#A19 INTEGERS 13 (2013)

TAKING THE CONVOLUTED OUT OF BERNOULLI CONVOLUTIONS: A DISCRETE APPROACH

Neil J. Calkin
Dept. of Mathematical Sciences, Clemson University, Clemson, South Carolina
calkin@ces.clemson.edu

Julia Davis
Dillsburg, Pennsylvania
juliablackburn87@gmail.com

Michelle Delcourt[1]
Dept. of Mathematics, University of Illinois at Urbana-Champaign, Urbana, Illinois
delcour2@illinois.edu

Zebediah Engberg
Department of Mathematics, Dartmouth College, Hanover, New Hampshire
zeb@dartmouth.edu

Jobby Jacob
School of Math. Sciences, Rochester Institute of Technology, Rochester, New York
jxjsma@rit.edu

Kevin James
Department of Math. Sciences, Clemson University, Clemson, South Carolina
kevja@clemson.edu

Received: 7/7/12, Accepted: 4/5/13, Published: 4/10/13

Abstract

In this paper we consider a discrete version of the Bernoulli convolution problem traditionally studied via functional analysis. We discuss several innovative algorithms for computing the sequences with this new approach. In particular, these algorithms assist us in gathering data regarding the maximum values. By looking at a family of associated polynomials, we gain insight on the local behavior of the sequence itself. This work was completed as part of the Clemson University REU, an NSF funded program[2].

[1]NSF Graduate Research Fellow DGE 1144245
[2]This research was supported by NSF grant DMS-0552799

1. Introduction

The classic Bernoulli convolution problem in analysis has an elegant discrete analogue as follows. Define two maps $\mathrm{dup}_n, \mathrm{shf}_n : \mathbb{R}^n \longrightarrow \mathbb{R}^{3n}$ defined by

$$\mathrm{dup}_n : (a_1, a_2, ..., a_{n-1}, a_n) \longmapsto (a_1, a_1, a_2, a_2, ..., a_{n-1}, a_{n-1}, a_n, a_n, \overbrace{0, ..., 0}^{n}) \quad (1)$$

$$\mathrm{shf}_n : (a_1, a_2, ..., a_{n-1}, a_n) \longmapsto (\overbrace{0, ..., 0}^{n}, a_1, a_1, a_2, a_2, ..., a_{n-1}, a_{n-1}, a_n, a_n). \quad (2)$$

Consider the finite sequences recursively given by $B_0 = (1)$ and $B_{n+1} = \mathrm{dup}_{3^n}(B_n) + \mathrm{shf}_{3^n}(B_n)$. B_n is called the n^{th} *level Bernoulli sequence*. The names "dup" and "shf" reference the duplication and shifting of the coordinates.

In this paper, we are primarily interested in the rate at which the maximum value of B_n is growing with n. We develop two independent algorithms to do this. The first gives a recursive method for computing the entire sequence B_n when n is small. By encoding the sequence as coefficients of a polynomial, the second gives a method for computing specified entries of B_n when n is larger. In addition, we provide numerical data concerning B_n.

1.1. Motivation

Classically, Bernoulli convolutions have been studied as a problem in functional analysis. A Bernoulli convolution is obtained as an infinite convolution of Bernoulli measures [1]. The Bernoulli measure, denoted by $b(X)$, is the measure corresponding to the discrete probability density function on the real line with value $1/2$ at 1 and -1. The Bernoulli convolution with parameter q for $0 < q < 1$ is the measure

$$\mu_q(X) = b(X) * b(X/q) * b(X/q^2) * ... .$$

This measure was first studied by Jessen and Wintner [4]. They showed that μ_q is continuous for any q.

A different perspective on Bernoulli convolutions is obtained through a functional equation. For $0 < q < 1$, consider the functional equation

$$F(t) = \frac{1}{2}F\left(\frac{t-1}{q}\right) + \frac{1}{2}F\left(\frac{t+1}{q}\right) \quad (3)$$

for t on the interval $I_q := [-1/(1-q), 1/(1-q)]$. It can be shown that there is a unique bounded solution $F_q(t)$ to the above equation. Moreover, $F_q(t)$ is the distribution function of μ_q, that is $F_q(t) = \mu_q((-\infty, t])$. For an introduction to this, refer to Chapter 5 in *Experimental Mathematics in Action* [1]. For a more in depth analysis, refer to *Sixty years of Bernoulli convolutions* [6].

