MATH381 Assignment 5

1 Introduction

Consider the following game:

To start with a score of zero and set a goal score of M. On each turn, we roll a fair, six-sided die. The roll is our new score. We continue to roll the die. With each roll, we add the roll to the score.

If the sum is M or greater the game ends. If not, and the sum is a prime number or twice a prime number, we divide our sum by 2 (rounded down) to get a new score. Otherwise, our new score is the sum. We only do this division bit starting with the second roll. We continue rolling and changing our score until we reach a score of M or more.

To investigate this game using Markov chain, which models the change of states over time in a probabilistic manner, we start with the situation that $M = 11$ for the expected number of turns to finish the game and the distribution of the number of rolls until the game ends. Then, we investigate how the expected number of turns in the game changes with the value of M.

2 Dice Game with M = 11

In this game, we kept rolling until we won with a score larger or equal to 11. To model the game with a Markov chain, we define a chain with $M+1$ states: $0, 1, 2, ..., 10, W$. The states $0, 1, 2, ..., 10$ indicate having that score during the game: if we have a score of 3, we will say that we are in the state 3. These states are not absorbing, since we will move on to another state to continue the game. The state W indicates having a score that is equal to or larger than 11, meaning that we have won the game. Since we stop playing the game once we won, and we stay at state W forever once we reach this state, the state W is an absorbing state.

In order to know the probability of transitioning from one state to another, we construct a transition matrix A using Sage code (All codes used in this project are included in Appendix), with each entry a_{ij} of A giving the probability of transitioning from state i to state j in one step.

Below is the transition matrix A generated by Sage code:

$$A = \begin{pmatrix} 0 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/6 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 1/6 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 1/6 & 1/6 & 0 & 0 \\ 0 & 0 & 1/6 & 1/3 & 0 & 1/6 & 0 & 0 & 1/6 & 1/6 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 0 & 1/6 & 0 & 0 & 1/6 & 1/6 & 0 & 1/6 \\ 0 & 0 & 0 & 1/6 & 0 & 1/6 & 0 & 0 & 1/6 & 1/6 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 1/6 & 0 & 0 & 1/6 & 1/6 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1/6 & 0 & 0 & 0 & 1/6 & 0 & 2/3 \\ 0 & 0 & 0 & 0 & 0 & 1/6 & 0 & 0 & 0 & 0 & 0 & 5/6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Notice that in the transition matrix A, columns of states $0, 7, 10$ are all zeros. State 0 is the starting state, and we don't have any subtraction rules in the game to go back to this state. State 7 is a prime number and state 10 is twice a prime number. Since the twice of these two states are 14 and 20, both of which are larger than $W = 11$, we will never enter these states or come back to them from larger states.

Given the transition matrix A, we can put it into the canonical form for further analysis:

$$A = \left(\begin{array}{c|c} Q & R \\ \hline O & J \end{array}\right)$$

where J is an identity matrix (with 1's on the diagonal and 0's elsewhere) and O is a matrix of all zeros. Q and R are non-negative matrices that arise from the transition probabilities between non-absorbing states.

The ij entry of Q^k is the probability of transferring from state i to state j in k steps. To find the expected number of times that the chain is in state j after starting at state i before reaching the absorbing state W (winning the game), we add the ij entry of Q^k together, with k ranges from 1 to infinity. We define the result matrix of the sum $I + Q + Q^2 + Q^3 + ...$ as matrix N. Thus, N_{ij} represents the expected number of times that the chain is in state j after starting at state i before reaching the absorbing state W. As we star

By using Sage code, we generate matrix N below:

