# Supplemental Notes for Bernoulli Factory Algorithms

This version of the document is dated 2021-03-06.

**Peter Occil**

## 1 General Factory Functions

As a reminder, the *Bernoulli factory problem* is: Given a coin with unknown probability of heads of $\lambda$, sample the probability $f(\lambda)$.

The algorithms for **general factory functions**, described in my main article on Bernoulli factory algorithms, work by building randomized upper and lower bounds for a function $f(\lambda)$, based on flips of the input coin. Roughly speaking, the algorithms work as follows:

1. Generate a uniform(0, 1) random number, $U$.
2. Flip the input coin, then build an upper and lower bound for $f(\lambda)$, based on the outcomes of the flips so far.
3. If $U$ is less than or equal to the lower bound, return 1. If $U$ is greater than the upper bound, return 0. Otherwise, go to step 2.

These randomized upper and lower bounds come from two sequences of polynomials: one approaches the function $f(\lambda)$ from above, the other from below, where $f$ is a continuous function that maps the interval (0, 1) to (0, 1). (These two sequences form a so-called *approximation scheme* for $f$.) One requirement for these algorithms to work correctly is called the *consistency requirement*:

*The difference—*

- *between the degree-(n−1) upper polynomial and the degree-n upper polynomial, and*
- *between the degree-n lower polynomial and the degree-(n−1) lower polynomial,*

*must have non-negative coefficients, once the polynomials are elevated to degree n and rewritten in Bernstein form.*

The consistency requirement ensures that the upper polynomials "decrease" and the lower polynomials "increase". Unfortunately, the reverse is not true in general; even if the upper polynomials "decrease" and the lower polynomials "increase" to $f$, this does not mean that the scheme will ensure consistency. Examples of this fact are shown in the section "**Schemes That Don't Work**" later in this document.

In this document, **fbelow**($n$, $k$) and **fabove**($n$, $k$) mean the $k^{\text{th}}$ coefficient for the lower or upper degree-$n$ polynomial in Bernstein form, respectively, where $k$ is an integer in the interval [0, $n$].

### 1.1 Approximation Schemes

A *factory function* $f(\lambda)$ is a function for which the Bernoulli factory problem can be solved (see "**About Bernoulli Factories**"). The following are approximation schemes for $f$ if it belongs to one of certain classes of factory functions. It would be helpful to plot the desired function *f* using a computer algebra system to see if it belongs to any of the classes of functions described below.

**Concave functions.** If *f* is known to be *concave* in the interval [0, 1] (which roughly means that its rate of growth there never goes up), then **fbelow**($n$, $k$) can equal $f(k/n)$, thanks to Jensen's inequality.

**Convex functions.** If *f* is known to be *convex* in the interval [0, 1] (which roughly means that its rate of growth there never goes down), then **fabove**($n$, $k$) can equal $f(k/n)$, thanks to Jensen's inequality. One example is $f(\lambda) = \exp(-\lambda/4)$.

**Twice differentiable functions.** The following method, proved in the appendix, implements **fabove** and **fbelow** if $f(\lambda)$—

- has a defined "slope-of-slope" everywhere in [0, 1] (that is, the function is *twice differentiable* there), and
- in the interval [0, 1]—
  - has a minimum of greater than 0 and a maximum of less than 1, or
  - is convex and has a minimum of greater than 0, or
  - is concave and has a maximum of less than 1.

Let $m$ be an upper bound of the highest value of abs($f''(x)$) for any $x$ in [0, 1], where $f''$ is the "slope-of-slope" function of *f*. Then for all $n$ that are powers of 2:

- **fbelow**($n$, $k$) = $f(k/n)$ if *f* is concave; otherwise, min(**fbelow**(4,0), **fbelow**(4,1), ..., **fbelow**(4,4)) if $n < 4$; otherwise, $f(k/n) - m/(7*n)$.
- **fabove**($n$, $k$) = $f(k/n)$ if *f* is convex; otherwise, max(**fabove**(4,0), **fabove**(4,1), ..., **fabove**(4,4)) if $n < 4$; otherwise, $f(k/n) + m/(7*n)$.

My **GitHub repository** includes SymPy code for a method, `c2params`, to calculate the necessary values for $m$ and the bounds of these polynomials, given *f*.

> **Note:** For this method, the "slope-of-slope" function need not be discontinuous (Y. Peres, pers. comm., 2021).

**Hölder and Lipschitz continuous functions.** I have found a way to extend the results of Nacu and Peres (2005)**(1)** to certain functions with a slope that tends to a vertical slope. The following scheme, proved in the appendix, implements **fabove** and **fbelow** if $f(\lambda)$—

- is $\alpha$-***Hölder continuous*** in [0, 1], meaning its vertical slopes there, if any, are no "steeper" than that of $m*\lambda^{\alpha}$, for some number $m$ greater than 0 (the Hölder constant) and for some $\alpha$ in the interval (0, 1], and
- in the interval [0, 1]—
  - has a minimum of greater than 0 and a maximum of less than 1, or
  - is convex and has a minimum of greater than 0, or
  - is concave and has a maximum of less than 1.

