Contents

Chapter 1

Introduction

This report contains a description of the research conducted at Strukton Systems, Hengelo, as a master's thesis of Maaike Petit-Kruyswijk, student Applied Mathematics at Twente University. It aims to develop a measuring system for installation on a (work) train to increase the driver's safety.

This research was initiated by Strukton Systems. Strukton is one of the large contractors in the railway business. They have 50 % of the market in the Netherlands and work in Sweden, Denmark, Belgium, Germany, Surinam and several other countries. Their area of work includes maintenance of tracks and rail vehicles, designing and building trains, complete power management on trains, power distributors, condition monitoring and video inspections of the tracks.

Comfort vs. safety
When maintenance has to be done on the tracks, there is always a struggle to balance the hindrance for train passengers and the safety for working personnel. Passengers prefer to have as much train movements as usual, while for workers it would be safest if no trains at all drive on the tracks they are working on or the adjacent tracks. For years a compromise has been in use: maintenance is mainly done at night or in weekends, adjacent tracks are in use and trains on the track under construction divert to the adjacent track for the shortest possible time. Recently there have been several accidents or near-accidents. These were due to work trains driving on tracks which were in use. Normally the train detection system would detect a vehicle and set the signal behind the vehicle to red. However, some work vehicles are light and the automatic system does not detect them. Therefore a measuring system has to be developed to inform the driver of the status of his detection. A second system can be developed to better the detection properties every time the measuring system issues a 'not detected' warning.

Train circuits and ATB
The train detection system in use in the Netherlands uses currents through rails and relay technology to detect trains. This principle is called train circuits and assumes a good conduction between rail and wheel. Generally, we can make a distinction between two situations. Either an ATB-signal is send into the tracks (an extra current with, in theory, certain well-defined properties), or there is no additional current to the train detection current. We would like to give a conclusion about the train detection for both these situations, but we will find we can only draw conclusions in the first case, using mainly filtering techniques. For the second case we will need more or different information than the current due to the train circuits. For this, a system is under development at Strukton Systems, but it falls outside of the scope of this report.

Approach
For our measuring system, we measure the currents through the axles of a train. These will in general differ from the currents through the rails, since we have to assume the wheel-rail contact has at least some resistance. However, it can only weaken the properties of the signal, so if the measured signal in the axles has all properties we are looking for, the current through the rails will certainly have those properties. We will write Matlab-functionalities to decide if the properties are present. These will form our measuring program.

Overview
In this report we will first discuss the theory of train circuits and ATB in Chapter 2: Train theoretical background. This is quite specialized knowledge and we will use large parts of the section on ATB in the development of the model, therefore we advise the reader to read this chapter carefully. After this we describe what we measured and how in Chapter 3: Train data. In the same chapter we give an overview of the data we obtained. Thirdly, in Chapter 4: Analysis we describe the mathematical tools we will need in the development of our system. It is assumed the reader is familiar with Calculus and Analysis to the level of a bachelor. Chapter 5: Development follows the development of two preliminary models. The first is based on Fourier Analysis, the second on filtering and signal manipulation. The chapter after this, Chapter 6: Final Model describes the system we developped to analyse data measured on a train while it is driving on tracks and time is running. After this we have a short mathematical intermezzo. Chapter 8: Frequency Resolution contains an attempt to generate optimal windows which minimize leakage for a given frequency resolution. We conclude this report with a section on results and future work, Chapter 9: Conclusions and future work.

Chapter 2

Train theoretical background

In this chapter we will sumarize the knowledge we deem necessary for the reader to acquire to understand the choises made in this report. We will start by explaning the train detection system in use in The Netherlands since the Marshall-plan in the late forties, early fifties. After this we will discuss some solutions people have used for a problem with the detection system, called 'Loss of Shunt'. Lastly, we will explain a system known as ATB. We will use the characteristic shape of the signals generated by this system to develop our measuring system. The background knowledge was gained from reading [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12].

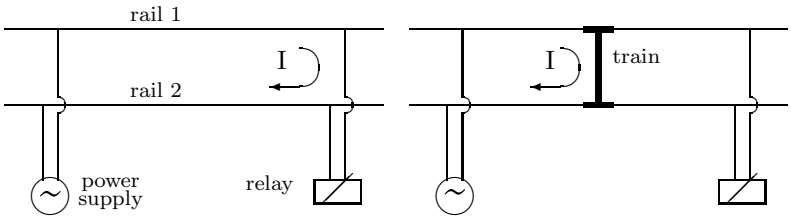
2.1 Summary train detection by track circuits

The Dutch rail infrastructure is divided into sections. These sections vary in length from about 50 meters, for instance at a crossing or a train station, to 1250 meters at long straight stretches. The rails from two sections are insulated electrically by means of an isolated joint (Dutch: elektrische scheidingslas). Now consider a single section. On one end of the section an alternating voltage supply at 75 Hz is connected to the tracks, and on the other end a fail-safe relay is placed. Fail-safe means that the relay is closed or up if a current flows through it and it opens or drops if the current stops. Now if there is no train in the section, a current flows from the supply through rail one, through the relay and via rail two back to the supply and the relay is up. When a train enters the section, its wheels and axles short the rails (the wheels have good 'shunting') causing the current to run from the supply, trough track one, through the train wheels and axles and through rail two back to the supply. As a result no current flows through the relay, which subsequently drops (the time delay can be up to 0.2 seconds), causing the train signal behind the train to turn red (Figure 2.1). The undesirable phenomenon where the train does not short the rails (and consequently the relay doen not drop) is called 'Loss of Shunt'.

There are two types of sections: single- and double-legged sections. Double-legged sections are sections where the return DC current from the overhead wire, due to trains further on the tracks, goes through both rails. Coils are used to pass the return current from one isolated section to another, while stopping the alternating currents. We can encouter noise resulting from rectifiers for the current in the overhead wire. In a single-legged section, only one of the rails is used for the return current. All return-rail parts are electrically connected, so in effect, only one of the rails has electrically separated sections.

