Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 7 (2013) 11 – 19

Procedia Technology

The 2013 Iberoamerican Conference on Electronics Engineering and Computer Science

Parallelization of a Monte Carlo Ray Tracing Algorithm for Channel Modelling in Underwater Wireless Optical Communications

V. Guerra*, C. Quintana, J. Rufo, J. Rabadan and R. Perez-Jimenez

Institute for Technological Development and Innovation in Communications (IDeTIC), Parque Científico Tecnológico, Edificio Polivalente II, 2a Planta, University of Las Palmas de Gran Canaria, Las Palmas 35017, Spain

Abstract

In this paper, an algorithm to calculate the underwater wireless optical impulse response is presented. It is based on a modified Monte Carlo Ray Tracing algorithm and takes into account the most significant phenomena of the underwater channel. In order to reduce the simulation time, two parallelization schemes are proposed, one based on a multiprocessor architecture and other based on the use of GPU (Graphics Processing Unit). Several simulation results are presented, including scenario channel simulations and calculation of time computation complexity for each algorithm implementation.

Selection and peer-review under responsibility of CIIECC 2013

Keywords: Underwater Wireless Optical Communications, Impulse Response, Monte Carlo Ray Tracing, CUDA, OpenMP

1. Introduction

Underwater Wireless Optical Communications (UWOC) are a novel application field based on visible light communications (VLC). Traditionally, underwater communications have been based on ultrasounds or low frequency (LF) emissions. In the case of LF, this is the only suitabe radio band to perform submarine transmissions. At this frequency, water presents a less hostile behaviour, but the needed antenna dimensions are usually prohibitive and the achieved bit rate is very small. Regarding ultrasounds, pressure waves have kilometric ranges, allowing low speed links at high distances. However, for certain applications, the transmitted acoustic power may be harmful for the marine fauna.

UWOC is a good alternative to the previous technologies in middle range scenarios where high datarates are required, such as underwater wireless sensor networks (UWSN) or the communication with an unmanned underwater vehicle (UUV) [1]. In UWOC, it has been demonstrated that the best transmission window is green-blue (450 nm - 500 nm), due to the minimum value of the extinction coefficient [2]. The

*Corresponding author. Tel: +34 928459972
Email address: vguerra@idetic.eu (V. Guerra)

Selection and peer-review under responsibility of CIIECC 2013
doi:10.1016/j.protcy.2013.04.002

main characteristic of the underwater channel is its reduced coherence time, namely, the interval where the channel may be considered almost stationary are short. This implies that the invariance assumption can not be properly made, as demonstrated in [3].

In regard to the signals on the receiver, the optical paths followed by the emitted rays, whether from a laser source or a LED source, depend on the variation of the refractive index inside de medium and the presence of different kinds of particles, which generate scattering. As the refractive index depends on the salinity, temperature and wavelength, the pointing between emitter and receiver could be obstructed in turbulent scenarios.

In free space optics (FSO), indoor channel impulse response calculation has been a very important research guideline since the first infrared models proposed by Barry and Khan [4]. This first models were numerically afforded using recursive algorithms, which are very expensive computationally. Later, different simulation strategies based on Monte Carlo schemes were proposed [5] [6], reducing the simulation time. In this work, an impulse response calculation algorithm for UWOC, based on Monte Carlo, is presented. The results of the simulator can be applied to obtain a probabilistic approach of the coverage and the available bandwidth, depending on different parameters so as water turbidity, seawater surface agitation or employed wavelength.

The parallelization of Ray Tracing algorithms has been studied several times, using both multiprocessor and GPU architectures [7]. The obtained speedup and efficiency results have been always very positive, due to the independence between the emitted rays. This paper proposes the parallelization of the UWOC impulse response calculation algorithm using the commented architectures and obtaining speedup and efficiency results too.

2. Channel model

In UWOC, several physical phenomena produced by light-matter interaction occur. There are effects that affect directly the optical power, others modify the propagation direction and finally others change the polarization of the transmitted light. In this work as only medium-speed and short distance channels are taken into account, only LED sources will be considered, neglecting polarization effects. Also, turbulences will be neglected in order to make the analysis easier.

