On leveraging the chaotic and combinatorial nature of deterministic n-body dynamics on the unit m-sphere in order to implement a pseudo-random number generator

S. Halayka *

February 29, 2012

Abstract

The goal of this paper is to provide a short tutorial on how to implement a chaotic pseudo-random number generator using deterministic n-body dynamics on the unit m-sphere.

1 Nondeterministic and deterministic/chaotic-deterministic physical behaviour

In terms of the laws of physics – as far as how we model them today, anyway – there are two types of ways that a system can evolve over time: nondeterministically and deterministically.

An uncomplicated nondeterministic case is the evolution of an electron's position over time. As per the rules of quantum mechanics, our knowledge of an electron's position is given by a radially symmetric probability function that falls off with increasing radial distance in a nonlinear way. This means we have only a cloudy idea of where the electron is most likely to be found before we actually probe for its true position by using a photon. Upon probing for the electron's position, we would observe that it is most likely fairly close to the centre of the probability cloud, but that the two angles – θ, ϕ of the standard spherical coordinates – are truly and *unpredictably random*. If we were to repeat this probe experiment many times, we would find that there is *absolutely no pattern* in how the two angles that we measure are distributed over time.

An uncomplicated deterministic case is that of the Newtonian gravitational relationship between two bodies of equal mass that are on opposite sides of a shared circular orbit path. This coplanar orbit relationship is *anything but random*, insomuch that it produces prolonged and stable *repetitious cyclical motion*, which in turn forms a *pattern*. If one of the two bodies is perturbed so that it is moved just slightly off of the circular orbit path, then we would find that the two bodies generally readjust to the new location of their common centre of mass, form new – likely elliptical – orbit paths, and once again resume prolonged and stable pattern-forming repetitious cyclical motion.

A complicated deterministic case is that of the Newtonian gravitational relationships between three bodies of equal mass that are equidistantly distributed along a shared circular orbit path. Similar to the uncomplicated case of two bodies, this coplanar orbit relationship is *anything but random*, insomuch that it also produces prolonged *pattern*-forming *repetitious cyclical motion*. Unlike the uncomplicated case however, this case is *unstable*. If one of the three bodies is perturbed so that the three bodies are no longer equidistant, then we would find that this slight asymmetry in the strength of the gravitational interaction amongst the three bodies naturally feeds upon itself over time, causing two of the bodies to speed toward each other at an accelerating rate. This in turn causes the three bodies' common centre of mass to also speed around at an accelerating rate, which generally interrupts any chance that the three bodies will ever again re-establish anything remotely similar to their original prolonged equidistant circular orbit. This devolution from *predictable, repetitious cyclical motion* into *pseudo-random, non-repetitious acyclical motion* due to a slight asymmetry in interaction that feeds upon itself over time can be described as the transition into *chaos*. For more information on the chaotic nature of deterministic many-body dynamics, see [1, 2].

