Step 2a - 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"

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 $Ovlp := 2$

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

Filter ratio $FR := 100$

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

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

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

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

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

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

Derived parameters (some of then 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 Time grid $jto := 0 .. Nto - 1$ $tto_{jto} := jto \cdot dt \cdot FR$

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

shift (in samples) between overlapping segments $\quad Oshifti := \frac{Npsi}{Ovlp} \qquad Oshifto := \frac{Npso}{Ovlp}$

Frequency resolution of the input FFT $\quad dfi := (Npsi \cdot dt)^{-1} \qquad dfi = 50\ s^{-1}$

Read Doppler Frequency polynomials

$Fcd0 := READPRN(DpcFileName0)$

$Fcd1 := READPRN(DpcFileName1)$

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

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

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

Combine all data sets $Cf := Cf0 + Cf1$ $Cf_0 := Cf1_0$ 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 $Cpp_0 := 0$ $Cpp_{jpf+1} := 2\pi \cdot \frac{Cf_{jpf}}{jpf + 1}$ $Npp := Npf + 1$

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: $BWoh := BWo \cdot 0.5$

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

$$Phdopp_{jt} := Cpp_0 + Tspanp \cdot \sum_{jjp=2}^{Npf} \left[Cpp_{jjp} \cdot \left(\frac{tt_{jt}}{Tspanps} \right)^{jjp} \right]$$

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

$Fstartc \cdot Oshifti \cdot dt = 687.5$	$Pssc := Fstartc \cdot Oshifti \cdot dt - floor(Fstartc \cdot Oshifti \cdot dt)$	$Esc := exp(i \cdot 2 \cdot \pi \cdot Pssc)$	$Esc = -1$
$Fstart1 \cdot Oshifti \cdot dt = 587.5$	$Pss1 := Fstart1 \cdot Oshifti \cdot dt - floor(Fstart1 \cdot Oshifti \cdot dt)$	$Es1 := exp(i \cdot 2 \cdot \pi \cdot Pss1)$	$Es1 = -1$
$Fstart2 \cdot Oshifti \cdot dt = 187.5$	$Pss2 := Fstart2 \cdot Oshifti \cdot dt - floor(Fstart2 \cdot Oshifti \cdot dt)$	$Es2 := exp(i \cdot 2 \cdot \pi \cdot Pss2)$	$Es2 = -1$
$Fstart3 \cdot Oshifti \cdot dt = 887.5$	$Pss3 := Fstart3 \cdot Oshifti \cdot dt - floor(Fstart3 \cdot Oshifti \cdot dt)$	$Es3 := exp(i \cdot 2 \cdot \pi \cdot Pss3)$	$Es3 = -1$

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 ← vec(din · Wini)
    phc ← submatrix(Phcorr, skip, skip + Npsi − 1, 0, 0)
    ephc ← vec(exp(i · phc))
    din ← vec(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 ← vec(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.

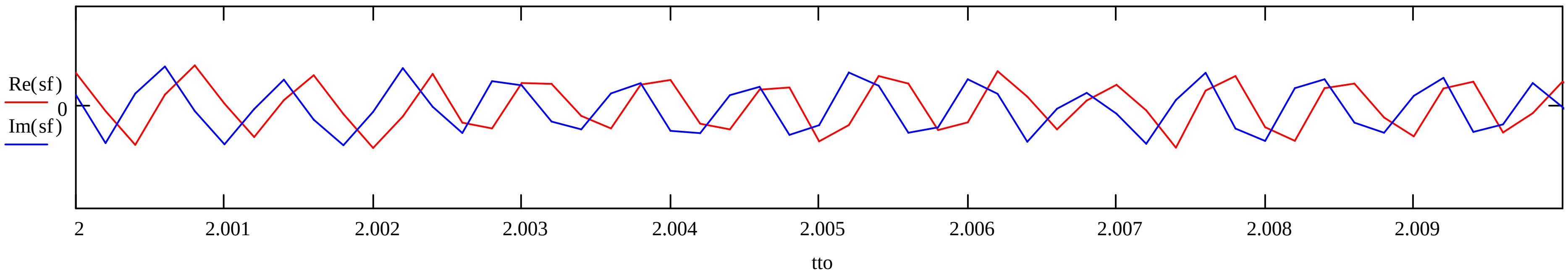
And it's faster !