To obtain the channel model, a direct ray (from source to receiver) and a bundle of random rays are generated. N-1 rays are emitted from the LED, where each ray has a polar angle generated with a probability density function (pdf) that follows the lambertian emission pattern of the source. Every emitted ray is then processed taking into account several random factors. For instance, a ray propagating through seawater has a certain probability to collide with a particle depending on the particle density and if a ray impacts with the seawater surface, depending on the agitation, it is reflected (or not) with different angles depending on the surface normal as Snell's law describes.

2.1. Seawater extinction and refractive index

Interaction between light, water molecules and underwater particles produces two effects implying an optical power loss over a given volume: scattering (β) and energy absorption (α). These phenomena depend on the temperature, salinity and wavelength of interest. Each time a ray suffers a collision (either with a particle, or with the surface or the bottom), the carried optical power is weighed by a factor related to the travelled distance (equation 1) and the nature of the collision.

$$L_{ext}(d) = \frac{e^{-c(\lambda)\cdot d}}{\pi d^2} \tag{1}$$

The equation above represents the optical power loss due to the extinction produced by seawater and the spherical propagation loss of every optical link. $c(\lambda) = \alpha(\lambda) + \beta(\lambda)$ is the extinction coefficient at a given wavelength. Table 1 shows the extinction coefficient for different kinds of seawater for the blue-green wavelength.

Water quality and underwater matter concentrations differ from a point to another of the globe. Four main water types are normally considered [8]:

Water type	$\alpha(m^{-1})$	$\beta(m^{-1})$	$c(m^{-1})$
Pure seawater	0.0405	0.0025	0.043
Clear ocean	0.114	0.037	0.151
Coastal ocean	0.179	0.219	0.298
Turbid harbor	0.266	1.824	2.19

Table 1. Absorption, scattering and extinction coefficient for different kinds of seawater

- Pure sea water. The limiting factor is absorption, due to the low density of scattering particles.
- Clear ocean water. The concentration of dissolved particles is higher and the effect of the scattering is also higher.
- Coastal ocean water. This water presents a higher concentration of organic matter, which affects absorption and scattering.
- Turbid harbour water. Very high concentration of dissolved matter.

The seawater refraction index has been deeply studied. Although it can not be defined analytically, there are empirical expressions as those obtained by Mattaus [9] and McNeil [10]. As it depends on the wavelength, the propagation implies chromatic dispersion (as occurs in fiber). This effect will specially affect WLED or RGB sources, as they present a broader spectrum. Furthermore, effects due to the transparent shielding of the light source or receiver, used to prevent from direct contact with seawater will be neglected as only directive sources are going to be considered. Each ray which arrives to the receiver will have an associated delay, corresponding to equation 2.

$$\tau = \frac{d}{v} = \frac{n(\lambda, S, T) \cdot d}{c_0} \tag{2}$$

2.2. Particle interaction

In this work, to model the presence of underwater suspended matter, a uniform distribution of particles has been considered. By this way, the probability of a ray to collide with a particle is defined by the next equation.

$$p_{particle} = 1 - e^{-\mu \cdot d} = 1 - e^{-\rho\sigma \cdot d} \tag{3}$$

Where ρ is the particle concentration and σ is the particle cross section. For big particles, the employed cross section is physical whilst for small particles, the cross section is the electromagnetic cross section obtained from Mie's theory. In a general scenario, there would exist many different kinds of particles, each one with its characteristics (complex refractive index, absorption, dimensions, etcetera), but in this work only two types have been taken into account.

The collision processes for big and small particles can be understood as independent processes, so the total collision probability could be expressed as:

$$p_{collide} = p_{small} + p_{big} - p_{big} \cdot p_{small} \tag{4}$$

Including equation 3, yields:

$$p_{collide} = 1 - e^{-(\mu_{small} + \mu_{big}) \cdot d} \tag{5}$$

The implemented algorithm calculates whether a ray collides with a particle or not, and then, decides if the collision occurred with a big or small particle. This conditioned probability is calculated by the next expressions. Notice that the expression to calculate the probability in the case of small particles is analogue to expression 6

$$p_{big|collide} = \frac{\mu_{big}}{\mu_{big} + \mu_{small}} \tag{6}$$

$$p_{big|collide} + p_{small|collide} = 1 \tag{7}$$

2.3. Surface and bottom model

Seawater surface is, generally, a plane whose normal can not be determined at every instant nor point, because is the superposition of a huge amount of harmonics produced by different phenomena, so as wind effect, Coriolis effect and gravitational attraction. The surface has been modelled as a random variable, whose polar angle follows a normal distribution (zero mean and standard deviation σ_{surf}) and its azimuthal angle is described by a uniform law. After a surface collision, the reflected ray presents a propagation direction which is calculated by the Snell's reflection law and its power is weighed by the Fresnel reflection coefficient.