Jessen and Winter, in [4], showed that $F_q(t)$ is either absolutely continuous or purely singular. The major question regarding the solutions of (3) is to determine the values of q which make $F_q(t)$ absolutely continuous. If $F_q(t)$ is absolutely continuous, rather than considering the function $F_q(t)$, one may consider its derivative $f_q(t) := F_q'(t)$. Upon differentiating, the functional equation for $F_q(t)$ yields the following equation:

$$f(t) = \frac{1}{2q} f\left(\frac{t-1}{q}\right) + \frac{1}{2q} f\left(\frac{t+1}{q}\right). \tag{4}$$

The existence of an absolutely continuous solution $F_q(t)$ to (3) is equivalent to the the existence of an $L^1(I_q)$ solution $f_q(t)$ to (4).

When $0 < q < 1/2$, Kershner and Wintner [5] proved that $F_q(t)$ is always singular. For these values of q, the solution $F_q(t)$ is an example of a *Cantor function*, a function that is constant almost everywhere. It can be shown that if $q = 1/2$, then the solution $F_q(t)$ is absolutely continuous.

The case when $q > 1/2$, however, is significantly harder and more interesting. In 1939, Erdős [3] showed that $F_q(t)$ is again singular for q of the form $q = 1/\theta$ with θ a Pisot number. There is little else that is known for other values of $q > 1/2$. One interesting result due to Solomyak [7] is that almost every $q > 1/2$ yields a solution $F_q(t)$ that is absolutely continuous. Hence it is surprising that no actual example of such a q is known. Specifically, the case when $q = 2/3$ remains a mystery.

In [1], Girgensohn asks the question of computing $f_q(t)$ for various values of q. The author starts an arbitrary initial function $f^0(t) \in L^1(I_q)$ and iterates the transform

$$T_q : f(t) \longmapsto \frac{1}{2q} f\left(\frac{t-1}{q}\right) + \frac{1}{2q} f\left(\frac{t+1}{q}\right) \tag{5}$$

to gain a sequence of functions $f^0, f^1, f^2, \ldots$. He shows that if this sequence of functions converges to a bounded function, then it converges to the unique solution of (4).

Calkin [2] specifically looked at the above process for $q = 2/3$. Rather than working on the interval I_q, we shift the entire interval to $[0, 1]$ for simplicity. The transform T_q now becomes the map $T : L^1([0,1]) \longrightarrow L^1([0,1])$ where

$$T : f(x) \longmapsto \frac{3}{4} f\left(\frac{3x}{2}\right) + \frac{3}{4} f\left(\frac{3x-1}{2}\right). \tag{6}$$

Geometrically, this transform (6) takes two scaled copies of $f(x)$: one on the interval $[0, 2/3]$ and the other on $[1/3, 1]$, and adds them. The scaling factor of $3/4$ gives us that

$$\int_0^1 f(x)dx = \int_0^1 Tf(x)dx.$$

In other words, the average value of $Tf(x)$ equals that of $f(x)$. In this setting, the question now reads: starting with the constant function $f^0(x) = 1$, does the iteration determined by the transform in (6) converge to a bounded function?

1.2. Discrete Version

Rather than viewing T as a transform on $[0, 1]$, we consider the discrete analogue mentioned earlier. The sequences $\mathrm{dup}(B_n)$ and $\mathrm{shf}(B_n)$ defined in (1) and (2) are analogous to the two shifted copies of the function on $[0, 1]$ in (6). We start with the sequence $B_0 = (1)$. We recursively generate sequences given by

$$B_{n+1} = \mathrm{dup}_{3^n}(B_n) + \mathrm{shf}_{3^n}(B_n). \tag{7}$$

We are interested in the global behavior of B_n. In particular, we investigate the growth rate of the maximum element of B_n as n grows large.

Before we discuss our results, we present some notations that are useful throughout this paper. We call B_n the *Bernoulli sequence on level n*. We refer to the map $(\mathrm{dup}_n + \mathrm{shf}_n) : \mathbb{R}^n \longrightarrow \mathbb{R}^{3n}$ seen in (7) as the process of *duplicate, shift, add* or DSA for short. We usually write the n^{th} level Bernoulli sequence as $B_n = (b_0, b_1, ..., b_t)$ where $t = 3^n - 1$. The fact that B_n has a total of 3^n terms follows directly from the definition of dup_n and shf_n in (1) and (2). We define $m_n := \max(B_n)$.

