Step 2a, revision 4 - Filter the tones, Detect the phase in a narrow tracking band

Input data file name DataFileName := "Data3.NoSpin.NoPhNoise.byte"

A priori Doppler Polynomial model file name DpcFileName0 := "Data3.Det.Pass0.tone0.fitcoeffs.txt" All zeroes so far

Pass1 Doppler Polynomial model file name DpcFileName1 := "Data3.Det.Pass1.tone0.fitcoeffs.txt"

Data set and processing essential parameters

Band Width $BW := 250 \cdot kHz$

Scan length (in samples) $Nt := 8 \cdot 1000 \cdot 1000$

N of points to input FFT segment $Npsi := 10 \cdot 1000$

Overlapping of FFT segments $Ovlp := 2$

Filter ratio $FR := 100$ Output bandwidth $BWo := BW \cdot FR^{-1}$ $BWo = 2500\ s^{-1}$

Number of samples in the output signal $Nto := \mathrm{floor}\left(Nt \cdot FR^{-1}\right)$ $Nto = 80000$

Number of points in output segment $Npso := Npsi \cdot FR^{-1}$ $Npso = 100$

Note the decimal number for FFT length !
And input length is abbreviated to make
a nice decimal number of output samples

Number of spectral points to extract to output FFT $Npfo := Npso \cdot 0.5 + 1$ $Npfo = 51$

Zero padding of output FFT $Npfz := Npso \cdot 0.5 - 1$ $jpfz := 0 .. Npfz - 1$ $Npfz = 49$

padding array for neg. frequencies $dpadd_{jpfz} := 0 + i \cdot 0$

Derived parameters (some of them are just for illustration)

Sampling rate $Sr := 2 \cdot BW$

And sampling interval $dt := Sr^{-1}$ Dimensionless $dtn := dt \cdot s^{-1}$

Total time span $Tspan := Nt \cdot dt$ $Tspan = 16\ s$

Input Time grid $jt := 0 .. Nt - 1$ $tt_{jt} := jt \cdot dt$

Output sampling interval $dto := dt \cdot FR$ $dto = 0.0002\ s$

Output Time grid $jto := 0 .. Nto - 1$ $tto_{jto} := jto \cdot dto$

Binning within FFT input and output segments $jpsi := 0 .. Npsi - 1$ $jpso := 0 .. Npso - 1$

Input Window function in time domain $$Wini_{jpsi} := \cos\left[\frac{\pi}{Npsi} \cdot (jpsi - 0.5 \cdot Npsi + 0.5)\right]$$

Output Window function in time domain $$Wino_{jpso} := \cos\left[\frac{\pi}{Npso} \cdot (jpso - 0.5 \cdot Npso + 0.5)\right]$$

Derived parameters , Continue

Number of FFT segments to process (accounting for overlap)

$$Nsegm := \mathrm{floor}\left(\frac{Nt}{Npsi}\right) \cdot Ovlp - (Ovlp - 1) \qquad Nsegm = 1599$$

shift (in samples) between overlapping segments

$$Oshifti := \frac{Npsi}{Ovlp} \qquad Oshifto := \frac{Npso}{Ovlp}$$

Frequency resolution of the input FFT

$$dfi := (Npsi \cdot dt)^{-1} \qquad dfi = 50\ s^{-1}$$

Filter set-up time

$$Tsetup := 0.5 \cdot Npsi \cdot dt \qquad Tsetup = 0.01\ s$$

Number of points at the start and the end of the output array to disgerard due to filter set-up time:

$$Npout\_setup := Tsetup \cdot dto^{-1} \qquad Npout\_setup = 50$$

Read Doppler Frequency polynomials

$Fcd0 := READPRN(DpcFileName0)$

$Fcd1 := READPRN(DpcFileName1)$

$Npf := Fcd0_0$ $\quad Npf = 3$ $\quad jpf := 0 .. Npf$

$Cf0_{jpf} := Fcd0_{jpf+2}$ Step 0 frequency polys (a-priory) are all zeroes here

$Cf1_{jpf} := Fcd1_{jpf+2}$ $\quad Tspanp := Fcd1_1$ $\quad Tspanp = 16.777216$ $\quad Tspanps := Tspanp \cdot s$

Combine all data sets $Cf := Cf0 + Cf1$ $\quad Cf_0 := Cf1_0$

Select the Doppler constant offset from the last pass data, well, it will not be used anyway, just keep a track on it

Make Phase polynomials $Cpp_0 := 0$ $\quad Cpp_{jpf+1} := 2\pi \cdot \frac{Cf_{jpf}}{jpf + 1}$ $\quad Npp := Npf + 1$

Frequency polynomial

$$Cf = \begin{pmatrix} 70048.91028919304 \\ 7052.868226490915 \\ -7888.813451185823 \\ 1.104737497866154 \end{pmatrix}$$

Phase polynomial

$$Cpp = \begin{pmatrix} 0 \\ 440130.2839129986 \\ 22157.23900708073 \\ -16522.292255857148 \\ 1.735317603720739 \end{pmatrix}$$

Defining the start/end bin of the filter, which will put tone lines in the center of the output band

We want to put the tone in the center of the output band: $BWoh := BWo \cdot 0.5$ with an accuracy of the input FFT resolution $dfi = 50\ s^{-1}$

We know, that several tones have certain offsets from the carrier line

			Start and End bins to extract		Frequency of the start bin	
carrier	$Fcc := Cf_0 \cdot Hz$	$Bsc := \mathrm{floor}\left[(Fcc - BWoh) \cdot dfi^{-1}\right]$	$Bsc = 1375$	$Bec := Bsc + Npfo - 1$	$Fstartc := Bsc \cdot dfi$	$Fstartc = 68750\ s^{-1}$
tones	$Fc1 := Fcc - 10000 \cdot Hz$	$Bs1 := \mathrm{floor}\left[(Fc1 - BWoh) \cdot dfi^{-1}\right]$	$Bs1 = 1175$	$Be1 := Bs1 + Npfo - 1$	$Fstart1 := Bs1 \cdot dfi$	$Fstart1 = 58750\ s^{-1}$
	$Fc2 := Fcc - 50000 \cdot Hz$	$Bs2 := \mathrm{floor}\left[(Fc2 - BWoh) \cdot dfi^{-1}\right]$	$Bs2 = 375$	$Be2 := Bs2 + Npfo - 1$	$Fstart2 := Bs2 \cdot dfi$	$Fstart2 = 18750\ s^{-1}$
	$Fc3 := Fcc + 20000 \cdot Hz$	$Bs3 := \mathrm{floor}\left[(Fc3 - BWoh) \cdot dfi^{-1}\right]$	$Bs3 = 1775$	$Be3 := Bs3 + Npfo - 1$	$Fstart3 := Bs3 \cdot dfi$	$Fstart3 = 88750\ s^{-1}$