*Little Red River Park, SK Canada – email: shalayka@gmail.com

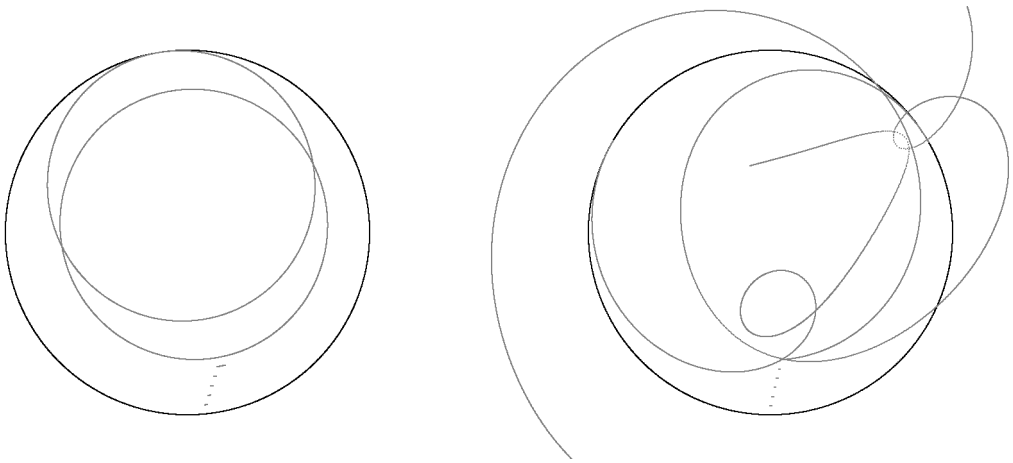

Figure 1: The figure on the left shows a two-body system in circular orbit (black), and then in elliptical orbit (gray) after perturbation. The figure on the right shows a three-body system in circular orbit (black), and then in chaotic orbit (gray) after perturbation.

2 Random and pseudo-random number generators

It is highly recommended that a series of randomly generated numbers be used to pad the beginning and end of a plaintext message that is to be subsequently turned into ciphertext by a robust encryption algorithm such as the AES map or the RSA map. If this padding does not occur, then the encryption algorithm generally produces a result that is not a whole lot more useful than the result given by the non-robust Cæsar substitution cipher (ie. a predictable mapping of symbols such as 'a' $\rightarrow$ 'z', 'b' $\rightarrow$ 'y', 'c' $\rightarrow$ 'x', *etc*).

To obtain a randomly (ie. nondeterministically) generated number, we could probe for the location of an electron and then use the two observed angles θ, ϕ in conjunction to form a single value. This can be done by normalizing each angle

$$\theta' = \frac{\theta}{2\pi}, \tag{1}$$

$$\phi' = \frac{\phi}{\pi}, \tag{2}$$

then converting them from real numbers into integers[1]

$$u = \text{floor}[\theta'(1+\text{maximum integer value})], \tag{3}$$

$$v = \begin{array}{ll} \text{maximum integer value} & \text{if } \phi' = 1 \\ \text{floor}[\phi'(1+\text{maximum integer value})] & \text{if } \phi' < 1 \end{array}, \tag{4}$$

and then combining these two integers into one by using the bitwise exclusive-or operation

$$w = u \oplus v. \tag{5}$$

On the other hand, if our activities aren't quite as critically important as something like encryption, then we could instead settle for pseudo-randomly (ie. algorithmically, deterministically) generated numbers. This sentiment is indirectly echoed by the legendary mathematician John von Neumann: "*Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.*" This is not meant to imply that pseudo-random number generators are necessarily "naughty", but instead that one must be very careful to never equate the quality of their inherently imperfect output with the quality of the perfect output given by a truly random number generator. The reason why the output of a pseudo-random number generator is inherently imperfect is that this output will always exhibit some kind of underlying long-term pattern that potentially exposes it and its deterministic generator to analysis, which may very well allow an attacker to be able to completely predict all future output of the generator. By definition, randomly generated numbers and their nondeterministic generator do not suffer from

[1] For any (unsigned) b-bit integer, the maximum integer value is $(2^b - 1)$.

this type of weakness, which is why it is highly recommended that randomly generated numbers should be the only kind used in critically important activities such as encryption.[2]

Most pseudo-random number generators are based entirely on integer arithmetic, and are generally executed quite rapidly on most digital computers [3]. Within the past couple of decades however, quite a few pseudo-random number generators have instead been based on iterative chaotic dynamical maps [4], which are in turn quite often – although not always – entirely based on real number arithmetic. The standard reasoning for using iterative chaotic dynamical maps is that although real number arithmetic is generally not executed as rapidly as integer arithmetic is, even the simplest of these maps naturally tend to produce pseudo-random output.