If *f* in [0, 1] has a defined slope at all but a countable number of points, and does not tend to a vertical slope anywhere, then *f* is ***Lipschitz continuous***, $\alpha$ is 1, and $m$ is the highest absolute value of the function's "slope". Otherwise, finding $m$ for a given $\alpha$ is non-trivial and it requires knowing where *f*'s vertical slopes are, among other things.**(2)** But assuming $m$ and $\alpha$ are known, then for all $n$ that are powers of 2:

- $\delta(n) = m*(2/7)^{\alpha/2}/((2^{\alpha/2}-1)*n^{\alpha/2})$.
- **fbelow**($n$, $k$) = $f(k/n)$ if *f* is concave; otherwise, min(**fbelow**(4,0), **fbelow**(4,1), ...,

**fbelow**(4,4)) if $n$ < 4; otherwise, $f(k/n) - \delta(n)$.
- **fabove**($n$, $k$) = $f(k/n)$ if $f$ is convex; otherwise, max(**fabove**(4,0), **fabove**(4,1), ..., **fabove**(4,4)) if $n$ < 4; otherwise, $f(k/n) + \delta(n)$.

  **Note:** Some functions $f$ are not $\alpha$-Hölder continuous for any $\alpha$ greater than 0. These functions have an exponentially steep slope and can't be handled by this method. One example is $f(\lambda)$ = 1/10 if $\lambda$ is 0 and $-1/(2*\ln(\lambda/2)) + 1/10$ otherwise, which has an exponentially steep slope near 0.

**Certain functions that equal 0 at 0.** This approach involves transforming the function $f$ so that it no longer equals 0 at the point 0. This can be done by dividing $f$ by a function ($h(\lambda)$) that "dominates" $f$ at every point in the interval [0, 1]. Unlike for the original function, there might be an approximation scheme described earlier in this section for the transformed function.

More specifically, $h(\lambda)$ must meet the following requirements:

- $h(\lambda)$ is continuous on the closed interval [0, 1].
- $h(0) = 0$. (This is required to ensure correctness in case $\lambda$ is 0.)
- $1 \ge h(1) \ge f(1) \ge 0$.
- $1 > h(\lambda) > f(\lambda) > 0$ for all $\lambda$ in the open interval (0, 1).
- If $f(1) = 0$, then $h(1) = 0$. (This is required to ensure correctness in case $\lambda$ is 1.)

Also, $h$ should be a function with a simple Bernoulli factory algorithm. For example, $h$ can be a polynomial in Bernstein form of degree $n$ whose $n$ plus one coefficients are [0, 1, 1, ..., 1]. This polynomial is easy to simulate using the algorithms from the section "**Certain Polynomials**".

The algorithm is now described.

Let $g(\lambda) = \lim_{\nu\to\lambda} f(\nu)/h(\nu)$ (in other words, the value that $f(\nu)/h(\nu)$ approaches as $\nu$ approaches $\lambda$.) If—

- $f(0) = 0$ and $f(1) < 1$, and
- $g(\lambda)$ is continuous on [0, 1] and belongs in one of the classes of functions given earlier,

then $f$ can be simulated using the following algorithm:

1. Run a Bernoulli factory algorithm for $h$. If the call returns 0, return 0. (For example, if $h(\lambda) = \lambda$, then this step amounts to the following: "Flip the input coin. If it returns 0, return 0.")
2. Run one of the **general factory function algorithms** for $g$(.), and return the result of that algorithm. This involves building polynomials that converge to $g$(.), as described earlier in this section. (Alternatively, if $g$ is easy to simulate, instead run another Bernoulli factory algorithm for $g$ and return the result of that algorithm.)

   **Notes:**

   1. It may happen that $g(0) = 0$. In this case, step 2 of this algorithm can involve running this algorithm again, but with new $g$ and $h$ functions that are found based on the current $g$ function. See the second example below.

   2. If—

      - $f$ is monotonically increasing,
      - $h(\lambda) = \lambda$, and

    - $f'(\lambda)$, the "slope" function of $f$, is continuous on [0, 1], maps (0, 1) to (0, 1), and belongs in one of the classes of functions given earlier,

  then step 2 can be implemented by taking $g$ as $f'$, except: (A) a uniform(0, 1) random number $u$ is generated at the start of the step; (B) instead of flipping the input coin as normal during that step, a different coin is flipped that does the following: "Flip the input coin, then **sample from the number *u***. Return 1 if both the call and the flip return 1, and return 0 otherwise."
  This is the "**integral method**" of Flajolet et al. (2010)**(3)** (the modified step 2 simulates $1/\lambda$ times the *integral* of $f$.).

