Riddler 220218 – The Lucky Coin

The Riddler 220218 asks if you can find the lucky coin. Starting with 1,000,000 coins, flip them all and discard the tails. Repeat until you have exactly one coin, if this does in fact occur; you might end up with none without ever having had exactly one coin. What is the probability that you end up with one coin?

Let p(n) be the probability that a sequence of flips starting with n coins has exactly one coin at some point in the sequence of flips. So we can set p(0) = 0 and p(1) = 1.

Now lets look at how to inductively increase n. Suppose n = 2. When you flip n coins, you can end up with any number from 0 to n heads, with i heads having a probability of $\frac{\binom{i}{n}}{2^n}$. So for 2 coins, the probability that you end up with 0,1, or 2 heads respectively is 1/4, 1/2, and 1/4. Now we can determine p(2) as the weighted sum of

$$p(2)=\frac{1}{4}\times p(2)+\frac{1}{2}\times p(1)+\frac{1}{4}\times p(0)=\frac{1}{4}\times p(2)+\frac{1}{2} \quad .$$

It is obvious that you have a 1/2 probability of ending up with 1 heads, and 1/4 with 0 heads; it is a little more subtle to realize that if you end up with 2 heads you are exactly where you started and can just use p(2) in the expression for p(2).

Then rearranging

$$\frac{3}{4}\times p(2)=\frac{1}{2}$$

$$p(2)=\frac{2}{3} \quad .$$

This also makes sense; there is only a 3/4 probability you leave the 2 head state, and of that, 1/2 goes to the 1 head state, and 1/4 to the 0 head state. So 2/3 = (1/2) / (3/4) is the probability you end up in the 1 state when you do eventually leave the 2 head state.

Now it is easy to determine the general formula for p(n) as

$$p(n)=\sum_{i=0}^{n} p(i)\times\frac{\binom{i}{n}}{2^n} \quad .$$

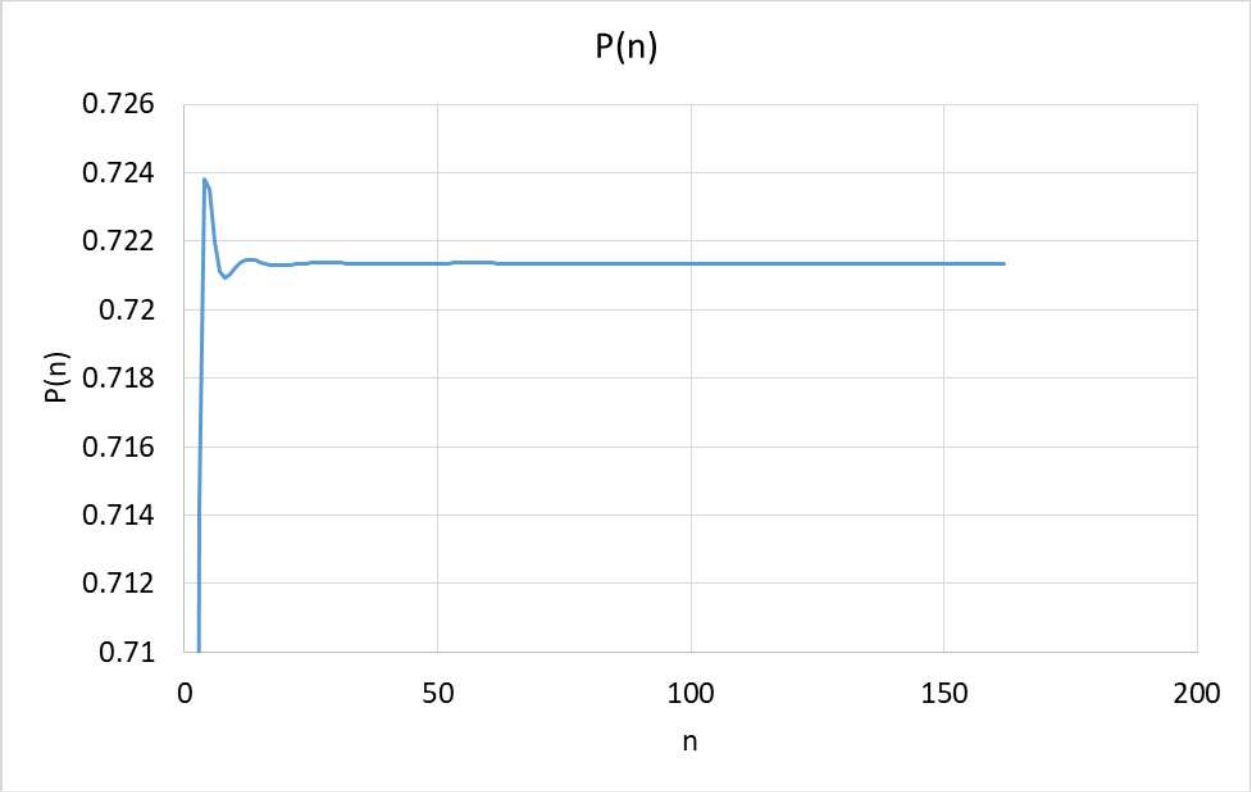
Using the fact that $\binom{n}{n}=1$ and the same rearrangement we did above for n=2, we can express p(n) explicitly. Note that the ending index for i has changed compared to above.

