Stochastic wave using Simplex Noise

Lennart Heib

August 2022

1 Preamble

The Wave generation is based on the description of Giacomo Moretti and Gianluca Rizello.

SCENARIO 2: Irregular stochastic waves (realistic)

1. Choose an energy period T_e and a significant wave height H_s (statistical wave parameters)
2. Consider a wave spectrum (e.g., Bretschneider, but there are others):

$$S_\omega(\omega) = 262.9\, H_s^2 T_e^{-4} \omega^{-5} \exp(-1054 T_e^{-4} \omega^{-4})$$

1. Discretise the frequency interval: $[\omega_1, \ldots, \omega_N]$, with fixed step $\Delta\omega$, and ω_N sufficiently larger than $2\pi T_e^{-1}$
2. Evaluate the excitation coeff. at the selected frequencies: $\Gamma_k = \Gamma(\omega_k)$
3. Compute: $A_k = \left(2\Delta\omega\, S_\omega(\omega_k)\right)^{0.5}$
4. Choose N random phases $\phi_k \in [0, 2\pi]$, and define $d(t)$ as:

$$d(t) = \sum_k \Gamma_k A_k \sin(\omega_k t + \phi_k)$$

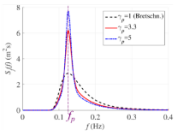

Figure 1: Algorythm for stochastic wave

2 Previous Version

Previously the algorithm was implemented exactly as described in Figure 1. The only source of randomness is the random shift of sine phases ϕ_k. The seed of this random shift was predefined so the wave would be the same each time the program executes. There were some minor problems with this version though.

1. Periodicity: Since the randomness only applies to the shift and is never changed the pattern will repeat.
2. The shape of the wave is dependent on the random shift. If two of the strong sine waves (waves near the maximum amplitude in Figure 1) have a similar random shift they will dominate the wave.

In Figure 2 the above descibed properties can be seen. The wave has a very similar pattern with a periodicity of about 50 seconds. Note, that the wave does not exacly repeat because of high frequency low amplitude waves contained in

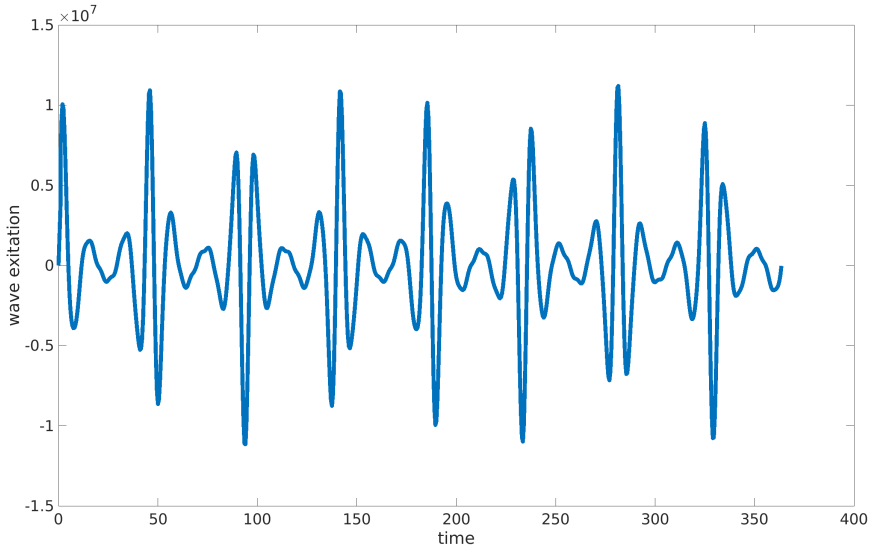

Figure 2: Example Wave generated using method described above

the signal. Still, there is enough similarity such that it is troublesome for our simulation.

