Limit laws of anticipated rejection and related algorithms

Axel Bacher

Coauthors: Olivier Bodini, Alice Jacquot, Andrea Sportiello

Université Paris Nord

October 9th, 2017

Outline

1. Anticipated rejection
2. “Recovering” algorithms
3. Density of the limit laws
4. Perspectives

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

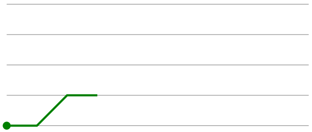
[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]



Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

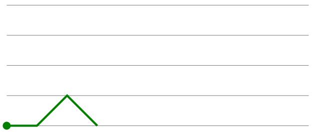
[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]



Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

- Complexity: $\mathcal{O}(\sqrt{n})$ tries,

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

- Complexity: $\mathcal{O}(\sqrt{n})$ tries, cost $\mathcal{O}(\sqrt{n})$ per try

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

- Complexity: $\mathcal{O}(\sqrt{n})$ tries, cost $\mathcal{O}(\sqrt{n})$ per try $\Rightarrow$ $\mathcal{O}(n)$.

Florentine algorithm

[Barcucci, Pinzani, Sprugnoli 1994]

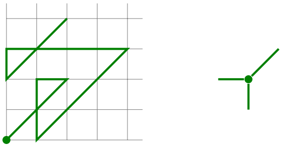
- Complexity: $\mathcal{O}(\sqrt{n})$ tries, cost $\mathcal{O}(\sqrt{n})$ per try $\Rightarrow \mathcal{O}(n)$.
- Limit law analysis **[Louchard 1999]**.

Florentine algorithm

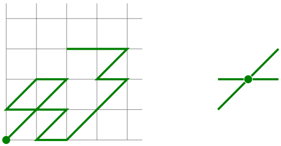
[Barcucci, Pinzani, Sprugnoli 1994]

- Complexity: $\mathcal{O}(\sqrt{n})$ tries, cost $\mathcal{O}(\sqrt{n})$ per try $\Rightarrow \mathcal{O}(n)$.
- Limit law analysis **[Louchard 1999]**.
- Motivation: directed animal random generation.

Florentine algorithms in the quarter-plane

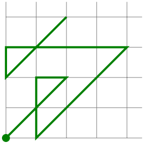
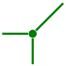


- Numer of tries $\mathcal{O}(n^{3/4})$.
- Cost of a try $\mathcal{O}(n^{1/4})$.
- Complexity $\mathcal{O}(n)$.

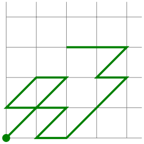
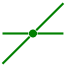


- Number of tries $\mathcal{O}(n^{2/3})$.
- Cost of a try $\mathcal{O}(n^{1/3})$.
- Complexity $\mathcal{O}(n)$.

Florentine algorithms in the quarter-plane

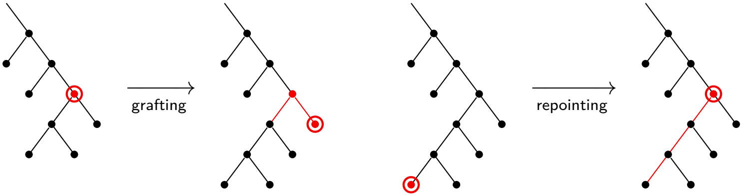
- Numer of tries $\mathcal{O}(n^{3/4})$.
- Cost of a try $\mathcal{O}(n^{1/4})$.
- Complexity $\mathcal{O}(n)$.

- Number of tries $\mathcal{O}(n^{2/3})$.
- Cost of a try $\mathcal{O}(n^{1/3})$.
- Complexity $\mathcal{O}(n)$.

- Efficient random generation of a wider set of quarter-plane walks **[Lumbroso, Mishna, Ponty 2016]**.
- Other families of walks: walks in a cone, d dimensions, etc.

