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 *

March 12, 2012

Abstract

The goal of this paper is to describe how to implement a pseudo-random number generator by using deterministic n-body dynamics on the unit m-sphere. Throughout this paper we identify several types of patterns in dynamics, along with ways to interrupt the formation of these patterns.

1 Nondeterministic and deterministic/chaotic-deterministic physical behaviour

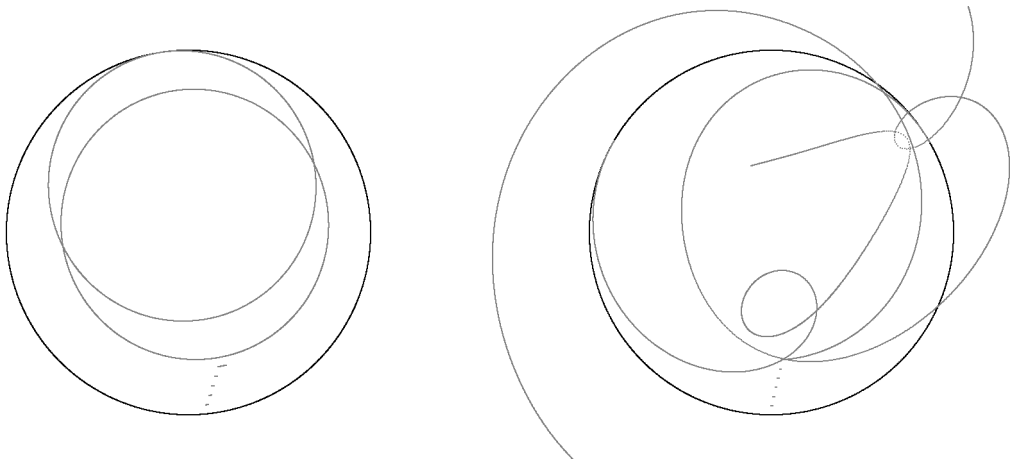
In terms of the laws of physics – as far as how we model them today, anyway – there are two types of ways that a system can evolve over time: nondeterministically and deterministically.

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 simply form new, elliptical orbit paths, and continue to undergo 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. Ultimately, this dynamic asymmetry in the strength of the gravitational interaction amongst the three bodies generally interrupts any chance that they will ever again re-establish anything remotely similar to their original prolonged equidistant circular orbit relationship. This devolution from *predictable, repetitious cyclical motion* into *pseudo-random, non-repetitious acyclical motion* due to some slight asymmetry 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



(a) A deterministic two-body system over time.

(b) A chaotic-deterministic three-body system over time.

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

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

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

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

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

On the other hand, we could instead consider settling for a series of pseudo-randomly (ie. algorithmically, deterministically) generated numbers. However, before making any kind of final decision on whether to use randomly or pseudo-randomly generated numbers, it seems best to always consider the following words of caution 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 "evil", but instead that we must always 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 exposes it and its deterministic generator

to analysis. If this analysis is eventually successful, it could allow an attacker to completely predict all future output of the generator, which would ultimately render the generator entirely useless for its intended purpose. By definition, randomly generated numbers and their nondeterministic generator do not suffer from this type of weakness. All said, pseudo-random number generators are never perfect, so we must be extremely cautious when deciding on whether or not to use them in activities that ideally require absolute perfection.[1]

Traditionally, the vast majority of pseudo-random number generators have been entirely based on integer operations [3,4]. Within the past couple of decades however, quite a few pseudo-random number generators have instead been based on chaotic dynamical maps, which are in turn quite often – although not always – entirely based on real number operations [5–11]. In spite of the fact that real number operations are often executed less quickly than their integer counterparts are on a digital computer, chaotic dynamical maps are still considered to be interesting because even the simplest of them can produce pseudo-random output under certain conditions.

An example of a single body (ie. $n = 1$) 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, $r = [0,4]$ represents a constant growth rate, and $x' = [0,1]$ represents the body's new position after one iteration of the map.