3 Finding a suitable random number generator

The initial idea was to dynamically change the random phase shift ϕ_k so that the wave is less periodic. The problem is that standard random number generators are discontinuous and we can not simply apply a random phase shift to the signal and expect a continuous wave. We could, for example, change ϕ_k depending on the previous ϕ_k.

$$\phi_k^{t+1} = \phi_k^t + \phi_r \tag{1}$$

Where ϕ_r is a random increment. While this would result in a smooth random signal, the problem is, that we can only create the signal iteratively. If I wanted to evaluate the wave function $d(t)$ at $t = 100$, for example, the algorithm would have to calculate all 99 previous steps. The goal was therefore to find a smooth random number generator that could be evaluated at specific times. Luckily the simplex noise or the patented perlin noise are designed exactly for this purpose.

4 Simplex Noise

Using the simplex noise algorithm, we can create a string of numbers that are random but smooth. In two dimensions the result can be nicely visualized. In Figure 3 we can see the noise (scaled between 0 and 1 mapped to a grey-scale

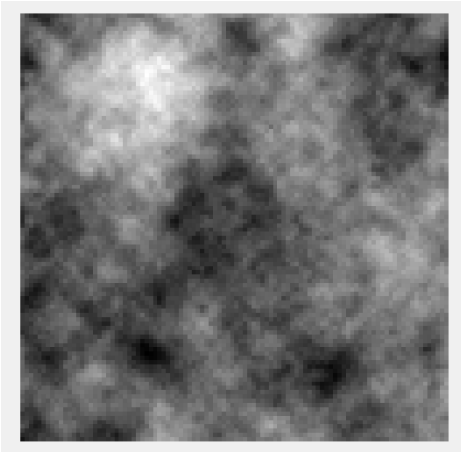

Figure 3: Two-dimensional simplex noise

image). The random numbers are not distributed randomly but rather clustered so the total noise is smoothed over the two-dimensional plane. Since we only need to a one dimensional smooth noise (we want the phase shift ϕ_k to smoothly be changed each increment) we can use the 2 dimensional noise and create a cyclic path through it. We want the path to visit each pixel/value exactly once and we alse want the path to loop. This kind of path is called a Hamiltonian Cycle.

5 Creating one dimensional cyclic noise from a two dimensional simplex

Without further restrictions there are infinitely many paths that fulfill this criterion and describe a Hamiltonian Cycle.

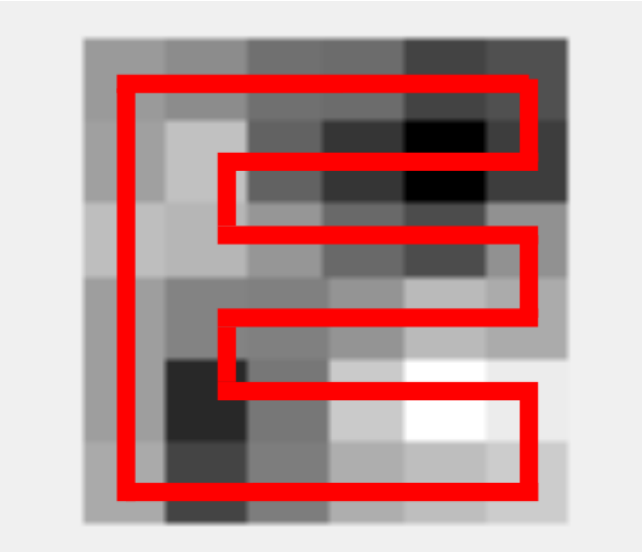

Figure 4: Hamiltonian Cycle on Simplex Map

Figure 4 shows such a Cycle. This particular path works only with an even number of rows but since the shape of the two dimensional input does not matter this condition can just be added as a restriction for the algorithm.

6 Evaluation and further elements

This implementation of the simplex noise fulfills both of our initial requirements. It is a smooth source of randomness and it can be accessed as an explicit function. Note, that this is only due to the cyclic nature of the path. There is however also a periodicity to this randomness, one of the problems we wanted to avoid using this implementation. This periodicity, however is arbitrarily large and not, like in the previous implementation, connected to the sinusoidal periodicity of the wave. It is for example easily feasible to create cycles with a periodicity of 10^4, i.e. a Simplex Map with 100x100 Pixels. And in this example this would only mean the random pattern repeats after 10000 iterations and not that the wave pattern would. In reality much smaller maps are sufficient. Some more algorithmic remarks:

1. Since the wave function also needs to be accessed at non-integer increments there is an additional linear interpolation between the discreet values.