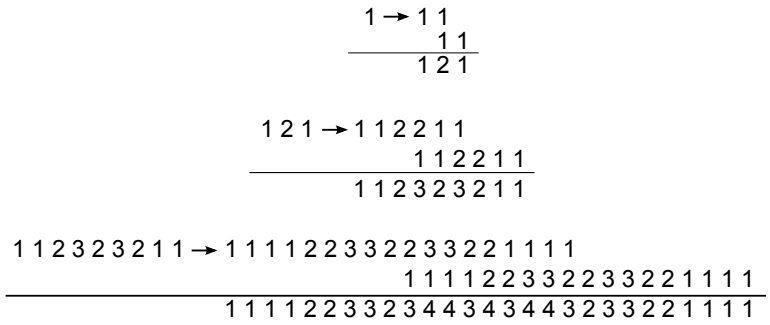

Figure 1: This shows the process of DSA at three low levels. The figure demonstrates computation of the Bernoulli sequence on level $n + 1$ from the Bernoulli sequence on level n for $n = 0, 1$ and 2 using the definition.

It is immediate from the definition that the maximum value of B_n must occur on the middle third of the Bernoulli sequence. Furthermore, because the sequence is palindromic, the maximum must occur on the first half of the middle third.

The following observation can be proved using induction on n and the definition of B_n in (7).

Observation 1.1. *The mean of the elements of* B_n *is* $(4/3)^n$.

The major question that we considered is whether m_n also grows like $(4/3)^n$. A positive result here is equivalent to the existence of a bounded solution to the functional equation (4) for $q = 2/3$. $F_q(t)$ is absolutely continuous at $q = 2/3$ if and only if two conditions hold: first, $m_n = O((4/3)^n)$, and secondly, the iterations of the step functions is pointwise continuous. Beginning with a computational approach, we address this question and other related questions throughout this paper.

2. Recursive Attempts at Computing the Bernoulli Sequence

2.1. The Naive Method: DSA

The process of *duplicate, shift, add* gives a naive method for computing Bernoulli sequences. Despite the simplicity in describing this process, DSA is not computationally feasible for large values of n. The primary shortfall of the DSA method is that each Bernoulli level B_n has 3^n terms—each successive computation takes roughly three times as long as the previous. Likewise, at each successive level we must keep track of three times as many entries as on the previous level. Using the DSA method without modifications, we have been able to compute B_n for $n = 1, 2, ..., 20$. However, the fact that one must know all 3^n entries of level n to compute level $n + 1$ limits the practicality of this algorithm.

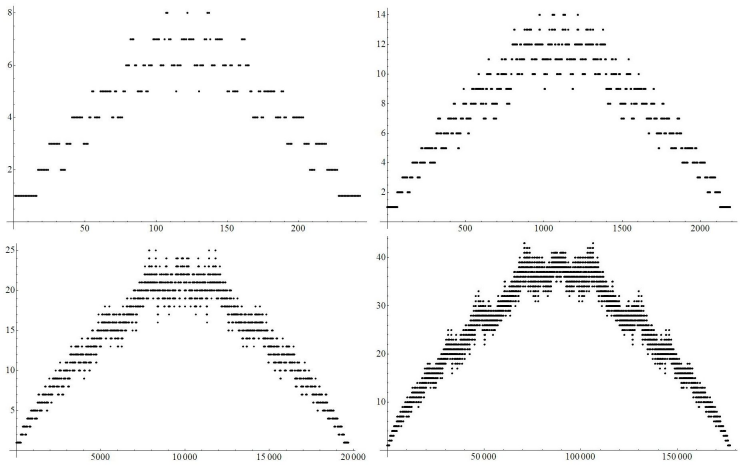

Figure 2: This shows some of the low level Bernoulli sequences generated using DSA. In the above plots, the horizontal axis gives the index and the vertical axis gives the entry corresponding to that index in the Bernoulli sequence. Moving clockwise from the top left, we see B_n for $n = 5, 7, 11$ and 9.