It is important to note that although the position variable x is indeed dynamic, insomuch that it may change from one iteration of the map to the next, the background field that drives this change is effectively static. For instance, where $x = 0.1$ and $r = 2.2$, we will find that $x' = 0.198$ always. This means that any given combination of x and r will always be associated with *one and only one* future-directed path that leads away from x. Unfortunately, for most combinations of x and r this one future-facing path that leads away from x always quickly ends in a short cycle that is to be perpetually repeated, thus forming an unavoidable pattern. For instance, using $x = 0.1$ and $r = 2.2$ as before, if we were to continue iterating the map by repeatedly using x' as the new value for x, then we would find that the path quickly ends in an extremely short cycle that consists of just one position:

$$0.1 \to 0.198 \to 0.349351 \to 0.500071 \to 0.55 \to 0.5445 \to 0.545643 \to 0.545417 \to 0.545462 \to 0.545453 \to 0.545455 \to 0.545454 \to 0.545455 \to 0.545455 \to 0.545455\ldots$$

This repeated cycle is definitely not something that we would try to leverage in order to implement a pseudo-random number generator. Similarly, using $x = 0.7$ and $r = 2.2$ quickly produces the same extremely short cycle:

$$0.7 \to 0.462 \to 0.546823 \to 0.545177 \to 0.54551 \to 0.545443 \to 0.545457 \to 0.545454 \to 0.545455 \to 0.545455 \to 0.545455\ldots$$

Using $x = 0.5$ and $r = 3.2$ quickly produces a short cycle of just two positions:

$$0.5 \to 0.8 \to 0.512 \to 0.799539 \to 0.512884 \to 0.799469 \to 0.513019 \to 0.799458 \to 0.51304 \to 0.799456 \to 0.513044 \to 0.799456 \to 0.513044 \to 0.799455 \to 0.513044 \to 0.799455 \to 0.513045 \to 0.799455 \to 0.513045 \to 0.799455 \to 0.513045\ldots$$

Using $x = 0.5$ and $r = 3.5$ quickly produces a short cycle of just four positions:

$$0.5 \to 0.875 \to 0.382813 \to 0.826935 \to 0.500898 \to 0.874997 \to 0.38282 \to 0.826941 \to 0.500884 \to 0.874997 \to 0.38282 \to 0.826941 \to 0.500884 \to 0.874997 \to 0.38282 \to 0.826941 \to 0.500884\ldots$$

This is not to say that the one future-facing path that leads away from x always ends in a short cycle for all combinations of x and r. For instance, using $x = 0.3$ and $r = 4$:

$$0.3 \to 0.84 \to 0.5376 \to 0.994345 \to 0.0224922 \to 0.0879454 \to 0.320844 \to 0.871612 \to 0.447617 \to 0.989024 \to 0.0434219\ldots$$

[1] 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 considered by some to be suitable for use in encryption activities. When deciding on whether or not to use a cryptographically strong pseudo-random number generator, we must first ask the question "*Are all of the data time-sensitive*"? Say for instance that all of the data contain messages like "*Meet me at location X in 5 minutes*". Clearly these are time-sensitive messages, and it simply may not matter if an attacker manages to decrypt them in the far future by exploiting some extremely subtle pattern in the output of the cryptographically strong pseudo-random number generator. Of course, if the data are *not* time-sensitive, then we 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.

Although there is some similarity between the early subpath $0.84 \to 0.5376 \to 0.994345 \to 0.0224922$ and the late subpath $0.871612 \to 0.447617 \to 0.989024 \to 0.0434219$ – insomuch that they form sequences of values of similar proportion – these two subpaths are not precisely identical and so no cycle is formed. However, do keep in mind that using $x = 0.5$ and $r = 4$ produces a short cycle of just one position:

$$0.5 \to 1 \to 0 \to 0 \to 0 \ldots$$

Clearly the logistic map is not very robust, insomuch that it does not produce a long cycle for most combinations of x and r.

