DIGITAL GENERATION OF NOISE-SIGNALS WITH ARBITRARY CONSTANT OR TIME-VARYING SPECTRA

J. Tückmantel, CERN, Geneva, Switzerland

Abstract

Noise sources in the RF system of an accelerator produce longitudinal emittance increase or particle loss. This noise is inherent, from the beam-control system electronics, external sources or high power components, or can be purposely injected for a specific need such as bunch distribution modification or controlled emittance increase. Simulations to study these effects on the beam require precise reproduction either of the total noise measured on the hardware, or of the noise spectrum to be injected and optimized to produce the desired changes. In the latter case the 'optimized' noise source has also to be created in real-time to actually excite the beam via the RF system. This paper describes a new algorithm to create noise spectra of arbitrary spectral density varying with cycle time. It has very good statistical properties and effectively infinite period length, important for long simulation runs. It is spectrally clean and avoids undesired mirror spectra. Coded in C++, it is flexible and fast. Used extensively in simulations it has also successfully created controlled emittance increase in the SPS by the injection of artificial real-time RF noise.

MOTIVATION

In the LHC coast undesired RF noise may blow up bunches and must be avoided. However, added noise is required to blow up bunches during the ramp to produce beam stability in both the SPS and in the LHC. In this case the noise excitation spectrum has to follow the varying synchrotron frequency spectrum. To study these cases, a very flexible digital noise generator was developed for a simulation program [1]. Priviously single tailored noise-spectra for blow-up were created by complex hardware means (e.g. [2]). For the blow-up in the SPS with commercial instruments difficulties were encountered due to spectral tails, mirror-spectra and low flexibility. To solve this the digital noise generator was used by uploading its data onto an **A**rbitrary **W**ave **G**enerator and then playing back the waveform (for up to 6 s at 10 kHz rate), successfully blowing up the proton bunches in the SPS.

For LHC about 20 min of slow blow-up are needed requiring too much data for direct playback from an AWG. But it could be realized by timesharing with calculation, buffering and output on a single platform; this will even allow spectral corrections in real-time (spectral feed back).

NOISE WITH CONSTANT SPECTRUM

We start with a complex array $\{g_n\}$ filled with time-domain *white noise* data for $1 \le n \le N$, with N a multiple of 4. The spectrum obtained from $\{g_n\}$ by Fourier-transform, restricted to positive frequencies exclusively, is weighted with the user supplied spectral (amplitude) weight function. An inverse Fourier-transform results in a set $\{r_n\}$ of N time-domain data with the desired, still time-invariant, noise spectrum.

The weight function has to be defined in the range 0 to 1; later it will be scaled linearly onto the range f_{Low} to f_{Up} for the (possibly time-varying) absolute frequency band. The weight function should be *proportional* to the square root of the desired local spectral 'power' density; absolute amplitudes are irrelevant at this stage.

Practically $\{g_n\}$ is created by

$$g_n = \exp(2\pi \cdot i \cdot x_{2n}) \cdot \sqrt{-2 \cdot \ln x_{2n+1}} \quad (1)$$

with x_{2n} and x_{2n+1} generated by two independent calls to a *top quality* pseudo-random generator with equi-distributed output between 0 and 1, the 'Mersenne Twister' [3]. The latter has extensive freedom from sequential correlations and a quasi-infinite period length of about 10^{6000} data samples. The arrays $\{Re(g_n)\}$ and $\{Im(g_n)\}$ then present (mutually correlated) zero-centred Gaussian distributions with $\sigma=1$ (see e.g. [4])

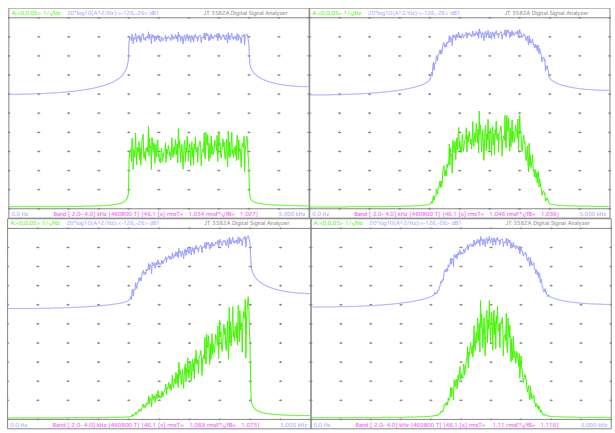

Fig. 1: FFT of noise data strings designed to target different (amplitude) spectral distribution: rectangular (top-left), trapezoidal (top-right), triangular (bottom-left, as used in the SPS blow-ups) and $\cos^2$ (bottom-right). There are *sharp* spectral ends and *no side-lobes nor tails*. (Green: linear scale, light blue: log scale; 'measured' in 'sliding average' mode with rectangular window)

