Richardson iteration in the context of L vs. BPL (lecture notes)

Course: Derandomizing Space-Bounded Computation, Winter 2025, University of Chicago
Instructor: William Hoza (`williamhoza@uchicago.edu`)

In these lecture notes, we explain how to use a technique called *Richardson iteration* to decrease error in space-bounded derandomization. First, we will use this technique to sketch the proof of the following theorem.

Theorem 0.1 (Low-error derandomization of BPL [AKMPSV20])**.** *Given a width-n length-n standard-order ROBP f and an error parameter $\varepsilon \in (0,1)$, it is possible to deterministically compute $\mathbb{E}[f]$ to within $\pm\varepsilon$ using $O(\log^{3/2} n + \log n \cdot \log\log(1/\varepsilon))$ bits of space.*

Afterward, we will use Richardson iteration to construct so-called *weighted pseudorandom generators* with very low error.

1 The inverse Laplacian perspective

It is natural to view the problem of derandomizing BPL as a *matrix powering* problem. If f is a width-w length-n standard-order ROBP, we can define its transition probability matrix $M \in [0,1]^{N\times N}$, where $N = w \cdot (n+1)$, by letting $M_{u,v}$ be the number of edges from u to v divided by two. Note that M is *not* a stochastic matrix, because $M_{u,v} = 0$ whenever u is in the final layer of the program.

For any pair of vertices u and v and any number $t \in \mathbb{N}$, the entry $M^t_{u,v}$ is equal to the probability of arriving at v if we start at u and take i random steps. In particular, if v_{start} is the start vertex and v_{acc} is the (wlog unique) accepting vertex, then $\mathbb{E}[f] = M^n_{v_{\text{start}},v_{\text{acc}}}$.

The first step of the proof of Theorem 0.1 is to recast the problem as a matrix *inversion* problem. Let $L = I - M$. The matrix L is called the *Laplacian matrix* of the directed graph f. This matrix L is invertible, and its inverse is given by

$$L^{-1} = M^0 + M^1 + \cdots + M^n.$$

This is because $M^{n+1} = 0$, so $(I - M)\cdot(M^0 + \cdots + M^n) = M^0 - M^{n+1} = I$. The matrices $M^0, \dots, M^n$ are supported on disjoint sets of entries, so $\mathbb{E}[f]$ is equal to the $(v_{\text{start}}, v_{\text{accept}})$ entry of L^{-1}.

2 Richardson iteration

Suppose we are given a matrix L. We would like to approximate L^{-1}. We are given an initial moderate-quality approximation $A \approx L^{-1}$. How can we compute a higher-quality approximation?

Intuitively (i.e., without worrying about issues of invertibility or convergence), we can relate the exact inverse L^{-1} to our initial approximation A by writing

$$L^{-1} = A \cdot (LA)^{-1} = A \cdot \sum_{\ell=0}^{\infty} (I - LA)^{\ell}.$$

Richardson iteration consists of truncating the sum above after some finite number of terms. That is, for each $m \in \mathbb{N}$, define

$$A^{(m)} = A \cdot \sum_{\ell=0}^{m} (I - LA)^{\ell}.$$

As m gets larger, $A^{(m)}$ gets more "expensive" because it involves more matrix arithmetic. However, if the entries of $I - LA$ are "small," then the approximation quality gets higher and higher as m gets large:

Proposition 2.1. $I - LA^{(m)} = (I - LA)^{m+1}$.

Proof. Let $E = I - LA$. Then

$$LA^{(m)} = (I - E) \cdot \sum_{\ell=0}^{m} E^{\ell} = \sum_{\ell=0}^{m} E^{\ell} - \sum_{\ell=0}^{m} E^{\ell+1} = I - E^{m+1}. \qquad \square$$

Corollary 2.2. *If L is the Laplacian of a length-n standard-order ROBP and $\|L^{-1} - A\|_1 \leq \varepsilon_0$, then $\|L^{-1} - A^{(m)}\|_1 \leq (n+1) \cdot (2\varepsilon_0)^{m+1}$.*

