

Estimating Rotational Speed with a Phase- Locked Loop

NIKLAS WILLEMSSEN



KTH Electrical Engineering

Masters' Degree Project
Stockholm, Sweden June 2008

XR-EE-RT 2008:015

Abstract

Volvo Construction Equipment Components AB (Volvo CE) develops electronic control systems for various vehicles in the field of construction equipment. Increasing demands on quality, safety and environmental care require more and more sophisticated and intelligent drive control systems. These new systems are dependant on high quality input data from various sensors. One of these sensors is of inductive type and is used for measuring rotational speed of a gear wheel, and the method of doing this is the subject of this thesis.

The current method applied at Volvo CE for measuring this speed needs to be changed in order to cope with these new demands, especially at low velocities. The alternative method presented in this paper is based on Phase-locked loop (PLL) techniques, which is a closed loop frequency control system.

In this thesis two different ways of implementing a PLL are investigated; a Quadrature phase-locked loop (QPLL) in software and a second order PLL in hardware. The QPLL is derived through optimization theory and implemented in Simulink. The second order PLL is the integrated circuit LM565 from the manufacturer National Semiconductor.

The QPLL is tested in a wheel loader and compared with the current method used at Volvo CE today. The results show that in the general case the QPLL is not better than the current method but at fast retardations and when changes of the velocity directions occur the QPLL outperforms the current method. A drawback with the QPLL is its behaviour at constant low velocities where it tends to lose track of the signal.

The hardware-PLL is tested in a test bench with similar working conditions as in a machine. The results show that this method has a limited frequency interval but in this interval the performance is as good as the current method and very robust.

The conclusion of this thesis is that PLL methods are interesting and could improve the quality of the measurements when the velocity is changing very fast but in the general case the current method is just as good. Implementing a PLL would require some extra expenses in terms of additional hardware but with the results presented in this thesis it does not seem cost effective.

Index

1. Introduction	7
2. The current method for measuring rotational speed	8
2.1. The inductive sensor	8
2.2. From sensor output to velocity estimation	9
2.2.1. Filtering and signal adaptation	9
2.2.2. Time delays caused by filtering and signal adaptation	11
2.2.3. The interrupt routine	11
2.2.4. Algorithms used in the ECU	11
3. The Sensor Output	13
4. Phase-locked loops – Theory	16
4.1. Basic working principles of a PLL	16
4.2. Mathematical derivation of a first order PLL	16
4.3. The second order PLL	17
4.4. Previous use of PLL's in the automotive industry	20
4.5. How can a PLL improve our measurements of rotational speed?	21
4.6. The gradient descent method for design of a more complete PLL	21
4.6.1. The algorithm	21
4.6.2. Simulations	23
4.7. The quadrature phase-locked loop (QPLL)	27
4.7.1. The algorithm	27
5. Implementation of a PLL in hardware	30
5.1. Necessary components	30
5.1.1. The phase detector	30
5.1.2. The voltage controlled oscillator (VCO)	30
5.1.3. The PI-controller/Loop filter	31
5.2. The LM565C Phase-locked loop integrated circuit	31
5.2.1. Limitations and tuning	32
5.2.2. Comments	33
6. The QPLL – Machine testing	34
6.1. Previous results	34
6.2. Signal processing	34
6.2.1. The current ECU estimation	34
6.2.2. The QPLL estimation	34
6.3. Results from machine testing	34
6.3.1. A hard brake	34
6.3.2. A Hard brake and change of direction	36
6.3.3. Periodic change in velocity	40
6.3.4. A constant very low velocity	41
7. The PLL in hardware – Results from a test bench	42
7.1. The test bench	42
7.2. Signal processing	43
7.2.1. The current ECU estimation	43
7.2.2. The PLL in hardware	43
7.2.3. The Encoder estimate	43
7.3. Results from the test bench	43
7.3.1. Periodic change in velocity	43
7.3.2. Step changes in velocity	44
7.3.3. Testing the limits	45
7.4. Robustness	47
8. Discussion and conclusion	49

8.1.	The QPLL.....	49
8.2.	The hardware-PLL	49
8.3.	Other possible improvements.....	50
9.	Future work	51
10.	Bibliography	52
10.1.	Scientific papers	52
10.2.	Web pages	52

1. Introduction

This thesis deals with the problem of measuring the rotational speed of a rotating gear wheel used for estimating the velocity of e.g. a Volvo CE wheel loader and other construction equipments. The current method for doing this has been working well for over 20years, but as more advanced control algorithms are being developed new and more accurate ways of estimating this frequency are needed.

The main focus of this paper is to investigate the possibility of using Phase-locked loop (PLL) methods as an alternative way to estimate the frequency of the rotating gear wheel. A Phase-locked loop is basically a closed loop control system normally used for phase/frequency control. Its usage is widely spread in the industry and PLL circuits can be found in everything from a cell phone to the engine of a car. In our case we will use a PLL as a frequency estimator for the gear wheel. A tutorial of PLL systems is given in [1].

As a first part of this thesis the current method used at Volvo CE is described and its limits in terms of frequency range and time delays are investigated. The characteristics of the sensor currently used are also discussed and its shortcomings explained.

In the second part we are building up a theory for the different Phase-locked loop methods and we motivate why we expect them to work and how they can improve our measurements. We start with a simple and intuitive PLL structure to illustrate the main functionality in a straight forward manner, and later moving on to more complex structures by the use of optimization theory. We also take a look at what electronic circuit blocks are needed to implement a PLL in hardware and the limitations that exist.

The next part consists of results from testing. The different methods were extensively tested both in a test bench as well as in a real machine and their results compared with the current method used today. Special attention is given to the behaviour at fast retardations and low velocities since it has been reported that the current method performs poorly during these conditions.

In the last part we discuss the results and differences between the methods previously investigated. Conclusions are drawn about the feasibility to use PLL methods for estimation of gear wheel speed and the necessary modifications needed for such an implementation. At the end some possible future work is presented.



Figure 1 Picture of a Volvo CE wheel loader [13]

2. The current method for measuring rotational speed

The method used today for measuring rotational speed uses an inductive sensor applied to a gear wheel in the gearbox. Since its velocity is directly proportional to the velocity of the wheels their velocity can also be calculated knowing it. The analogue signal from the sensor is filtered to reduce noise and then digitalized by transforming the signal into a square wave with one rising edge per gear that passes. This signal is sent to the onboard microcomputer, the ECU (Electronic Control Unit), which by knowing the distance between two gears and the time between two rising edges can estimate a velocity.

2.1. The inductive sensor

The sensors used in Volvo CE gearboxes today are of inductive type. These sensors react to changes in the magnetic field and a current is induced in them proportional to the rate of change. This change is caused by the differences in reluctance when a gear passes the sensor as to when the sensor is located between two gears, as seen in Fig. 2 and Fig. 3.

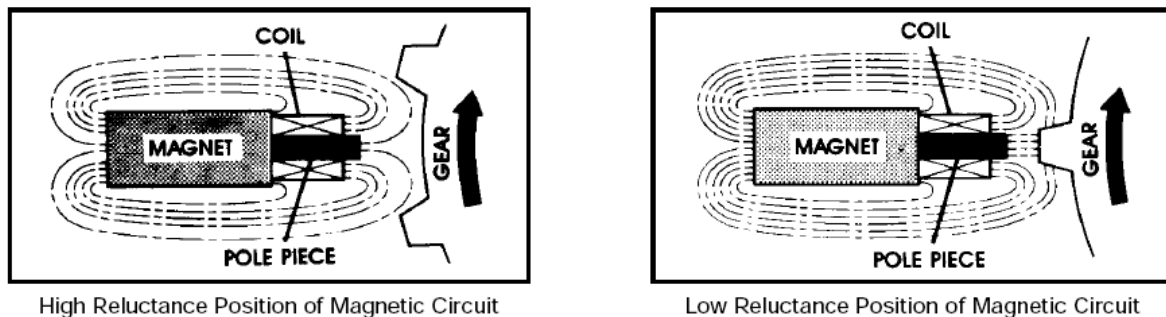


Figure 2 The inductive sensor seen from the side [8]

When the gear wheel has a high velocity the change in the magnetic field will be faster than at lower velocities and thus give a stronger signal since the strength of the output signal is proportional to the rate of change of the magnetic field. This fact will put a limit on the sensor at low velocities since the signal will be weakened and a gear passing by might not be detected in the output. The sensors used in Volvo CE transmissions are specified for frequencies (gears/s) from 20Hz to 4500Hz which with our gear wheel with 28 teeth correspond to velocities from 0.4km/h to 50km/h in a medium size wheel loader. Another limiting factor on the sensor is that we cannot detect zero velocity due to the fact that the gear wheel needs to be in constant movement in order to get an induced signal from the sensor. Since the gear wheel is symmetric and we can only measure the rate of change in the magnetic field it is also impossible to determine the direction in which the wheel is spinning.

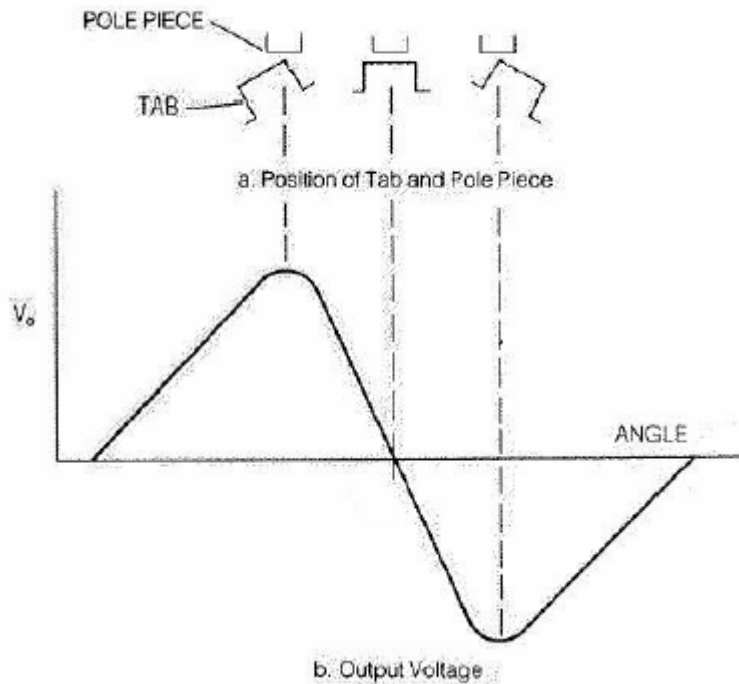


Figure 3 Gear position and corresponding sensor output [10]

Another factor that determines the strength of the sensor output is the air gap between the sensor and the gear wheel. In the HT-200 gearbox the ideal distance from sensor to gear wheel is 0.7mm when measuring the speed on the outgoing axle with a tolerance of +0.7mm/-0.4mm. This tolerance is due to tolerance stack-up in the different parts and the fact that the sensor is just tightened down in the transmission housing without any fine adjustment. The tolerance is of importance since e.g. a change of distance between the sensor and the gear wheel from 0.5mm to 1.0mm might diminish the sensor output amplitude with more than 50%.

2.2. From sensor output to velocity estimation

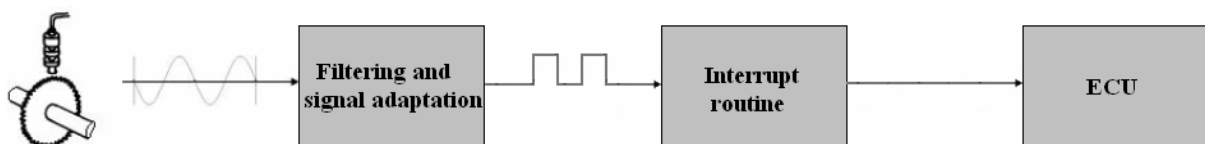


Figure 4 Processing of the sensor output

The output of the sensor will pass through three main blocks before an estimation of the velocity can be made as illustrated in Fig. 4. In the first block the signal will be filtered and digitalised in form of a square wave, the second block will count the pulses and give them a corresponding time stamp and the last block will use the information in the second block to calculate the velocity.

2.2.1. Filtering and signal adaptation

The first block in Fig. 4 deals directly with the noisy output signal of the sensor. This signal is filtered and transformed into a square wave in order for the interrupt routine to be able to further process the signal. This is done in five separate steps as shown in Fig. 5. A more detailed description of this process can be found in [8].

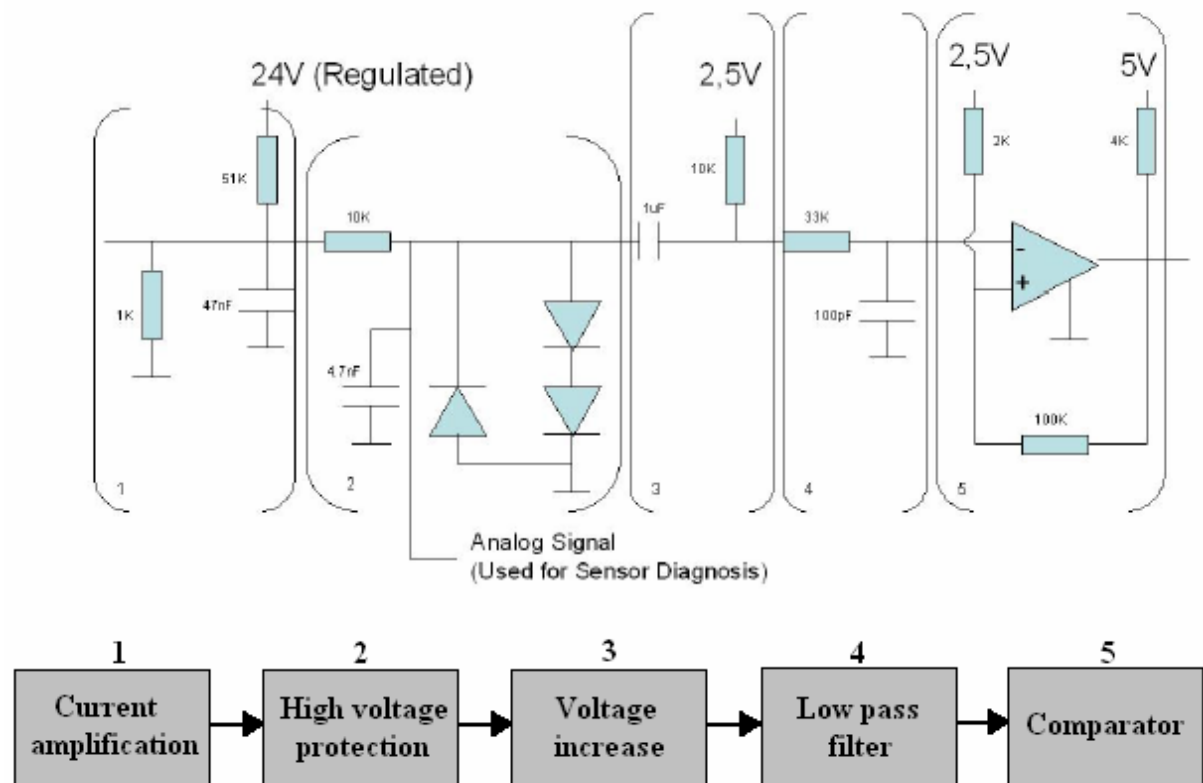


Figure 5 Filtering and signal adaptation, a circuit diagram of the five step process [8]

1. Current amplification

Since the sensor is of inductive type and can only generate a small current, it needs to be amplified.

2. High voltage protection

The second block makes sure that the voltage over it is not too high. Each diode allows for a potential drop of 0.7V which in turns limits the outgoing voltage to +1.4V/-0.7V. This block will also have a low pass character due to the resistor and capacitor.

3. Voltage increase

When the signal leaves the second block the voltage level is centred around zero and both positive and negative voltages are possible. The third block will centre the voltage around +2.5V in order to have only one possible sign of the voltage since this signal is easier to process. This block will also act as a high pass filter removing low frequency noise.

4. Low pass filter

This block is just an ordinary low pass filter removing all the high frequency noise.

5. Comparator

The comparator is used to create a square wave from the now filtered and adapted sensor output. This is done by the use of an inverting operational amplifier circuit, which means that a high value of the ingoing signal will produce a “low” on the square wave and vice versa. There is also some hysteresis in the comparator which triggers at 2.5V +0.63mV/-0.66mV. The resulting square wave will have an amplitude of 4.89V. This behaviour is illustrated in Fig. 6.

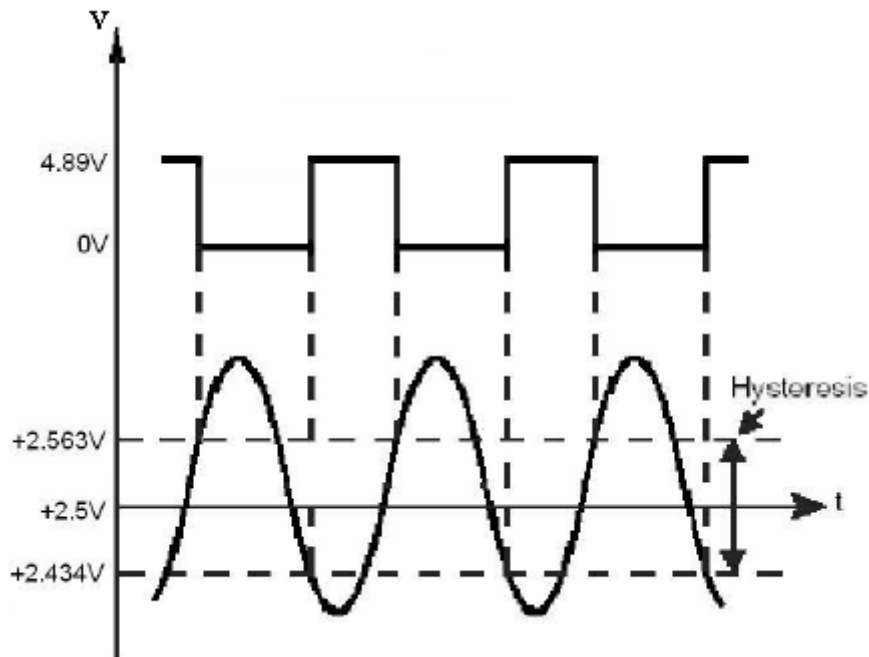


Figure 6 Illustration of the hysteresis in the comparator [8]

2.2.2. Time delays caused by filtering and signal adaptation

The cut off frequency of the low pass filters are 3400Hz (Block 2) and 48kHz (Block 4) respectively and the high pass filter (Block 3) at 16Hz. This gives a span from about 16Hz to 3400Hz where the filters do not decrease the signal strength. The time delay caused by Block 2 is in the order of a few microseconds and Block 4 between 30 μ s to 50 μ s for frequencies varying from 20Hz to 4500Hz. Block 3, the high pass filter, on the other hand can have significant time delays reaching from 60 μ s at around 4500Hz to 8ms at 20Hz. Clearly Block 3 will decide the time delay caused by the three filters; this delay is of importance especially at low frequencies.