To avoid periodicity and discontinuities for runs with more than N data, a second set $\{s_n\}$ is always used in parallel. It is created exactly as $\{r_n\}$ but with independent random numbers. With a constant parameter $0 \le \psi < 2\pi$ a new variable can be defined: $u_n = r_n \cdot \cos\psi + s_n \cdot \sin\psi$. It can be shown [4] that $\{u_n\}$ has the same spectral properties as its parent arrays. Here ψ will change as slowly as possible like $\psi_n = 2\pi \cdot n/N$, introducing a frequency component corresponding to the lowest one present in the initial

spectra, hence changing the spectral distribution only insignificantly with respect to the user-defined one.

Now at [n modulo N]=3N/4 $\{r_n\}$ and when n is multiple of N (i.e. [n modulo N]=0) $\{s_n\}$ can be replaced by a new, statistically independent array without discontinuities nor periodicities in $\{u_n\}$. Since even for indices only close to the above special ones the concerned weight functions $\sin\psi$ or $\cos\psi$ are close to zero, no transients appear at the change of arrays.

This procedure then guarantees, for all *practical* purposes, an *unlimited* supply of non-periodic noise data samples of high statistical quality having the desired spectrum on any sequential sub-set.

NOISE WITH VARIABLE SPECTRUM

When music, recorded (analogue) on a magnetic tape, is played back faster/slower than recorded, all frequencies appear higher/lower by the ratio of playback speed to recording speed; the same effect is used here in digital realization. Since one has to satisfy all irrational speed ratios, the time domain signal coming from the above Fourier transform has to be *interpolated smoothly*, or significant phantom side lobes may appear on the spectrum.

In a (discrete) spectral representation the highest represented frequency component f_{max} has in time domain only two data samples per oscillation, preventing smooth interpolation up to this frequency. But one can represent the precise equivalent of the present spectrum of N frequency channels as an L·N channel spectrum (with e.g. L=8) by appending at the high frequency end (L–1)·N channels with *zero amplitude* (Fig. 2). Then – after L·N-channel Fourier transform to time domain – all initial frequency channels are presented by L times the number of points per oscillation, allowing good smooth interpolation for the whole frequency range up to f_{max}. This method is applied here; practically it is incorporated in the creation of the stable spectra of the previous chapter, delivering an L times denser (complex) data stream ready for smooth interpolation

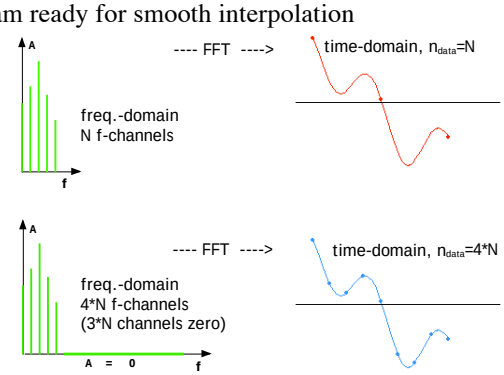

Fig. 2: Left plots: frequency domain. Right plots: time domain, dots→ discrete Fourier transform, lines→ the underlying (smooth) function. Top: original frequency domain representation, bottom: appended frequency domain representation. (displayed L=4, in the code L=8).

Such smooth interpolation with correspondingly chosen step-width allows *any* ratio of playback to original speed within the required range to be realized (Fig. 3). For a constant time step Δt (i.e. play-back rate $f_{Clock}=1/\Delta t$) it is then possible by continuous adjustment of the above interpolation step-width to create a data stream with a spectrum between f=0 and $f=\Delta f=f_{Up}(t)-f_{Low}(t)$ such that its amplitude distribution, x-scaled from [0,Δf] to [0,1], is identical to the user-defined spectral amplitude-function.