$$N = \begin{pmatrix} 1 & 4997/8304 & 2921/2076 & 4509/2768 & 1/6 & 4145/8304 & 1/6 & 0 & 2677/4152 & 1433/2768 & 0 \\ 0 & 351/173 & 366/173 & 381/173 & 0 & 45/173 & 0 & 0 & 132/173 & 93/173 & 0 \\ 0 & 963/1384 & 963/346 & 2589/1384 & 0 & 363/1384 & 0 & 0 & 567/692 & 681/1384 & 0 \\ 0 & 501/1384 & 501/346 & 3507/1384 & 0 & 357/1384 & 0 & 0 & 489/692 & 807/1384 & 0 \\ 0 & 753/2768 & 753/6923887/2768 & 1 & 1241/2768 & 0 & 0 & 909/1384 & 1619/2768 & 0 & \\ 0 & 27/173 & 108/173 & 189/173 & 0 & 243/173 & 0 & 0 & 90/173 & 87/173 & 0 \\ 0 & 265/2768 & 265/692 & 1855/2768 & 0 & 1001/2768 & 1 & 0 & 557/1384 & 1123/2768 & 0 \\ 0 & 49/1384 & 49/346 & 343/1384 & 0 & 441/1384 & 0 & 1 & 197/692 & 427/1384 & 0 \\ 0 & 21/692 & 21/173 & 147/692 & 0 & 189/692 & 0 & 0 & 381/346 & 183/692 & 0 \\ 0 & 9/346 & 18/173 & 63/346 & 0 & 81/346 & 0 & 0 & 15/173 & 375/346 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

To sum the first row of matrix N, we get the expected number of turns until absorption:

$$\frac{27539}{4152} \approx 6.632707129$$

To investigate the distribution of the number of rolls until the game ends, we calculate the probability of reaching state W at or within i rolls using transition matrix A, by calculating the value of $A^i_{1,M+1}$. We also calculate the probability of reaching state W on the ith roll by calculating $A^i_{1,M+1} - A^{i-1}_{1,M+1}$ while $i > 1$. Since we know that, as the number of rolls increases, the probability of winning at or within i rolls increases, we keep calculating probabilities until the probability approaches 1:

i	Probability of winning at/within i rolls)	Probability of winning on ith roll
1	$A^1_{1,12} = 0$	$p_1 = 0$
2	$A^2_{1,12} = 0.08333333333$	$p_2 = 0.08333333333$
3	$A^3_{1,12} = 0.2824074074$	$p_3 = 0.1990740741$
4	$A^4_{1,12} = 0.4297839506$	$p_4 = 0.1473765432$
5	$A^5_{1,12} = 0.5410236626$	$p_5 = 0.111239712$
6	$A^6_{1,12} = 0.6290509259$	$p_6 = 0.0880272633$
7	$A^7_{1,12} = 0.6994312986$	$p_7 = 0.0703803727$
8	$A^8_{1,12} = 0.7561567644$	$p_8 = 0.0567254658$
9	$A^9_{1,12} = 0.802056343$	$p_9 = 0.0458995786$
10	$A^{10}_{1,12} = 0.8392672293$	$p_{10} = 0.0372108863$
11	$A^{11}_{1,12} = 0.8694633095$	$p_{11} = 0.0301960802$
12	$A^{12}_{1,12} = 0.8939786908$	$p_{12} = 0.0245153813$
13	$A^{13}_{1,12} = 0.9138867869$	$p_{13} = 0.0199080961$
14	$A^{14}_{1,12} = 0.9300553679$	$p_{14} = 0.016168581$
15	$A^{15}_{1,12} = 0.9431876263$	$p_{15} = 0.0131322584$
16	$A^{16}_{1,12} = 0.9538540664$	$p_{16} = 0.0106664401$
17	$A^{17}_{1,12} = 0.9625178142$	$p_{17} = 0.0086637478$
18	$A^{18}_{1,12} = 0.969554938$	$p_{18} = 0.0070371238$
19	$A^{19}_{1,12} = 0.9752708577$	$p_{19} = 0.0057159197$
20	$A^{20}_{1,12} = 0.9799136344$	$p_{20} = 0.0046427767$
21	$A^{21}_{1,12} = 0.98368475$	$p_{21} = 0.0037711156$
22	$A^{22}_{1,12} = 0.9867478565$	$p_{22} = 0.0030631065$
23	$A^{23}_{1,12} = 0.9892358797$	$p_{23} = 0.0024880232$
24	$A^{24}_{1,12} = 0.991256789$	$p_{24} = 0.0020209093$
25	$A^{25}_{1,12} = 0.9928982826$	$p_{25} = 0.0016414936$
26	$A^{26}_{1,12} = 0.9942315941$	$p_{26} = 0.0013333115$
27	$A^{27}_{1,12} = 0.9953145832$	$p_{27} = 0.0010829891$
28	$A^{28}_{1,12} = 0.9961942465$	$p_{28} = 0.0008796633$
29	$A^{29}_{1,12} = 0.9969087574$	$p_{29} = 0.0007145109$
30	$A^{30}_{1,12} = 0.9974891226$	$p_{30} = 0.0005803652$