This is a tricky part to make in C

Make a filtered signal for major tone

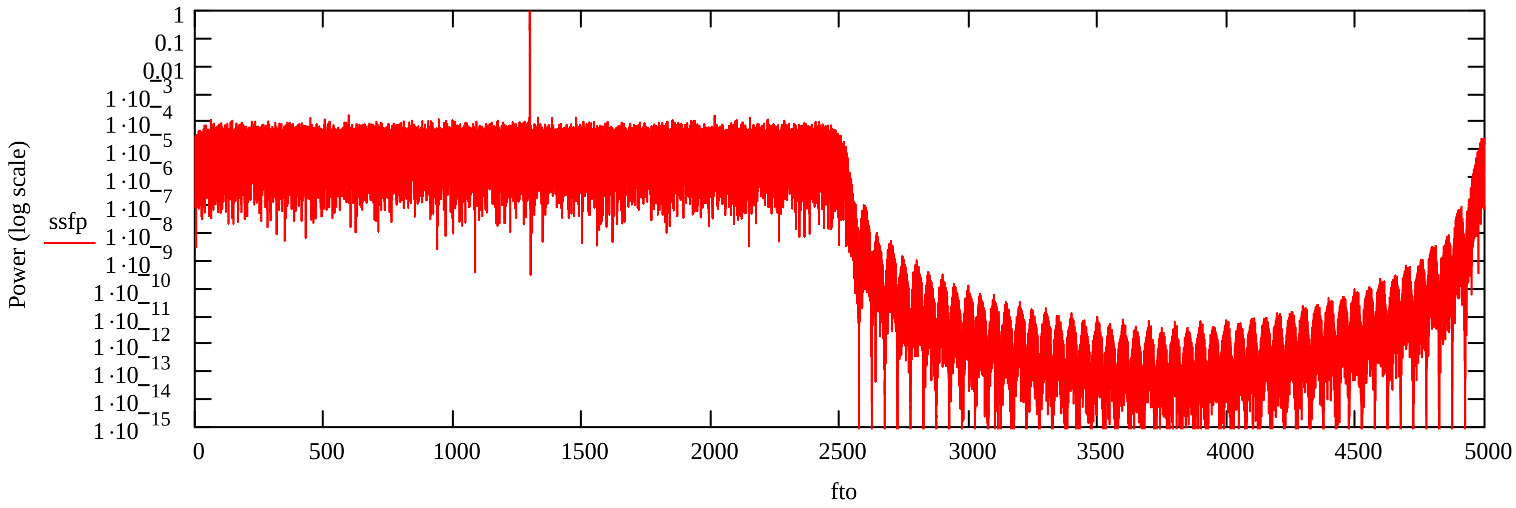
$sf := \mathrm{MakeFiltX}(Phdopp, Bsc, Bec, Esc)$ $\mathrm{length}(sf) = 80000$

fragment of the filtered complex signal in a time domain



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

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



Frequncy (Hz) in 2 kHz output band

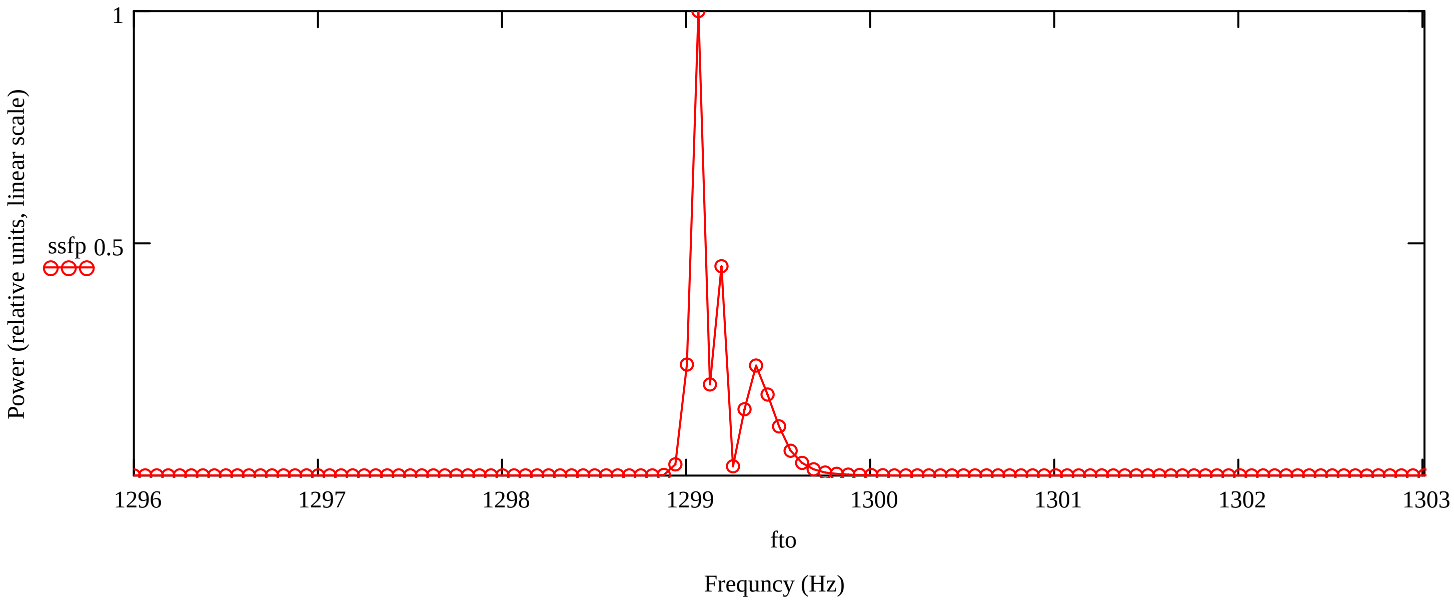
To make SNR estimation, extract the noise data in range 200 - 1000 Hz

