# Visualizing uncertain curves and surfaces via Gaussian Oscillators

Charles R. Hogg III

January 2015

**Abstract**

We study animations as tools to visualize Gaussian uncertainty in curves and surfaces. First, we define the *Gaussian oscillator*, a unifying concept which encompasses existing and future approaches. We then develop a new technique, the *delocalized oscillator*, which combines the best traits of previous ones: these animations feature fluid, natural motion, and also better represent the underlying distribution. We propose the first metric to judge animation techniques by the quality of individual animations (rather than the entire population): the Kolmogorov-Smirnov distribution. Finally, we suggest how to develop even better animations in the future by leveraging a key insight: that Gaussian animations are nothing more than time-domain Gaussian processes.

## 1 Introduction

Curves and surfaces abound in the sciences. It is vital to quantify their uncertainty — say, by computing a probability distribution. It is equally vital to visualize that uncertainty. Here, curves and surfaces present considerable challenges.

The most common technique is to plot upper and lower credible intervals along with the estimated curve. This useful technique conveys considerable information at a glance, but there is some information it cannot convey. Since intervals are pointwise, they tell us nothing about correlations between

points. In other words, we know the curve lies roughly within a boundary, but we don't know what it's doing within that boundary.

To convey this missing information, we need a complementary technique which depicts individual members of the distribution. The simplest such approach is to sample several curves from the distribution, and plot them together. Unfortunately, there is a fundamental tension: showing too few curves will not fully represent the distribution, while showing too many quickly clutters the figure.

Animations resolve this tension by introducing a time dimension: we show as many curves as necessary, but only one at a time. If each frame is correlated with its neighbours, the animation becomes *continuous*, and the curve appears as a quasi-physical moving object. The eye is naturally drawn to areas with more motion, which represent higher uncertainty. Continuous animations are thus an intuitive and interpretable way to visualize uncertainty in curves.

Naturally, all these considerations apply to uncertain *surfaces* too, but even more strongly. Credible intervals for surfaces are difficult to visualize because the surfaces obscure one another. Animations avoid this difficulty, because they never show more than one surface at a time.

This paper focuses on curves and surfaces whose underlying distribution is Gaussian. This is not nearly as restrictive as it might seem. It includes Gaussian processes[5], a flexible nonparametric class of models. Furthermore, parametric models often have a single posterior mode, which is Gaussian to leading order. Continuous paths through the Gaussian parameter space correspond to continuous animations of the function those parameters govern.

The literature contains two main approaches so far.

Historically, most animations have been based on *interpolation*[7, 2]. A series of independent draws from the Gaussian are generated and displayed at regular time intervals; these are called the *keyframes*. The frames at intervening times are weighted averages of the nearest two keyframes. The weighting favours the first keyframe at the beginning of the interval, and the second keyframe near the end, so that the curve or surface changes continuously from one keyframe to the next. Figure 1(a) illustrates the idea for a simple 1-dimensional Gaussian using three possible interpolating families: *linear*, *cubic spline*, and *trigonometric*.

Unfortunately, interpolation can easily yield incorrect statistics between the keyframes. In particular, the variance is usually underestimated. Figure 1(b) shows that only the trigonometric approach[2, 7] correctly preserves

