$). From the stair-step diagrams, we see that 0 is asymptotically stable when $b = -2$ (see Figure 4b), and 1 is asymptotically stable when $b = 2$ (see Figure 4e).

Diagrams were generated using the code in Listing 8.



(a) $b < -2$ (b) $b = -2$ (c) $-2 < b < 0$

(d) $0 < b < 2$ (e) $b = 2$ (f) $b > 2$

Figure 4: 3.6 typical stair-step diagrams

Appendix

Listing 6: `cbrt.py` – cube root difference equation simulation

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument(type=float, dest='number', metavar='X')
parser.add_argument('--guess', required=True, type=float)

args = parser.parse_args()

n = 0
x = args.guess
while True:
    print(f'n = {n}    x_n = {x}')

    if input() == 'q':
        break

    x = (2/3 * x) + args.number/(3 * x**2)
    n += 1
```

Listing 7: `euler.py` – Euler method for $\sqrt{2}$

```
"""
Step-by-step Euler's method for x' = -f(x) / f'(x), f(x) = x^2 - 2
"""
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('--start', type=float)
parser.add_argument('--step', type=float)

args = parser.parse_args()

n = 0
x_n = args.start
h = args.step

while True:
    print(f'x_{n} = {x_n}')

    if input() == 'q':
        break

    x_n = x_n - h * (x_n**2 - 2) / (2*x_n)
    n += 1
```

Listing 8: `stairstep.py` – Stair-step diagram generator

```
"""
Code for generating stairstep diagrams
"""
import matplotlib.pyplot as plt
import numpy as np


def stairstep(f, start, x_lim, y_lim):
    fig, ax = plt.subplots()
    x = np.linspace(*x_lim, 500)

    ax.plot(x, f(x), label='y = f(x)')
    ax.plot(x, x, label='y = x')
    ax.set_xlabel('x')
    ax.set_ylabel('y')

    hw = .02 * min(x_lim[1] - x_lim[0], y_lim[1] - y_lim[0])
    hl = 1.5 * hw
    for x0 in start:
        pts = [(x0, 0)]
        for step in range(20):
            f_x0 = f(x0)
            if abs(f_x0) > 1e12:
                break
            pts.append((x0, f_x0))
            pts.append((f_x0, f_x0))
            x0 = f_x0
        pts = np.array(pts)
        midpoints = (pts[1:] + pts[:-1]) / 2
        deltas = pts[1:] - pts[:-1]

        p = ax.plot(pts[:, 0], pts[:, 1])
        for mp, delta in zip(midpoints, deltas):
            d = min(np.max(np.abs(delta)) * .3, hl)
            delta = np.sign(delta) * d
            ax.arrow(mp[0] - delta[0]/2, mp[1] - delta[1]/2, delta[0], delta[1],
                     head_width=d/1.5,
                     width=0,
                     color=p[0].get_color(),
                     length_includes_head=True)

    ax.set_xlim(*x_lim)
    ax.set_ylim(*y_lim)
    ax.set_aspect('equal')
    ax.hlines([0], *x_lim)
    ax.vlines([0], *y_lim)

    ax.legend()

    return fig, ax


if __name__ == '__main__':
    # =========== 3.4 (a) ===========
    '''
```

```
    # Case: 1 < lambda < 3
    stairstep(lambda x: 2.2*x*(1 - x), [-.25, .06, .8], (-1.5, 1.5), (-1, 1))
    plt.savefig('3.4a lambda lt 3.png')
    plt.show()

    # Case: lambda = 3
    stairstep(lambda x: 3*x*(1 - x), [-.25, .06, .8], (-1.5, 1.5), (-1, 1))
    plt.savefig('3.4a lambda eq 3.png')
    plt.show()

    # Case: lambda > 3
    stairstep(
        lambda x: 3.8 * x * (1 - x), [-.05, 1 - 1 / 3.8 + .01],
        (-1.5, 1.5), (-1, 1)
    )
    plt.savefig('3.4a lambda gt 3.png')
    plt.show()