Time delays are also caused by the hysteresis block when converting the sine-like wave into a square wave. These delays are harder to estimate but given the assumption that the input is a perfect sine wave we will have delays of up to 10-15ms at low frequencies (20Hz) and only a few microseconds at high frequencies (4500Hz). The reason that the delays are much larger at low frequencies is that the input amplitude is lower and the sine wave is oscillating slower taking longer time to reach the triggering point.

2.2.3. The interrupt routine

This routine is the first step that involves some low level software. Every rising edge of a square wave is counted and given a corresponding time stamp which is stored in the hardware and later sent to the software for processing. The hardware can only store one time stamp which is overwritten every time a new interrupt (rising edge) occurs.

2.2.4. Algorithms used in the ECU

The ECU calculates a velocity every 10ms. It obtains the number of gears that have passed by from the interrupt routine, the time of the last gear in the previous period is stored in the software and the time of the last gear in the current period is still in the hardware register. With this information and knowing the angle between two gears the ECU can simply divide

the distance covered by the gear with the elapsed time in order to get an estimation of the mean velocity during the current period.

This algorithm works well when we have frequencies higher than 100Hz (interrupts/s) which guarantee that at least one interrupt per sampling period occur. When this is not the case we do not get an interrupt every 10ms and the algorithm will act in a different way. When the ECU encounters a sample period without any interrupts the velocity is set to 100Hz, seeing that this is the highest possible mean velocity in a sample period with no interrupts, or the latest known value if this was lower than 100Hz. If another period goes by without any interrupts the estimated frequency is set to 50Hz, and for every additional sample-free period; 33Hz, 25Hz, 20Hz, 16Hz, 14Hz, 12Hz, 11Hz, 10Hz, 0Hz, but only if this velocity is lower than the latest known. For example if the gear wheel is left rotating for a while at 16Hz and then makes a quick drop to zero, the first period without a gear the velocity estimate will remain at 16Hz, and for the following five periods as well, but then it will drop to 14Hz, 12Hz etc. until it reaches zero.

A drawback with this algorithm is that we only get the mean velocity over a 10ms period and if the velocity is changing very quickly the velocity at the end of a period might be far from the mean value. Another drawback is what happens when the gear wheel makes a sudden drop to zero velocity. It will take the ECU 100ms to detect this with the algorithm mentioned above.

3. The Sensor Output

In this chapter we will study the output from the sensor. How does its amplitude vary with the frequency and in what frequency interval can it operate properly?

The following plots are all from a sensor on a Volvo L150F wheel loader, which is a medium size wheel loader of latest model.

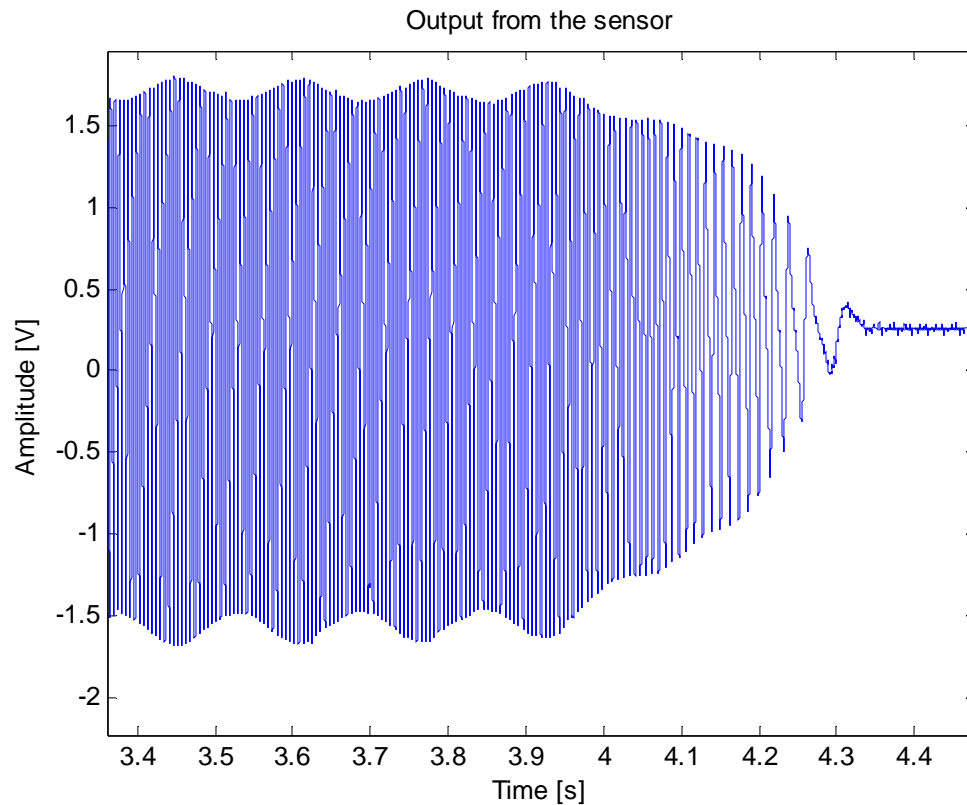


Figure 7 Sensor output during a hard brake which occurs at 4s

Fig. 7 shows the sensor output when the wheel loader drives at a constant, relatively low velocity, and then makes a hard brake at 4s. We can see that both the frequency and the amplitude go to zero, as expected. When the gear wheel is rotating slower the induced voltage in the sensor will also be smaller, which in turn gives a smaller amplitude, as mentioned previously. The periodic small change in amplitude at constant frequency is due to the fact that the gear wheel is not completely round but this wobbling behaviour is of less importance.

Fig 8 – Fig 10 show the sensor output at 200Hz, 10Hz and 5Hz respectively.

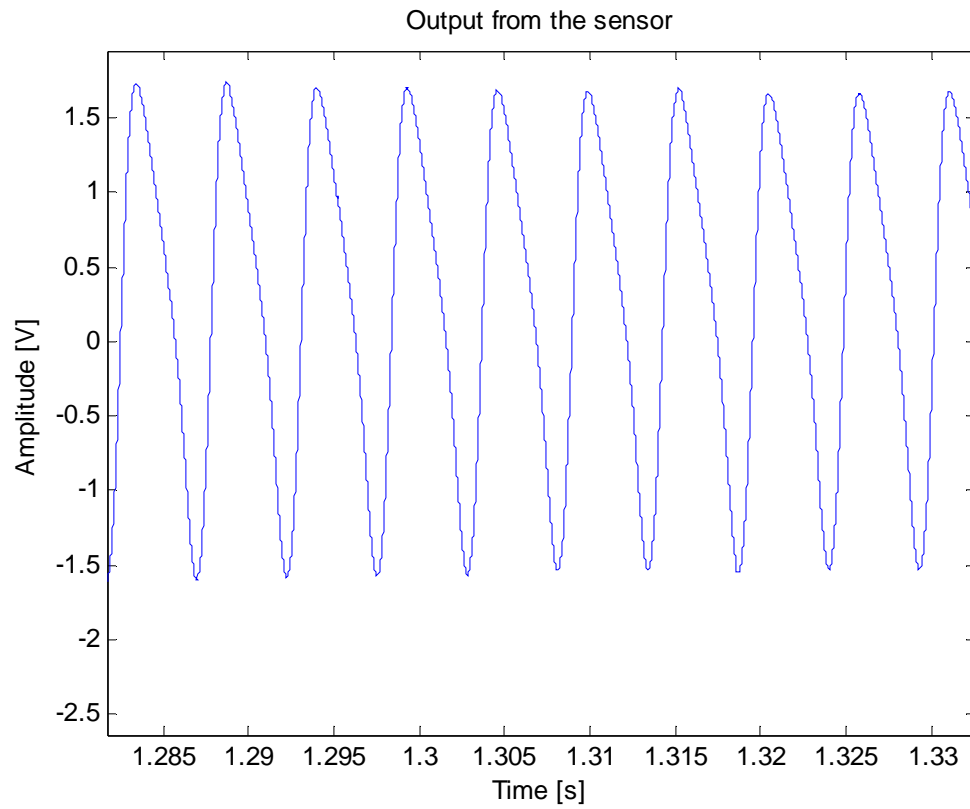


Figure 8 Enlarged plot of the sensor output at 200Hz (4km/h)

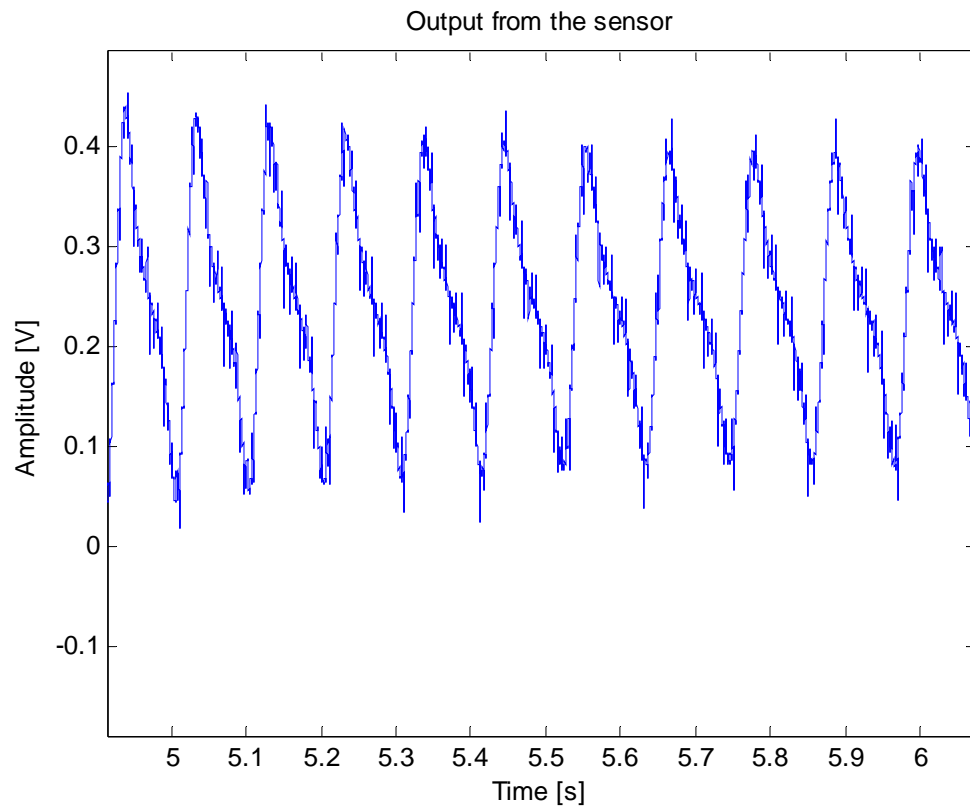


Figure 9 Enlarged plot of the sensor output at 10Hz

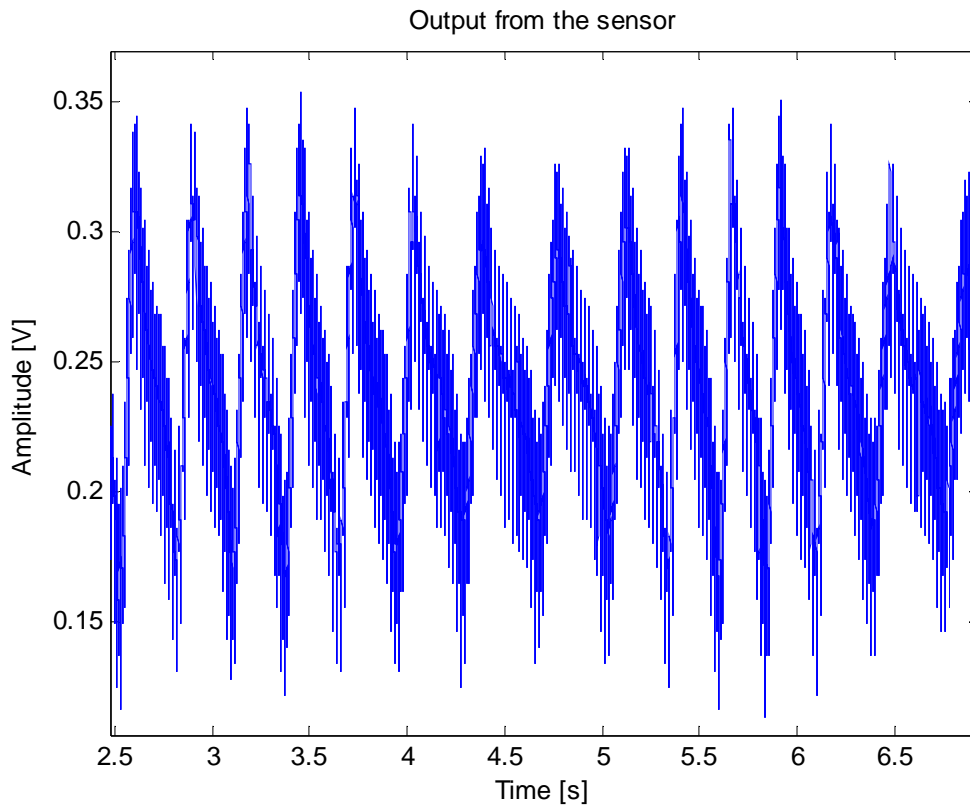


Figure 10 Enlarged plot of the sensor output at 5Hz

It can be seen that at high frequencies (200Hz) we have a strong and relatively noise-free signal which will be easy to extract a measurement from. At 10Hz we start to notice some ripples in the signal but the peaks are still easily detectable. At 5Hz the signal is very noisy and it is very hard to detect the exact location of the peaks etc. The sensor is only specified to be able to work down to 20Hz but in our case it produces a good signal all the way down to 10Hz. It is hard to say if this is true in all machines seeing as the sensor assembly has some tolerances and we might just have gotten a machine with the sensor placed in its optimal position.

As a conclusion we see that at high frequencies we have a strong and noise-free signal from which it will be easy to estimate a velocity, but at lower frequencies, 10Hz – 20Hz, the signal is weak and noisy and below this value we will have trouble estimating the velocity with the current sensor solution.

4. Phase-locked loops – Theory

In this chapter we will investigate the behaviour of a so called phase-locked loop (PLL), an electronic circuit widely used in various fields of the industry today. This device takes a periodic signal as input and generates a sine wave with the same frequency/phase as output. Previously this device has almost only been used for tracking of high frequency signals but we will investigate if it can be used for lower frequencies as well.

4.1. Basic working principles of a PLL

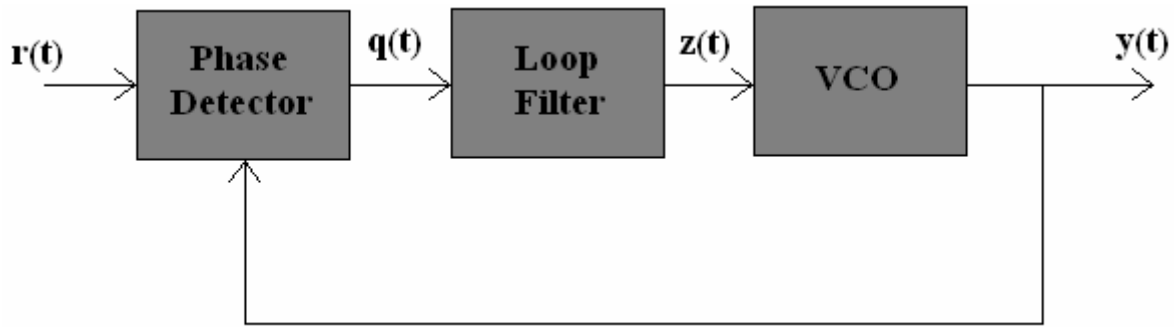


Figure 11 Block diagram of a general PLL structure

A PLL consists generally of three main parts; a phase detector, a loop filter and a voltage controlled oscillator (VCO). The phase detector detects the difference in phase between the input and the output and sends this information to the loop filter for further process. The filtered signal is sent to the VCO which will change its oscillating frequency to reduce the phase difference. If e.g. the output has a lower phase than the input the VCO will automatically increase its own oscillating frequency until it “locks” on and has the same frequency and phase as the input. A PLL may also be configured to have multiples of the input frequency as its output, allowing it e.g. to be sampled at a higher rate, or simply give the frequency directly as its output.

4.2. Mathematical derivation of a first order PLL

Here a simple example will follow illustrating the mathematical equations that a PLL obeys.

Assume that the input and the output are on the form:

$$r(t) = V_i \sin(\omega_i t + \varphi_i(t)) = V_i \sin \psi_i(t) \quad (4.1)$$

$$y(t) = V_o \cos(\omega_o t + \varphi_o(t)) = V_o \cos \psi_o(t) \quad (4.2)$$

Where ω is the angular frequency and φ the phase with the corresponding subscripts for the input and the output respectively.

A phase detector is a nonlinear element of various designs but in our example we will assume that it is a simple multiplier, just multiplying the two inputs together giving:

$$\begin{aligned}
q(t) &= r(t)y(t) = V_i V_o \sin(\omega_i t + \varphi_i(t)) \cos(\omega_o t + \varphi_o(t)) = \\
&= \frac{1}{2} V_i V_o \sin[(\omega_i - \omega_o)t + \varphi_i(t) - \varphi_o(t)] + \frac{1}{2} V_i V_o \sin[(\omega_i + \omega_o)t + \varphi_i(t) + \varphi_o(t)]
\end{aligned} \tag{4.3}$$

The main idea with doing this is that the signal is now separated into two different frequencies $(\omega_i - \omega_o)$ rad/s and $(\omega_i + \omega_o)$ rad/s. When the input and the output have somewhat similar frequencies the first part of (4.3) will have a much smaller frequency than the second part. This is used in the loop filter, which has low-pass character that filters out the higher frequency component. In our simplified example we will assume that the loop filter completely cancels the second term of (4.3) whilst leaving the first term unaltered giving:

$$z(t) = \frac{1}{2} V_i V_o \sin[(\omega_i - \omega_o)t + \varphi_i(t) - \varphi_o(t)] = K \sin \psi_e(t) \tag{4.4}$$

We see that $z(t)$ is a function of the phase difference of the two signals. If this difference is relatively small the following approximation can be made:

$$z(t) = K \sin \psi_e(t) \approx K \psi_e(t) \tag{4.5}$$

Making the output of the loop filter linearly dependant of the difference in phase, this linear dependence is used to control the VCO. The VCO has a free running frequency (ω_c) which is the frequency of the sine wave that it sends out when no control input is applied to it. When a control voltage is present the change of the VCO frequency will be linearly dependent on the amplitude of this voltage. Seeing as the control voltage is linearly dependent on the phase difference, the change in VCO frequency will also be a linear function of the phase difference and this is exactly what we want. If the difference is positive we want the VCO to “speed up” and “lock on” to the phase quickly and if it is negative we want the VCO to “slow down”. The output frequency of the VCO:

$$\omega_o = \omega_c + K_{VCO} z(t) \tag{4.6}$$

Where K_{VCO} is called the sensitivity of the VCO and is normally given in [Hertz/Volt].