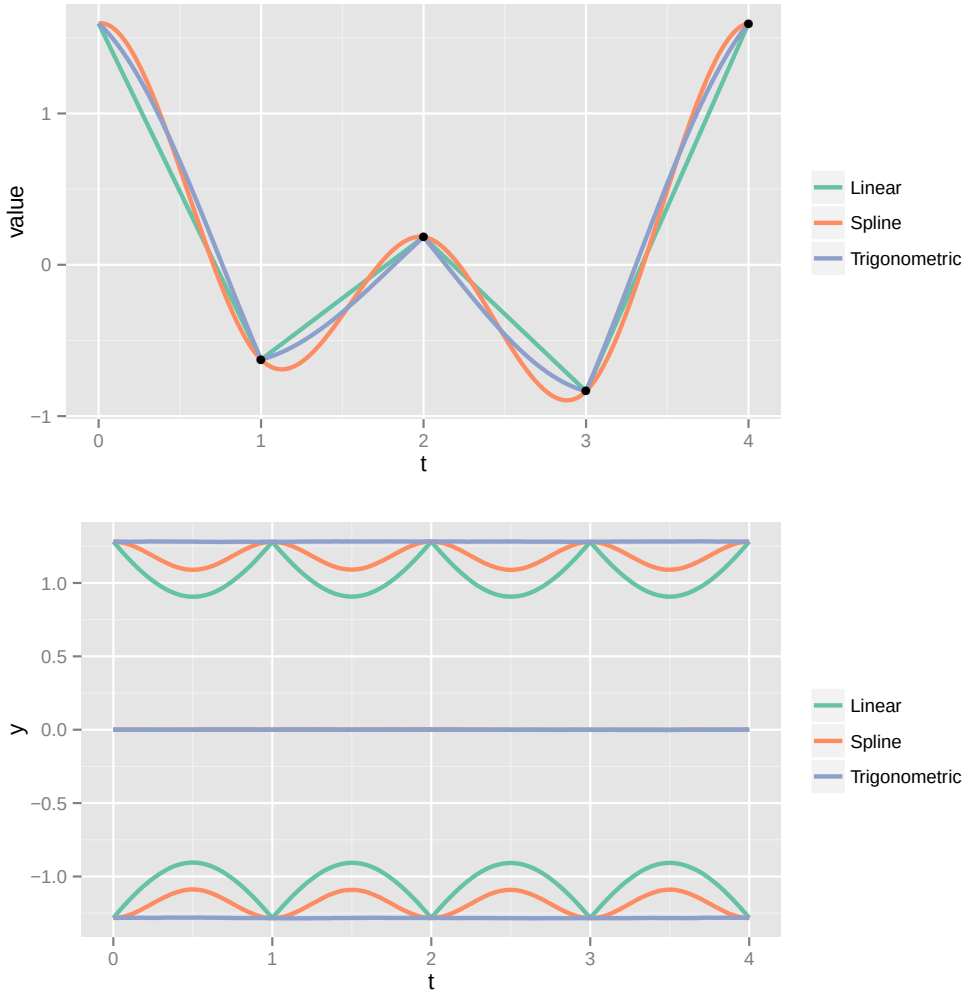


Figure 1: Interpolation illustrated for a standard 1-D Gaussian. (a) A single set of normal draws interpolated by three methods. It is not obvious from this figure which of these methods, if any, is best. (b) Pointwise quantiles for interpolating 1,000,000 sets of random draws. These show that linear and spline interpolation are unacceptable, because the population statistics are different between the keyframes. Only trigonometric interpolation yields the same marginal distribution at all times.

statistical properties between keyframes.

A creative alternative was pioneered by Hennig[3], who generated marginally normal great-circle orbits through parameter space. The starting point is a random draw from the multivariate Gaussian. Motion begins in the direction of a second random draw, and is constrained so that the particle's distance from the origin is constant. Since no point on a circle is special, every frame in these circular timetraces is fully equivalent to every other. To the best of this author's knowledge, Hennig's paper is the first published example of a smooth, keyframe-free Gaussian animation.

Each approach has advantages and weaknesses. Interpolation is flexible: it can incorporate arbitrarily many independent draws, yielding paths which better represent the Normal distribution. The downside is that not all frames are equivalent: keyframes have preferred status, and the motion changes discontinuously there, which is distracting.

Great circle animations[3] do not have this defect; they treat all frames equally in every way. They also represent the normal distribution more faithfully than interpolated animations for the same number of samples, as Section 4.1 will elaborate. Their shortcoming is that they cannot incorporate more than two samples; therefore, they can only visualize a small part of the distribution.

This paper proposes a unifying concept which encompasses these and future approaches: the *Gaussian oscillator*. We then introduce a novel Gaussian oscillator technique, which combines the extensibility of interpolation with the smooth, natural motion of the great circles. We develop quality metrics to assess the strengths and weaknesses of each method. Finally, we suggest how to leverage the Gaussian oscillator framework to construct even better animations in the future.

# 2 Gaussian oscillators

This section will define and explain the *Gaussian oscillator*: the key concept for analyzing Gaussian animations.

First, consider an $m$-dimensional Gaussian, with mean vector $\mu$ and covariance matrix $K$. To draw a sample from this distribution, we need the lower-triangular Cholesky decomposition $L$ of $K$, which satisfies $LL^\intercal = K$. If $\xi \equiv (\xi_1, \cdots, \xi_m)^\intercal$ represents $m$ i.i.d. draws from the *standard* normal $N(0, 1)$, then $\alpha \equiv \mu + L\xi$ is a draw from $N(\mu, K)$.

Now imagine that each $\xi_i$ is a random function of time, $\xi_i(t)$, such that $\xi_i(t) \sim N(0,1)\ \forall t$ (marginally). It follows that $\alpha(t) \sim N(\mu, K)\ \forall t$ (again, marginally). Thus, the problem of animating arbitrary multidimensional Gaussian distributions reduces to that of animating a single-dimensional standard Gaussian. This latter problem is the focus of the rest of the paper.

This motivates us to define a *Gaussian oscillator* as a stochastic process, $\xi(t)$, with the following properties.

$$\xi(t) \sim N(0,1)\ \forall t \tag{1}$$

$$\lim_{\Delta t \to 0} \langle \xi(t)\xi(t+\Delta t) \rangle = 1\ \forall t \tag{2}$$

Equation 1 means that every frame of the animation is marginally Gaussian. Equation 2 means that the animation is *continuous*.

Most $m$-dimensional Gaussian animations can be analyzed by combining $m$ independent Gaussian oscillators. Great circle animations are an exception, since they sample directly from the $m$-dimensional distribution. Each individual dimension of a great circle animation fulfills the definition of a Gaussian oscillator. However, the oscillators are not independent, since the $m$-dimensional vector is constrained to have constant magnitude.

## 2.1 Connection with Gaussian processes

Readers familiar with the concept may have noticed that Gaussian oscillators are also Gaussian *processes*[5], with a few constraints. Specifically, the mean function $\mu(t)$ is zero everywhere, and the covariance function $k(t,t')$ is 1 when $t = t'$ (and continuous in that neighbourhood).

Remarkably, nobody seems to have pointed out this connection before. It is extremely useful, since future Gaussian animation research can leverage the well-studied menagerie of covariance functions[5].

There is some danger of confusion here, because often the curves and surfaces being animated are also modeled using Gaussian processes. The difference is that the Gaussian oscillator is a *time*-domain Gaussian process, while the curve or surface lives in the *space* domain[1].

[1] The "space" here might be an abstract space. It might even be (physical) time — for example, with time-series analysis. However, the point is that this is distinct from the "animation time" domain where the Gaussian oscillator lives.

Treating Gaussian oscillators as Gaussian processes is likely to be more useful for theoretical analysis than for practical computations[2]. Consider a trigonometric interpolation of $k$ keyframes and $n$ total frames; typically, $n \gg k$. Computing $\xi(t)$ as a Gaussian process requires $n$ i.i.d. normal draws, and an $\mathcal{O}(n^3)$ matrix inversion! By contrast, explicit interpolation requires only $k$ normal draws (and simple sines and cosines instead of the matrix inversion).