$$p(n)=\frac{2^n}{2^n-1}\times\sum_{i=0}^{n-1} p(i)\times\frac{\binom{i}{n}}{2^n}$$

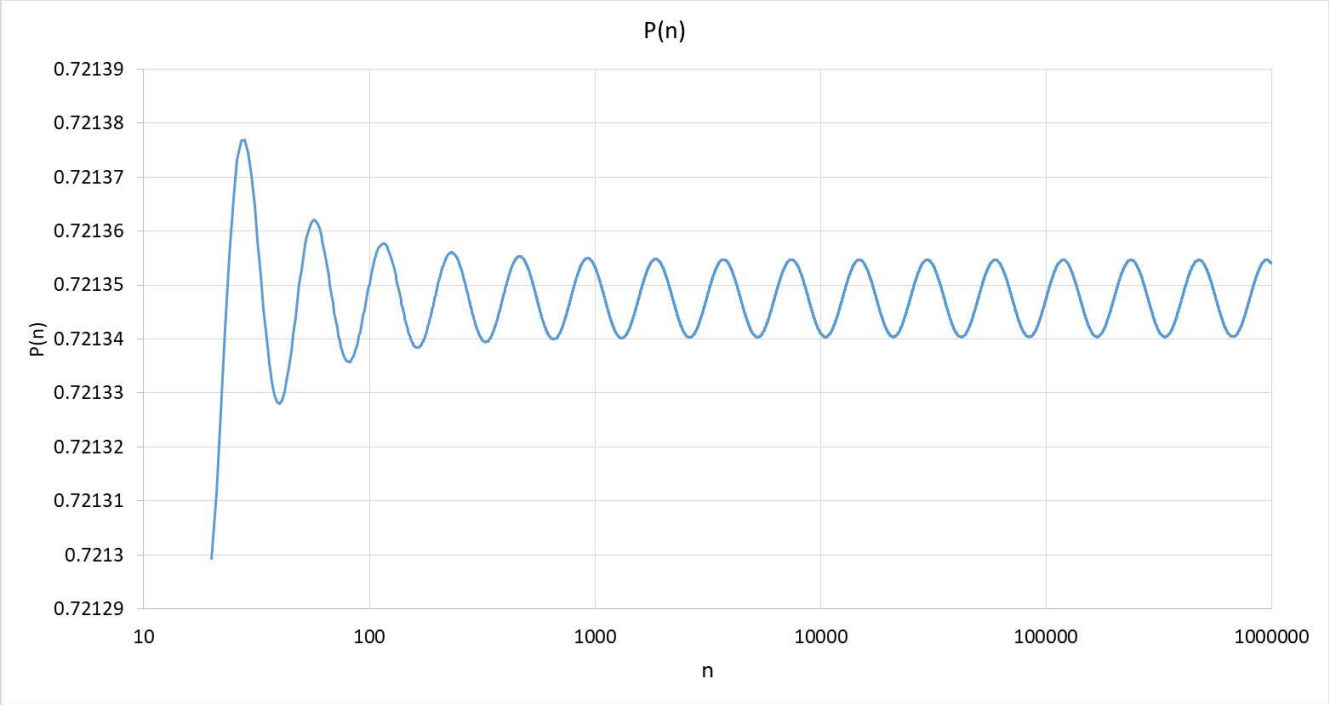
Since computing p(n) involves a sum over n-1 values of previous p(i), the formula is quadratic in n, which makes it barely tolerable for n=1000000 using double precision floating point.

The answer is 0.721353963.