Binary trees

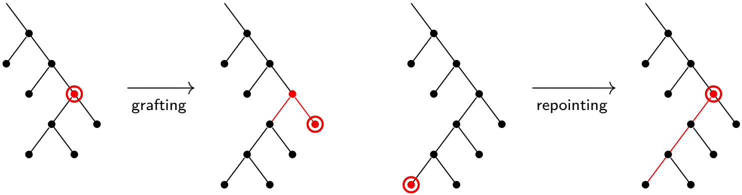


Random binary tree [B., Bodini, Jacquot 2013]

Start from a pointed leaf and repeat n times:

- graft a new leaf to the left or right (flip a coin) and point it;
- flip a coin; if tails, repoint;
- If repointing failed, delete the tree and start over.

Binary trees



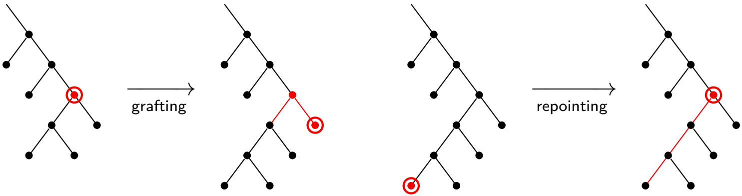
Random binary tree [B., Bodini, Jacquot 2013]

Start from a pointed leaf and repeat n times:

- graft a new leaf to the left or right (flip a coin) and point it;
- flip a coin; if tails, repoint;
- If repointing failed, delete the tree and start over.

- At each iteration, the tree is uniformly distributed.

Binary trees



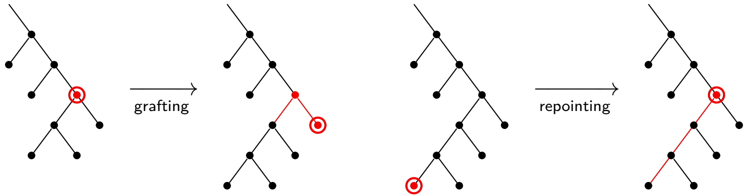
Random binary tree [B., Bodini, Jacquot 2013]

Start from a pointed leaf and repeat n times:

- graft a new leaf to the left or right (flip a coin) and point it;
- flip a coin; if tails, repoint;
- If repointing failed, delete the tree and start over.

- At each iteration, the tree is uniformly distributed.
- Complexity in random bits: $\mathcal{O}(\sqrt{n}) \times \mathcal{O}(\sqrt{n}) = \mathcal{O}(n)$.

Binary trees



Random binary tree **[B., Bodini, Jacquot 2013]**

Start from a pointed leaf and repeat n times:

- graft a new leaf to the left or right (flip a coin) and point it;
- flip a coin; if tails, repoint;
- If repointing failed, delete the tree and start over.

- At each iteration, the tree is uniformly distributed.
- Complexity in random bits: $\mathcal{O}(\sqrt{n}) \times \mathcal{O}(\sqrt{n}) = \mathcal{O}(n)$.
- This is a variant of Rémy's algorithm, which has complexity $\mathcal{O}(n \log n)$.

Limit law of anticipated rejection

- Let $(X_i)_{i \geq 0}$ be i.i.d. positive random variables such that, for $x > 0$:

$$\frac{\mathbf{P}[X \geq xt]}{\mathbf{P}[X \geq t]} \xrightarrow[t \to \infty]{} x^{-\alpha}, \qquad 0 < \alpha < 1.$$

- Let for $t > 0$:

$$i(t) = \min\{i \mid X_i \geq t\} \quad \text{and} \quad S(t) = X_0 + \cdots + X_{i(t)-1}.$$

Limit law of anticipated rejection

- Let $(X_i)_{i\geq 0}$ be i.i.d. positive random variables such that, for $x > 0$:

$$\frac{\mathbf{P}[X \geq xt]}{\mathbf{P}[X \geq t]} \xrightarrow[t\to\infty]{} x^{-\alpha}, \qquad 0 < \alpha < 1.$$