In order to find a more robust chaotic dynamical map, consider the possibility that the general result of the logistic map somehow implies that the movement of the body is driven by interaction with one or more extra *hidden* bodies that may also move around. Because the background field is effectively static, this means that we would find that the interaction amongst the visible and hidden bodies is always the same whenever the visible body is at some specific x. This means that the movement of all of the bodies is somehow synchronized, like in the case of determinstic two-body dynamics that was mentioned in the previous section. Perhaps we could somehow avoid the problems associated with the logistic map by simply replacing it with a chaotic dynamical map that allows the movement of the bodies to fall out of synchronization, like in the case of chaotic-determinstic three-body dynamics that was mentioned in the previous section.

All said, we generally want to be able to avoid repeated cycles – patterns – as much as possible when designing a pseudo-random number generator, and as such, the remainder of this paper will focus on the use of a chaotic dynamical map that 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 partially constrained to move along the unit two-sphere embedded in flat 3D space. One benefit that comes from partially constraining the movement of the bodies as such is that none of the bodies will ever get kicked out of the system by strong interaction.

To keep the computation relatively fast, we will use simple attraction-repulsion-deflection equations that rely 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 of the positions 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 constants α, β, γ, δ that we will use as exponents in the attraction-repulsion-deflection equations.

The length of the shortest curved geodesic along the unit two-sphere that spans from the ith body 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 body 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

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

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

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 help to disrupt the otherwise very frequent emergence of short-lived and unstable nearly-coplanar orbits.

Once all $\vec{A}_i$ have been calculated, we will take the $\vec{A}_i$ that has the longest length[2]

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

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[3]

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

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

These two angles will then be normalized, converted into integers, and finally combined into one integer using a 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 positions to generate w_ζ in order to avoid revealing all of the positions 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 generally keeps the true length of $\vec{A}_i$ – and thus the true equations of dynamics that are used to generate $\vec{A}_i$ – from ever becoming directly encoded in the movement of the bodies. 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 equations

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

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

At this stage the pseudo-random number generator is ready for the next iteration, which can be achieved by once again performing the calculations associated with Eqs. (7 - 16) in conjunction with the new 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 combination of $\hat{X}_i$ and the constants α, β, *etc*, and thus there is now generally more than one future-directed path that leads away from $\hat{X}_i$. This is because the value of $\vec{A}_i$ does not rely on just one dynamic position and some constants – like it would in the case of the logistic map – but on all three dynamic positions and some constants. Fortunately, there is an instability inherent to three-body dynamics, which means that per-position future-directed paths quickly ending in a short cycle are now the exception rather than the norm. Also, it is important to note that the path followed by *the entire system as one* depends on all of the map's parameters – the system's state – and as such there is only one future-directed path for any given combination of $\hat{X}_i$, *the other two dynamic positions*, and the constants α, β, *etc*. Fortunately, thanks again to the instability inherent to three-body dynamics, for most system states this one *per-state* future-directed path that leads from state to state does not quickly end in a short cycle.[4]

For more information on chaotic pseudo-random number generators that provide more than one future-directed path per position, see [10, 11].