Integrate the phase

$$Phdopp_{jt} := Tspanp \cdot \left[\sum_{jjp=2}^{Npf} \left[Cpp_{jjp} \cdot \left(\frac{tt_{jt}}{Tspanps} \right)^{jjp} \right] \right]$$

Make a segment time shift phase correction coefficient, actually a start bin of the filter can be selected in such way, that this coeff will be +1, although it can be even complex

$Fstartc \cdot Oshifti \cdot dt = 687.5$ $\quad Pssc := Fstartc \cdot Oshifti \cdot dt - floor(Fstartc \cdot Oshifti \cdot dt)$ $\quad Esc := \exp(i \cdot 2 \cdot \pi \cdot Pssc)$ $\quad Esc = -1$

$Fstart1 \cdot Oshifti \cdot dt = 587.5$ $\quad Pss1 := Fstart1 \cdot Oshifti \cdot dt - floor(Fstart1 \cdot Oshifti \cdot dt)$ $\quad Es1 := \exp(i \cdot 2 \cdot \pi \cdot Pss1)$ $\quad Es1 = -1$

$Fstart2 \cdot Oshifti \cdot dt = 187.5$ $\quad Pss2 := Fstart2 \cdot Oshifti \cdot dt - floor(Fstart2 \cdot Oshifti \cdot dt)$ $\quad Es2 := \exp(i \cdot 2 \cdot \pi \cdot Pss2)$ $\quad Es2 = -1$

$Fstart3 \cdot Oshifti \cdot dt = 887.5$ $\quad Pss3 := Fstart3 \cdot Oshifti \cdot dt - floor(Fstart3 \cdot Oshifti \cdot dt)$ $\quad Es3 := \exp(i \cdot 2 \cdot \pi \cdot Pss3)$ $\quad Es3 = -1$

Major function to do Phase Tracking,
Down-conversion, Filtering and Hilbert transform

```
MakeFiltX(Phcorr, Fbinstart, Fbinend, Es) :=
  for jjo ∈ 0 .. Nto − 1
    fout_jjo ← 0
  for jsegm ∈ 0 .. Nsegm − 1
    skip ← jsegm · Oshifti
    din ← READBIN(DataFileName, "byte", 0, 1, skip, Npsi)
    din ← din − 127
    din ← vectorize(din · Wini)
    phc ← submatrix(Phcorr, skip, skip + Npsi − 1, 0, 0)
    ephc ← vectorize(exp(i · phc))
    din ← vectorize(din · ephc)
    sp ← cfft(din)
    spo ← submatrix(sp, Fbinstart, Fbinend, 0, 0)
    spo_0 ← 1 · Re(spo_0)
    spo_(Npfo−1) ← 1 · Re(spo_(Npfo−1))
    spop ← stack(spo, dpadd)
    dout ← icfft(spop)
    dout ← vectorize(dout · Wino)
    for jjso ∈ 0 .. Npso − 1
      fout_(jjso+jsegm·Oshifto) ← fout_(jjso+jsegm·Oshifto) + dout_jjso · Es^jsegm
  return fout
```

Phase integration should be done differently in C

Multiple tones can be extracted in parallel,
using only 1 "big" input Fourier transform
and several "small" output FFTs.

Mathematically this filter is equivalent to PUB,
but better, because it allows arbitrary positioning
of output channels, (within the input FFT granularity),
and even imply different bandwidth for output channels.

And it's faster !

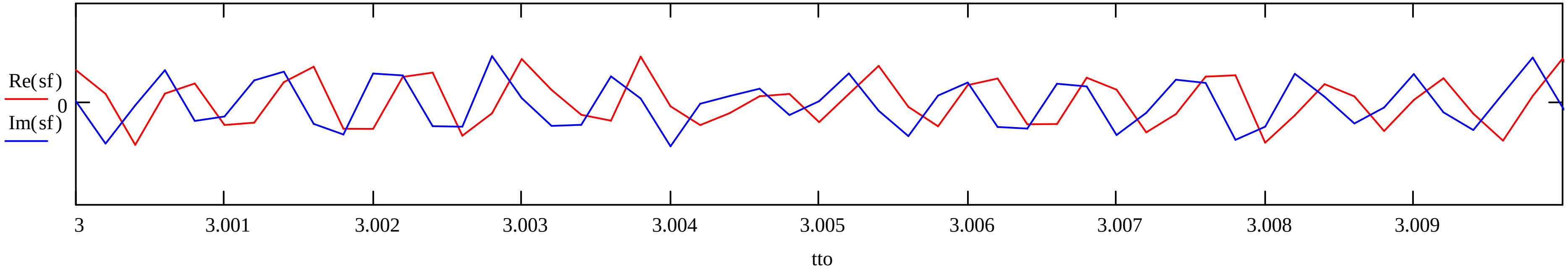
Although more demanding for higher precision..

This is a tricky part to make in C

Make a filtered signal for major tone

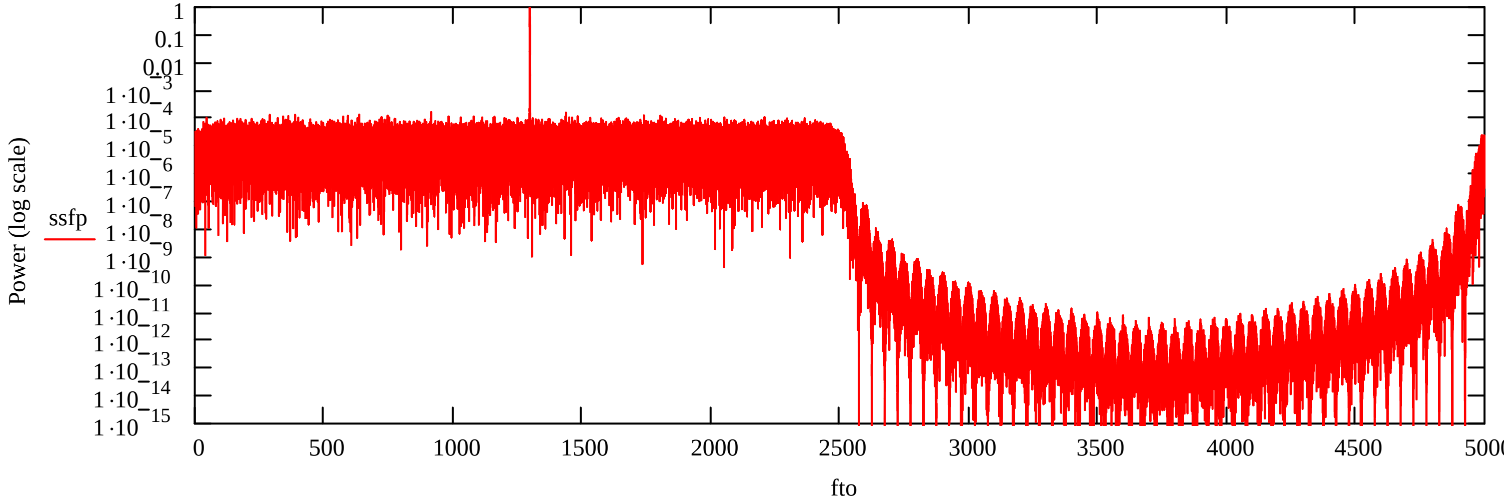
$sf := MakeFiltX(Phdopp, Bsc, Bec, Esc)$ $length(sf) = 80000$

fragment of the filtered complex signal in a time domain



Full length spectrum (two-sided FFT), shows good suppression of negative frequencies