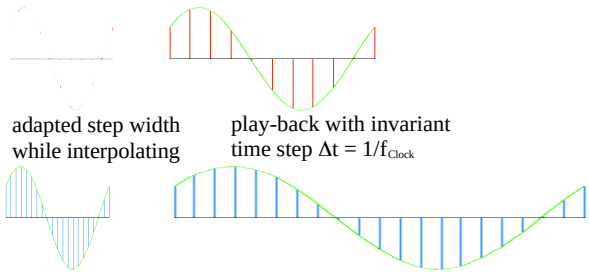

Fig. 3: Left plots: Smoothly interpolated data from *the same* Fourier transform into time-domain; left-bottom: half the step width of left-top. Right plots: playback with *identical* time-steps $\Delta t=1/f_{Clock}$: 'bottom procedure' produces data of half the frequency of 'top procedure'.

This spectrum has to be mixed up with f_{Low} to get the band between f_{Low} and f_{Up} as desired. This can be done easily by multiplying with the (unit) complex phase factor e_k, turning at time $t=k\cdot\Delta t$ in the complex plane after each clock-tick Δt (in positive sense) as

$$(2)\quad e_{k+1} = e_k \cdot \exp(2\pi \cdot i \cdot f_{Low}^{(k)} / f_{Clock}); \quad |e_k| = 1$$

The upper index k of f_{Low} indicates that f_{Low} might have to be adapted to match the required, possibly time dependent, lower band limit frequency $f_{Low}(t=k\cdot\Delta t)$.

The real part (ignoring the imaginary part) of the final complex data is used as noise-data-output, avoiding any mirror spectra due to always-positive frequencies.

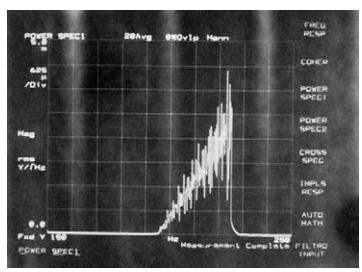

Fig. 4: Screenshot (Polaroid) from a digital signal analyzer: the measured (constant) 'triangular noise' (195 Hz – 225 Hz) was created by the digital noise generator and played back from an AWG at 10 kHz rate (applied for bunch blow-up in the SPS in coast).

Before transferring the data, it is scaled by a constant factor such that the rms-value of the total output stream corresponds to a user-defined rms-value X_0; intrinsically this rms-value remains constant even for variations of f_{Up} and f_{Low}. This means also that a change of the spectral *bandwidth* f_{Up}-f_{Low} causes the *local* noise 'power' to change inversely to the bandwidth scaling-factor. This can

be modified with the time dependent *relative* amplitude $a_{rel}(t)$ having the initial (and default) value 1.

The noise data is observed *stroboscopically* at the rate f_{Clock}, hence any line at frequency f (< $f_{Clock}/2$) has a twin line at f'= f_{Clock}–f (>$f_{Clock}/2$). To avoid unpredictable results, the desired spectral density function should only be specified for a range between $0 \le f_{Low} < f_{Up} < f_{Clock}/2$ (not f_{Clock}). In any case f_{Clock} has to be chosen large enough to avoid too much granularity of the output.

MONOCHROMATIC LINES

In LHC klystrons are used as power amplifier; these have a strong dependence of the RF phase from the DC voltage. Therefore RF phase noise shows up at multiples of the power grid frequency. The measured phase noise line spectrum has been reconstructed and the beam simulated [5]. Also 'brushing through a bunch' with a sliding monochromatic line to produce 'hollow bunches' has shown the desired effects on the bunch profile.