- Let for $t > 0$:

$$i(t) = \min\{i \mid X_i \geq t\} \quad \text{and} \quad S(t) = X_0 + \cdots + X_{i(t)-1}.$$

Theorem [B., Sportiello 2015]

The random variable $S(t)/t$ tends in distribution to D_α, with:

$$\mathbf{E}\left[e^{zD_\alpha}\right] = \left(1 - \sum_{n=1}^{\infty} \frac{\alpha}{n-\alpha}\frac{z^n}{n!}\right)^{-1}.$$

Limit law of anticipated rejection

- Let $(X_i)_{i\geq 0}$ be i.i.d. positive random variables such that, for $x > 0$:

$$\frac{\mathbf{P}[X \geq xt]}{\mathbf{P}[X \geq t]} \xrightarrow[t\to\infty]{} x^{-\alpha}, \qquad 0 < \alpha < 1.$$

- Let for $t > 0$:

$$i(t) = \min\{i \mid X_i \geq t\} \quad \text{and} \quad S(t) = X_0 + \cdots + X_{i(t)-1}.$$

Theorem [B., Sportiello 2015]

The random variable $S(t)/t$ tends in distribution to D_α, with:

$$\mathbf{E}\left[e^{zD_\alpha}\right] = \left(1 - \sum_{n=1}^{\infty} \frac{\alpha}{n-\alpha} \frac{z^n}{n!}\right)^{-1}.$$

- If $\alpha \geq 1$, the scaling factor is superlinear and the limit law exponential.

Limit law of anticipated rejection

- Let $(X_i)_{i\geq 0}$ be i.i.d. positive random variables such that, for $x > 0$:

$$\frac{\mathbf{P}[X \geq xt]}{\mathbf{P}[X \geq t]} \xrightarrow[t\to\infty]{} x^{-\alpha}, \qquad 0 < \alpha < 1.$$

- Let for $t > 0$:

$$i(t) = \min\{i \mid X_i \geq t\} \quad \text{and} \quad S(t) = X_0 + \cdots + X_{i(t)-1}.$$

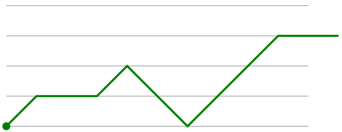
Theorem [B., Sportiello 2015]

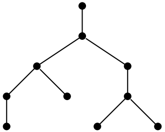
The random variable $S(t)/t$ tends in distribution to D_α, with:

$$\mathbf{E}\left[e^{zD_\alpha}\right] = \left(1 - \sum_{n=1}^{\infty} \frac{\alpha}{n-\alpha} \frac{z^n}{n!}\right)^{-1}.$$

- If $\alpha \geq 1$, the scaling factor is superlinear and the limit law exponential.
- The law D_α is the Darling-Mandelbrot law. **[Darling 1952, Lew 1994]**

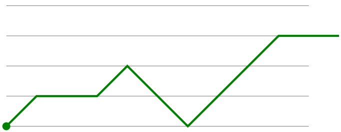
Second round of rejection

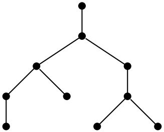




- A second round of rejection may occur when the size n is reached, with probability tending to p.

Second round of rejection

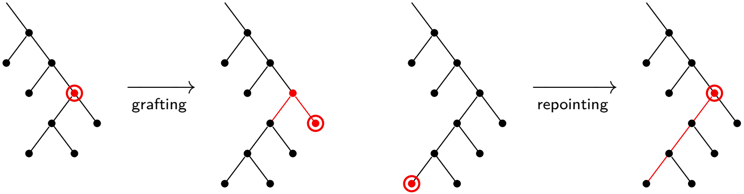




- A second round of rejection may occur when the size n is reached, with probability tending to p.
- If $p = \beta/(1+\beta)$, the complexity has limit law $D_{\alpha,\beta}$, with:

$$\mathbf{E}\left[e^{zD_{\alpha,\beta}}\right] = \left(1 - \sum_{n=1}^{\infty} \frac{\alpha + \beta n}{n - \alpha} \frac{z^n}{n!}\right)^{-1}.$$