An example of a single body (ie. $n = 1$) iterative chaotic dynamical map is the discrete-time version of the logistic function

$$x' = rx(1-x), \tag{6}$$

where $x = [0,1]$ represents the single-component position variable of the system's sole body, and $r = [0,4]$ represents a constant growth rate. It is important to note that although the position variable x is indeed dynamical, insomuch that it changes from one iteration to the next, the background field that drives this change is entirely static. For instance, where $x = 0.51$ and $r = 3.58$, we will find that $x' = 0.894642$ always. In other words, the static background field inherent to this map guarantees that the combination of position $x = 0.51$ and growth rate $r = 3.58$ will always be associated with a set of *one and only one* future-facing path that leads away from x. This is potentially problematic because for some r this one future-facing path from x to x' always eventually leads back to x after only a small number of iterations, in which case the one path forms one cycle that is to be perpetually repeated, ultimately forming a pattern. We generally want to avoid repeated cycles – patterns – when designing a pseudo-random number generator, and as such, the remainder of this paper will focus on the implementation of an iterative chaotic dynamical map that instead relies on a dynamic background field.

3 Using deterministic three-body dynamics on the unit two-sphere in order to obtain a per-position future-directed path set of size larger than one

In order to obtain a dynamic background field, we will use a system of $n = 3$ interacting (ie. coupled) bodies that are constrained to move along the unit two-sphere embedded in flat 3D space. To keep the computation relatively fast, we will use a simple attraction-repulsion-deflection equation (ie. a highly-modified version of Newtonian gravitation) that relies primarily on the lengths of the shortest curved geodesics along the unit two-sphere (ie. the lengths of the minor great circle arcs) that span from body to body. Like the logistic map given in Eq. (6), the system will be of the discrete-time variety.

To initialize the system, we will first assign distinct (pseudo-)randomly generated positions $\hat{X}_i = (x_i, y_i, z_i)$, $i \in \{0, 1, 2\}$ on the unit two-sphere to the three bodies. It seems best to ensure that the bodies are not equidistant in any way after initialization. It also seems best to ensure that both of the angular components of the spherical coordinates are each non-zero for at least one body after initialization, so as to adequately cover the entire space represented by the unit two-sphere. Next, we will assign distinct (pseudo-)randomly generated values from the interval $[2,4]$ to four variables α, β, γ, δ that we will use as exponents in the attraction-repulsion-deflection equation.

The length of the shortest curved geodesic along the unit two-sphere that spans from the ith to the jth body, where $j \in \{0, 1, 2\}$ and $i \neq j$, is

$$d_{ij} = \text{acos}(\hat{X}_i \cdot \hat{X}_j). \tag{7}$$

The unit tangent vector $\hat{N}_{ij}$ that points along the shortest curved geodesic that spans from the ith to jth body will be used to calculate the attraction and repulsion, and the unit tangent vector $\hat{O}_{ij}$ that is orthogonal to both $\hat{X}_i$ and $\hat{N}_{ij}$ will be used to calculate the deflection

$$\hat{O}_{ij} = \text{normalize}(\hat{X}_i \times \hat{X}_j), \tag{8}$$

[2]Not all pseudo-random number generators are created equal, insomuch that some are better than others at interrupting the formation of patterns in their output. The patterns in the output of some pseudo-random number generators are so very subtle that these generators are often classified as "cryptographically strong", and are often considered to be suitable for use in encryption activities. If one is debating whether or not to use such a "cryptographically strong" pseudo-random number generator, then one should first ask themselves the question "*Are all of the data extremely time-sensitive?*" Say for instance that all of the data contain messages like "Meet me at location X in 5 minutes". Clearly this is an extremely time-sensitive message, and it simply may not matter if an attacker manages to decrypt them a year from now by exploiting some pattern in the output of the pseudo-random number generator. User discretion is advised however, and this footnote should not be considered as a general endorsement of these so-called "cryptographically strong" pseudo-random number generators. Ultimately, if one's data are *not* extremely time-sensitive, then one should always seriously consider using a hardware-based random number generator that relies entirely on quantum mechanical (ie. nondeterministic) processes. All said, paranoid caution is not necessarily a bad thing when it comes to matters of secure privacy.