$spnoise := \mathrm{submatrix}\left(ssfp, 200 \cdot Hz \cdot dfto^{-1}, 1100 \cdot Hz \cdot dfto^{-1}, 0, 0\right)$ $\mathrm{length}(spnoise) = 14401$ $\mathrm{stdev}(spnoise) = 0.000013676033142$ $SNR := \mathrm{stdev}(spnoise)^{-1}$

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

And here we have: $dbSNR := 10 \cdot \log(SNR, 10)$ $dbSNR = 48.64039855427141$

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, 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)

Determine the frequency of the max power

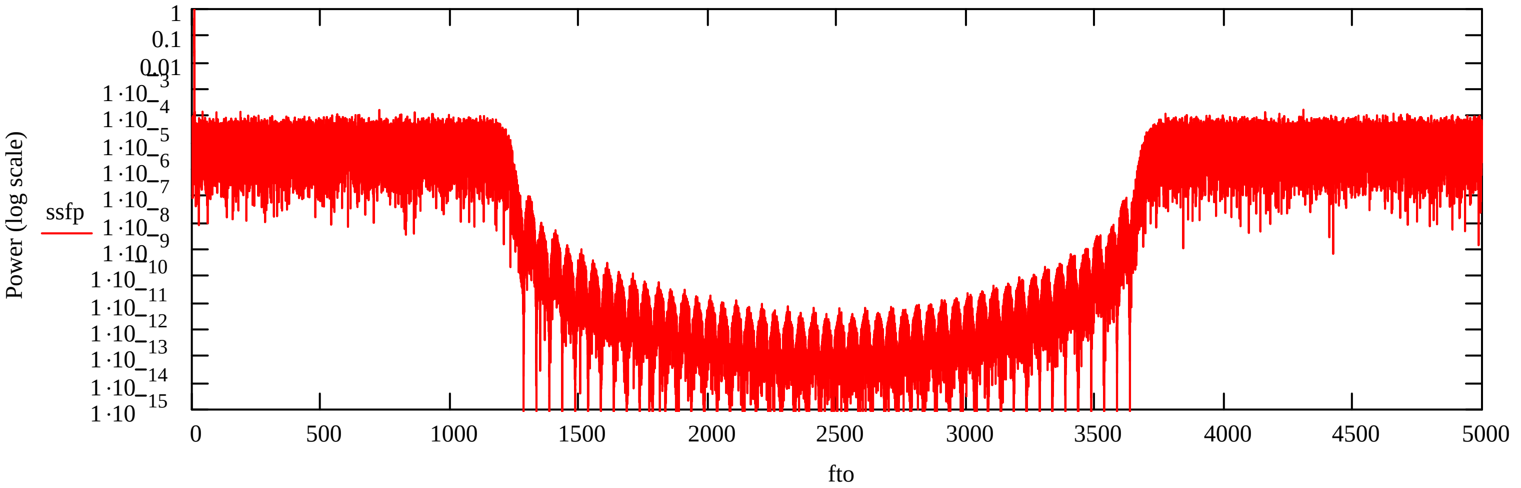
$xf := \text{FindMax}(ssfp, fto, 0 \cdot Hz, 2500 \cdot Hz)$ $\quad fmax := xf_1 \cdot dfto$ $\quad fmax = 1299.0616432101856\ s^{-1}$