Regarding the sea bottom, to model the dispersive effect, a lambertian model as the one used by Barry [4] has been employed for simplicity. Although there exist more exact models, a general model has been used because the associated coefficients would imply a well defined knowledge of the sea bottom.

2.4. Integration

After the generation of the initial N rays, each one with its random propagation direction, M rays will arrive to the receiver after several collisions. Furthermore, each ray will have a different delay, because of the different followed optical paths. The next expression shows this.

$$h(t,\lambda) = \sum_{i=1}^{M} P_i \cdot \delta\left(t - \frac{d_i \cdot n(\lambda,S,T)}{c_0}\right) \tag{8}$$

Generally, this impulse response depends on the wavelength, because all the involved parameters depends on it. Technically, the definition of an impulse response implies a linear and temporal invariant assumption. The underwater channel can be considered linear, because there are not polynomic terms in the equations. However, invariance can only be assumed in short time periods, which are normally determined by the data rate of the transmission.

The contribution of each ray can be expressed as:

$$P_i = \frac{P_E}{N} L_{ext} \cdot \prod_{j=1}^{k} L_j \tag{9}$$

Where L_{ext} is the seawater extinction term, k is the number of "hops" of the ray before arriving to the receiver, L_j depends on the type of collision, which are:

- Surface collision. $L_j = \frac{R(\theta_i)}{\pi d_j^2}$
- Bottom collision. $L_j = \rho_{bottom} cos(\theta_i) \frac{cos(\theta)}{\pi d_j^2}$
- Mie scattering. $L_j = \frac{Mie(\theta)}{\pi d_j^2}$
- Receiver arrival. $L_k = \frac{Acos(\varphi)}{\pi d_k^2}$

3. Implementation

Monte Carlo Ray Tracing algorithms are normally implemented using recursive structures in order to reduce the development stage, because after a collision, each ray can be interpreted as a new emitter. Nonetheless, in this work an iterative solution has been afforded, implementing the stack manually because the employed GPU did not support recursivity. The algorithm can be resumed in the following steps:

1. Maximum allowed hops and minimum allowed power. If a ray has made a certain number of hops and has not arrived to the receiver, it is dismissed in order to not diverge the calculation time. If its power is lower than a predefined threshold it is also dismissed.
2. Ray generation. Each ray is generated following the associated probability functions.
3. Travelled distance calculation and event type. At a given scenario, with its associated geometry, the event type decision is made comparing a randomly generated particle collision distance and the distance to the different planes of the geometry (bottom, surface and receiver plane).
4. State update. After the above decision, the state of the ray is updated (direction, power, number of hops, etcetera). If the ray arrives to the receiver, the calculation of the considered ray is done.
5. Generate new random rays if necessary.

3.1. Geometrical considerations

There are some geometrical aspects that should be explained in order to understand the developed algorithm better. The orthogonal cartesian coordinate system has been defined considering the Y-axis as the one which defines the height and the Z-axis as the one parallel to the sea surface.

Spherical coordinates have been used to calculate the scattered, diffused and emitted angles. As the scattering patterns are defined respect to the propagation direction, a vectorial base has been defined for each ray using the Gram-Schmidt method, applying the corresponding base transformations to obtain the actual angles and vectors. Regarding the collision with the sea surface, the reflected ray vector is defined by Snell's law.

Finally, a simulation parameter named MAXHOPS has been defined to establish a limit in the number of collision that a ray can suffer before being dismissed. Considering a big particle free scenario, the collisions with particles will produce new hops. Taking into account that collisions follow an exponential law as shown in expression 3, the average collision distance is $E[d_{col}] = \mu^{-1}$. Hence, considering an scenario where emitter and receiver were at a distance D, the MAXHOPS parameter should be established at value defined by $MAXHOPS \geq \lceil D \cdot \mu \rceil$.