2.2. The Improvement: DEM

We now consider an alternate approach that addresses some of the issues arising from the DSA algorithm. The process we call *double, enlarge, merge*, abbreviated DEM, is a way of encoding the Bernoulli sequence B_n as a sequence of length $2(2^n - 1)$. The advantage with DEM is that this auxiliary sequence is in the order of 2^n as opposed to the 3^n size increase required for the DSA process. The DEM algorithm is based on the observation that in a given Bernoulli sequence, many individual entries are consecutively repeated. Rather than keeping consecutive repeats, we only keep the index where the Bernoulli sequence either increases or decreases. For this encoding to be meaningful, it is important to note that when comparing two consecutive entries in a Bernoulli sequence, the difference between these entries will never be greater than one in absolute value. This can be proved inductively from the definition in (7).

Given the Bernoulli sequence $B_n = (b_0, b_1, ..., b_t)$ with $t = 3^n - 1$, the DEM representation is $(d_1, ..., d_r)$ where d_i is the index of the i^{th} jump in B_n up to a sign (by jump, we mean two consecutive entries which are not equal). Suppose the i^{th} jump occurs at index j in B_n, so b_j and b_{j+1} are different. Then

$$d_i = \begin{cases} j+1 & \text{if } b_j < b_{j+1} \\ -(j+1) & \text{if } b_j > b_{j+1}. \end{cases}$$

The DEM representation of $B_2 = (1, 1, 2, 3, 2, 3, 2, 1, 1)$ is $(2, 3, -4, 5, -6, -7)$.

We now describe how to translate the DSA method to this new representation. The process of *duplicate* becomes *double*: each element from the original list is multiplied by two. The process of *shift* becomes *enlarge*: each element from the doubled list is modified by 3^n as follows: if the element is positive, we add 3^n, for negative elements we subtract 3^n. The process of add translates to *merge*: we discard the original sequence and consider the new lists attained in the *double* and *enlarge* processes. We then add two additional elements 3^n and $-2(3^n)$ to these lists, respectively. Finally, we merge sort the elements of the two lists according to their absolute value to obtain the DEM representation of the next Bernoulli level.

2.3. Data

Using the DEM method, we are able to compute the Bernoulli sequence B_n up to level $n = 27$. For each of these levels, the maximum value is of particular interest. Typically, the maximum is attained many times on a given level. Although we found no clear pattern regarding the location of the maximum values, we do see obvious patterns in the convergence of the maximums themselves.

Below we provide a table detailing the data we collected using the DSA and DEM algorithms.

Level n	m_n	$m_n(3/4)^n$	Level n	m_n	$m_n(3/4)^n$
0	1	1	14	99	1.763976853
1	2	1.5	15	131	1.750613395
2	3	1.6875	16	176	1.763976853
3	4	1.6875	17	232	1.743931662
4	6	1.8984375	18	309	1.742052425
5	8	1.8984375	19	410	1.733595860
6	11	1.957763672	20	545	1.728310507
7	14	1.868774414	21	728	1.731481719
8	18	1.802032471	22	962	1.716022061
9	25	1.877117157	23	1283	1.716468012
10	33	1.858345985	24	1705	1.710782128
11	43	1.816110849	25	2266	1.705263476
12	56	1.773875713	26	3024	1.706768563
13	75	1.781794801	27	4025	1.703805423

Table 1: m_n denotes the maximum value at each level. The maximums m_n appear to grow like $(4/3)^n$. The fact that $m_n(3/4)^n$ seems to be converging, albeit slowly, supports this claim.

3. A Polynomial Approach to DSA

3.1. Translating DSA Into a Polynomial Recursion

By encoding these sequences as coefficients of polynomials, the process of *duplicate, shift, add* gives a particularly nice recursive relation among the polynomials. Let $B_n = (b_0, b_1, ..., b_t)$ be the Bernoulli sequence on level n where $t = 3^n - 1$. Consider the polynomial $p_n(x) := b_0 + b_1 x + ... + b_t x^t$.

We describe how these polynomials behave under the process of DSA. We see that the duplication $b_0, b_0, b_1, b_1, ..., b_r, b_r$ corresponds to the polynomial $(1 + x)p_n(x^2)$. Shifting the sequence 3^n places to the right corresponds to multiplication by x^{3^n}. By adding the duplicate and the shift of the sequence, we arrive at the sequence on the next level. This yields the recursive relation

$$p_{n+1}(x) = (1 + x)p_n(x^2)\left(1 + x^{3^n}\right) \tag{8}$$

with $p_0(x) = 1$.