$$\hat{N}_{ij} = \hat{O}_{ij} \times \hat{X}_i. \tag{9}$$

The ith body's combined tangent vector – which by design is generally longer than the circumference of the unit two-sphere – is

$$\vec{A}_i = \sum_{j=0, j\neq i}^{n-1} \left[\hat{N}_{ij} ({d_{ij}}^{\alpha} - {d_{ij}}^{-\beta}) + \hat{O}_{ij} ({d_{ij}}^{\gamma} - {d_{ij}}^{-\delta}) \right]. \tag{10}$$

The reason for having the deflection terms is that they generally halt the otherwise frequent emergence of short-lived and unstable coplanar orbits.

Once all $\vec{A}_i$ have been calculated, we will take the $\vec{A}_i$ that has the longest length[3]

$$\ell_i = \text{length}(\vec{A}_i), \tag{11}$$

which we will denote as $\vec{A}_\zeta$, and then use the corresponding body's position $\hat{X}_\zeta$ to generate the angular components of the spherical coordinates[4] [5]

$$\theta_\zeta = \pi + \text{atan}_2(-z_\zeta, -x_\zeta), \tag{12}$$

$$\phi_\zeta = \text{acos}(y_\zeta). \tag{13}$$

These two angles will then be normalized, converted to integers, and finally combined into one using the bitwise exclusive-or operation – as was done in Eqs. (1 - 5) – *in order to obtain the pseudo-randomly generated output integer* w_ζ. Here we use just one of the bodies to generate w_ζ – rather than using all three of them – in order to avoid revealing the entire current state of the system in the highly likely event that the exclusive-or operation becomes undone by analysis.

Once the pseudo-randomly generated output integer has been calculated, we will move each body along the curved geodesic that is associated with its combined tangent vector $\vec{A}_i$. Since the length of $\vec{A}_i$ is generally greater than the circumference of the unit two-sphere, it will generally be found that the curved geodesic winds around the unit two-sphere one or more times before coming to an end. This gives an effective curved geodesic length of

$$\ell'_i = \ell_i \bmod 2\pi, \tag{14}$$

which beneficially keeps the true length of $\vec{A}_i$ – and thus the very equation that was used to generated $\vec{A}_i$ – from ever becoming directly encoded in the evolving state of the system. We will then normalize $\vec{A}_i$ and use it in conjunction with the effective length to perform the desired movement along the unit two-sphere via the parametric circle equation

$$\hat{A}_i = \text{normalize}(\vec{A}_i), \tag{15}$$

$$\hat{X}'_i = \hat{X}_i \cos(\ell'_i) + \hat{A}_i \sin(\ell'_i). \tag{16}$$

At this stage the pseudo-random number generator is ready for the next iteration, which can be achieved by performing the calculations associated with Eqs. (7 - 16) in conjunction with the new body positions $\hat{X}'_i$.

It is important to note that there is generally more than one possible value for $\vec{A}_i$ for any given $\hat{X}_i$, and thus there is generally more than one future-directed path that starts at $\hat{X}_i$. This is because $\vec{A}_i$ does not rely on just one dynamical body position – like it would in the case of the logistic map – but on all three dynamical body positions. Ultimately, by generally increasing the number of future-directed paths (and thus potential cycles) per position to some value greater than one, we have generally eliminated the absolute need for a body to follow one cycle repeatedly, thus generally interrupting the formation of the related pattern.

Caution should be taken to avoid the following pitfalls:

1. The first pitfall relates to the singularities in the spherical coordinates at the North and South poles of the unit two-sphere, where $\phi = 0$ or $\phi = \pi$ (ie. where $y = 1$ or $y = -1$). In this rare pitfall, the solution to Eq. (12) cannot be determined because $x_\zeta = 0$. To avoid this rare pitfall, one can instead use the x value from the body's previous position, since the body must have moved along a geodesic of constant θ in order to arrive precisely at a pole.

[3] ... or the shortest length, or whatnot, if one so desires ...