**Examples:**

1. If $f(\lambda) = (\sinh(\lambda)+\cosh(\lambda)-1)/4$, then $f$ is bounded from above by $h(\lambda) = \lambda$, so $g(\lambda)$ is 1/4 if $\lambda = 0$, and $(\exp(\lambda) - 1)/(4*\lambda)$ otherwise. The following SymPy code computes this example: `fx = (sinh(x)+cosh(x)-1)/4; h = x; pprint(Piecewise((limit(fx/h,x,0), Eq(x,0)), ((fx/h).simplify(), True)))`.

2. If $f(\lambda) = \cosh(\lambda) - 1$, then $f$ is bounded from above by $h(\lambda) = \lambda$, so $g(\lambda)$ is 0 if $\lambda = 0$, and $(\cosh(\lambda)-1)/\lambda$ otherwise. Since $g(0) = 0$, we find new functions $g$ and $h$ based on the current $g$. The current $g$ is bounded from above by $H(\lambda) = \lambda*3*(2-\lambda)/5$ (a degree-2 polynomial that in Bernstein form has coefficients [0, 6/10, 6/10]), so $G(\lambda) = 5/12$ if $\lambda = 0$, and $-(5*\cosh(\lambda) - 5)/(3*\lambda^2*(\lambda-2))$ otherwise. $G$ is bounded away from 0 and 1, so we have the following algorithm:

    1. (Simulate $h$.) Flip the input coin. If it returns 0, return 0.
    2. (Simulate $H$.) Flip the input coin twice. If neither flip returns 1, return 0. Otherwise, with probability 4/10 (that is, 1 minus 6/10), return 0.
    3. Run a Bernoulli factory algorithm for $G$ (which involves building polynomials that converge to $G$, noticing that $G$ is twice differentiable) and return the result of that algorithm.

**Certain functions that equal 0 at 0 and 1 at 1.** Let $f$, $g$, and $h$ be functions as defined earlier, except that $f(0) = 0$ and $f(1) = 1$. Define the following additional functions:

- $\varphi(\lambda)$ is a function that meets the following requirements:
    - $\varphi(\lambda)$ is continuous on the closed interval [0, 1].
    - $\varphi(0) = 0$ and $\varphi(1) = 1$.
    - $1 > f(\lambda) > \varphi(\lambda) > 0$ for all $\lambda$ in the open interval (0, 1).
- $q(\lambda) = \lim_{\nu\to\lambda} \varphi(\nu)/h(\nu)$.
- $r(\lambda) = \lim_{\nu\to\lambda} (1-g(\nu))/(1-q(\nu))$.

Roughly speaking, $\varphi$ is a function that bounds $f$ from below, just as $h$ bounds $f$ from above. $\varphi$ should be a function with a simple Bernoulli factory algorithm, such as a polynomial in Bernstein form. If both $\varphi$ and $h$ are polynomials of the same degree, $q$ will be a rational function with a relatively simple Bernoulli factory algorithm (see "**Certain Rational Functions**").

Now, if $r(\lambda)$ is continuous on [0, 1] and belongs in one of the classes of functions given earlier, then $f$ can be simulated using the following algorithm:

1. Run a Bernoulli factory algorithm for $h$. If the call returns 0, return 0. (For example, if $h(\lambda) = \lambda$, then this step amounts to the following: "Flip the input coin. If it returns 0, return 0.")

2. Run a Bernoulli factory algorithm for *q*(.). If the call returns 1, return 1.
3. Run one of the **general factory function algorithms** for *r*(.). If the call returns 0, return 1. Otherwise, return 0. This step involves building polynomials that converge to *r*(.), as described earlier in this section.

   **Example:** If $f(\lambda)$ = (1−exp($\lambda$))/(1−exp(1)), then *f* is bounded from above by $h(\lambda) = \lambda$, and from below by $\varphi(\lambda) = \lambda^2$. As a result, $q(\lambda) = \lambda$, and $r(\lambda)$ = (2 − exp(1))/(1 − exp(1)) if $\lambda$ = 0; 1/(exp(1)−1) if $\lambda$ = 1; and (−$\lambda$*(1 − exp(1)) − exp($\lambda$) + 1)/($\lambda$*(1 − exp(1))*($\lambda$ − 1)) otherwise. This can be computed using the following SymPy code: `fx=(1-exp(x))/(1-exp(1)); h=x; phi=x**2; q=(phi/h); r=(1-fx/h)/(1-q); r=Piecewise((limit(r, x, 0), Eq(x,0)), (limit(r,x,1),Eq(x,1)), (r,True)).simplify(); pprint(r)`.