Recovering algorithm for binary trees

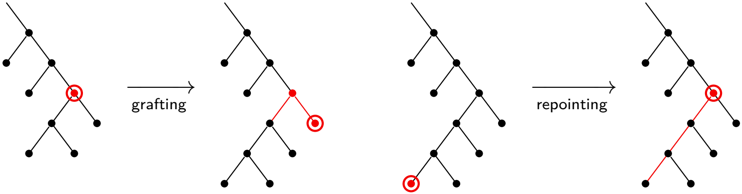


Random binary tree **[B., Bodini, Jacquot 2013]**

Start from a pointed leaf and repeat n times:

- graft a new leaf to the left or right (flip a coin) and point it;
- flip a coin; if tails, repoint;
- If repointing failed, pick a new point uniformly at random.

Recovering algorithm for binary trees



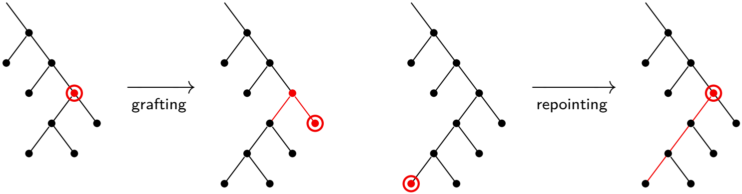
Random binary tree **[B., Bodini, Jacquot 2013]**

Start from a pointed leaf and repeat n times:

- graft a new leaf to the left or right (flip a coin) and point it;
- flip a coin; if tails, repoint;
- If repointing failed, pick a new point uniformly at random.

- Average cost in random bits: $2n + \mathcal{O}(\log^2 n)$ (entropic algorithm).

Recovering algorithm for binary trees



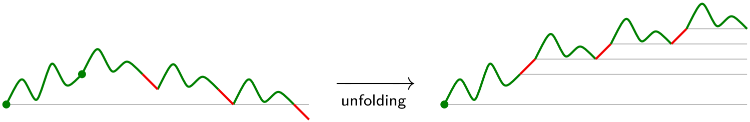
Random binary tree [B., Bodini, Jacquot 2013]

Start from a pointed leaf and repeat n times:

- graft a new leaf to the left or right (flip a coin) and point it;
- flip a coin; if tails, repoint;
- If repointing failed, pick a new point uniformly at random.

- Average cost in random bits: $2n + \mathcal{O}(\log^2 n)$ (entropic algorithm).
- Does not work on unary-binary trees (uniformity is lost).

Recovering algorithm for Dyck prefixes

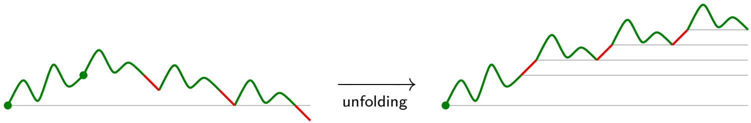


Random Dyck prefix **[B. 2016]**

Start from the empty path and repeat n times:

- Add a random step to P.
- If P is not a Dyck prefix, pick a point uniformly at random and unfold.

Recovering algorithm for Dyck prefixes



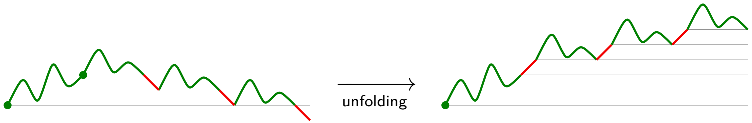
Random Dyck prefix **[B. 2016]**

Start from the empty path and repeat n times:

- Add a random step to P.
- If P is not a Dyck prefix, pick a point uniformly at random and unfold.

- At each iteration, the path is uniformly distributed.

