scientific reports

Check for updates

OPEN

Advantages of imperfect dice rolls over coin flips for random number generation

Douglas T. Pfeffer[1], Christopher R. Allemang[2], Shashank Misra[2], William Severa[2] & J. Darby Smith[2]✉

With an eye toward neural-inspired probabilistic computation, recent work has examined the development of true random number generators via stochastic devices. Typically, these devices are operated in a two-state regime to produce a sequence of binary outcomes (i.e., coin flips). However, there is no guarantee that stochastic devices will infallibly produce fair outputs and small deviations from a uniform distribution may have unwanted complications in applications. Using mathematical analysis, we contend that opting instead for a multi-state device (i.e., a dice roll) has benefits in these unfair paradigms. To demonstrate these benefits, we apply this framework to the analysis of a tunnel diode operated in a stochastic regime. In particular, interpreting the binary stochastic output of the tunnel diode as a multi-state die roll output also sees advantages in remaining closer to uniform. Overall, our approach provides a compelling argument for mathematical driven co-design and development of novel probabilistic computing devices and hardware.

Following renewed interest in brain-inspired stochasticity, stochastic devices are increasingly being considered as sources for true random number generation (tRNG) in probabilistic computing. Neural-inspired stochasticity is based on the observation that communication in the brain is inherently stochastic with approximately 10^{15} random synapse events per second[1]. Consequently, the development and utility of stochastic devices as tRNGs on such a massive scale, mimicking synapses in the brain, is an active area of study for probabilistic computation[2–5].

Digital and in-memory computing applications typically utilize stochastic devices that can be interpreted as a stochastic fluctuation between two distinct binary states. In this manner, the measured output of devices can be interpreted as a binary coin flip. Devices under active study include memristive devices[6], p-bits[2,7], percolating networks of nanoparticles[8], as well as magnetic tunnel junctions (MTJs) and tunnel diodes (TDs)[1,9–15]. Beyond the rich literature on two-state stochastic devices, some devices and hardware can be used in a multi-state regime. Of note are certain MTJs[16] and single photon avalanche diodes[17]. Though not yet operated in a stochastic regime, the multi-state MTJs have recently found use in noise resilient neural networks[18,19] and are of interest to self-learning neuromorphic hardware[20]. Additionally, some multi-state memristors have been used to create random bitstreams[21].

Typically, the desired operation of binary coin flip devices is for fair 50/50 outcomes on 0 and 1[9,13,22]. Presumed fair bitstreams are often used with a binary encoding scheme to produce random integers or decimals. In addition to fair coin flips, changes to input pulse amplitude and external factors like temperature can bias or weight device output, favoring one outcome more than another. This behavior can be exploited to produce samples from non-uniform distributions like the exponential distribution[10]. However, devices can also produce fluctuating weights, and devices can behave differently from one another[11]. A significant, and likely impractical, amount of resources would be required for devices to repeatably produce a fair outcome. Though we can statistically quantify such fluctuations, these perturbations can have subtle and damaging effects in application use.

Since fluctuations in device operation are likely to bias otherwise fair outputs, we ask: If given a choice between a possibly unfair binary device (e.g., a coin) and a possibly unfair q-state device (e.g., a q-sided die) to produce random numbers, which will produce numbers closest to uniform? This question forms the basis of our investigation. That is, we will consider the benefits of a slightly unfair dice-like random output over the alternative of a slightly unfair binary output. Ultimately, we conclude that if you are trying to generate random numbers from an unfair device, you should choose a die over a coin. In other words, an unfair die-like device will remain closer to uniform in distribution than a similarly unfair coin. We furthermore find that interpreting unfair binary output as a die has similar uniform performance improvements.

[1]University of Tampa, Tampa, FL, USA. [2]Sandia National Laboratories, Albuquerque, NM, USA. ✉email: jsmit16@sandia.gov

We demonstrate this conclusion by first developing an analytic, non-statistical foundation to analyze distributions associated with unfair coins and dice. When similarly unfair coins and dice are compared within this framework, we find that those with more sides deviate less from uniform. Moreover, when considering dependence between coin tosses and dice rolls, we find that protection from non-uniformity scales with the number of sides. After these theoretical considerations, we apply our theory to the analysis of a real stochastic two-state device (a tunnel diode), finding that interpreting binary output as dice rolls also provides a measurable impact.

Our timely investigation provides an integral toolset for those seeking to design and analyze stochastic devices for use in probabilistic computing. For example, when we develop the analytic framework for analyzing distributions associated with unfair dice and coins, we establish easily computable upper and lower bounds for how far a given device's distribution deviates from uniform. A key utility for these bounds is that a device or hardware designer, responding to physical constraints, can take a threshold for uniform deviation and use our mathematical results to determine the number of states and their outcome probabilities to produce a distribution whose deviation from uniform meets their threshold requirements. Thus, the mathematical properties of computing with dice and coins directly informs device and hardware designers on how to meet probabilistic requirements for various applications.

Overall, our mathematical results provide a framework for the mathematical co-design of stochastic devices—a marked reversal of the typical paradigm in which new hardware developments are followed by mathematical algorithms. Our results demonstrate the immediate utility of multi-state over two-state while also highlighting a mathematical pathway to device and hardware design.

Results

At first glance, the differences between coins and dice may seem inconsequential. This is due to the seemingly intuitive nature of coins and dice. However, it turns out that dice offer tangible benefits in specific situations if the device is unfair. An unfair device gives rise to a complex probability distribution that, as we find, is strikingly counterintuitive.

In the results that follow, we specifically address slightly unfair coins and dice. If one were dealing with real devices that were tuned to be as close to fair as possible, this would be reflective of that setting. While we choose to only explicitly address this case, our results are quite general and can be applied to any settings.

The curious behavior of weighted dice

This section develops a mathematical framework to rigorously study multi-state random number generation with a goal of informing hardware construction and experimental design. Of note is that every result and tool established here is analytic in nature and requires no statistics. This is important because, while sampling statistics on coins and dice is very accessible, the sheer number of outcomes for practical use of random number generation ($2^{32} = 4,294,967,296$) is overwhelming. Therefore, an analytic, non-statistical approach is necessary.

A fair coin is one for which a random toss returns 'heads' (or, 'H') and 'tails' (or, 'T') in a 50/50 ratio. If we label 'heads' with a 1, 'tails' with a 0, and let $x_i \in \{0, 1\}$ denote the outcome of the i^{th} toss, we say that a fair coin is one for which $p_1 = \mathbb{P}[x_i = 1]$ and $p_0 = \mathbb{P}[x_i = 0]$ are both equal to 0.5. Critically, these are independent probabilities, i.e., there is no dependence on *i*. In general, we will denote $p_j = \mathbb{P}[x_i = j]$ to be the (independent) probability of returning *j* on the i^{th} toss of the coin. A sequence of *n* fair coin tosses can then be converted to a value in [0, 1] by forming $\sum_{i=1}^{n} \frac{x_i}{2^i}$. As an example, if we toss a fair coin eight times and receive the outcomes HHTHTTT, then this corresponds to the binary sequence 11010000 which gets encoded into

$$\sum_{i=1}^{8} \frac{x_i}{2^i} = \frac{1}{2^1} + \frac{1}{2^2} + \frac{0}{2^3} + \frac{1}{2^4} + \frac{0}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{0}{2^8} = 0.8125.$$

This process even allows us to encode an infinite sequence of tosses which helps build the associated cumulative distribution function (CDF). Specifically, if we put $X = \sum_{i=1}^{\infty} x_i 2^{-i}$, then $F(x) = \mathbb{P}[X \leq x]$ is the CDF obtained after tossing the coin an infinite number of times. When the given coin is fair, this process provides a uniform random variable and hence the CDF will be the line $y = x$ on [0, 1].