We can expect good behaviour for small differences in phase angle but what happens when the difference is large and $K \sin \psi_e(t) \neq K \psi_e(t)$ or if the difference in frequency between the input and the VCO output is large as well, periodically changing the phase difference? The system described above is a first order PLL and can not deal with these constraints in a proper way; we need a second order PLL to do so. The second order PLL will have a proportional integrator in the loop and will act as a PI-controller for the system. If the input frequency is much larger than the VCO frequency the proportional integrator will integrate this difference and act as an input bias to the VCO. When the frequency difference is small enough the PLL will be able to “lock on” to the phase and frequency as in a first order PLL.

4.3. The second order PLL

In Fig. 12 follows a simple Simulink model of a second order PLL:

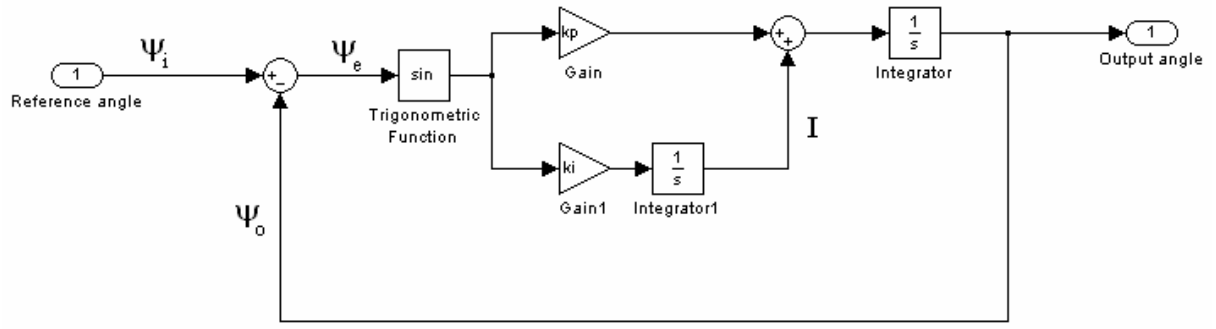


Figure 12 A simple model of a second order PLL system

We can see that the difference from a first order PLL is the proportional integrating action after the $\sin()$ function. The corresponding voltage level after this PI-controller is proportional to a frequency, and an integration will give us the corresponding phase angle which is fed back through the system.

Consider the corresponding system of equations:

$$\dot{\psi}_o = k_p \sin(\psi_e) + I \quad (4.7)$$

$$\dot{I} = k_i \sin(\psi_e)$$

and make the following substitution:

$$x_1 = \psi_e \quad (4.8)$$

$$x_2 = \omega_e - I$$

Where $\omega_e = \omega_i - \omega_o$

One important fact to notice is that at time zero $I = 0$ and the initial value of x_2 is:

$$x_2(0) = \omega_e, \text{ i.e. the difference in frequency} \quad (4.9)$$

This gives us the new system:

$$\begin{cases} \dot{x}_1 = \dot{\psi}_i - k_p \sin(x_1) + x_2 \\ \dot{x}_2 = -k_i \sin(x_1) \end{cases} \quad (4.10)$$

We will now study the case when $\dot{\psi}_i = C$ i.e. the input have a constant frequency and phase angle and can be incorporated in x_2 . We want the system to converge to $x_1 = 0$ since x_1 is the error in phase.

A phase portrait of the system is given in Fig. 13.

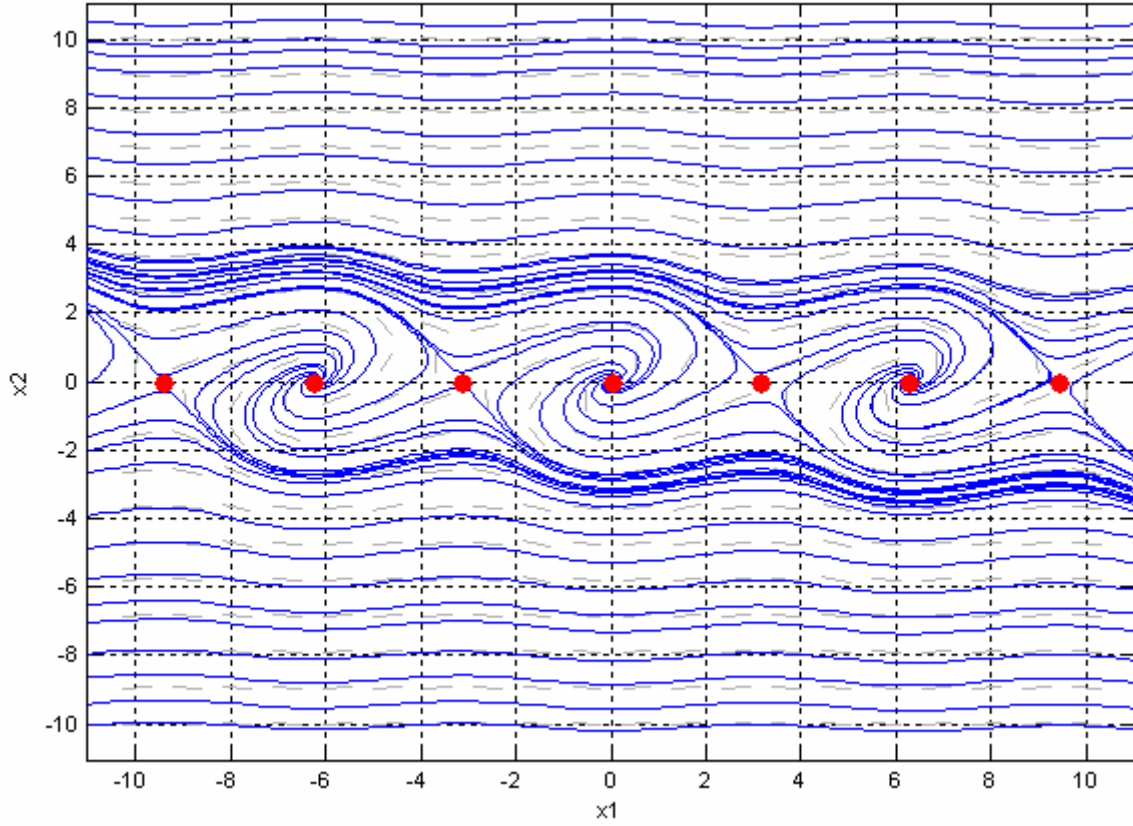


Figure 13 A phase portrait of the system

The equilibrium points, with $k_i = 1$, $k_p = 1$ are:

$$\begin{aligned} x_1^* &= n\pi \\ x_2^* &= 0 \end{aligned} \quad n = 1, 2, \dots \quad (4.11)$$

By a simple linearization procedure it can be seen that even n 's will give a stable node while odd n 's will give an unstable saddle point. Global asymptotic stability can be proven for the even integer n point if $\{k_i, k_p\} > 0$. Which one of these equilibriums the solution converges to depends on the initial values of $x_1(0)$ and $x_2(0)$. If x_1 and x_2 both are small, i.e. we have a small initial phase and frequency difference the solution will converge to $x_1 = 0$, $x_2 = 0$ and the PLL will be in lock with both the frequency and phase of the input. But if x_2 is large we will have a so called cycle slip and the solution will converge to one of the equilibrium point further away from $x_1 = 0$, which is a multiple of 2π . An example trajectory line of this phenomenon is illustrated in Fig. 14.

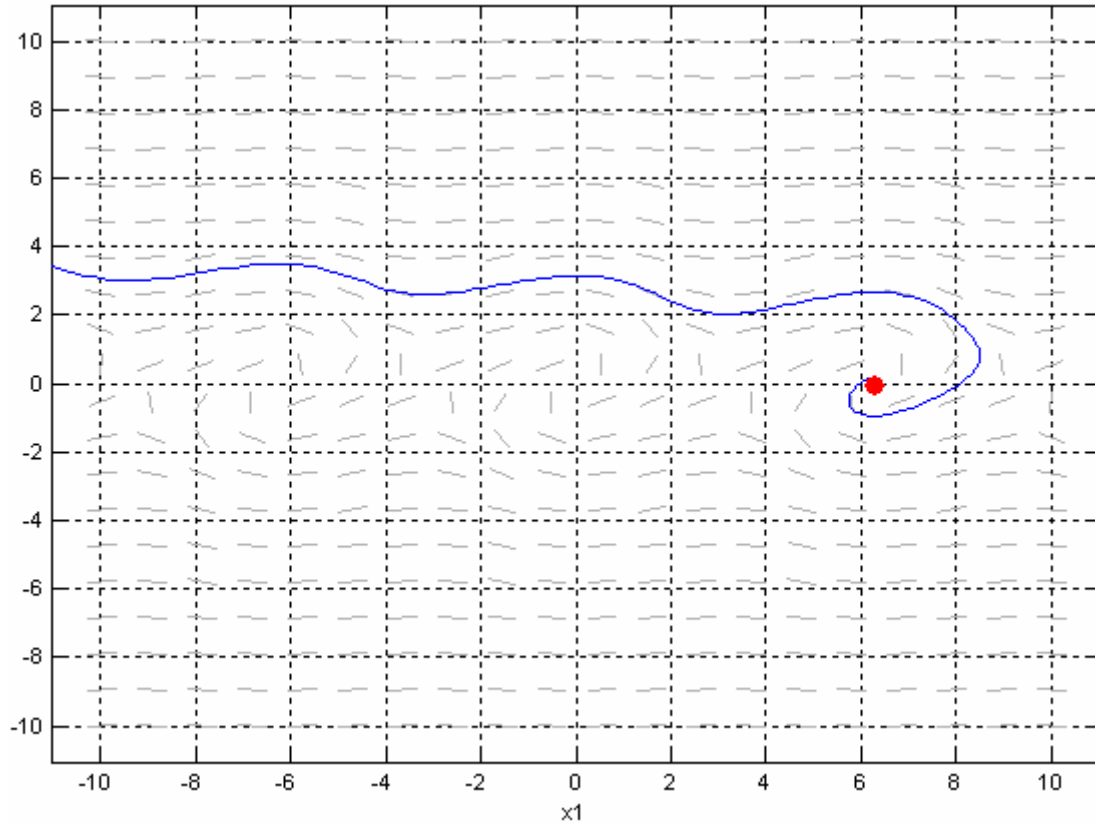


Figure 14 A 2π cycle slip

These cycle slips are highly undesirable in a PLL since they make the PLL converge slower, but it will always converge to the following value after the transient period:

$$\begin{aligned} x_1^* &= n2\pi \\ x_2^* &= 0 \end{aligned} \tag{4.12}$$

The time it takes for the system to converge is called the pull in time, T_p , and can be approximated by the following equation, derived in [1]:

$$T_p = \frac{\omega_e^2}{k_i k_p} \tag{4.13}$$

We see that if we have a large frequency difference we need high gains in the PI-controller.

4.4. Previous use of PLL's in the automotive industry

In the 1970's the automotive industry began to use PLL components to regulate the engine speed. The system consists of three parts; a phase detector, a loop filter and an engine serving as the VCO in this case.

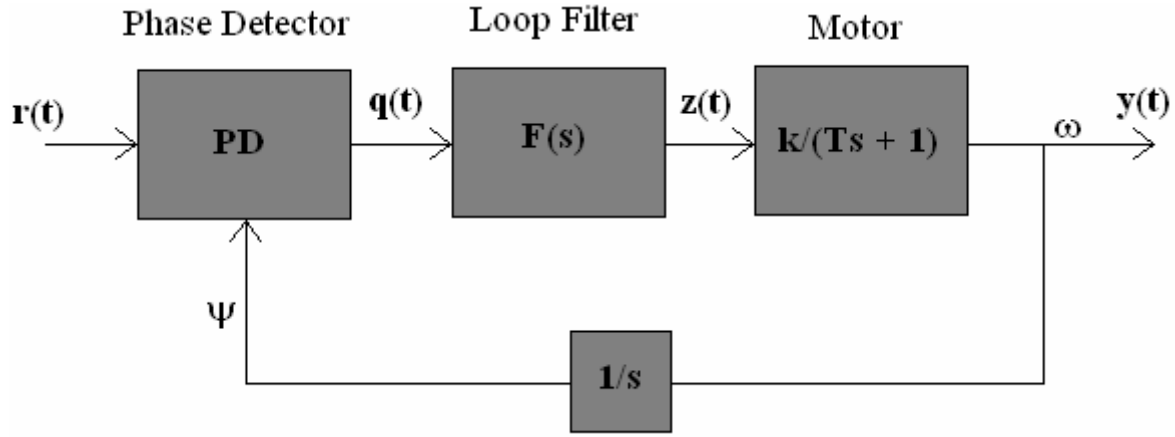


Figure 15 PLL implementation for regulating engine speed

The reference signal tells the PLL the desired speed (frequency) and phase and as in any other PLL the structure will adjust the engine (VCO) frequency accordingly.

4.5. How can a PLL improve our measurements of rotational speed?

The idea is that the current solution does not make use of all information in the analogue signal. When it is sampled only the peaks are registered and no consideration is taken to the shape of the signal. A PLL works directly with the analogue signal and should be able to extract more information. For example if an abrupt change in velocity occurs between two gears the current method would not notice this change until the next gear passes. A PLL instead, would directly notice that the signal is changing as it is working with the analogue sensor output and would react faster to the change.

Another advantage with the PLL is that the control voltage into the VCO is proportional to the frequency and this signal can be sent directly to the ECU via its analogue ports for an immediate frequency estimation directly from the analogue signal. With the current method the signal needs to pass through a filtering procedure which induces time delays.

4.6. The gradient descent method for design of a more complete PLL

Now we will try a more sophisticated approach when designing the PLL structure. A cost function will be defined and the gradient descent method will be used to minimize it. This cost function will contain information of the error both in phase, frequency and amplitude giving a more complete view of the signal, explained extensively in [2].

4.6.1. The algorithm

Let the input signal be on the following form:

$$r(t) = \sum_{i=0}^{\infty} A_i(t) \sin \psi_i(t) \quad (4.14)$$

Where A_i is the corresponding amplitude and $\psi_i = \int_0^t \omega_i(\tau) d\tau + \phi_i(t)$ the corresponding angle.

Assume now that we are only interested in extracting one sinusoidal component, $y(t)$, of this function, e.g. the main frequency in a noisy measurement.

$$y(t) = A(t) \sin \psi(t) \quad (4.15)$$

\Leftrightarrow

$$y(t) = A(t) \sin\left(\int^t \omega(\tau) d\tau + \varphi(t)\right) \quad (4.16)$$

We also define the parameter vector for our cost function:

$$\theta = [A \quad \omega \quad \varphi] \quad (4.17)$$

Now we want to find a $y(t, \theta)$ such that this cost function is minimized:

$$J(t, \theta) = [r(t) - y(t, \theta)]^2 = e(t, \theta)^2 \quad (4.18)$$

This signal represents the squared difference between the input and the output. If the input only consists of one sine wave the error will always be zero but since we will have disturbances when measuring, the input will consist of more than one frequency and we will have an error.

To minimize this error we will use the gradient descent method. This method states that solving this equation will minimize the cost function with respect to the parameter vector.

$$\frac{d\theta(t)}{dt} = -M \frac{\partial J(t, \theta(t))}{\partial \theta(t)} \quad (4.19)$$

Where M is a diagonal matrix deciding the trade off between convergence rate and stability properties.

It should be said that the gradient descent method can only guarantee stability if the cost function is a quadratic function, i.e. contains only polynomial terms with two as the highest order, and it is not in our case due to our trigonometric functions etc. . However, the stability and uniqueness of the solution in our case are proven in [2]. More explicitly the equations can be written as follows:

$$\begin{bmatrix} \frac{d\hat{A}(t)}{dt} \\ \frac{d\hat{\omega}(t)}{dt} \\ \frac{d\hat{\varphi}(t)}{dt} \end{bmatrix} = - \begin{pmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{pmatrix} \begin{bmatrix} \frac{\partial}{\partial \hat{A}} \left[r(t) - \hat{A}(t) \sin\left(\int^t \hat{\omega}(\tau) d\tau + \hat{\varphi}(t)\right) \right]^2 \\ \frac{\partial}{\partial \hat{\omega}} \left[r(t) - \hat{A}(t) \sin\left(\int^t \hat{\omega}(\tau) d\tau + \hat{\varphi}(t)\right) \right]^2 \\ \frac{\partial}{\partial \hat{\varphi}} \left[r(t) - \hat{A}(t) \sin\left(\int^t \hat{\omega}(\tau) d\tau + \hat{\varphi}(t)\right) \right]^2 \end{bmatrix} \quad (4.20)$$

Performing the derivations and using that $e(t) = r(t) - y(t)$, one gets:

$$\begin{aligned}
\frac{d\hat{A}(t)}{dt} &= 2m_{11}e(t)\sin\left(\int^t \hat{\omega}(\tau)d\tau + \hat{\phi}(t)\right) \\
\frac{d\hat{\omega}(t)}{dt} &= 2m_{22}e(t)\hat{A}(t)\cos\left(\int^t \hat{\omega}(\tau)d\tau + \hat{\phi}(t)\right) \\
\frac{d\hat{\phi}(t)}{dt} &= 2m_{33}e(t)\hat{A}(t)\cos\left(\int^t \hat{\omega}(\tau)d\tau + \hat{\phi}(t)\right)
\end{aligned} \tag{4.21}$$

The second equation has an explicit dependence of the time variable. Since the input $r(t)$ is supposed to be periodic we can expect a periodic solution and to get rid of the explicit time dependency we can replace t by its average value during the period, motivated in [3], $t = m_4$.

Since $\psi = \int^t \omega(\tau)d\tau + \phi(t)$ we can write:

$$\frac{d\hat{\psi}}{dt} = \hat{\omega}(t) + \frac{d\hat{\phi}}{dt} \tag{4.22}$$

With use of (4.21) the dependence of $\phi(t)$ can be removed:

$$\frac{d\psi(t)}{dt} = \hat{\omega}(t) + \frac{m_{33}}{m_{22}m_4} \frac{d\hat{\omega}(t)}{dt} \tag{4.23}$$

The system of equations put in its final form:

$$\begin{aligned}
\frac{d\hat{A}(t)}{dt} &= \mu_1 e(t) \sin\left(\int^t \hat{\omega}(\tau)d\tau + \hat{\phi}(t)\right) \\
\frac{d\hat{\omega}(t)}{dt} &= \mu_2 e(t) \hat{A}(t) \cos\left(\int^t \hat{\omega}(\tau)d\tau + \hat{\phi}(t)\right) \\
\frac{d\hat{\psi}(t)}{dt} &= \hat{\omega}(t) + \mu_3 \frac{d\hat{\omega}(t)}{dt}
\end{aligned} \tag{4.24}$$

Where: $\mu_1 = 2m_{11}$, $\mu_2 = 2m_{22}m_4$, $\mu_3 = \frac{2m_{33}}{m_{22}m_4}$

How the μ -parameters should be chosen is explained in [9] but to name a few properties; μ_1 determines how fast the PLL tracks amplitude changes and μ_2 and μ_3 controls the convergence in phase.

4.6.2. Simulations

The equations described above were implemented in Simulink in order to simulate the system as seen in Fig. 16.