Move the line to 10 Hz bin:

$sfc_{jto} := sf_{jto} \cdot \exp\left[2\pi \cdot i \cdot (fmax - 10Hz) \cdot tto_{jto}\right]$

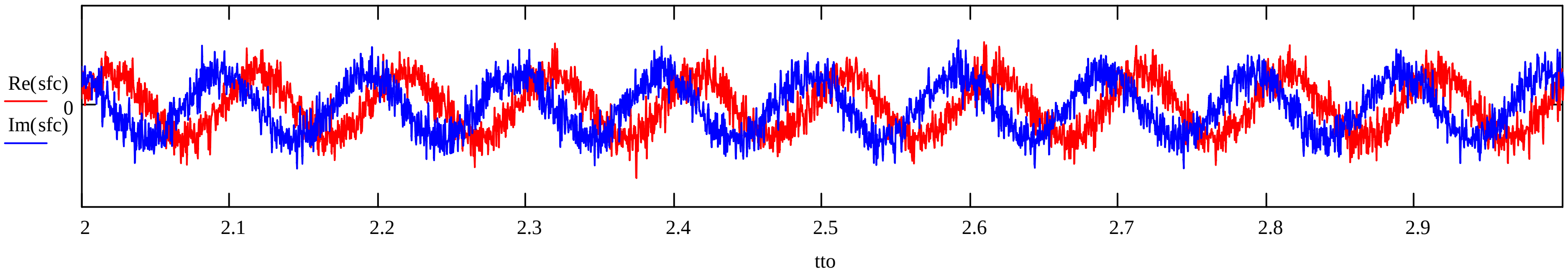
Check the spectrum $\quad ssf := \text{cfft}(sfc)$ $\quad ssfp := \overrightarrow{\left(|ssf|\right)^2}$ $\quad xssfp := \max(ssfp)$ $\quad ssfp := ssfp \cdot xssfp^{-1}$ $\quad dfto := Tspan^{-1}$ $\quad fto_{jto} := dfto \cdot jto$

Line is close to the DC edge of the band now, 10 Hz



Frequncy (Hz) in 2 kHz output band

Check the time domain pattern, resembles a sine wave



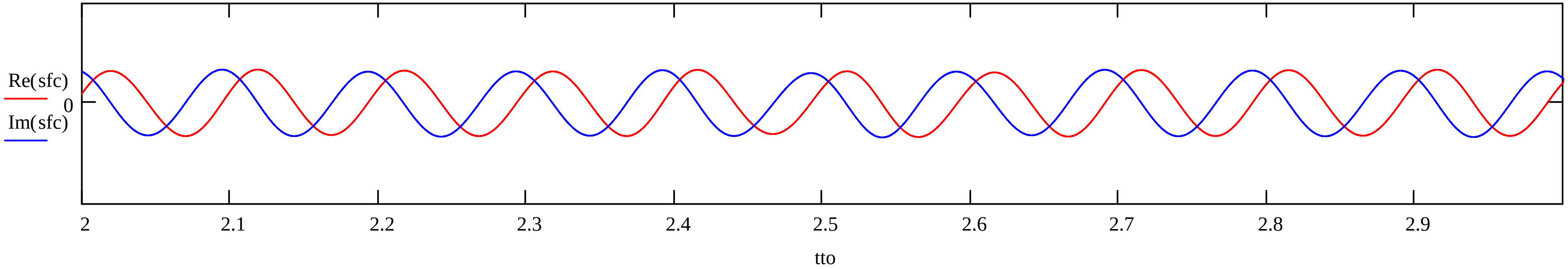
Make 20 Hz wide filter around the line

$$ssff_{jto} := \text{if}\left(\left|fto_{jto} - 10 \cdot Hz\right| < 10 \cdot Hz, ssf_{jto}, 0\right)$$