The entire process described thus far works analogously for a fair *q*-sided die. In such a case, we have that every $p_j = \mathbb{P}[x_i = j]$ is equal to $\frac{1}{q}$ for all $j = 1, \ldots, q$ and therefore the encoded value in [0, 1] for a sequence of *n* tosses looks like $\sum_{i=1}^{n} \frac{x_i}{q^i}$. As an example, if we have a fair 7-sided die, then the probability of returning any given side on a random toss is equal to $\frac{1}{7} \approx 0.1429$. Labeling the sides with $0, 1, \ldots, 6$, if we rolled the finite septenary sequence of outcomes 22045, we encode it to a value in [0, 1] by considering

$$\sum_{i=1}^{5} \frac{x_i}{7^i} = \frac{2}{7^1} + \frac{2}{7^2} + \frac{0}{7^3} + \frac{4}{7^4} + \frac{5}{7^5} = 0.32849\ldots$$

This process still allows us to encode an infinite sequence of tosses which helps build the associated CDF in the same way. Since the die we are using is still fair, this process again provides a uniform random variable and hence the CDF is still $y = x$. We illustrate the encoding process with a simple cartoon in Fig. 1a.

The *q*-ary encoding scheme given by $\sum_{i=1}^{\infty} x_i q^{-i}$ is chosen for two reasons. The first is that it casts the outcomes of a two-sided coin and *q*-sided die into the same space, making for reasonable comparisons. Secondly, it is the natural way to read base *q* decimals in base 10. This conversion is standard when converting random

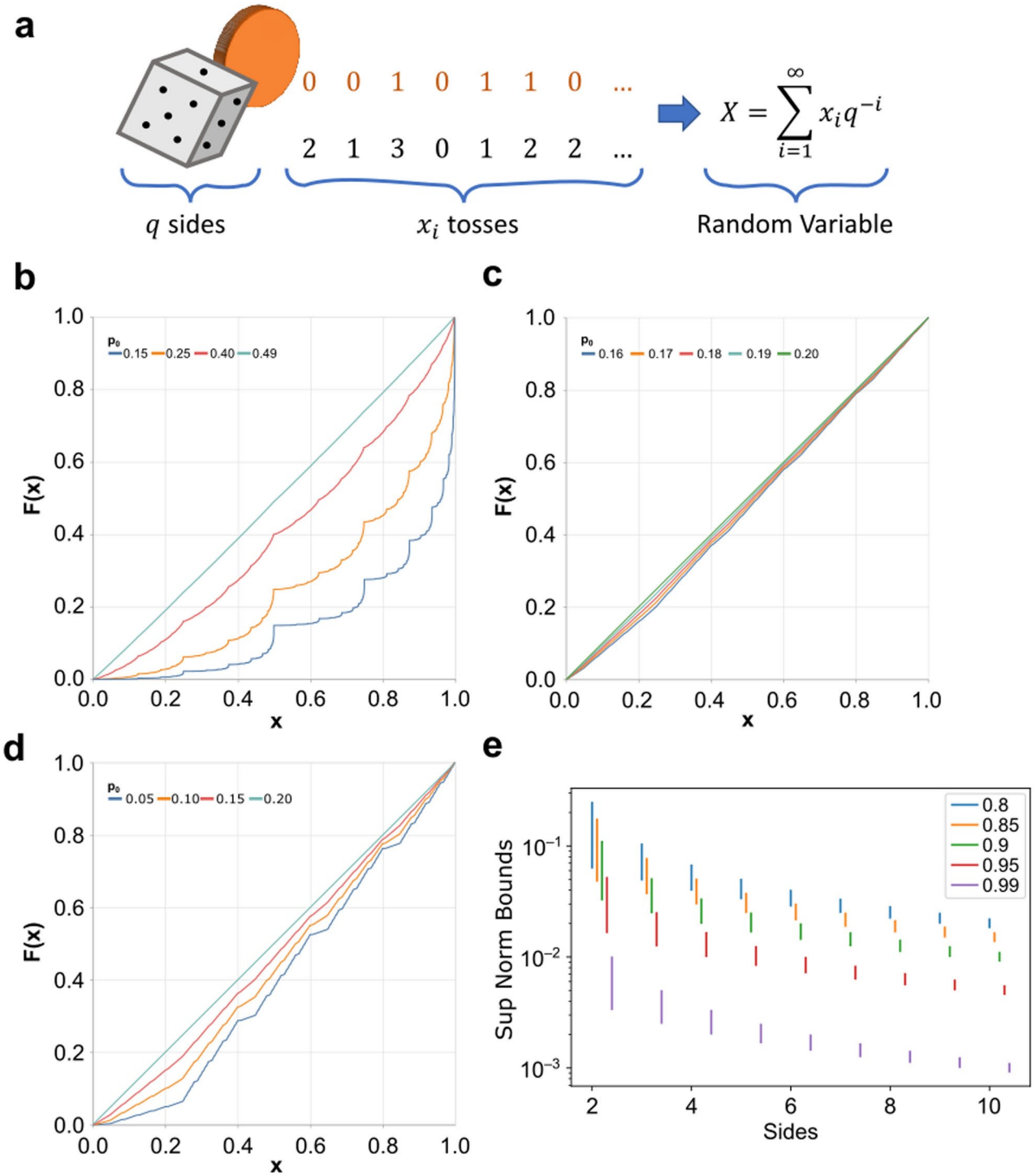

Fig. 1. (**a**) Illustration demonstrating encoding of tosses of coins or dice as a random variable. (**b–d**) The CDF of a coin (**b**) or a 5-sided die (**c, d**) is plotted for various probabilities of outcomes. In (**b**) the probability of tails is given, p_0, where the probability of heads is $p_1 = 1 - p_0$. In **c** and **d**, the probability of rolling a zero is given, p_0, where the probability of all other outcomes is $(1 - p_0)/4$. In each, the case where $p_0 = 0.20$ represents a fair die. (**e**) Analytic bounds on the deviation from uniform for various unfair coins and dice. Each color represents the ratio of the probability of zero to the corresponding fair probability of zero for the number of sides.

(binary) bitstreams to uniform samples, ubiquitous in stochastic applications (see e.g., Alaghi and Hayes[23]), and hence $\sum_{i=1}^{\infty} x_i q^{-i}$ serves as the appropriate dice analog. While this encoding assumes an infinite number of rolls, applications will involve a finite number of rolls. Indeed, most conventional random number generators typically only produce 32- or 64-bit numbers. One should view increasing numbers of finite flips or rolls as

an increasingly accurate approximation of a uniform distribution on [0, 1]. Moreover, the accuracy of this approximation increases exponentially with the number of flips or rolls, and hence it only takes a modest number of actions to achieve a reasonable approximation.

Unfortunately, real stochastic computing devices (and perhaps even real coins[24]) may not reliably behave as fair. In the case of 2-state devices (i.e., coins), Billingsley[25] (Ex. 31.1) shows that even small deviations from fair result in an uneven, fractal-like CDF. We plot the analytic, non-empirical CDF from Billingsley[25] (Ex. 31.1) for a few unfair coin weightings. Figure 1b shows that as the bias tends away from 50%, the CDF becomes more rugged. While not visible to the eye, the CDF of even slightly unfair coins (49% heads in Fig. 1b) is still fractal-like and infinitely self-similar. Formally, an unfair coin produces a *singular measure*—a probability distribution that produces a CDF that, while has obvious growth, is technically flat and has zero derivative almost everywhere you can think of. With an eye toward analyzing modern multi-state devices and hardware, Theorem 1 sees us generalize these results to q-sided dice, where a proof is found in the Supplemental Material. Unless stated otherwise, the results that follow assume that successive outcomes of flipping a given coin or rolling a given die is interpreted as a sequence of independent, identically distributed (i.i.d.) random data points.