[4] The author's preference of associating y with ϕ is an artifact of the author's experience using OpenGL, and is in no way set in stone.

[5] The $\text{atan}_2(a, b)$ function is included with most modern computer programming languages. This function returns the principal value of the function $\arg(a + ib)$.

2. The second pitfall relates to the possibility that motion can lead to bodies that share a position. To avoid this rare pitfall, one can eliminate any duplicates by providing the problematic body with a new and distinct (pseudo-)randomly generated position.

3. The third pitfall relates to extremely short combined tangent vectors of length$(\vec{A}_i) \sim 0$. When $\vec{A}_i$ has a length of zero, it cannot be normalized, and it cannot be used to move the corresponding body in any meaningful way. To avoid this rare pitfall, one can provide the problematic body with a new (pseudo-)randomly generated combined tangent vector $\vec{A}_i$ of length greater than zero.

4. The fourth pitfall relates to extremely long combined tangent vectors $\vec{A}_i$ as encoded using floating point variables on a digital computer. As the length of the vector increases to ultra-macroscopic proportions, there is an accompanying decrease in the precision of the microscopic details encoded by the floating point variables to the right of their decimal places. This increasingly betrays the discrete nature of the floating point variables at the macroscopic scale, which is taken to be very undesirable because it is tantamount to truncating away a large number of future-directed paths per position. As such, if one wishes to alter the system described here in any way, then one must take very good care to ensure that the system doesn't become too self-truncating. To avoid this potentially common pitfall, one can either select an attraction-repulsion-deflection equation that does not often produce extremely long vectors – but still generally longer than the circumference of the unit two-sphere – or increase the precision of the floating point variables (ie. from 64-bit to 128-bit or greater), or decrease the size of the output integers (ie. from 32-bit to 16-bit, or even all the way down to 1-bit).

For the remainder of this paper we will refer to this pseudo-random number generator as the "m-sphere mixer".

For further information on chaotic pseudo-random number generators that provide more than one future-directed path per position see [5, 6].

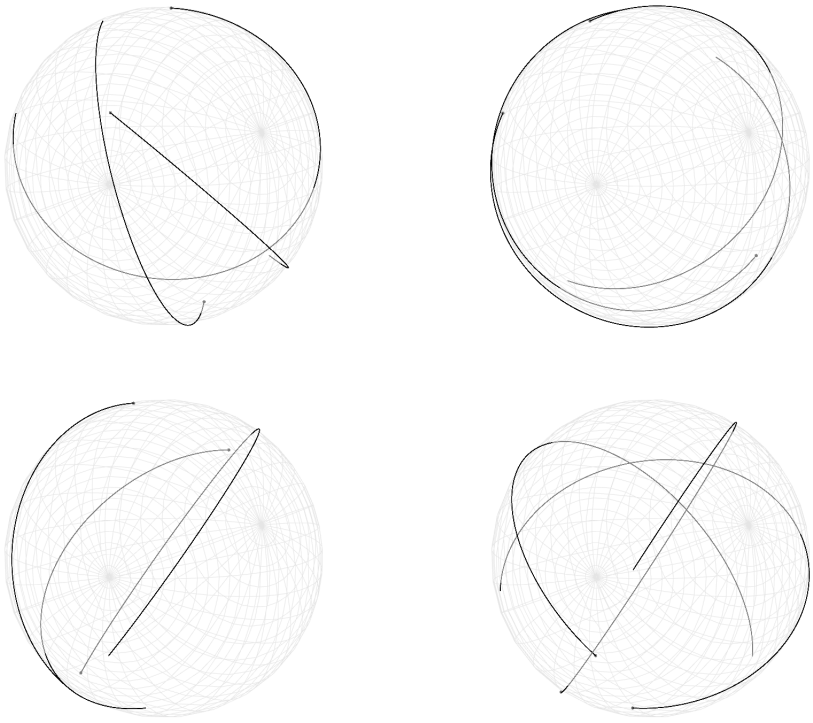

Figure 2: The four figures – in order from top-left to top-right to bottom-left to bottom-right – show three bodies moving along the unit two-sphere during four consecutive iterations.

