TÉCNICO LISBOA

Instituto Superior Técnico

Digital Signal Processing Systems (SPDSina)

(LEEC/MBEiom)

3rd Quarter 2022/2023

Bandpass Tracking Filter

Version 3.0, revised May 2023

Prof. Gonçalo Tavares

Credits: The laboratory work was initially developed by Profs. Moisés Piedade and João Vital. Afterwards it was subject to some modifications by Profs, Moisés Piedade, Teresa Almeida, Jorge Martins and Gonçalo Tavares. The current version by Prof. Gonçalo Tavares has undergone significant changes to accommodate the latest's techniques in DSP techniques. to port the original work (written in assembly language) to the latest signal processors and DSP hardware platforms from Texas Instruments, which use C language and finally to translate the updated guide to English.

Introduction and working principle

The main goal of this laboratory project is to provide the students with a hands-on hardware application of many of the theoretical material taught in the theoretical-practical classes of this course. Although all the work required could be done by means of computer simulation with programs such as Matlab or Octave (and some of it will be) it is very important to use the laboratory equipment and the DSP boards to generate and verify the expected results.

The objective of this work is the development of a pass-band tracking filter, which automatically adjusts its center frequency to the frequency of the most significant component in the input signal. This type of system is important in many applications, e. g., satellite communications where the carried frequency changes due to the relative movement between the satellite and the Earth (Doppler effect). Another example is for instance in systems which try to cancel the noise produced by the motor inside a vehicle. This type of noise usually has periodic components with a frequency which changes with the working regime of the motor and need to be tracked. A general (simplified) tracking filter is represented in the diagram in Figure 1:

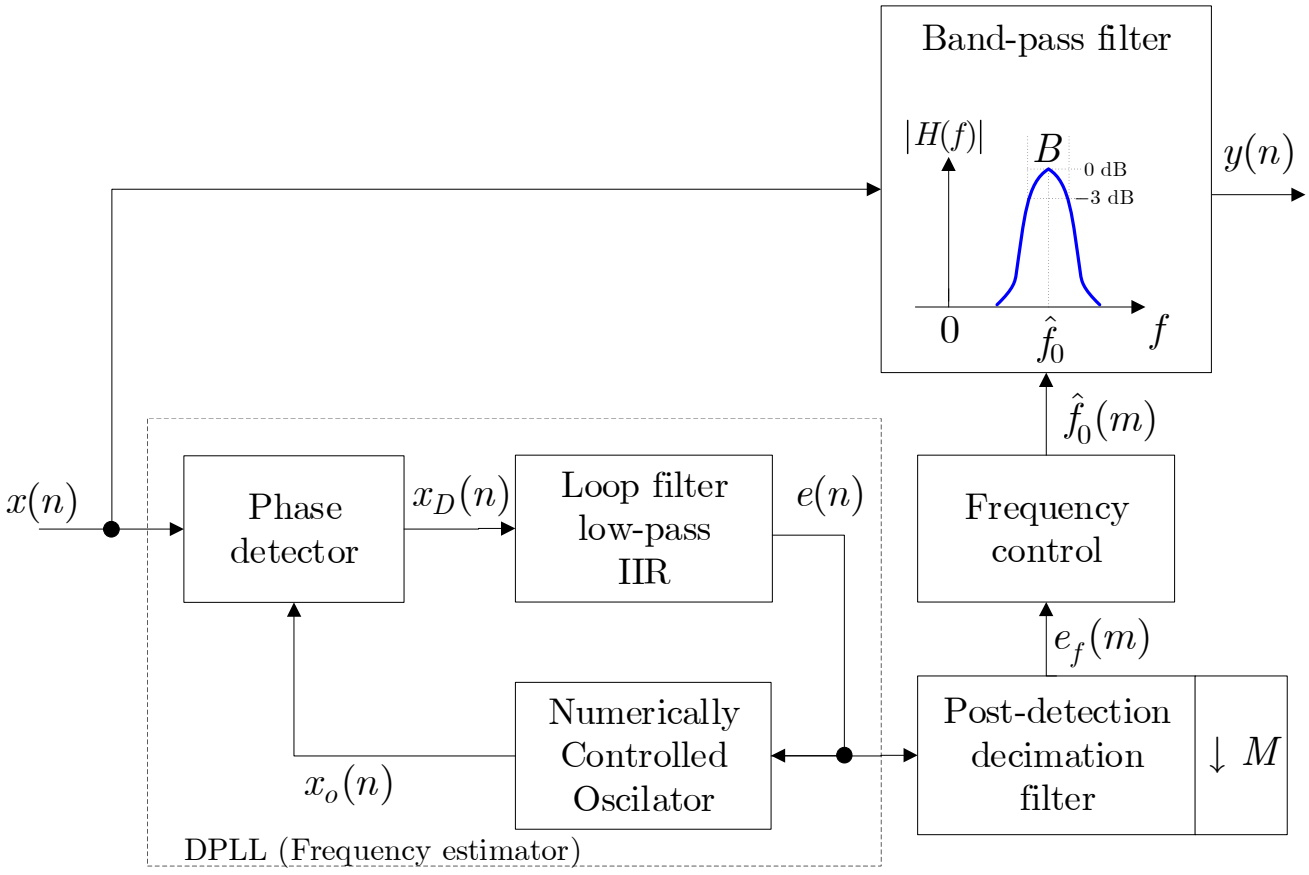

Figure 1: Block diagram of a general band-pass tracking filter.

As you can see this may be viewed as an *adaptive* system which continuously (tries to) track the frequency of the incoming signal and sets the central filter frequency accordingly. This type of system is designed and will only work with narrowband signals, i.e., signals with its energy concentrated in a narrow bandwidth (such as a sinusoid) which are in general contaminated by additive noise, which is to be mitigated by the tracking filter. Looking at Figure 1 we see that signals have two different time variables. The variable n is indexed to the system sampling rate $f_s = 1/T_s$ so these signals have been sampled and/or are processed at every $t_n = nT_s$ second. On the other hand, the variable m is indexed to the lower sampling rate $f_s^{'} = f_s/M$ where M is an integer. So, these signals are processed M times slower, at every $t_m = mMT_s$. We will see that this operation (lowering the sampling frequency within the program) is called a *decimation* and why it is important. Looking again at Figure 1 it is possible to identify a number of different blocks: A Digital Phase Locked Loop (DPLL) which works as the frequency estimator, producing a near-DC signal $e(n)$, a post detection Cascade-Integrator-Comb (CIC) lowpass filter which filters $e(n)$ and decimates if by a factor M producing $e_f(m)$. Note that for every set of M samples of $e(n)$, only **one** sample of $e_f(m)$ is produced. This is required to filter $e(n)$ adequately and to avoid programming the filter central frequency at a very high rate (at f_s/M instead of f_s) which might lead to an unstable system behavior. The sample $e_f(m)$ is then used to program the frequency of the bandpass filter in the "frequency control" block which produces the control sample $\hat{f}_o(m)$. A detailed description of each block is given in the following sections.

Digital Phase Locked Loop (DPLL)

This block is responsible for estimating the most significant frequency present in the input signal. It is composed by a phase detector, a loop filter, and a numerically controlled (NCO) oscillator. Simply put, the phase detector computes the phase **difference** between the input signal and the NCO signal. This difference is low-pass filtered to remove undesired high-frequencies components resulting from phase detector operation and is fed to the NCO. In some circumstances and due to the negative feedback, the DPLL will acquire synchronization meaning that the frequency produced by the NCO is, on average, equal to the input frequency. When this happens both signals are **frequency-locked** and $e(n)$ can be used as an estimate of the input frequency.

Phase detector

The phase detector to be used is a simple multiplier which multiplies the sample of the input signal $x(n) = A_1 \sin\left(2\pi \frac{f_i}{f_s} n + \phi\right)$ by the sample generated on the previously developed NCO $x_o(n) = A_2 \cos\left(2\pi \frac{f_o}{f_s} n + \hat{\phi}\right)$. Its static characteristic, that is the mean output value (as would be obtained by a perfect integrator) as a function of the signal's phase difference is $\left\langle x_D(\hat{\phi} - \phi) \right\rangle = \frac{A_1 A_2}{2} \sin(\hat{\phi} - \phi)$. This characteristic is represented in Figure 2 and has a sinusoidal shape. For best performance it should be linear in the working range $-\frac{\pi}{2} < \hat{\phi} - \phi < \frac{\pi}{2}$ (positive phase detector gain) or in $\frac{\pi}{2} < \hat{\phi} - \phi < \frac{3\pi}{2}$ (negative phase detector gain). Nevertheless for $\left|\hat{\phi} - \phi\right| \approx 0$ or $\left|\hat{\phi} - \phi\right| \approx \pi$, the characteristic is close to linear, so these should be identified as sweep-spot values for use in the system. Later we will learn how to overcome this nonlinear issue.