From the table above, we can see that, when $M = 11$, we are most likely to win the game at the 3rd, 4th, and 5th rolls, which have the highest probabilities. The median game length is 5 rolls, as the probability of winning the game at or within 5 rolls is about 0.541, which is around 50%. To show the distribution of the number of rolls until the game ends, we plot the probabilities in the above table in plots (Fig.1 & 2) using Sage code:

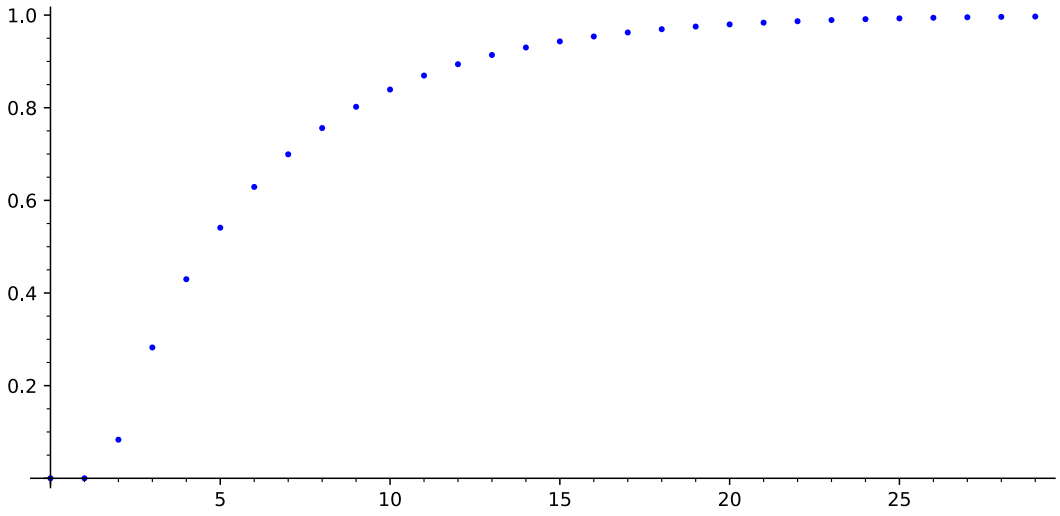

Figure 1: The distribution of the probabilities ending at or before the given number of rolls when M =11

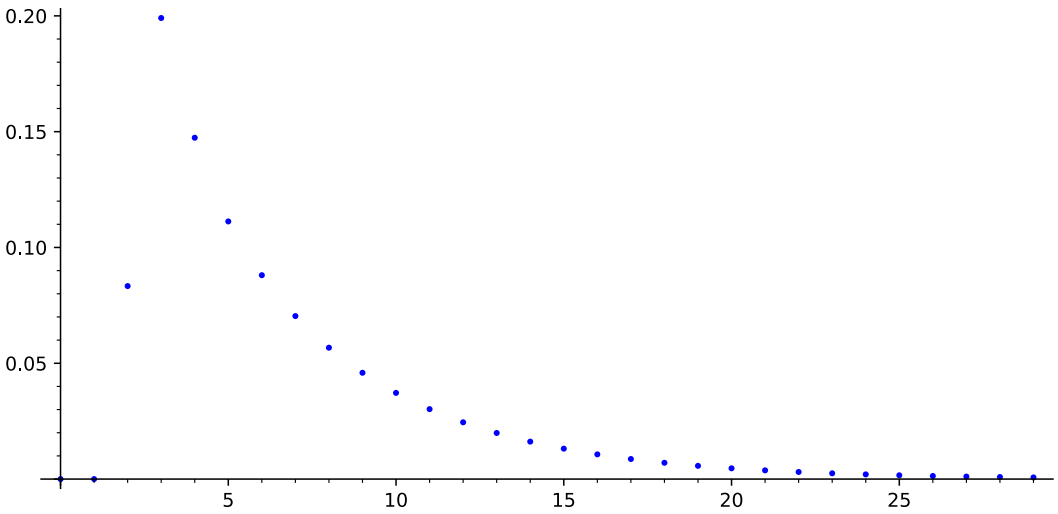

Figure 2: The distribution of the probabilities ending on the given number of rolls when M =11