### 2.2 Offline and online animations

It is useful to distinguish between "offline" and "online" animations. An "offline" animation is one which is fully constructed before it is viewed. The classic example is an animated `.gif`. For "online" animations, later frames are constructed while earlier frames are being viewed. A modern example would be any javascript plotting library which uses `CSS3` transitions.

Note that the online/offline distinction has nothing to do with internet connectivity. For example, an animated `.gif` is the archetype of an "offline" animation, but it is usually shared over the internet. Moreover, "online" animations based on javascript and CSS do not require an internet connection, since the computation happens on the client side.

Offline animations suffer from a fundamental tension. With too few frames, they represent their underlying distribution poorly (as Section 4.1 will elaborate). With too many frames, the file size becomes impractically huge. For this reason, online animations are preferable when available.

## 3 A New Way to Generate Animations

We introduce our new animation technique by comparing it to interpolation. Both techniques involve generating some number $N$ of i.i.d. standard normal variates, then using them deterministically to produce a continuous timetrace.[3] With interpolation, each variate influences only the local region between its nearest neighbours. In this paper's technique, each variate's influence is spread out across the entire timetrace.

[2] Gaussian processes with compact support are a notable exception, as discussed in the appendix.

[3] These random variates can be viewed as coefficients for a set of basis functions, as Figure 2 shows.

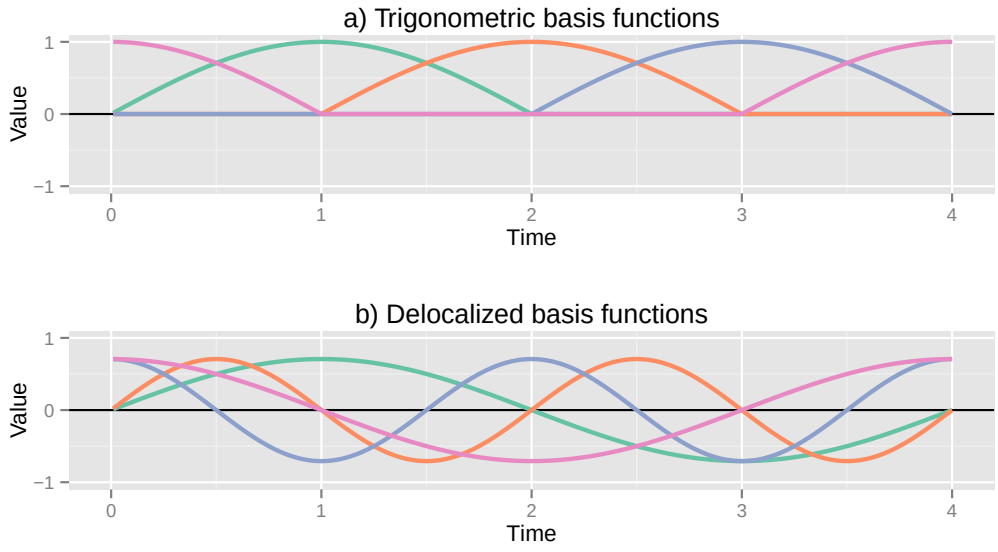


Figure 2: Basis functions for two types of Gaussian oscillators (each based on 4 independent draws, and periodic with period 4). a) Trigonometric interpolation basis functions localize the influence of each sample, but the slope is discontinuous at the boundaries. b) Delocalized oscillator basis functions spread the influence of each sample across the entire timetrace, and are perfectly smooth everywhere.

Explicitly, given $N$ i.i.d. normal variates $\{\epsilon_1, \cdots, \epsilon_n\}$, the corresponding *delocalized oscillator* $\xi(t)$ is given by

$$\xi(t) = \sqrt{\frac{2}{N}} \sum_{i=1}^{N/2} \left[ \epsilon_{2i-1} \sin\left(\frac{2\pi i t}{N}\right) + \epsilon_{2i} \cos\left(\frac{2\pi i t}{N}\right) \right]. \tag{3}$$

$\xi(t)$ is an offline technique. It produces periodic timetraces, with period $N$. Note that $\xi(t)$ is only defined for even $N$, since every $\sin(\cdot)$ needs a corresponding $\cos(\cdot)$. Considering the application, this restriction seems mild.

To verify that $\xi(t)$ is a Gaussian oscillator, the relations $\langle \epsilon_i \rangle = 0$ and $\langle \epsilon_i \epsilon_j \rangle = \delta_{ij}$ will be useful. First, note that $\xi(t)$ is a sum of Gaussian random variables at each time $t$; therefore, $\xi(t)$ also has a Gaussian distribution. This means that its mean $\langle \xi(t) \rangle$ and covariance $\langle \xi(t_1)\xi(t_2) \rangle$ characterize it completely.

From the definition in Equation 3, and the property $\langle \epsilon_i \rangle = 0$, it is clear that $\langle \xi(t) \rangle = 0$. The covariance requires more work. It involves a double sum, but since $\langle \epsilon_i \epsilon_j \rangle = \delta_{ij}$, only the $i = j$ terms survive:

$$\langle \xi(t_1)\xi(t_2) \rangle = \frac{2}{N} \sum_{i=1}^{N/2} \left[ \sin\left(\frac{2\pi i t_1}{N}\right) \sin\left(\frac{2\pi i t_2}{N}\right) + \cos\left(\frac{2\pi i t_1}{N}\right) \cos\left(\frac{2\pi i t_2}{N}\right) \right] \tag{4}$$

$$= \frac{2}{N} \sum_{i=1}^{N/2} \cos\left(\frac{2\pi i (t_2 - t_1)}{N}\right). \tag{5}$$

Note that this depends only on $(t_2 - t_1)$: these Gaussian oscillators are *stationary*, so no keyframes are singled out. Note too that $\langle \xi(t)^2 \rangle = 1 \ \ \forall t$, as required for Gaussian oscillators.