4 Basic error correction

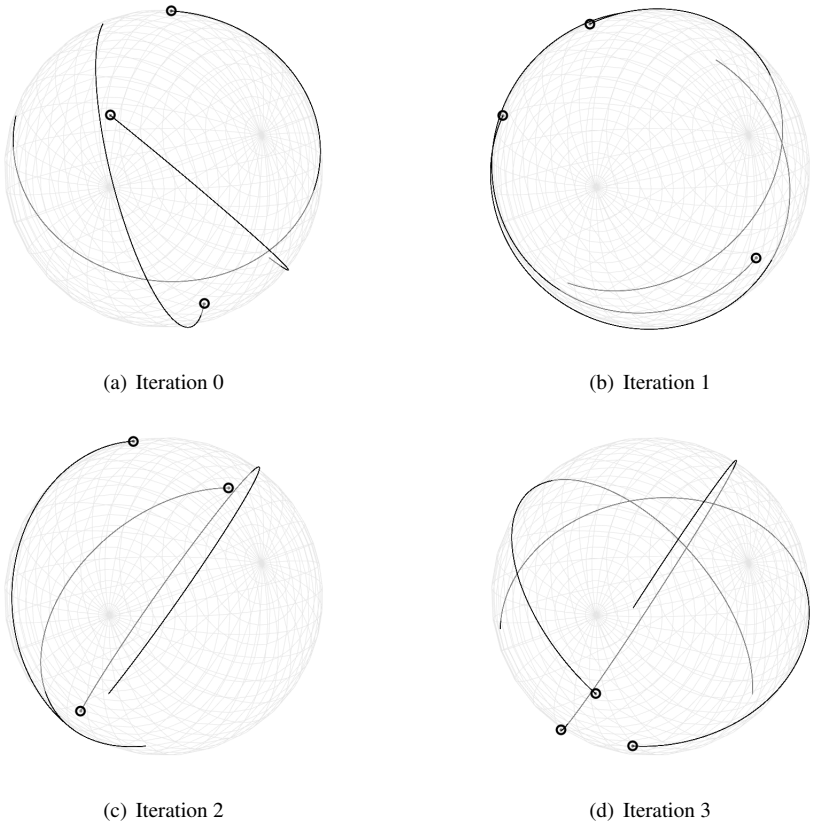
Caution should be taken to avoid the following pitfalls:

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 $x = z = 0$. To avoid this rare pitfall, we can instead use the x, z values 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.

[2] ... or the shortest length, or whatnot, if we so desire ...

[3] The $\text{atan}_2(a, b)$ function is included with most modern computer programming languages. This function returns the principal value of the function $\arg(a + bk)$, where k is the imaginary number $\sqrt{-1}$.

[4] The logistic map has one future-directed path per position and one future-directed path per system state. Since there is only one position variable in the system state however, these two future-directed path types are actually one and the same. These two future-directed path types only become distinct when the system state starts to include additional variables.



(a) Iteration 0

(b) Iteration 1

(c) Iteration 2

(d) Iteration 3

Figure 2: Four consecutive iterations of the chaotic dynamical map. For each iteration, the three bodies are moved along curved geodesics (black arcs) from $\hat{X}_i$ (highlighted using small black circles) to $\hat{X}_i'$ (not highlighted).

2. The second pitfall relates to the possibility that motion can lead to bodies that share a position. To avoid this rare pitfall, we can eliminate a duplicate by providing the problematic body with a new and distinct (pseudo-)randomly generated position. As was the case when generating the initial positions, it seems best to avoid equidistant bodies, and to ensure that the entire space represented by the unit two-sphere is adequately covered.

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, we can provide the problematic body with a new (pseudo-)randomly generated combined tangent vector of length greater than zero.

4. The fourth pitfall relates to extremely long combined tangent vectors $\vec{A}_i$ as represented by floating point variables on a digital computer. As the length of the combined tangent 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 we wish to augment the system described here in any way, then we must take very good care to ensure that the system doesn't become too self-truncating. To avoid this potentially common pitfall, we can increase the precision of the floating point variables that represent the combined tangent vector components. For instance, if we use b-bit floating point variables to represent the position components, and a $(b/2)$-bit integer to represent the pseudo-randomly generated output variable, then it seems best to use $(2b)$-bit – or greater – floating point variables to represent the combined tangent vector components.