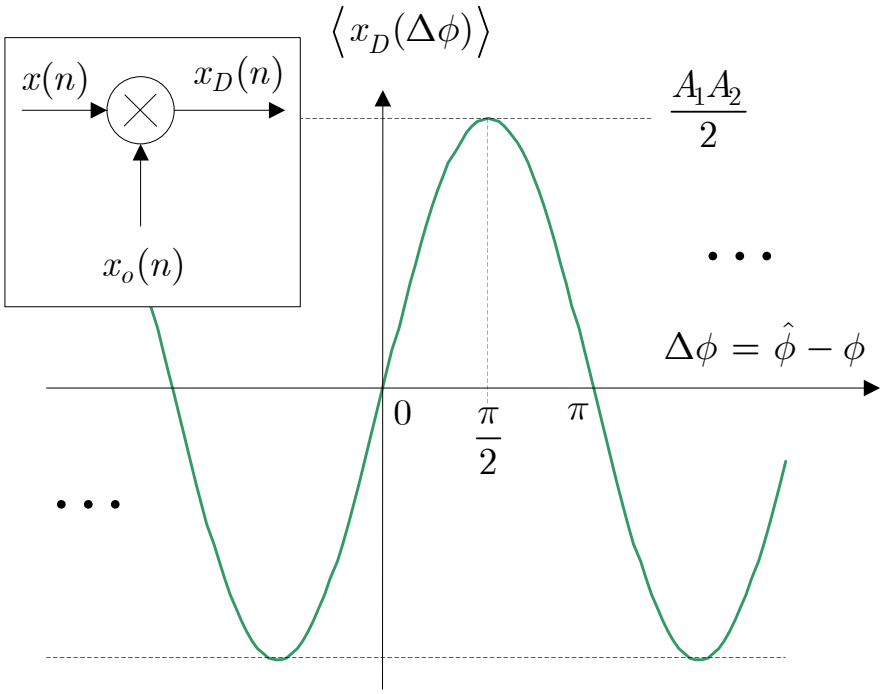

Figure 2: Static characteristic of the sinusoidal phase detector.

(T) Compute the phase detector gain $K_D = \frac{\left\langle x_D(\Delta\phi) \right\rangle}{\partial \Delta\phi}$ (number/rad) and verify it is not linear.

Loop filter

In our first attempt to have an operational system, the loop filter is a simple low-pass first order filter obeying the difference equation.

$$e(n) = \alpha e(n-1) + (1-\alpha)x_D(n)\,. \tag{1}$$

(T) Based on the difference equation (1):

1. Determine the digital filter frequency response $H(z) = \dfrac{E(z)}{X_D(z)}$ where $E(z)$ and $X_D(z)$ are the Z transforms of output $e(n)$ and input $x_D(n)$ signals respectively. Sketch the pole-zero plot of the filter.
2. Show that for the filter to be sable is necessary that $\left|\alpha\right| < 1$ and to be lowpass it is necessary that $0 < \alpha < 1$. If $-1 < \alpha < 0$ what type of filter results?
3. The filter should be designed to have a cut-off frequency of $f_c = 100$ Hz. Since $f_c \ll f_s = 16$ kHz show that the approximation $f_c \approx \dfrac{1-\alpha}{\alpha} \cdot \dfrac{f_s}{2\pi}$ holds and use it to determine the required value of α.

Numerical controlled Oscillator (NCO)

The NCO has already been developed in the first part of the project, so all is needed is to integrate its code into the tracking filter main project. It should generate a sinewave with frequency $f_{0_{\min}} \le f_o \le f_{0_{\max}}$ for a control input signal $-1 \le e(n) \le 1$ that is $f_o = f_{0l} + K_0 e(n)$ where $K_0 = \left(\dfrac{f_{0_{\max}} - f_{0_{\min}}}{2}\right)$ and $f_{0l} = \dfrac{f_{0_{\min}} + f_{0_{\max}}}{2}$ is the NCO central (also called *free-running*) frequency, which occurs when $e(n) = 0$.