$ssf := cfft(sf)$ $ssfp := \overrightarrow{(|ssf|)^2}$ $xssfp := max(ssfp)$ $ssfp := ssfp \cdot xssfp^{-1}$ $dfto := Tspan^{-1}$ $fto_{jto} := dfto \cdot jto$



Frequncy (Hz) output band

Determine the frequency of the max power $xf := FindMax(ssfp, fto, 0 \cdot Hz, 2500 \cdot Hz)$ $fmax := xf_1 \cdot dfto$ $fmax = 1299.061579255918\ s^{-1}$

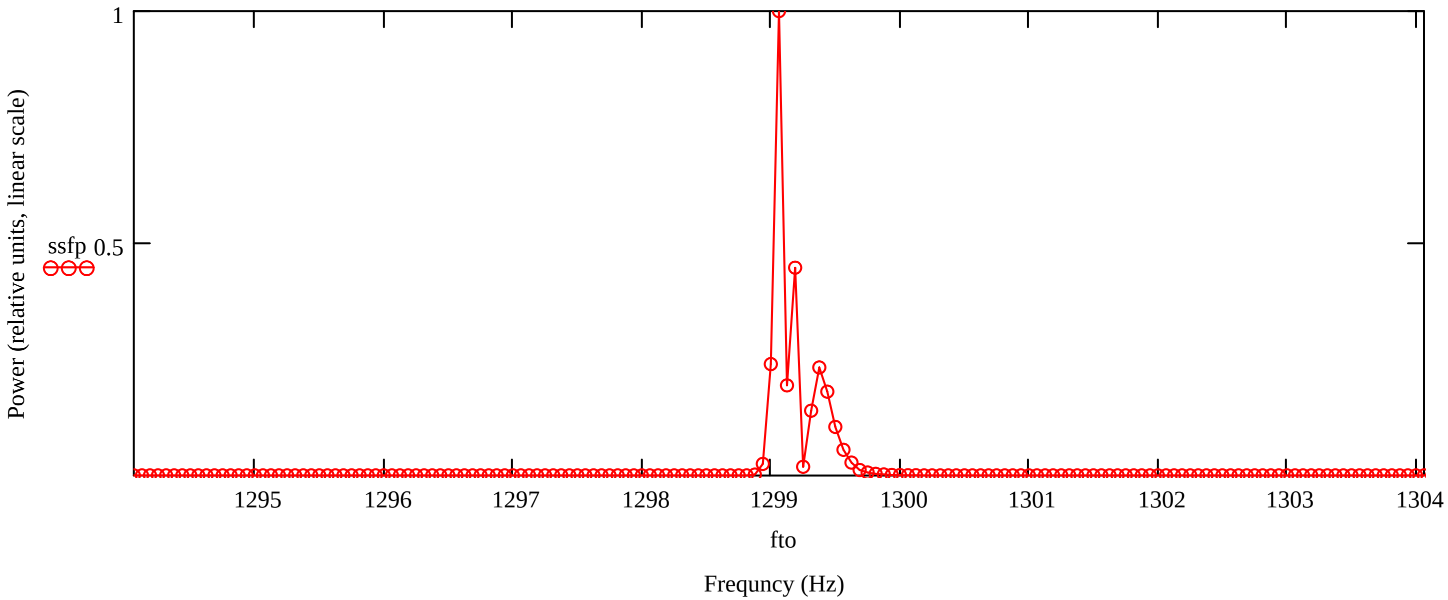
To make SNR estimation, extract the noise data in range 200 - 1000 Hz

$spnoise := submatrix\left(ssfp, 200 \cdot Hz \cdot dfto^{-1}, 1100 \cdot Hz \cdot dfto^{-1}, 0, 0\right)$ $length(spnoise) = 14401$ $stdev(spnoise) = 0.000013463398363$ $SNR := stdev(spnoise)^{-1}$

Predicted SNR (based on how this test signal was generated) is 53 dB for 1/16 of a Hz resolution

And here we have: $dbSNR := 10 \cdot log(SNR, 10)$ $dbSNR = 48.708453038649395$

Zoom into central line, all the power is concentrated in a sub-Hz wide line, SNR at 50 dB level.



Line power is still split between many spectral bins (each of 1/16 Hz wide), that's because the phase correction applied was not error free. Although, now in the narrow band we can make further perfection, like PLL it.

Detecting the phase of the tone in narrow band (call it PLL)

Set a narrower bandwidth, say 100 Hz. Decide to move the line to the center of this narrower band $BWn := 20 \cdot Hz$ $Ftarg := 0.5 \cdot BWn$

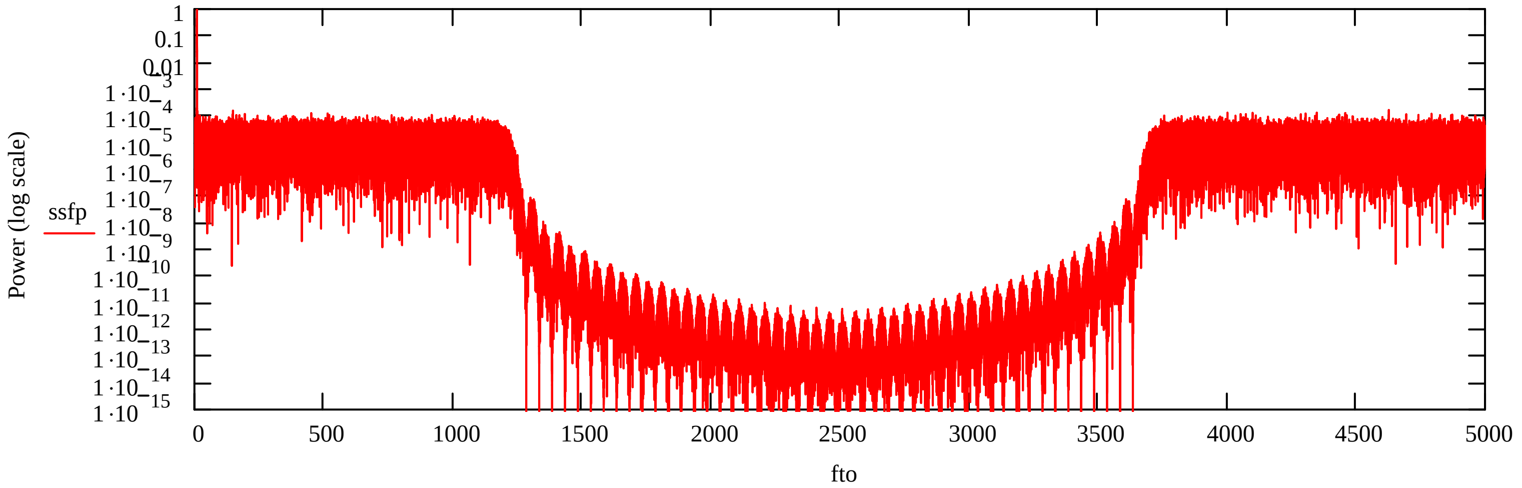
Check if the target frequency of is exactly representable in the frequency grid $\mathrm{floor}\left(\frac{Ftarg}{dfto}\right) \cdot dfto = 10\ s^{-1}$ Yes it is