3.2. Parallel implementation

Two different parallel implementations have been studied, apart from the reference sequential implementation. One was made using OpenMP, a multiprocessor based paralellization library, and other was made using CUDA C over a Tesla M2050 GPU [11]. In OpenMP, as each ray behaves independently, the "for" loop which contains the calculation of each ray has been parallelized using the common "parallel for" directive. The CUDA C implementation has been achieved translating the main function to a CUDA C kernel function, which uses "device" defined functions to keep a modular programming paradigm. As CUDA does not define a native random number generator, a Mersenne Twister [12] has been implemented inside the GPU.

The Tesla M2050 GPU architecture is represented in figure 1. The GPU is formed by groups of processors that execute the same instruction at the same time, each one with its own data set. Each "multiprocessor" share a small amount of memory and a special function unit, where transcendent and complex functions are calculated. The NVidia GPU devices operate in a SIMT (Single Instruction Multiple Thread) way, and when a conditional branch is fetched, many processors are disabled in order to maintain the data dependency.

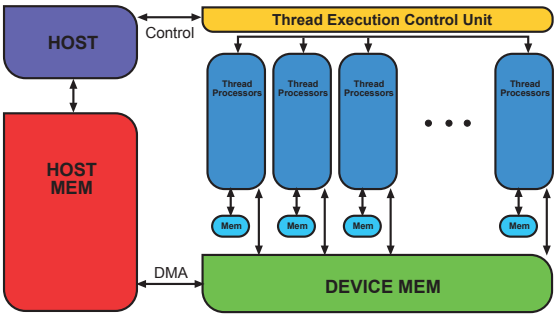

Fig. 1. NVidia Tesla architecture

Lambertian order	1 (pure)	Small particle radius	$8\mu m$
Concentration	$4 \cdot 10^9 parts/m^3$	Distance to the surface/bottom/receiver	$5m$
Number of rays	10^5	Bottom reflectivity	$\rho = 0.5$
Surface agitation	$\sigma = 2.5$	Wavelength	$470nm$

Table 2. Simulation parameters

4. Numerical results

In this section, different simulations are presented to demonstrate the results of the proposed algorithm. Also, speedup and efficiency graphs are depicted. The scenario defined in table 2 has been used as reference to obtain numerical results, considering pure seawater.

Several simulations have been made changing different parameters in order to observe the effect over the impulse response. For instance, figure 2 depicts the broadening of the impulse response with the distance. For a distance of 5 meters, there exist a huge amount of instants where there are not energy contributions, it is due to the lack of simulated rays.

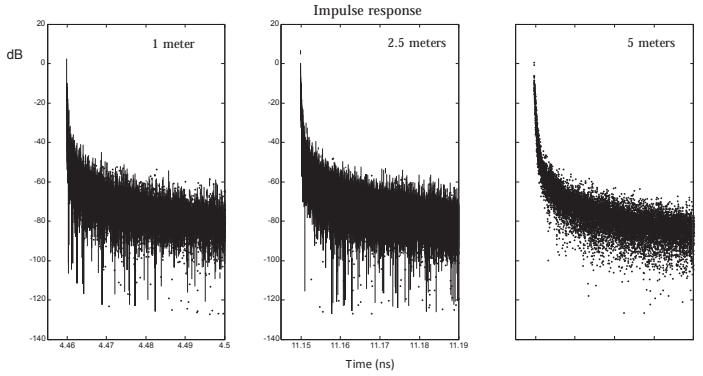

Fig. 2. Variation of the impulse response with the distance

Other effect can be observed in figure 3. In this case, as the lambertian factor of the source increments, the dependency of the delay spread of the impulse response is reduced, as the source is getting more similar to a laser.