Post-detection CIC filter

After the loop is synchronized, its filter output will have the correct mean value (required to drive the NCO) but will also exhibit a small "noise" component due to the sinewave multiplication by the phase detector (at $f = 2f_o$). To reduce this undesirable component, it is possible to reduce the filter bandwidth f_c but the cost is a slower DPLL which will take a longer time to synchronize and a reduction in the acquisition range. In addition, as mentioned previously there is the need to reduce the update rate by a factor of *M*. When combined,

these two operations (filtering and then reducing the sample rate) form a *decimator* as shown in Figure 1. This type of operation is known as *multirate signal processing* and will be covered in detail in the theoretical classes. In this project, the decimator will be implemented using a special multiplierless filter structure called a *cascade integrator-comb* (CIC) filter which is represented in Figure 3.

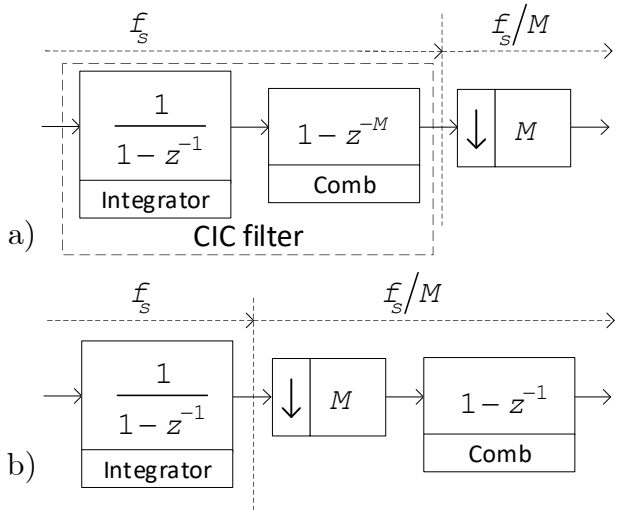

Figure 3: Decimator by M using a one-stage CIC filter: a) classical architecture; b) efficient computation architecture to be implemented.

The decimation factor is M meaning that the output sampling frequency (at which the bandpass filter central frequency is updated) is $f_s / M = 1.9531$ Hz for $M = 8192$.

(T, S) Decimator simulation and analysis:

1. Determine the frequency response (amplitude and phase) of the CIC filter and sketch it $0 \leq f \leq 5f_s / M$
2. Compute the filter attenuation at $f = 4$ Hz.
3. Write a Matlab or Octave script which implements the decimator represented in Figure 3. Test it using a sinewave as input signal, with unit amplitude and frequency $f_i = 4$ Hz. Check the signal attenuation using the result from 2.

After the loop is synchronized, its filter output will have the correct mean value (required to drive the NCO) but will also exhibit a small "noise" component due to the sinewave multiplication by the phase detector (at $f = 2f_o$). To reduce this undesirable component, it is possible to reduce the filter bandwidth f_c but the cost is a slower DPLL which will take a longer time to synchronize and a reduction in the acquisition range. In addition, as mentioned previously there is the need to reduce the update rate by a factor of M. When combined, these two operations (filtering and then reducing the sample rate) form a *decimator* as shown

in Figure 3. This type of operation is known as *multirate signal processing* and will be covered in detail in the theoretical classes.

Second order bandpass IIR filter

The passband filter is a second order infinite impulse response (IIR) filter (biquadratic section) with programmable central frequency f_0 and -3dB **bilateral** bandwidth B (in this project only f_0 will be changed in real time, the bilateral bandwidth is fixed at $B = 400$ Hz). The filter transfer function is:

$$T_{BP}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} = K_B \frac{1 - z^{-2}}{1 - 2K_f(1 - K_B) z^{-1} + (1 - 2K_B) z^{-2}}. \tag{2}$$

The amplitude and the phase characteristic of the filter frequency response $T_{BP}(f) = T_{BP}(z)\big|_{z=e^{j2\pi f T_s}} = |T_{BP}(f)| e^{j \arg\{T_{BP}(f)\}}$ is plotted in Figures 4a and 4b respectively for three different central frequencies. Note that the filter exhibits conjugate arithmetic symmetry only around $f_0 = \frac{f_s}{4} = 4$ kHz that is, $T_{BP}(f + f_0) = T_{BP}^*(f - f_0)$.