4 On augmenting the m-sphere mixer in order to further interrupt the formation of patterns

One way to increase the number of future-directed paths per position would be to convert the m-sphere mixer to use a unit sphere of higher dimension $m > 2$. For instance, there exists both a well-defined binary (ie. two argument) cross product operation and a set of spherical coordinates in flat 7D space, and so the conversion of the m-sphere mixer to use an embedded unit six-sphere would provide a much larger set of future-directed paths per position with only a bare minimum of difficulty.

Another way to increase the number of future-directed paths per position would be to increase the number of bodies from three to some other odd number larger than three. It is important to note that the complexity of the dynamics is related to the number of bodies in a nonlinear way, since the number of pairs of bodies is $(n^2 - n)/2$. As such, in the case where we decide to increase the number of bodies to some odd number far greater than three, we may wish to consider implementing the m-sphere mixer on a dedicated graphics processing unit, or on a specialized central processing unit such as those of the Intel Sandy Bridge or AMD Fusion variety. This is because these particular types of processing units can be extremely adept at rapidly solving parallelizable problems (ie. problems with many independent terms), and are extremely adept at rapidly performing the individual geometry-related operations (ie. $\cdot$, $\times$, $+$, $-$, cos, sin, atan_2, *etc*).

In any case where we decide to increase the number of bodies or spatial dimensions, we must always take care to ensure that the initial body positions sufficiently cover the space, but that the body positions cannot readily evolve to cover the space in any significantly homogeneous way – lest the body positions evolve to form any kind of prolonged equilibrium. As mentioned before, when initializing the system it seems best to ensure that the bodies are not equidistant in any way. Also, when initializing the system it seems best to ensure that all of the m angular components of the spherical coordinates are each non-zero for at least one body. In addition to this, when initializing the system it seems best to ensure that there are not enough bodies to cover the antipodal points that correspond to each and every one of the $(m+1)$ axes of the flat $(m+1)$D space that the m-sphere is embedded within. As such, it seems sufficiently cautious to never use more than $(2m-1)$ bodies for any given m (ie. three bodies for the two-sphere, five bodies for the three-sphere, etc). A rough count of the positions on a unit m-sphere, as implemented using b-bit floating point variables on a digital computer, is

$$\text{positions} = 2^{bm}. \tag{17}$$

An accompanying rough count of the future-directed paths per position – including the "non-paths" that immediately lead back to the same position – is

$$\text{paths} = 2^{bm(n-1)}. \tag{18}$$

To compare, the count of the future-directed paths per position for the logistic map is always 1 since $n = 1$. Altogether, a rough count of the map's keys for any one set of constants α, β, *etc*, is

$$\text{keys} = \text{positions} \times \text{paths} = 2^{bmn}. \tag{19}$$

For instance, where $n = 3$, $m = 2$, and $b = 64$ (ie. double-precision floating point variables), the map's key count is roughly 2^{384}.

One other way to further interrupt the formation of patterns seems to be by eliminating the symmetries that are inherent to the equations of dynamics. For instance, we may wish to somehow make the equations of dynamics radially asymmetric, or non-monotonic. Furthermore, we may even wish to (pseudo-)randomly assigning distinct equations – or at least parts of the equations, such as the values of some exponents – to each of the bodies *at each iteration*. Perhaps Noether's theorem relating to the symmetries of physical systems can somehow be used as an inspirational guide when searching for ways to further interrupt the formation of patterns in dynamics.

