Exam2

October 7, 2021

1 Exam 1

1.1 Decoding the secret message

Note: Make sure you have the message file in the same directory as this program!

[67]:
```
import numpy as np
import math

def Decode_msg(string, initial_point, parameter):
    # we need to reconstruct the same set of chaos points,
    # so this part of the code is literally just copied from the encoder
    # the only difference is that we generate the same amount of points as in␣
 ↪the
    # input string, instead of multiplying that amount by 3

    def log_func(x,r):
        return r*x*(1-x)

    R = 3.9+parameter/100000

    r0=R

    X_0=initial_point/100000
    # chaos points should be the same length as the encoded message
    Steps=len(string)

    X_old=X_0

    Points=[]

    start_sequence = np.random.randint(Steps//3)

    for t in range(0,Steps):
        Points.append(X_old)
        X_new=log_func(X_old,r0)
        X_old=X_new
```

```
    # here are the reconstructed chaos points
    Points=np.array(Points)

    # next, let's convert the encoded message back into the right form:
    # a list of numbers from zero to 1
    encoded_msg = np.array([ord(char) for char in string])/100000

    # let's get our chaos points into the right format to subtract them
    # the first two decimal places need to be zeroed out
    first_two = np.array([np.trunc(num*100)/100 for num in Points])

    Points -= first_two

    # all decimals after the 5th place need to be removed
    Points = np.trunc(Points*100000)/100000

    # now let's subtract the two lists
    encoded = encoded_msg - Points

    # remove the zeroes
    encoded = list(filter(lambda num: num != 0, encoded))

    # change back to integer values
    encoded = [int(np.round(num*100000)) for num in encoded]

    decoded_str = ''

    for num in encoded:
        try:
            decoded_str += chr(num)
        except:
            # For some reason I still don't understand, some of the characters␣
 ↪got corrupted.
            # Since most of the message decrypted fine, I realized that they had
            # just been offset by some constant amount (turned out to be -1000).
            # Though I was able to fix things, I still don't know why this␣
 ↪happens.
            # It's not like specific characters got corrupted, either.
            # Maybe my program is at fault, or there's a bug in the encryption␣
 ↪process.
            fix_offset = chr(num+1000)
            decoded_str += fix_offset

    return decoded_str

# I was not told which was the initial point and which was the parameter, but␣
 ↪guess & check is easy in this case
```

```
key1, key2 = 3567, 2673
code_string = open("Secret_msg","r").read()
print(Decode_msg(code_string, key1, key2))
```

```
When you are old and grey and full of sleep,
And nodding by the fire, take down this book,
And slowly read, and dream of the soft look
Your eyes had once, and of their shadows deep;

How many loved your moments of glad grace,
And loved your beauty with love false or true,
But one man loved the pilgrim soul in you,
And loved the sorrows of your changing face;

And bending down beside the glowing bars,
Murmur, a little sadly, how Love fled
And paced upon the mountains overhead
And hid his face amid a crowd of stars.
```