From these two plots, we can see that, the probability of winning on the ith roll decreases and is approaching 0 as the number of rolls becomes larger. However, the probability never reaches 0 except when $i = 1$, since we only have one absorbing state and will eventually win the game. Even though it's not that possible to win after a large number of rolls, it is still possible. Besides, the cumulative probability of winning before or at the ith roll increases and is approaching 1 as the number of rolls increases. Similar to probabilities in **Fig.1**, the cumulative probability will never reach 1, since we might have bad luck and win the game after playing for a large number of rolls.

3 Dice Game with the value of M changing

We hope to further explore how the expected number of rolls in the game changes with the value of M and estimate the function between them. To do this, we first define $f(M)$ to be the expected number of rolls when the score goal is M. Then, we calculate the expected values of as many values of M as possible, as with more data points, we could estimate the function $f(M)$ better. Later, we use these values to plot to see the pattern and try to capture the behavior of $f(M)$.

To calculate the expected value for many values of M, we put the Sage code in the previous section into a for loop on M. We calculated values of M from 11 to 200. We stopped at M = 200 since M larger than 200 will take more than 10mins for Sage to calculate and plot data. Fig.3 is the plot of $f(M)$ drawn by Sage code.

From the plot, we notice that the expected number is overall increasing with the increase of M, which makes sense since even if we roll 6 all the time, with larger M we have to roll more times.

Another thing we notice is that some of the adjacent Ms have similar expected values, while $180 \leq M \leq 190$, $f(M)$ have a huge increase gap as M increases. We believe this happens due to the prime numbers and twice prime numbers presenting in this range.

To capture the behavior of the function $f(M)$, we are planning to seek for two functions c and d with:

$$d(M) < f(M) < c(M)$$

for at least all the Ms we calculated. We want $c(M)$ and $d(M)$ to be "close" to $f(M)$ to bound $f(x)$. By observation, we think $f(M)$ looks like an exponential function. To check our thoughts, we calculated the log value of all the expected numbers and draw a plot to see whether $log(f(M))$ looks linear.

Fig.4 is the plot of $log(f(M))$ drawn by Sage code.

The plot of $log(f(M))$ looks pretty linear. To find the bound lines of $log(f(M))$, we first use $(11, log(f(11)))$ and $(200, log(f(200)))$ to mimic the linear regression line of $log(f(M))$. By coding, we get the equation of the regression line:

$$log(f(M)) = 0.004982176463 * M + 0.003564707331$$

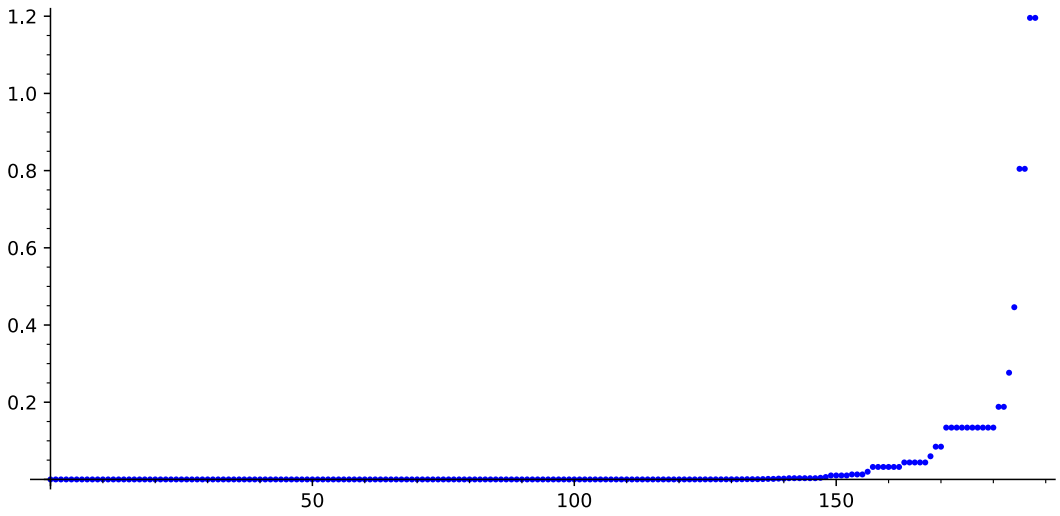

Figure 3: $f(M)$ with M from 11 to 200

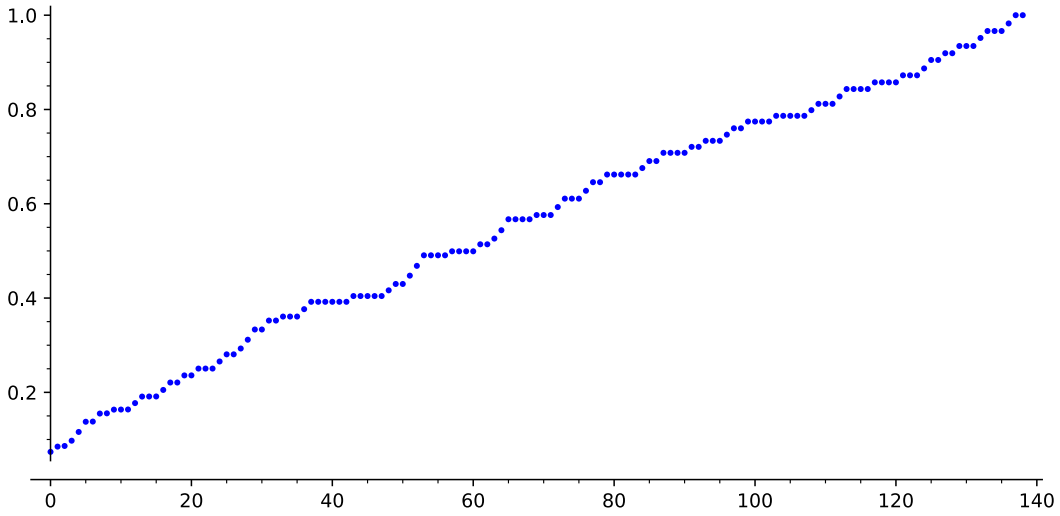

Figure 4: $log(f(M))$ with M from 11 to 200

To draw the bound lines, we change the y-intersect of the regression line to see at which values the lines $log(c(M))$ and $log(d(M))$ are close to $log(f(M))$ that

$$log(c(M)) < log(f(M)) < log(d(M))$$

while these three lines have no intersections. By trying several y-intersects, we found the bound lines to be:

$$log(c(M)) = 0.004982176463 * M - 0.04643529267$$

$$log(d(M)) = 0.004982176463 * M + 0.08356470733$$

Fig. 5 drawn by Python code gives us the plot showing $log(f(M))$ and its two bound lines.

As we have the equations of $log(c(M))$ and $log(d(M))$, we could draw $c(m)$ and $d(m)$ to check whether these two lines bound $f(M)$ as well. However, when we calculated $c(M)$ and $d(M)$ and draw the plot using Python code, we found that these two lines are not bounding $f(M)$ (Fig.6). $c(M)$ is overlapping with $d(M)$, which theoretically should not happen.

After checking the Python code used for $c(M)$ and $d(M)$ calculations and plotting to make sure they are correct, we suspect that this situation happens due to the small value we choose for the y-intercepts of $log(c(M)$ and $logd(M)$