2.1.1 Disadvantages of traindetection using train circuits

Train detection by using currents assumes there is good conduction between wheels and rails. With the large and heavy trains built before 1980 this is indeed the case. Unfortunately, we cannot assume good conduction for modern trains. These are lighter, often build with different materials than steel, not always running on electricity and they are better aligned. Their lesser weight makes shorting more difficult. New trains are often built using aluminium, which is not as good a conductor as steel. Further, since running on electricity means guiding large return currents through the rails, lowering the resistance, non-electric propulsion has a negative effect on train detection properties. Lastly, since modern trains are better aligned, they do not grind the corrosion and pollution (such as leaf-residu) from the tracks as well as the old trains do.



(a) No train in the section (b) Train in the section

Figure 2.1: Scheduled representation of train detection

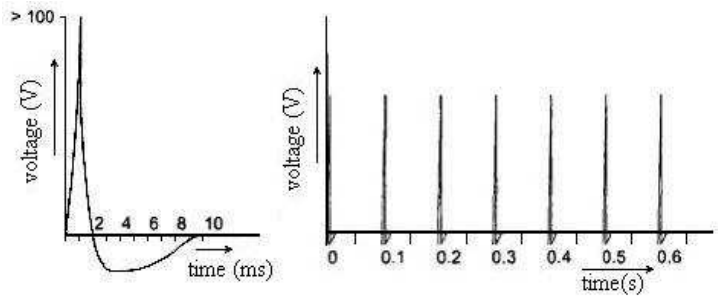


Figure 2.2: PSSSL signal

2.2 Partial solutions for 'Loss of Shunt' from literature

A lot of research has been done trying to improve train detection, that is, to improve shunting between wheels and rails. It is an old subject: the first patent describing a solution dates back to 1933 [13]. Below we give a short impression of systems currently in use, as described in [6] and [2]. First we discuss methods which have to be installed on the tracks, after that methods which are applied from a moving vehicle. None of these guarantee detection will take place, except PSSSL.

2.2.1 Railinfra solutions

Peak voltage train circuits

(Dutch: PiekSpanningSpoorStroomLopen (PSSSL)) This method is used on parts of the tracks that are not frequently used or are mainly used by poorly shunting material, such as light diesel trains. For a short time, a high voltage is supplied to the tracks, resulting in the film of corrosion or pollution being pierced. This is repeated in a pattern similar to an electric fence (Figure 2.2) at 10 Hz +/- 0.2 Hz. It is an alternative detection system. There are three types of occurrences of PSSSL. In the first case, the piek voltage is always applied to the rails, train or no train. In the second case, it is turned off if there is no train in the section and turned on in case there is. In the third case it is combined with ATB (see Section 2.3). When combined with ATB, PSSSL is switched on, but is turned off as soon as a train is detected in the section. ATB can then be switched on.

One disadvantage of PSSSL is the risk it can bring for small animals touching the rails. A second disadvantage is that for nation wide coverage, all tracks would have to be adapted, which is time-consuming and expensive.

Anti-leaf methods

Several methods are applied to reduce the loss of shunt resulting from leaves on the rails:

- Fell trees.

- Management of vegetation.
- Anti-leaf fences along the railway.
- Anti-leaf gutters along the tracks.
- Watersprinklers along the tracks.

With exception of the first, all these methods are expensive and hardly effective.

2.2.2 Vehicle solutions

Direct cleaning of the tracks

There are several techniques in use to scrub the insulating film from the tracks. The most common of these are discussed below.

- Brushing. For this technique stationary or rotating bruches are installed beneath a (work)train. Disadvantages are the fact that brushes wear out and the brushing train cannot go faster than 6 km/h. The advantage of brushing is the prize: it is cheap.
- Electrical dragbrushes or rollers. By inducing a voltage from one brush to another across the track it is hoped that the insulating film is pierced and broken down where possible.
- Grinding. Grinding means removing a thin layer of steel from the rails, and is therefore used as few times as possible. This method is usually effective, since the insulating film is completely removed. However, sometimes corriosion and contamination of the tracks happens very fast. There is a known case where the effect of grinding on shunting had worn out within half an hour.
- High pressure cleaning with water. This is quite effective. Disadvantages are the complicated equipment and the amount of water one needs to bring along. The maximum velocity for this technique is 60 km/h.
- Chemical cleaning. Tracks can be cleaned with a chemical product like lemon oil or Natrium-bicarbonate. The disadvantage of this is that often the tracks have to be cleaned afterwards when too much is applied or for environmental reasons. Sometimes cleaning with water is satisfactory, but cleaning with high pressure or even by hand can be necessary. This means the processing speed is very low.
- Optical cleaning. In the UK a system has been developed by the name of LaserThor. This is a laser-based system to clean slippery tracks and increase adhesion. The equipment can only be mounted to a worktrain. Currently it is attempted to raise the effective speed from 60 to 80 km/h. This system might work for improvement of train detection.

Friction improving products

Several products exist for application to the tracks from a (working) train. They increase the friction, but might also improve the shunting properties. An example is Sandite, a mixture of sand, water and a binding agent. The sand particles help piercing the insulating layer. This product can be applied at 60 km/h. A disadvantage is that the product can have an insulating effect when applied too richly. The same holds for spreading sand.

TCA

The Track Circuit Assister uses high frequency induction to pierce the insulating film. A large induction loop is installed beneath the train (Figure 2.3). The result is comparable to the peak voltage train circuits described above. A disadvantage is that this system only works when the wheel-rail connection behaves like a non-linear resistor. This wheel-rail connection behaves very dynamically and can switch often between behaving as non-linear resistor, linear resistor, coil or capacitor. Secondly, it cannot be equipped by all trains.

Figure 2.3: TCA induction loop beneath a train

Opposing current

In 1935 a patent was granted to mr. P.M. Bourdon [13] for a train detection improvement system. His idea was compensating the current in the rails by supplying an opposing current from the train. This current is increased in magnitude until the relay drops. It is not sure whether this system has actually been used.