# 4 Comparison

This section introduces criteria to judge the quality of each type of Gaussian oscillator. We will assume throughout that the animation is composed of independent Gaussian oscillators, which is usually the case. Great circle animations[3] violate this assumption, so we must begin by clarifying their role.

Great circle animations are very similar to the $N = 2$ case of delocalized oscillators ($DO_2$ for short). Both techniques require $2d$ random variables

to visualize a $d$-dimensional Gaussian, and both techniques produce orbits which are confined to a (randomly-oriented) plane. In fact, great circle animation is a subset of measure zero of the $DO_2$ family: the former always produces circles, but the latter can also produce ellipses. Therefore, we assume that our results for $DO_2$ animations are a useful heuristic for the quality of the great circle animations.[4]

## 4.1 Statistical Properties

How well does an animation technique represent the Gaussian distribution? This question has two possible meanings, but previous literature has only considered one.

So far, people have taken this question to refer to a *family* of animations. Specifically, each family's distribution at each individual time is marginally Gaussian. This interpretation is certainly important, which is why we built it in to the definition of the Gaussian oscillator (Equation 1).

However, the properties of *individual* animations are even more important. (After all, a typical visualization contains only a single animation, not a family of them.) The frames of an individual animation define a probability distribution. The closer this distribution approximates the standard Gaussian, the better the technique.

To illustrate the distinction between family and individual, consider the simplest Gaussian oscillator: a single Gaussian draw, constant for all time. The marginal distribution over this family is clearly the standard Gaussian at all times. But the marginal distribution for a *single* animation is a point mass, which is very different from a Gaussian.

This motivates the following two-pronged approach: first, measure how well an individual animation represents the Gaussian; second, compute the *distribution* of this metric across all animations in the family. The Kolmogorov-Smirnov (KS) statistic — essentially, the widest vertical gap between two cumulative distribution functions — is a convenient, widely-used technique to compare two distributions. The KS *distribution* summarizes the fidelity of a Gaussian oscillator *family*: distributions which tend toward smaller values indicate better techniques.[5]

[4] Basically, this assumption amounts to ignoring the fact that the dimensions of a great circle animation are not independent.

[5] Of course, we could compute distributions for any other metric which compares probability distributions. The KS metric is not special; we chose it because it is both

As a basis for comparison, consider the KS distribution for a given number $N$ of independent normal draws, without interpolation.[6] Larger samples represent their distributions more faithfully; therefore, we expect their KS distributions to shift towards zero. Figure 3(a) shows the results for 2, 10, and 50 draws. It is clear that larger samples indeed have smaller KS statistics.

The next three subfigures compare two types of Gaussian oscillator — trigonometric interpolation, and the delocalized oscillators this paper introduces — to the equivalent number of independent draws. The figure shows that interpolating between the independent draws (orange curve) more faithfully represents the normal distribution than the independent draws themselves (green curve), since the orange curve puts more mass on lower KS values. However, the delocalized oscillators (purple curve) are significantly better than either in every case shown here.

Thus, not only is their motion more smooth and natural, but the delocalized oscillators also represent the standard normal more faithfully.

However, this conclusion only holds for a fixed number of samples. If more samples can be added continually (i.e., for an online animation), then the fidelity will continue to improve the longer the animation runs. Interpolation has this capability, and delocalized oscillators do not.

## 4.2 Kinematic Properties

Gaussian animations treat the curve as a quasi-physical moving object. We therefore investigate the most basic kinematic properties of that motion: velocity and acceleration. These distributions capture information which eludes the marginal *position* distribution (which is the same by definition for all Gaussian oscillators).

Figure 4 shows the results. The velocity curves for interpolation change discontinuously at every keyframe. This makes the acceleration effectively infinite there. At every keyframe, the oscillator receives an "impulse," which distractingly changes its motion. By contrast, the delocalized oscillators' motion is perfectly fluid. Each time is statistically equivalent to each other time — not just for the oscillator's position, but for its velocity and acceleration too.

simple and widely used.

[6] To be clear: these are not *themselves* Gaussian oscillators; they are merely a benchmark.