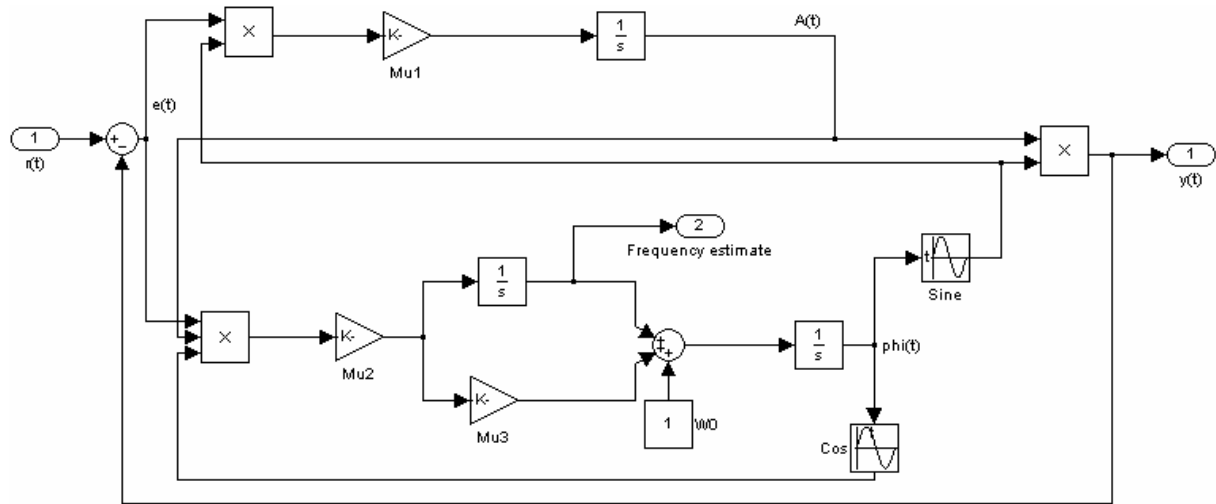


Figure 16 The gradient descent method implemented in Simulink

W_0 in Fig. 16 is used for specifying the frequency of interest and is chosen independently of the μ -parameters. The solution will converge to the frequency closest to W_0 . Usually the lowest frequency component is of interest and W_0 can be set to zero.

The system is simulated with a unitary 30rad/s sine wave with a noise amplitude of 0.1 added.

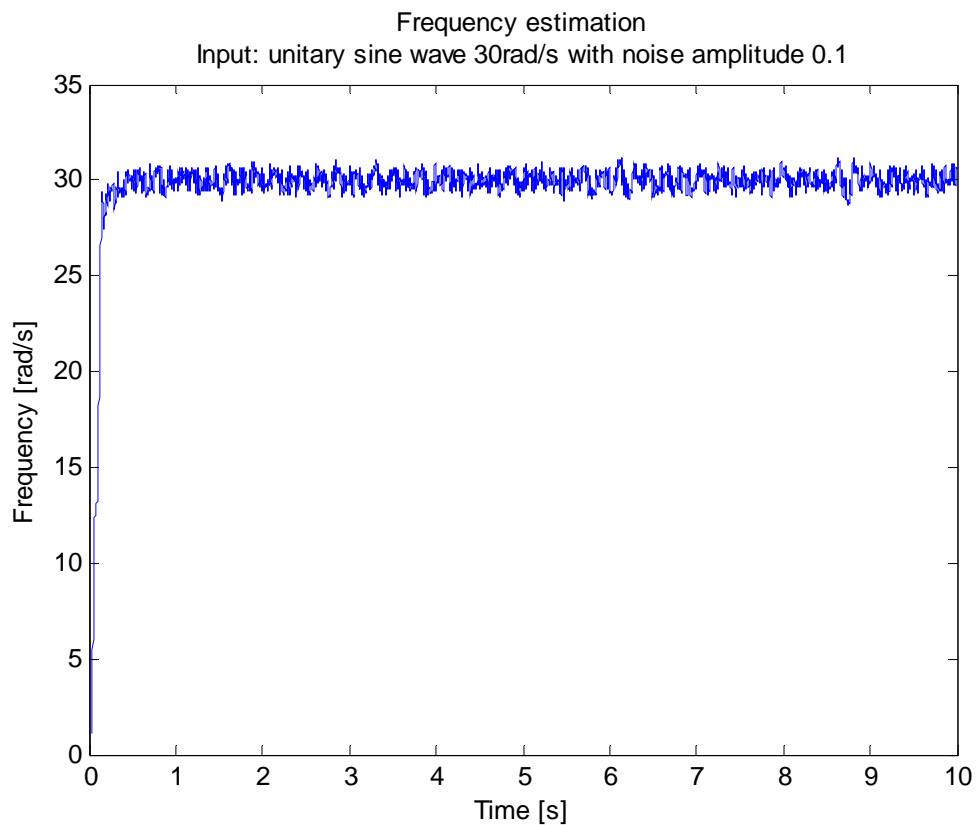


Figure 17 The frequency tracking output

The frequency tracking output in Fig 17 has a rise time of about 0.15s when a step change from 0rad/s to 30rad/s occurs. This time corresponds to only $\frac{3}{4}$ of a period of the input sine wave, whereas it would have taken the current method at least a whole period to track this frequency.

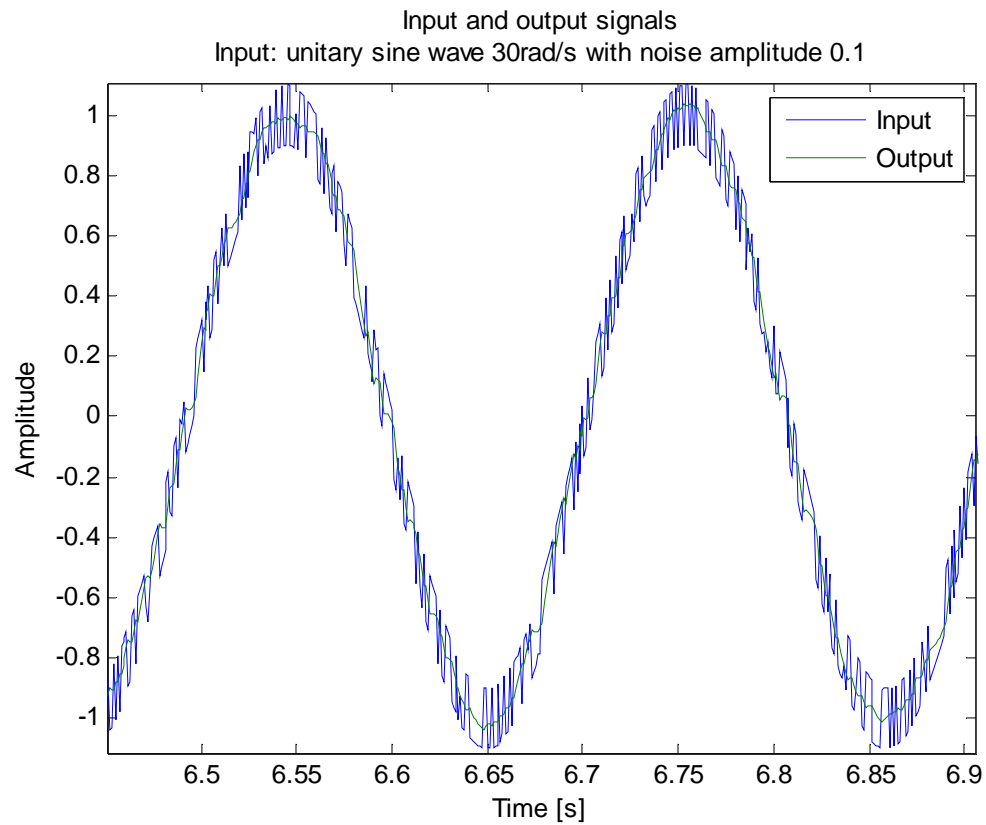


Figure 18 Input and output signals

The output follows the input very precisely and is very robust with respect to the disturbance, which can be seen in Fig. 18 and in the enlarged plot of a peak in Fig 19.

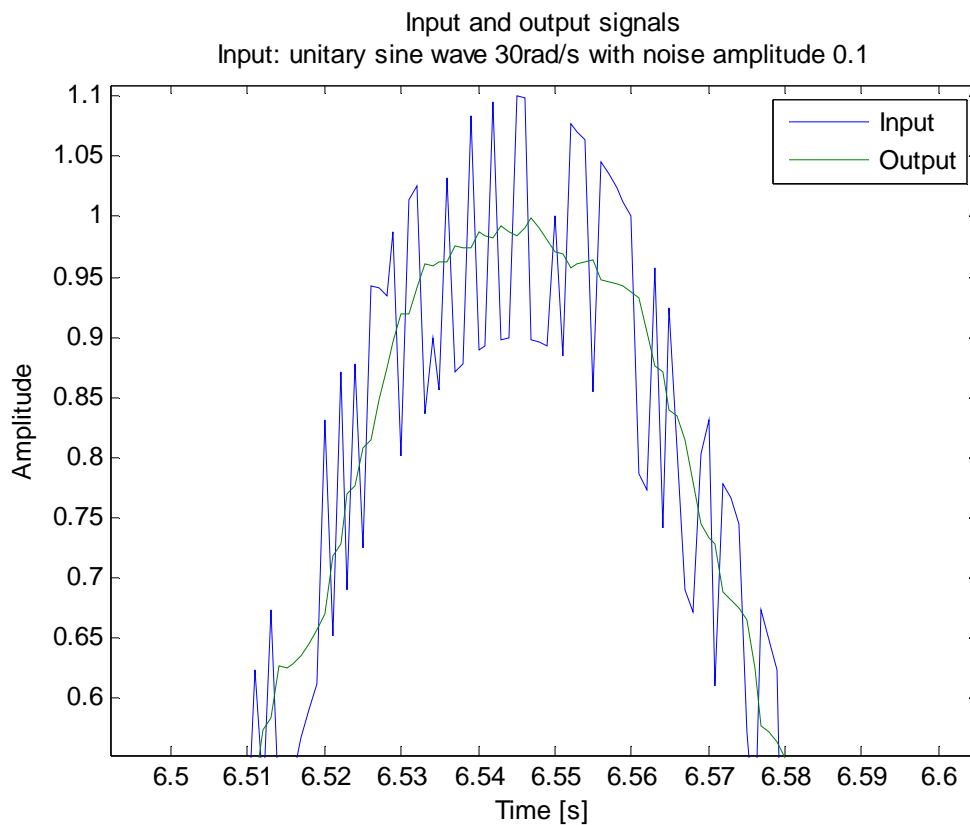


Figure 19 A Zoom of the input and output signals

This was quite an easy test for the PLL since we only had a step change in frequency at zero time and after that the frequency, phase and amplitude were kept constant with only some white noise. Now we will simulate the system with a more challenging and realistic signal. The sine wave will start with a frequency of 10rad/s and will increase its frequency linearly to 60rad/s during the first 5s, at the same time its amplitude will be doubled. This is basically what happens when the vehicle is accelerating, the frequency will of course increase but also the amplitude since faster changes in magnetic reluctance will give a larger sensor output as described earlier. After the first five seconds the frequency will drop quickly to zero for the remaining five seconds. The result is shown in Fig. 20.

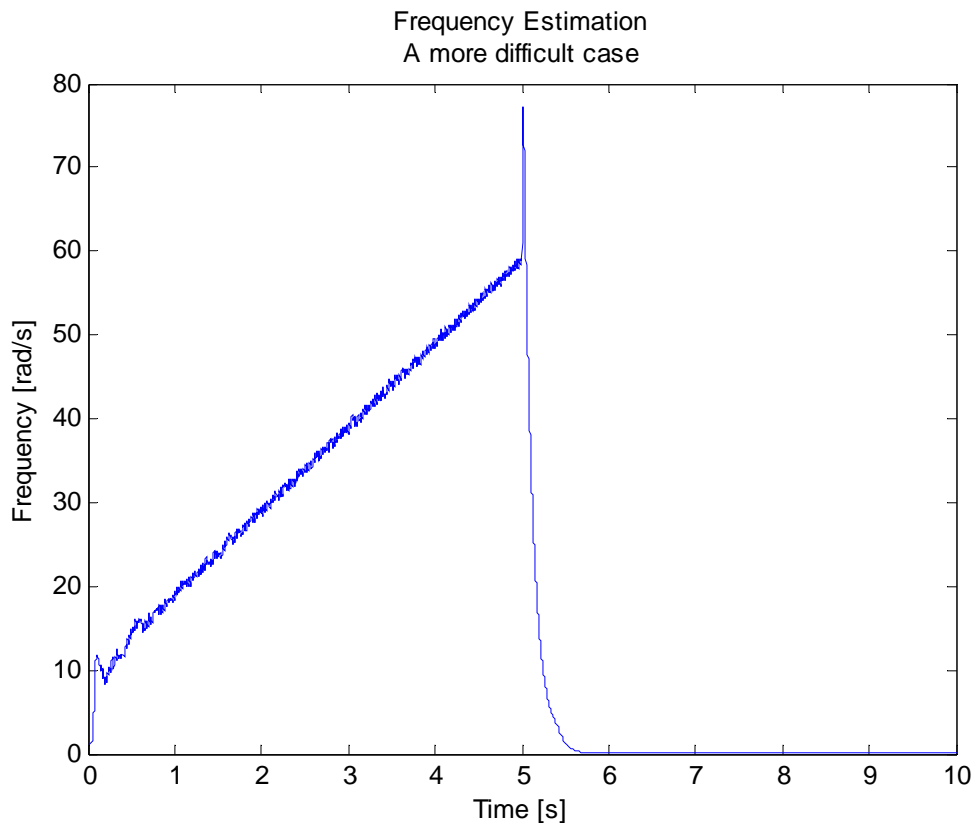


Figure 20 The frequency tracking output

As can be seen the PLL tracks the changing frequency well despite of the change in amplitude. The strange behaviour around 5s is due to the fact that when we change the input from a sine wave to a linear drop in amplitude the signals are not in phase and we have a discontinuous signal at this point. This will of course not happen in reality. Its also interesting to notice that the rise time to 10rad/s is less than 0.1s, only a 1/6 of a period, which mean that it is able to track the signal six times faster than the current method used today.

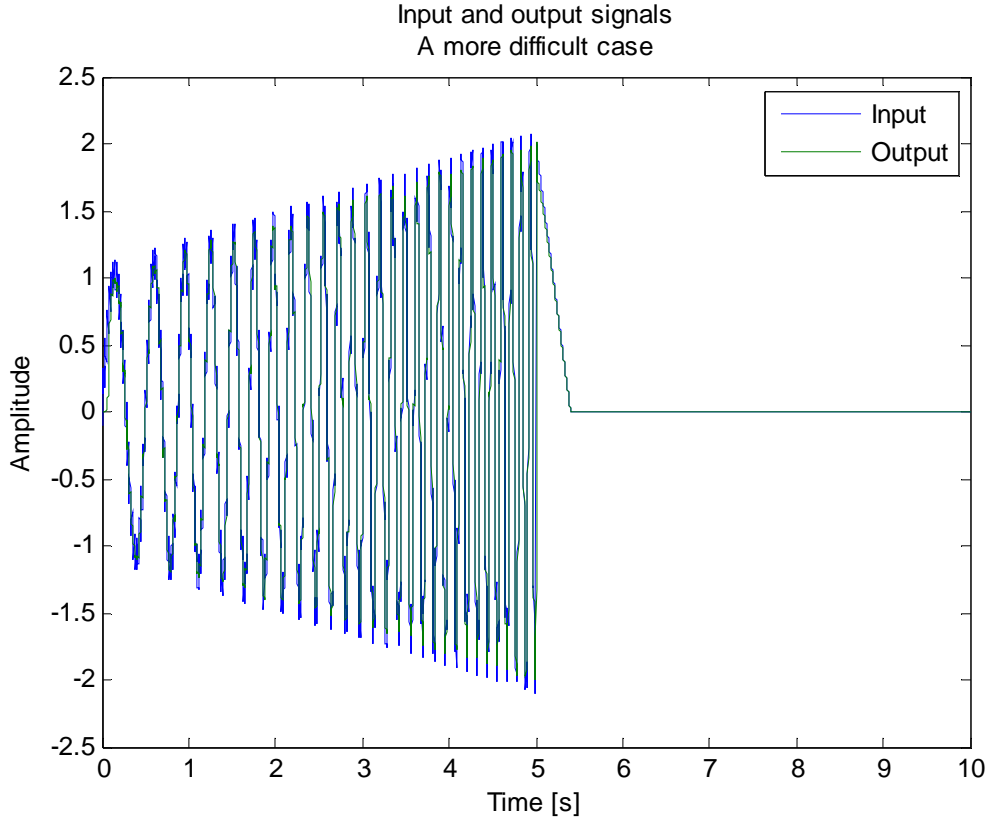


Figure 21 Input and output signals

As before the signals are practically on top of each other and the tracking is excellent.

Because the PLL is especially designed to extract a sine wave component from a periodic signal we could have expected the good results we got, with input signals consisting of only a single frequency sine wave with some noise as done above. The real signal from the sensor is not of course a perfect sine wave and might vary in amplitude and frequency more abruptly than in the previous case.

4.7. The quadrature phase-locked loop (QPLL)

A more robust way of tracking inputs which are less sinusoidal than the previous ones is with a so called quadrature phase-locked loop (QPLL). This method is also based on the previously mentioned gradient descent method but modified to include both the in-phase and quadrature phase amplitudes, see [3]. This is the method that we will apply when trying to track the real sensor output later on.

4.7.1. The algorithm

Let the input be the same as before:

$$r(t) = \sum_{i=0}^{\infty} A_i(t) \sin \psi_i(t) \quad (4.25)$$

Let the output be the sum of an in-phase and a quadrature phase component with the same fundamental frequency:

$$y(t) = k_s \sin \psi(t) + k_c \cos \psi(t) \quad (4.26)$$

Where $\psi(t) = \int_0^t \omega_0 + \Delta\omega(\tau) d\tau$ and ω_0 is the free running frequency of the VCO and $\Delta\omega$ is the difference in frequency between the free running and the estimated one.

Now the parameter vector will consist of the in-phase amplitude, the quadrature amplitude and the frequency difference. This is fundamentally different from the previous method.

$$\theta = [k_s \quad k_c \quad \Delta\omega] \quad (4.27)$$

Following the same procedure with the gradient descent method, more extensively explained in [4], and one arrives at this system of equations, with a similar structure as before:

$$\begin{pmatrix} \dot{k}_s \\ \dot{k}_c \\ \dot{\Delta\omega} \end{pmatrix} = 2e(t) \begin{pmatrix} -\mu_s & 0 & 0 \\ 0 & -\mu_c & 0 \\ 0 & 0 & -\mu_f \end{pmatrix} \begin{pmatrix} -\sin\psi(t) \\ -\cos\psi(t) \\ t(-k_s \cos\psi(t) + -k_c \sin\psi(t)) \end{pmatrix} \quad (4.28)$$

We notice the explicit time dependency in the third equation and with the same motivation as before we will incorporate it into μ_f .