2. To have an additional hyper parameter for the randomness I introduced a warp that stretches the input Map. For example a warp factor of 0.5 would create an additional increment between 2 values (linear interpolation) making the random noise "slower".

3. For the implementation I use a different map for each individual instance of randomness. Since the final wave is composed of a multitude if small waves, there are several separate instances. For example the random shift of wave 1 there is an individual map that corresponds that randomness. The maps are stored in a "persistent" variable so they are only created once.

7 Wave generation

Section 1 already gives an algorithm for creating a stochastic wave. The only modification added are the random number generation. The frequency shift as well as the amplitude of the individual waves are then changed every increment with the smooth random noise.

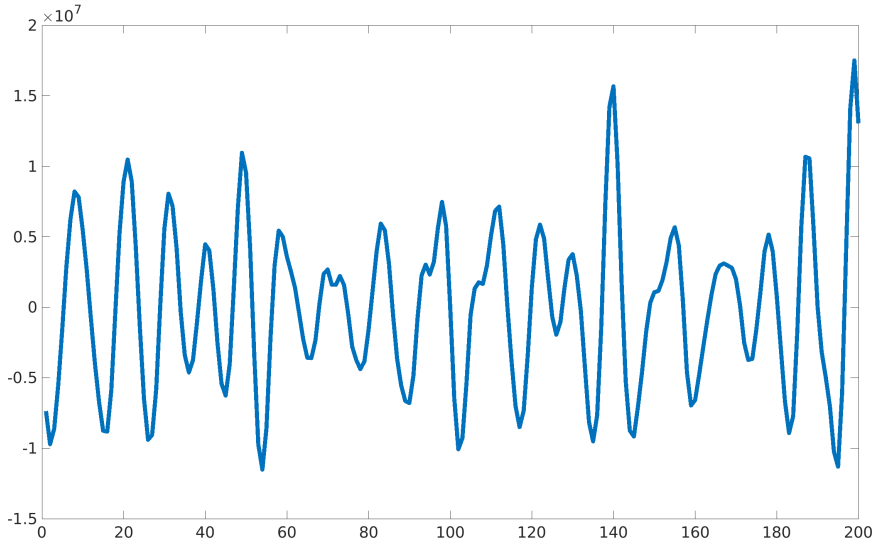

Figure 5: Stochastic Wave with simplex noise

8 Code structure

The code should have most sections explained. Still, here is a brief structure of the code:

1. SimplexNoise(m,Seed): Generates the two dimensional Simplex Maps. Needs dimension (must be even to work with Path and also square for convinience sake) and a Seed.This Code was not written by me but is available here

2. HamiltonianPathForSimplex(Simplex,p): Takes an integer and a Simplex map and returns the value at the position of the path.

3. Simplex(time,[Seed,warpFactor]): Stores the Maps based on Seed (Map generated with Seed 1 is stored as Map.v1 in a persistent variable). Also handles interpolation between integer increments as well as the warping factor.

4. SimplexStochasticWave(time,[warpFactor, Freqency Range, Amplitude Warp , PhaseWarp , Seed , TWave, HWave, NFreq, Debug]).Creates the wave function. Here all the hyper-parameters can be chosen in the arguments:

   - WarpFactor: High warp factor means slower RNG. Different Factor for Amp and Phase.
   - Frequency Range: The wave algorithm described in Section 1 requires the selection of frequencies. The Frequency range defines how large this range is.
   - NFreq: similar to the Frequency range we need to specify how many individual frequencies we want
   - Seed: This integer seed value influences all the indicidual seeds of the individual waves. New Seed –¿ completely new wave. Every aspect changes.
   - TWave, HWave: Period and Hight of dominant Wave. Not really hyper-parameters but rather define the shape of the wave.
   - Debug: If the debug flag is set the normal execution stops and the function enters into a loop where the wave is continuously drawn.

5. RunWaveExample.m Runs an example wave generation. Unlike the Debug function of the SimplexStochasticWave(). Here, the waves is drawn at once and not continuously (much faster).

9 TODO

1. Test different hyperparameters

2. Currently the Number of frequencies affects the amplitude of the wave. Needs fixin.