Theorem 1 *Consider a q-sided die, where* $x_i \in \{0, \ldots, q-1\}$ *denotes the outcome of the* i^{th} *toss and* $p_j := \mathbb{P}[x_i = j]$ *for* $j \in \{0, \ldots, q-1\}$ *with each* $p_j < 1$. *Given* $X = \sum_{i=1}^{\infty} x_i q^{-i}$, *if* $F(x) = \mathbb{P}[X \leq x]$ *is the CDF obtained after tossing the die an infinite number of times, then*

(i) *F is continuous;*
(ii) *F is strictly increasing on* [0, 1] *if and only if* $p_j > 0$ *for all j;*
(iii) F' *exists almost everywhere in* [0, 1];
(iv) *If* $p_j = \frac{1}{q}$ *for all j, then* $F(x) = x$ *on* [0, 1]; *and*
(v) *If* $p_k \neq \frac{1}{q}$ *for some k, then F is singular.Additionally, F is given by the following recursion formula:*

$$F(x) = \begin{cases} p_0 F(qx) & \text{if } 0 \leq x \leq \frac{1}{q} \\ p_0 + p_1 F(qx-1) & \text{if } \frac{1}{q} \leq x \leq \frac{2}{q} \\ \quad \vdots & \quad \vdots \\ (p_0 + p_1 + \ldots + p_{q-2}) + p_{q-1} F(qx-(q-1)) & \text{if } \frac{q-1}{q} \leq x \leq 1 \end{cases}. \tag{1}$$

In mathematical terms, this theorem states that creating a random variable by tossing a die infinitely often produces a distribution defined by (1) which is singular unless the die is fair. In the context of an actual multi-state device or hardware, this theorem says that unless your device is absolutely perfect and unbiased in its generation of outcomes, its distribution will always be unintuitive and extremely difficult to work with.

While the mathematical results in this section provide a framework to rigorously study *arbitrary* multi-state random number generators, we look at some simple examples to illustrate their consequences. A convenient way to create an unfair q-sided die is to choose an unfair probability for p_0 (so that $p_0 \neq \frac{1}{q}$) and, for the remaining $q-1$ sides, distribute the remaining probability evenly so that $p_j = \frac{1-p_0}{q-1}$ for each $j = 1, \ldots, q-1$. This choice allows us to simplify numerical comparisons by only putting a control on a single quantity, the probability of zero, which is defined for all multi-state cases. This is a simplifying choice on our part that facilitates direct numerical comparisons; this simplification is unnecessary for mathematical results. In this numerical setup, when p_0 is very close to $\frac{1}{q}$, all sides of our die are close to fair, whereas when p_0 is far from $\frac{1}{q}$, the resulting die's sides are all very far from fair. Thus, when we generate unfair dice in this way, we focus on the *ratio to fair* for p_0. For example, if we consider an unfair 5-sided die with $p_0 = 0.05$, then it has a 0.25 ratio to fair since $0.05/0.20 = 0.25$.

We are now ready to see the effects of Theorem 1. In Fig. 1c, we showcase the CDF of four biased 5-sided dice whose ratio to fair range from 0.80 ($p_0 = 0.16$) to 0.95 ($p_0 = 0.19$) and in Fig. 1d, we showcase four exaggerated cases whose ratio to fair range from 0.25 ($p_0 = 0.05$) to 0.75 ($p_0 = 0.15$). As a reference, a fair 5-sided die ($p_0 = 0.20$) is plotted on each. Each of these plots is calculated directly from (1) and does not rely on sampling a die. We see that as the the die gets further away from fair, the fractal-like behavior of the CDF becomes more visible and the graph starts to deviate from the uniform $y = x$.

With an eye toward comparing the results from two different devices or hardware, we can compare the probability measures obtained from pairs of same-sided dice with differently weighted outcomes. In Theorem 2, we assert that the support for the associated distributions of differently weighted dice are disjoint (i.e., the probability measures are *mutually singular*). Again, the proof is given in the Supplemental Material. We note that this novel result was recently referred to as a 'folk theorem'—a theorem generally regarded as true, but no known proof has been published—by Cornean et al.[26], where the authors reference Section 14 of Billingsley's *Ergodic Theory and Information*[27] for a discussion on the matter.

Theorem 2 *Consider two q-sided dice with sides taken from* $Q := \{0, \ldots, q-1\}$. *Let* $x_i \in Q$ *denote the outcome of the* i^{th} *toss of one die with corresponding probabilities* $p_j := \mathbb{P}[x_i = j]$ *for* $j \in Q$, *and let* $\widetilde{x}_k$ *and* $\widetilde{p}_k$ *be defined similarly for the second die (with* $k \in Q$). *Given* $X = \sum_{i=1}^{\infty} x_i q^{-i}$ *and* $\widetilde{X} = \sum_{i=1}^{\infty} \widetilde{x}_i q^{-i}$, *put* $F(x) = \mathbb{P}[X \leq x]$ *and* $\widetilde{F}(x) = \mathbb{P}[\widetilde{X} \leq x]$ *as the respective cumulative distribution functions obtained after tossing the corresponding die an infinite number of times. If there exists an outcome* $t \in Q$ *so that* $p_t \neq \widetilde{p}_t$, *then the associated probability measures,* μ *and* $\widetilde{\mu}$, *are mutually singular.*

In mathematical terms, concluding that the associated distributions of differently weighted, same-sided dice have disjoint support says that, in the infinite limit, the two dice will never produce the same number. Hence, two distinct same-sided dice will always sample from different distributions on different outcomes. In the context of an actual multi-state device, this theorem says that unless you have two absolutely identical devices, they will always produce different outcomes from different distributions.

The above theorems discussed distributions associated with computational devices that behave like unfair coin tosses or die rolls. Now, given an imperfect device, we want to develop a way to quantify how close it is to an ideal, fair reference (i.e., how close is the associated distribution, $F(x)$, is to the uniform distribution $y = x$). Though there are many ways to make such a comparison, we opted to compute the unfair distribution's absolute deviation from fair. In mathematical terms, this is the sup-norm metric given by

$$\|x - F(x)\|_\infty := \sup\{|x - F(x)| \;\; x \in [0,1]\}, \tag{2}$$

where the *supremum* is determining the largest potential value of the set considered. We opted for this comparison metric since, from a statistical viewpoint, this maximal distance between the fair and unfair distributions is computing the well-known Kolmogorov-Smirnov (KS) statistic (see the Methods section for more details).

Explicit analytic forms for the sup-norm difference between weighted die CDFs and the uniform distribution are intractable due to the fractal properties of the weighted die CDF. Nonetheless, these objects do admit tractable bounds. With $C[0,1]$ denoting the space of continuous functions on $[0,1]$, we obtain the following (a proof of which is given in the Supplemental Material):

Theorem 3 *For $q \in \{2,3,\ldots\}$, let $T : C[0,1] \to C[0,1]$ be the operator defined by*

$$(Tf)(x) = \begin{cases} p_0 f(qx) & \text{if } 0 \le x \le \frac{1}{q} \\ p_0 + p_1 f(qx-1) & \text{if } \frac{1}{q} \le x \le \frac{2}{q} \\ \quad\vdots & \quad\vdots \\ (p_0 + p_1 + \ldots + p_{q-2}) + p_{q-1} f(qx-(q-1)) & \text{if } \frac{q-1}{q} \le x \le 1 \end{cases}. \tag{3}$$

Given the sequence of functions $(f_n)_{n=0}^{\infty}$ defined by $f_0 = x$ and $f_{n+1} = T f_n$, the distribution function F in Theorem 1, and $p_{\max} = \max\{p_0, \ldots, p_{q-1}\}$, it follows that $f_n \to F$ and

$$\left(\frac{1}{1+p_{\max}}\right)\|x - f_1(x)\|_\infty \le \|x - F(x)\|_\infty \le \left(\frac{1}{1-p_{\max}}\right)\|x - f_1(x)\|_\infty. \tag{4}$$

Thus, to estimate the quantity in (2), it suffices to understand the quantity $\|x - f_1\|_\infty$, where f_1 is a (finite) piece-wise linear function and has no fractal-like components—an object that is much more reasonable to compute. Not only does this provide bounds on how far the unfair distribution is from uniform, it also bounds the best-case KS test statistic for your device or hardware.