3.2. Explicit Formula for $p_n(x)$

This formula (8) allows us to explicitly solve for $p_n(x)$.

Theorem 3.1. *For $n \geq 1$, the polynomials $p_n(x)$ satisfy*

$$p_n(x) = \prod_{i=0}^{n-1} \left(1 + x^{2^i}\right) \prod_{j=0}^{n-1} \left(1 + x^{2^{n-1}(3/2)^j}\right). \tag{9}$$

Proof. We proceed by induction on n. We know $p_1(x) = 1 + 2x + x^2$, and thus the formula holds for the case $n = 1$. Assume the formula holds for $p_n(x)$. We have

$$\begin{aligned} p_{n+1}(x) &= (1+x)p_n(x^2)\left(1 + x^{3^n}\right) \\ &= (1+x)\prod_{i=0}^{n-1}\left(1 + (x^2)^{2^i}\right)\prod_{j=0}^{n-1}\left(1 + (x^2)^{2^{n-1}(3/2)^j}\right)\left(1 + x^{3^n}\right) \\ &= (1+x)\prod_{i=0}^{n-1}\left(1 + x^{2^{i+1}}\right)\prod_{j=0}^{n-1}\left(1 + x^{2^n(3/2)^j}\right)\left(1 + x^{2^n(3/2)^n}\right) \\ &= \prod_{i=0}^{n}\left(1 + x^{2^i}\right)\prod_{j=0}^{n}\left(1 + x^{2^n(3/2)^j}\right). \end{aligned}$$

□

3.3. A Bound on the Coefficients

By factoring p_n, we can obtain a bound on how fast the coefficients grow with the level n.

Theorem 3.2. *We have $m_n = O((\sqrt{2})^n)$.*

Proof. Define polynomials q_n, r_n, s_n by

$$q_n(x) = \prod_{i=0}^{n-1}\left(1 + x^{2^i}\right) \qquad s_n(x) = \prod_{\substack{1 \leq j \leq n-1 \\ j \text{ odd}}}\left(1 + x^{2^{n-1}(3/2)^j}\right)$$

$$r_n(x) = \prod_{\substack{1 \leq j \leq n-1 \\ j \text{ even}}}\left(1 + x^{2^{n-1}(3/2)^j}\right) = \prod_{j=1}^{\lfloor (n-1)/2 \rfloor}\left(1 + x^{2^{n-1}(9/4)^j}\right).$$

We see that

$$p_n(x) = q_n(x)\left(1 + x^{2^{n-1}}\right) r_n(x)s_n(x).$$

Consider the polynomial $q_n(x)r_n(x)$. Because $9/4 > 2$, we are left with distinct powers of x when we expand $q_n(x)r_n(x)$. In other words, the coefficients of $q_n(x)r_n(x)$ are all either 0 or 1. Hence the coefficients of $q_n(x)r_n(x)(1 + x^{2^{n-1}})$ are all either 0, 1, or 2. In particular, the coefficients are bounded. On the other hand, there are at most $n/2$ terms in the product defining $s_n(x)$. Thus there are at most $2^{n/2}$ nonzero terms in the polynomial $s_n(x)$. Also, the coefficients of $s_n(x)$ are bounded as in the case of $r_n(x)$. Therefore the coefficients of $p_n(x)$ are all $O(2^{n/2}) = O((\sqrt{2})^n)$. □

3.4. An Algorithmic Implementation: PIP

The fact that the Bernoulli sequence can be realized as the coefficients of an explicitly defined polynomial provides us with an algorithm for computing specified values on high levels. The algorithms DSA and DEM are useful for computing entire levels, though this task becomes impossible for large n due to the recursive nature of the algorithm. The following algorithm, which we dub PIP, allows us to compute values on much higher levels. The algorithm is non-recursive—we need no lower levels to compute entries of B_n. The name PIP stands for *polynomial isolated point*; this algorithm computes the entry corresponding to a given index on a given level.