Proof. Let M be the transition probability matrix of the ROBP. Then

$$\begin{aligned}
\|L^{-1} - A^{(m)}\|_1 = \|L^{-1} \cdot (I - LA^{(m)})\|_1 &\leq \|L^{-1}\|_1 \cdot \|I - LA^{(m)}\|_1 \\
&= \|L^{-1}\|_1 \cdot \|(I - LA)^{m+1}\|_1 && \text{(Proposition 2.1)} \\
&\leq \|L^{-1}\|_1 \cdot \|I - LA\|_1^{m+1} \\
&= \|L^{-1}\|_1 \cdot \|L \cdot (L^{-1} - A)\|_1^{m+1} \\
&\leq \|L^{-1}\|_1 \cdot (\|L\|_1 \cdot \|L^{-1} - A\|_1)^{m+1} \\
&\leq \left(\sum_{i=0}^{n} \|M^i\|_1 \right) \cdot ((\|I\|_1 + \|M\|_1) \cdot \varepsilon_0)^{m+1} \\
&= (n+1) \cdot (2\varepsilon_0)^{m+1}. \qquad \square
\end{aligned}$$

Proof sketch of Theorem 0.1. Let M be the transition probability matrix of f, and let $L = I - M$. First, we use the Saks-Zhou algorithm to compute a matrix A such that $\|A - L^{-1}\|_1 \leq 0.1$. (Remember, each entry of L^{-1} is the expectation of some subprogram of f.) This uses $O(\log^{3/2} n)$ bits of space.

Now let $E = I - LA$ and $A' = A \cdot \sum_{\ell=0}^{m} E^{\ell}$ for a suitable value $m = O(\log(n/\varepsilon))$. The algorithm: Output the $(v_{\text{start}}, v_{\text{acc}})$ entry of A'. The correctness of this algorithm is immediate from Corollary 2.2. We omit the efficiency analysis because it is a bit annoying. It involves the nontrivial fact that the product of n given n-bit integers can be computed using $O(\log n)$ bits of space. $\square$

3 Weighted PRGs

In this section, we describe a more sophisticated application of Richardson iteration. We will use Richardson iteration to construct a *weighted pseudorandom generator* (WPRG).

Definition 3.1 (WPRG [BCG20]). A *weighted PRG* (WPRG) is a pair (G, ρ), where $G\colon \{0,1\}^s \to \{0,1\}^n$ and $\rho\colon \{0,1\}^s \to \mathbb{R}$. We say that the WPRG fools $f\colon \{0,1\}^n \to \{0,1\}$ with error ε if

$$\left| \mathop{\mathbb{E}}_{x \sim U_s}[f(G(x)) \cdot \rho(x)] - \mathbb{E}[f] \right| \leq \varepsilon.$$

Crucially, the weights $\rho(x)$ are allowed to be negative. The expectation $\mathbb{E}_{x \sim U_s}[f(G(x)) \cdot \rho(x)]$ can be thought of as the expectation of f under a "pseudodistribution" in which some strings might have "negative probability" of occurring.

A standard PRG is the special case $\rho \equiv 1$. For some purposes, WPRGs are just as useful as unweighted PRGs. For example, an explicit optimal WPRG fooling standard-order ROBPs would imply $\mathsf{L} = \mathsf{BPL}$, because we can exhaustively try all seeds to compute the value $\mathbb{E}_{x \sim U_s}[f(G(x)) \cdot \rho(x)]$, just like the unweighted case. At the same time, WPRGs are sometimes *easier to construct* than unweighted PRGs are, because we are allowed to use negative weights whenever it's convenient. Indeed, we will sketch the proof of the following theorem:

Theorem 3.2 (Low-error WPRGs for standard-order ROBPs). *For every $w, n \in \mathbb{N}$ and $\varepsilon \in (0,1)$, there exists an explicit WPRG that fools width-w length-n standard-order ROBPs with error ε and seed length $\widetilde{O}(\log(wn) \cdot \log n + \log(1/\varepsilon))$.*

For comparison, recall that the INW PRG fools standard-order ROBPs using a seed of length $O(\log(wn/\varepsilon) \cdot \log n)$. Theorem 3.2 is superior for small values of ε such as $\varepsilon = 1/n^{\log n}$.