**Other functions that equal 0 or 1 at the endpoints 0 and/or 1.** If *f* does not fully admit an approximation scheme under the convex, concave, twice differentiable, and Hölder classes:

| If *f*(0) = | And *f*(1) = | Method |
|---|---|---|
| > 0 and < 1 | 1 | Use the algorithm for **certain functions that equal 0 at 0**, but with $f(\lambda) = 1 - f(1-\lambda)$.<br>*Inverted coin*: Instead of the usual input coin, use a coin that does the following: "Flip the input coin and return 1 minus the result."<br>*Inverted result:* If the overall algorithm would return 0, it returns 1 instead, and vice versa. |
| > 0 and < 1 | 0 | Algorithm for **certain functions that equal 0 at 0**, but with $f(\lambda) = f(1-\lambda)$. (For example, cosh($\lambda$)−1 becomes cosh(1−$\lambda$)−1.)<br>Inverted coin. |
| 1 | 0 | Algorithm for **certain functions that equal 0 at 0 and 1 at 1**, but with $f(\lambda) = 1-f(\lambda)$.<br>Inverted result. |
| 1 | > 0 and ≤ 1 | Algorithm for **certain functions that equal 0 at 0**, but with $f(\lambda) = 1-f(\lambda)$.<br>Inverted result. |

**Specific functions.** My **GitHub repository** includes SymPy code for a method, `approxscheme2`, to build a polynomial approximation scheme for certain factory functions.

**Open questions.**

- Are there factory functions used in practice that are not covered by the approximation schemes in this section?
- Are there specific functions (especially those in practical use) for which there are practical and faster formulas for building polynomials that converge to those functions (besides those I list in this section or the main **Bernoulli Factory Algorithms** article)?

## 1.2 Schemes That Don't Work

In the academic literature (papers and books), there are many approximation schemes that involve polynomials that converge from above and below to a function. Unfortunately, most of them cannot be used as is to simulate a function *f* in the Bernoulli Factory setting, because they don't ensure the consistency requirement described earlier.

The following are approximation schemes with counterexamples to consistency.

**First scheme.** In this scheme (Powell 1981)**[4]**, let $f$ be a $C^2$ continuous function (a function with continuous "slope" and "slope-of-slope" functions) in [0, 1]. Then for all $n \geq 1$:

- **fabove**($n$, $k$) = $f(k/n) + M / (8*n)$.
- **fbelow**($n$, $k$) = $f(k/n) - M / (8*n)$.

Where $M$ is an upper bound of the maximum absolute value of $f$'s slope-of-slope function (second derivative), and where $k$ is an integer in the interval [0, $n$].

The counterexample involves the $C^2$ continuous function $g(\lambda) = \sin(\pi*\lambda)/4 + 1/2$.

For $g$, the coefficients for—

- the degree-2 upper polynomial in Bernstein form (**fabove**(5, $k$)) are [0.6542..., 0.9042..., 0.6542...], and
- the degree-4 upper polynomial in Bernstein form (**fabove**(6, $k$)) are [0.5771..., 0.7538..., 0.8271..., 0.7538..., 0.5771...].

The degree-2 polynomial lies above the degree-4 polynomial everywhere in [0, 1]. However, to ensure consistency, the degree-2 polynomial, once elevated to degree 4 and rewritten in Bernstein form, must have coefficients that are greater than or equal to those of the degree-4 polynomial.

- Once elevated to degree 4, the degree-2 polynomial's coefficients are [0.6542..., 0.7792..., 0.8208..., 0.7792..., 0.6542...].

As we can see, the elevated polynomial's coefficient 0.8208... is less than the corresponding coefficient 0.8271... for the degree-4 polynomial.

*The rest of this section will note counterexamples involving other functions and schemes, without demonstrating them in detail.*

**Second scheme.** In this scheme, let $f$ be a Lipschitz continuous function in [0, 1] (that is, a continuous function in [0, 1] that has a defined slope at all but a countable number of points, and does not tend to a vertical slope anywhere). Then for all $n \geq 2$:

- **fabove**($n$, $k$) = $f(k/n) + L*(5/4) / \text{sqrt}(n)$.
- **fbelow**($n$, $k$) = $f(k/n) - L*(5/4) / \text{sqrt}(n)$.

Where L is the maximum absolute "slope", also known as the Lipschitz constant, and (5/4) is the so-called Popoviciu constant, and where $k$ is an integer in the interval [0, $n$] (Lorentz 1986)**[5]**, (Popoviciu 1935)**[6]**.

There are two counterexamples here; together they show that this scheme can fail to ensure consistency, even if the set of functions is restricted to "smooth" functions (not just Lipschitz continuous functions):

1. The function $f(\lambda) = \min(\lambda, 1-\lambda)/2$ is Lipschitz continuous with Lipschitz constant 1/2. (In addition, $f$ has a kink at 1/2, so that it's not differentiable, but this is not essential for the counterexample.) The counterexample involves the degree-5 and degree-6 upper polynomials (**fabove**(5, $k$) and **fabove**(6, $k$)).
2. The function $f = \sin(4*\pi*\lambda)/4 + 1/2$, a "smooth" function with Lipschitz constant $\pi$. The counterexample involves the degree-3 and degree-4 lower polynomials (**fbelow**(3, $k$) and **fbelow**(4, $k$)).

It is yet to be seen whether a counterexample exists for this scheme when $n$ is restricted to powers of 2.

**Third scheme.** Same as the second scheme, but replacing (5/4) with the Sikkema constant, $S$ = (4306+837*sqrt(6))/5832 (Lorentz 1986)**(5)**, (Sikkema 1961)**(7)**, which equals about 1.09. In fact, the same counterexamples for the second scheme apply to this one, since this scheme merely multiplies the offset to bring the approximating polynomials closer to $f$.

