**Noname manuscript No.**
(will be inserted by the editor)

# On protocols for increasing the uniformity of random bits generated with noisy quantum computers

**Elías F. Combarro · Federico Carminati · Sofia Vallecorsa · José Ranilla · Ignacio F. Rúa**

Received: date / Accepted: date

**Abstract** Generating random numbers is important for many real-world applications, including cryptography, statistical sampling and Monte Carlo simulations. Quantum systems subject to a measurement produce random results via Born's rule, and thus it is natural to study the possibility of using such systems in order to generate high-quality random numbers. However, current quantum devices are subject to errors and noise, which can make the output bits deviate from the uniform distribution. In this work, we propose and analyse two protocols that can be used to increase the uniformity of the bits obtained when running a circuit with a Hadamard gate and a measurement in a noisy quantum computer. These protocols may be used prior to other standard processes, such as randomness amplification. We conduct experiments on both a quantum simulator and a real quantum computer, obtaining results that suggest that these protocols are useful to improve the probability of the generated bits passing statistical tests for uniformity.

**Keywords** Random bits · Quantum computers · Monobit test · Uniform distribution

Elías F. Combarro
Department of Computer Science, University of Oviedo, Spain
Openlab, CERN, Switzerland
E-mail: efernandezca@uniovi.es

Federico Carminati
Openlab, CERN, Switzerland
E-mail: Federico.Carminati@cern.ch

Sofia Vallecorsa
Openlab, CERN, Switzerland
E-mail: Sofia.Vallecorsa@cern.ch

José Ranilla
Department of Computer Science, University of Oviedo, Spain
E-mail: ranilla@uniovi.es

Ignacio F. Rúa
Department of Mathematics, University of Oviedo, Spain
E-mail: rua@uniovi.es

## 1 Introduction

Random numbers of good quality are vital for practical applications in fields as diverse as cryptography [23], probabilistic algorithms [17] and stochastical and Monte Carlo simulations [3,12]. This need has been traditionally fulfilled with the use of either pseudorandom number generators [19,16] or physical sources of random bits [4,21,29]. In the last decades, the use of quantum information technologies, including quantum computers, has emerged as an alternative way of generating random numbers [14,24,18].

Randomness is embedded in the theory of quantum physics [6], where the outcome of most phenomena cannot be completely determined and only the probabilities of different results can be given. This unpredictability is considered by many as an unavoidable property of physical reality [26] and, as such, can be exploited to design processes to obtain perfect random bits that can be used even in the most demanding applications.

The most simple way of generating random bits by using quantum phenomena is to consider a quantum system which is capable of being in two different states and prepare it in an uniform superposition of those two basic states. Upon measurement, the system will collapse in one of the two states with exactly the same probability in virtue of Born's rule [7]. Theoretically, then, such a system is a perfect source of random bits.

In practice, however, there are many technical difficulties that make it very hard to implement such a system and the resulting source of bits can be not completely uniform because of noise and errors [30,10,9]. A possible way to overcome these problems is the use of different random sources in combination with randomness extractors [27] to amplify the uniformity of the resulting bits. This, however, usually involves the use of several different quantum systems and somehow complicated protocols [2,22].

In this paper, we focus on practical ways of improving the uniformity of random bits generated with quantum computers while keeping the resources needed on the bare minimum: a quantum computer with a single qubit and no additional sources of randomness. We introduce two different families of protocols that can be used with any circuit-based quantum computer and conduct experiments on both a quantum simulator and on one of the quantum computers of the IBM Quantum Experience [15]. The results show that, in the presence of noise and hardware imperfections, these methods can greatly improve the probability of passing the standard tests for uniformity when compared with the straightforward quantum circuit for random bit generation.

The rest of the paper is organised as follows. In Section 2, we present the basic concepts of quantum computing that are needed to understand the protocols proposed in this work. In Section 3, we introduce the first protocol, that takes into account the measurement errors of current computers and try to mitigate their effects, while in Section 4 we propose another protocol that is less demanding in the number of quantum circuits that have to be executed in order to obtain a string of random bits. Section 5 is devoted to explaining the settings of the experiments, both on the quantum simulator and on actual quantum hardware, and to presenting and discussing their results. In that section, we also propose some variants of the protocols. Finally, in Section 6 we raise some conclusions and propose ideas for future work.

## 2 Generating random bits with a quantum computer

The field of quantum computing [25] studies the possibility of using quantum effects such as superposition, entanglement and interference to perform computational tasks in a way that is advantageous (for instance, asymptotically faster) over what is possible with classical algorithms and computers. Examples of quantum algorithms that are better than their classical counterparts (or, at least, than the best classical algorithms that we currently have) include Shor's algorithms for factoring integers and computing discrete logarithms [28], and Grover's method for searching in an unsorted database [13].

In quantum computing, the basic unit of information is the quantum bit or qubit, which is physically implemented by means of a quantum system capable of being in two different states. These states are usually denoted in Dirac notation by $|0\rangle$ and $|1\rangle$, as a counterpart to the values 0 and 1 of a classical bit. Mathematically, a qubit is represented by a normalized vector in a Hilbert space of dimension 2 with orthonormal basis $\{|0\rangle, |1\rangle\}$ (which is usually called the computational basis). Thus, a qubit $|\varphi\rangle$ has the form

$$|\varphi\rangle = \alpha |0\rangle + \beta |1\rangle$$

where $\alpha, \beta$ are complex numbers, called amplitudes, such that $|\alpha|^2 + |\beta|^2 = 1$.

The actual state of a qubit cannot be observed directly, but when it is measured (in the computational basis) we obtain $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. This is usually called Born's rule and relies on the normalization of the qubit state so that the probabilities add up to 1. After the measurement, the qubit loses its previous state and collapses to the basis state that was obtained, meaning that if we do not further act on the qubit, subsequent measurements will always produce the same result.