However, looking at p(n) on some plots reveals some interesting behaviour.
First, look at the first 160 or so points. P(n) appears to rapidly converge to some value, so nothing particularly interesting.



This appears to be what one might expect, a rapid convergence to some final value. However, plotting values for n>20 on a log scale, and zooming into the region of supposed convergence reveals something entirely different.



We use a log scale on n to accommodate the huge range, and see that p(n) does not converge but appears to decay to a steady oscillation around some value. Also, the frequency of the oscillation appears constant on the log plot, so in fact the frequency is decreasing exponentially.

One concern is that this is purely numerical noise since the values can become very small. This concern is disposed with by using 150 bit floating point, and seeing the same behaviour out to n=5000. This also confirms that double FP is adequate for a dozen or so digits. The high precision FP is too slow to go out to 1M.

Since there is an explicit formula for p(n) using only rationals, it is also possible to compute exact rational answers, but this seems to have exponential time and going to a few hundred is all that is tolerable.

We can try and model the steady state behaviour using a $\sin(\log(n))$ term for the oscillatory behaviour, and a $1+\frac{a}{b+n}$ factor for the magnitude. Including appropriate constants for magnitude and phase we propose the model

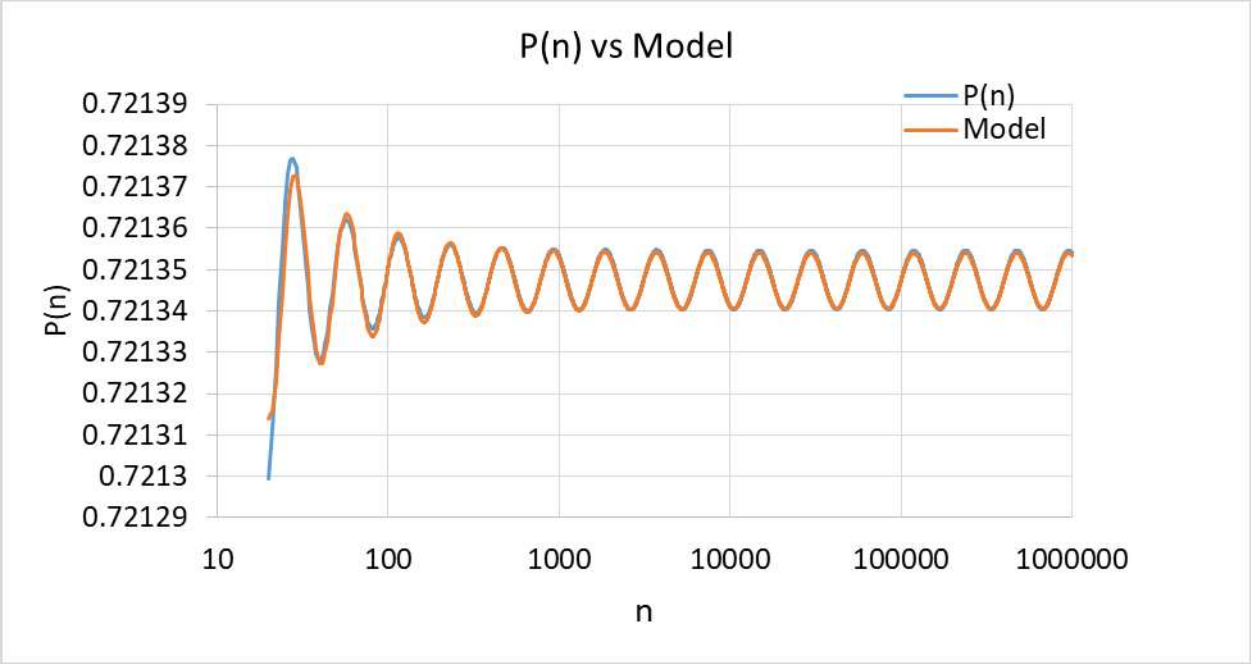
$$p'(n)=a_0+a_1\times\sin(a_2+\log(n)\times a_3)\times\left(1+\frac{a_4}{n+a_5}\right) \quad .$$

This can be tuned to capture the behaviour for large n quite well, but not as well for smaller n. See plot next page.

I am reminded of John von Neumann's statment:

> "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk."

I am utterly devoid of any insight into this behaviour.


P(n) vs Model
P(n)
Model
0.72139
0.72138
0.72137
0.72136
0.72135
0.72134
0.72133
0.72132
0.72131
0.7213
0.72129
P(n)
10
100
1000
10000
100000
1000000
n