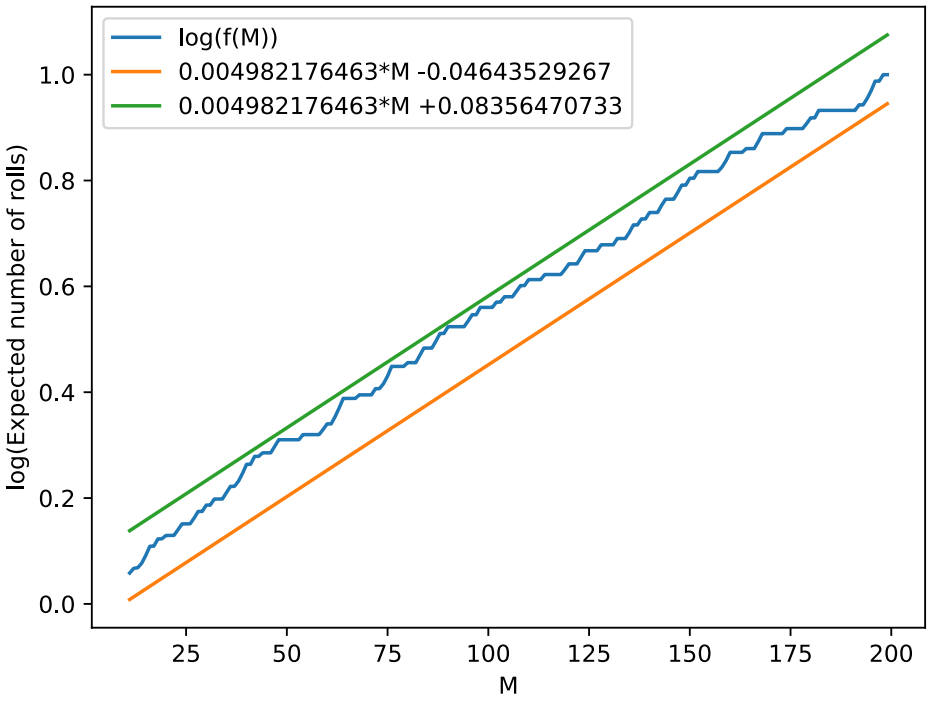

Figure 5: $log(f(M))$ with bound lines

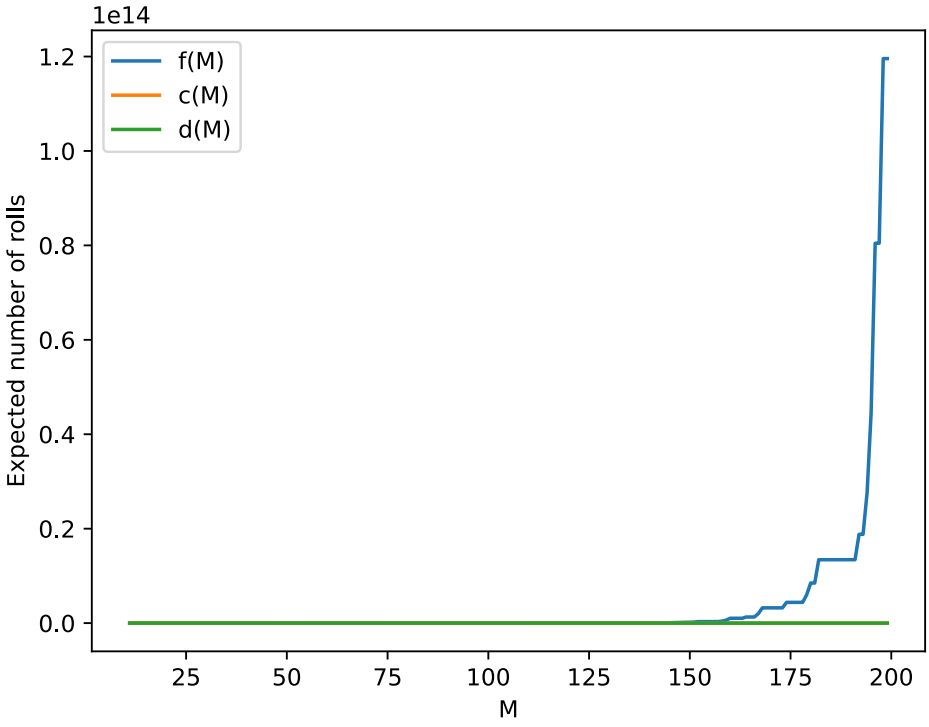

Figure 6: $f(M)$ with "bound lines"

while bounding. Such small y-intercepts of bounding lines might be related to the choice of the regression line of $log(f(M))$. We didn't use the real regression line for later analysis, which might influence the values of slopes and y-intercepts of bounding lines and then lead to such a final result. Also, we might need to use other bound strategies rather than moving the linear regression line of $log(f(x))$ up and down in order to get the right bounding lines for $f(M)$. Moreover, we said that the $log(f(M)$ looks linear, while it might not be linear if we calculate more expected values of larger M. This might be another potential reason that we didn't get bounding lines between $f(M)$. Further research and analysis are needed to check which part goes wrong and then estimate the distribution of the expected number of rolls.