Actel Axle

This system uses dynamic control. The voltage over the two tracks is measured in one of the train axles. This signal is compared to the reference zero (zero voltage means there is a short circuit). If the signal differs from zero, a correctional signal precisely cancelling the measured sigal is sent into the tracks. This could be done using either a current or a voltage. This way, the train becomes an active component in the train detection system. The drawback to this method is that for measuring the voltage, two wheels on one axle have to be electrically isolated. This can be done in two ways: insertind a rubber band between the wheel and the steel tires, but this will create imbalances in the wheels. The other option is cutting one axle in two. This is extremely costly and impractical, even unsafe in some cases.

2.3 ATB-EG

The detection of the train, that is, the dropping of the train detection relay (TR) is a sign for the ATB-system (Automatic Train Influencing, Dutch: Automatische Trein Beïnvloeding) to switch on. The system most widely used is ATB-EG (ATB first generation, Dutch: ATB Eerste Generatie). This system was developed to increase the safety on the tracks. It monitors the actual speed limits, taking into account the color of the train signals, and displays the current limit in the driver's cabin. If the driver drives faster than permitted, a warning signal is given. The driver then has to respond within a few seconds by braking. If he does not react, the system intervenes and brings the train to a full stop as fast as possible.

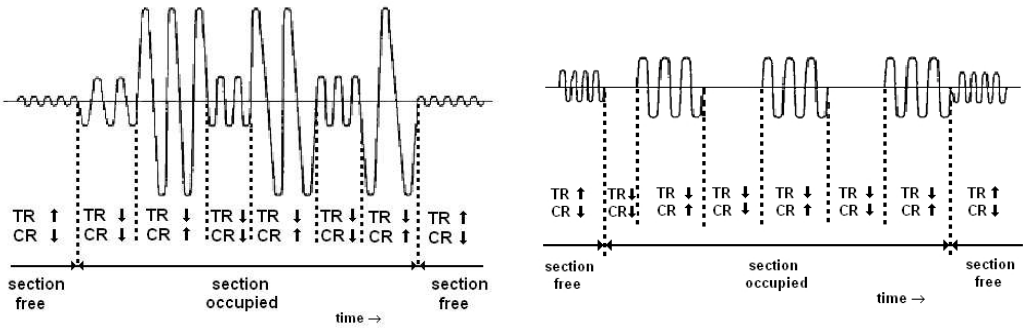
The working is as follows. As the TR drops, the systems supplies an additional voltage with a larger amplitude to the rails on top of the train detection voltage. As a result, an increased current flows through the rails. This increased current is rhythmically interrupted by a second relay (code relay, CR) and the frequency of the interruption is a code for the maximum velocity in the section (Figure 2.4). It takes into account the color of the railway light in the next section. These codes are allowed to deviate at most 3 pulses per minute or 0.05 Hz from the values in the table. The code is always offered to the front of the train. So depending on the direction of the train, relay-sided or supply-sided coding is used. In most sections, the coding can be done at either end of the section to facilitate train driving in both directions. During supply-sided coding all current is interrupted, during relay-sided coding only the larger current is interrupted. A schematic representation of the current through the rails for both supply-sided and relay-sided coding is given in Figure 2.5.

We summarize the characteristics of ATB-EG.

- The basic frequency is 75 Hz. It is allowed for the actual frequency to deviate +/- 3 Hz from this.
- The coding is achieved by adding an extra current to the rails and switching this current on and off with one of the frequencies from Table 2.4. This coding can be done at the side of the supply or at the side of the relay. In the case the coding is done at relay side it is sent in having a phase opposing

Code	Frequency	Denotation
96 pulses /minute	1.6 Hz	max. 140 km/h
120 pulses /minute	2 Hz	max. 120 km/h
147 pulses /minute	2.45 Hz	not in use
180 pulses /minute	3 Hz	max. 80 km/h
220 pulses /minute	$3\frac{2}{3}$ Hz	max. 60 km/h
270 pulses /minute	4.5 Hz	not in use
75 pulses /minute	1.25 Hz	turn off ATB
no interruption of current	0 Hz	max. 40 km/h

Figure 2.4: Possible ATB codes



(a) Current through the tracks for supply-sided coding (b) Current through the tracks for relay-sided coding

Figure 2.5: Sketch of ATB-currents through the tracks

the phase of the supply. Further, in the case of coding at relay side, the down-period of the coding is nearly zero (Figure 2.5. For coding at supply-side the down period is equal to the (small) current the supply is supplying [8].

- The duty cycle of the code is between 40 and 60 %. In other words, the down period is just as long as the up period.
- The high current is between 6.5 and 25 A, the low current should be below 3 A. The area between 3 and 6.5 is called the 'forbidden area'. No currents should have an amplitude in the forbidden area, to facilite distinguishing the code from any noise present.

Chapter 3

Train measurements

In this chapter, we will examine the data we obtained in a lab setting and by measuring the current through the train axles while driving on tracks between Almelo and Deventer, Overijssel, Netherlands. We will try to determine if the properties the signal should have according to the theory are indeed present in the measurements.

3.1 Measuring arrangement

We use three different measuring arrangements, two on vehicles and one lab arrangement. The lab arrangement is at RIO, Rail Infra Training center. For the two arrangements on a vehicle we use a so-called clamp meter, a special type of ammeter. The working is as follows: we place a toroide coil (Fig 3.1) around the axle through which we want to measure the current. Then by magnetic induction a current is induced in the coil and by measuring the induced current with an ordinary ammeter, we can deduce the current through the axle [14].

3.1.1 Arrangement on Ploeg 14

The first arrangement (also chronologically) is the arrangement on a locomotive called the Ploeg 14 (Figure 3.2). We measure the current through both axles; front and back. We use two clamps with dividable cores, so they could be installed without removing the wheels from the axles (Figure 3.3). Since this clamp basically is a current transformator, it is easily understood its output is current. Since our measuring device, a Data Acquisition Unit, can only register voltages, we used a resistor (10 Ω) to convert currents to voltages. Another advantage of this conversion method is that we can choose the resistor such that our measuring range is fully utilized. The schematic representation of the complete arrangement is in Appendix B.