    # ============ 3.4 (c) ============

    # Case: lambda < -1/4
    stairstep(lambda x: -1 - x ** 2, [.25, -.5, 0], (-1.5, 1.5), (-3.5, .25))
    plt.savefig('3.4c lambda lt -14.png')
    plt.show()

    # Case: lambda = -1/4
    stairstep(lambda x: -1 / 4 - x ** 2, [-.15, -.75, ], (-1.5, 1.5), (-3.5, .25))
    plt.savefig('3.4c lambda eq -14.png')
    plt.show()

    # Case: -1/4 < lambda < 3/4
    stairstep(lambda x: 1/4 - x**2, [-.4, .15, .6, -1.5], (-2.5, 2.5), (-4, 1))
    plt.savefig('3.4c lambda gt -14.png')
    plt.show()

    # Case: lambda = 3/4
    stairstep(lambda x: 3/4 - x**2, [-1, .15, .6, -1.7], (-2.5, 2.5), (-4, 1))
    plt.savefig('3.4c lambda eq 34.png')
    plt.show()

    # Case: lambda > 3/4
    stairstep(lambda x: 1.1 - x**2, [-1, .15, .6, -1.7], (-2.5, 2.5), (-4, 1))
    plt.savefig('3.4c lambda gt 34.png')
    plt.show()

    # ============ 3.4 (d) ============

    # Case: |lambda| < sqrt(3) / 2
    stairstep(
        lambda x: (1/2)**2 - x ** 2, [-.4, .15, .6, -1.5],
        (-2.5, 2.5), (-4, 1)
    )
    plt.savefig('3.4d lambda lt sqrt(3)2.png')
    plt.show()

    # Case: |lambda| = sqrt(3) / 2
```

```
    stairstep(lambda x: 3/4 - x ** 2, [-1, .15, .6, -1.7], (-2.5, 2.5), (-4, 1))
    plt.savefig('3.4d lambda eq sqrt(3)2.png')
    plt.show()

    # Case: |lambda| > sqrt(3) / 2
    stairstep(lambda x: 1.1 - x ** 2, [-1, .15, .6, -1.7], (-2.5, 2.5), (-4, 1))
    plt.savefig('3.4d lambda gt sqrt(3)2.png')
    plt.show()
    '''
    # ============ 3.6 ============

    # Case: b < -2
    stairstep(
        lambda x: -2.5*x*(1.5/2.5 - x), [-.25, .05, .9, 1.2],
        (-1, 1.5), (-.5, 2)
    )
    plt.savefig('3.6 b lt -2.png')
    plt.show()

    # Case: b = -2
    stairstep(lambda x: -2*x*(1/2 - x), [-.25, .6, 1.1], (-1, 1.5), (-.5, 2))
    plt.savefig('3.6 b eq -2.png')
    plt.show()

    # Case: -2 < b < 0
    stairstep(lambda x: -1*x*(0 - x), [-.25, .6, 1.2], (-1, 1.5), (-.5, 2))
    plt.savefig('3.6 b gt -2.png')
    plt.show()

    # Case 0 < b < 2
    stairstep(lambda x: 1*x*(2 - x), [-.25, .6, 1.2], (-1, 1.5), (-.5, 2))
    plt.savefig('3.6 b gt 0.png')
    plt.show()

    # Case b = 2
    stairstep(lambda x: 2*x*(3/2 - x), [-.25, .6, 1.2], (-1, 1.5), (-.5, 2))
    plt.savefig('3.6 b eq 2.png')
    plt.show()

    # Case b > 2
    stairstep(lambda x: 3*x*(4/3 - x), [-.25, .6, 1.2], (-1, 1.5), (-.5, 2))
    plt.savefig('3.6 b gt 2.png')
    plt.show()
```