Given all of this, it seems that if the m-sphere mixer is eventually found to be an unsatisfactory pseudo-random number generator, then it is most likely only because the map – the real number arithmetic – as described in this paper is not complicated enough. That said, any extraneous bitwise operations of any kind should be considered undesirable.[6]

[6]The m-sphere mixer as described in this paper "relies" on a "necessary" series of $(m-1)$ bitwise exclusive-or operations per iteration, in order to merge the m angular component integers into one output integer. However, even this series of bitwise exclusive-or operations is not absolutely necessary, and it could be skipped altogether by simply using only 1 of the m angular components to generate the iteration's output integer. The point to take away from this is that the quality of the pseudo-randomness of the angular components over time has – and should always have – absolutely nothing to do with bitwise operations of any kind.

5 Conclusion

In this paper we have identified several different types of patterns in dynamics that we would ideally like to avoid, so that we may create a satisfactory chaotic pseudo-random number generator.

Of first importance was the interruption of the formation of patterns related to maps that have only one future-directed path per position. This interruption was achieved through the use of many-body dynamics.

Of second importance was the interruption of the formation of patterns related to short-lived and unstable coplanar orbits. This interruption was achieved through the use of deflection terms in the equations of dynamics.

Of third importance was the observation that there are many additional symmetries that are inherent to the equations of dynamics (ie. radial symmetry, temporal symmetry), and that we may wish to augment the m-sphere mixer in order to interrupt the formation of the related patterns.

Of fourth importance was the observation that the solutions $\hat{X}_i'$ of the equations of dynamics – which are arrived at via a mod operation – do not directly encode the equations of dynamics. This makes it non-trivial to perfectly re-derive the previous iteration's body positions from the current iteration's body positions. In addition to this, only one of the current iteration's body positions is encoded into the pseudo-randomly generated output integer, which in and of itself makes it non-trivial to perfectly re-derive all n – where n is unknown to the attacker – of the current iteration's body positions from the pseudo-randomly generated output integer. Also, note that if we were to use double-precision positions and exponents to initialize both a double-precision version and a single-precision version of the m-sphere mixer, that we would find that the output of these two different versions would start to widely diverge after only a few iterations. This is because the single-precision version of the m-sphere mixer must first convert its copy of the double-precision position and exponent values into single-precision values, which generally truncates the values – thus slightly altering them. Naturally, this slight difference in the initial conditions feeds upon itself over time – thus forming a compounding error in the solutions of the equations of dynamics – which inevitably makes the difference large.[7] This means that any slightly imperfect guess at the current and/or previous iteration's body positions will inevitably grow in error over time, and that any slightly imperfect guess will inevitably be rendered practically useless. All said, perfectly reverse-engineering the full behaviour of the m-sphere mixer over time from patterns in its output would be a non-trivial endeavour.

Of final importance is the previously unmentioned observation that the m-sphere mixer operates in a closed curved (ie. spherical) space, whereas some other pseudo-random number generators operate in a closed flat (ie. toroidal) space. Since parallel lines always remain parallel in a toroidal space, but not so in a spherical space, it seems that a spherical space bears the natural ability to interrupt the formation of patterns related to parallel motion, whereas a toroidal space does not. Also, unlike some other pseudo-random number generators, the m-sphere mixer does not rely at all on momentum. As such, it seems that the m-sphere mixer bears the natural ability to interrupt the formation of patterns related to inertial motion, whereas some other pseudo-random number generators do not.

6 Afterword

A template-based C++ partial implementation of the m-sphere mixer can be downloaded from the author's corresponding Google Code project [7]. The implementation includes functionality to easily write large amounts of pseudo-randomly generated numbers to a binary file on disk, so that the effectiveness of the implementation can be independently analyzed by using at least the NIST tests. Regardless of the results of independent testing for any implementation of the m-sphere mixer, the author will never recommend using it for critically important activities such as encryption.

The author wishes to thank JH, AJ, and SB for helpful discussion and support during the development of the m-sphere mixer.

References

[1] Binney J, Tremaine S. Galactic dynamics, 2E. (2008) ISBN: 9780691130279

[2] Ruijl B, van Loon E, Hopman E. Random number generation using a mechanical deterministic-chaotic process. (2010) Preprint.