The physical transformations that the state of a qubit (and, in general, a quantum system) can suffer are given by linear functions that preserve the normalization of the state. Thus, they can be represented in the computational basis as unitary matrices, that is, complex matrices $U$ such that its conjugate transpose $U^\dagger$ is also its inverse, i.e., $U \cdot U^\dagger = U^\dagger \cdot U = I$, where $I$ denotes the identity matrix.

Any such matrix is a possible transformation that can act on the state of a qubit and, in the computational model of quantum circuits, it is usually called a quantum gate. The quantum circuit model is the most used used in quantum computing and the one that the great majority of current quantum computers implement. In it, one or more qubits, initially in the state $|0\rangle$, are acted upon by quantum gates and, at the end of the computation, measured to obtain a result.

Although there exist gates that can act on several qubits at once, in this paper we will work with just 1-qubit gates. Among them, the most important ones are the Pauli gates $X, Y$ and $Z$, the exponentials of these Pauli gates $R_X(\theta), R_Y(\theta)$ and $R_Z(\theta)$ and the Hadamard gate $H$. These gates are given, in the computational basis, by the following matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$R_X(\theta) = e^{-i\frac{\theta}{2}X} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$R_Y(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$R_Z(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$

We will also consider a quantum gate derived from $R_Y(\theta)$ that we will denote by $A(\theta)$, which is given by

$$A(\theta) = R_Y(2\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Notice that, when restricted to real-valued amplitudes, the action of $A(\theta)$ is just a counterclockwise rotation of angle $\theta$.

With these definitions, now it is easy to see how a simple quantum circuit can (theoretically) lead to the generation of perfect random bits. We start with a qubit in state $|0\rangle$, we apply the Hadamard gate $H$ to obtain

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

and we measure the qubit. The probability of obtaining $|0\rangle$ is $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$ which is equal to the probability of obtaining $|1\rangle$.

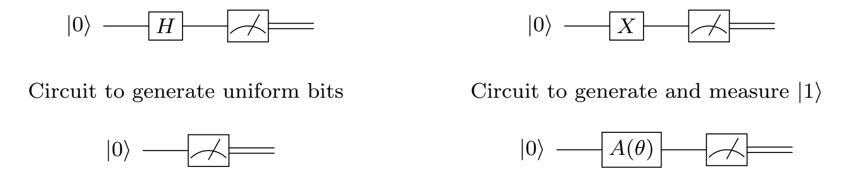
This is usually represented by a quantum circuit like the one in Figure 1a. In this kind of representation, each wire is a qubit and the different gates are drawn insides boxes and applied from left to right. The initial state of each qubit (usually $|0\rangle$) is written at the beginning of each wire and the meter inside a box is not a quantum gate but denotes a measurement of the corresponding qubit.

Thus, a simple circuit like the one in Figure 1a which can be executed by any circuit-based quantum computer, is all that is theoretically needed to sample from a uniform distribution of bits. However, it has been noted [30, 10, 9] that, in practice, current quantum computers executing such a circuit can produce sequences that are far from being uniform. Our goal in this paper is to present some simple, practical protocols that can be used with current quantum computers to increase the uniformity of random bits generated with circuits similar to the one in Figure 1a. We describe them in the following two sections.

## 3 First protocol

In theory, the use of the circuit of Figure 1a produces perfect random bits when run on an ideal quantum computer. However, current quantum computers are subject to noise and errors in the application of the gates. Actually, practical experiments show that in real life the results of using such a procedure can be far from uniform [30, 10, 9].

One important source of this deviation is the presence of measurement errors [9]. The implementation of measurements in current quantum computers is



Circuit to generate uniform bits — Circuit to generate and measure $|1\rangle$

Circuit to generate and measure $|0\rangle$ — Circuit to account for measurement errors

Fig. 2: Quantum circuits used in our protocols

imperfect and it can sometimes happen that the system is in one basic state but a different one is measured. We will denote by $e_0$ the probability of measuring $|0\rangle$ when the real state of the system is $|1\rangle$, and by $e_1$ the probability of measuring $|1\rangle$ when the system is in state $|0\rangle$. On the quantum computer of the IBM Quantum Experience [15] that we will be using for our experiments in Section 5, $e_1$ is typically between 0.03 and 0.04 and $e_0$ around 0.06 or 0.07 (but can, sometimes, be as high as 0.13). This is considerably higher than the errors introduced by the 1-qubit quantum gates, which are usually below 0.001 and, sometimes, as low as 0.0001. Obviously, these measurement errors greatly affect even simple circuits like the one in Figure 1a, making the output noticeably diverge from uniformity.

In this section, we propose a simple protocol that uses the values $e_0$ and $e_1$ to modify the circuit in Figure 1a in order to try to make the output closer to uniformity in actual executions. The values of $e_0$ and $e_1$ can be estimated with simple circuits and then those values can be used in the protocol. Namely, to estimate $e_0$, we can use the circuit in Figure 1b. The use of the $X$ gate sets the state of the qubit to $|1\rangle$ so, when we perform the measurement, we should always obtain $|1\rangle$ as a result. If we execute this circuit a number of times $N$ and we count the number of times $N_0$ that we obtain $|0\rangle$ as a result, we can estimate $e_0$ by $\frac{N_0}{N}$. Similarly, we can execute the circuit in Figure 1c $N$ times, count the number of times $N_1$ that the result is $|1\rangle$ and use $\frac{N_1}{N}$ to estimate $e_1$.

Notice that

$$A\left(\frac{\pi}{4}\right)|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\\ 1 & 1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = H|0\rangle$$

Thus, since all the circuits we will work with start with qubits in state $|0\rangle$, we can consider the application of the $H$ gate as an application of the $A(\theta)$ rotation with a $\frac{\pi}{4}$ angle. The main idea of the protocol is to use the values of $e_0$ and $e_1$ (or their estimations) to correct the value of $\theta$ in the $A(\theta)$ rotation so that the joint action of the rotation and the measurement (taking into account the possible errors) is equivalent to a $\frac{\pi}{4}$-rotation and then execute the circuit in Figure 1d with that choice of $\theta$.