An increased μ_f will make the system converge faster but it will also be less stable. μ_s and μ_c will determine the relationship between the in-phase and quadrature-phase amplitudes.

After some rearranging of these equations the system is put on its final form:

$$\begin{aligned} \dot{k}_s &= 2\mu_s e(t) \sin\psi(t) \\ \dot{k}_c &= 2\mu_c e(t) \cos\psi(t) \\ \dot{\Delta\omega} &= 2\mu_f e(t) [k_s \cos\psi(t) - k_c \sin\psi(t)] \\ \dot{\psi} &= \omega_0 + \Delta\omega \\ y(t) &= k_s \sin\psi(t) + k_c \cos\psi(t) \\ e(t) &= r(t) - y(t) \end{aligned} \quad (4.29)$$

The following system of equation is implemented in Simulink as seen below.

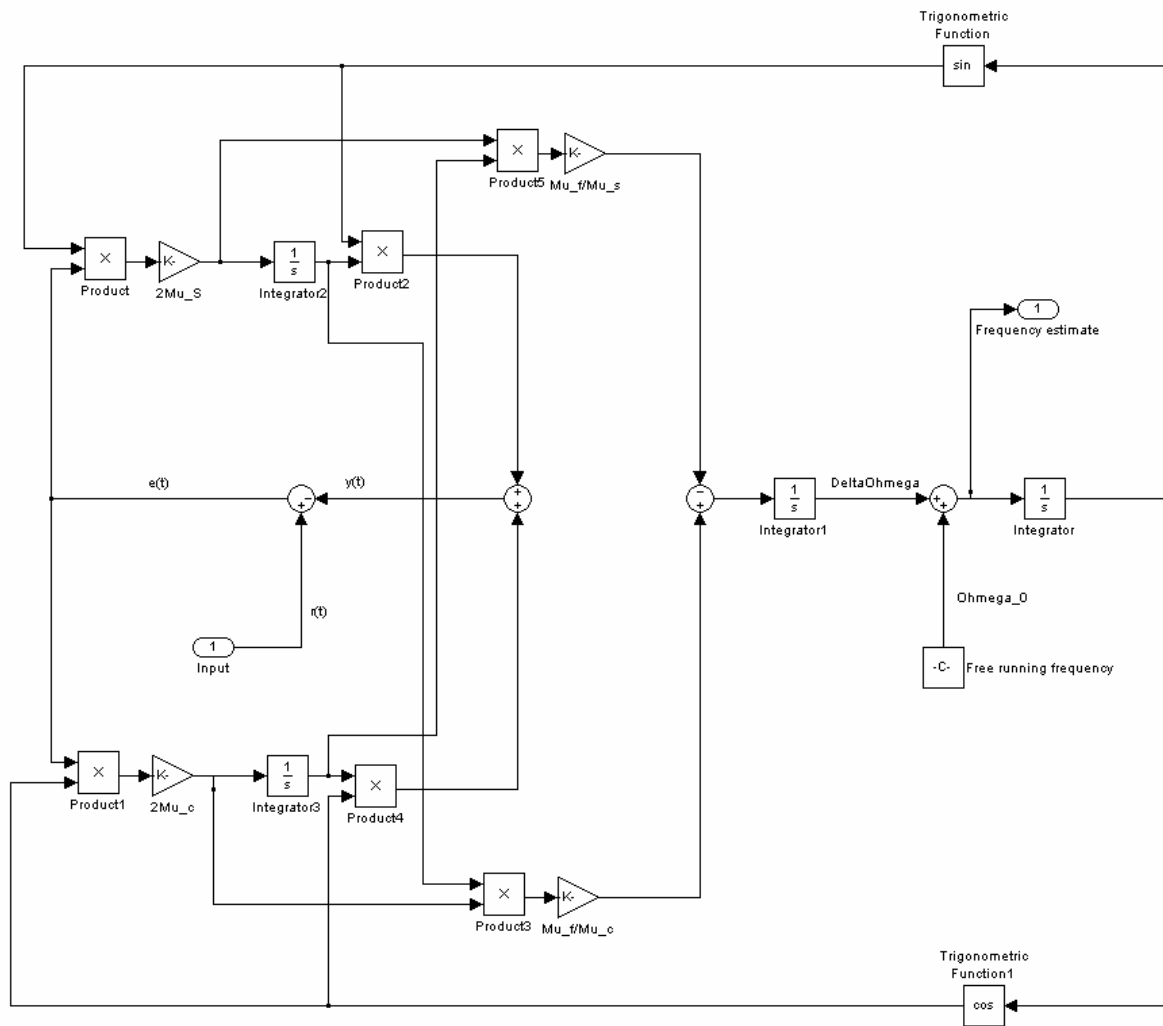


Figure 22 The QPLL implemented in Simulink

5. Implementation of a PLL in hardware

In this chapter we will see how a second order PLL can be implemented in hardware. We will give a short description of how the different parts of a PLL look in hardware and what limitations the PLL have in terms of frequency range etc.

5.1. Necessary components

Now follows a short description of how each part of the PLL can be implemented by using simple circuit blocks. We will not derive the parts mathematically and just simply show how each part looks in order to get a feeling for their complexity and basic structure.

5.1.1. The phase detector

The phase detector's only objective is to multiply two signals together. This is an easy task when simulating a PLL in software but when a multiplication is to be done in hardware there are some constraints on how well this can be done. Normally a building block called a balanced mixer is used. This block is also called a four quadrant multiplier since it can perform multiplications with both negative and positive signals, i.e. it can operate in all four quadrants. Below follows a simple implementation of a balanced mixer:

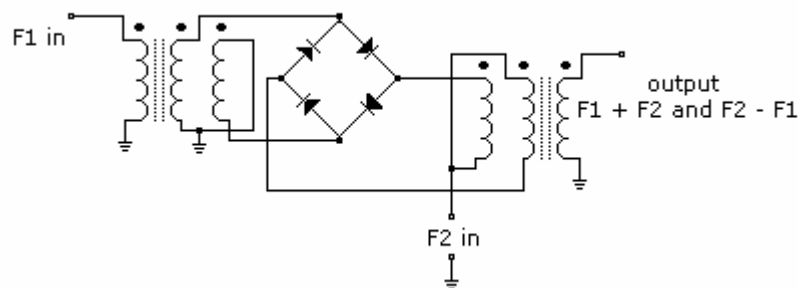


Figure 23 A balanced mixer [11]

We will not go into detail how this mixer works but a more comprehensive description can be found in [6]. The advantages of such a mixer are a relative low loss of signal and a balanced output with no traces of the initial input frequencies, see [11].

5.1.2. The voltage controlled oscillator (VCO)

The implementation of the VCO also requires some circuitry as seen below:

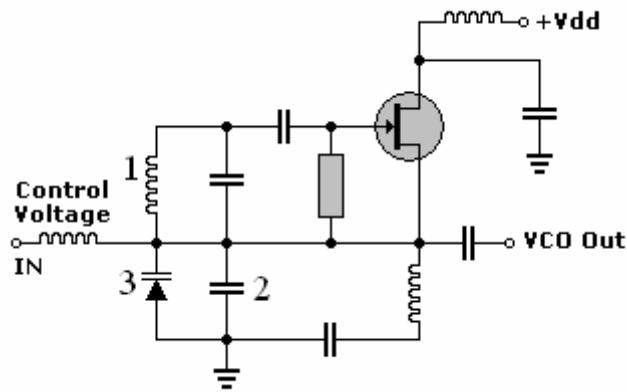


Figure 24 The VCO [15]

Again without going into any detail we can just mention that coil (1) and capacitor (2) are used for deciding the free running frequency of the VCO. (3) Is a varicap or varactor diode which alters its effective capacitance depending of the voltage placed upon it, this will change the resonance frequency accordingly and we will have an output frequency that is proportional to the input voltage.

5.1.3. The PI-controller/Loop filter

The PI-controller is the most basic circuit component in the PLL consisting of only two op-amps, the first configured to be the proportional part and the second to be the integral action[12].

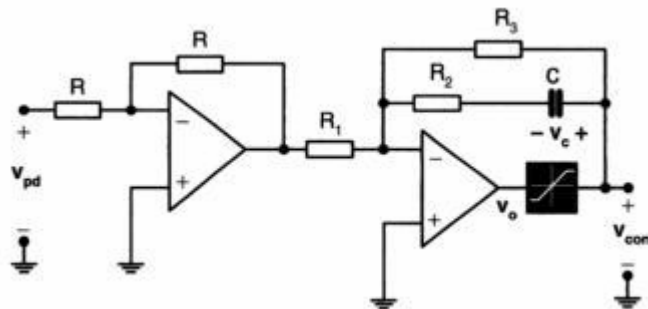


Figure 25 The PI-controller [12]

5.2. The LM565C Phase-locked loop integrated circuit

As seen above, implementing a PLL in hardware requires quite a lot of circuitry but there also exist integrated circuits containing all these components on one chip. In our experiments we will use the LM565C from the manufacturer National Semiconductor.

The LM565C is a general purpose second order phase-locked loop with highly linear VCO and a balanced mixer. The free running frequency of the VCO is decided with an external resistor and capacitor and the loop may also be broken between the VCO and the phase detector in order to be able to insert a frequency multiplying circuit if desired.

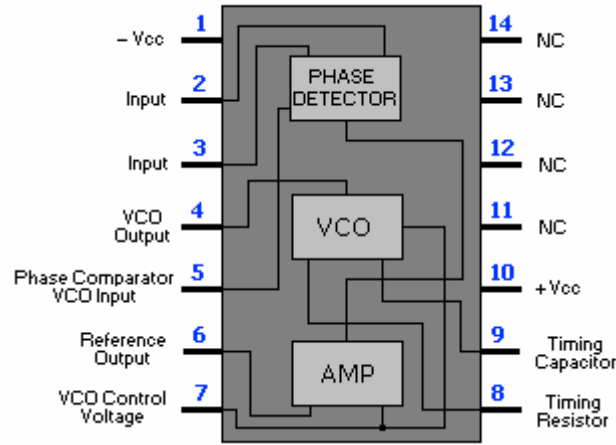


Figure 26 The LM565C pin-out [13]

The sensor output enter the PLL on Pin 2 and the VCO output on Pin 5, these signals are multiplied and sent to the AMP circuit which filters out the high frequency component and amplifies the signal. This signal is then sent into the VCO which generates a signal with a corresponding frequency and this signal is then sent back to the phase detector as one of its inputs. Our signal of interest, the frequency estimate, is extracted after the AMF circuit and can be read on pin 7. A complete description of this circuit is available in its specification [13].

5.2.1. Limitations and tuning

A hardware-PLL will always have limitations that do not exist in software. The capacitors in the filters have limitations on their energy discharging speed and the VCO can not change its frequency instantly etc. These facts put new limitations on the behaviour of the PLL that are not so easy to analyse. Luckily the manufacturers of PLL circuits provide practical guidelines of the real performance of their PLL.

-Free running frequency

This is the frequency of the VCO when it is not excited by an input:

$$f_0 = \frac{0.3}{R_0 C_0} \quad (5.1)$$

Here R_0 and C_0 are the tuning resistor and capacitor respectively (Pin 8 and Pin 9).

In our case since we are primarily interested in low frequencies, i.e. frequencies around 100Hz, we will set the free running frequency around that point. With $R_0 = 26k\Omega$ and $C_0 = 104nF$ we get $f_0 = 110Hz$.

-Hold-in range

This is the frequency range in which the PLL is able to remain locked to the signal if initially at lock.

$$f_h = f_0 \pm \frac{8f_0}{V_c} \quad (5.2)$$

Here V_c is the applied voltage.

Seeing as we want to be able to track as low frequencies as possible we choose $V_c = 8V$ in order to get a frequency range of 0Hz – 220Hz.

- Capture range

The capture range is the frequency range in which the PLL can acquire lock if not locked initially. This range is always smaller than the hold in range but can not be described easily with an equation.

$$f_c < f_h$$

5.2.2. Comments

The hold-in range and the capture range might look like they should depend on the same type of parameters but these two concepts are very different. The hold-in range gives us an interval where we can track the signal if initially at lock. Since the loop will be locked the phase detector signal will be a DC-signal (zero frequency difference) and thus not limited by the loop filter. So the hold-in range is essentially only decided on how much the VCO can deviate from its free running frequency and not by the loop-filter. The capture range on the other hand is dependant on the loop-filter since a too large initial frequency difference will be filtered out as well and not just the high frequency component.

It is obvious that a hardware-PLL has much greater limitations than the corresponding software model. A big difference from a software-PLL is the limited hold-in range. In hardware we will only be able to lock in an interval close to the free running frequency decided by the applied voltage, whereas in software we have an infinite hold-in range. Another difference is that it is possible to adaptively tune the parameters for a PLL in software, however not done here, this is much harder to do in hardware and not even possible for some parameters since they are already integrated in the circuit.

Since the hardware-PLL has a limited interval in which it can operate it can never completely replace the current method for estimating rotational speed since this frequency is in the interval 0Hz – 4500Hz. The idea is to only use the information from the PLL at frequencies near 100Hz and otherwise keep using the current method. This is not a big drawback because the current method works quite well at frequencies above 100Hz since it is guaranteed to get a new gear reading every sample interval. Alternatively several differently tuned hardware-PLL could be used with a switch between them.

6. The QPLL – Machine testing

Here we will present the results from when the QPLL implemented in software was tested in a real wheel loader. The results are compared to the estimation of the current method used in the ECU today.

6.1. Previous results

Prior to the machine testing the QPLL was tested in a test bench. In these tests the QPLL seemed very promising and it was decided to test it with data from a wheel loader. The results from the test bench will not be presented in this report since the results from the machine testing is more relevant.

6.2. Signal processing

6.2.1. The current ECU estimation

A model of the current method of estimating the velocity was created in Matlab. In this model all time delays caused by the filters were neglected, which can be of importance and delay the signal of up to 25ms. The exact time delay of these filters are difficult to calculate and by neglecting their influence we can ensure that our model of the current method will be better than the one actually being used, giving the QPLL a harder target to beat. Otherwise our model acts in exactly the same way as the current method previously described, reading the number of gear passages every 10ms and calculating the velocity. If no gears have passed in that interval it uses the same algorithm as the current method used in the ECU today.

6.2.2. The QPLL estimation

The QPLL is left to work with the unfiltered analogue signal. Even though the QPLL sends out an analogue signal, when presenting the results, this signal will be sampled every 10ms to mimic the behaviour of what would have happened in the ECU.

6.3. Results from machine testing

These results come from tests performed in a L150F series wheel-loader, which is of medium size and latest model. The PLL method (QPLL) is compared to the current ECU estimation and in the enlarged plots the gear passages are also marked. Since these tests were done in a real machine no high resolution encoder was available to serve as a reference signal. But zero velocities can be estimated by looking at the time of the last gear passage and at high velocities the current ECU estimate is a good reference. Both signals were sampled at 20 μ s.

6.3.1. A hard brake

A normal driving situation for a wheel loader is driving into a pile of gravel to fill its bucket. When the wheel loader hits the pile its velocity will drop to zero very quickly and the following simulation aims to copy this situation illustrated in Fig. 27 and Fig. 28.

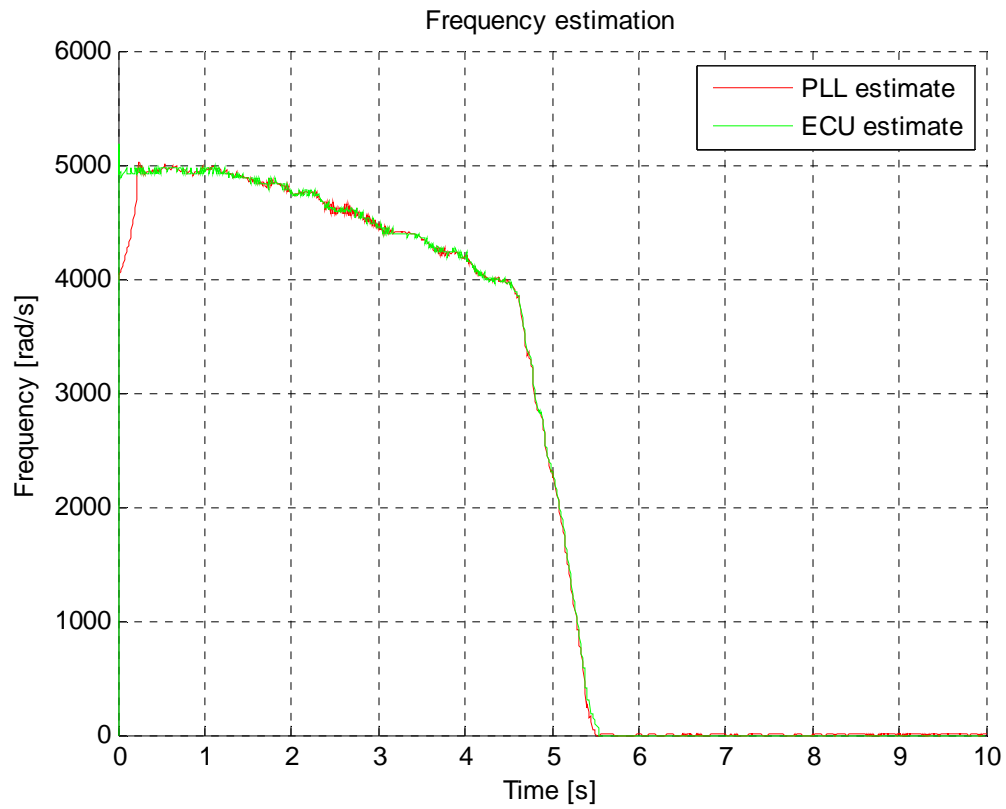


Figure 27 A hard brake to zero velocity

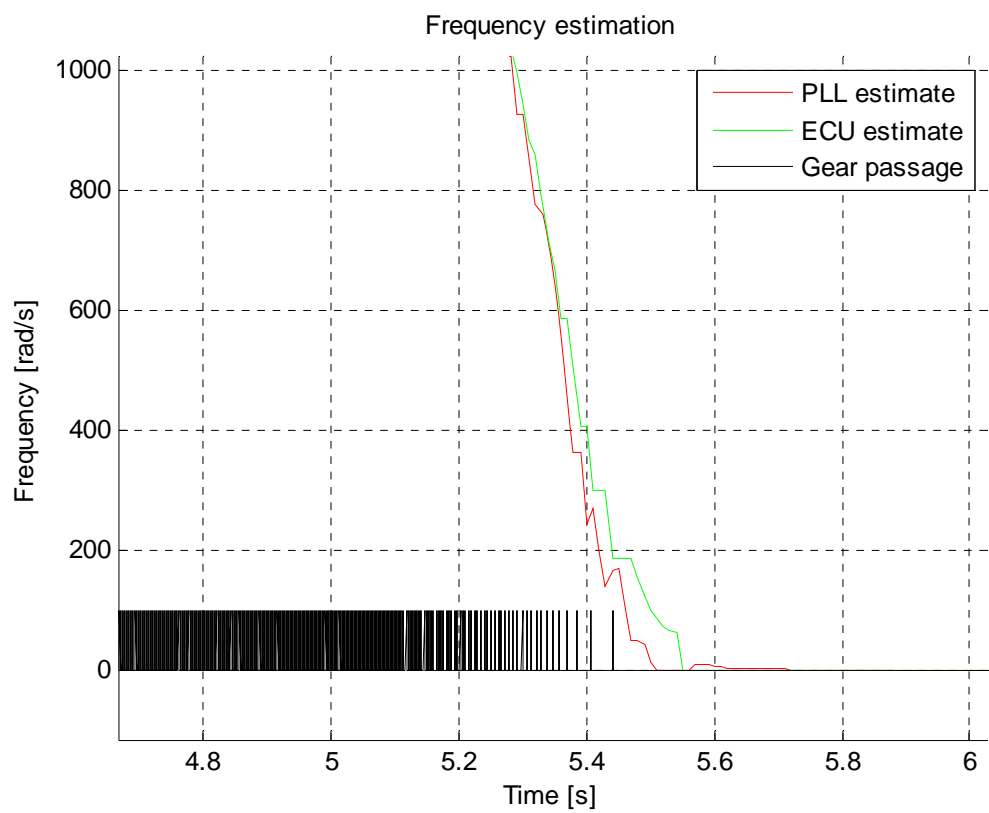


Figure 28 Enlarged plot

It is clear that the PLL estimate and the current method are very similar, especially at high velocities. The PLL has no problem tracking the signal even at frequency changes as high as 4000rad/s^2 occurring around 5s. When detecting zero velocity the PLL is almost twice as fast when measuring from the last gear passage, which is a good result seeing as the filter delays in the current method would make this difference even larger. The free running frequency of the VCO is set to 4000rad/s , a difference of 1000rad/s from the real initial value. This large initial frequency difference is no problem for the PLL that only experiences a limited hold in time before reaching lock at 0.3s.