Recovering algorithm for Dyck prefixes



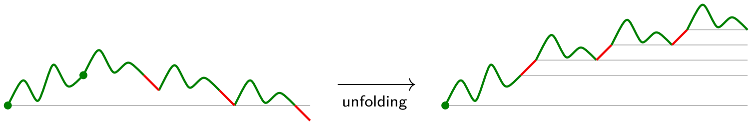
Random Dyck prefix **[B. 2016]**

Start from the empty path and repeat n times:

- Add a random step to P.
- If P is not a Dyck prefix, pick a point uniformly at random and unfold.

- At each iteration, the path is uniformly distributed.
- Cost: $n + \mathcal{O}(\log^2 n)$ random bits, $\mathcal{O}(n)$ time.

Recovering algorithm for Dyck prefixes



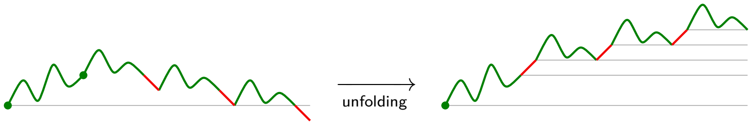
Random Dyck prefix **[B. 2016]**

Start from the empty path and repeat n times:

- Add a random step to P.
- If P is not a Dyck prefix, pick a point uniformly at random and unfold.

- At each iteration, the path is uniformly distributed.
- Cost: $n + \mathcal{O}(\log^2 n)$ random bits, $\mathcal{O}(n)$ time.
- Possible extension to m-Dyck paths $(+1/-m)$,
  entropic if we have an entropic source of $\mathrm{Bernoulli}\left(\frac{1}{1+m}\right)$.

Recovering algorithm for Dyck prefixes



Random Dyck prefix **[B. 2016]**

Start from the empty path and repeat n times:

- Add a random step to P.
- If P is not a Dyck prefix, pick a point uniformly at random and unfold.

- At each iteration, the path is uniformly distributed.
- Cost: $n + \mathcal{O}(\log^2 n)$ random bits, $\mathcal{O}(n)$ time.
- Possible extension to m-Dyck paths $(+1/-m)$, entropic if we have an entropic source of $\mathrm{Bernoulli}\left(\frac{1}{1+m}\right)$.
- Does not work on Motzkin or Schröder paths.

Limit laws

- Let B_n and M_n be the cost in random bits and memory accesses of the “recoveries” in the Dyck prefix algorithm.

Theorem

The variable B_n tends to a Gaussian law, with:

$$\mathbf{E}[B_n] \sim \frac{\log^2 n}{4 \log 2}, \qquad \mathbf{V}[B_n] \sim \frac{\log^3 n}{6 \log^2 2}.$$

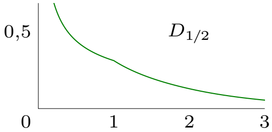
The variable M_n/n tends to $L_{1/2}$, where the law L_α is defined by:

$$L_\alpha = \sum_{x \in \mathrm{Poisson}_{(0,1]} \frac{\alpha}{x}} \mathrm{Unif}[0,x]$$

$$\mathbf{E}\left[e^{zL_\alpha}\right] = \exp\left(\sum_{n=1}^{\infty} \frac{\alpha}{n(n+1)} \frac{z^n}{n!}\right).$$

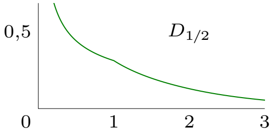
Density of the law D_α

$$\mathbf{E}\left[e^{zD_\alpha}\right] = \left(1 - \sum_{n=1}^{\infty} \frac{\alpha}{n-\alpha}\frac{z^n}{n!}\right)^{-1}$$



Density of the law D_α

$$\mathbf{E}\big[e^{zD_\alpha}\big] = \left(1 - \sum_{n=1}^{\infty} \frac{\alpha}{n-\alpha}\frac{z^n}{n!}\right)^{-1}$$



- The Laplace transform of D_α takes the form:

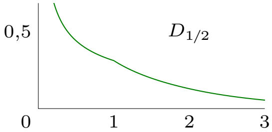
$$\mathbf{E}\big[e^{-zD_\alpha}\big] = \frac{A(z)}{1-B(z)},$$

$$A(z) = \frac{z^{-\alpha}}{\Gamma(1-\alpha)}$$

$$B(z) = \int_z^\infty \frac{e^{-y}y^{-1-\alpha}}{\Gamma(-\alpha)}\,dy.$$

Density of the law D_α

$$\mathbf{E}\left[e^{zD_\alpha}\right] = \left(1 - \sum_{n=1}^{\infty} \frac{\alpha}{n-\alpha}\frac{z^n}{n!}\right)^{-1}$$



- The Laplace transform of D_α takes the form:

$$\mathbf{E}\left[e^{-zD_\alpha}\right] = \frac{A(z)}{1-B(z)}, \qquad A(z) = \frac{z^{-\alpha}}{\Gamma(1-\alpha)}$$

$$B(z) = \int_z^\infty \frac{e^{-y}y^{-1-\alpha}}{\Gamma(-\alpha)}\,dy.$$

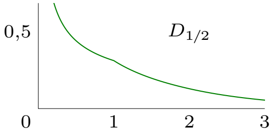
- Its density is therefore:

$$f(x) = \sum_{k=0}^{\infty} a * b^{*k}(x), \qquad a(x) = \frac{\sin(\alpha\pi)}{\pi}x^{\alpha-1}$$

$$b(x) = -\frac{\sin(\alpha\pi)}{\pi}\frac{(x-1)^\alpha}{x}\mathbf{1}_{x>1}$$

Density of the law D_α

$$\mathbf{E}\left[e^{zD_\alpha}\right] = \left(1 - \sum_{n=1}^{\infty} \frac{\alpha}{n-\alpha}\frac{z^n}{n!}\right)^{-1}$$



- The Laplace transform of D_α takes the form:

$$\mathbf{E}\left[e^{-zD_\alpha}\right] = \frac{A(z)}{1-B(z)}, \qquad A(z) = \frac{z^{-\alpha}}{\Gamma(1-\alpha)}, \qquad B(z) = \int_z^\infty \frac{e^{-y}y^{-1-\alpha}}{\Gamma(-\alpha)}\,dy.$$

- Its density is therefore:

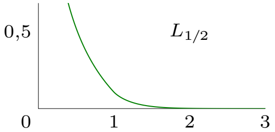
$$f(x) = \sum_{k=0}^{\infty} a * b^{*k}(x), \qquad a(x) = \frac{\sin(\alpha\pi)}{\pi}x^{\alpha-1}, \qquad b(x) = -\frac{\sin(\alpha\pi)}{\pi}\frac{(x-1)^\alpha}{x}\mathbf{1}_{x>1}$$

  and satisfies:

$$xf'(x) + (1-\alpha)f(x) = -\alpha f * f(x-1).$$

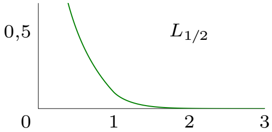
Density of the law $L_{1/2}$

$$\mathbf{E}\left[e^{zL_{1/2}}\right] = \exp\left(\sum_{n=1}^{\infty} \frac{1}{2n(n+1)}\,\frac{z^n}{n!}\right)$$



Density of the law $L_{1/2}$

$$\mathbf{E}\left[e^{zL_{1/2}}\right] = \exp\left(\sum_{n=1}^{\infty} \frac{1}{2n(n+1)} \frac{z^n}{n!}\right)$$



- The Laplace transform of $L_{1/2}$ takes the form:

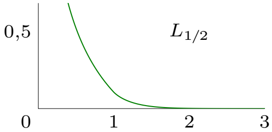
$$\mathbf{E}\left[e^{-zL_{1/2}}\right] = A(z)\exp\big(B(z)\big),$$

$$A(z) = e^{\frac{1-\gamma}{2}} e^{-\frac{1}{2z}} z^{-1/2}$$

$$B(z) = \int_z^{\infty} \frac{e^{-y}}{2y^2}\,dy.$$

Density of the law $L_{1/2}$

$$\mathbf{E}\left[e^{zL_{1/2}}\right] = \exp\left(\sum_{n=1}^{\infty} \frac{1}{2n(n+1)} \frac{z^n}{n!}\right)$$