And get the signal back to time domain

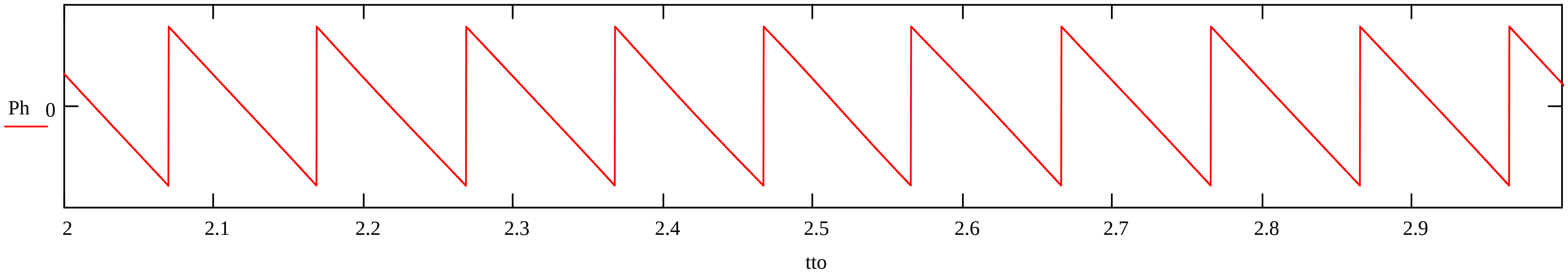
$$sfc := \text{icfft}(ssff)$$

Check the time domain pattern, even more likely the sine wave, not much of noise



Get the phase

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



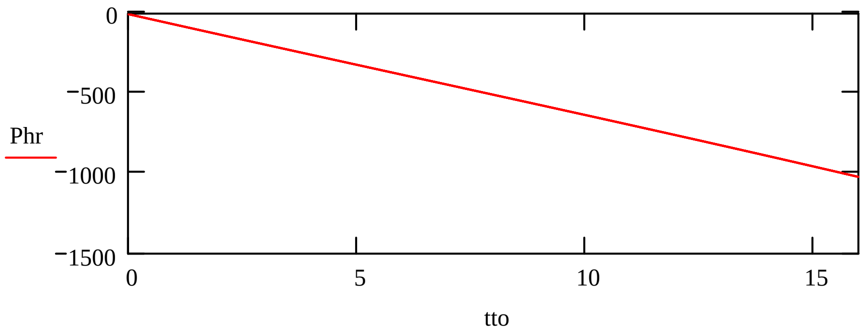
Phase is wrapping over 2π each cycle of 10 Hz

De-wrap

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

$Phr := \mathrm{DeWrap}(Ph)$ is a de-wrapped phase

This one looks like a dull straight line

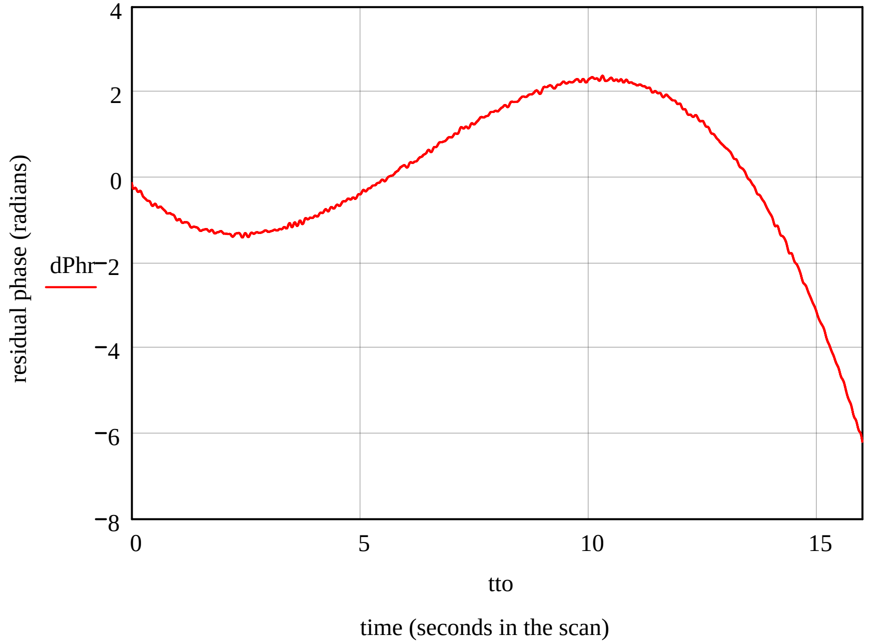


Make Line Fit

$$Lph := \mathrm{line}\left(tto \cdot s^{-1}, Phr\right) \qquad LPhr := Lph_0 + Lph_1 \cdot tto \cdot s^{-1}$$

$dPhr := Phr - LPhr$ remove the trend line

This one looks familiarly similar to what we saw when comparing the detected phase at first iteration with the input phase model