6.3.2. A Hard brake and change of direction

This test was done by driving in one direction and then putting the gear in reverse. The torque converter will absorb the braking energy and eventually the vehicle will change its direction. Since our inductive sensor can not detect the direction of the rotating gear wheel it will only look like the velocity is rising again after the reaching zero velocity.

Fig 29 – Fig 34 show three different test cases for different retardations and their corresponding enlarged plots then the zero velocity occurs.

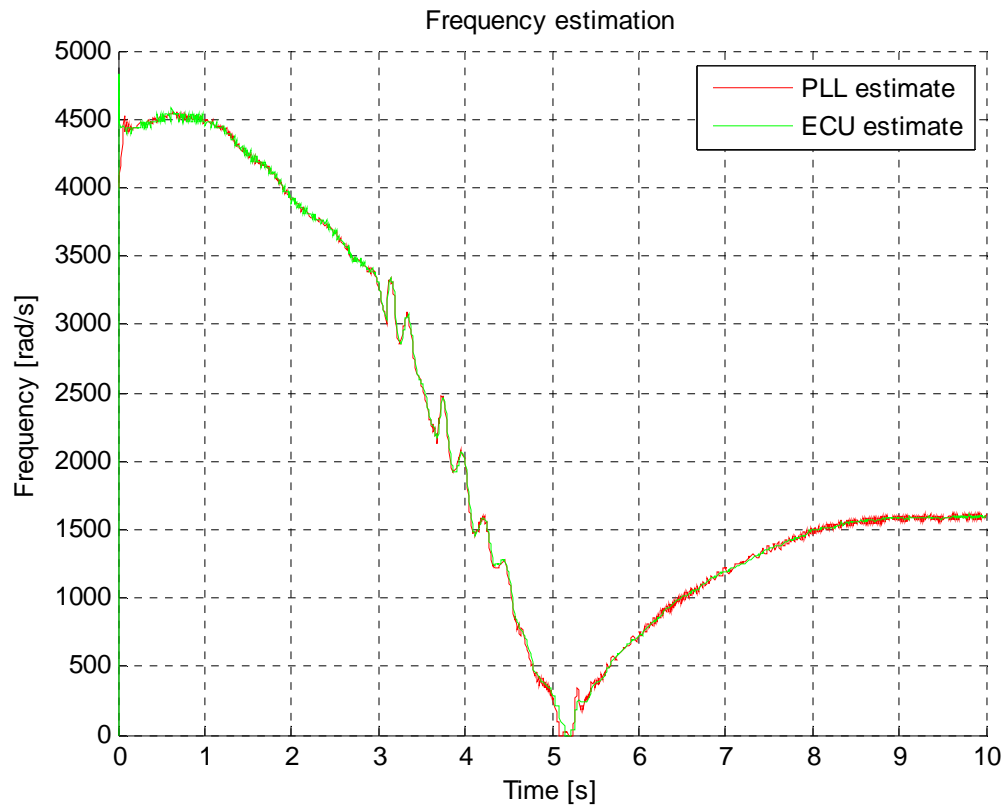


Figure 29 Retardation: 1500rad/s^2

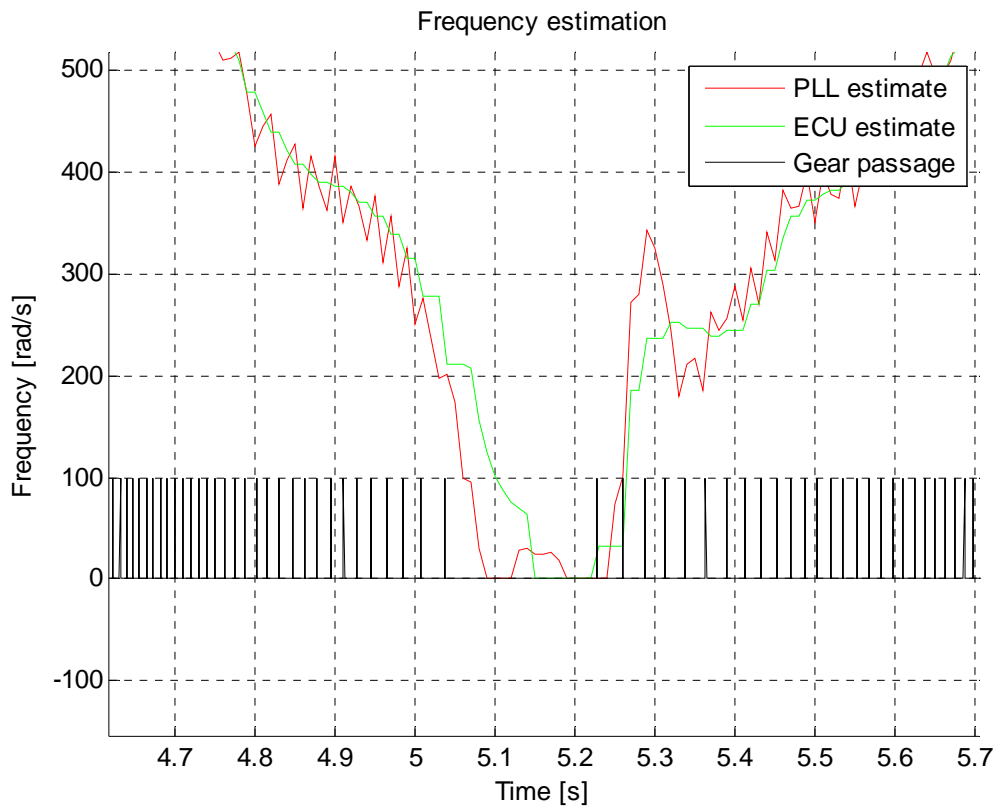


Figure 30 Enlarged plot

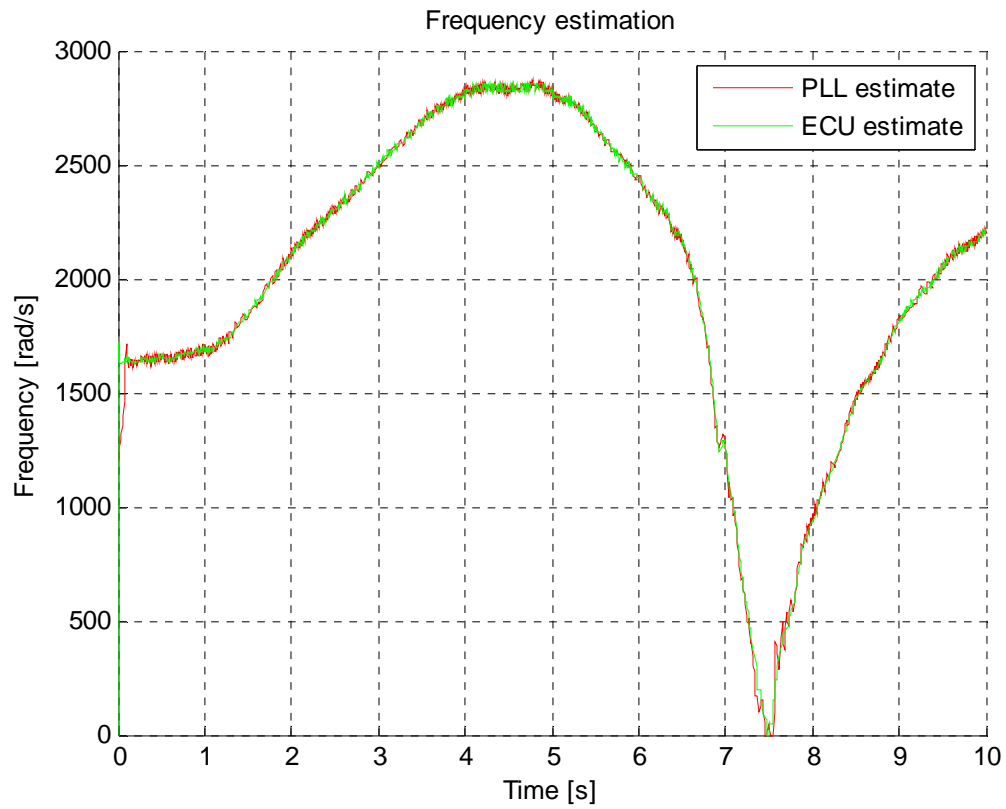


Figure 31 Retardation: 2200rad/s^2

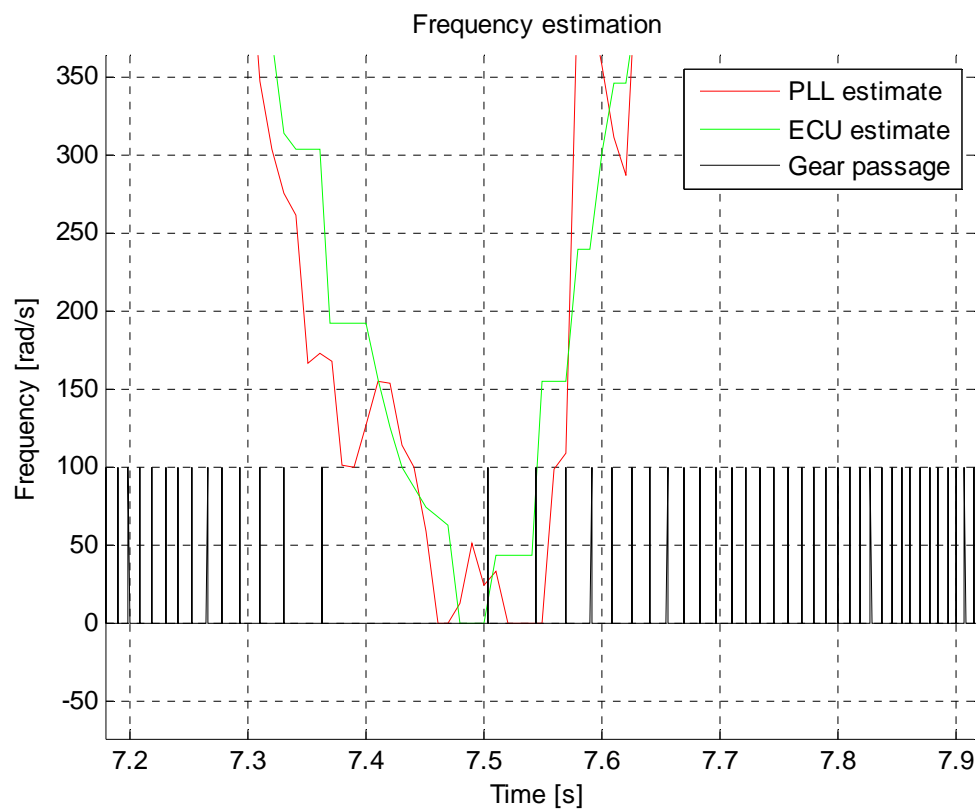


Figure 32 Enlarged plot

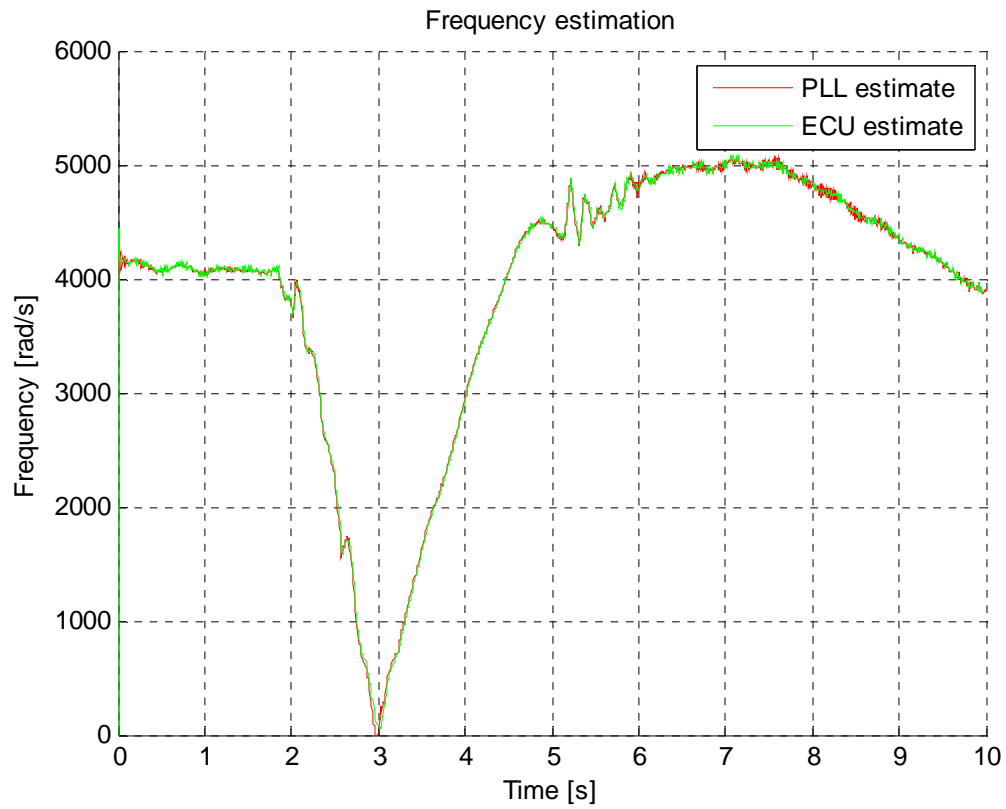


Figure 33 Retardation: 4000rad/s^2

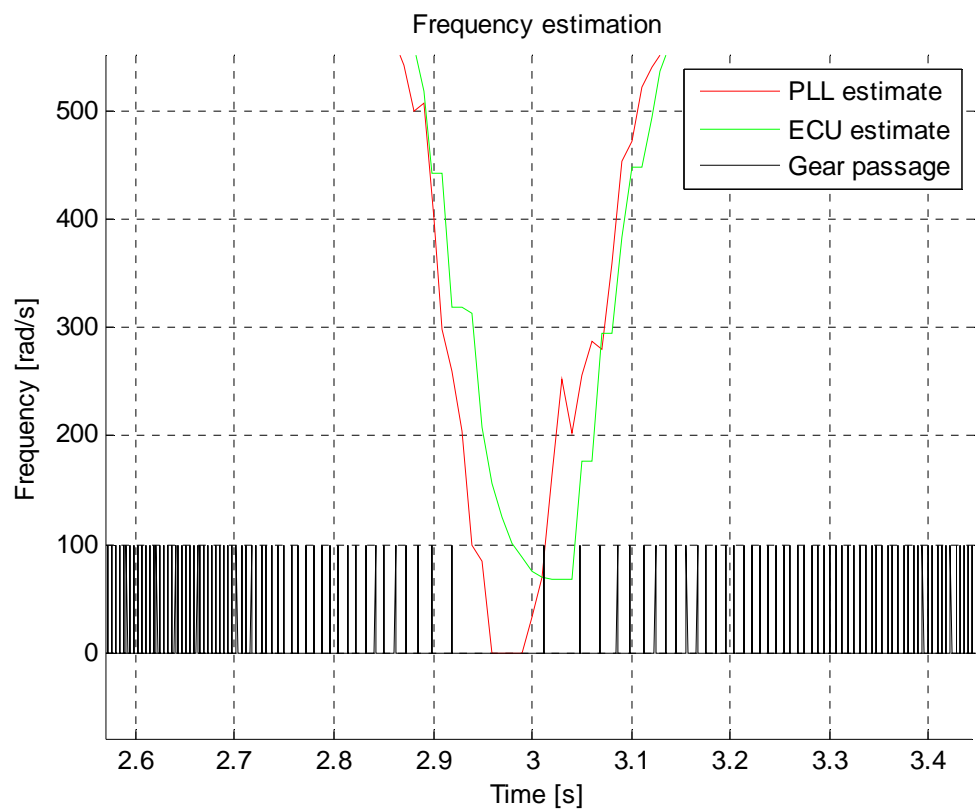


Figure 34 Enlarged plot

We see that the PLL estimate always manages to detect zero velocity faster than the current method being used and in the last test case, Fig 34, with the largest retardation the current method even fails to detect that a zero velocity has occurred. Drawbacks with the PLL are that its estimate is more oscillatory and its derivative can sometimes be large. The signal could be smoothed by a low-pass filter or a rate limiter but this would introduce some delays.

