Step 2 - Filter the tones

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"

Pass2 Doppler Polynomial model file name

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 FFTsegment $Npsi := 10 \cdot 1000$

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

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

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

Read Dorrpler Frequency polynomials

$Fcd0 := READPRN(DpcFileName0)$

$Fcd1 := READPRN(DpcFileName1)$

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

$Cf0_{jpf} := Fcd0_{jpf+2}$

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

Combine all data sets $Cf := Cf0 + Cf1$ $\quad Cf_0 := Cf1_0$ Select the Doppler constant offset from the last pass data

Make Phase polynomials $Cpp_0 := 0$ $\quad Cpp_{jpf+1} := 2\pi \cdot \frac{Cf_{jpf}}{jpf + 1}$ $\quad Cpp_1 := 0$ $\quad Npp := Npf + 1$

Frequency polynomial

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

Phase polynomial

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

$BWoh := BWo \cdot 0.5$

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

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

carrier $Fcc := Cf_0 \cdot Hz$ $\quad Bsc := floor\left[(Fcc - BWoh) \cdot dfi^{-1}\right]$ $\quad Bsc = 1375$ $\quad Bec := Bsc + Npfo - 1$ $\quad Fstartc := Bsc \cdot dfi$ $\quad Fstartc = 68750\ s^{-1}$

tones $Fc1 := Fcc - 10000 \cdot Hz$ $\quad Bs1 := floor\left[(Fc1 - BWoh) \cdot dfi^{-1}\right]$ $\quad Bs1 = 1175$ $\quad Be1 := Bs1 + Npfo - 1$ $\quad Fstart1 := Bs1 \cdot dfi$ $\quad Fstart1 = 58750\ s^{-1}$

$Fc2 := Fcc - 50000 \cdot Hz$ $\quad Bs2 := floor\left[(Fc2 - BWoh) \cdot dfi^{-1}\right]$ $\quad Bs2 = 375$ $\quad Be2 := Bs2 + Npfo - 1$ $\quad Fstart2 := Bs2 \cdot dfi$ $\quad Fstart2 = 18750\ s^{-1}$

$Fc3 := Fcc + 20000 \cdot Hz$ $\quad Bs3 := floor\left[(Fc3 - BWoh) \cdot dfi^{-1}\right]$ $\quad Bs3 = 1775$ $\quad Be3 := Bs3 + Npfo - 1$ $\quad Fstart3 := Bs3 \cdot dfi$ $\quad Fstart3 = 88750\ s^{-1}$

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

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

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

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

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

Major function to do Phase Tracking, 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 parallell, using only 1 "big" input Fourier transform and several "small" output FFTs.

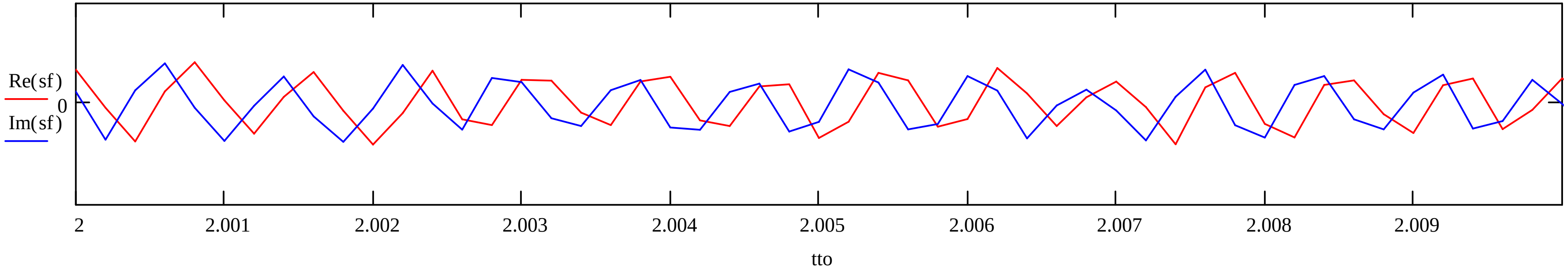
Mathematically this filter is equivalent to PFB, but better, because it allows arbitrary positioning of output channels, with different width.