The original proof of Theorem 3.2 is due to Braverman, Cohen, and Garg [BCG20]. We will present a later and simpler proof, which was discovered independently by Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRST21] and by Pyne and Vadhan [PV21].

3.1 Applying Richardson iteration to a PRG

The idea behind the proof of Theorem 3.2 is that we will start with an unweighted PRG with error $1/\operatorname{poly}(wn)$ and seed length $O(\log(wn) \cdot \log n)$, such as Nisan's PRG G_{Nisan}. Then, we will apply Richardson iteration to decrease the error down to ε. Richardson iteration will increase the seed length a little, but (after applying another trick) the final seed length will only be $\widetilde{O}(\log(wn) \cdot \log n + \log(1/\varepsilon))$.

It is not immediately clear how to implement this plan, because Richardson iteration seems to be an operation on *matrices*, not an operation on PRGs. Our goal is to construct a (weighted) PRG that fools *all* width-w length-n standard-order ROBPs, so we don't have access to the matrix M like we did in the proof of Theorem 0.1.

The solution is to "reverse-engineer" Richardson iteration, i.e., we will construct the WPRG in such a way that Richardson iteration is what happens in the *analysis* of the WPRG. To explain the details, we will use the following formalism.

Definition 3.3 (Pseudodistributions)**.** For our purposes, a *pseudodistribution* over $\{0,1\}^n$ is any formal real linear combination of n-bit strings. For example, the following is a pseudodistribution over $\{0,1\}^3$:

$$X = 0.4 \cdot 000 + 0.1 \cdot 010 - 0.3 \cdot 111 + 0.7 \cdot 110.$$

A *distribution* is the special case that the coefficients are nonnegative and the coefficients add up to 1. For example,

$$U_n = \sum_{x \in \{0,1\}^n} 2^{-n} \cdot x.$$

A pseudodistribution with 2^s terms is equivalent to a WPRG with seed length s. If X and Y are pseudodistributions over $\{0,1\}^n$, then so are $X + Y$ and $X - Y$. We also define the *tensor product* of pseudodistributions as follows.

Definition 3.4 (Tensor product of pseudodistributions)**.** Let $\sum_{i=1}^{R_1} a_i \cdot x^{(i)}$ be a pseudodistribution over $\{0,1\}^{n_1}$, and let $\sum_{i=1}^{R_2} b_i \cdot y^{(i)}$ be a pseudodistribution over $\{0,1\}^{n_2}$. Then we define

$$\left(\sum_{i=1}^{R_1} a_i \cdot x^{(i)}\right) \otimes \left(\sum_{i=1}^{R_2} b_i \cdot y^{(i)}\right) = \sum_{i=1}^{R_1}\sum_{j=1}^{R_2} (a_i b_j) \cdot x^{(i)} y^{(j)},$$

a pseudodistribution over $\{0,1\}^{n_1+n_2}$.

Using these operations, we are now ready to explain how to reverse-engineer Richardson iteration.

Definition 3.5 (Reverse-engineering Richardson iteration)**.** Let X be a (pseudo)distribution over $\{0,1\}^n$, say $X = \sum_{k=1}^{R} a_k \cdot x^{(k)}$. For each $0 \le i \le j \le n$, define

$$X_{i \to j} = \sum_{k=1}^{R} a_k \cdot (x_{i+1}^{(k)}, \dots, x_j^{(k)}),$$

a pseudodistribution over $\{0,1\}^{j-i}$. Furthermore, for every $0 \le i < j \le n$, define

$$\Delta_{i \to j} = U_1 \otimes X_{i+1 \to j} - X_{i \to j},$$

so $\Delta_{i\to j}$ is a pseudodistribution over $\{0,1\}^{j-i}$. Finally, for every $m \in \mathbb{N}$, we define a pseudodistribution $X^{(m)}$ over $\{0,1\}^n$ by the formula

$$X^{(m)} = \sum_{\ell=0}^{m} \sum_{\substack{i_0,i_1,\dots,i_\ell \in \mathbb{N} \\ 0 \le i_0 < \cdots < i_\ell = n}} X_{0\to i_0} \otimes \Delta_{i_0\to i_1} \otimes \cdots \otimes \Delta_{i_{\ell-1}\to i_\ell}. \tag{1}$$