4 Appendix

4.1 Sage Code for Game with M = 11

```
M = 11

# transition matrix A
A = zero_matrix(QQ, M+1, M+1);
for i in range(6):
    A[0,i+1]=1/6
A[M,M]=1;
for i in range(1,M):
    for j in range(1,7):
        score = i + j
        if (score >= M):
            put = M
        else:
            if (is_prime(score) or is_prime(floor(score/2))):
                put = floor(score/2)
            else:
                put = score
        A[i,put]+=1/6
A

# calculate the expected number of times before winning in state j after starting from state i
Q = A[:M, :M]
N = (matrix.identity(M)-Q).inverse()
N

# calculate the expected number of turns until absorption
sum(N[0][i] for i in range(M))

# Distribution Plot of Probabilities of ending the game at ith roll
probs=[]
p_now = 0
p_pre = 0
for i in range(30):
    Ai = A^i
    p_now = Ai[0,M]
    p_exact = p_now-p_pre
    probs.append(p_exact)
    p_pre = p_now

list_plot(probs)
```

4.2 Sage Code for Game with M between 11 and 200

```
# Calculate expected number of rolls for each M value
ExpectedValue = []
for M in range(11, 200):
```

```
    A = zero_matrix(QQ, M+1, M+1);
    for i in range(6):
        A[0,i+1]=1/6
    A[M,M]=1;
    for i in range(1,M):
        for j in range(1,7):
            score = i + j
            if (score >= M):
                put = M
            else:
                if (is_prime(score) or is_prime(floor(score/2))):
                    put = floor(score/2)
                else:
                    put = score
            A[i,put]+=1/6

    Q = A[:M, :M]
    N = (matrix.identity(M)-Q).inverse()

    e =sum(N[0][i] for i in range(M))
    ExpectedValue.append(e)

# Plot M vs. Expected number of rolls
list_plot(ExpectedValue)

# Plot log(f(M))
log_EV = [log(x, e) for x in ExpectedValue]
list_plot(log_EV)

# Calculate slope and y-intercept of the linear regression line for log(f(M))
%python
k = (log_EV[-1]-log_EV[0])/(200-11)
b = log_EV[0] - k*11

# Plot log(f(M)) with its two bound lines
%python
import matplotlib.pyplot as plt
log_c = []
log_d = []
for i in range(11, 200):
    log_c.append(k*i + b-0.05)
    log_d.append(k*i + b+0.08)
y1 = log_EV
y2 = log_c
y3 = log_d
x = range(11,200)
fig, ax = plt.subplots()

ax.plot(x, y1, label = 'log(f(M))')
ax.plot(x, y2, label = '0.004982176463*M␣-0.04643529267')
ax.plot(x, y3, label = '0.004982176463*M␣+0.08356470733')

ax.legend()
ax.set_xlabel('M')
ax.set_ylabel('Expected␣number␣of␣rolls')

plt.show()

# Calculate c(M) & d(M)
%python
import matplotlib.pyplot as plt
import numpy as np
```

```
c = np.exp(log_c)
d = np.exp(log_d)

# Plot f(M) with its two "bound lines" c(M) & d(M)
y4 = ExpectedValue
y5 = c
y6 = d
x = range(11,200)
fig, ax = plt.subplots()

ax.plot(x, y4, label = 'f(M)')
ax.plot(x, y5, label = 'c(M)')
ax.plot(x, y6, label = 'd(M)')

ax.legend()
ax.set_xlabel('M')
ax.set_ylabel('Expected␣number␣of␣rolls')

plt.show()
```