Concretely, this theorem provides tangible evidence to our claim that unfair dice deviate less from uniform than unfair coins. In particular, if we fix a ratio to fair for p_0 and create an unfair die by evenly distributing the remaining probability weights, then Fig. 1e illustrates that the bounds in (4), and hence the absolute deviation from uniform, decreases as the number of sides increases. We approached this analysis by focusing solely on the probability of zero out of ease. Our method of assigning probabilities for dice is meant to capture the expectation that a device or hardware would already be close to fair, either natively or with tuning. However, the tools presented here are analytic, require no statistics, and can be extended to more esoteric scenarios.

Importantly, Theorem 3 establishes a framework for the mathematical co-design of new devices and hardware. When an application requires a certain threshold for uniform deviation, the bounds given in (4) prescribe a method for meeting the uniform deviation tolerance. Specifically, since f_1 is piece-wise linear, continuous, and increasing, computing $\|x - f_1(x)\|_\infty$ reduces to checking $\max_{0<j<q}\left\{\left|\frac{j}{q} - \sum_{i=0}^{j-1} p_i\right|\right\}$, a maximum over a finite set of size $q-1$. Thus, if application context provides an allowable threshold for our deviation from uniform and a desired number of sides, then we can analytically select a set of p_i's that produce a bounding range within the required threshold. This process can then be used alongside any given physical constraints to build or tune a q-state device or hardware for which $L \le \|x - F(x)\|_\infty \le M$ for some prescribed bounds L and M.

All in all, we have provided a framework to rigorously study multi-state random number generation that will extend to device or hardware construction and experimental design. While we will revisit this in the discussion, for now we merely remark that (4) would imply that if given the choice among a collection of biased coins and dice, choosing the one with the most sides would allow one to maximize the chances that produced digits would be closest to uniform. This is, of course, assuming that no single outcome would have a probability arbitrarily close to 1, forcing the right-hand side of (4) to increase without bound.

Resilience to dependent behavior

Despite the utility and pervasiveness of the i.i.d. setting, there exist instances where dependent behavior is unavoidable. Without the application of correction schemes, stochastic devices and hardware can introduce unwanted dependencies and autocorrelation in collected bitstreams[28–30]. Unfortunately, exploring dependent behavior analytically through CDFs is intractable. Instead, we can examine empirical CDFs generated through simulation. To perform an initial investigation, we consider only a specific simplified model of stochastic device

and hardware dependency. In this simplified model, we assume there is a fixed *latching probability* that a device, coin, or die does not switch from its current state, and hence returns the same value twice. Though this represents a rather simple model of dependency, it has great utility in that with generalizations to time- or state-dependent latching probabilities, it can extend to cover any conceivable type of dependence.

Simulation of the CDFs is challenging. Even for a modest number of sides and rolls the number of possible sequences of outcomes can rapidly explode. As previously noted, there are over 4 billion possible outcomes when generating a 32-bit number via a coin with some bias. Hence, sampling to obtain a decent facsimile of the CDF could require absurd numbers of samples. While this further highlights the utility and novelty of our analytic results above, it also demonstrates that analytic tools for the dependent cases are a critical need. To determine when we have created enough samples for the scenarios we consider in this section, we employ a stopping criteria based on the arclength of the empirical CDF. Simply put, we sample until the arclength of the CDF converges. For the cases considered here, our minimum sample size was 2.75 million generated numbers. Specific details on this convergence are in the Methods and further information on the arclength of singular distributions is included in the Supplemental Material.

To observe the impact of latching on deviation from uniform, we simulated weighted coins with a p_0 ratio to fair in the range [0.80, 1.00] at latching probabilities in the range [0.00, 0.55]. We simulate these CDFs for 4-, 8-, 12-, and 16-bit numbers. Using the metric in (2), the absolute deviations from a uniform distribution are plotted as points in Fig. 2a. As expected, the coin produces less deviation from uniform as the probability of tails approaches fair (i.e. 1.00 ratio to fair). To help visualize these results, we performed a linear regression, grouping points by their latching probability. The resulting lines are plotted in Fig. 2a along with the 95% confidence interval for each linear fit. Visualized in this manner, the ratio to fair is inversely related to uniform deviation while latching probabilities appear to increase the effective baseline deviation.

In contrast, a corresponding simulation for a weighted 4- and 5-sided die are visualized in Fig. 2b,c. The choice to examine 4- and 5-sided dice is arbitrary; we merely have made a choice to demonstrate that even in the face of dependent tosses, an increased number of sides decreases the deviation from uniform. To perform an equitable comparison, we weight p_0 for the dice such that their ratio to fair is the same as the biased coins considered in Fig. 2a and, as we did in the previous section, we distribute the remaining probability evenly among the remaining sides. Additionally, we use the same latching probabilities as in the coin case. The number of rolls in each dice trial are chosen to have at least the same representational capability as the coin. Again, linear fits are performed on groups of latching probabilities. Comparing the deviations, we observe roughly an order-of-magnitude less absolute deviation across the board moving from the coin to either die, suggesting that the distributions from dice are less susceptible to these latching and biasing errors than those from coins. We also observe a more shallow slope in the relationship between the dice weighting and the absolute deviation compared to the coin. This suggests that increased numbers of sides also offers protection from the magnitude of dependency effects.

Though the results in this section focus on an academic study of a specific type of dependence, we stress that the analysis process is still immediately useful in binary device work. For one, any regularly observed two state device driven by a Poisson switching process would be completely characterized by this simple mode of dependence: either the device switched one or more times between observations, or no Poisson events occurred and the device was in the same state. The aggregate probability of these events represents a probability of flipping (and possibly getting the same outcome) or a probability of latching and remaining at the same outcome. Poisson driven states are extremely relevant in quantum computing, but also in probabilistic computing architectures of *p*-bits for Gibbs sampling[4].

Though our choice of examining 4- and 5-sided dice was arbitrary, domain wall-magnetic tunnel junctions with 5 achievable states do exist[19]. Assuming the possibility of some dependent state behavior if such devices were operated in a stochastic manner, these simulation and analysis methods provide a framework for assessing the anticipated deviation from uniform behavior. Indeed, our analytic results combined with our empirical CDF methods stand to provide an impressive toolbox for analyzing any-state devices for their suitability as uniform random number generators.

Deriving dice from tunnel diode data

Thus far we have considered dice-like behavior as an alternative to coin-like behavior. This analysis has provided insight into potential benefits afforded by existing or yet-to-be-developed multi-state devices or hardware. In this section, we will make the case that interpreting output from existing coin flip devices as dice rolls will produce distributions that are closer to uniform when the device exhibits some dependence from toss-to-toss.

We can construct a 2^b-sided die from binary bitstreams in a standard way by interpreting b bits as an integer in the range from 0 to $2^b - 1$. It is counter-intuitive that there should be any difference afforded by the die interpretation. Indeed, if coin tosses are independent, then absolutely no benefit or change in distribution can be found. In the Supplemental Material, we provide a proof of this claim. Hence, if there is a measurable change in uniformity based on the interpretation of device data, it would be an indicator that the coin flip device does not operate as independent coin tosses and instead exhibits some dependence from toss-to-toss.