**Fourth scheme.** In this scheme, which relates to a result from Kopotun et al. (2017)**(8)**, let $f$ be a nondecreasing and Lipschitz continuous function in [0, 1]. Then for all $n\ge 2$:

- **fabove**($n$, $k$) = $f(k/n)$ + sqrt(1−(2*$k/n$−1)$^2$)*$L/n$.
- **fbelow**($n$, $k$) = $f(k/n)$ − sqrt(1−(2*$k/n$−1)$^2$)*$L/n$.

Where $L$ is the Lipschitz constant for $f$.

This counterexample has $f$ be a degree-12 polynomial in Bernstein form with coefficients [0, 61/625, 1273/10000, 697/5000, 1573/10000, 2411/5000, 271/500, 5903/10000, 374/625, 6013/10000, 6017/10000, 1107/1250, 8983/10000]. This polynomial is nondecreasing and Lipschitz continuous with Lipschitz constant ($L$) slightly less than 1.37277. And the counterexample involves the upper polynomials of degree 16 and 32 generated by **fabove**(16, $k$) and **fabove**(32, $k$), respectively.

**Note on "clamping".** For any approximation scheme, "clamping" the values of **fbelow** and **fabove** to fit the interval [0, 1] won't necessarily preserve the consistency requirement, even if the original scheme met that requirement.

Here is a counterexample that applies to any approximation scheme.

Let $g$ and $h$ be two polynomials in Bernstein form as follows:

- $g$ has degree 5 and coefficients [10179/10000, 2653/2500, 9387/10000, 5049/5000, 499/500, 9339/10000].
- $h$ has degree 6 and coefficients [10083/10000, 593/625, 9633/10000, 4513/5000, 4947/5000, 9473/10000, 4519/5000].

After elevating $g$'s degree, $g$'s coefficients are no less than $h$'s, as required by the consistency property.

However, if we clamp coefficients above 1 to equal 1, so that $g$ is now $g'$ with [1, 1, 9387/10000, 1, 499/500, 9339/10000] and $h$ is now $h'$ with [1, 593/625, 9633/10000, 4513/5000, 4947/5000, 9473/10000, 4519/5000], and elevate $g'$ for coefficients [1, 1, 14387/15000, 19387/20000, 1499/1500, 59239/60000, 9339/10000], some of the coefficients of $g'$ are less than those of $h'$. Thus, for this pair of polynomials, clamping the coefficients will destroy the consistent approximation property.

# 2 Achievable Simulation Rates

In general, the number of input coin flips needed by any Bernoulli factory algorithm for a factory function $f(\lambda)$ depends on how "smooth" the function $f$ is.

The following table summarizes the rate of simulation (in terms of the number of input coin flips needed) that can be achieved *in theory* depending on $f(\lambda)$, assuming the unknown probability of heads, $\lambda$, lies in the interval [$\epsilon$, 1−$\epsilon$] for some $\epsilon$ > 0. In the table below, $\Delta(n, r, \lambda) = O(\max(\text{sqrt}(\lambda*(1-\lambda)/n),1/n)^r)$, that is, $O((1/n)^r)$ near $\lambda = 0$ or 1, and $O((1/n)^{r/2})$ elsewhere.

| Property of simulation | Property of $f$ |
|---|---|
| Requires no more than $n$ input coin flips. | If and only if $f$ can be written as a polynomial in Bernstein form of degree $n$ with coefficients in [0, 1] (Goyal and Sigman 2012)**(9)**. |
| Requires a finite number of flips on average. Also known as "realizable" by Flajolet et al. (2010)**(3)**. | Only if $f$ is Lipschitz continuous (Nacu and Peres 2005)**(1)**. |
| Number of flips required, raised to power of $r$, is finite on average and has a tail that drops off uniformly for all $\lambda$. | Only if $f$ is $C^r$ continuous (has $r$ or more continuous derivatives, or "slope" functions) (Nacu and Peres 2005)**(1)**. |
| Requires more than $n$ flips with probability $\Delta(n, r + 1, \lambda)$, for integer $r \geq 0$ and all $\lambda$. (The greater $r$ is, the faster the simulation.) | Only if $f$ is $C^r$ continuous and the $r$th derivative is in the Zygmund class (has no vertical slope) (Holtz et al. 2011)**(10)**. |
| Requires more than $n$ flips with probability $\Delta(n, \alpha, \lambda)$, for non-integer $\alpha > 0$ and all $\lambda$. (The greater $\alpha$ is, the faster the simulation.) | If and only if $f$ is $C^r$ continuous and the $r$th derivative is $(\alpha - r)$-Hölder continuous, where $r$ = floor($\alpha$) (Holtz et al. 2011)**(10)**. |
| "Fast simulation" (number of flips required has a tail that drops off exponentially). Also known as "strongly realizable" by Flajolet et al. (2010)**(3)**. | If and only if $f$ is real analytic (is $C^\infty$ continuous, or has continuous $k$th derivative for every $k$, and agrees with its Taylor series "near" every point) (Nacu and Peres 2005)**(1)**. |
| Average number of flips bounded from below by $(f'(\lambda))^2*\lambda*(1-\lambda)/(f(\lambda)*(1-f(\lambda)))$, where $f'$ is the first derivative of $f$. | Whenever $f$ admits a fast simulation (Mendo 2019)**(11)**. |