Intuitively, the pseudodistribution $\Delta_{i\to j}$ corresponds to the error matrix $E = I - LA$ in Richardson iteration. The tensor product $X_{0\to i_0} \otimes \Delta_{i_0\to i_1} \otimes \cdots \otimes \Delta_{i_{\ell-1}\to i_\ell}$ corresponds to the matrix product $A \cdot E^\ell$. The sum over all sequences of indices $i_0, i_1, \dots, i_\ell$ is necessary because of the way that matrix multiplication is defined.

To rigorously show that we have successfully reverse-engineered Richardson iteration, we use the notion of *pseudoexpectation*.

Definition 3.6 (Pseudoexpectation)**.** Let $X = \sum_{i=1}^{R} a_i \cdot x^{(i)}$ be a pseudodistribution over $\{0,1\}^n$, and let $f\colon \{0,1\}^n \to \mathbb{R}^{w\times w}$. We define

$$\widetilde{\mathbb{E}}[f(X)] = \sum_{i=1}^{R} a_i \cdot f(x^{(i)}).$$

We also use the following convenient notation. Let $N = w \cdot (n+1)$ be the number of vertices in a width-w length-n standard-order ROBP with layers $V_0, \dots, V_n$. For any matrix $M \in \mathbb{R}^{N\times N}$, such as the transition probability matrix, the Laplacian matrix, etc., and for any $i, j \in \{0, 1, \dots, n\}$, we define $M_{i\to j}$ to be the $w \times w$ block of M consisting of all entries $M_{u,v}$ where $u \in V_i$ and $v \in V_j$.

Lemma 3.7 (We successfully reverse-engineered Richardson iteration)**.** *Let $f\colon \{0,1\}^n \to \{0,1\}^{V_0\times V_n}$ be a width-w standard-order ROBP with layers $V_0, \dots, V_n$.*[1] *Let $N = w\cdot(n+1)$ be the number of vertices, let $M \in [0,1]^{N\times N}$ be the transition probability matrix, and let $L = I - M$ be the Laplacian matrix. Let X be a (pseudo)distribution over $\{0,1\}^n$. For every $0 \le i \le j \le n$ and every $u \in V_i$, $v \in V_j$, let $A_{u,v} = \widetilde{\mathbb{E}}[f_{u\to v}(X_{i\to j})]$. For all other pairs of vertices u, v, let $A_{u,v} = 0$, so $A \in [0,1]^{N\times N}$. Let $m \in \mathbb{N}$, and define $A^{(m)} = A \cdot \sum_{\ell=0}^{m} (I - LA)^\ell$. Then $\widetilde{\mathbb{E}}[f(X^{(m)})] = A^{(m)}_{0\to n}$.*

The proof relies on a couple of basic properties of pseudoexpectation.

Fact 3.8. *Let $f\colon \{0,1\}^n \to \mathbb{R}^{w\times w}$, let X and Y be pseudodistributions over $\{0,1\}^n$, and let $c \in \mathbb{R}$. Then*

$$\widetilde{\mathbb{E}}[f(X + cY)] = \widetilde{\mathbb{E}}[f(X)] + c \cdot \widetilde{\mathbb{E}}[f(Y)].$$

Fact 3.9. *Let $f\colon \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \mathbb{R}^{w\times w}$ be a function of the form $f(x,y) = g(x)\cdot h(y)$, where $g\colon \{0,1\}^{n_1} \to \mathbb{R}^{w\times w}$ and $h\colon \{0,1\}^{n_2} \to \mathbb{R}^{w\times w}$. Let X and Y be pseudodistributions over $\{0,1\}^{n_1}$ and $\{0,1\}^{n_2}$ respectively. Then*

$$\widetilde{\mathbb{E}}[f(X \otimes Y)] = \widetilde{\mathbb{E}}[g(X)] \cdot \widetilde{\mathbb{E}}[h(Y)].$$