Move the line to the target frequency by rotation if with a frequency: $Frot := fmax - Ftarg$ $Frot = 1289.061579255918\ s^{-1}$

$$sfc_{jto} := sf_{jto} \cdot \exp\left(2\pi \cdot i \cdot Frot \cdot tto_{jto}\right)$$

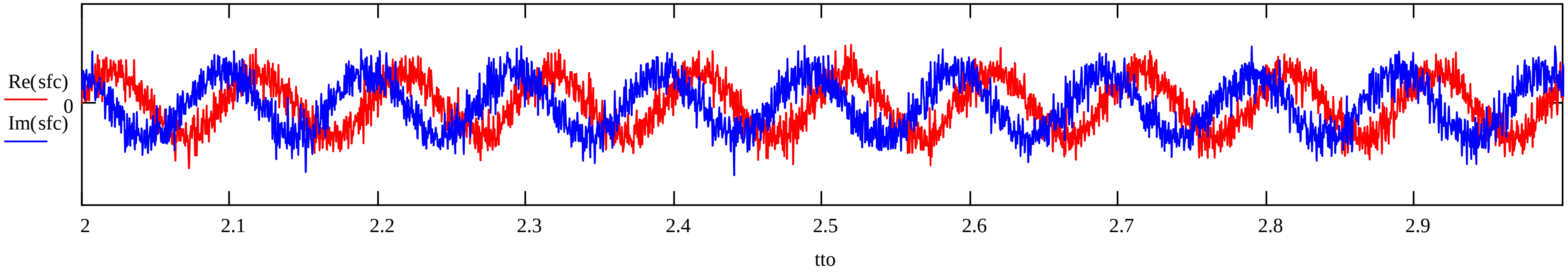
Check the spectrum $ssf := \mathrm{cfft}(sfc)$ $ssfp := \overrightarrow{\left(|ssf|\right)^2}$ $xssfp := \max(ssfp)$ $ssfp := ssfp \cdot xssfp^{-1}$ $dfto := Tspan^{-1}$ $fto_{jto} := dfto \cdot jto$

Line is close to the DC edge of the band now



Frequncy (Hz) in output band

Check the time domain pattern, resembles a sine wave, with noise still representing the full noise in 2500 Hz band



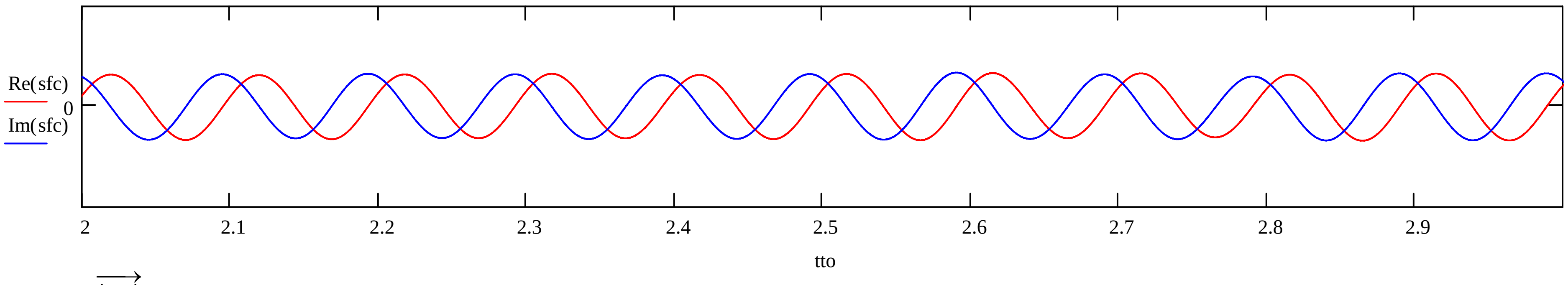
Make the BWn - wide filter around the line (note it's a one side filter)

$$ssff_{jto} := \mathrm{if}\left(fto_{jto} < BWn, ssf_{jto}, 0\right)$$

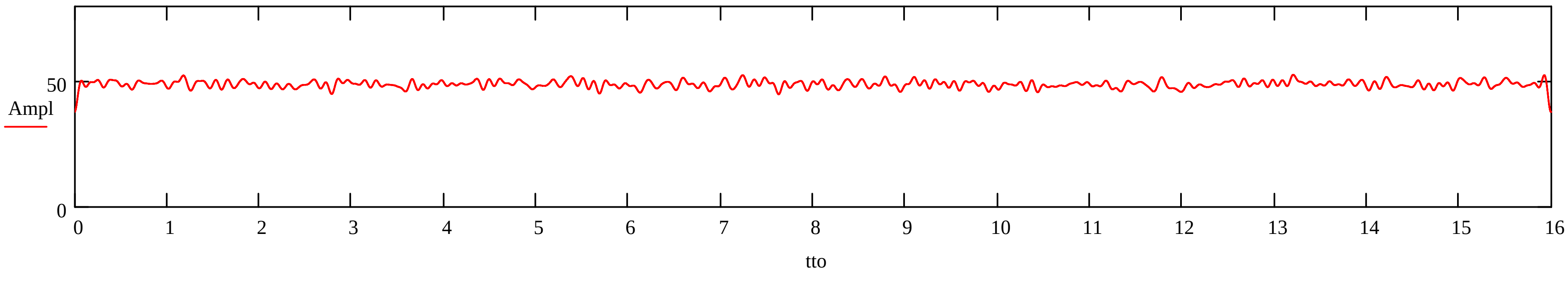
And get the signal back to time domain

$$sfc := \mathrm{icfft}(ssff)$$

Check the time domain pattern, even more likely the sine wave, not much of noise left after the last filter