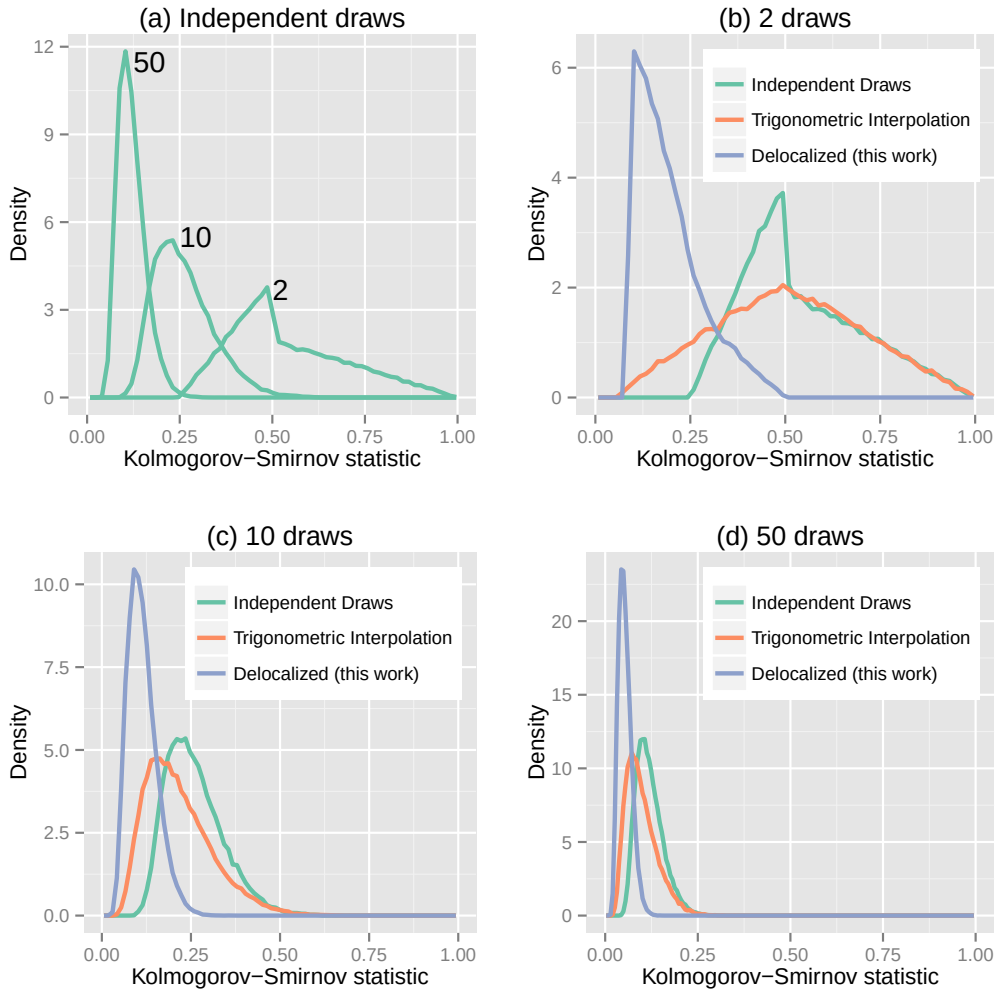


Figure 3: How faithfully does each Gaussian oscillator represent the normal distribution? This figure uses the Kolmogorov-Smirnov distribution to answer that question. (Each curve is a histogram of 500,000 draws.) (a) Independent draws from the normal distribution (which are not Gaussian oscillators) provide a basis for comparison. As expected, taking more draws shifts the distribution towards smaller values, indicating a better fit. (b), (c), and (d) show the Gaussian oscillators for 2, 10, and 50 draws, respectively, alongside the same number of independent draws. Trigonometric interpolation is always more faithful than the same number of independent draws, but the delocalized oscillators this paper introduces are significantly better than both.

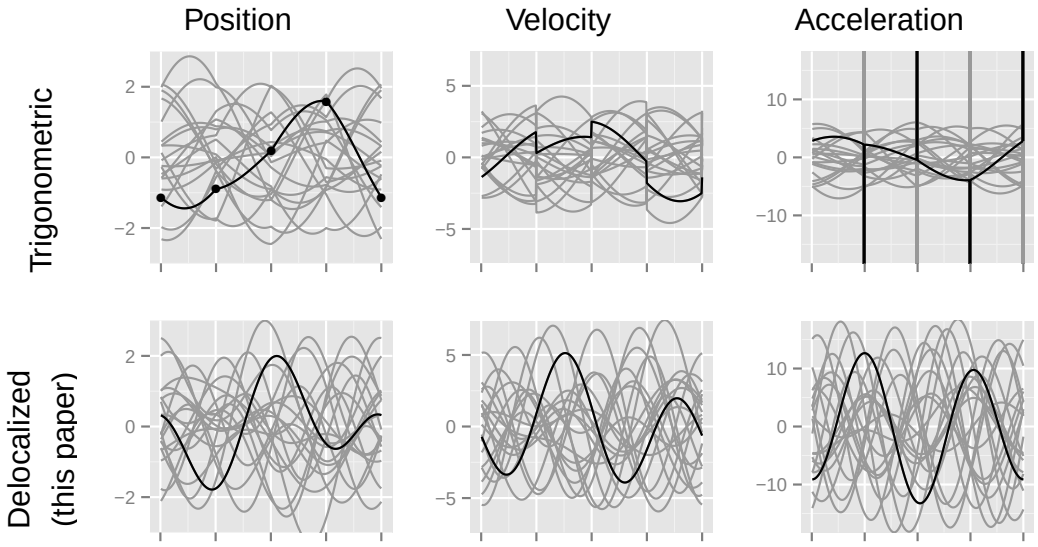


Figure 4: The basic kinematics of two types of Gaussian oscillators: the standard trigonometric oscillators, and the "delocalized oscillators" this paper introduces. Both populations are illustrated with 20 sample paths (grey), one of which is randomly chosen and highlighted for illustration. The velocity and acceleration graphs for the Trigonometric oscillators clearly show that keyframes are treated specially. By contrast, all times for the delocalized oscillators are statistically equivalent.

To understand these different behaviours, we view these oscillators as Gaussian processes, and compare their covariance functions $k(t_1, t_2)$. Delocalized oscillators have a *stationary* covariance (Equation 5): it depends only on $(t_2 - t_1)$, not on $t_1$ or $t_2$ individually. Trigonometric interpolation has a non-stationary covariance:

$$k_{\mathrm{TI}}(t_1, t_2) = \sum_{i=1}^{N} b_i(t_1) b_i(t_2), \tag{6}$$

where $b_i(t)$ is the $i$th basis function (see Figure 2).[7] This non-stationarity gives $k_{\mathrm{TI}}(t_1, t_2)$ a stepped, "blocky" appearance (see Figure 5(a)).

To avoid any frames being treated specially, we should prefer Gaussian oscillators with stationary covariance. If we further want motion that is smooth and natural, we should choose a covariance which is continuous and differentiable at $t_1 = t_2$.[8]

## 4.3 Time Correlation

In Monte Carlo simulations, autocorrelation is undesirable. In Gaussian animations, it is desirable at short time intervals; otherwise, the motion could not be continuous. Nevertheless, minimizing autocorrelation at longer times is still a virtue.