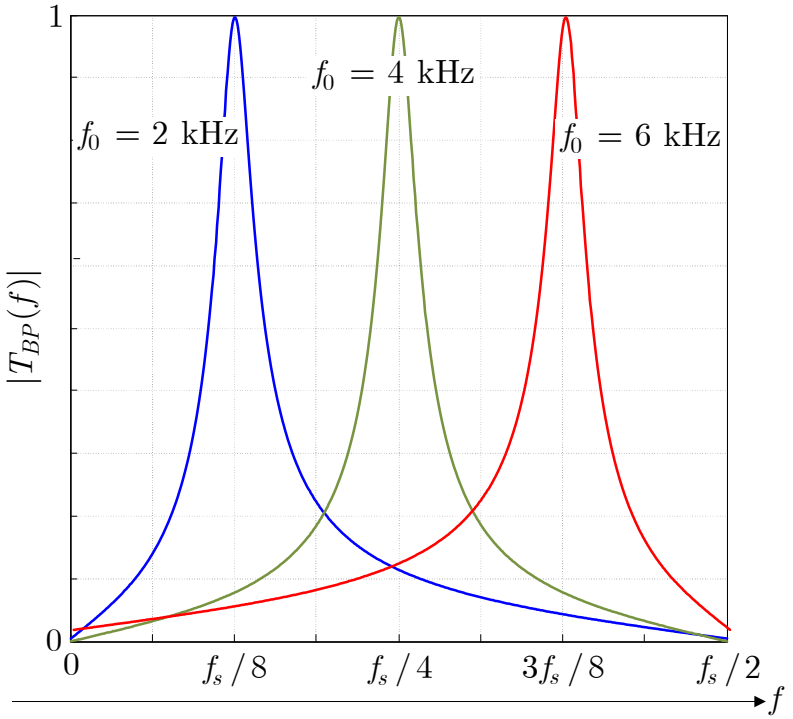

Figure 4a: Magnitude response of the second order bandpass IIR filter.

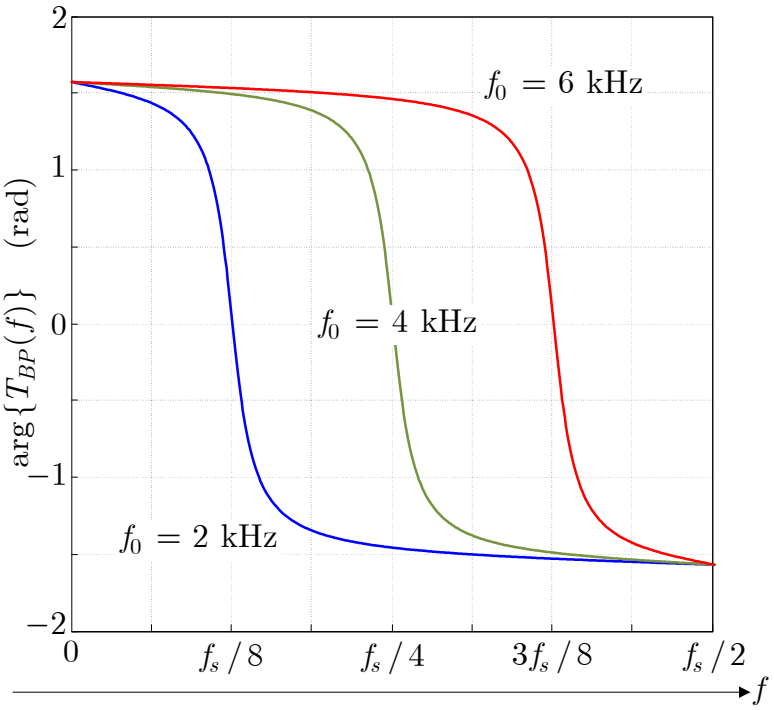

Figure 4b: Phase response of the second order bandpass IIR filter.

The filter's phase is exactly zero at $f = f_0$ and tends to $\pi/2$ as $f \to 0$ (and to $-\pi/2$ at $f = f_s/2$). The parameters K_f and K_B control the central frequency f_0 and the bilateral filter bandwidth B at -3 dB gain. They are given by:

$$K_f = \cos\left(2\pi \frac{f_0}{f_s}\right) \tag{3}$$

and

$$K_B = \frac{1}{1 + \cot\left(\pi \frac{B}{f_s}\right)}. \tag{4}$$

The filter implementation should be done using the direct form II, with the signal flow diagram represented in Figure 5:

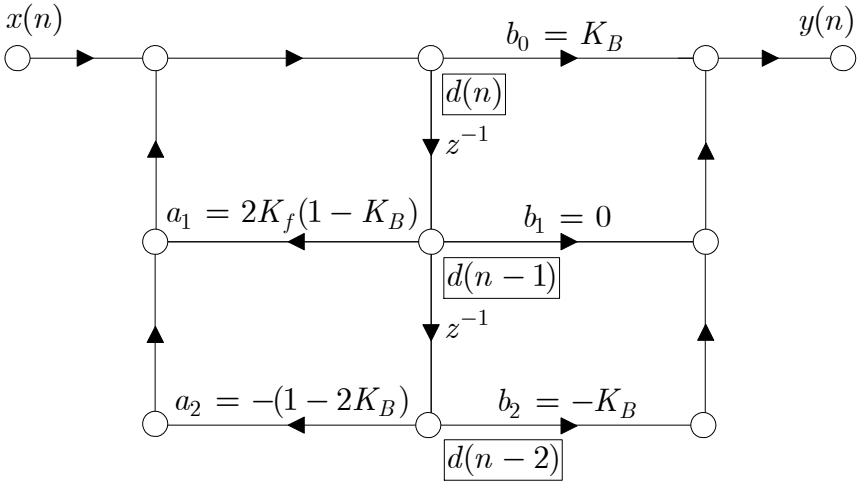

Figure 5: Signal flow diagram (direct form II) of the second order III bandpass filter.

The computation of the filter could be done using the auxiliary sequence $d(n)$ and determining the output accordingly to:

$$\begin{aligned} d(n) &= x(n) + a_1 d(n-1) + a_2 d(n-2) \\ y(n) &= b_0 d(n) + b_1 d(n-1) + b_2 d(n-2) \end{aligned} . \tag{5}$$

Because the input sample $x(n)$ enters directly in the computation (5) the dynamic range of the integer 16-bit words becomes occupied meaning that to compute $d(n)$ with enough accuracy, words with more bits (32-bit) are necessary, which increases the signal processing complexity and memory usage. Instead, you may compute the filter using the direct form I as

$$y(n) = a_1 y(n-1) + a_2 y(n-2) + b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) . \tag{6}$$

The advantage in this case is that the delay lines for $x(n)$ and $y(n)$ may be implemented with 16-bit integer variables. In addition, all partial products in (6) can be added using the 32-bit native adder/accumulator of the ALU so there will be no truncation involved until the output $y(n)$ is saved. This is not the case in (5) because truncation occurs at an intermediate stage, when $d(n)$ is saved.

(T) Filter analysis:

- Generate a table with the coefficients b_0, b_2, a_1 and a_2 for the three frequencies $f_0 = 2$ kHz, $f_0 = 4$ kHz and $f_0 = 6$ kHz, indicating their real values and the integer values in the chosen format for their representation.
- Write an *Octave* script which computes and replicates the results presented in Figures 4a and 4b. You can use the *Octave* function `freqz()` after loading the signal processing package with the command **`pkg load signal`.** The same function is available in *Matlab*.
- Explain what happens if you set either $f_0 = 0$ or $f_0 = \frac{f_s}{2}$ in equation (2), namely what kind of filter is obtained.

Frequency control

The goal of this processing block is to compute K_f in equation (3), which controls the filter central frequency using the information in the error signal $e_f(n)$. The central frequency should be 2 kHz $\leq f_0 \leq$ 6 kHz for $-0.5 \leq e_f(n) \leq 0.5$. To compute K_f start by using a linear

approximation as shown in Figure 6: noting that $f_0 = \frac{f_s}{4}[e_f(n)+1]$ then $K_f = \cos\left(2\pi\frac{f_0}{f_s}\right) = \cos\left(\frac{\pi}{2}[e_f(n)+1]\right) = -\sin\left(\frac{\pi}{2}e_f(n)\right) \approx -\frac{\pi}{2}e_f(n)$. After this compute the coefficient $a_1 = 2K_f(1-2K_B)$, the only one which depends on K_f, Check the fixed-point arithmetic, the computations and if the coefficient is properly computed in you C-code program.