$$Ampl := \overrightarrow{|sfc|}$$

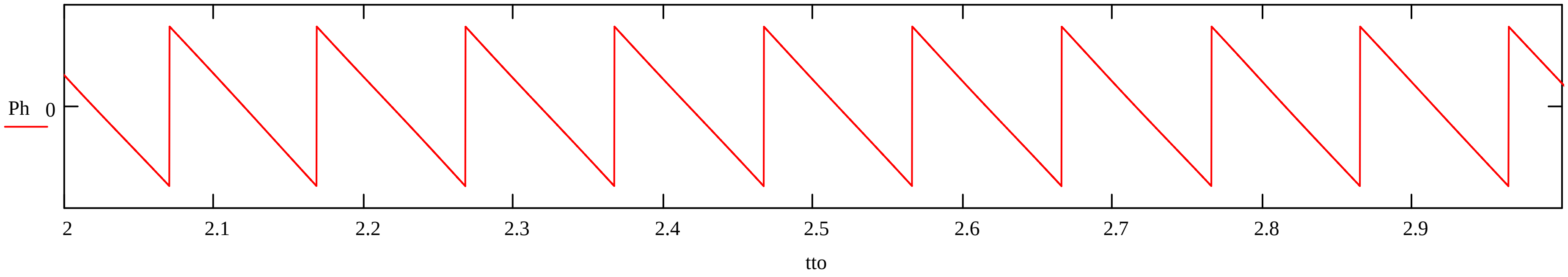


Note the amplitude drop at the start and at the end of the scan, which is due to filter set-up time

Get the phase

$$Ph_{jto} := \arg\left(sfc_{jto}\right)$$

Phase is wrapping over 2π each cycle of Ftarg



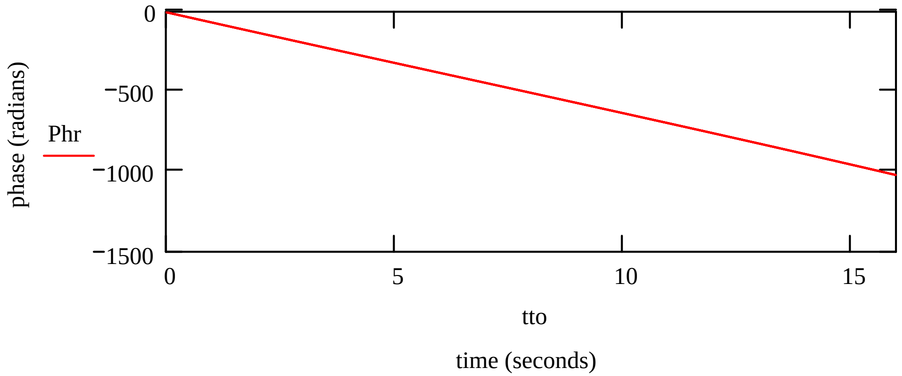
De-wrapping function

$$\mathrm{DeWrap}(ph) := \left| \begin{array}{l} np \leftarrow \mathrm{length}(ph) \\ dph_0 \leftarrow 0 \\ \text{for } jj \in 1 .. np-1 \\ \quad dph_{jj} \leftarrow \mathrm{if}\left(\left|ph_{jj} - ph_{jj-1}\right| < \pi, 0, \mathrm{sign}\left(ph_{jj} - ph_{jj-1}\right)\right) \\ qph_0 \leftarrow 0 \\ \text{for } jj \in 1 .. np-1 \\ \quad qph_{jj} \leftarrow qph_{jj-1} + dph_{jj} \\ \text{for } jj \in 0 .. np-1 \\ \quad phc_{jj} \leftarrow ph_{jj} - 2\pi \cdot qph_{jj} \\ \text{return } phc \end{array} \right.$$

$$Phr := \mathrm{DeWrap}(Ph)$$

is a de-wrapped phase

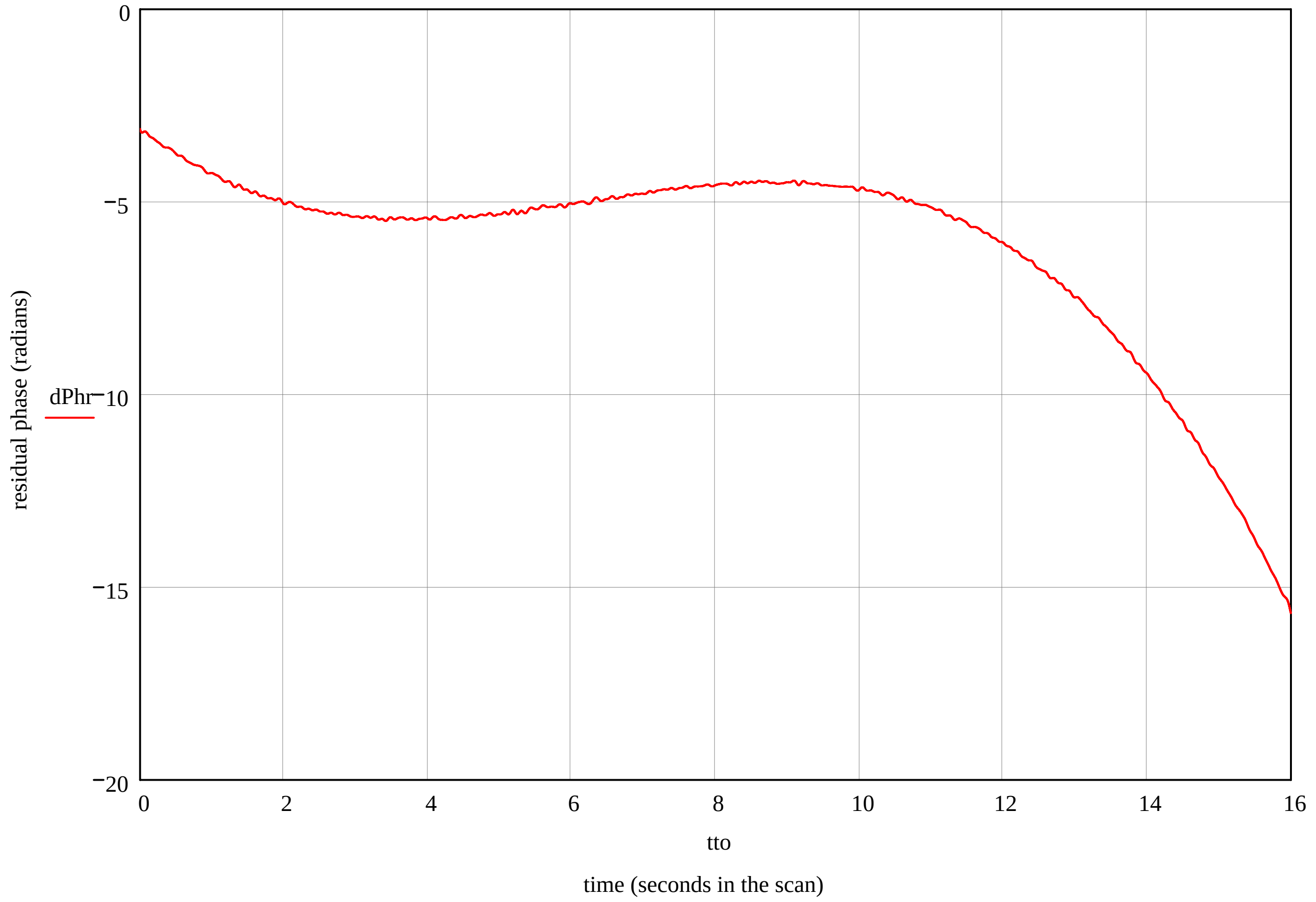
This one looks like a dull straight line