A useful metric is the autocorrelation at time intervals of one unit. For interpolation, this is the mean time between independent draws; hence, we could reasonably hope for a unit-time autocorrelation near 0. The best possible result would be an autocorrelation of *exactly* 0, for all time intervals longer than this.[9]

Trigonometric interpolation achieves perfectly uncorrelated unit-time intervals, but only for keyframes. Between the keyframes, the same draw $\varepsilon_i$ contributes to both values, meaning the correlation will be nonzero. In the worst case, the two values at $\Delta t = \pm\frac{1}{2}$ relative to a keyframe have a correlation of $\frac{1}{2}$, which is rather high.

[7] Note that the covariance from one keyframe to the next (e.g., $k_{\mathrm{TI}}(0, 1)$) is 0, as we would expect. But the covariance *between* keyframes can be quite high, even for the same interval, e.g., $k_{\mathrm{TI}}(0.5, 1.5) = \frac{1}{2}$.

[8] Since covariance functions are symmetric, this implies the slope should be 0 here.

[9] If the animation is periodic, this condition applies to the *smallest* time interval between two points. For example, an interval of one period should be treated as an interval of 0.

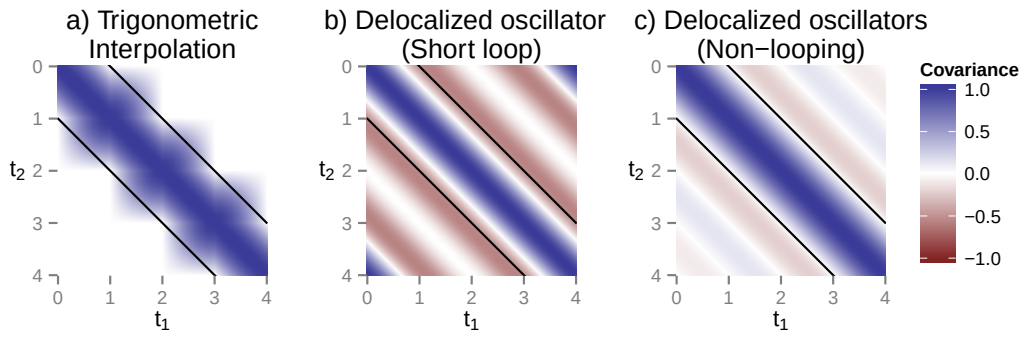


Figure 5: Example covariance functions $k(t_1, t_2)$ for various types of animations. The black bars represent the unit-time autocorrelation; the ideal covariance function would be zero everywhere outside this central stripe. a) Trigonometric interpolation. The covariance vanishes at long times, but is nonstationary (leading to jerky motion). b) Looping delocalized oscillators with a short period. The covariance is stationary, but intermediate times have strong anticorrelation. c) Delocalized oscillators in the infinite-period limit (i.e., non-looping). The anticorrelation is still nonzero, but much smaller.

The unit-time autocorrelation for delocalized oscillators is

$$\langle \xi(t)\xi(t+1) \rangle = \frac{2}{N} \sum_{i=1}^{N/2} \cos\left(\frac{2\pi i}{N}\right) = \frac{2}{N}. \tag{7}$$

Thus, delocalized oscillators are slightly anticorrelated at single-unit intervals[10]. For small $N$, this autocorrelation is worse than interpolation, even becoming as high as -1 (perfect anticorrelation) for the $N = 2$ case.[11] However, for $N > 4$, delocalized oscillators are always better than the worst case for interpolation. Higher $N$ values improve the autocorrelation (and, as noted earlier, the fidelity as measured by the KS distribution).

These benefits of higher $N$-values motivate us to consider the infinite-$N$ limit. The sum in the covariance function (Equation 5) can be converted to an integral by taking $x \equiv \frac{2i}{N}$:

$$\lim_{N\to\infty} \frac{2}{N} \sum_{i=1}^{N/2} \cos\left(\frac{2\pi i(t_2 - t_1)}{N}\right) = \int_0^1 \cos\left(\pi(t_2 - t_1)x\right)\, dx \tag{8}$$

$$= \operatorname{sinc}\left(\pi\left(t_2 - t_1\right)\right), \tag{9}$$

where $\operatorname{sinc}(0) = 1$, and $\operatorname{sinc}(x) \equiv \sin(x)/x \quad \forall x \neq 0$.

Thus, in the infinite limit, the autocorrelation vanishes at all intervals which are nonzero integer multiples of 1. The disadvantage is the considerable computational expense, both in time and space (as mentioned in Section 2.1). Furthermore, most intervals longer than one unit have a small but nonzero autocorrelation (see Figure 5(c)).

## 4.4 Summary of Metrics

We now summarize the relative merits of the various Gaussian animation techniques.

The most important metric is the Kolmogorov-Smirnov distribution, because it measures how well individual animations represent the standard normal. Delocalized oscillators and great circles exhibit significantly better

[10] This is valid at all times, since unlike interpolated animation, the delocalized oscillators are stationary.

[11] This anticorrelation of -1 clearly also applies to great circle animation. One time unit corresponds to half the period; therefore, all values are negated.

KS distributions than interpolation, if the number of independent samples is fixed. Taking more samples improves every technique except great circles, which cannot use more than two. These considerations point to delocalized oscillators as the most faithful offline technique considered here. Interpolation, however, performs better in the online case, since it can perpetually incorporate new random samples.

The next most important criterion is the quality of motion. Ideally, no frames should have preferred status, and the motion should be smooth. Delocalized oscillators and great circles excel here, while interpolation produces distracting, jerky motion. (Note that any future techniques with smooth, stationary covariance would also yield excellent quality motion.)