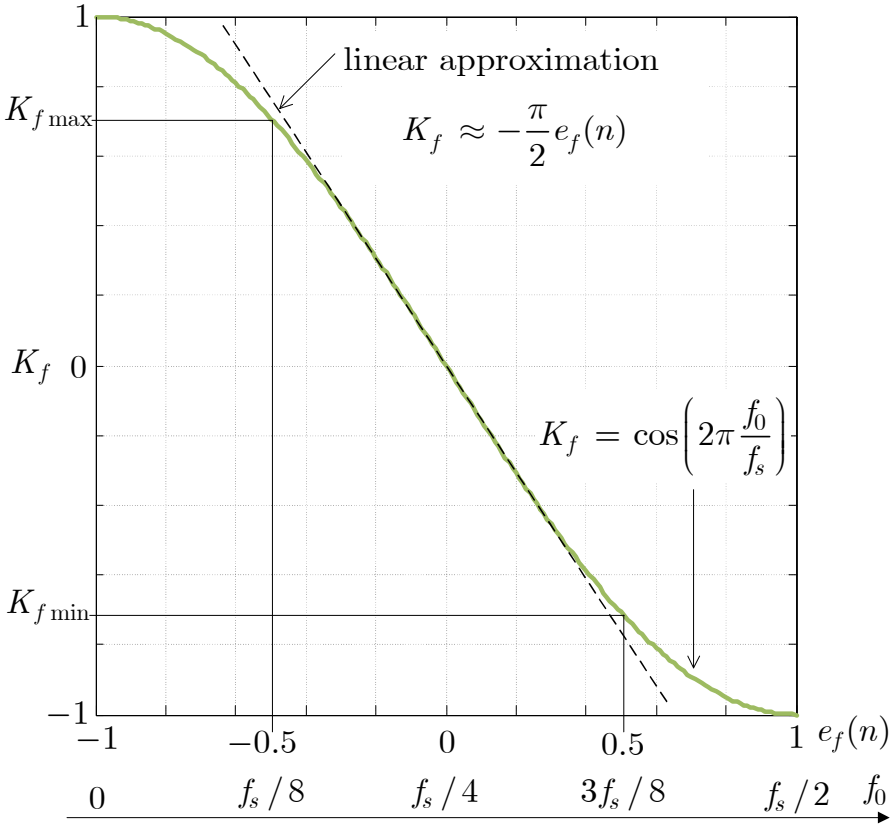

Figure 6: Variation of K_f as a function of $e_f(n)$ (notice the equivalent scale for f_0).

Note: The bandwidth of the bandpass filter is fixed so the value of K_B in equation (4) should be computed offline. This means that coefficients b_0, b_2 and a_2, which depend only on K_B, should also be computed offline. Only coefficient a_1 needs to be computed in real-time using equation (3).

Development methodology (laboratory)

The following are general guidelines to follow in the development of this project. To test the system, you should inject a sinusoidal signal from the Workbench signal generator with a peak-to-peak amplitude of 1.2 Vpp, zero DC offset and variable frequency.

1. (L) Passband IIR filter development:

1.1 Write the code of the difference equation of the filter using direct form I in equation (6). Be careful with the arithmetic fixed-point format used; use Q_{15} for the coefficients and the input variables (why?). Start with Q_{15} for the output variables and check if no overflows occur for the largest input signal amplitude. Otherwise keep decreasing the arithmetic precision (Q_{14}, Q_{13},...) until the dynamic range is enough to avoid overflows;

1.2 Test the filter for the 3 values of K_f (fix its value in each test) corresponding to the frequencies $f_0 = 2$ kHz, $f_0 = 4$ kHz and $f_0 = 6$ kHz using as input a sinewave from the bench generator.

2. (L) DPLL development:

2.1 Program and test the loop filter;

2.2 Integrate in your code the controlled oscillator developed in the previous project;

2.3 Close the DPLL loop using the multiplier phase detector;

2.4 Test the DPLL and determine the lock and hold range by performing one ascending and one descending frequency sweep from an unsynchronized state, fill a table like Table 1 below.

Table 1: DPLL characteristics with first gain value.

	Parameters	NCO gain K_0 =			
	Amplitude	A_{max} =1.2 Vpp		A_{max} / 2 =0.6 Vpp	
	Loop filter f_c	f_c = 100 Hz	f_c = 10 Hz	f_c = 100 Hz	f_c = 10 Hz
Ascending sweep	f_C^-				
	f_L^+				
Descending sweep	f_C^+				
	f_L^-				
	$\Delta f_C = f_C^+ - f_C^-$				
	$\Delta f_L = f_L^+ - f_L^-$				

2.5 Check that the input sinewave coming from the generator has maximum amplitude $A_{\max}$ without distortion (this should be about 1.2 Vpp). Increase the value of the NCO gain K_0 until the hold range extends from about 2000 Hz to 6000 Hz. Then performing one ascending and one descending frequency sweep starting from an unsynchronized state, fill Table 2 below (see Notes on last page).

Table 2: DPLL characteristics with second gain value.