Proof of Lemma 3.7. Let $f_{i\to j}\colon \{0,1\}^{j-i} \to \{0,1\}^{V_i\times V_j}$ denote the subprogram of f consisting only of layers $V_i, \dots, V_j$. Then

$$\begin{aligned}
\widetilde{\mathbb{E}}[f(X^{(m)})] &= \sum_{\ell=0}^{m} \sum_{\substack{i_0,i_1,\dots,i_\ell \in \mathbb{N} \\ 0 \le i_0 < \cdots < i_\ell = n}} \widetilde{\mathbb{E}}[f(X_{0\to i_0} \otimes \Delta_{i_0\to i_1} \otimes \cdots \otimes \Delta_{i_{\ell-1}\to i_\ell})] \\
&= \sum_{\ell=0}^{m} \sum_{\substack{i_0,i_1,\dots,i_\ell \in \mathbb{N} \\ 0 \le i_0 < \cdots < i_\ell = n}} \widetilde{\mathbb{E}}[f_{0\to i_0}(X_{0\to i_0})] \cdot \widetilde{\mathbb{E}}[f_{i_0\to i_1}(\Delta_{i_0\to i_1})] \cdots \widetilde{\mathbb{E}}[f_{i_{\ell-1}\to i_\ell}(\Delta_{i_{\ell-1}\to i_\ell})].
\end{aligned}$$

[1] I.e., $f(x)_{u,v} = f_{u\to v}(x)$.

The first term, $\widetilde{\mathbb{E}}[f_{0\to i_0}(X_{0\to i_0})]$, is simply $A_{0\to i_0}$. Now let us consider a term of the form $\widetilde{\mathbb{E}}[f_{i\to j}(\Delta_{i\to j})]$ where $i < j$. We have

$$\begin{aligned}\widetilde{\mathbb{E}}[f_{i\to j}(\Delta_{i\to j})] &= \mathbb{E}[f_{i\to i+1}(U_1)] \cdot \widetilde{\mathbb{E}}[f_{i+1\to j}(X_{i+1\to j})] - \widetilde{\mathbb{E}}[f_{i\to j}(X_{i\to j})] \\ &= M_{i\to i+1} \cdot A_{i+1\to j} - A_{i\to j} \\ &= (MA - A)_{i\to j} \\ &= (I - LA)_{i\to j},\end{aligned}$$

where the last step relies on the fact that $i \neq j$ and hence $I_{i\to j} = 0$. Thus, overall, if we let $E = I - LA$, then we get

$$\begin{aligned}\widetilde{\mathbb{E}}[f(X^{(m)})] &= \sum_{\ell=0}^{m} \sum_{\substack{i_0,i_1,\dots,i_\ell\in\mathbb{N} \\ 0\le i_0<\cdots<i_\ell=n}} A_{0\to i_0} \cdot E_{i_0\to i_1} \cdots E_{i_{\ell-1}\to i_\ell} \\ &= \sum_{\ell=0}^{m} (AE^\ell)_{0\to n} \\ &= A^{(m)}_{0\to n}.\end{aligned}$$

□

3.2 Using correlated seeds

Let us summarize where we are at so far. Let X be the output distribution of Nisan's PRG G_{Nisan}, with error $1/\operatorname{poly}(wn)$ and seed length $s = O(\log(wn) \cdot \log n)$. By Lemma 3.7 and Corollary 2.2, if we choose a suitable value $m = \Theta(\log(1/\varepsilon)/\log(wn))$, then the "reverse-engineered-Richardson-iteration" pseudodistribution $X^{(m)}$ fools width-w length-n ROBPs with error ε. (Assume $\varepsilon < 1/(wn)$, because otherwise Theorem 3.2 is not interesting.)

We can interpret a pseudodistribution such as $X^{(m)}$ as a WPRG. However, the seed length of $X^{(m)}$ is too large. Indeed, the tensor product in Eq. (1) means we are effectively running G_{Nisan} up to $m + 1$ times on independent seeds, so the seed length is bigger than $m \cdot s = \Theta(\log(1/\varepsilon) \cdot \log n)$, which is more than we can afford.