3.1.2 Arrangement on KROL

The second arrangement is the arrangement on a so-called KROL (Crane on lorries, Dutch: kraan op lorries) (Figure 3.4). This is a small crane which can also dig, drag and equip an electromagnet. It has tires for driving on the road and it can equip four small (approximately 70 cm) steel wheels for driving on railway tracks. The axles of these steel wheels has a special form: they are 'wrap around', Figure 3.5. The axle is connected to the steel frame of the KROL on both sides of the wheel. So if we want to measure all current going from one rail, through the two steel wheels on that rail, through the KROL, to the other rail, we need 4 clamps: 2 for each wheel on the left (or right) side of the KROL. Since we knew we were going to use this particular KROL for measuring before it was built, we could design the clamps to be built-in. So these clamps are not dividible and fit perfectly into the hollow wheels (Figure 3.6). We now have 4 data series to measure. As we did on the Ploeg 14, we use a resistor of 10 Ω to convert currents to voltages. The schematic representation of the complete arrangement is in B.

Figure 3.1: Example of a toroide coil

Figure 3.2: Ploeg 14, reprofiling locomotive

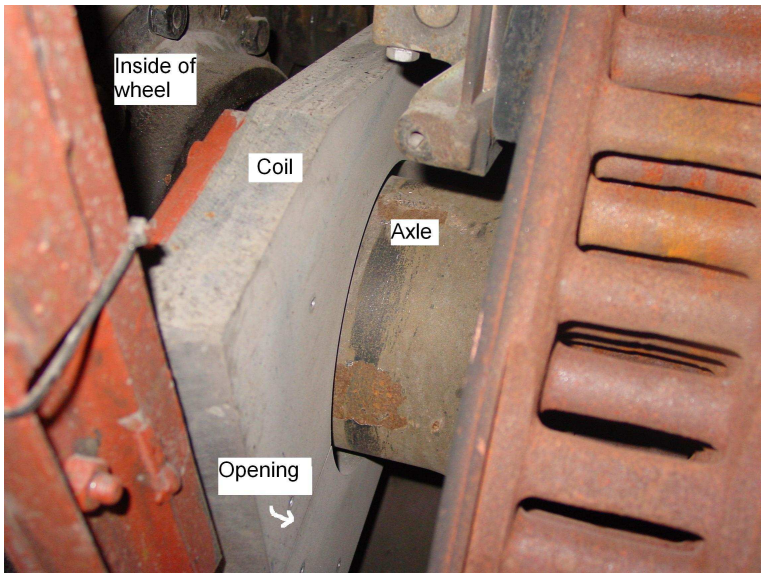


Figure 3.3: Divisible clamp on the Ploeg 14

Figure 3.4: The KROL used for measuring

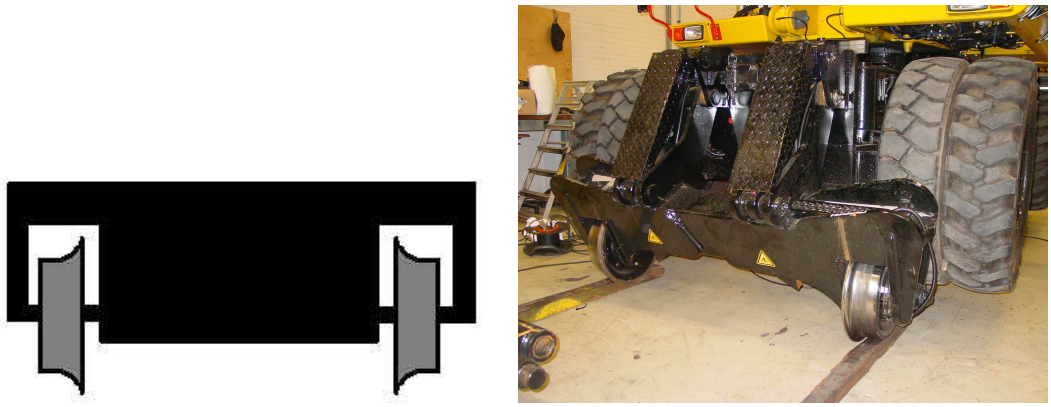


Figure 3.5: The axle of the KROL is 'wrapped around' the wheels

Figure 3.6: The coil is integrated in the wheel

(a) The supply

(b) The relay

(c) The code generator

Figure 3.7: Some parts of the measuring arrangement (real size: about 35 cm high)

3.1.3 Arrangement at RIO

The lab arrangement is at RIO, Rail Infra Trainingcenter. Here it is possible to build scaled railway sections with varying length and the same relay technology as outside on the tracks. It was decided to do these tests when it became clear the measurements on the Ploeg 14 showed little about sections without ATB code. So the purpose of these measurements was studying what happens on tracks without ATB. This is done on a predesigned board of approximately two by one meter, where components can be pinned into. Pieces of tracks can be connected into longer tracks by wires, the actual relays are used and the resistance of the track ballast can be set to (almost) 0 Ω, 1 Ω, 3 Ω, 5 Ω or (near) infinity Ω. To one side of both rails the 'supply-side' circuit is connected, to the other side the 'relay-side' circuit. A train was simulated by placing a single variable resistor between the two 'rails'. The quality of the shorting of the rails can be varied by increasing or decreasing the resistance of the resistor. Some parts are shown in Figure 3.7.

3.2 Overview of measured track data

In this section we will show some recurring properties of the data we measured with the Ploeg 14 and the KROL. We will consider how ATB codes behave compared to the theory and we will investigate special events, such as crossings. Lastly we will examine what noise frequencies we can identify. This section will help us to keep the real signals in mind while developing our program(s) in Section 5.2.