Finally, we want to maximize the mutual independence of all frames, by minimizing autocorrelation beyond what is necessary. Here, no single technique is clearly best. Interpolation limits autocorrelation to intervals of at most two units, but the precise distance is different for different frames. Delocalized oscillators with long period have low autocorrelation at intervals beyond 1 unit, but it never quite reaches zero.

Overall, delocalized oscillators considerably improve on the state of the art for offline Gaussian animations. They represent the normal distribution much more faithfully, and they do so with fluid, natural motion, treating no frames specially.

# 5 Future Work

## 5.1 Animating non-Gaussian distributions

This paper has shown how to visualize arbitrary Gaussian distributions using continuous animations. The question naturally arises which other distributions can be visualized in this way. Some distributions are clearly impossible, but several others seem tractable.

First, consider a "nearly-Gaussian" distribution, i.e., one whose probability density $\rho(x)$ differs from the Gaussian density $\nu(x)$ by some known factor (which naturally depends on $x$). The time which a standard Gaussian oscillator spends at $x$ is proportional to $\nu(x)$. To make it proportional to $\rho(x)$ instead, we can modulate the delay before the next frame by a factor $(\rho(x)/\nu(x))$. In essence, we are warping time to capture the subtle departures of the true distribution from the Gaussian.

A more challenging case is distributions with multiple regions that are connected only thinly (or even disconnected entirely). Time warping seems infeasible here for several reasons. First, the oscillator would need to move infinitely quickly in the intermodal region. Furthermore, any single Gaussian is likely to be a very poor approximation to the distribution as a whole, so the time warping would need to be extreme.

A better alternative would be to show a separate animation for each region. A Gaussian can better approximate individual modes rather than the entire distribution. And the total probability mass of each mode can be indicated beside its animation.

Discrete distributions present a greater challenge: obviously, they cannot be represented by continuous animations. However, even here there are special cases which admit partial success. Consider a Poisson distribution (representing, e.g., neutron counts in a scattering curve[4]). If we perform an Anscombe transform[1], the transformed data follows the Gaussian distribution to an excellent approximation. One could animate the transformed distribution with Gaussian oscillators, "un-transform" the result, and round to the nearest integer. While the result would not be truly continuous, it would change in an ordered way which is easy for the eye to follow.

Animations are highly useful for visualizing complex distributions. With the Gaussian case now well in hand, other distributions beg to be explored.

## 5.2 Alternative time-domain covariance functions

Let us consider again the criteria outlined in Section 4, and try to engineer a superior Gaussian oscillator.

The most important criterion is the KS distribution, since it is vital that the animation represents the normal distribution faithfully. The best possible way to fulfill this is to use an online technique, which can perpetually incorporate new random samples. We also want smooth, natural motion, with no frames treated specially; this implies a smooth, stationary covariance function. Finally, the autocorrelation should be zero for all intervals longer than one unit.

We can satisfy all these criteria simultaneously if our covariance has *compact support*. For example, consider this piecewise polynomial covariance

function:

$$k(\tau) = \begin{cases} (1-\tau)^6(12.8\tau^3 + 13.8\tau^2 + 6\tau + 1) & 0 \leq \tau \leq 1 \\ 0 & \tau > 1, \end{cases} \tag{10}$$

where $\tau$ is the time interval magnitude $|t_2 - t_1|$. Curves sampled from this distribution are twice differentiable everywhere[5]; hence, their motion will be smooth and natural. Their autocorrelation completely vanishes at all intervals longer than one unit: the best possible result. The covariance is stationary, so no frames can be special in any way. Finally, they permit infinite non-repeating animations with finite resources[12], just as interpolation does.

This, then, is the bright future we envision for animated uncertainty visualization. `R` packages which output interactive web apps, such as `shiny`[6], can be augmented to support animated output. Rather than a static image, or a finite looping `.gif`, the curve continuously moves in a smooth and natural way: never repeating its position, and visualizing more and more of the distribution.

# 6 Conclusions

Animations are a useful and beautiful technique to visualize uncertainty in curves and surfaces. This paper introduced a unifying concept, the "Gaussian oscillator," to compare existing techniques and pave the way for future ones. For the first time, we have measured the quality of individual animations rather than populations; in so doing, we have uncovered significant differences between techniques. We also introduced a new technique, *delocalized oscillators*, with many advantages: its motion is fluid and natural; it is efficient to compute; and it represents the underlying distribution more completely. Finally, we have revealed that Gaussian animations were simply a special case of Gaussian processes all along. We expect this relationship will bear much fruit and lead to even better animations in the future.

[12] The appendix explains in detail how to do this.

# 7 Acknowledgements

I gratefully acknowledge the help and support which made this work possible. First, I am thankful to the National Institute of Standards and Technology, where I was employed when I initially developed delocalized oscillators. Bob McMichael greatly expanded the utility of this work by pointing out that it applies to parametric models, not just Gaussian processes. Finally, Yarin Gal introduced me to the delightful and mesmerizing animations of Philipp Hennig.

This paper is written using the `knitr` package[8], which enabled me to include the `R` source code to generate my figures directly in the source text. This source is available on github: `https://github.com/chiphogg/paper_gaussian_oscillators`. I am also thankful to Melissa DeLucchi, who provided valuable feedback on a draft of this paper.

# Appendix: efficient online animations

Suppose we have a Gaussian oscillator timetrace, consisting of $n$ points. Can we extend the timetrace by computing the $(n+1)$st? If so, what is the computational cost?

This appendix will show that we *can* extend Gaussian oscillators, but the computational cost is unacceptably high. Most Gaussian oscillators are $\mathcal{O}(n^2)$ in both time and space. Oscillators with *compact support* are the exception: they can be computed very efficiently ($\mathcal{O}(1)$ with respect to $n$). This suggests that compact support oscillators represent the future of online Gaussian animations.