Finally, the variation of the impulse response respect of the particle radius has been obtained. As Mie's theory describes, the smaller the particle, the wider the scattering pattern is, and as the particle grows in

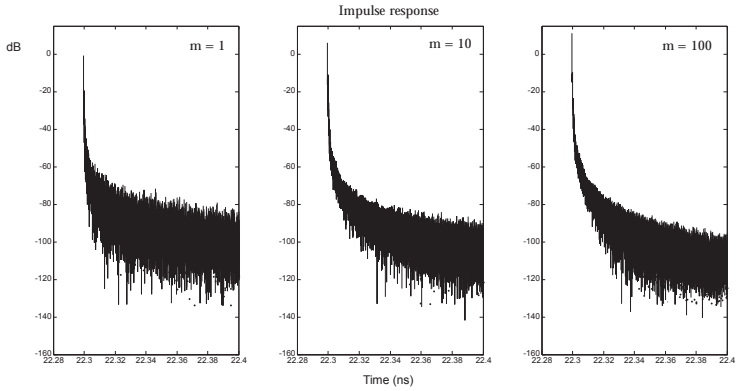

Fig. 3. Variation of the impulse response with the lambertian index of the emitter

size, the propagation direction is less affected. Figure 4 shows this effect.

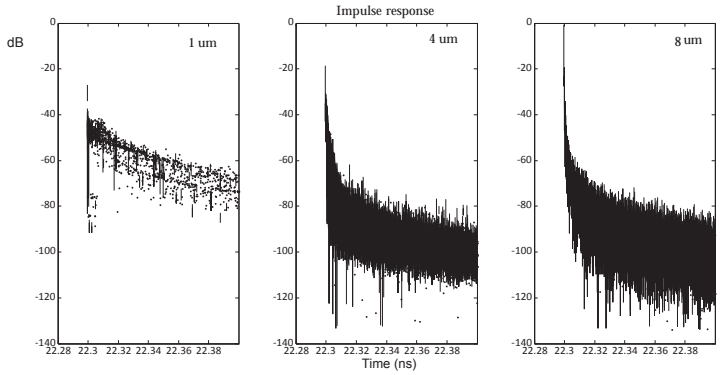

Fig. 4. Variation of the impulse response with the radius of the scattering particles

Regarding the parallelization results, the calculation time of the reference scenario defined in table 2 has been 68381.74 ms approximately.

During the calculation, the generated tree structures associated to each initial ray differ in depth and width, producing an asymmetry in the calculation times and hence, reducing the hardware efficiency. Several schedules and chunk sizes have been tested. The best combination has been dynamic scheduling with chunk size 100. Figure 5 (right figure) shows the results.

In regard of the CUDA C implementation, the used GPU has 448 equivalent cores, but speedups near to the cores number would not be achievable due to the SIMT architecture. The measured loading and saving time of the GPU transactions was 3 ms and it is neglected because it is much smaller than the calculation time. Using the Nvidia CUDA C compiler (NVCC) it is complicated to know the number of actual active processors in order to give a realistic approximation of the efficiency. Therefore, the worst case has been considered for the calculation (448 active cores). CUDA C organizes the computation in thread blocks and threads per block, and the compiler tries to assign a thread block to a hardware block of processors if possible. Figure 5 (left and centre) shows this effect.

It can be observed that the higher the number of threads per block used, the higher the efficiency is, it happens because it has been assumed that all the cores are active in every simulation. Also, an speedup of 40 is achieved using almost all the potential of the GPU. Theoretically, for a fully parallelizable algorithm, an speedup of 448 would be achievable, but in this case, because of the existence of random branches in the trace of the algorithm, this upper bound is not reached. The central graph of the figure 5 shows an efficiency

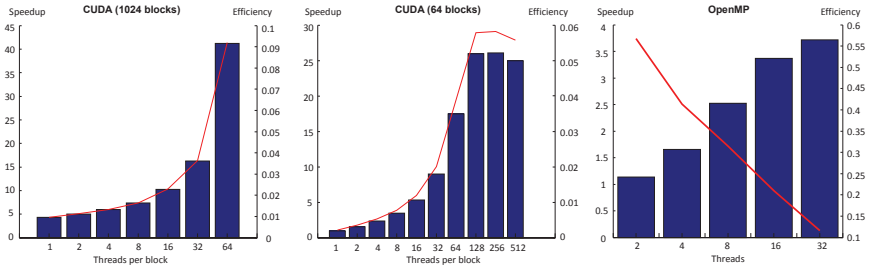

Fig. 5. Speedup and efficiency of the CUDA C (left and centre) and the OpenMP (right) implementations

decrement when more than 128 threads per block are used, this occurs because beyond this point all the available hardware resources are being used. Furthermore, the speedup also suffers a decrement because the thread control unit produces a higher overhead computation time.