We use data collected from an experimental setup featuring a stochastic TD. Roughly, a TD is pulsed with current that is near the peak tunneling current for the device while an oscilloscope in conjunction with a custom MATLAB script interpret the output voltage as a bitstream. The custom script is used to convert an analog voltage signal to a digital bitstream and mediate drift in coin flip weight. The TD circuit diagram is illustrated in Fig. 3a and additional details on the TD, its operation, and the weight mediation method can be found in the Methods. Using this setup, we collected 507 million bits from a TD.

While the employed data collection method ensures the massive bitstream remains fair on average (see Fig. 3b), it does not remove dependence. That is, if X_j is the j^{th} bit collected, $\mathbb{P}[X_j = 1 \,|\, X_0 = 1] \neq 0.5$. We

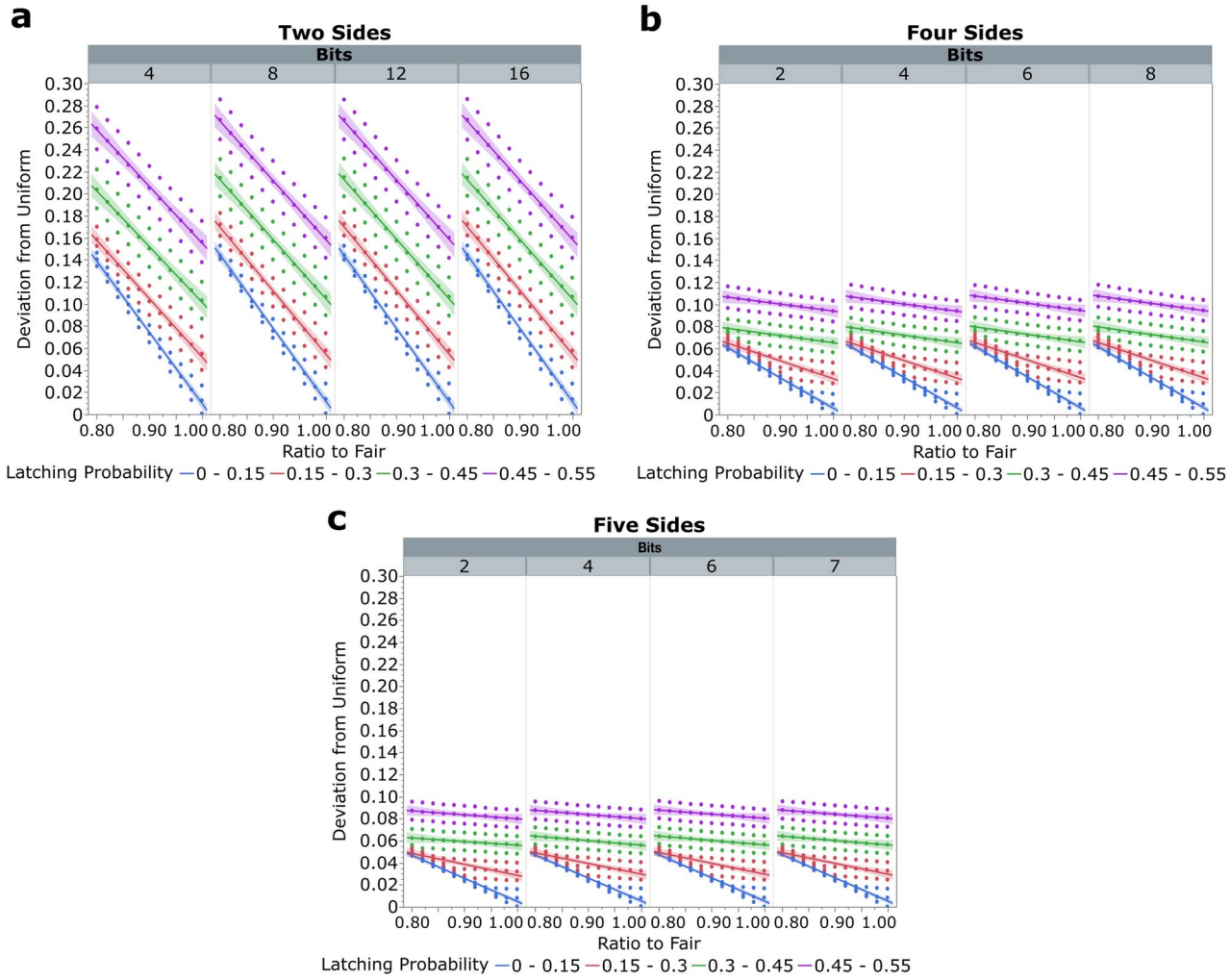

Fig. 2. Deviation from uniform for an empirical CDF under various p_0 ratio to fair and various probabilities of latching, i.e. returning the same result repeatedly. (**a**) Displays the deviation for a coin or 2-sided die, (**b**) is a 4-sided die, and (**c**) is a 5-sided die. We use the term 'bits' here to match that in (**a**), and the number of bits have been chosen to compensate for the increased number of outcomes. In each, lines show linear fits, with the shaded regions showing a 95% confidence interval for each fit's predicted values.

illustrate this dependency artifact in the blue curve of Fig. 3c for $j = 1, \ldots, 400$. This curve is produced by taking a rolling window average of seeing a 1 j bits after having seen a 1. Since this curve is not flat, there is some memory or dependence in having obtained a 1, making it more likely to return a 1 again.

To examine the effect of a dice interpretation on the TD data, we will need to interpret the TD as dice with sides equal to a power of two. Our process is illustrated in Fig. 3d. We split the 507 million bits bitstream into thirds, with each new strand consisting of 169 million bits. As the separation distance is extreme, we view each of the three strands as independent. The first third we interpret exclusively as the output from a coin. The second and third thirds we read as the most significant bit and least significant bit, respectively, of a binary number in 0, 1, 2, 3. These we treat as the outcome of a four-sided die. Even though the composite strands of bits are assumed independent, there is still dependence within each of the two strands. This translates into a dependence effect within the four outcomes, as showcased in the blue curve of Fig. 3e, where we again plot the probability of "rolling" a 1 j "rolls" after obtaining a 1.

We additionally construct an 8-sided die by taking the third, first, and second thirds as the most significant to the least significant bit, respectively. Again, we see dependence present as illustrated in the blue curve of Fig. 3f. All cases present dependence which visually appears to decay at similar rates towards the respective fair probability (0.50, 0.25, and 0.125).

Given the dependence present in the raw TD data and given our previous latching results (Fig. 2), we expect that the dice-interpreted raw TD data will be closer to uniform than the coin-interpreted data. We test this hypothesis through the use of a chi-square and a Kolmogorov-Smirnov (KS) test statistic. The chi-square hypothesis test, while being sensitive to small and large sample sizes[31], is a test that compares samples from discrete distributions to expected frequencies. Heuristically, we can think of this as a test on an empirical PDF from the data where any single observation appearing more or less often than expected influences the test