To determine the value of $\theta$ that we should use, we will make a simplified assumption. We will suppose that, when executing the circuit of Figure 1d, the measurement error occurs after the state has collapsed to either $|0\rangle$ or $|1\rangle$ and that then it has a probability (given by $e_0$ and $e_1$) of flipping to the opposite state, leading to the final result that we observe.

**Algorithm 1** Protocol 1

```
Input: $N, L$ non-negative integers
Output: A sequence of $L$ bits
$N_0 \leftarrow 0$
for i:=1 to $N$ do
    Run the circuit of Figure 1b
    if the result is $|0\rangle$ then
        $N_0 \leftarrow N_0 + 1$
    end if
end for
$e_0 \leftarrow \frac{N_0}{N}$
$N_1 \leftarrow 0$
for i:=1 to $N$ do
    Run the circuit of Figure 1c
    if the result is $|1\rangle$ then
        $N_1 \leftarrow N_1 + 1$
    end if
end for
$e_1 \leftarrow \frac{N_1}{N}$
$\theta \leftarrow \arccos\sqrt{\frac{1-2e_0}{2(1-e_0-e_1)}}$
for i:=1 to $L$ do
    Run the circuit of Figure 1d
    if the result is $|0\rangle$ then
        Output 0
    end if
    if the result is $|1\rangle$ then
        Output 1
    end if
end for
```

Then, the probability of obtaining $|0\rangle$ from the execution of the circuit in Figure 1d is given by

$$\cos^2(\theta) \cdot (1 - e_1) + \sin^2(\theta) \cdot e_0$$

that is, the probability of measuring $|0\rangle$ given that the actual state was $|0\rangle$ plus the probability of measuring $|1\rangle$ given that the actual state was $|1\rangle$.

We want this probability to be $\frac{1}{2}$, from where we should choose $\theta$ to be

$$\theta = \arccos\sqrt{\frac{1-2e_0}{2(1-e_0-e_1)}}$$

Notice that if $e_0 = e_1 = 0$, then the value of $\theta$ is $\frac{\pi}{4}$, as expected.

Once we have computed $\theta$ from the estimations of $e_0$ and $e_1$, we run the circuit of Figure 1d $L$ times and output the bits that we measure.

In summary, the protocol we propose is the one given in Algorithm 1.

## 4 Second protocol

In the previous section, we have proposed a protocol to correct the angle of rotation from the $\frac{\pi}{4}$ of the circuit of Figure 1a to a value $\theta$ for the circuit of Figure 1d that needs a priori estimations of the probabilities $e_0$ and $e_1$ of measurement error.

This approach has the drawback of requiring the execution of additional circuits (those in Figures 1b and 1c) to obtain such estimations.

In this section, we take an alternative approach. We run the circuit of Figure 1d with some value of $\theta$ and then use the outputs of the circuit to determine a new value $\theta'$ that, hopefully, will provide closer to uniform bits. The idea is that, because of the measurement and gate errors, when we run the circuit of Figure 1d with an angle $\theta$ instead of obtaining $|0\rangle$ with probability $\cos^2(\theta)$ and $|1\rangle$ with probability $\sin^2(\theta)$, we obtain $|0\rangle$ with some probability $p_0$ and $|1\rangle$ with some probability $p_1$. Since $p_0 + p_1 = 1$, there exists an angle $\theta_{real}$ such that $p_0 = \cos^2(\theta_{real})$ and $p_1 = \sin^2(\theta_{real})$ and we can assume that, instead of running the circuit of Figure 1d with angle $\theta_{old} = \theta$, we are running it with angle $\theta_{real}$. If we estimate $\theta_{real}$, then the value

$$\theta_{new} = \theta_{old} + \left(\frac{\pi}{4} - \theta_{real}\right)$$

would give us an angle that takes into account the actual bias of the quantum computer in which we are running the circuit in order to obtain a distribution with equal probability of measuring $|0\rangle$ and $|1\rangle$.

Since we know that $p_0 = \cos^2(\theta_{real})$ and $p_1 = \sin^2(\theta_{real})$, in order to estimate $\theta_{real}$ we run the circuit of Figure 1d with some $\theta$ $L$ times, count $L_0$ the number of times we measure $|0\rangle$ and $L_1$ the number of times that we measure $|1\rangle$ and estimate $\theta_{real}$ by using

$$\theta_{real} = \arctan\sqrt{\frac{L_1}{L_0}}$$

Notice that, in order to determine the new angle $\theta_{new}$, we need the values $\theta_{old}$, $L_0$ and $L_1$. Once we have the protocol running, we will use the values from the previous sequence. The initial setup will be $\theta_{old} = \frac{\pi}{4}$ and $L_0 = L_1 = \frac{L}{2}$. This can be seen as a kind on Bayesian-line process in which we have a prior belief (at the beginning, we do not have any reason to suppose that the actual quantum computer deviates from uniformity in one direction more than in the other), and we use the results we obtain to update our posterior believes.

The whole protocol is summarized in Algorithm 2. Notice that, in contrast with the first protocol, here we generate $K$ sequences of $L$ bits (and not just one). Note also that, in order to do that, we run exactly $K \cdot L$ circuits, while in Protocol 1 we need to run $2N + L$ circuits to generate just $L$ bits.

## 5 Experiments

To test the effectiveness of the protocols proposed in Sections 3 and 4, we have conducted millions experiments on quantum simulators and several thousands experiments on actual quantum computers. In the following subsections, we describe the settings of the experiments and how we propose to evaluate the performance, and we present and discuss the results we have obtained.