	Parameters	NCO gain K_0 =			
	Amplitude	$A_{\max}$ =1.2 Vpp		$A_{\max}$ / 2 =0.6 Vpp	
	Loop filter f_c	f_c = 100 Hz	f_c = 10 Hz	f_c = 100 Hz	f_c = 10 Hz
Ascending sweep	f_C^-				
	f_L^+				
Descending sweep	f_C^+				
	f_L^-				
	$\Delta f_C = f_C^+ - f_C^-$				
	$\Delta f_L = f_L^+ - f_L^-$				

The results in this table will allow you to check the behavior of the DPLL and compare it qualitatively with the theoretical expected results for an analog PLL (the results are quantitatively different because the PLL in this project is a digital one). Please comment on these results in your report.

3. (T, L) Develop the bandpass filter frequency control block **using temporarily** $e(n)$ as $e_f(m)$. As stated before, you must compute the approximation $K_f \approx -\frac{\pi}{2} e_f(n)$ and then the filter coefficient $a_1 = 2K_f(1-2K_B)$. Note that, because the NCO gain has been readjusted, the output of the post-detection filter (and of the DPLL error signal) may no longer be confined to $-0.5 \le e_f(n) \le 0.5$. Check this by driving the input frequency to one extreme of the lock range (but still maintaining synchronization), stop program execution and check the variable representing $e_f(n)$. If it is not ± 0.5 you will need to rescale it by multiplying it by a suitable factor.

4. (L) Develop the post detection filter and the decimation mechanism using a CIC filter. Implement the CIC decimator block using the most favorable implementation strategy. To test the CIC filter/decimator use a smaller value $M = 32$ and as input a sinewave from

the bench generator with frequency varying from DC to $f_s/(2M) = 250$ Hz. Observe both the integrator output and the CIC output. Comment on these results.

5. **(L)** Connect all blocks together and test the complete system. Make sure you are using $A_{\max}$ and $f_c = 100$ Hz. The bandpass filter central frequency should track the frequency

Table 3: Tracking filter global characterization.

	Parameters	NCO gain $K_0 =$ $M =$	$A = A_{\max} \approx 1.2$ V $f_c =$
	Input frequency (kHz)	Output amplitude (V)	
		Approximate control	Exact control
Ascending frequency sweep	1.0		
	1.5		
	2.0		
	2.5		
	3.0		
	4.0		
	4.5		
	5.0		
	5.5		
	6.0		
	6.5		
	7.0		
Descending frequency sweep	7.0		
	6.5		
	6.0		
	5.5		
	5.0		
	4.5		
	4.0		
	3.5		
	3.0		
	2.5		
	2.0		
	1.5		
	1.0		

of the incoming sinewave in the range 2 kHz to 6 kHz. Measure these ranges accurately. Fill the column "Approximate control" in Table 3 above which characterizes the tracking filter global performance. In the same graph, sketch the amplitude response for both frequency sweeps.

6. Modify the frequency control block to have the exact control variable $K_f = -\sin\left(\frac{\pi}{2} e_f(m)\right)$ and, afterwards $a_1 = 2K_f(1-K_B)$ as before. For this, use the sine LUT already available in your program and interpolation. Repeat the test of the complete system filling the column "Exact control" in Table 3. Compare the results of the approximate with the exact control strategy.

Note on 2.5: The hold range should increase linearly with K_0. Thus, following a simple proportional rule between the old and the new (desired) hold range, you can compute a first approximation for the new value of K_0.

Note on 4: If you are unable to implement the decimation CIC filter, it may be replaced by a first order lowpass filter like the one in the DPLL loop, with a cutoff frequency of $f_c = 1$ Hz (need to recompute the value of α), followed by a simple decimator with factor M (keep only one sample out of every M filter output samples and use it to program the passband filter central frequency).

Table 4: Main system parameters.

Block	**Parameter**	**Symbol**	**Value**
Band-pass filter	Minimum resonance frequency	$f_{0_{\min}}$	2 kHz
	Central resonance frequency	f_0	4 kHz
	Maximum resonance frequency	$f_{0_{\max}}$	6 kHz
	Bilateral bandwidth @ -3dB	B	400 Hz
DPLL loop filter	Cut-off frequency	f_c	100 Hz
Controlled oscillator	Minimum frequency	$f_{0_{\min}}$	2 kHz
	Maximum frequency	$f_{0_{\max}}$	6 kHz
Post-detection and decimation filter	Decimation factor	M	8192 (may vary)
General	Sampling frequency	$f_s = 1/T_s$	16 kHz
	Maximum input signal peak-to-peak amplitude	—	1.2 Vpp