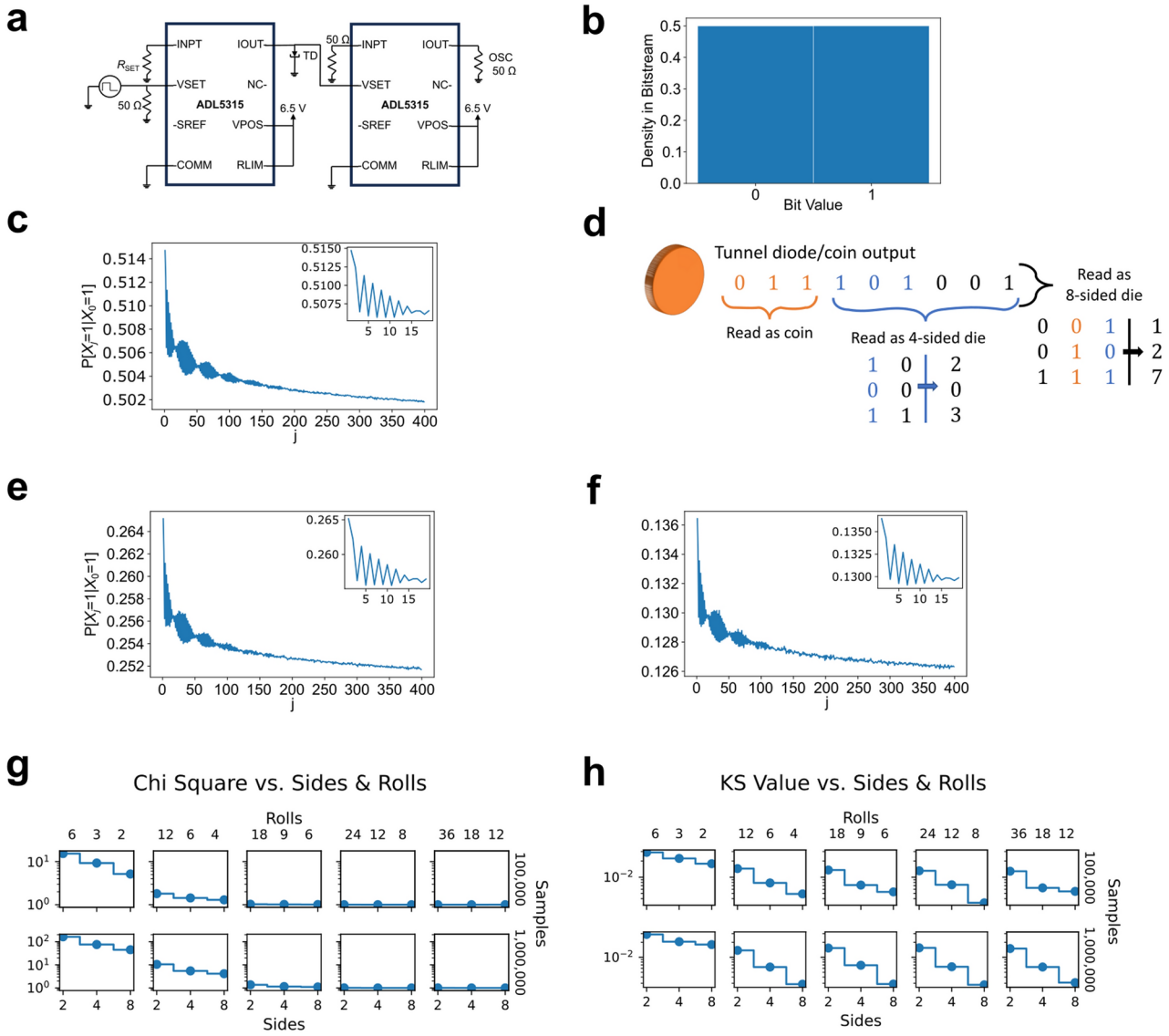

Fig. 3. (**a**) Circuit diagram for TD bitstream collection. (**b**) Histogram of TD bit density over the first 169 million bits. On average, the bitstream is fair. (**c**) Dependency plot for the TD data. The probability of seeing a 1 j flips after obtaining a 1 is shown. Inset shows $j = 1, \ldots, 20$. (**d**) Illustration of interpreting bits from a TD or coin as dice with 4 or 8 sides. (**e, f**) Dependency plots for the TD data interpreted as (**e**) a four-sided die, and (**f**) an eight-sided die. (**g**) Chi-square values per degree of freedom and (**h**) KS test statistic values for the TD data for coin, 4-, and 8-sided dice interpretations.

statistic. We specifically use a chi-square test that is normalized by the degrees of freedom. Effectively, this means we should expect uniform samples to produce normalized chi-square values near 1. Since the data is inherently discrete, the chi-square test is the most appropriate test for this data. In the large-roll limit, however, the number of possible outcomes explodes and the KS test is more appropriate.

The KS test compares the entire empirical CDF to an expected one, measuring the greatest distance between the two. This test measures the exact quantity our analytic comparison based on the sup-norm metric in (2) measures, providing a direct connection between our comparison metric and a statistical hypothesis test. It should be noted that the KS test is meant to apply to continuous data where here we examine the discrete outcomes of tossing dice a finite number of times. As such, when employing the test, we must treat the empirical CDF as defined for a continuous random variable. A smaller KS test statistic corresponds to more uniform behavior. Together, these two tests allow us to consider measures of fit for both the empirical PDF and empirical CDF of the collected data. Further details on both tests are included in the Methods.

We calculated the test statistics using the TD data interpreted as 2-, 4-, and 8-sided dice. For each, the number of rolls or flips were chosen so that the number of possible outcomes were equal to 2^6, 2^{12}, 2^{18}, 2^{24}, and 2^{32}. Results for two sample levels of encoded numbers can be found in Fig. 3g,h. In all cases, test statistics closer to uniform are achieved when increasing the number of sides in the interpretation. Notably, increasing the number of interpreted sides while holding the number of possible outcomes constant does not reduce the number of

random samples that can be produced by a bitstream of some finite length. Passing the hypothesis tests requires the test user to specify confidence levels. Though we address the notion of passing tests in the Methods and in the Supplemental Material, the biggest impact of these results should be taken to be that processing dependent coin-like data from a real device as dice will improve the uniformity of produced digits.

We conclude this section with a brief discussion on our construction of dice data from coin output. We would have liked to compare a binary device with a multi-state one, tuned to be similarly close to fair. Instead, as the tunnel diode was the stochastic device available to us, we compared a raw read of the bitstream with a dice-interpretation of the bitstream. We made an arbitrary choice to divide the bitstream and process it as described in Fig. 3d. The most important aspect of this choice was to ensure we did not accidentally encode the same information in two different ways, producing identical chi-square and KS results. For instance, suppose we took the entire bitstream, encoding it as 32 bit binary decimals. Then we took the whole bitstream again and processed it as encoded decimals from 12 rolls of an 8-sided die. The resulting numbers from each of these processes would be *exactly the same* since we would just be encoding the numbers in equivalent ways. Hence we needed effective independent output. The key aspect of dice rolls constructed from coin data that confers some immunity to the dependence in the coin data is the use of whole bitstreams which are independent from one another, even as they have some internal dependency in successive bits. While future work will introduce truly independent trials from multiple TDs, this work divided one long TD bitstream into equal parts, such that the first coinflip in each segment are separated far enough in time as to be independent. Hardware work will be needed to accurately estimate the resources required by different implementations, either from operating multiple coin flip devices in parallel, buffering bitstreams from a single coin flip device, or transducing signals from a multi-state dice roll device. The choice of partition alignment does not dramatically affect the results of the statistical tests. In the Supplemental Material, we perform the tests on two other orderings of the bitstream demonstrating no significant changes. Furthermore, we only saw a benefit from our interpretation *because* our device exhibited dependence from toss-to-toss. Removing this dependence would remove this benefit (see Supplemental Material) but may come at additional hardware cost.

Discussion

The goal of our research was to compare coin (binary) and dice (multi-state) based random number generation, determining whether one had inherent benefits over the other. In advancement of this goal, we developed an analytic, non-statistical foundation to analyze the departure from uniform of distributions associated with unfair coins and dice. Within this framework, we observed that when similarly unfair coins and dice are compared, those with more sides deviate less from uniform. When considering dependence between tosses and rolls, we found considerable protection from non-uniformity when utilizing dice. When considering data from a stochastic two-state device (a tunnel diode), we came to the conclusion that interpreting binary output as dice rolls, when dependence is present, again provided a measurable impact. All together, we have developed a theoretical basis and established empirical evidence for choosing dice over coins when departures from fair are expected. We hope this basis motivates the construction and development of stochastic multi-state devices so that these benefits may be empirically demonstrated more broadly.