3.2.1 ATB code in reality

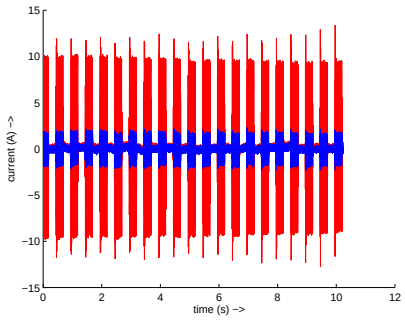
In this section we will review some examples of measured, as opposed to theoretic, ATB code. In general, we noticed that the code frequencies follow the prescribed frequencies very accurately. The current level is also in accordance with the theory. The duty-cycle however can be a bit off, in most of these cases the duty cycle is longer (up-period is longer than down-period). The restriction on the forbidden area is violated most often. There are quite a few measurements in which the down-period is well above 3 A effective value.

ATB code 120, relay-sided coding

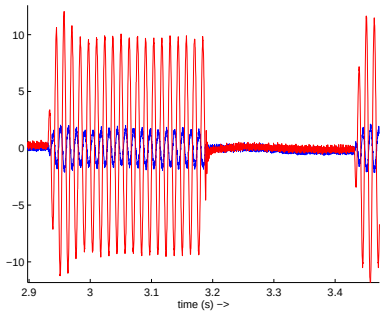
In Figure 3.8 we see a typical example of ATB code 120 which is coded at the relay (2.3). Characteristic for relay sided coding is the virtual absence of current in the down-period, since the supply curretn is not present. The top value of the current is between 6.5 and 25 A and the amplitude of the down period is below 3 A, as required. If we look at a close-up of the FFT of the absolute value of the signal, we see modulation of the code-frequency around the peak of the basis frequency at 75 Hz (+/- 3Hz). So all properties agree with the theory.

ATB code 120, supply-coded

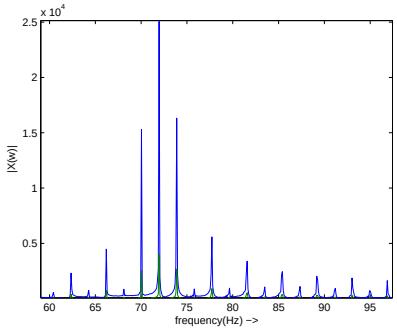
Figure 3.9 is an example of code 120 coded at the supply side of the section. Since the ATB current is added to the supply current in this situation, the level of the current in between pulses is higher. In this



(a) Red: 1^{st} axle, Blue: 2^{nd} axle



(b) Detail of signal



(c) Detail of magnitude of FFT of |data|

Figure 3.8: ATB code 120, relay-coded

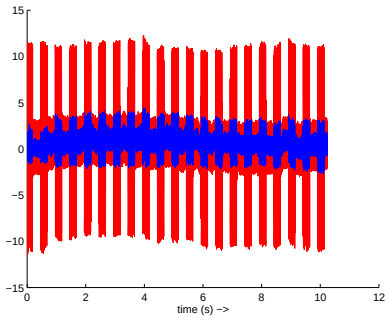
measurement and most others, the down period remains beneath 3 A. Still, 75 Hz is the dominant frequency and the modulation by the 2 Hz frequency is easily recognizable. Since all 'neat' codes look like this code in noise level and current level we will not treat the other codes (147, 180, 220, 270, 96 and 75 pulses per minute).

Transition in code

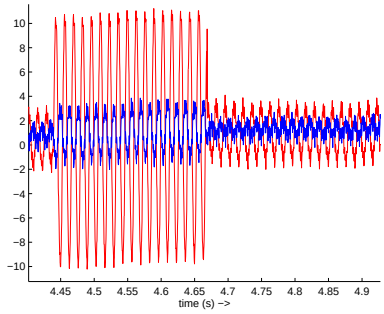
Often a change of code will be measured. This can be due to the train passing into the next section or a railway signal changing color. We see in Figure 3.10 that the number of pulses just switches from one value to another, in this measurement at about the 6^{th} second. Note that there are two different codes present in the FFT of the absolute value of the signal, that is, we can see their modulation. However, we cannot determine if one came before the other or if they were present at the same time. We need to keep in mind our program should be able to detect a change in code.

3.2.2 PSSSL

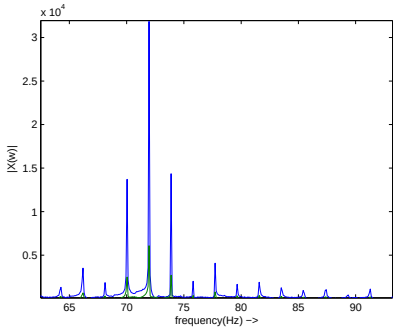
In Figure 3.11 we see the result if we measure PSSSL (see Section 2.2.1). Note the likeness of the measured signal to Figure 2.2. The noise at 600 Hz will be explained in Section 3.2.5. PSSSL can also be combined with ATB code, as long as it is made sure the two signals are never simultaneously present in the tracks. PSSSL has to be switched off before the code can be supplied to the tracks. Therefore, in a section with both PSSSL and ATB we will typically measure PSSSL during the first two seconds, this is the time the PSSSL-train detection relay needs to drop (yes, it differs from the usual train detection relay), and after that we can see code. When trying to detect PSSSL, we need to keep in mind the special shape this 10 Hz signal has. However, the presence of PSSSL does not necessarily mean a train was detected, so we cannot conclude anything from data with PSSSL.



(a) Red: 1^{st} axle, Blue: 2^{nd} axle



(b) Detail of signal



(c) Magnitude of FFT of |data|

Figure 3.9: ATB code 120, relay-coded

3.2.3 Special events

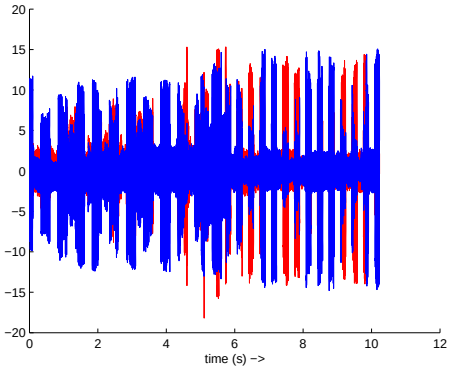
In this subsection we show some possibilities for events while measuring on a riding train. We can pass train stations, switches, cross a road, go from one section to another or we can ride on a railway yard.