6.3.3. Periodic change in velocity

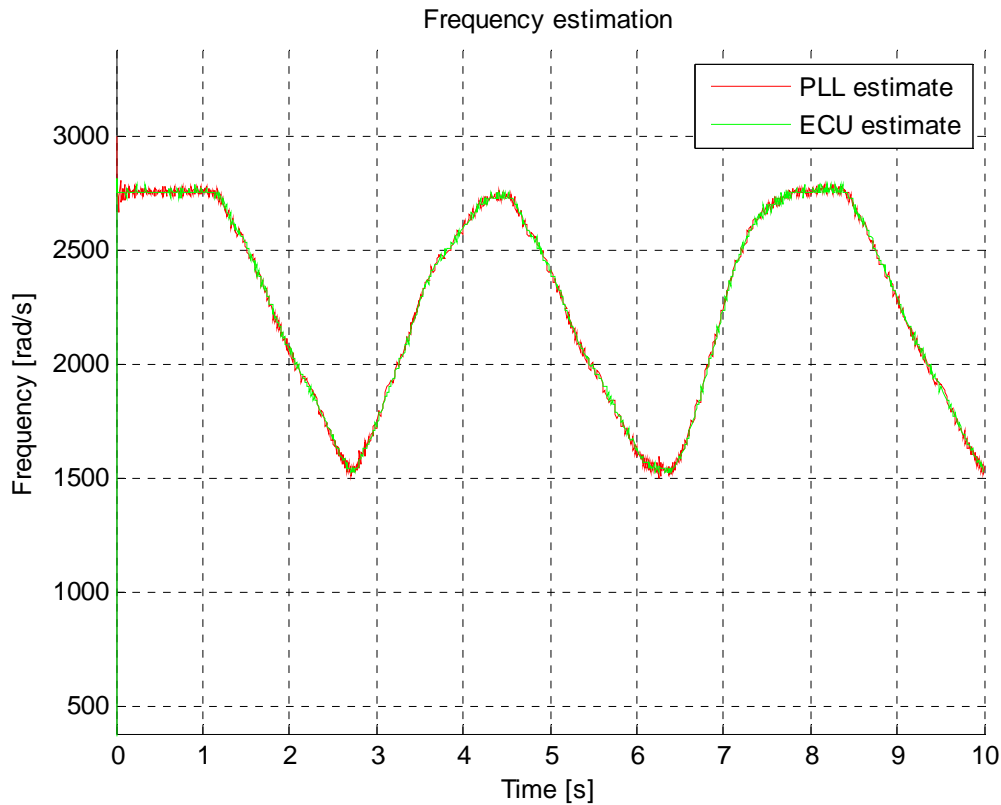


Figure 35 Periodic change in velocity, 2700rad/s – 1500rad/s

For periodic changes in velocity at higher frequencies both the PLL and the current method manage to track the signal without any problems as seen in Fig. 35.

6.3.4. A constant very low velocity

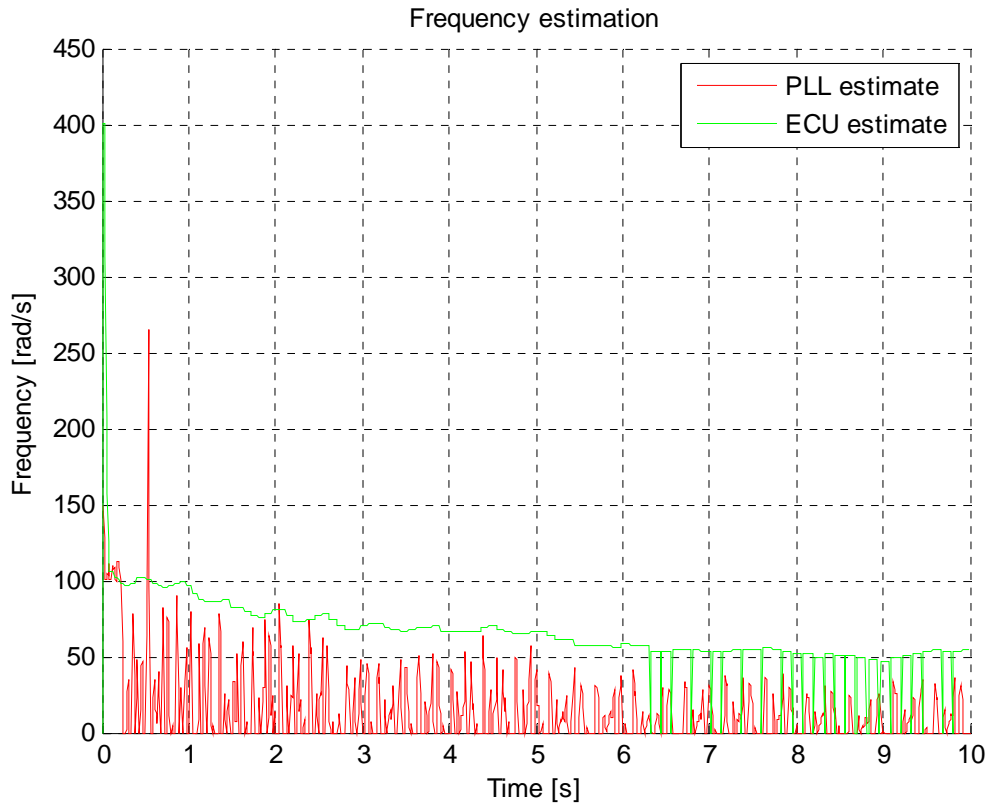


Figure 36 A constant very low velocity

When the velocity is at a constant very low value the PLL does not manage to lock at all. The drop out occurs at 100rad/s which corresponds to 16Hz. As mentioned earlier the sensor are only specified to work above 20Hz and below this point we start to get a noisy signal with a low amplitude and clearly this affects the performance of the PLL. The current method has some hysteresis in it and is therefore less sensitive and manages to track the signal further down to 60rad/s or 10Hz.

7. The PLL in hardware – Results from a test bench

We will now present the results from our hardware-PLL. These tests were done in a test bench with similar working conditions as in a wheel loader. We will especially investigate the limits that exist in the hardware and in which interval we can operate.

7.1. The test bench

The test bench consists of a stepping motor that drives a gear wheel and an inductive sensor is mounted on the side in a similar fashion as in a real gearbox. To get an accurate reference signal a high resolution encoder wheel is mounted on the side of the axis as seen in Fig. 37.

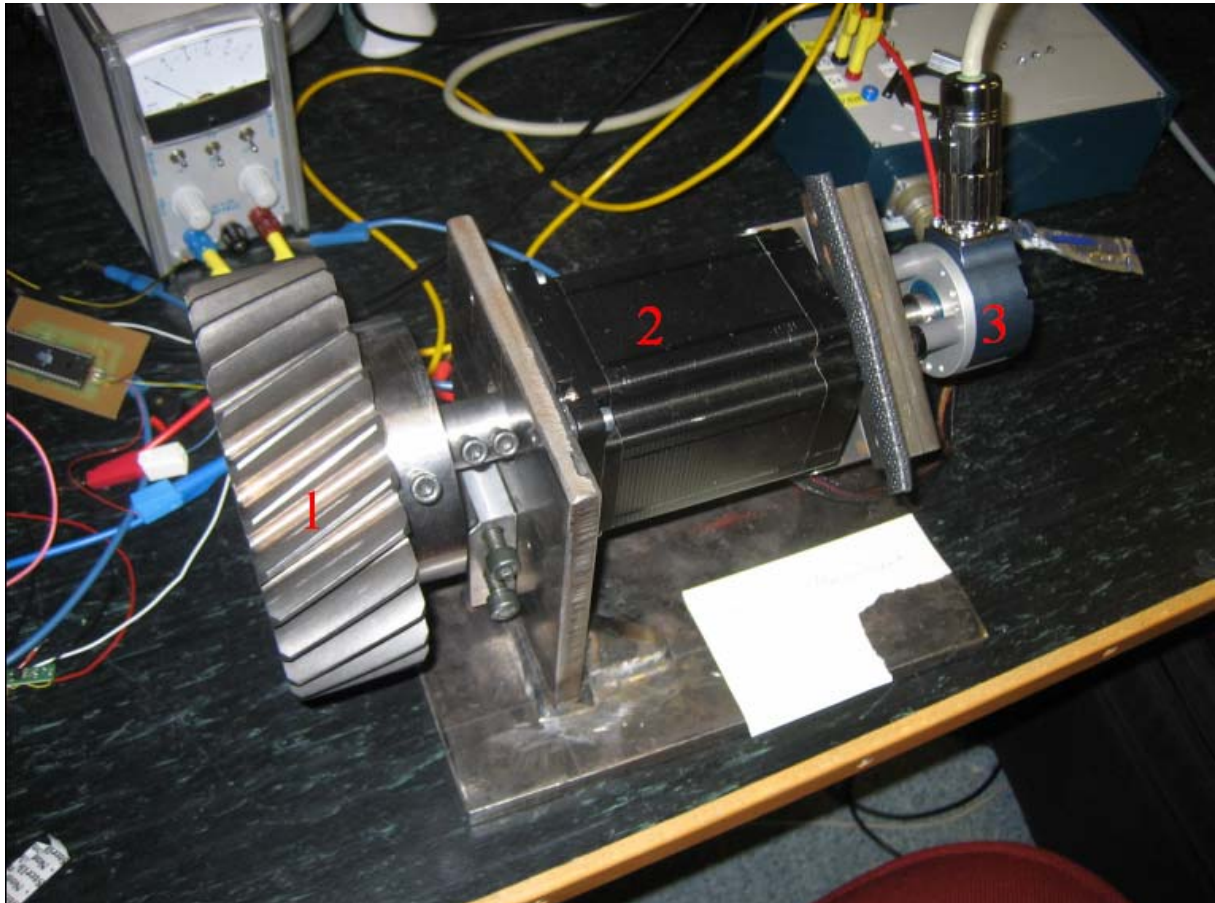


Figure 37 Photo of the test bench. (1) the gear wheel, (2) the stepping motor, (3) the encoder

There are some limitations in the test bench that are hard to deal with. First of all the gear wheel is not completely round giving an over layered sinusoidal signal in the output from the sensor, this also occurs in a real gearbox. Secondly the stepping motor does not rotate the gear wheel evenly resulting in a rather noisy reading from the encoder. The stepping motor also has some limitations in the rate of change in velocity that is possible and when crossing over this limit the stepping motor begins to stall. In some cases at certain frequencies a strange self-oscillating behaviour might also appear giving an extremely noisy measurement. The measurement system used has a built-in anti aliasing filter and all signals are sampled with an interval of $20\mu\text{s}$.

7.2. Signal processing

7.2.1. The current ECU estimation

The current ECU estimation is the same as the one used in the QPLL case in Section 6.2.1. The signals delays caused by the filters are omitted and our model calculates a new velocity estimate every 10ms as described before.

7.2.2. The PLL in hardware

Once again the PLL works directly with the analogue signal but in this case an extra low-pass filter is added to the PLL output to get rid of the noise induced by the unevenly rotating gear wheel.

7.2.3. The Encoder estimate

The encoder is mounted on the axis of the gear wheel and sends out 5000pulses per revolution. This is more than 150 times more than the gear wheel and this should be a good reference signal.

7.3. Results from the test bench

7.3.1. Periodic change in velocity

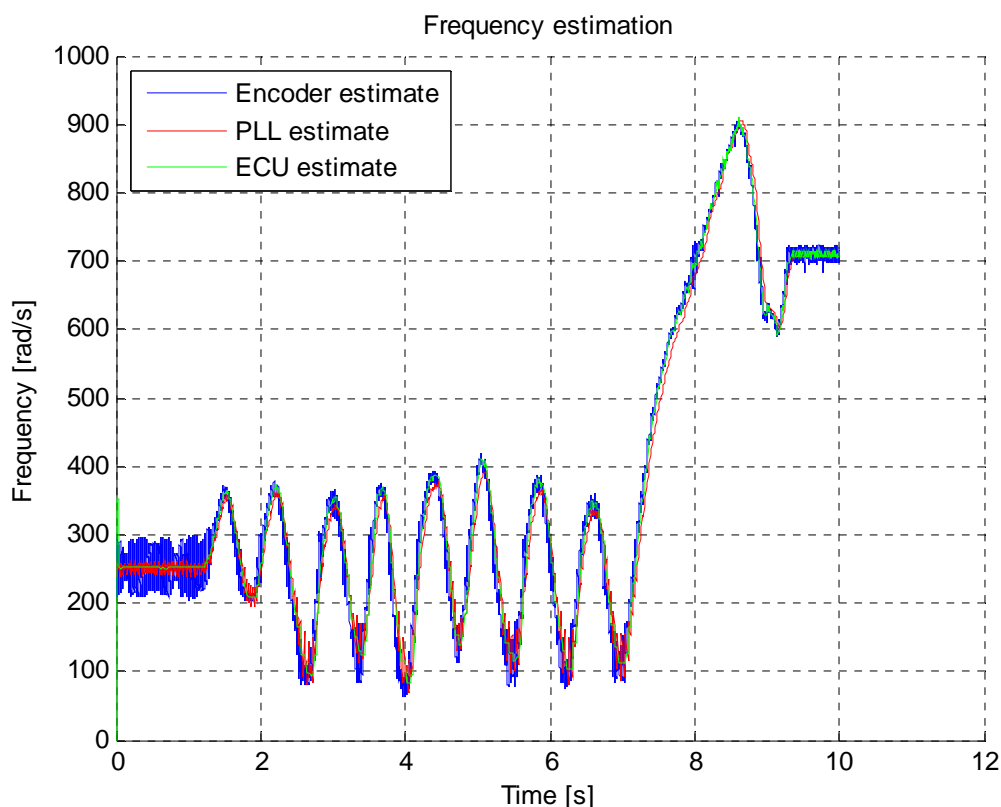


Figure 38 Periodic change in velocity

As seen in Fig. 38 both the PLL and the current method manage to track this quickly varying signal without any difficulties. The noisy encoder estimate reveals that the gear wheel is not

rotating evenly, especially during the first second, but both the PLL and the current method are able to reduce this noise in their measurements.

7.3.2. Step changes in velocity

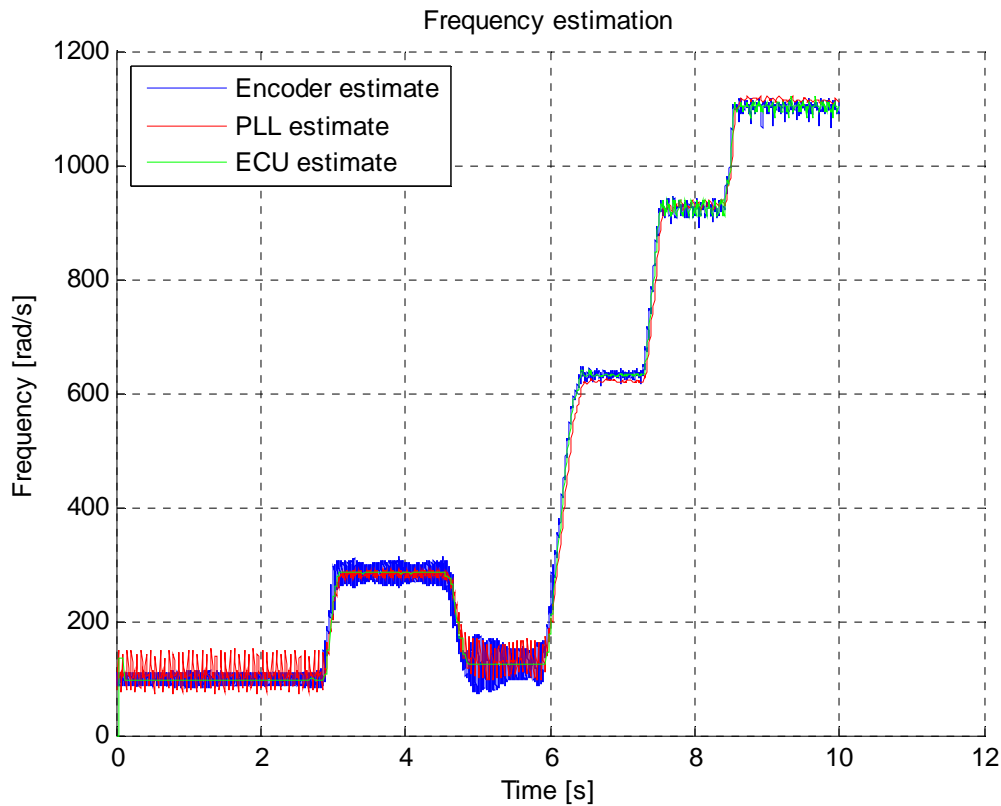


Figure 39 Step changes in velocity

As seen in Fig. 39 the tracking of the PLL is very good, but at low velocities it is noticeable that the PLL is affected by the increased noise and lower sensor output, making its own output noisy as well. It is interesting to see how noise-free the ECU estimate is, especially at low frequencies. This is due to its low frequency algorithm previously described in Chapter 2 which will have a longer period to take the mean value of when few gears are passing by the sensor.