To check the quality, remove the dPhr from signal

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

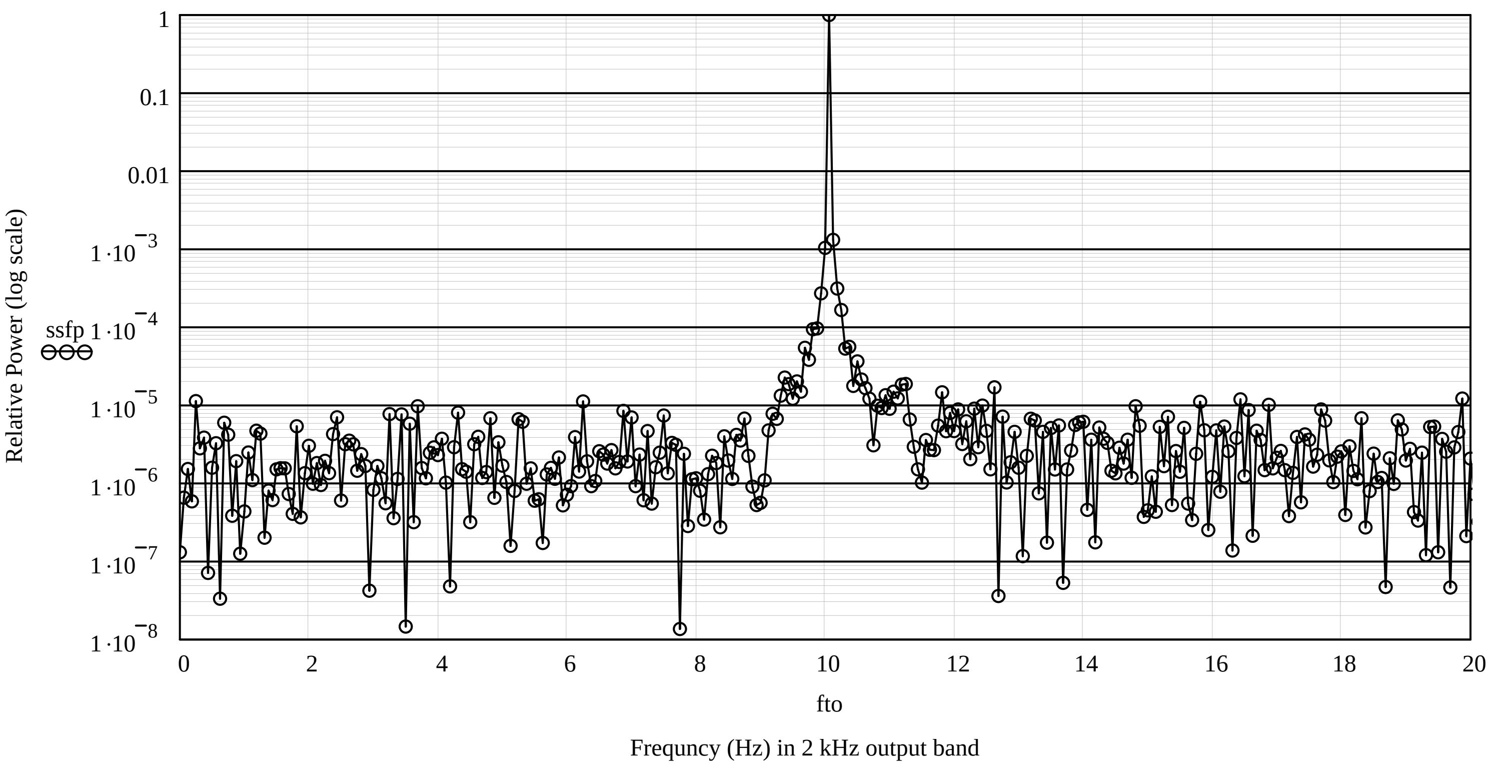
Check the spectrum

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

$rmsf := GetRMS(ssfp, fto, 10 \cdot Hz, 9 \cdot Hz, 1Hz)$ $\quad SNR := \dfrac{1 - rmsf_0}{rmsf_1}$ $\quad SNR = 310180.21816440055$ $\quad dbSNR := 10 \cdot \log(SNR, 10)$

$dbSNR = 54.916140971010925$ achieved, in a good agreement with a predicted value of 53 dB

Residual spectrum is squeezed into a single bit at an ultimate frequency resolution (full scan long single FFT) Resolution $dfto = 0.0625\ s^{-1}$



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

This phase is 2.5 ms sampled (final filter width 20 Hz)

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