Note that de-wrapping works better for a significantly oversampled signal, like it this case it is 25 times oversampled

It looks dull because it has a huge offset of central frequency of the line

remove the trend line $\quad dPhr := Phr + 2\pi \cdot Ftarg \cdot tto$



To check the quality, remove the dPhr from the signal

$$sfcc_{jto} := sfc_{jto} \cdot \exp\left(-i \cdot dPhr_{jto}\right)$$

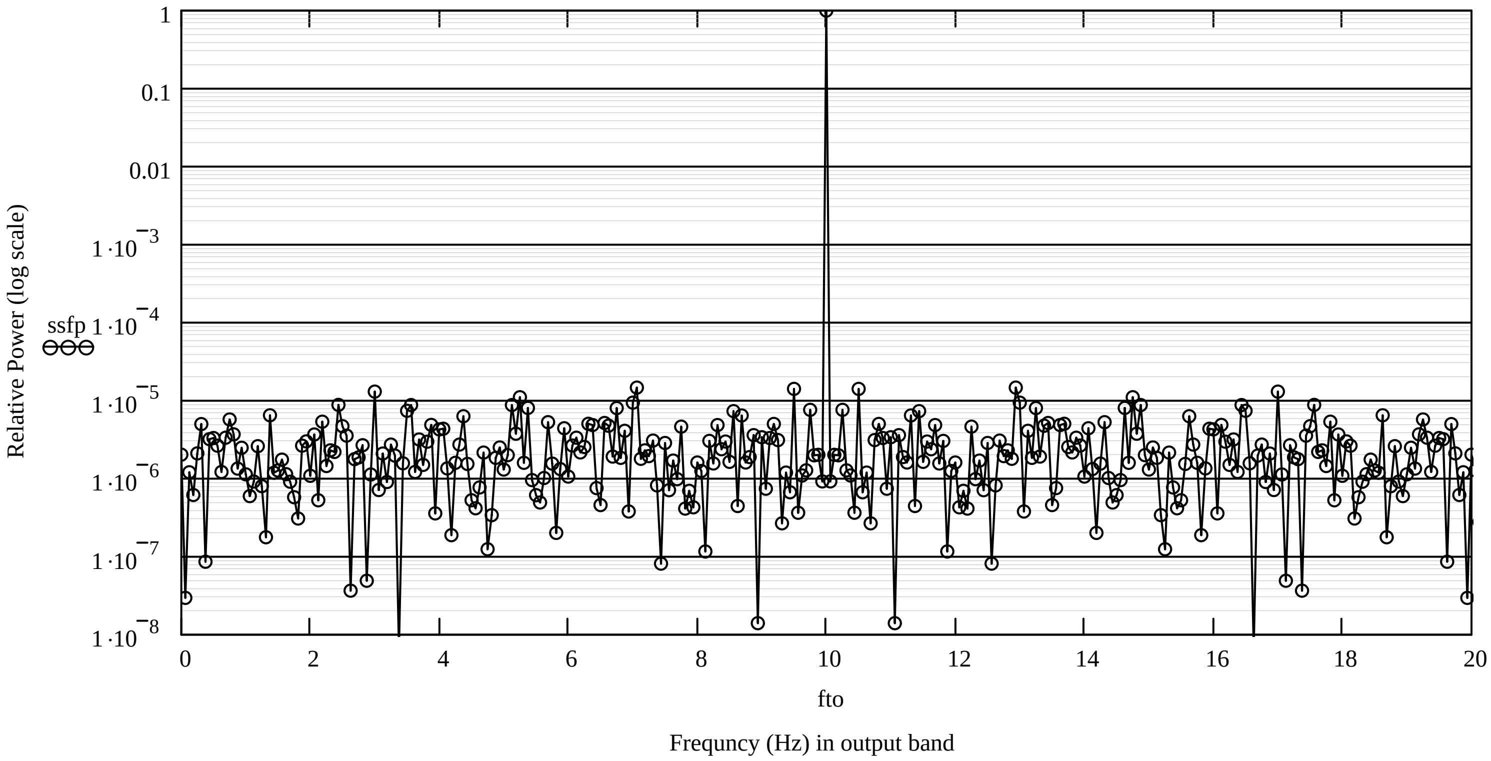
Check the spectrum

$$ssf := cfft(sfcc) \qquad ssfp := \overrightarrow{\left(|ssf|\right)^2} \qquad xssfp := \max(ssfp) \qquad ssfp := ssfp \cdot xssfp^{-1}$$

$$rmsf := GetRMS(ssfp, fto, Ftarg, 0.4 \cdot BWn, 0.1 \cdot BWn) \qquad SNR := \frac{1 - rmsf_0}{rmsf_1} \qquad SNR = 346071.6452928409 \qquad dbSNR := 10 \cdot \log(SNR, 10)$$

$dbSNR = 55.39166017678077$ achieved, in a good agreement with a predicted value of 53 dB, **even better!** but there is a rational explanation for that

Residual spectrum is squeezed into a single bit at an ultimate frequency resolution (full scan long single FFT)



Resolution $dfto = 0.0625\ s^{-1}$

Also note, that noise in this spectrum is simmetrical with respect to the line, becouse the detected phase absorbed the "phase noise" of noise.