**Algorithm 2** Protocol 2

```
Input: $L, K$ non-negative integers
Output: $K$ sequences of $L$ bits
$L_0 \leftarrow \frac{L}{2}$
$L_1 \leftarrow \frac{L}{2}$
$\theta_{old} = \frac{\pi}{4}$
for i:=1 to $K$ do
    $\theta_{real} = \arctan\sqrt{\frac{L_1}{L_0}}$
    $\theta_{new} = \theta_{old} + (\frac{\pi}{4} - \theta_{real})$
    $L_0 \leftarrow 0$
    $L_1 \leftarrow 0$
    for j:=1 to $L$ do
        Run the circuit of Figure 1d with $\theta = \theta_{new}$
        if the result is $|0\rangle$ then
            Output 0
            $L_0 \leftarrow L_0 + 1$
        end if
        if the result is $|1\rangle$ then
            Output 1
            $L_1 \leftarrow L_1 + 1$
        end if
    end for
end for
```

## 5.1 Settings of the experiments and evaluation

We have implemented the protocols described in previous sections using IBM's Qiskit quantum programming language [1]. This allowed us to run them on both quantum simulators and on actual quantum hardware from the IBM Quantum Experience [15]. Among the available physical quantum devices, we have selected IBM Armonk which, as the rest of devices available on the platform, uses superconducting transmon technology and supports general rotation gates. This computer has just one qubit, which makes it unsuitable for running most quantum algorithms. However, it is perfect for our protocols, since we don't need more than one qubit and, in addition, we avoid crosstalk from other qubits that could introduce extra noise.

Since the objective of our protocols is increasing the uniformity of the individual generated bits, to evaluate the performance we have used the monobit or frequency test from the NIST Test Suite for Random and Pseudorandom Number Generators [5]. This test checks whether the proportion of 0's and 1's in a string of bits follows that of a sequence of bits generated with a perfect uniform distribution. This is the most basic test for uniformity and, as noted in [5], "all the subsequent tests depend on the passing of this test". This test is also part of some other popular test suites such as Dieharder [8] or TestU01 [20]. The test works as described in Algorithm 3.

The quantum computing paradigm is probabilistic instead of deterministic, and thus each execution of the same circuit can lead to different results. Thus, in each execution on the computers of the IBM Quantum Experience one of the parameters is the number of repetitions or shots. In all our experiments, we have selected the number of shots to be 8192, which is the maximum allowed. Thus, from each circuit that we executed on either the quantum simulator or on IBM

**Algorithm 3** Monobit test

```
Input: A string $s$ of $N$ bits
Output: "YES" if the test is passed, "NO" otherwise
$N_0 \leftarrow$ number of 0's in $s$
$N_1 \leftarrow$ number of 1's in $s$
$D \leftarrow \frac{|N_0 - N_1|}{\sqrt{2N}}$
$p \leftarrow \frac{2}{\sqrt{\pi}} \int_D^\infty e^{-x^2} dx$
if $p \geq 0.01$ then
    Output "YES"
else
    Output "NO"
end if
```

Armonk we obtained a binary string of length 8192 and, consequently, we set $N = 8192$ in Algorithm 3.

### *5.1.1 Settings of the experiments on the quantum simulator*

As a first way of validating the protocols introduced in Sections 3 and 4, we have run them on the quantum simulator provided with Qiskit. The use of the simulator is much faster than executing the protocols on actual quantum hardware but it also allows the inclusion of models of noise that mimic the behaviour of the actual quantum computers.

For our experiments, we have chosen to run first the circuit of Figure 1a in perfect conditions, that is, with no noise. Then, we have introduced a noise model taken from IBM Armonk, the real quantum computer that we will be using in the second part of our experimentation. In this setup, for the gate errors we have used the values given by the model retrieved from the actual quantum computer (which were all around $10^{-4}$), and then we have considered three different settings for the readout errors: $e_0 = 0.065$ and $e_1 = 0.035$, $e_0 = 0.07$ and $e_1 = 0.03$, and $e_0 = 0.06$ and $e_1 = 0.04$. These figures are typical values that can be observed on IBM Armonk, and they capture different situations with more or less asymmetry. The case with $e_0 = 0.06$ and $e_1 = 0.04$ should be the more favorable for the circuit of Figure 1a to produce bit sequences that pass the frequency test, while the case $e_0 = 0.07$ and $e_1 = 0.03$ should make it much more difficult. We have chosen these different values to test the behaviour our protocols under a range of situations.

In all the settings, we have run the circuit of Figure 1a 8192 times (to be consistent with the maximum number of shots of the actual quantum computer) and then performed the monobit test on the output bit sequence. We have repeated the process $2^{20}$ times (that is, we have tested more than one million bit sequences of length 8192). For Protocol 1, we have selected $N = L = 8192$ and we have, again, run the protocol $2^{20}$ times. For Protocol 2, we have selected $K$ to be 128 and $L$ to be 8192 and we have run the whole protocol $2^{13}$ times, again obtaining a total of $2^{20}$ bit sequences of length 8192.

### *5.1.2 Settings of the experiments on the actual quantum computer*

To establish a baseline for the IBM Armonk quantum computer, we have run the circuit of Figure 1a without any modification or correction for a little more than 10,000 times (each execution consists of the maximum number of shots, which, as mentioned, is 8192). Then, we have run Protocols 1 and 2 exactly as described in Sections 3 and 4 for more than 1,000 times each. We have also run some modifications of these protocols, described in the following list:

- **Variants of Protocol 1**:
  - **Original**: The one described in Section 3, with no modification, and setting $N = L = 8192$ (the maximum number of shots).
  - **Calibration**: IBM provides calibrations of their quantum computers, including values of the measurement errors $e_0$ and $e_1$. We use these calibrations instead of estimating $e_0$ and $e_1$ ourselves.
  - **Initial**: To decrease the number of of times we need to run the circuits of Figures 1b and 1c, we perform just one initial estimation of $e_0$ and $e_1$ and then execute Protocol 1 with those values of $e_0$ and $e_1$ 128 times (so that we get $8192 \cdot 128 = 2^{20}$ bits)
  - **Initial 10**: Similar to **initial**, but instead of running the circuits of Figures 1b and 1c just once, we run each of them 10 times and take $e_0$ and $e_1$ as the average of the individual estimations. Then, we run Protocol 1 with those values of $e_0$ and $e_1$ 128 times.
  - **Blocks**: Similar to **initial**, but instead of 128 times, we run Protocol 1 with the estimated values of $e_0$ and $e_1$ just 8 times.
  - **Accumulated**: We run Protocol 1 128 times, but instead of performing an independent estimation of $e_0$ and $e_1$ each time, we accumulate the values of the estimations and use their average for each execution of the circuit of Figure 1d.
  - **Reject**: We run Protocol 1 128 times, each time conducting the monobit test with the 8192 bits of the output. If the test is passed, we keep the values of $e_0$, $e_1$ and $\theta$ for the next sequence and we do not run the circuits of Figure 1b and 1c. If the test fails, then we run the circuits to estimate $e_0$ and $e_1$ (we *reject* the previous values of $e_0$ and $e_1$).
- **Variants of Protocol 2**:
  - **Original**: The one described in Section 3, with no modification, and setting $K = 128$ and $L = 8192$
  - **Reject**: After each execution of the inner *For* loop of Algorithm 2, we run the monobit test with the 8192 bits just obtained. If the test is passed, we do not update the value of $\theta$ (that is, we keep $\theta_{new} = \theta_{old}$). If the test fails, we update the value of $\theta$ as in the original protocol.
  - **Half**: Exactly as the **original** protocol, but with the update $\theta_{new} = \theta_{old} + \frac{\frac{\pi}{4} - \theta_{real}}{2}$.
  - **Reject+Half**: This is a mixture of both **reject** and **half**. We run the monobit test with each sequence of 8192 bits. If the test is passed, we keep the value of the angle unchanged. If it fails, we update it with $\theta_{new} = \theta_{old} + \frac{\frac{\pi}{4} - \theta_{real}}{2}$.

The main goal of introducing these variants is, in the case of Protocol 1, to try to reduce the number of extra circuits that we need to run while keeping the ability to correct the angle with estimations of the measurement errors. In the case of Protocol 2, the variants aim to prevent the algorithm from overcompensating when one of the runs is far from uniform (see the explanation in Section 5.2 and also Figures 3a and 3b).

Table 1: Number of 8192-bit sequences obtained with each variant

| **Protocol** | **Number of sequences** |
|---|---|
| Baseline (Hadamard + measurement) | 10897 |
| Protocol 1 - Original | 1729 |
| Protocol 1 - Calibration | 1167 |
| Protocol 1 - Initial | 1198 |
| Protocol 1 - Initial 10 | 1033 |
| Protocol 1 - Blocks | 1470 |
| Protocol 1 - Accumulated | 1620 |
| Protocol 1 - Reject | 1232 |
| Protocol 2 - Original | 2257 |
| Protocol 2 - Reject | 1192 |
| Protocol 2 - Half | 1303 |
| Protocol 2 - Reject + Half | 1028 |

As mentioned above, we have run these variants at least one thousand times each. Although the access to the quantum computers in the IBM Quantum Experience is free, the jobs enter a priority queue and can be subject to a considerable delay. Also, the computers are sometimes down or go for maintenance or calibration, causing some of the executions to abort before completion. For these reasons, the experiments presented in this paper took almost three months to complete. Since some of the variants were introduced later on to improve the effectiveness of the methods, not every protocol could be run the same number of times. In any case, every variant was used to produce at least 1000 sequences. The total number of sequences of 8192 bits that we obtained with each of the variants is presented in Table 1.

## 5.2 Results and discussion

In this section, we first present the results of the experiments performed with the simulators and, then, those obtained with the IBM Armonk quantum computer. In both cases, we have computed the proportion of the sequences that pass the monobit test and also the average and the standard deviation of the p-values of those tests. These latter figures of merit are interesting because they complement the information of how many tests have been passed to include an estimation of the margin with which those tests have been passed.

Table 2 shows these values for the simulation experiments. Note, first, that when no noise is present, the circuit of Figure 1a behaves as expected from a perfectly uniform source. About 99% of the tests are passed (consistent with our setting the p-value threshold for a test to be passed at 0.01) and, moreover, the p-values have average and standard deviation compatible with those of a uniform distribution over the interval $[0, 1]$ (cf. [5], Section 4.2.2), namely 0.5 and $\frac{1}{\sqrt{12}} \approx 0.2887$. When we introduce gate and readout errors, however, this circuit is no longer able to reproduce the behaviour of a uniform source. In fact, the number of passed tests quickly drops and the average of the p-values also falls well below 0.5, while their standard deviation also decreases, suggesting that it is more infrequent to pass the tests by a wide margin. This is more remarkable when the difference

Table 2: Results of the simulations

| **Protocol** | **Pr tests** | **Av $p$-values** | **Std $p$-values** |
|---|---|---|---|
| Hadamard + measurement (no noise) | 0.9899 | 0.4999 | 0.2886 |
| H + measurement ($e_0 = 0.065$, $e_1 = 0.035$) | 0.4210 | 0.0483 | 0.1158 |
| Protocol 1 ($e_0 = 0.065$, $e_1 = 0.035$) | 0.9861 | 0.4854 | 0.2927 |
| Protocol 2 ($e_0 = 0.065$, $e_1 = 0.035$) | 0.9403 | 0.4031 | 0.3070 |
| H + measurement ($e_0 = 0.07$, $e_1 = 0.03$) | 0.1344 | 0.0092 | 0.0399 |
| Protocol 1 ($e_0 = 0.07$, $e_1 = 0.03$) | 0.9857 | 0.4847 | 0.2927 |
| Protocol 2 ($e_0 = 0.07$, $e_1 = 0.03$) | 0.9376 | 0.4035 | 0.3072 |
| H + measurement ($e_0 = 0.04$, $e_1 = 0.02$) | 0.7592 | 0.1688 | 0.2322 |
| Protocol 1 ($e_0 = 0.04$, $e_1 = 0.02$) | 0.9859 | 0.4853 | 0.2926 |
| Protocol 2 ($e_0 = 0.04$, $e_1 = 0.02$) | 0.9426 | 0.4044 | 0.3066 |