Section transitions

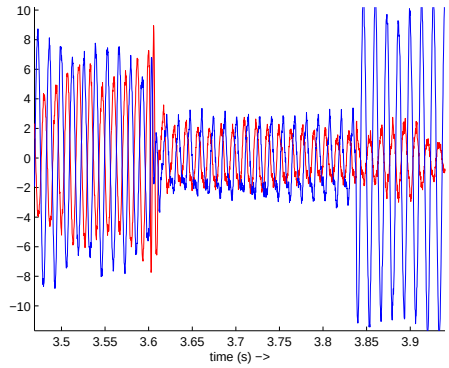
From railway theory we know that during a section transition the phase of the train detection and the ATB signal can shift 180 degrees with respect to the signal in the previous section ([4], specification ATBEG, aspect 8). The reason for this is that, if an electric separation weld is bridged, the added signal of two sections is zero or small, resulting in the train detection relay opening and red signals. From theory we also know the traindetection relay can have an OFF-delay up to 0.2 seconds, so it is possible we detect no ATB signal for 0.2 seconds while passing into a new section over an electric separation weld. In Figure 3.12 some section transitions are shown. Note that we can see the train is driving forwards. The fist axle (red) is disturbed first and 0.3s later the second axle changes section. This agrees with the speed of the train (approximately 90 km/h) and the distance between the axles (8.10 m). Also note that a high current (we could not measure above 22.5 A) can run through the axle while crossing the electrical separation weld.

Crossings

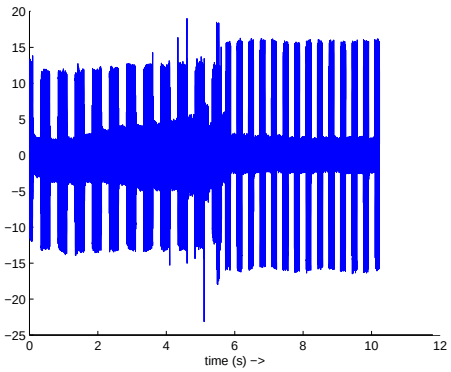
Crossings are special cases. For the ATB system in the tracks it is very difficult to reliably achieve a certain current (above 6.5 A) on a crossing. This especially holds in the winter, since cars can drive brine and other moist pollution onto the tracks. This results in a low resistance from one rail through the ballast (grit) and concrete to the other rail, hence a large loss of energy. Therefore, most crossings are equipped with train circuits as usual, but with ATB-loops and sometimes with another detection mechanism (axle-counters, pedals). An ATB-loop is a wire carrying the ATB current which normally is sent into the rails. This can be



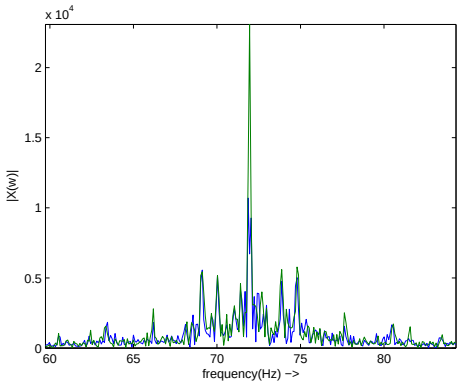
(a) Red: 1^{st} axle, Blue: 2^{nd} axle



(b) Detail of signal

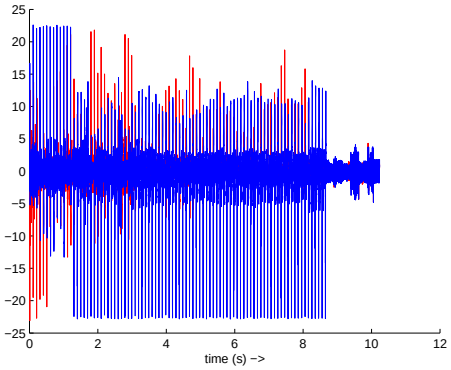


(c) Sum of signal in both axles

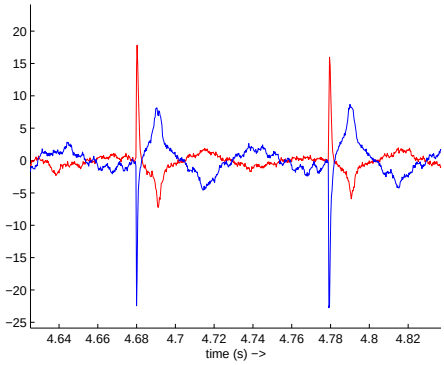


(d) Magnitude of FFT of |data|

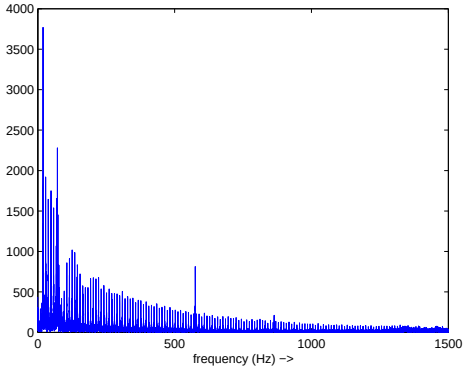
Figure 3.10: ATB code 120 and 180, 1228 1 vrije baan 60