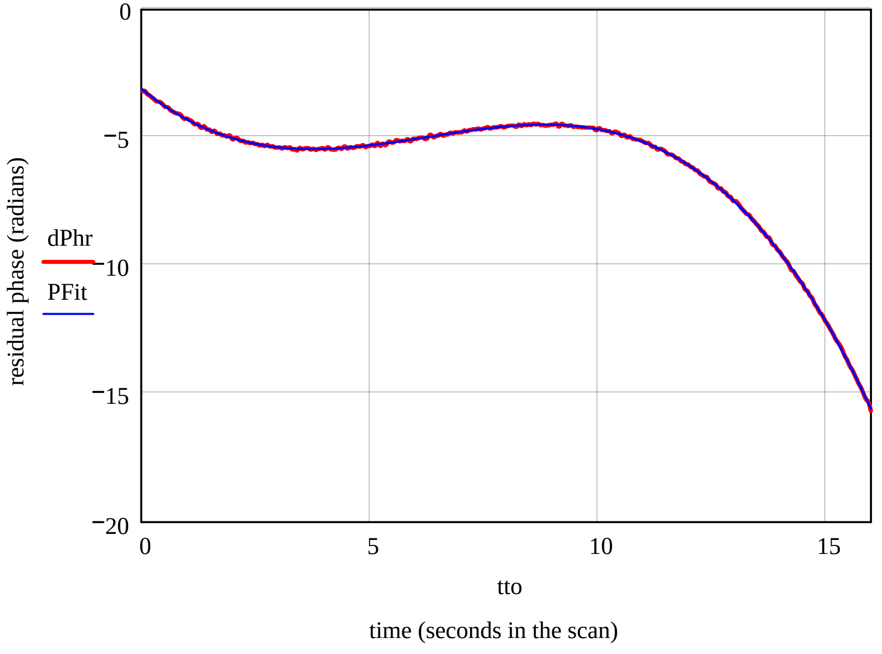
This spectrum is not a result, it's just illustration that phase was tracked and part of the noise was absorbed by the created phase model

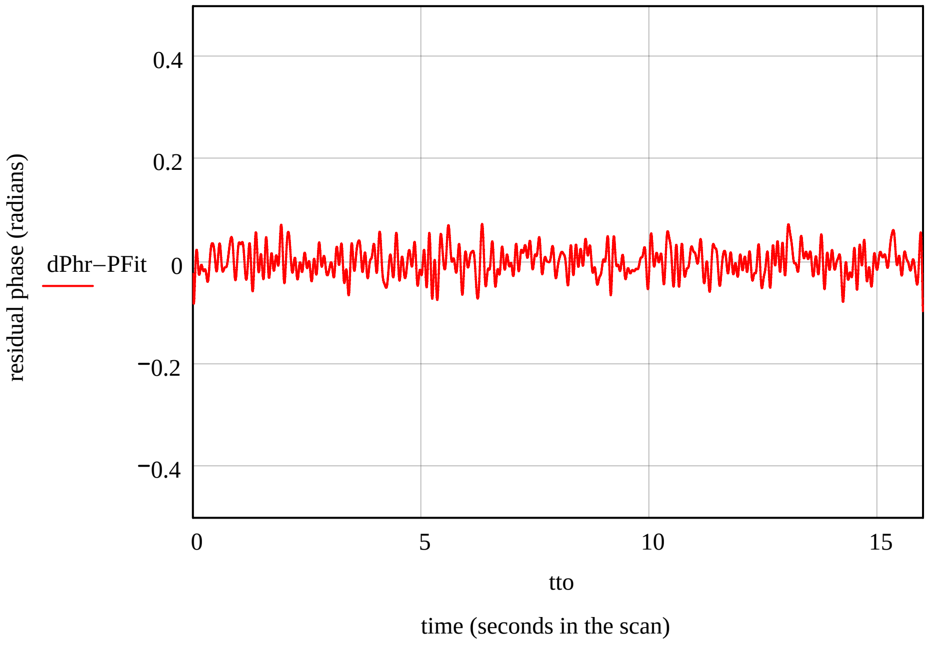
Make a "noise-free" residual phase:

$wto_{jto} := 1 \qquad Nppf := 4 \qquad PFit := PolyfitW(tto, dPhr, wto, Nppf)$

difference between residual phase and fit

$stdev(dPhr - PFit) = 0.026789318367263$





4th order polynomial fit over 16s interval effectively is a coherent integration over 4 s, so it's almost noise free remove the PFit from the signal