7.3.3. Testing the limits

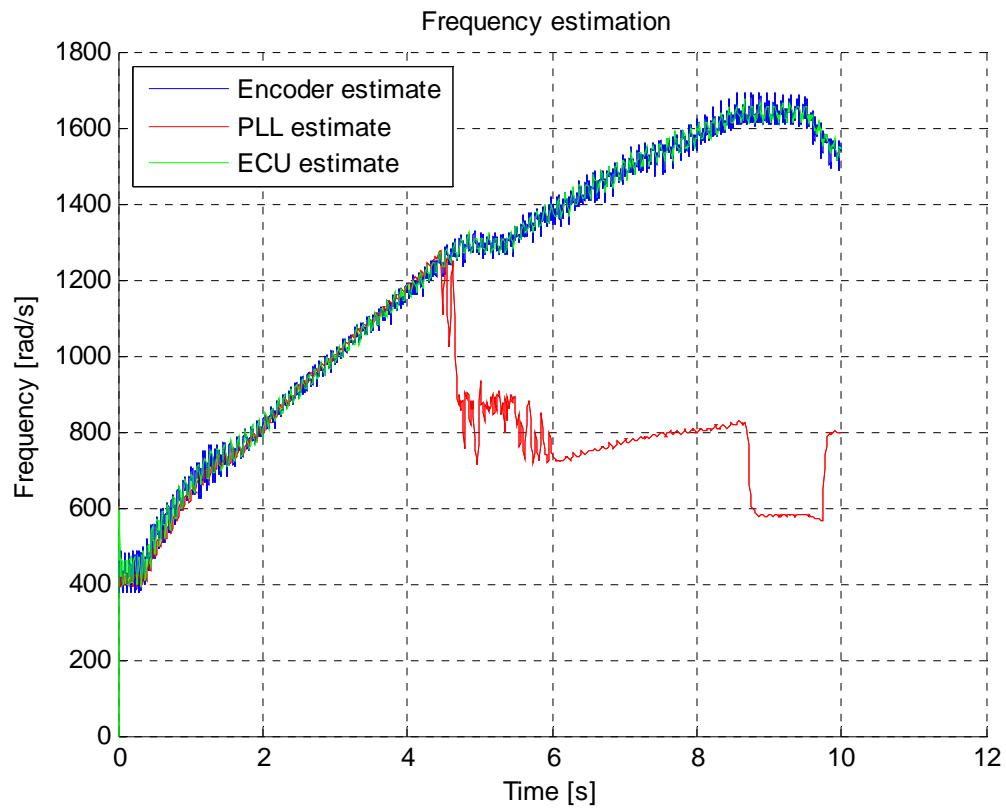


Figure 40 Upper limit frequency drop-out

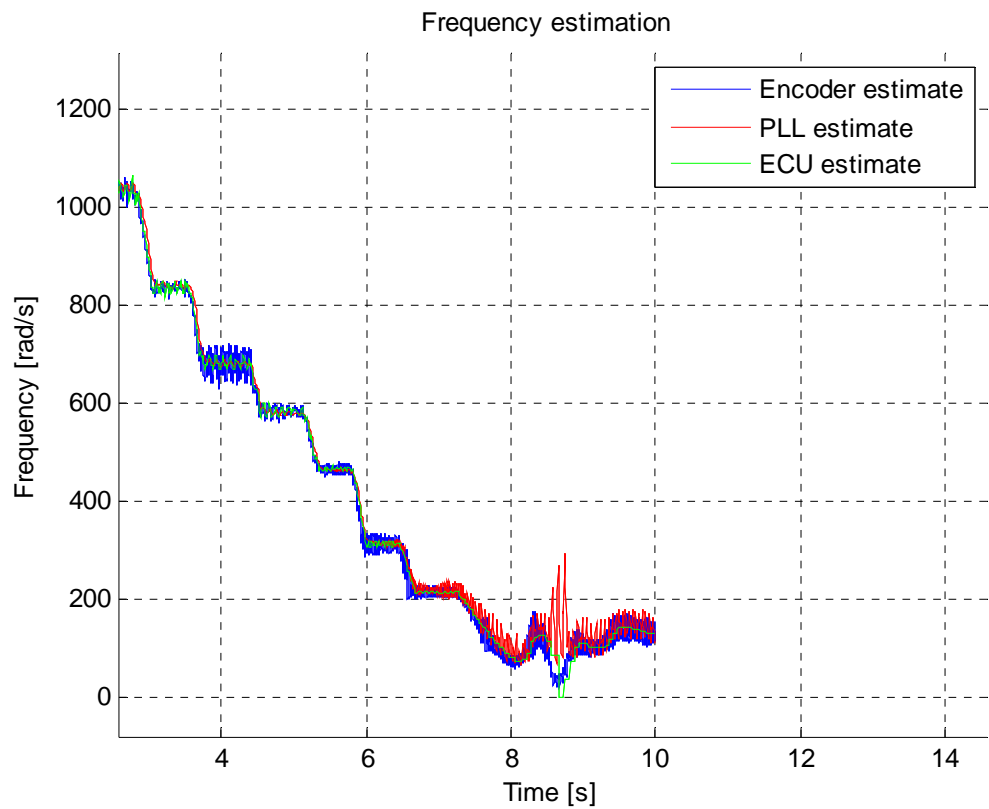


Figure 41 Lower limit frequency drop-out

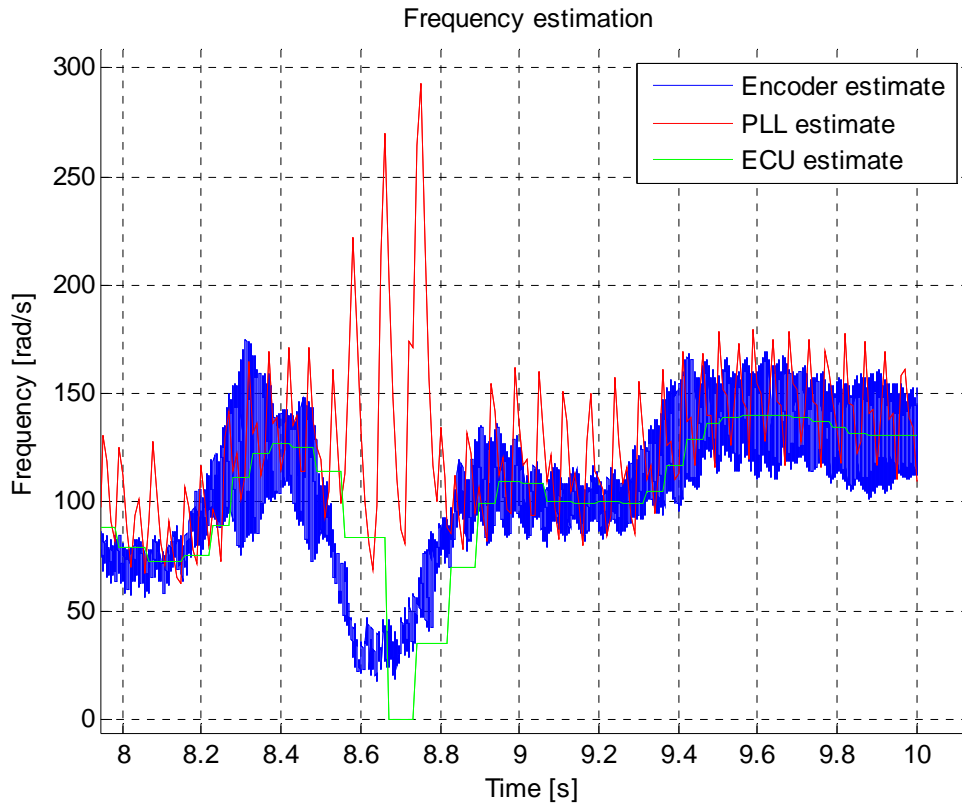


Figure 42 Enlarged plot

In Fig. 40 we see that the PLL loses its lock on the sensor output just over 1200rad/s or 200Hz. This value was predicted in Chapter 5 and its reasons discussed there as well.

We are more surprised to see that the PLL loses its lock for low frequencies as well, in our case at 100rad/s or 16Hz. As discussed earlier the sensor is only specified to work down to 20Hz and under this value the amplitude is low and we saw that below 10Hz the sensor output was very noisy. This in combination with the unevenly rotating motor makes it hard for the PLL to remain at lock. Another known fact is that when the frequency goes to zero the two different frequencies coming out of the phase detector will both converge to zero as well and at a certain point the frequency component which is the sum of the two inputs will get low enough in frequency to pass through the low-pass filter and destroying the lock.

In Fig. 42 we see an enlarged plot of when the PLL loses its lock. Between 8.4s – 8.8s the PLL is not locked to the signal and the behaviour in this interval is difficult to analyze. It is also clear that the current method used in the ECU today also has problems in this interval because at 100rad/s we will only have 16 gear passages per second forcing the ECU to guess the velocity with its low frequency algorithm.

It should also be said that 100rad/s corresponds to a velocity about 0.3km/h or less than 0.1m/s, this is an extremely low velocity for a wheel loader.

7.4. Robustness

As described in Chapter 2 we have large tolerances in the mounting of the sensor and this highly affects the strength of its output. This fact in combination with the wobbling behaviour of the gear wheel requires a high level of robustness in terms of amplitude variations of the input signal into the PLL. Since the PLL is designed to only track variations in phase and frequency it should be relatively robust to amplitude variations.

This robustness was tested by removing the sensor from its housing and manually vary its distance to the gear wheel. The maximum tolerance in the mounting is only $+0.7\text{mm}/-0.4\text{mm}$ but in our tests we could change the distance up to 5mm without losing track of the signal. These results demonstrate a remarkable robustness in terms of amplitude differences.

In our test the frequency of the gear wheel was kept constant and we only varied the sensor distance as seen in Fig. 43 and Fig. 44.

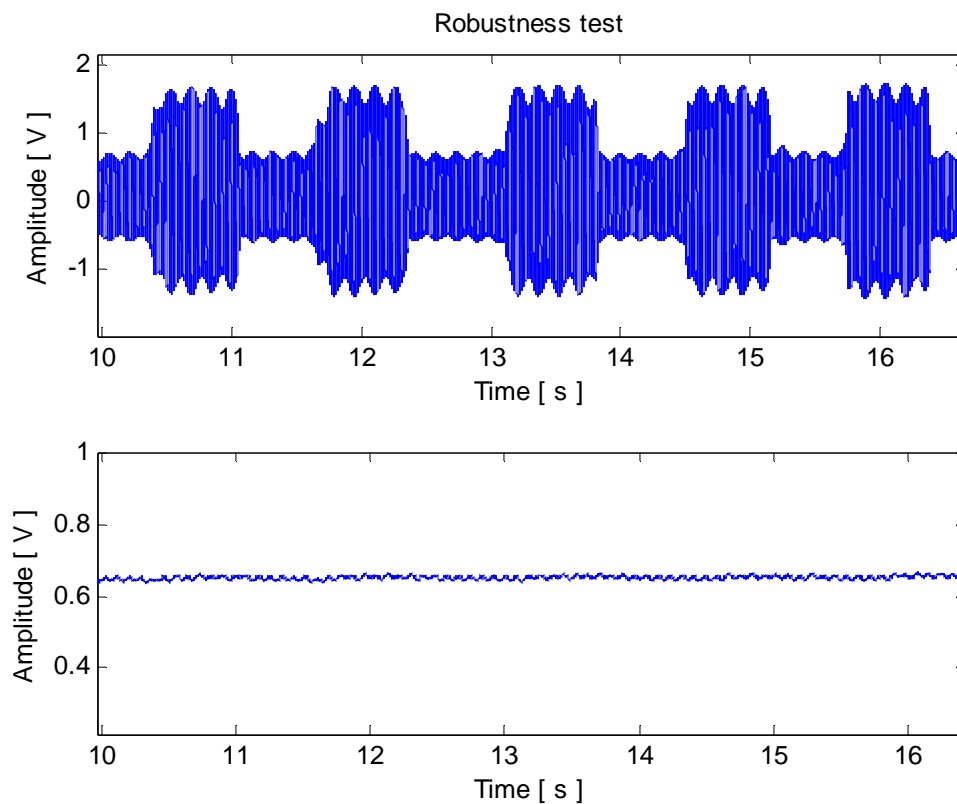


Figure 43 Few number of changes in amplitude

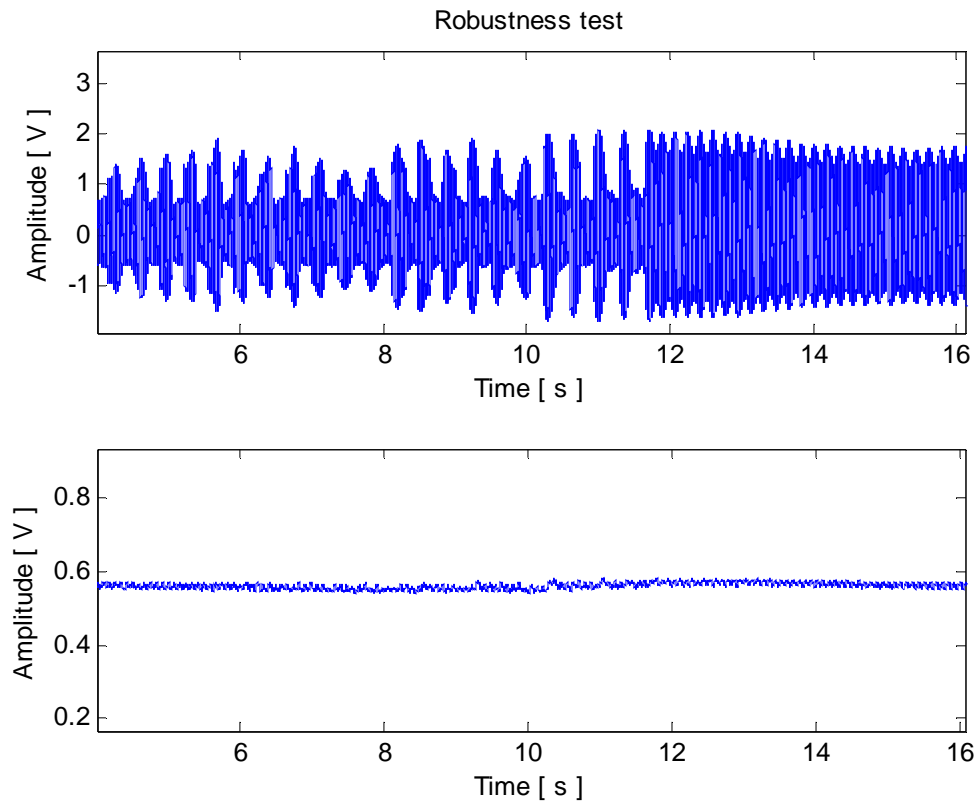


Figure 44 Large number of changes in amplitude

It can be seen that the strength in the sensor output, the first subplot, is varying very abruptly while in the second subplot the PLL retains its voltages level, unaffected by the changes in amplitude. Even in the second case when the changes in amplitude occur frequently the PLL remains unaffected.

8. Discussion and conclusion

8.1. *The QPLL*

As seen in the results chapter both the current method and the QPLL have little problem estimating the velocity at high or constant values and it is only when the velocity is changing quickly that we notice any difference between the estimates. When doing hard brakes to zero, e.g. when driving into a pile to fill a wheel loaders bucket, this difference is of importance. In the best cases the QPLL is almost twice as fast in detecting this stop compared with the current method. A drawback with the QPLL method is its behaviour at constant low velocities, losing its lock at 20Hz while the current method managed to track the velocity down to 10Hz.

As mentioned before the QPLL operates directly with the unfiltered signal while our model of the ECU estimate has some hysteresis in it. Despite this fact the QPLL delivers a relatively noise-free output even at low frequencies where the sensor output is weak. However at constant low velocities ($< 20\text{Hz}$) the QPLL is not able to remain its lock to the signal due to the weak and noisy output.

A real implementation of a QPLL would require some extra hardware e.g. a Field-Programmable Gateway Array (FPGA) or a micro processor of some sort where the QPLL algorithm could be programmed. These extra expenses have to be weighted against the improvements that a QPLL might give us. The QPLL and the current method both have their advantages and disadvantages. The QPLL manages to track fast changes in velocity better than the current method but on the other hand the current method has a better tracking at low frequencies when the velocity is varying slowly. At the moment it does not seem cost effective to implement a QPLL seeing as its overall performance is almost the same as the current method.

8.2. *The hardware-PLL*

The results from the hardware-PLL measurements showed, as expected, a very limited frequency interval in which we could operate. In this interval the hardware-PLL has no problem tracking the signal, but neither does the current method which also provides a less noisy signal. However our model of the current method is idealised without any filter delays etc. and comparing it to the hardware-PLL is a little unfair. The hardware-PLL's biggest drawback is that it unlocks at low frequencies ($< 20\text{Hz}$) and we can not detect zero velocity with this circuit in a good way, which is necessary that we are able to do. The upper frequency limit is of less importance since a series of differently tuned PLL circuits could cover the entire interval by switching between them.

Since the hardware-PLL can not track low frequencies it is not a good solution in its current form. It would also be quite expensive to implement because we would need several PLL circuits to cover the entire interval. An alternative to this is to only use the hardware-PLL for frequencies under 100Hz and let the current method calculate the velocity at higher frequencies seeing as it works well at frequencies over 100Hz where it is guaranteed to get a new reading every 10ms. But the problem with the hardware-PLL losing its lock for low frequencies ($< 20\text{Hz}$) would still remain and the method can not be used unless this problem is solved.

8.3. *Other possible improvements*

Both the performance of the PLL methods and the current method are highly limited by the sensor solution used today, especially at low velocities. The current method can not give a certain estimate until a new gear passes the sensor and is forced to make guesses using its low frequency algorithm; this puts a fundamental limit on its performance. The PLL methods are not dependant explicitly on gear passages since they measure the sine-like sensor output continuously but at low velocities the amplitude of the signal gets smaller and its shape more uneven, limiting the performance of the PLL methods as well.

To drastically improve this measurement a new type of sensor solution is needed. The most obvious solution is to increase the number of teeth on the gear wheel since this increase would make more teeth pass by the sensor in a given interval. However this would require large modifications of the gearbox and there are limits on how many teeth a gear wheel might have. More teeth would also inevitably have to be smaller which could make the sensor output weaker and decrease the signal to noise ratio. Available on the market are completely new types of more modern sensors like Hall-elements and Giant Magneto Resistive (GMR) types which are based on completely different ways of measuring the rotational speed. These methods are presented in the thesis by Martin Törnquist (LiTH) also conducted at Volvo CE.

9. Future work

We have seen that the performance of the QPLL is only slightly better than the current method but also has some drawbacks, and the hardware-PLL has a very limited interval where it can operate and even in this interval it is not better than the current method. Before thinking about actually implementing one these methods they need to be improved if possible.

Possible improvements to the QPLL could be to adaptively change its internal constants to increase its performance. Since the QPLL is a complex system that is hard to analyze this would have to be done empirically and could be very time consuming. This improvement has been discussed with the authors of [3] and is considered as an interesting starting point for future work.

The functionality of the hardware-PLL at low frequencies needs to be improved. Its current un-locking behaviour at 20Hz is too high and needs to be lowered in order for it to be an interesting solution. One way of lowering this limit might be to filter the input to the hardware-PLL in a better way trying to get a cleaner signal at low frequencies as well. Since the output from the sensor can vary over a wide range of frequencies (0Hz – 4500Hz) designing a good filter can be hard, especially if the noise at low frequencies has a frequency that is in our interval of operation.

Even with these improvements at place we would still be very limited by the current sensor solution and the only way to drastically improve these measurements is to choose another sensor solution. The ones discussed in the previously mentioned thesis by Martin Törnquist seems promising and are probably the best way to improve the measurement of rotational speed for Volvo CE.

10. Bibliography

10.1. Scientific papers

- [1] Lennart Harnefors, *Nonlinear and adaptive systems*, 2000, Teletransmissionsteori Mälardalen University
 - [2] A.K Ziarani, A. Konrad, *A method of extraction of nonstationary sinusoids*, 2004, Elsevier
 - [3] M.Karimi-Ghartemani, A. K. Ziarani, *Performance characterization of a non-linear system as both an adaptive notch filter and a phase-locked loop*, 2004, International journal of adaptive control and signal processing
 - [4] Masoud Karimi-Ghartemani, Houshang Karimi, M. Reza Iravani, 2004, *A Magnitude/Phase-Locked Loop System Based on Estimation of Frequency and In Phase/Quadrature-Phase Amplitudes*, IEEE Transactions on industrial electronics, Vol. 51, No. 2
 - [5] Guan-Chyun Hsieh, James C. Hung, 1996, *Phase-Locked Loop Techniques-A Survey*, IEEE Transactions on industrial electronics, Vol. 43, No. 6
 - [6] B. Gilbert, *A precise four-quadrant multiplier with subnanosecond response*, 1968, IEEE journal of solid state circuits Vol. 3, No. 4
 - [7] Nikolaos I. Margaris, *Theory of the Non-linear Analog Phase Locked Loop*, 2006, Springer
 - [8] Markus Larsson, *Rotational Speed Measurement using Passive Inductive Sensors*, 2005, Internal document Volvo Construction Equipment Components AB
 - [9] Melinda Au, *Three phase sensorless speed measurement for locked axle detection*, 2006, Clarkson University
 - [10] W.Ribbens, *Engine crankshaft angular position sensor*, 1998, Understanding automotive electronics 5th edition Woburn
-

10.2. Web pages

- [11] Ian Purdie's Amateur Radio Tutorial Pages
http://my.integritynet.com.au/purdic/dbl_bal_mix.htm
- [12] Modeling tutorial for Matlab and Simulink
<http://www.me.cmu.edu/ctms/modeling/opamp/freebodydiagram/mainframes.htm>
- [13] Data sheet for the LM565C phase-locked loop
<http://www.national.com/ds/LM/LM565.pdf#page=1>
- [14] Volvo CE picture archive
http://pro.vce.ams.llr.se/CumulusE_Z/VCE_ImageGallery/ProUsers.jsp
- [15] PLL history and introduction
<http://www.uoguelph.ca/~antoon/gadgets/pll/pll.html>