Table 3: Results on quantum hardware (Protocol 1 + Baseline)

| **Protocol** | **Pr tests** | **Av $p$-values** | **Std $p$-values** |
|---|---|---|---|
| Baseline (Hadamard + measurement) | 0.3962 | 0.0890 | 0.1949 |
| Protocol 1 - Original | 0.6640 | 0.2564 | 0.3073 |
| Protocol 1 - Calibration | 0.2408 | 0.0605 | 0.1681 |
| Protocol 1 - Initial | 0.4482 | 0.1526 | 0.2582 |
| Protocol 1 - Initial 10 | 0.3843 | 0.1122 | 0.2284 |
| Protocol 1 - Blocks | 0.5837 | 0.2174 | 0.2929 |
| Protocol 1 - Accumulated | 0.5389 | 0.1265 | 0.2318 |
| Protocol 1 - Reject | 0.6623 | 0.2580 | 0.3116 |

between $e_0$ and $e_1$ gets bigger, because the asymmetry between 0's and 1's is then more noticeable.

Protocol 1, however, seems to be able to deal with noise and errors in all the situations under study achieving results that are very close to those of the theoretically perfect circuit. Moreover, the results in the three cases are extremely similar, showing that this protocol is stable under a series of different conditions. The main drawback of this method is the necessity of running two extra circuits per random bit generated, which makes it less efficient than Protocol 2. Also, as we show below, these good simulation results do not seem to translate very well to execution on actual quantum hardware.

Protocol 2, on the other hand, does not behave as well as Protocol 1 in the simulations, although it is still able to achieve around 94% of success in the monobit test. This difference with Protocol 1 may come from those initial runs in which Protocol 2 has yet no information about the behaviour of device and needs to accumulate some data to learn an angle that is able to produce sequences with approximately the same number of 0's and 1's. This is achieved, however, even in the most difficult setting (when $e_0 = 0.07$ and $e_1 = 0.03$), improving the chance of passing the frequency test from 0.1344 (Hadamard circuit with no correction) to 0.9376.

Now, we turn to the results obtained with our experiments on the actual quantum computer. In Table 3, we present the results obtained by the simple Hadamard + measurement circuit and by Protocol 1 when run on IBM Armonk. As we can see, the circuit consisting of a Hadamard plus a mesure that in the simulation produced results consistent with those of a uniform source, only passes the monobit test about 40% of the time and their p-value average and standard deviation are quite low. This is explained by the gate errors and the readout errors of the actual quantum computer. Notice that, in fact, these results are quite similar to the ones obtained in the simulations when we use the gate errors retrieved from the actual IBM Armonk and we set the readout errors to $e_0 = 0.065$ and $e_1 = 0.035$.

Table 3 also shows that using Protocol 1 can increase the probability of success on quantum hardware, up to almost two thirds of the time and also leads to a higher average of the p-values and a standard deviation which is more similar to that expected from a uniform source. This increase, however, is much lower than the one we obtained in the simulations. This is the reason why we introduced the different variants of Protocol 1 described in Section 5.1.2. This behaviour is consistent with the difference between results of simulations and those obtained with quantum hardware that have been previously observed in other problems [11]. What is more, this increase in the chances of passing the test is obtained at the cost of executing two extra circuits to generate each random bit (the ones that we need to estimate the errors $e_0$ and $e_1$). This is the second reason for trying to introduce variants with less overhead.

Let us, then, analyse the results obtained with the proposed variants. If we try to get rid of the extra executions by using the device calibrations provided by IBM, we find that the results are not improved and, in fact, the proportion of sequences that pass the test falls to almost a fourth of the total and the average of p-values also falls. This may seem surprising at first, but the explanation is clear: these calibrations are not performed continuously but can be several hours old. The behaviour of the device, however, seems to vary with time (this is also supported by the results of the rest of experiments) so, by the moment we use them, the calibrations may be far off from the actual readout errors of the computer.

If we, on the other hand, perform the estimation of the $e_0$ and $e_1$ errors not as frequently as once per sequence, we observe that the results are not very different from the baseline when we only conduct the estimation at the beginning of a long run of sequences (initial and initial 10 variants. In fact, it only leads to an improvement if we perform the estimation as frequently as one per eight sequences (blocks). Using an average of the estimations (accumulated variant) does not help in obtaining better results than with the original formulation of Protocol 1. This is consistent with what we observed when using IBM calibrations: because of the wait in the queue, several hours will pass between the first and last run of a batch of 128 executions of the protocol. Thus, if we use an average of the $e_0$ and $e_1$ estimations, we are taking into account values that are several hours old and, as argued above, may not reflect the current behaviour of the computer.

The only variant of Protocol 1 that achieves results comparable to the original is the one that we have called *reject.* With this variant, we keep angles that pass the test and only refresh our $e_0$ and $e_1$ estimations when the uniformity conditions are not met. This seems reasonable, for once that we find a parameter that produces uniform enough bits, it makes sense to use it unless we find evidence that it is not working correctly. An advantage of this variant is that it reduces the number of additional circuits that must be run to generate random sequences. With the original protocol, we need to run three circuits to produce one bit. The reject variant fails the test about one third of the times, so we only need to run, in average, $\frac{5}{3}$ circuits to obtain one  bit.