Our algorithm is based on the following idea. Suppose $S = \{a_1, ..., a_n\}$ is a sequence of positive integers. Consider the polynomial

$$f(x) = \prod_{i=1}^{n}(1 + x^{a_i}) = \sum_{j=1}^{m} \alpha_j x^j.$$

Then α_j is the number of ways that j can be written as a sum of distinct elements from S [8]. This idea is applicable to the coefficients of our polynomial because our polynomial is a product of terms of the form $(1 + x^a)$. Applying this idea to our situation, we find that the coefficient b_j of x^j in $p_n(x)$ (which is the j^{th} entry on the n^{th} Bernoulli sequence) is precisely the number of ways that j can be written as a sum of distinct terms in the sequence

$$S = \{1, 2, 4, ..., 2^{n-2}, 2^{n-1}, 2^{n-1}, 2^{n-2}3, 2^{n-3}3^2, ..., 2^2 3^{n-3}, 2^1 3^{n-2}, 3^{n-1}\}. \quad (10)$$

The values in the sequence S are just the exponents of the factors of $p_n(x)$ in (9).

We now outline an algorithm that can be used to calculate the entry b_j for a fixed level n. Let $S = \{a_1, ..., a_n\}$ where each $a_i > 0$. Let $\mathrm{N}_S(k)$ denote the number of ways k can be written as a sum of elements from S. Our entire algorithm is based on the following observations:

- For any $i \in \{1, ..., n\}$, we have

$$\mathrm{N}_S(k) = \mathrm{N}_{S\setminus\{a_i\}}(k) + \mathrm{N}_{S\setminus\{a_i\}}(k - a_i). \quad (11)$$

- If $k > \sum_{s\in S} s$, then $\mathrm{N}_S(k) = 0$.
- If $k < 0$, then $\mathrm{N}_S(k) = 0$.

These three relations provide an algorithm for computing the j^{th} entry on a given Bernoulli level. However, our sequence S in (10) takes on a particularly nice form—taking advantage of this we can increase the efficiency of our algorithm. Let

S be as in (10). Suppose $0 < k < 2^{n-1}$. Any element of S containing a positive power of 3 is strictly larger than k itself, so these terms become useless when writing k as a sum of elements in S. Hence, we are left with the distinct powers of 2 in S. But k can now be written uniquely; this is simply the binary expansion of k. So $N_S(k) = 1$ in this case.

Based on the above ideas, we implement a tree branching algorithm to compute $N_S(k)$ for various values of k and n. See Figure 3 for a specific example of the PIP algorithm.

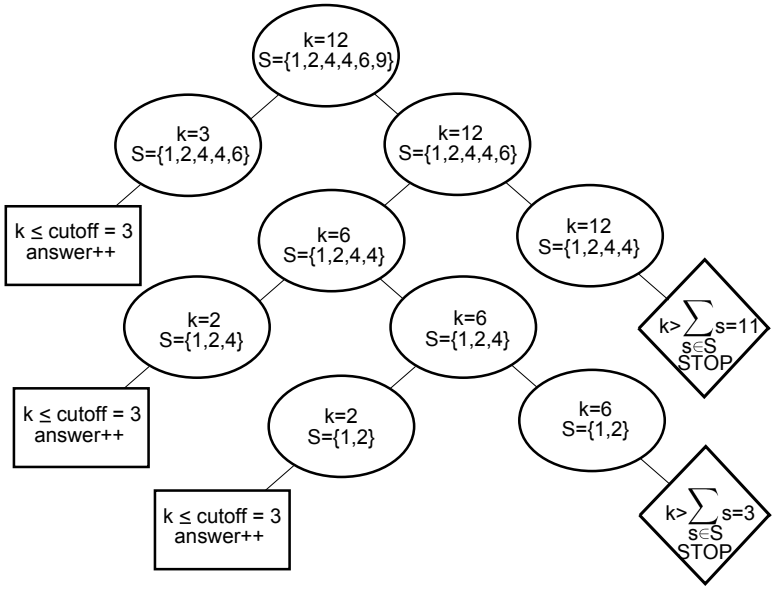

Figure 3: The above flowchart illustrates the PIP algorithm. It shows the computation of $\mathrm{N}_S(k)$ for $k = 12$ and $S = \{1, 2, 4, 4, 6, 9\}$. The boxes containing the word "cutoff" account for the fact that if $0 < k < 2^{n-1}$, then there is a unique way to write k as a sum of elements in S (the binary expansion of k). As the diagram suggests, $\mathrm{N}_S(k) = 3$, corresponding to the fact that there are three boxes containing the word answer++. Hence the 12^{th} entry on the 3^{rd} Bernoulli level is 3.