To that end, our findings are well-poised to be of principal use to designers of stochastic devices and hardware (e.g., hardware random number generators). It is an unavoidable conclusion that such devices cannot consistently operate as perfectly fair machines. As such, our conclusions imply that, where possible, moving from a two-state device to a multi-state device can mitigate deficiencies in probability bias. Moreover, our results provide a framework for mathematical-driven co-design of novel stochastic devices. Given a threshold for uniform deviation, perhaps prescribed by application use-case needs, our bounds given in (4) for $\|x - F(x)\|_\infty$ provide a method of selecting device outcome probabilities, p_i. Thus, by knowing the threshold and number of sides ahead of time, physical device constraints can be utilized to analytically select a set of p_i's that produce a bounding range below the required threshold. This could likely be combined with evolutionary algorithm and reinforcement learning techniques, similar to the works of Cardwell et al. and Schuman et al.[10,12], to rapidly explore the complex trade-off space. Hence, our theoretical mathematical results stand to have a genuine, multifaceted impact on stochastic device design.

That mathematics can so strongly influence future design is significant. Often, hardware is adopted because it first dramatically impacts target areas, and then sees spillover use as algorithms adapt to new standards. GPUs are a prime example of this. While conceived for graphics processing, GPUs have enjoyed massive success with single-instruction, multiple-execution algorithms in parallel computing. Due to this success, GPUs quickly proliferated and mathematics followed with tailored algorithms. This could be thought of as an example of the "Hardware Lottery"[32]. Here we have a stark mathematical-driven co-design that upends this paradigm. The properties of computing with dice and coins can inform device designers on how to meet probabilistic requirements for applications of interest.

Methods

Distributional model

The basic distributional model makes use of pRNGs built into Python's popular `numpy` package[33]. For a q-sided die, rolls are simulated by drawing from the numbers 0 to $q-1$, with replacement, using probabilities defined as desired. Optionally, the model implements a 'latching mechanism' wherein there is a specific probability that a subsequent roll will produce the same output as the original roll. If latching does not occur, the random draw occurs as normal.

Metric approximations and convergence criteria

In all of our empirical work, we necessarily use a finite bit- (roll-, respectively) length. For finite lengths, we extend F in the obvious way $F(x) := \frac{\#\{s<x : s\in D\}}{\#D}$ where D is a set of random draws and $\#$ refers to the number of elements in a set. We approximate the sup-norm using the *Absolute Deviation from Uniform* defined by

$$\max_{s\in D} \{|F(s) - s|\}.$$

In general, approximating these metrics is challenging. When sampling coin/dice models with dependent behavior for Fig. 2, we use a simple criteria to determine the number of samples we draw. Specifically, we generate samples in batches of size B, and, at each batch, we compute the arclength using

$$\sum_{s\in \widehat{D}} \sqrt{(s_i - s_{i-1})^2 + (F(s_i) - F(s_{i-1}))^2}$$

where $\widehat{D}$ is defined as the sorted indexed set $D \cup \{0, 1\}$. After a warm-up period p batches, we halt the sampling process when the arclength across the previous w batches varies by less than an error amount ϵ. In the results presented in this paper, we use $B = 250{,}000$, $p = 10$, $w = 10$, and $\epsilon = 0.0005$. This resulted in a median 3,000,000 draws, with minimum and maximum values of 2,750,000 and 9,250,000 respectively.

Arclength is a natural stopping criteria for sampling the CDF. Analytically, any singular distribution on the unit interval necessarily will have an arclength of 2. Hence, the CDF from rolling an unfair die infinitely often will have an arclength of 2. The finite rolls along the way will have some finite arclength between $\sqrt{2}$ and 2. More details on arclength for singular distributions are included in the Supplemental Material.

Tunnel diode experimental setup and collection

Operating a tunnel diode as a coin flip device requires a current pulse near the peak tunnel current[22]. Here, we use the circuit illustrated in Fig. 3a to operate a tunnel diode as a coin flip device and generate a bitstream. The left current mirror (ADL5315) converts a square wave voltage at VSET, generated by a Keysight 33500B Series Waveform Generator, into a current pulse applied to the tunnel diode (TD) at IOUT. The voltage applied at VSET is mirrored at INPT and the current at INPT, set by R_{SET}, is mirrored at IOUT. Thus, both the amplitude of the square wave voltage and the resistance of R_{SET} set the current amplitude. The right ADL5315 conditions the voltage across the TD to be measured by an oscilloscope (OSC), an Agilent Technologies DSO-X 3034A, at IOUT terminated in 50 Ω. The DC bias applied to VPOS is supplied by an Agilent Technologies E3631A. A custom MATLAB (The MathWorks, Inc.) script is used to interface with the Keysight 33500B and Agilent DSO-X 3034A to generate the bitstream. The frequency of the square wave voltage is 100 kHz with a duty cycle of 10%. The oscilloscope window is set to 0.1 sec resulting in 10k pulses per oscilloscope window. The analog pulses are analyzed using the Signal Processing Toolbox to determine whether a 0 bit or 1 bit was generated. If the weight of the oscilloscope window is $> 51\%$ or $< 49\%$ the data is rejected. Furthermore, if the weight of the oscilloscope window is $> 62.2\%$ or $< 39.2\%$, the amplitude of the square wave voltage is stepped down or up, respectively, by the minimum step size. This process is repeated until enough oscilloscope windows are accepted to have approximately 500 million bits.

Statistical tests

The Chi-square test

We utilized a chi-square test statistic to measure the deviation of encoded dice data from uniform. The chi-square test is typically used on discrete or categorical data. In the general sense, a prescribed or expected frequency of the i^{th} category is given, E_i. Data is collected and the frequency of the i^{th} category is recorded, X_i. For N possible categories, the chi-square test statistic is given as

$$\chi^2 = \sum_{i=1}^{N} \frac{(X_i - E_i)^2}{E_i}. \tag{5}$$

If we assume that the data are distributed as expected, then the test statistic χ^2 is chi-square distributed with $N - 1$ degrees of freedom. This assumption is called the null hypothesis. The null hypothesis is rejected whenever we receive a χ^2 value that would be exceptionally rare under the chi-square distribution, where 'exceptionally rare' is according to the discretion of the user.

We employ a variant of the chi-square test. First, since we are comparing to uniform, $E_i = M/N$ for all i, where M is the total number of samples taken. If the null hypothesis is true, making X_i sample frequencies from uniform draws, and if N is large, then the numerator in (5) represents a sum of squared standard errors. Hence, the test statistic χ^2 is a sum of N standard normal squared variates. By definition, this is chi-square distributed with N degrees of freedom. For large N, this is approximately normally distributed with mean N and variance $2N$. It follows, then, that χ^2/N is normally distributed with mean 1 and variance $2/N$.

We therefore perform our chi-square hypothesis test on the statistic χ^2/N. Provided the null hypothesis is true, this statistic should be about 1. We select an approximate 95% confidence interval and will accept statistic values within two standard deviations about 1. The rejection region for the null hypothesis is then any value falling outside of this confidence interval. Compactly, we will say our data fails the hypothesis test if

$$\left|\frac{\chi^2}{N} - 1\right| > 2\sqrt{\frac{2}{N}} \tag{6}$$

The Kolmogorov-Smirnov hypothesis test
The Kolmogorov-Smirnov test is used to compare an expected CDF with an empirical one. Explicitly, let f_M represent the empirical CDF obtained after M samples and let F represent the given expected distribution. The KS test statistic is defined as

$$K = \|f_M(x) - F(x)\|_\infty. \tag{7}$$

The null hypothesis for the test is that the collected data came from a random process with the given or expected CDF. When the null hypothesis is true, the test statistic converges to zero with increasing M. The rate of this convergence, and hence the calculation of the p-value, is related to the expected maximum absolute value obtained along the path of a Brownian bridge—a white noise process on the unit time interval, constrained to start and end at zero. The KS test heuristically forms an expectation on sampling, namely that the observed CDF should remain within an envelope around the true CDF where the size of the envelope depends on the number of samples obtained. For more information on the KS test, see Berger et al.[34].