$sfcc_{jto} := sfc_{jto} \cdot \exp\left(-i \cdot PFit_{jto}\right)$

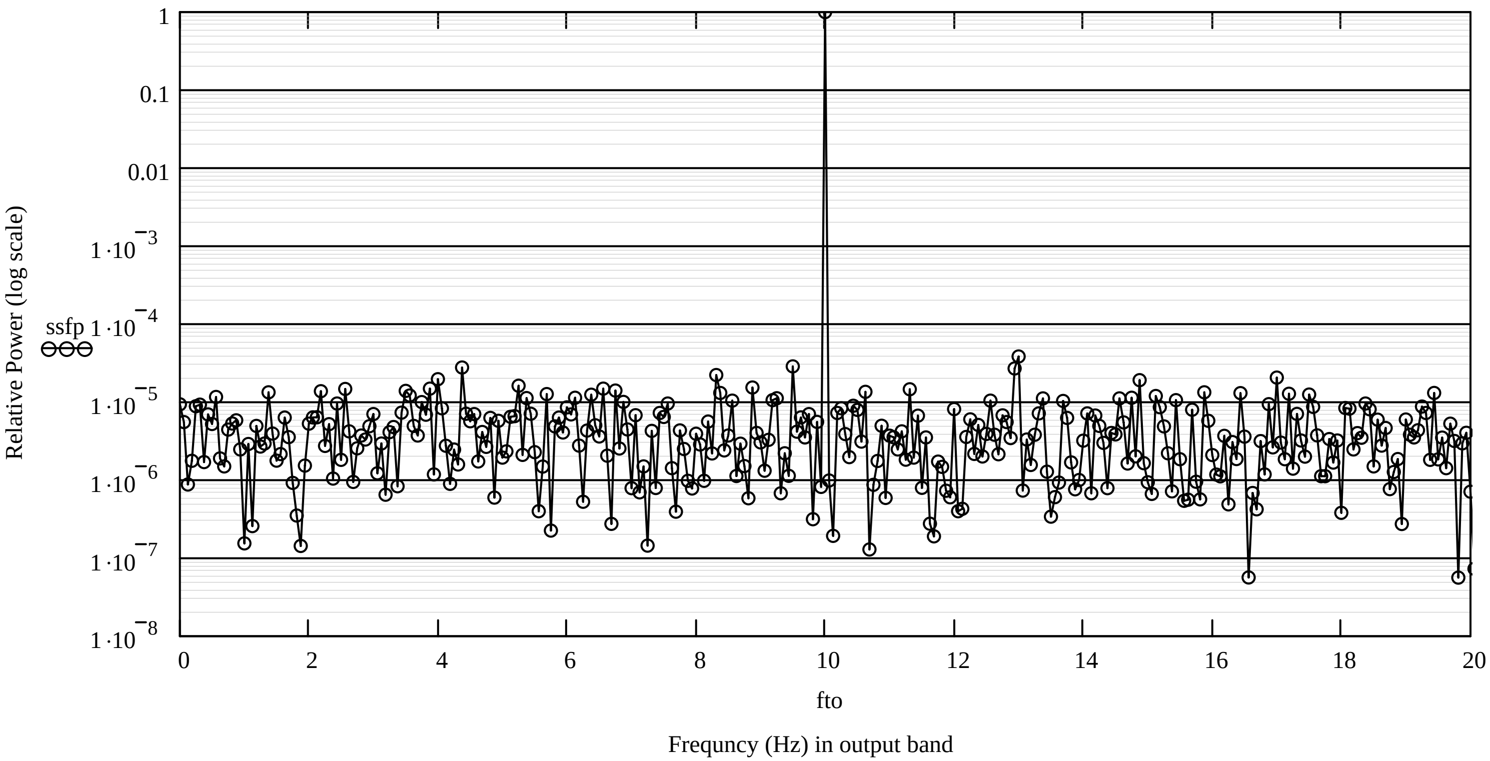
Check the spectrum

$ssf := cfft(sfcc)$ $\quad ssfp := \overrightarrow{\left(\left|ssf\right|\right)^2}$ $\quad xssfp := \max(ssfp)$ $\quad ssfp := ssfp \cdot xssfp^{-1}$

$rmsf := GetRMS(ssfp, fto, Ftarg, 0.4 \cdot BWn, 0.1 \cdot BWn)$ $\quad SNR := \dfrac{1 - rmsf_0}{rmsf_1}$ $\quad SNR = 183610.3433839685$ $\quad dbSNR := 10 \cdot \log(SNR, 10)$

$dbSNR = 52.63897142813222$ achieved, in a good agreement with a predicted value of 53 dB, now it's right !

Residual spectrum is squeezed into a single bit at an ultimate frequency resolution (full scan long single FFT)



Resolution $\quad dfto = 0.0625\ s^{-1}$

Also note, that noise in this spectrum is not simmetrical any more.

dPhr is **the result, the final product**, residual phase after other detected phases were subtracted.

Full "SKY" phase can be reconstructed by adding all subtracted phases, just keep a record on what was subtracted.

So, read the input phase model

$$Phid := READPRN("Data3.NoSpin.NoPhNoise.PhaseModel.txt")$$

$$tti := Phid^{\langle 0 \rangle} \qquad Phm := Phid^{\langle 2 \rangle}$$

Spline the input model into the output time grid

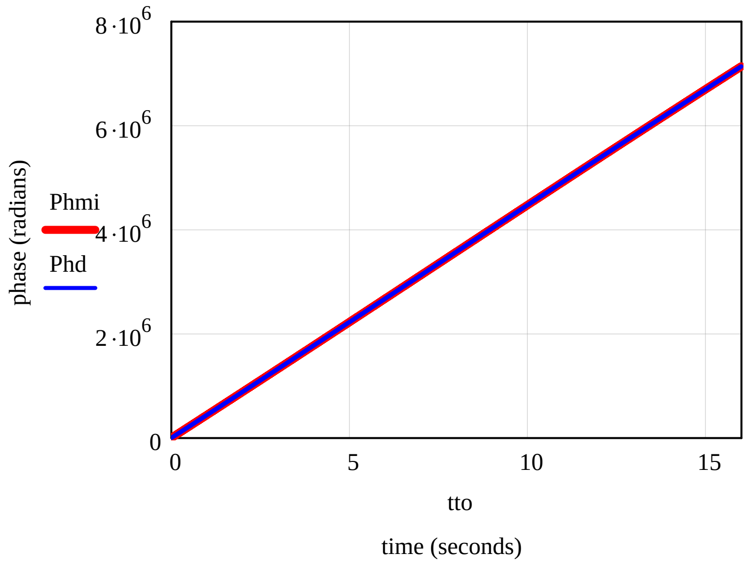
$$Phmi := Spline\left(tti, Phm, tto \cdot s^{-1}\right)$$

Reconstruct the full detected phase in video band
by adding the terms which were removed

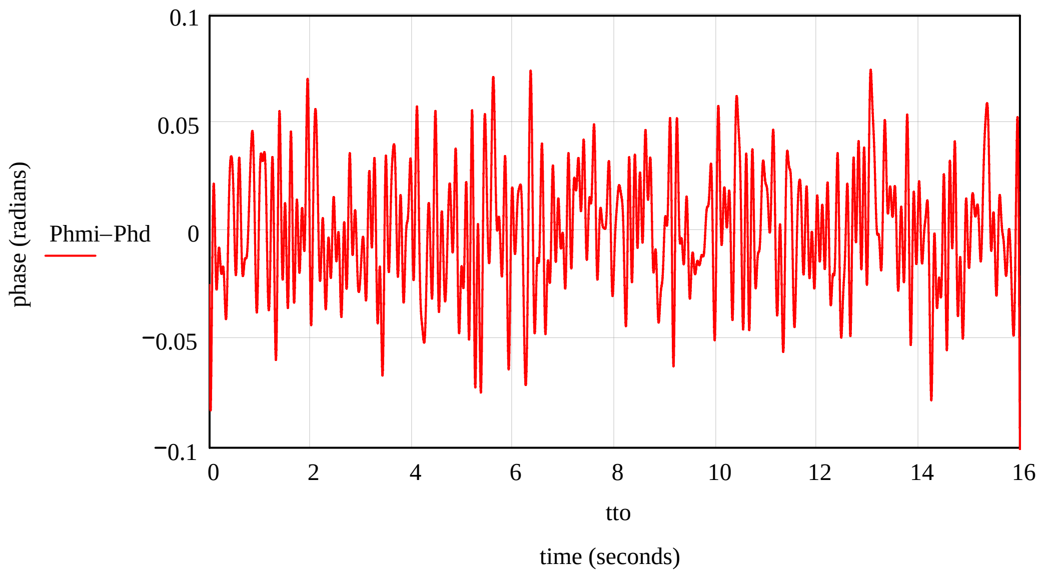
$$Phd_{jto} := -dPhr_{jto} + 2 \cdot \pi \cdot Ftarg \cdot tto_{jto} + 2\pi \cdot Frot \cdot tto_{jto} + 2\pi \cdot Fstartc \cdot tto_{jto} + Tspanp \cdot \left[\sum_{jjp=2}^{Npf} \left[Cpp_{jjp} \cdot \left(\frac{tto_{jto}}{Tspanps}\right)^{jjp}\right]\right]$$

residual ------------------- constant frequency terms ------------------------ polynomial from frequency lock

Both input and detected phases



Diffrence between the input phase model and detected one



Phase noise $stdev(Phmi - Phd) = 0.026854373491077$ in $BWn = 20\ s^{-1}$ wide band

bias $mean(Phmi - Phd) = 0.000254676110411$