3.5. Data

Our PIP algorithm is used to compute isolated points on B_n for a given n. It is infeasible to compute entire levels using PIP, so we are not able to calculate the global maximum of a given level with this method. Instead, PIP provides us with local information on the Bernoulli sequence. In particular, the PIP algorithm gives evidence that the Bernoulli sequence B_n is converging uniformly as n grows large.

Furthermore, we can gather local data on much higher levels than we were able to with DSA or DEM.

We now describe one way in which PIP can be used to gather local information on the Bernoulli sequence. Fix α in $[0,1]$. Consider the index $k = \lceil \alpha(3^n - 1) \rceil$. We now use PIP to compute the entry b_k at index k for levels $n = 1, 2, ..., 50$ (we have used PIP to compute individual entries on levels as high as $n = 70$). Finally, we take the quotient of b_k by $(4/3)^n$. This data is presented in Table 2.

Level	0.36	0.38	0.4	0.42	0.44	0.46	0.48	0.5
1	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000
2	1.687500	1.687500	1.687500	1.687500	1.687500	1.125000	1.125000	1.125000
3	1.265625	1.687500	1.687500	1.687500	1.687500	1.265625	1.265625	1.687500
4	1.265625	1.582031	1.582031	1.265625	1.582031	1.265625	1.582031	1.898438
5	1.186523	1.186523	1.661133	1.423828	1.898438	1.423828	1.423828	1.898438
6	1.245850	1.423828	1.779785	1.423828	1.779785	1.423828	1.423828	1.779785
7	1.201355	1.334839	1.735291	1.601807	1.601807	1.334839	1.601807	1.601807
8	1.201355	1.301468	1.802032	1.701920	1.501694	1.401581	1.501694	1.802032
9	1.201355	1.351524	1.877117	1.802032	1.501694	1.351524	1.576778	1.802032
10	1.182584	1.407838	1.858346	1.689405	1.464151	1.520465	1.520465	1.576778
11	1.309289	1.435995	1.816111	1.562700	1.393759	1.520465	1.647170	1.689405
12	1.330407	1.393759	1.742199	1.615494	1.488789	1.520465	1.647170	1.710523
13	1.354164	1.401679	1.710523	1.591737	1.472950	1.520465	1.591737	1.520465
14	1.336346	1.425436	1.674887	1.567979	1.496708	1.478890	1.621433	1.639251
15	1.296256	1.469981	1.643706	1.576888	1.496708	1.496708	1.630342	1.630342
16	1.312960	1.433231	1.603615	1.573548	1.533457	1.493367	1.583570	1.583570
17	1.315466	1.450771	1.578559	1.563525	1.525940	1.510906	1.563525	1.623661
18	1.324862	1.460167	1.578559	1.544733	1.516544	1.482718	1.572921	1.668762
19	1.314996	1.429160	1.594063	1.530638	1.530638	1.496812	1.577149	1.640574
20	1.319224	1.436559	1.585606	1.525353	1.534867	1.474614	1.569750	1.617318
21	1.305747	1.446073	1.591156	1.534074	1.541209	1.479370	1.560236	1.584020
22	1.309314	1.455586	1.589372	1.539425	1.539425	1.478776	1.566182	1.641102
23	1.311098	1.446221	1.581345	1.547898	1.547898	1.485019	1.577331	1.650913
24	1.312436	1.453914	1.585358	1.551243	1.535189	1.483012	1.578334	1.609440
25	1.310178	1.455419	1.592382	1.563785	1.537446	1.478748	1.576579	1.610443
26	1.304910	1.448270	1.603482	1.563409	1.533495	1.479877	1.581470	1.609690
27	1.308438	1.448129	1.609832	1.559881	1.526864	1.478607	1.577237	1.601789
28	1.308967	1.444848	1.615017	1.556601	1.532790	1.485486	1.579142	1.612160
29	1.315555	1.448896	1.616049	1.554379	1.529853	1.476517	1.576046	1.618192
30	1.308114	1.451872	1.616346	1.554914	1.527055	1.471516	1.577594	1.610810
31	1.304274	1.449729	1.617150	1.550718	1.531565	1.470757	1.577639	1.608846
32	1.304073	1.447017	1.616983	1.552593	1.536219	1.473034	1.575094	1.610252
33	1.307213	1.447419	1.616631	1.551990	1.531498	1.474767	1.572030	1.613467
34	1.308757	1.447589	1.615463	1.549579	1.532515	1.474315	1.575458	1.615463
35	1.308474	1.449637	1.616141	1.549989	1.529139	1.473750	1.574780	1.617582
36	1.308125	1.449245	1.617667	1.550127	1.530548	1.474640	1.574123	1.616396
37	1.307886	1.450556	1.617897	1.552057	1.530150	1.474251	1.574417	1.617445
38	1.307892	1.450008	1.619012	1.551682	1.529727	1.473464	1.574763	1.617743
39	1.307544	1.450535	1.619714	1.550310	1.529070	1.473370	1.575921	1.617233
40	1.307356	1.450019	1.620015	1.550705	1.529959	1.475150	1.575666	1.616817
41	1.308606	1.449664	1.620206	1.550265	1.529659	1.475490	1.574831	1.616616
42	1.308272	1.448889	1.620105	1.550073	1.529771	1.476246	1.575167	1.617084
43	1.309044	1.449437	1.619960	1.550729	1.529682	1.477035	1.574522	1.616727
44	1.309736	1.449494	1.619917	1.550938	1.529485	1.475599	1.574488	1.617941
45	1.308585	1.449791	1.619751	1.550583	1.529578	1.475369	1.574587	1.617180
46	1.308839	1.449973	1.619832	1.550568	1.529498	1.476223	1.574642	1.615671
47	1.309522	1.450225	1.619787	1.550845	1.529697	1.476398	1.574457	1.617048
48	1.309481	1.449946	1.619833	1.550384	1.529741	1.476457	1.574743	1.616217
49	1.309458	1.450029	1.619892	1.550488	1.529521	1.476353	1.575004	1.616225
50	1.309438	1.450046	1.619848	1.550631	1.529865	1.476265	1.574653	1.616802