For our tests, we utilize the `scipy.stats` function `kstest`[35]. This built-in function provides the p-value of the test statistic. A p-value is the probability of obtaining the observed data under the null hypothesis. The null hypothesis is then rejected if the p-value is too small, where 'too small' is according to the discretion of the user. We chose to reject the null hypothesis if the p-value was smaller than 0.05.

Data availability
Data from the tunnel diode will be made available upon reasonable request to the authors. Please contact the corresponding author, J. Darby Smith (jsmit16@sandia.gov) to obtain the tunnel diode data.

Code availability
The authors will open-source the basic distributional (latching) model and the scripts used to generate and evaluate the rolls upon acceptance of this manuscript, pending organizational approval.

Received: 19 December 2024; Accepted: 28 March 2025
Published online: 07 April 2025

References
1. Misra, S. et al. Probabilistic neural computing with stochastic devices. *Adv. Mater.* **35**(37), 2204569 (2022).
2. Kaiser, J. & Datta, S. Probabilistic computing with p-bits. *Appl. Phys. Lett.* **119**, 150503 (2021).
3. Chowdhury, S., Camsari, K. Y. & Datta, S. Accelerated quantum Monte Carlo with probabilistic computers. *Commun. Phys.* **6**, 85 (2023).
4. Chowdhury, S. *et al.* A full-stack view of probabilistic computing with p-bits: Devices, architectures and algorithms. *IEEE J. Explor. Solid-State Comput. Devices Circuits* (2023).
5. Aadit, N. A. et al. Massively parallel probabilistic computing with sparse Ising machines. *Nat. Electron.* **5**, 460–468 (2022).
6. Gaba, S., Sheridan, P., Zhou, J., Choi, S. & Lu, W. Stochastic memristive devices for computing and neuromorphic applications. *Nanoscale* **5**, 5872–5878 (2013).
7. Camsari, K. Y., Faria, R., Sutton, B. M. & Datta, S. Stochastic p-bits for invertible logic. *Phys. Rev. X* **7**, 031014 (2017).
8. Acharya, S. K. et al. Stochastic spiking behavior in neuromorphic networks enables true random number generation. *ACS Appl. Mater. Interfaces* **13**, 52861–52870 (2021).
9. Liu, S. et al. Random bitstream generation using voltage-controlled magnetic anisotropy and spin orbit torque magnetic tunnel junctions. *IEEE J. Explor. Solid-State Comput. Dev. Circuits* **8**, 194–202 (2022).
10. Cardwell, S. G. *et al.* Device codesign using reinforcement learning. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5 (IEEE, 2024).
11. Maicke, A. et al. Magnetic tunnel junction random number generators applied to dynamically tuned probability trees driven by spin orbit torque. *Nanotechnology* **35**, 275204 (2024).
12. Schuman, C. *et al.* Device codesign using reinforcement learning and evolutionary optimization. In *Machine Learning with New Compute Paradigms* (2024).
13. Rehm, L. *et al.* Stochastic magnetic actuated random transducer devices based on perpendicular magnetic tunnel junctions. arXiv:2209.01480 (2022).
14. Dubovskiy, A. *et al.* One trillion true random bits generated with a field programmable gate array actuated magnetic tunnel junction. arXiv:2404.14307 (2024).
15. Rehm, L. et al. Temperature-resilient random number generation with stochastic actuated magnetic tunnel junction devices. *Appl. Phys. Lett.* **124**, 052401 (2024).
16. Safranski, C. et al. Reliable sub-nanosecond switching in magnetic tunnel junctions for MRAM applications. *IEEE Trans. Electron. Dev.* **69**, 7180–7183 (2022).
17. Whitehead, W., Nelson, Z., Camsari, K. Y. & Theogarajan, L. CMOS-compatible Ising and Potts annealing using single-photon avalanche diodes. *Nat. Electron.* **6**, 1009–1019 (2023).
18. Leonard, T. et al. Shape-dependent multi-weight magnetic artificial synapses for neuromorphic computing. *Adv. Electron. Mater.* **8**, 2200563 (2022).
19. Leonard, T., Liu, S., Jin, H. & Incorvia, J. A. C. Stochastic domain wall-magnetic tunnel junction artificial neurons for noise-resilient spiking neural networks. *Appl. Phys. Lett.* **122**, 262406 (2023).
20. van Weerdenburg, W. M. *et al.* Stochastic syncing in sinusoidally driven atomic orbital memory. arXiv:2309.17000 (2023).
21. Vasileiadis, N., Dimitrakis, P., Ntinas, V. & Sirakoulis, G. C. True random number generator based on multi-state silicon nitride memristor entropy sources combination. In *2021 International Conference on Electronics, Information, and Communication (ICEIC)*, 1–4 (IEEE, 2021).
22. Bernardo-Gavito, R. et al. Extracting random numbers from quantum tunnelling through a single diode. *Sci. Rep.* **7**, 17879 (2017).
23. Alaghi, A. & Hayes, J. P. Computing with randomness. *IEEE Spectr.* **55**, 46–51 (2018).

24. Bartoš, F. *et al.* Fair coins tend to land on the same side they started: Evidence from 350,757 flips (2023). arXiv:2310.04153.
25. Billingsley, P. *Probability and Measure. Wiley Series in Probability and Statistics* (John Wiley & Sons, Inc., 2012).
26. Cornean, H. D., Herbst, I. W., Møller, J., Sørensen, K. S. & Støttrup, B. B. Characterization of random variables with stationary digits. *J. Appl. Probab.* **59**, 931–947 (2022).
27. Billingsley, P. *Ergodic Theory and Information* (Wiley, New York, 1965).
28. Liu, Y., Wang, Z., Li, Z., Wang, X. & Zhao, W. A spin orbit torque based true random number generator with real-time optimization. In *2018 IEEE 18th International Conference on Nanotechnology (IEEE-NANO)*, 1–4 (IEEE, 2018).
29. Daniels, M. W., Madhavan, A., Talatchian, P., Mizrahi, A. & Stiles, M. D. Energy-efficient stochastic computing with superparamagnetic tunnel junctions. *Phys. Rev. Appl.* **13**, 034016 (2020).
30. Fu, Z. et al. An overview of spintronic true random number generator. *Front. Phys.* **9**, 638207 (2021).
31. Bergh, D. Chi-squared test of fit and sample size-a comparison between a random sample approach and a chi-square value adjustment method. *J. Appl. Meas.* **16**, 204–217 (2015).
32. Hooker, S. The hardware lottery. *Commun. ACM* **64**, 58–65 (2021).
33. Harris, C. R. et al. Array programming with numpy. *Nature* **585**, 357–362 (2020).
34. Berger, V. W. & Zhou, Y. Kolmogorov-smirnov test: Overview. In *Wiley Statsref: Statistics Reference Online* (2014).
35. Virtanen, P. et al. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nat. Methods* **17**, 261–272 (2020).

Acknowledgements

The authors would like to acknowledge Suma G. Cardwell and James B. Aimone for reviewing this work and for their invaluable comments. The authors acknowledge support from the DOE Office of Science (ASCR/BES) Microelectronics Co-Design project COINFLIPS. This work was performed, in part, at the Center for Integrated Nanotechnologies, an Office of Science User Facility operated for the U.S. DOE Office of Science. This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

Author contributions

DTP, JDS provided the mathematical analysis in the manuscript. WMS provided the simulation and computation results and visualization. CRA, SM collected the TD data and accompanying text. JDS analyzed the TD dataset. All authors contributed to the manuscript.

Declarations

Competing Interests

The authors claim no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-96492-8.

Correspondence and requests for materials should be addressed to J.D.S.