[7] Recall the two different three-body systems that were mentioned in the first section – perfectly equidistant *versus* not-quite-perfectly equidistant. Recall how this slight difference in the initial conditions fed upon itself over time in order to become a large difference.

[3] Matsumoto M, Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. (1998) ACM Transactions on Modeling and Computer Simulation (TOMACS) - Special issue on uniform random number generation - TOMACS Homepage archive, Volume 8, Issue 1.

[4] Álvarez G, Li S. Some basic cryptographic requirements for chaos-based cryptosystems. (2006) International Journal of Bifurcation and Chaos, Volume 16.

[5] Orúe AB, Álvarez G, Guerra A, Pastor G, Romera M, Montoya F. Trident, a new pseudo random number generator based on coupled chaotic maps. (2010) arXiv:1008.2345v2 [cs.CR]

[6] Orúe AB, Montoya F, Hernández Encinas L. Trifork, a new pseudorandom number generator based on lagged Fibonacci maps. (2010) Journal of Computer Science and Engineering, Volume 2, Issue 2.

[7] Halayka S. Chaotic pseudo-random number generator, v1.3. (2012) `http://chaotic-prng.googlecode.com/files/prng_v1.3.zip`

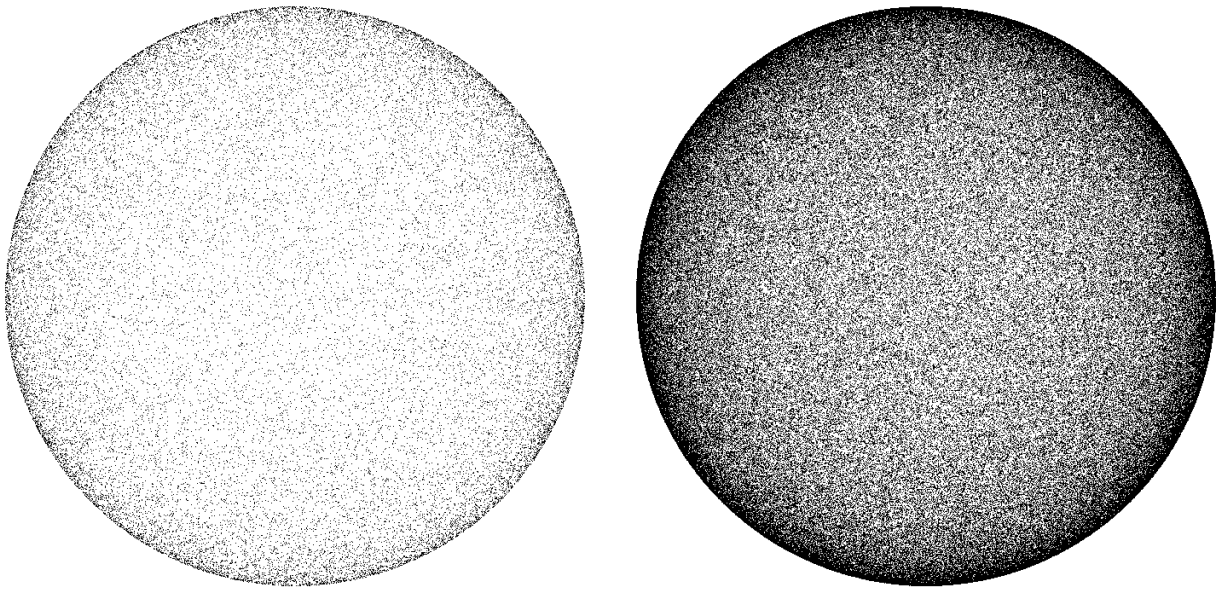

Figure 3: The figures show the positions on the unit two-sphere that were selected to subsequently produce the output of the pseudo-random number generator. The figure on the left shows the positions selected after 10^5 iterations, and the figure on the right shows the positions selected after 10^6 iterations. Please keep in mind that the bunching together of the positions toward the "edge" of the two-sphere is primarily an artifact of the perspective projection that was used to generate the figures.