To improve the seed length, we will use the INW generator to generate a *pseudorandom sequence of seeds* for Nisan's PRG, instead of running Nisan's PRG on independent seeds. To explain this in more detail, let us assume for simplicity's sake (and wlog) that each individual bit of X has a uniform marginal distribution over $\{0, 1\}$. That way, the pseudodistribution $\Delta_{i\to j} := U_1 \otimes X_{i+1\to j} - X_{i\to j}$ can be equivalently written as $\Delta_{i\to j} = X_{i\to i+1} \otimes X_{i+1\to j} - X_{i\to j}$. Consequently, after expanding, the pseudodistribution $X^{(m)}$ has the form

$$X^{(m)} = \sum_{k=1}^{K} \mu_k \cdot X^{(m,k)},$$

where $K = O(n)^m$, $\mu_k \in \{\pm 1\}$, and each $X^{(m,k)}$ is a distribution of the form

$$X^{(m,k)} = X_{0\to i_0} \otimes \cdots \otimes X_{i_{\ell-1}\to i_\ell}$$

for some $\ell \le 2m$ and some $0 \le i_0 < i_1 < \cdots < i_\ell = n$.

Let $\Sigma = \{0, 1\}^s$ be the domain of G_{Nisan}. The distribution $X^{(m,k)}$ can be written as

$$X^{(m,k)} = (G_{\text{Nisan}}(R_0)_{0\to i_0}, G_{\text{Nisan}}(R_1)_{i_0\to i_1}, \dots, G_{\text{Nisan}}(R_\ell)_{i_{\ell-1}\to i_\ell}),$$

where $R_0, \dots, R_\ell \in \Sigma$ are sampled independently and uniformly at random. A key observation is that we can write

$$f(X^{(m,k)}) = g(R_0, \dots, R_\ell),$$

where g is a width-w length-$(\ell + 1)$ ROBP over the alphabet Σ. We can fool such programs with a short seed:

Theorem 3.10 (Fooling ROBPs over a large alphabet)**.** *For every $w, n, s \in \mathbb{N}$ and $\delta \in (0,1)$, there exists an explicit PRG that fools width-w length-m ROBPs over the alphabet $\{0,1\}^s$ with error δ and seed length*

$$O(s + \log(wm/\delta) \cdot \log m).$$

Proof idea. It is straightforward to generalize the INW generator to the large-alphabet setting. □

Proof sketch of Theorem 3.2. We use the pseudodistribution $X^{(m)}$, except that instead of sampling $R_0, \dots, R_\ell$ independently, we use the PRG from Theorem 3.10 with $\delta = \varepsilon/K$ to sample them. The overall seed length of the WPRG is therefore

$$O(\log K + s + \log(wm/\delta) \cdot \log m) = \widetilde{O}(\log(wn) \cdot \log n + \log(1/\varepsilon)).$$

By the triangle inequality, sampling $R_0, \dots, R_\ell$ pseudorandomly only creates $K\delta = \varepsilon$ additional error. □

References

[AKMPSV20] AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. "High-precision estimation of random walks in small space". In: *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1295–1306. DOI: 10.1109/FOCS46700.2020.00123.

[BCG20] Mark Braverman, Gil Cohen, and Sumegha Garg. "Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs". In: *SIAM J. Comput.* 49.5 (2020), STOC18–242–STOC18–299. ISSN: 0097-5397. DOI: 10.1137/18M1197734.

[CDRST21] Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. "Error Reduction for Weighted PRGs Against Read Once Branching Programs". In: *Proceedings of the 36th Computational Complexity Conference (CCC)*. 2021, 22:1–22:17. DOI: 10.4230/LIPIcs.CCC.2021.22.

[PV21] Edward Pyne and Salil Vadhan. "Pseudodistributions That Beat All Pseudorandom Generators (Extended Abstract)". In: *Proceedings of the 36th Computational Complexity Conference (CCC)*. 2021, 33:1–33:15. DOI: 10.4230/LIPIcs.CCC.2021.33.