The results of the experiments with Protocol 2 and its variants are presented in Table 4, again with the baseline for quick comparison. We can readily notice that, in this case, all the results improve those obtained when no protocol is used and just the circuit of Figure 1a is run, and, in fact, the results with all the variants of Protocol 2 get probabilities of passing the monobit test that are higher than any of those obtained with Protocol 1 and its variants. The average and standard deviation of p-values are also closer to what we would expect from a perfect random source (although still not as good as those obtained with the simulations). A possible explanation for the better results of Protocol 2 on quantum hardware is that Protocol 1 only focuses on measurement errors as the source of the deviation from the uniform distribution, while Protocol 2 models the behaviour of the system as a whole, allowing to correct the influence of noise and errors of different types.

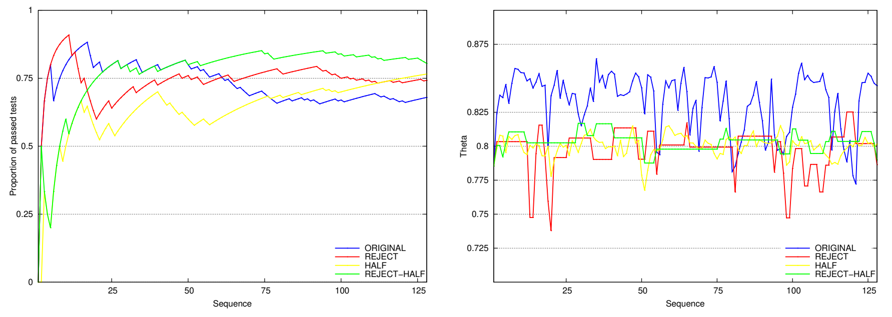
It is also interesting to note that, in this case, the proposed variants always perform better than the original protocol. In contrast with Protocol 1, Protocol 2 is iterative in the sense that each subsequent execution learns from the previous ones and tries to obtain a better approximation of the angle that more uniform bitstrings is able to produce. Then, it again makes sense to not discard too quickly those angles that have been working well in the past. That is precisely the aim of variants reject and half, but with a different approach. The reject variant, as in the case of Protocol 1, keeps those angles that pass the monobit test. The half variant, on the other hand, prevents the protocol to make sudden and abrupt changes in the angle that is used, by reducing by half the amount of increment that is used in each update. As it can be seen from Table 4, this strategy pays off, and leads to an increase in the ratio of frequency tests passed when compared with the original protocol and the reject version. Even better, the improvements of these two variants can be combined to obtain, with the reject + half version, the best performance overall.

An illuminating insight is obtained by analysing the behaviour of both the accumulated ratio of passed tests and the evolution of the angle in Protocol 2 and its variants. In Figure 3a, we present the proportion of passed tests in typical runs of Protocol 2, Protocol 2 - Reject, Protocol 2 - Half and Protocol 2 - Half + Reject from the first to the 128th sequence of an execution. As it can be appreciated, the versions that perform the full angle upgrade (namely, the original and the reject variant) are faster in obtaining a high proportion of passed tests because they adjust the value of $\theta$ more aggressively and it is easier for them to get to an angle that is close to obtaining uniform samples. However, in the long run, they are also more affected by sudden changes and fluctuations in the behaviour of the device. The variants with a more conservative update (half and reject + half) are eventually better because once they identify a good angle is more difficult for them to be driven away from it but they still retain some flexibility and possibility of adapting to changes.

This is even more clearly appreciated in Figure 3b where we present the evolution of the angle for the same runs of Figure 3a. The original version of Protocol 2 shows the highest variability, with a lot of peaks and sudden direction changes. This is somehow mitigated by the reject variant, which shows some plateaus when a good angle (one that produces several passed tests in a row) is reached, but this version is still subject to some abrupt fluctuations. The versions that only perform

Table 4: Results on quantum hardware (Protocol 1 + Baseline)

| **Protocol** | **Pr tests** | **Av** $p$**-values** | **Std** $p$**-values** |
|---|---|---|---|
| Baseline (Hadamard + measurement) | 0.3962 | 0.0890 | 0.1949 |
| Protocol 2 - Original | 0.6832 | 0.2637 | 0.3095 |
| Protocol 2 - Reject | 0.7374 | 0.2820 | 0.3104 |
| Protocol 2 - Half | 0.7698 | 0.3155 | 0.3174 |
| Protocol 2 - Reject + Half | 0.8123 | 0.3143 | 0.3111 |



Proportion of passed tests as the number of sequences increases

Evolution of angle as the number of sequences increases

Fig. 4: Proportion of passed tests and evolution of angle as the number of sequences increases

half correction to the angle are clearly more stable, and seem to oscillate gently around an ideal value of $\theta$. Of the two of them, half + reject shows less peaks and is the more steady overall. This is consistent with the results of Table 4 and of Figure 3a.

## 6 Conclusions and future work

In this paper, we have studied the possibility of generating random bits by executing a circuit consisting of a Hadamard gate and a measurement. Our experimental results show that, although perfectly uniform in theory, in practice this method fails to pass the statistical frequency test most of the times.

To try to mitigate this, we have introduced two simple protocols that can be executed with the most simple quantum resource: just one qubit is sufficient. The first of the protocols focuses on the errors produced in the readout, using additional circuits to estimate these errors and then adjusting the rotation angle in the 1-qubit quantum gate that is used to produce the sequence of bits. This protocol works very well in the experimental simulations that we have conducted, obtaining results that are close to those of the theoretical circuit even in the presence of different noise levels. The success rate of this protocol, however, is not so good when run on actual quantum hardware, so we introduced some variants to, on the one hand, try to reduce the number of additional circuits that need to be run for error estimation and, on the other, aim to obtain higher success rates.

The second protocol presented in the paper is an adaptive algorithm that requires no additional circuits for error estimation but uses the same runs that produce the random bit sequences to learn an angle that makes the underlying quantum computer behave closer to uniformity. As in the case of Protocol 1, we tested this method both on a quantum simulator under different noise situations and on an actual quantum computer. In the former case, the results are not so good as those of Protocol 1, although this new method is still able to cope with different noise levels and obtain success rates that are around 94%. When run on actual quantum hardware, however, Protocol 2 clearly outperforms Protocol 1. What is more, we introduced some variants to the protocol, increasing even more the chance of passing the frequency test and making it more stable.