Note that although the deflection terms in the equations of dynamics do help to disrupt the otherwise very frequent emergence of short-lived and unstable nearly-coplanar orbits, a very few of these orbits do still manage to survive for longer than

one iteration. Given this, and the previously mentioned pitfalls, perhaps we may wish to consider adding in some kind of prediction-correction component to the pseudo-random number generator. This predictor-corrector would proactively detect any upcoming problematic state along the system's current future-directed path, and then try to avoid the problematic state by switching the system from its current future-directed path to a new and different future-directed path before the problematic state is ever encountered. For instance, if the predictor-corrector was to look ahead and find that in 100 iterations that two of the bodies will form some kind of orbit, or come to share the same position, or whatnot, then the predictor-corrector could simply nudge the positions of the problematic bodies – or (pseudo-)randomly reassign them new positions – during the *current* iteration. Hopefully, because small differences in initial conditions tend to become large differences over time, the nudged bodies will no longer become problematic after the 100 iterations have occurred.[5] The fact is that problematic states can still occur, and if we do not have a predictor-corrector component actively seeking out and avoiding these problematic states, then there is a chance that the pseudo-random number generator could catastrophically fail insomuch that it could start to produce entirely repetitive output. Although a predictor-corrector component does entail extra computation that is above and beyond what would be required if we were simply checking for problematic bodies in the current iteration, this extra computation would only be required in the rare situations where the pseudo-random number generator is initialized or where a problematic state is detected and avoided. As such, the cost of this extra computation would be amortized and thus practically inconsequential. The correction process would be tantamount to performing surgery, whereby the current system state's one future-directed path is cut off and then a new future-directed path is created and sewn on between the current system state and the new next system state.

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

5 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 positions sufficiently cover the space, but that the positions cannot readily evolve to cover the space in any significantly homogeneous way – lest the bodies 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 of the positions. In addition to this, when initializing the system it seems best to ensure that there are not enough bodies to cover the $2(m+1)$ antipodal positions that correspond to the 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 the unit m-sphere, in terms of b-bit floating point variables, 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

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

[5] 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.

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 – system states – without considering the constants α, β, *etc*, is

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

For instance, where $n = 3$, $m = 2$, and $b = 64$, the map's key count is roughly 2^{384}. More generally, where c is the number of single-component constants, a rough count of the map's keys is

$$\text{all keys} = 2^{b(mn+c)}. \tag{20}$$

For instance, where $n = 3$, $m = 2$, $b = 64$, and there are four single-component constants α, β, γ, δ, the map's general key count is roughly 2^{640}. Of course, once the constants have been assigned specific values and we start to iterate the map, it is the lesser key count 2^{384} that truly matters.

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 assign distinct equations of dynamics – 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.

6 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 implement a satisfactory chaotic pseudo-random number generator.

Of first importance was the interruption of the formation of patterns related to chaotic dynamical maps that have only one future-directed path per position. This interruption was achieved through the use of three-body dynamics on the unit two-sphere.

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 the movement of the bodies from $\hat{X}_i$ to $\hat{X}'_i$ – which is generally arrived at via the mod operation – does not directly encode the true equations of dynamics. This makes it less than easy for an attacker to reverse-engineer the true equations of dynamics by analyzing the movement of the bodies from $\hat{X}_i$ to $\hat{X}'_i$. Consequently, this makes it less than easy to consistently predict the next iteration's positions from the current iteration's positions, and also less than easy to consistently re-derive (ie. retrodict) the previous iteration's positions from the current iteration's positions. Presumably, it would be even less easier to reverse-engineer the true equations of dynamics if they were not just nonlinear, but also radially asymmetric, non-monotonic, and dynamic. Also, only one of the current iteration's positions is encoded into the pseudo-randomly generated output integer, which makes it less than easy to consistently re-derive all n of the current iteration's positions – let alone re-derive the true value of n – from the pseudo-randomly generated output integer. Finally, 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 copies 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 to the equations of dynamics – which inevitably makes the difference large.[6] This means that any slightly imperfect guess at the positions of the bodies will inevitably grow in error over time and become practically useless. All said, entirely reverse-engineering the m-sphere mixer by analyzing the patterns in its output would be a less than easy endeavour. For more information on reverse-engineering the equations of dynamics from position data, see [12].