To apply this in practice a set of M monochromatic lines of given (real) amplitude $a^{(m)}$, frequency $f^{(m)}$ and starting phase $\psi^{(m)}$ ($1 \le m \le M$) can be 'switched on', alone or superimposed on the other 'smooth' noise. Each line starts with the complex status time-domain variable

$$a_0^{(m)} = a^{(m)} \cdot \exp\left(2\pi \cdot i \cdot \psi^{(m)}\right) \quad (3)$$

and advances (positively) at each clock-tick – labelled by the index k – as

$$a_{k+1}^{(m)} = a_k^{(m)} \cdot \exp\left(2\pi \cdot i \cdot f^{(m)} / f_{Clock}\right) \quad (4)$$

All lines can be produced in two modes: either with stable $f^{(m)}$ – as for power grid multiples – or automatically appearing as an integral part of a varying 'smooth' spectrum; in this case $f^{(m)}$ defines the position with respect to the *initial* f_{Low} and f_{Up}. The user may 'update manually' any 'fixed' f_m .

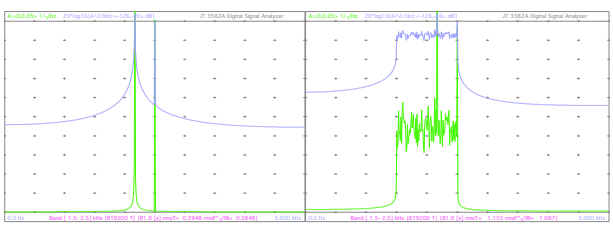

Fig. 5: Snapshot of two fixed lines (left) and a variable smooth spectrum with incorporated lines (right). (Green: linear scale, light blue: log scale)

PRACTICAL APPLICATIONS

The package (more details in [4]) is written in C++ and kept as self-sufficient as possible. Initialization is done by a call defining basic parameters such as f_{Clock}, the desired rms output value X_0 (valid for a_{rel}=1) and the spectral distribution function (-pointer); this function is valid for the whole run. Before any output the initial f_{Low}, f_{Up} and relative amplitude factor a_{rel} (if ≠1) have to be defined.

From then on each call to "NextVNoise( )" delivers one (real) noise data corresponding to the following clock-tick. As long as f_{Low}, f_{Up} and a_{rel} are not changed the spectrum remains constant; it is up to the user to update these parameter as function of time (clock-tick k) calling one of the *supplied* functions as "SetBandPos(flow,fup)". The output has a total rms-value of X_0 (for a_{rel}=1) as defined by the user; it is his task to convert (in hardware or software) the numerical output to the desired e.g. MV or degree.

Monochromatic lines – if required – have to be initialized and requested independently; the 'stable' line-frequencies may be updated as desired by similar calls.

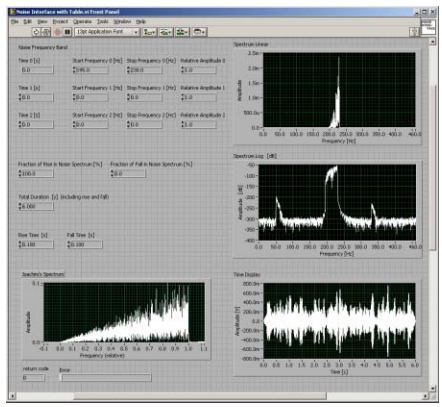

Fig. 6: **G**raphical **U**ser **I**nterface screenshot (digital) as used for the SPS blow-up tests. Top-left: GUI-fields for input specifications of the spectral shape (as trapezium) and three linear interpolation points for $f_{Low}(t)$, $f_{Up}(t)$ and $a_{rel}(t)$; bottom-left: [0–1] normalized spectrum, bottom-right: time-domain output representation. The absolute noise-amplitude was adjusted with the AWG output level.

CONCLUSION

The package has proven its utility and versatility in applications in real-time [6][7] and off-line simulations [5] for CERN's accelerators and future collider LHC.

ACKNOWLEDGMENT

The author wants to thank T. Bohl, T. Linnecar and E. Shaposhnikova for helpful discussions and U. Wehrle for his contributions to the hardware realization and the GUI.

REFERENCES

[1] J. Tückmantel, 'NoisySync', a simulation program for RF noise in synchrotrons, CERN, unpublished

[2] S. Ivanov, O. Lebedev, RUPAC 2002, Russia, Obninsk, October 2002

[3] Makoto Matsumoto and Takuji Nishimura, http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

[4] J. Tückmantel, CERN-LHC-Project-Report-1055

[5] J. Tückmantel, CERN-LHC-Project-Note-404

[6] CERN-AB-Note-2008-020

[7] G. Papotti et al., this conference TUPP059