There are several open questions that we would like to explore in future works. For instance, it would be interesting to study whether these protocols also help in passing other statistical tests that are commonly used to check the uniformity of pseudorandom and true random generators  and if the methods described in the paper offer similar performance on quantum computers with a higher number of qubits. Also, we would like to explore the possibility of combining these protocols with other, more traditional ways of obtaining random bits from quantum computers, such as the use of randomness amplification and expansion. Finally, we would be interested in studying the behaviour of Protocol 2 and its variants in longer runs, once that a good angle has been learned after the initial warm-up period.

**Acknowledgements** This work was supported in part by the Ministry of Economy, Industry and Competitiveness from Spain/FEDER under grant MTM-2017-83506-C2-2-P, by Ministry of Science and Innovation from Spain/FEDER under grant RTI2018-098085-B-C44, and by he Regional Ministry of the Principality of Asturias under grants FC-GRUPIN-IDI/2018/000193 and FC-GRUPIN-IDI/2018/000226.

## References

1. Abraham, H., AduOffei, Akhalwaya, I.Y., et al.: Qiskit: An open-source framework for quantum computing (2019)
2. Acín, A., Masanes, L.: Certified randomness in quantum physics. Nature **540**, 213–219 (2016)
3. Asmussen, S., Glynn, P.: Stochastic Simulation: Algorithms and Analysis. Springer (2007)
4. Bakiri, M., Guyeux, C., Couchot, J.F., Oudjida, A.K.: Survey on hardware implementation of random number generators on fpga: Theory and experimental analyses. Computer Science Review **27**, 135 – 153 (2018)
5. Bassham, L.E., Rukhin, A.L., Soto, J., et al.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, USA (2010)
6. Bera, M.N., Acín, A., Kuś, M., Mitchell, M.W., Lewenstein, M.: Randomness in quantum mechanics: philosophy, physics and technology. Reports on Progress in Physics **80**(12), 124001 (2017)
7. Born, M.: Statistical interpretation of quantum mechanics. Science **122**(3172), 675–679 (1955)
8. Brown, R.G.: Dieharder: a random number test suite. URL https://webhome.phy.duke.edu/ rgb/General/dieharder.php. Last accessed December 2nd, 2020
9. Combarro, E., Carminati, F., Vallecorsa, S., Ranilla, J., Rúa, I.: Quantum random numbers generated by the cloud superconducting quantum computer. Bristol Quantum Information Technologies Workshop (BQIT:20) (2020)

10. Combarro, E., Carminati, F., Vallecorsa, S., Ranilla, J., Rúa, I.: Two simple protocols for improving the uniformity of quantum random bits in the presence of noise. 20th Computational and Mathematical Methods in Science and Engineering Conference (2020)
11. Combarro, E.F., Ranilla, J., Rúa, I.: A quantum algorithm for the commutativity of finite dimensional algebras. IEEE Access **7**, 45554–45562 (2019)
12. Gentle, J.E.: Random Number Generation and Monte Carlo Methods. Springer (2004)
13. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, STOC '96, pp. 212–219. ACM, New York, NY, USA (1996)
14. Herrero-Collantes, M., Garcia-Escartin, J.C.: Quantum random number generators. Rev. Mod. Phys. **89**, 015004 (2017)
15. IBM: IBM Q Experience. URL https://quantum-computing.ibm.com/. Last accessed December 2nd, 2020
16. James, F., Moneta, L.: Review of high-quality random number generators. Comput Softw Big Sci **4**(2) (2020)
17. Karp, R.M.: An introduction to randomized algorithms. Discrete Applied Mathematics **34**(1), 165 – 201 (1991)
18. Kollmitzer, C., Schauer, S., Rass, S., Rainer, B.: Quantum Random Number Generation Theory and Practice: Theory and Practice. Springer (2020)
19. L'Ecuyer, P.: Random number generation. In: J.E. Gentle, W.K. Härdle, Y. Mori (eds.) Handbook of Computational Statistics: Concepts and Methods. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
20. L'Ecuyer, P., Simard, R.: Testu01: A c library for empirical testing of random number generators. ACM Trans. Math. Softw. **33**(4) (2007)
21. Li, L., Yu, F., Tang, Q., Song, Y., Xu, Q., Cai, S.: A Survey on True Random Number Generators Based on Chaos. Discrete Dynamics in Nature and Society **2019**, 1–10 (2019)
22. Liu, Y., Zhao, Q., Li, M.H., et al.: Device-independent quantum random-number generation. Nature **562**(7728), 548–551 (2018)
23. Luby, M.: Pseudorandomness and cryptographic applications. Princeton, NJ: Princeton Univ. Press (1996)
24. Ma, X., Yuan, X., Cao, Z., Qi, B., Zhang, Z.: Quantum random number generation. npj Quantum Information **2**(1), 16021 (2016)
25. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press (2011)
26. Schlosshauer, M., Kofler, J., Zeilinger, A.: A snapshot of foundational attitudes toward quantum mechanics. Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics **44**(3), 222 – 230 (2013)
27. Shaltiel, R.: An introduction to randomness extractors. In: L. Aceto, M. Henzinger, J. Sgall (eds.) Automata, Languages and Programming, pp. 21–41. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
28. Shor, P.: Algorithms for quantum computation: Discrete logarithms and factoring. Proceedings of FOCS pp. 124–134 (1994)
29. Stipčević, M., Koç, Ç.K.: True random number generators. In: Ç.K. Koç (ed.) Open Problems in Mathematics and Computational Science. Springer, Cham (2014)
30. Tamura, K., Shikano, Y.: Quantum random numbers generated by the cloud superconducting quantum computer. International Symposium on Mathematics, Quantum Theory, and Cryptography: Proceedings of MQC 2019 (2019)