## 3 Notes

- (1) Nacu, Şerban, and Yuval Peres. "**Fast simulation of new coins from old**", The Annals of Applied Probability 15, no. 1A (2005): 93-115.
- (2) Specifically, the constant $m$ is an upper bound of abs($f(x)-f(y)$)/(abs($x-y$)$^\alpha$) for all $x$, $y$ pairs, where $x$ and $y$ are each in [0, 1] and $x$ != $y$. However, this bound can't directly be calculated as it would involve checking an infinite number of $x$, $y$ pairs.
- (3) Flajolet, P., Pelletier, M., Soria, M., "**On Buffon machines and numbers**", arXiv:0906.5560 [math.PR], 2010.
- (4) Powell, M.J.D., *Approximation Theory and Methods*, 1981
- (5) G. G. Lorentz. Bernstein polynomials. 1986.
- (6) Popoviciu, T., "Sur l'approximation des fonctions convexes d'ordre supérieur", Mathematica (Cluj), 1935.
- (7) Sikkema, P.C., "Der Wert einiger Konstanten in der Theorie der Approximation mit Bernstein-Polynomen", Numer. Math. 3 (1961).
- (8) Kopotun, K.A., et al., "**Interpolatory pointwise estimates for monotone polynomial approximation**", arXiv:1711.07083 [math.CA], 2017.
- (9) Goyal, V. and Sigman, K., 2012. On simulating a class of Bernstein polynomials. ACM Transactions on Modeling and Computer Simulation (TOMACS), 22(2), pp.1-5.
- (10) Holtz, O., Nazarov, F., Peres, Y., "New Coins from Old, Smoothly", *Constructive Approximation* 33 (2011).

- [11] Mendo, Luis. "An asymptotically optimal Bernoulli factory for certain functions that can be expressed as power series." Stochastic Processes and their Applications 129, no. 11 (2019): 4366-4384.
- [12] Levy, H., *Stochastic dominance*, 1998.
- [13] Henry (https://math.stackexchange.com/users/6460/henry), Proving stochastic dominance for hypergeometric random variables, URL (version: 2021-02-20): **https://math.stackexchange.com/q/4033573** .

# 4 Appendix

## 4.1 Proofs for Hölder Function Approximation Scheme

This section shows mathematical proofs for some of the approximation schemes of this page.

There is a straightforward extension to lemma 6(i) of Nacu and Peres (2005)**[1]** to certain functions with a slope that tends to a vertical slope. Specifically, it applies to any *Hölder continuous* function, which means a continuous function whose slope doesn't go exponentially fast to a vertical slope.

**Lemma 1.** *Let f(λ) be a continuous and nondecreasing function, and let $X_k$ be a hypergeometric(2*n, k, n) random variable, where n≥1 is a constant integer and k is an integer in [0, 2*n] . Then* ***E****[f($X_k$/n)] is nondecreasing as k increases.*

*Proof.* This is equivalent to verifying whether $X_{m+1}/n \succeq X_m/n$ (and, obviously by extension, $X_{m+1} \succeq X_m$) in terms of first-degree stochastic dominance (Levy 1998)**[12]**. This means that the probability that ($X_{m+1} \le j$) is less than or equal to that for $X_m$ for each $j$ in the interval [0, $n$]. A proof of this was given by the user "Henry" of the *Mathematics Stack Exchange* community**[13]**. □

**Lemma 2.** *Let f(λ) be a continuous function that maps [0, 1] to [−1, 1], and let X be a hypergeometric(2*n, k, n) random variable.*

1. *If f is α-Hölder continuous with Hölder constant M, then—*

   abs(**E**[$f$($X$/$n$)] − $f$($k$/(2*$n$))),   (1)

   *is bounded from above by M*(1/(2*n))$^{\alpha/2}$, for all n≥1 that are integer powers of 2.*

2. *If f is α-Hölder continuous with Hölder constant M, then the expression (1) is bounded from above by M*(1/(7*n))$^{\alpha/2}$, for all n≥4 that are integer powers of 2.*

3. *If f has a second derivative whose absolute value is bounded from above by M, then the expression (1) is bounded from above by (M/2)*(1/(7*n)), for all n≥4 that are integer powers of 2.*

4. *If f is convex, nondecreasing, and bounded from below by 0, then the expression (1) is bounded from above by E[f(Y/n)] for all n≥1 that are integer powers of 2, where Y is a hypergeometric(2*n, n, n) random variable.*