Of fourth 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 conserved linear 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] Again, recall from the first section how a slight difference in the initial conditions became a large difference.

7 Afterword

A template-based C++ partial implementation of the *m*-sphere mixer can be downloaded from the author's corresponding Google Code project [13]. This implementation includes functionality to easily write large amounts of pseudo-randomly generated numbers to a binary file on disk, so that the effectiveness of this implementation can be independently analyzed by using at least the NIST tests [14]. 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.

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.

[3] L'Ecuyer P. Tables of linear congruential generators of different sizes and good lattice structure. (1999) Mathematics of Computation, Volume 68, Number 225.

[4] 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.

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

[6] Patidar V, Sud KK. A pseudo random bit generator based on chaotic logistic map and its statistical testing. (2009) Informatica, Volume 33, Number 4.

[7] Patidar V, Sud KK. A novel pseudo random bit generator based on chaotic standard map and its testing. (2009) Electronic Journal of Theoretical Physics, Volume 6, Number 20.

[8] Pareek NK, Patidar V, Sud KK. A random bit generator using chaotic maps. (2010) International Journal of Network Security, Volume 10, Number 1.

[9] Suneel M. Cryptographic pseudo-random sequences from the chaotic Hénon map. (2009) Sadhana, Volume 34, Part 5.

[10] 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]

[11] 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.

[12] Cubitt TS, Eisert J, Wolf MM. Extracting dynamical equations from experimental data is NP hard. (2012) Phys. Rev. Let. Preprint.

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

[14] National Institute of Standards and Technology. Random number generation. (2008) `http://csrc.nist.gov/groups/ST/toolkit/rng/index.html`

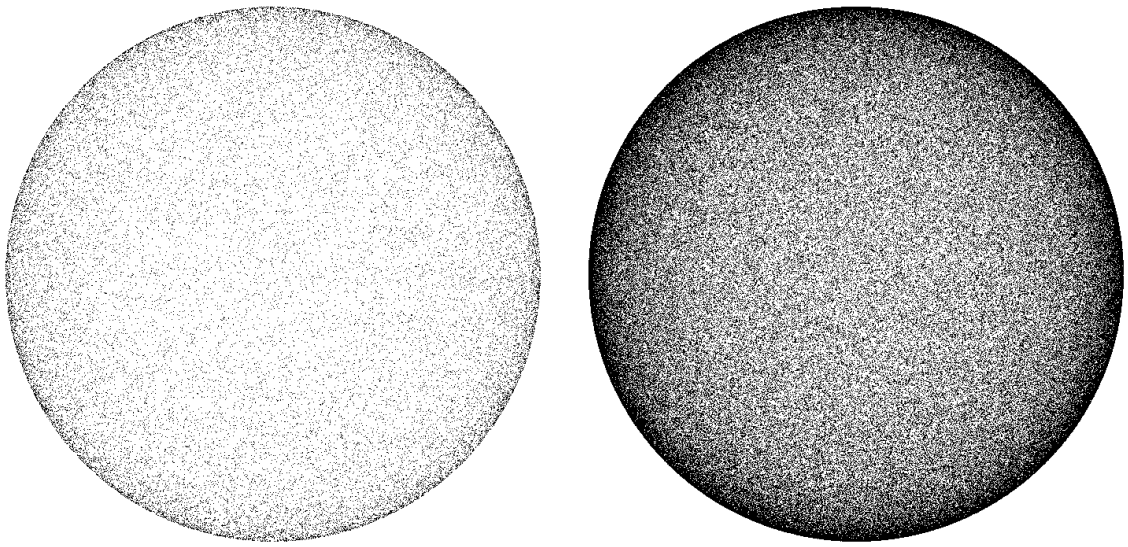


Figure 3: The figures show the positions on the unit two-sphere that were selected to subsequently produce the output of the pseudo-random number generator. The figure on the left shows the positions selected after 10^5 iterations, and the figure on the right shows the positions selected after 10^6 iterations. 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.