(a) Red: 1^{st}, Blue: 2^{nd} axle

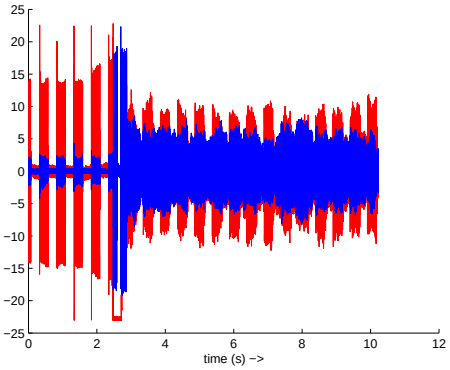


(b) Detail of signal

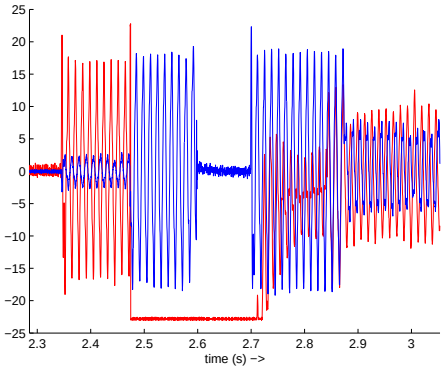


(c) Magnitude of FFT of |data|

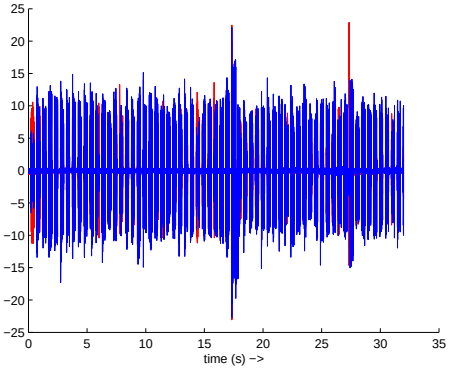
Figure 3.11: PSSSL



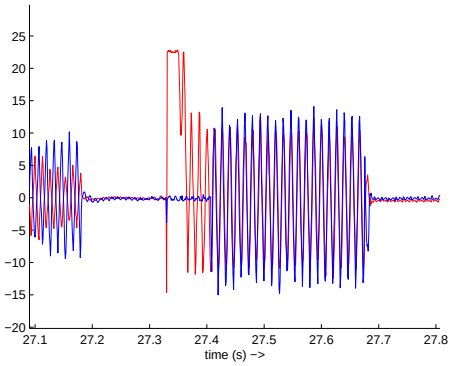
(a) Section transition at 12.41h



(b) Detail of section transition at 12.41h

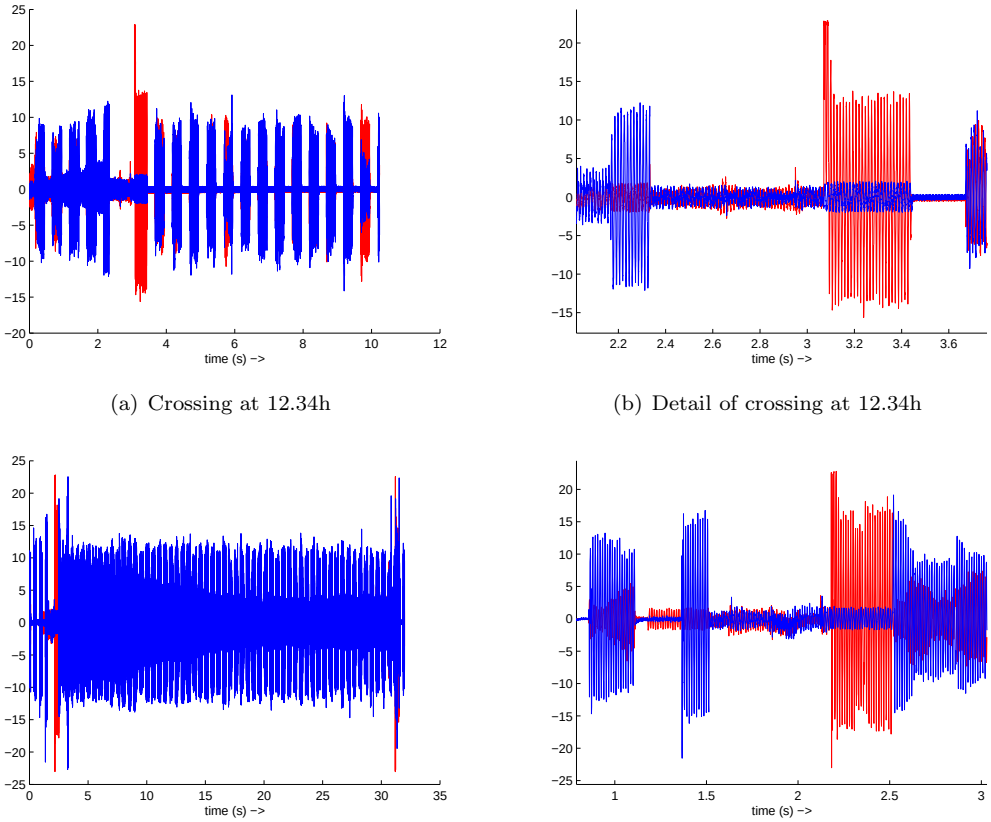


(c) Two section transitions at 12.47h



(d) Detail of section transition at 12.47h

Figure 3.12: Section transitions



(a) Crossing at 12.34h

(b) Detail of crossing at 12.34h

(c) Crossing at 12.50h

(d) Detail of crossing at 12.50h

Figure 3.13: Crossings

detected by the recording coils in the train, but it cannot be measured in the axles of the train. This is why we often measure no code at a crossing (Figure 3.13). PSSSL is not allowed at a crossing.