*Proof.*

1. abs(**E**[$f(X/n)$] − $f$(k/(2*$n$))) ≤ **E**[abs($f(X/n)$ − $f(k/(2*n))$)] ≤ $M$***E**[abs($X/n$ − $k/(2*n)$)]$^{\alpha}$ (by the definition of Hölder continuous functions) ≤ $M$*(**E**[abs($X/n$ − $k/(2*n)$)]$^2$)$^{\alpha/2}$ = $M$***Var**[$X/n$]$^{\alpha/2}$ ≤ $M$*(1/(2*$n$))$^{\alpha/2}$.
2. For all integers $n$≥4, abs(**E**[$f(X/n)$] − $f$(k/(2*$n$))) ≤ $M$***Var**[$X/n$]$^{\alpha/2}$ = $M$*($k$*(2*$n$−$k$)/(4*(2*$n$−1)*$n^2$))$^{\alpha/2}$ ≤ $M$*($n^2$/(4*(2*$n$−1)*$n^2$))$^{\alpha/2}$ = $M$*(1/(8*$n$−4))$^{\alpha/2}$ ≤ $M$*(1/(7*$n$))$^{\alpha/2}$.
3. For all integers $n$≥4, abs(**E**[$f(X/n)$] − $f$(k/(2*$n$))) ≤ ($M$/2)***Var**[$X/n$]$^{\alpha/2}$ = ($M$/2)*($k$*(2*$n$−$k$)/(4*(2*$n$−1)*$n^2$)) ≤ ($M$/2)*($n^2$/(4*(2*$n$−1)*$n^2$)) = ($M$/2)*(1/(8*$n$−4)) ≤ ($M$/2)*(1/(7*$n$)).
4. Let $X_k$ be a hypergeometric(2*$n$, $k$, $n$) random variable. By Lemma 1 and the assumption that $f$ is nondecreasing, **E**[$f(X_k/n)$] is nondecreasing as $k$ increases, so take **E**[$f(X_n/n)$] = **E**[$f(Y/n)$] as the upper bound. Then, abs(**E**[$f(X/n)$] − $f(k/(2*n))$) = abs(**E**[$f(X/n)$] − $f$(**E**[$X/n$])) = **E**[$f(X/n)$] − $f$(**E**[$X/n$]) (by Jensen's inequality, because $f$ is convex and bounded by 0) = **E**[$f(X/n)$] − $f(k/(2*n))$ ≤ **E**[$f(X/n)$] (because $f$ is bounded by 0) ≤ **E**[$f(Y/n)$]. □

   **Notes:**

   1. **E**[.] means expected or average value, and **Var**[.] means variance. A hypergeometric(2*$n$, $k$, $n$) random variable is the number of "good" balls out of $n$ balls taken uniformly at random, all at once, from a bag containing 2*$n$ balls, $k$ of which are "good".
   2. $f$ is $\alpha$-Hölder continuous if its vertical slopes, if any, are no "steeper" than $M$*$\lambda^{\alpha}$, where $\alpha$ is in the interval (0, 1] and $M$ is greater than 0. An $\alpha$-Hölder continuous function in [0, 1] is also $\beta$-Hölder continuous for any $\beta$ less than $\alpha$.
   3. Parts 2 and 3 exploit a tighter bound on **Var**[$X/n$] than the bound given in Nacu and Peres (2005, Lemma 6(i) and 6(ii), respectively)**(1)**. However, for technical reasons, these bounds are proved only for all integers n≥4.
   4. For part 3, as in Lemma 6(ii) of Nacu and Peres 2005, the second derivative need not be continuous (Y. Peres, pers. comm., 2021).

**Theorem 1.** *Let f(λ), α, and M be as described in part 1 of Lemma 2, except f maps [0, 1] to the interval [ε, 1−ε] for ε in (0, 1/2). By forming two sequences of polynomials in Bernstein form with coefficients* ***fabove****(n, k) for the upper polynomials, and* ***fbelow****(n, k) for the lower polynomials, the result is an approximation scheme that meets conditions (i), (iii), and (iv) of Proposition 3 of Nacu and Peres (2005)****(1)****, for all n≥1 that are integer powers of 2, and thus can be used to simulate f via the algorithms for general factory functions described at the top of this page:*

- ***fbelow****(n, k) = f(k/n) − δ(n).*
- ***fabove****(n, k) = f(k/n) + δ(n).*

*Where* $\delta(n) = M/((2^{\alpha/2}-1)*n^{\alpha/2})$.