## The general case

Let us treat the timetrace as a Gaussian process. This means we have generated $n$ i.i.d. standard normal variates, $\xi^{(n)} \equiv \{\xi_1, \cdots, \xi_n\}$, and the lower-Cholesky decomposition $L^{(n)}$ of the covariance matrix $K^{(n)}$. The first $n$ points are given by

$$\alpha^{(n)} = L^{(n)} \xi^{(n)}. \tag{11}$$

The covariance matrix $K^{(n+1)}$ for the first $(n+1)$ points can be written

in block diagonal form:

$$K^{(n+1)} = \begin{bmatrix} K^{(n)} & k^{(n)} \\ k^{(n)\intercal} & 1 \end{bmatrix}, \tag{12}$$

where $K^{(n)}$ is the covariance matrix for the first $n$ points, and $k^{(n)}$ is a vector giving the covariance of the new point with each old point. The Cholesky root can be similarly decomposed:

$$L^{(n+1)} = \begin{bmatrix} L^{(n)} & 0 \\ \ell^{(n)\intercal} & \gamma \end{bmatrix}, \tag{13}$$

where $L^{(n)}$ is the Cholesky root of $K^{(n)}$, $\ell^{(n)}$ is an unknown vector, and $\gamma$ is an unknown scalar.

The $(n+1)$st point is

$$\alpha_{n+1} = \begin{bmatrix} \ell^{(n)\intercal} & \gamma \end{bmatrix} \xi^{(n+1)}. \tag{14}$$

At this point, we drop the $(n)$ and $(n+1)$ superscripts for convenience. The definition of the Cholesky root gives the following block matrix equation:

$$\begin{bmatrix} LL^\intercal & L\ell \\ \ell^\intercal L^\intercal & \ell^\intercal \ell + \gamma^2 \end{bmatrix} = \begin{bmatrix} K & k \\ k^\intercal & 1 \end{bmatrix}, \tag{15}$$

which yields the vector equation

$$L\ell = k \tag{16}$$

and the scalar equation

$$\ell^\intercal \ell + \gamma^2 = 1 \tag{17}$$

Since $L$ is lower-triangular and $k$ is known, Equation 16 can be trivially solved for $\ell$, and we can compute the $(n+1)$st point.

However, the computational cost is catastrophic. Imagine we leave the animation running, so that $n$ continues to increase. The time to compute each new point by Equation 16 is $\mathcal{O}(n^2)$. The storage also increases as $\mathcal{O}(n^2)$, since we require the entire matrix $L^{(n)}$. This animation will consume all system resources at an ever-increasing pace, while delivering new points ever more slowly.

## Compact Support

To prevent this disaster, suppose the covariance matrix has compact support, so that the new value is correlated only with a finite number of previous values; say, at most $m$. This means that $\{k_1, \cdots, k_{n-m}\}$ are all 0. Equation 16 implies that $\{\ell_1, \cdots, \ell_{n-m}\}$ are also 0.

This yields enormous computational savings. Since $L$ is lower-triangular, only the final $m \times m$ submatrix actually contributes to the calculation of $\ell$ (by Equation 16). Similarly, Equation 14 only uses the $m$ most recent random variates, $\{\xi_{n-m+1}, \cdots, \xi_n\}$; all the preceding ones can be discarded. The important point to notice is that our computational requirements no longer grow without limit as the animation continues to run.

We can do still better.

Imagine now that the time interval between successive points is constant. Then each row of $K$ (after the $m$th) is equivalent to the previous row, shifted by one column. The same will be true for $L$. Since each row has the same $m$ nonzero values, they need only be computed once.

We also only need to store $m$ random values, since the $n$th value can replace the $(n-m)$th (which is no longer needed). The following R snippet[13] illustrates how to generate an infinite timetrace, given a precomputed vector `ell` of length `m`:

```
xi <- rnorm(m)
while (TRUE) {
  alpha <- xi %*% ell
  UpdateAnimation(alpha)
  xi <- c(tail(xi, -1), rnorm(1))
}
```

To summarize: extending an $n$-point Gaussian oscillator timetrace is generally $\mathcal{O}(n^2)$ in both time and space, which is unacceptable. But if the Gaussian oscillator has compact support, both costs drop to $\mathcal{O}(m^2)$ (where $m$ is small and constant). And if we furthermore evaluate that oscillator only at constant time intervals, all costs drop further to $\mathcal{O}(m)$.

[13] Note that this code is intended only to illustrate the concept; it is too inefficient for a real implementation.

# References


[1] F. J. Anscombe. The transformation of poisson, binomial and negative-binomial data. *Biometrika*, 35(3-4):246–254, 1948.

[2] Charles R Ehlschlaeger, Ashton M Shortridge, and Michael F Goodchild. Visualizing spatial data uncertainty using animation. *Computers & Geosciences*, 23(4):387–395, 1997.

[3] Philipp Hennig. Animating Samples from Gaussian Distributions. Technical Report 8, Max Planck Institute for Intelligent Systems, Spemannstraße, 72076 Tübingen, Germany, September 2013.

[4] Charles R. Hogg, Joseph B. Kadane, Jong Soo Lee, and Sara A. Majetich. Error analysis for small angle neutron scattering datasets using bayesian inference. *Bayesian Analysis*, 5(1):1–33, 03 2010.

[5] CE Rasmussen and CKI Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, 1 2006.

[6] RStudio and Inc. *shiny: Web Application Framework for R*, 2014. R package version 0.10.2.2.

[7] John Skilling. Bayesian solution of ordinary differential equations. In C.Ray Smith, GaryJ. Erickson, and PaulO. Neudorfer, editors, *Maximum Entropy and Bayesian Methods*, volume 50 of *Fundamental Theories of Physics*, pages 23–37. Springer Netherlands, 1992.

[8] Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2014. R package version 1.8.