This is a tricky part to make in C

Make a filtered signal for major tone

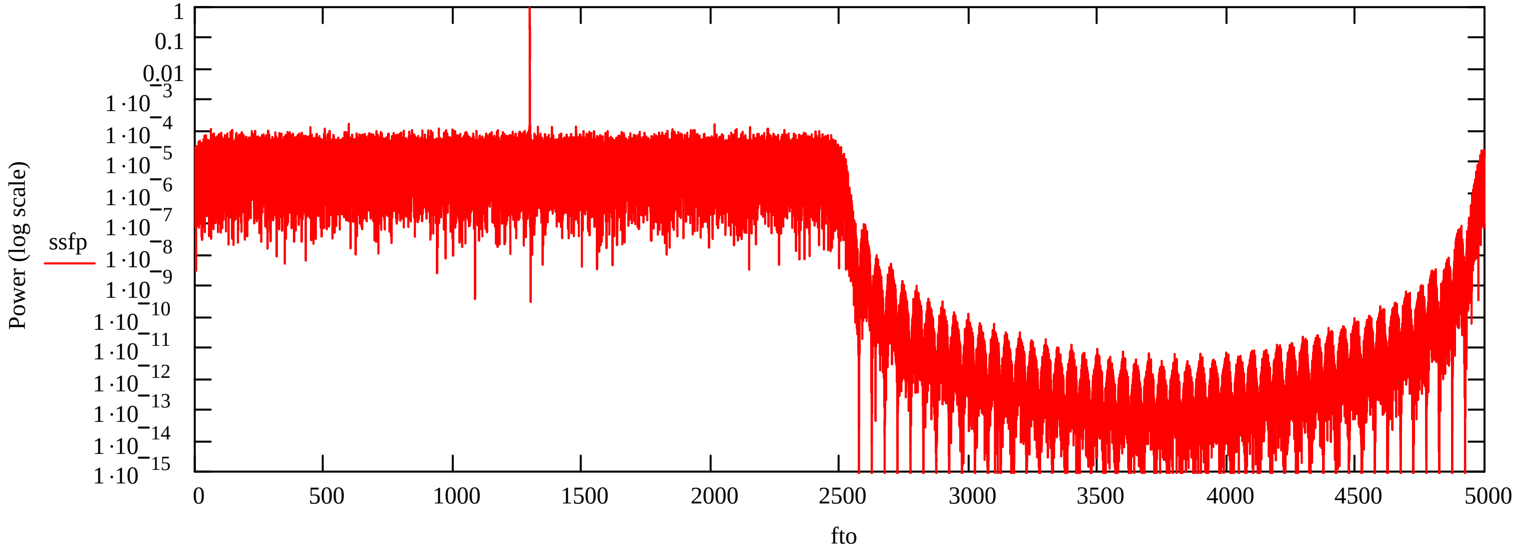
$sf := \text{MakeFiltX}(Phdopp, Bsc, Bec, Esc)$ $\qquad$ $\text{length}(sf) = 80000$

fragment of the filtered complex signal in a time domain



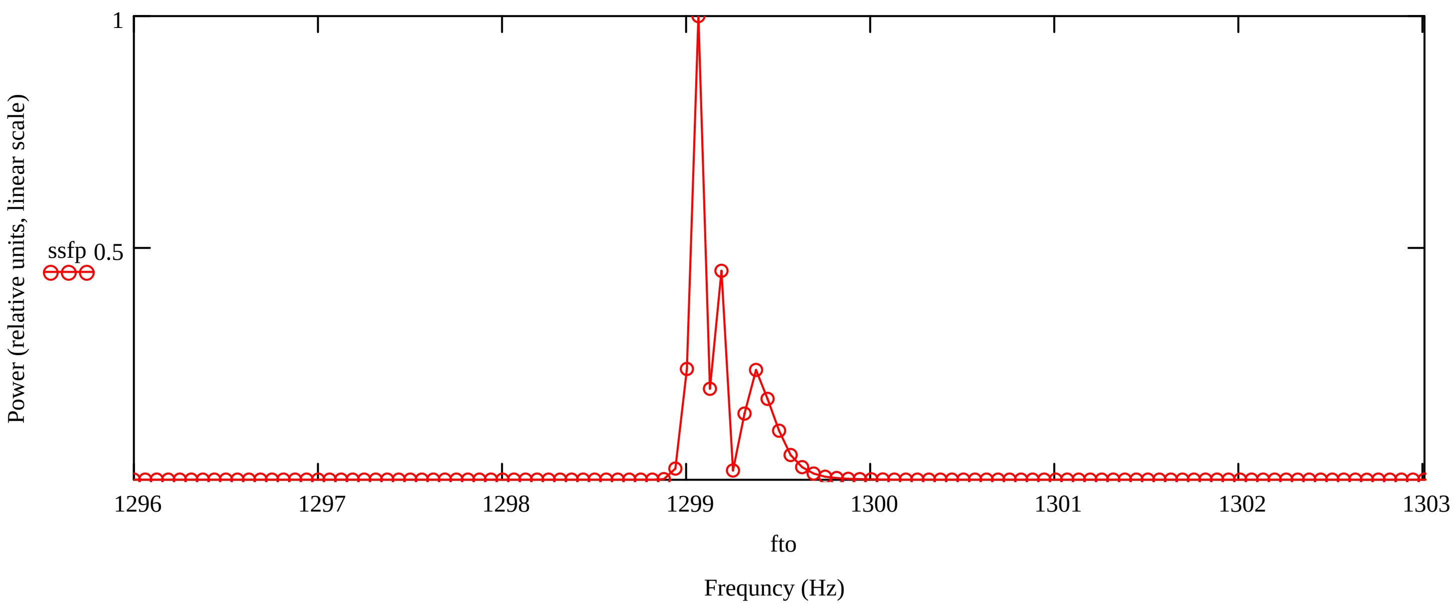
Full length spectrum (two-sided FFT), shows good suppression of negative frequncies

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



Frequncy (Hz) in 2 kHz output band

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



So, just write this narrow band complex signal into file, PLL it later.