- The Laplace transform of $L_{1/2}$ takes the form:

$$\mathbf{E}\left[e^{-zL_{1/2}}\right] = A(z)\exp\big(B(z)\big),$$

$$A(z) = e^{\frac{1-\gamma}{2}} e^{-\frac{1}{2z}} z^{-1/2}$$

$$B(z) = \int_z^{\infty} \frac{e^{-y}}{2y^2}\, dy.$$

- Its density $f(x)$ is therefore:

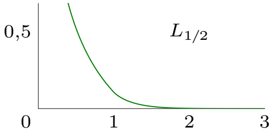
$$f(x) = \sum_{k=0}^{\infty} \frac{a * b^{*k}(x)}{k!},$$

$$a(x) = e^{\frac{1-\gamma}{2}} \frac{\cos\sqrt{2x}}{\sqrt{\pi x}}$$

$$b(x) = \frac{x-1}{2x}\mathbf{1}_{x>1}$$

Density of the law $L_{1/2}$

$$\mathbf{E}\left[e^{zL_{1/2}}\right] = \exp\left(\sum_{n=1}^{\infty} \frac{1}{2n(n+1)} \frac{z^n}{n!}\right)$$



- The Laplace transform of $L_{1/2}$ takes the form:

$$\mathbf{E}\left[e^{-zL_{1/2}}\right] = A(z)\exp\big(B(z)\big),$$

$$A(z) = e^{\frac{1-\gamma}{2}} e^{-\frac{1}{2z}} z^{-1/2}$$

$$B(z) = \int_z^{\infty} \frac{e^{-y}}{2y^2}\,dy.$$

- Its density $f(x)$ is therefore:

$$f(x) = \sum_{k=0}^{\infty} \frac{a * b^{*k}(x)}{k!},$$

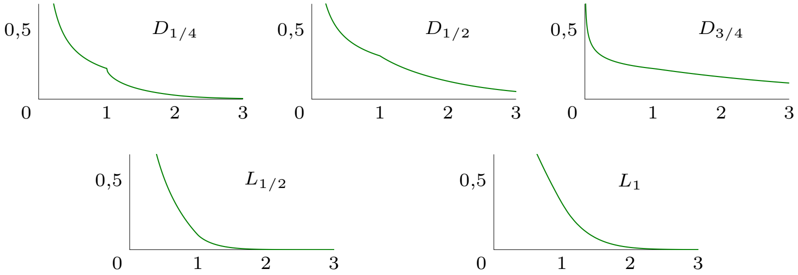
$$a(x) = e^{\frac{1-\gamma}{2}} \frac{\cos\sqrt{2x}}{\sqrt{\pi x}}$$

$$b(x) = \frac{x-1}{2x}\mathbf{1}_{x>1}$$

and satisfies:

$$2xf''(x) + 3f'(x) + f(x) = f(x-1).$$

Distribution tails

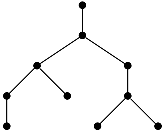


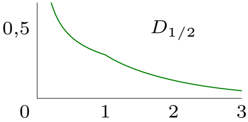
- The tails of D_α and L_α are of the form: **[Lew 1994]**

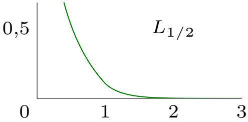
$$\mathbf{P}[D_\alpha \geq x] = \frac{e^{-a_0}}{\alpha} e^{-a_0 x} + \mathcal{O}(e^{-a_1 x})$$

$$\mathbf{P}[L_\alpha \geq x] = \left(\frac{\alpha e}{x \log^2 x} \right)^x e^{o(x)}.$$

Perspectives







- Can we make the "recovery" idea work with other walks or trees? (Motzkin, Schröder, $+a/-b$, etc.)
- Are there other interesting distributions with similar properties? (ex: Dickman function in number theory)