Table 2: We consider various values for α (appearing in the top row), and compute associated entry of the Bernoulli sequence. We chose these specific α values because the only portion of the sequence of interest is the first half of the middle third.

4. Conclusion

Studying Bernoulli convolutions through a discrete lens sheds much new insight on the subject. Many of our algorithms would not have been discovered without combinatorial thinking—for example, the PIP algorithm is made possible by the fact that the coefficients of a polynomial can be thought of as counting the number of representations of integers as sums from a certain sequence. The discrete point of view is a very simple way to think about Bernoulli convolutions (the *duplicate, shift, add* method could be explained to a small child), but a computer has trouble computing more than a handful of Bernoulli sequences. In particular, studying Bernoulli convolutions via combinatorics has led to the discovery and development of two elegant algorithms (DEM and PIP). Using these algorithms we were able to generate the entire Bernoulli sequences at many new levels (up to 27) and also were also able to calculate individual entries of B_n at levels as high as 70.

We conjecture that $m_n = O((4/3)^n)$. Our two algorithms provide much data to support this claim.

References

[1] D. H. Bailey, J. M. Borwein, N. J. Calkin, R. Girgensohn, D. R. Luke, and V. H. Moll. *Experimental mathematics in action*. A K Peters Ltd., Wellesley, MA, 2007.

[2] N. Calkin. Counting kings, collectings coupons, and other applications of linear algebra to combinatorics. *Clemson University REU Colloquium Lecture*, 2008.

[3] P. Erdős. On a family of symmetric Bernoulli convolutions. *Amer. J. Math.*, 61:974–976, 1939.

[4] B. Jessen and A. Wintner. Distribution functions and the Riemann zeta function. *Trans. Amer. Math. Soc.*, 38(1):48–88, 1935.

[5] R. Kershner and A. Wintner. On symmetric Bernoulli Convolutions. *Amer. J. Math.*, 57(3):541–548, 1935.

[6] Y. Peres, W. Schlag, and B. Solomyak. Sixty years of Bernoulli convolutions. In *Fractal geometry and stochastics, II (Greifswald/Koserow, 1998)*, volume 46 of *Progr. Probab.*, pages 39–65. Birkhäuser, Basel, 2000.

[7] B. Solomyak. On the random series $\sum \pm\lambda^n$ (an Erdős problem). *Ann. of Math. (2)*, 142(3):611–625, 1995.

[8] H. S. Wilf. *generatingfunctionology*. A K Peters Ltd., Wellesley, MA, third edition, 2006.