5. Conclusions

In this paper, an UWOC impulse response calculation algorithm, based on Monte Carlo Ray Tracing, has been presented. Moreover, its parallel implementation has been made and the speedup and efficiency results have been shown. It has been demonstrated that the efficiency of the GPU implementation depends on the correlation of the actions that the threads than comprise the execution must do. Nonetheless, the absolute speedup values obtained, show that a GPU solution is more effective than a common multiprocessor architecture. The reference scenario, which was sequentially calculated in approximately 1 minute, was parallelly calculated in 1.5 seconds. If an economic efficiency were defined, relating the monetary cost of the GPU and OpenMP implementation to the obtained results, it would be demonstrated that GPU solutions present performances several times greater.

It has been shown that speedups near to the theoretical upper limit are not achievable because of the SIMT computation of the GPU device. This implies that probably, if the simulation strategy were changed making use of different statistic characteristics, a most efficient code would be programmable. For instance, different simulation stages depending on the type of collision events and the number of hops could be considered, taking advantage of the SIMT calculation style of the GPU.

Regarding the channel simulation results, the phenomena that produce most delay spread in the impulse response are the nature of suspended particles and their concentration, the distance between emitter and receiver and the type of emitter. The first three parameters affect directly to the delay spread, whilst the kind of emitter makes the impulse response nearer to a Dirac's delta as its lambertian coefficient increments. There are other effects that have not been considered in this work, such as turbulences, fauna, optical fouling, etcetera. Further research will include these effects in order to generate a better approximation of a real underwater channel.

References

[1] D. Anguita, D. Brizzolara, G. Parodi, Q. Hu, Optical wireless underwater communication for auv: Preliminary simulation and experimental results, in: OCEANS, 2011 IEEE - Spain, 2011, pp. 1 –5. doi:10.1109/Oceans-Spain.2011.6003598.

[2] J. Smart, Underwater optical communications systems part 1: variability of water optical parameters, in: Military Communications Conference, 2005. MILCOM 2005. IEEE, 2005, pp. 1140 –1146 Vol. 2. doi:10.1109/MILCOM.2005.1605832.

[3] J. Simpson, B. Hughes, J. Muth, A spatial diversity system to measure optical fading in an underwater communications channel, in: OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges, 2009, pp. 1 –6.

[4] J. Barry, J. Kahn, W. Krause, E. Lee, D. Messerschmitt, Simulation of multipath impulse response for indoor wireless optical channels, Selected Areas in Communications, IEEE Journal on 11 (3) (1993) 367 –379. doi:10.1109/49.219552.

[5] J. Lopez-Hernandez, R. Perez-Jimenez, A. Santamaria, Modified monte carlo scheme for high-efficiency simulation of the impulse response on diffuse ir wireless indoor channels, Electronics Letters 34 (19) (1998) 1819 –1820. doi:10.1049/el:19981173.

[6] O. Gonzalez, C. Militello, S. Rodriguez, R. Prez-Jimenez, A. Ayala, Error estimation of the impulse response on diffuse wireless infrared indoor channels using a monte carlo ray-tracing algorithm, Optoelectronics, IEE Proceedings - 149 (56) (2002) 222 –227. doi:10.1049/ip-opt:20020545.

[7] B. Zhou, X. Hu, D. Chen, Memory-efficient volume ray tracing on gpu for radiotherapy, in: Application Specific Processors (SASP), 2011 IEEE 9th Symposium on, 2011, pp. 46 –51. doi:10.1109/SASP.2011.5941076.
[8] F. Hanson, S. Radic, High bandwidth underwater optical communication, Appl. Opt. 47 (2) (2008) 277–283. doi:10.1364/AO.47.000277.
URL http://ao.osa.org/abstract.cfm?URI=ao-47-2-277
[9] W. Matthus, in: Empirische Gleichungen fr den Brechungsindex des Meerwassers, no. 33, 1975.
[10] G. T. McNeil, Metrical fundamentals of underwater lens system, in: Opt. Eng, no. 16, 1977, pp. 128 – 139.
[11] NVidia, Nvidia tesla m2050 specification, http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_M2050_M2070_Apr10_LowRes.pdf, consultado el 10 de Julio de 2012.
[12] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. 8 (1) (1998) 3–30. doi:10.1145/272991.272995.
URL http://doi.acm.org/10.1145/272991.272995