*Proof.* Follows from part 1 of Lemma 2 above as well as Remark B and the proof of Proposition 10 of Nacu and Peres (2005)**(1)**. The term $\delta(n)$ is found as a solution to the functional equation $\delta(n)$ = $\delta$(2*$n$) + $M$*(1/(2*$n$))$^{\alpha/2}$, and functional equations of this kind were suggested in the proof of Proposition 10, to find the offset by which to shift the approximating polynomials. □

> **Note:** For specific values of $\alpha$, the functional equation given in the proof can be solved via linear recurrences; an example for $\alpha$ = 1/2 is the following SymPy code: `rsolve(Eq(f(n),f(n+1)+z*(1/(2*2**n))**`

`((S(1)/2)/2)),f(n)).subs(n,ln(n,2)).simplify()`. Trying different values of $\alpha$ suggested the following formula for Hölder continuous functions with $\alpha$ of $1/j$ or greater: $(M^* \sum_{i=0,\ldots,(j^*2)-1} 2^{i/(2^*j)})/n^{1/(2^*j)} = M / ((2^{1/(2^*j)}-1)^*n^{1/(2^*j)})$; and generalizing the latter expression led to the term in the theorem.

**Theorem 2.** *Let f(λ) and M be as described in part 2 of Lemma 2, except f maps [0, 1] to the interval [ε, 1−ε] for ε in (0, 1/2). Then the following approximation scheme determined by* ***fabove*** *and* ***fbelow*** *meets conditions (i), (iii), and (iv) of Proposition 3 of Nacu and Peres (2005)*[**(1)**]*, for all n≥1 that are integer powers of 2:*

- ***fbelow****(n, k) = min(****fbelow****(4,0),* ***fbelow****(4,1), ...,* ***fbelow****(4,4)) if n < 4; otherwise, f(k/n) − η(n).*
- ***fabove****(n, k) = max(****fabove****(4,0),* ***fabove****(4,1), ...,* ***fabove****(4,4)) if n < 4; otherwise, f(k/n) + η(n).*

*Where* $\eta(n) = M^*(2/7)^{\alpha/2}/((2^{\alpha/2}-1)^*n^{\alpha/2})$.

*Proof.* Follows from part 2 of Lemma 2 above as well as Remark B and the proof of Proposition 10 of Nacu and Peres, including the observation in Remark B of the paper that we can start the algorithm from $n = 4$; in that case, the upper and lower polynomials of degree 1 through 3 above would be constant functions, so that as polynomials in Bernstein form, the coefficients of each one would be equal. The term $\eta(n)$ is found as a solution to the functional equation $\eta(n) = \eta(2^*n) + M^*(1/(7^*n))^{\alpha/2}$, and functional equations of this kind were suggested in the proof of Proposition 10, to find the offset by which to shift the approximating polynomials. □

**Note:** The term $\eta(n)$ was found in a similar way as the term $\delta(n)$ in Theorem 1.

**Theorem 3.** *Let f(λ) and M be as described in part 3 of Lemma 2, except f maps [0, 1] to the interval [ε, 1−ε] for ε in (0, 1/2). Then the following approximation scheme determined by* ***fabove*** *and* ***fbelow*** *meets conditions (i), (iii), and (iv) of Proposition 3 of Nacu and Peres (2005)*[**(1)**]*, for all n≥1 that are integer powers of 2:*

- ***fbelow****(n, k) = min(****fbelow****(4,0),* ***fbelow****(4,1), ...,* ***fbelow****(4,4)) if n < 4; otherwise, f(k/n) − M/(7*n).*
- ***fabove****(n, k) = max(****fabove****(4,0),* ***fabove****(4,1), ...,* ***fabove****(4,4)) if n < 4; otherwise, f(k/n) + M/(7*n).*

*Proof.* Follows from part 3 of Lemma 2 above as well as Remark B and the proof of Proposition 10 of Nacu and Peres, noting that the solution to the functional equation $\kappa(\text{n}) = \kappa(2^*n) + (M/2)^*(1/(7^*n))$ is $M/(7^*n)$. Notably, this exploits the observation in Remark B of the paper that we can start the algorithm from $n = 4$; in that case, the upper and lower polynomials of degree 1 through 3 above would be constant functions, so that as polynomials in Bernstein form, the coefficients of each one would be equal. □

**Proposition 1.**

1. *Let f be as given in Theorem 1, 2, or 3, except f is concave and may have a minimum of 0. The approximation scheme of that theorem remains valid if* ***fbelow****(n, k) = f(k/n), rather than as given in that theorem.*

2. *Let f be as given in Theorem 1, 2, or 3, except f is convex and may have a maximum of 1. The approximation scheme of that theorem remains valid if* ***fabove****(n, k) = f(k/n), rather than as given in that theorem.*

3. *Theorems 1, 2, and 3 can be extended to all integers n≥1, not just those that are*

*powers of 2, by defining—*

- ***fbelow**(n, k) = (k/n)***fbelow**(n−1, max(0, k−1)) + ((n−k)/n)***fbelow**(n−1, min(n−1, k)), and*
- ***fabove**(n, k) = (k/n)***fabove**(n−1, max(0, k−1)) + ((n−k)/n)***fabove**(n−1, min(n−1, k)),*

*for all n≥1 other than powers of 2. Parts 1 and 2 of this proposition still apply to the modified scheme.*

*Proof*. Parts 1 and 2 follow from Theorem 1, 2, or 3, as the case may be, and Jensen's inequality. Part 3 also follows from Remark B of Nacu and Peres (2005)**[(1)]**. □

## 5 License