Switches

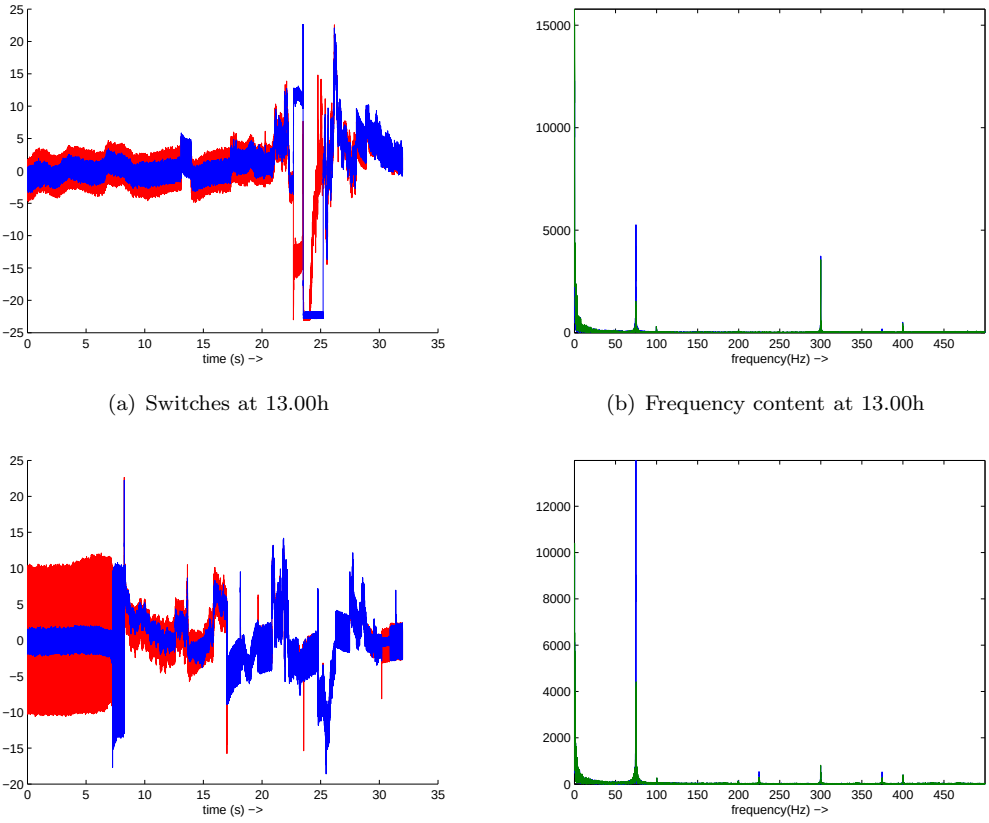
When crossing a switch we measure a very irregular current pattern (Figure 3.14). The 75 Hz frequency is present in this signal, but we measure no code. In many cases ATB-loops are used on switches to control the currents which otherwise could be flowing in two directions.

Bridge

Twice we crossed a bridge; once going from Almelo to Deventer, once on our way back. In Figure 3.15 the two data series are displayed. We can conclude that ATB code can be present on a bridge and that the measurements, at least on this particular bridge, are not more or less noisy than on ordinary tracks.

Curves

We passed one curve in our test traject. Because the train scrapes over the rails a little when the fixed wheels are forced to follow a curve, we expect the detection in a curve of the rails to be good. In Figure 3.16 the data series measured in the curve is displayed.



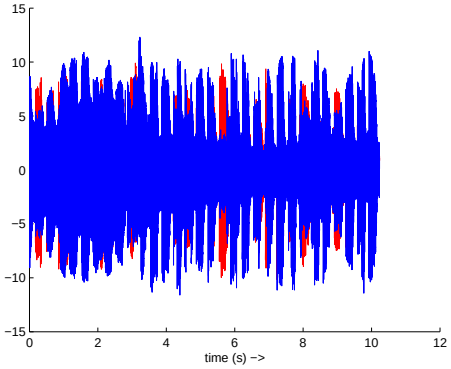
(a) Switches at 13.00h

(b) Frequency content at 13.00h

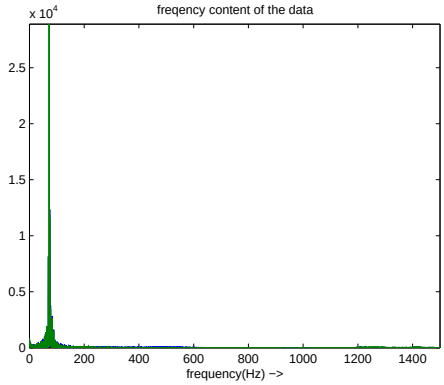
(c) Switches at 13.02h

(d) Frequency content at 13.02h

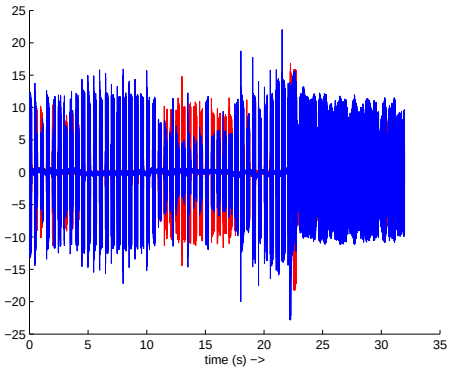
Figure 3.14: Switches



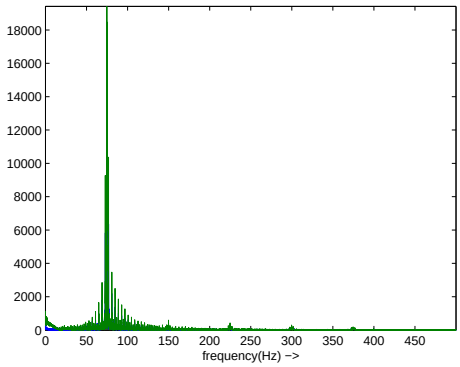
(a) Bridge at 13.00h



(b) Frequency content at 13.00h



(c) Bridge at 13.02h



(d) Frequency content at 13.02h

Figure 3.15: Passing a bridge

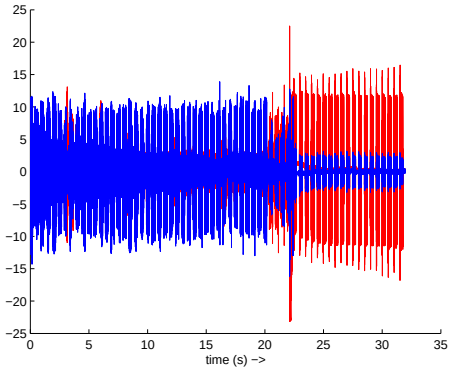


Figure 3.16: Straight stretch, then a bend (starting at $t \approx 22$) at 12.49h

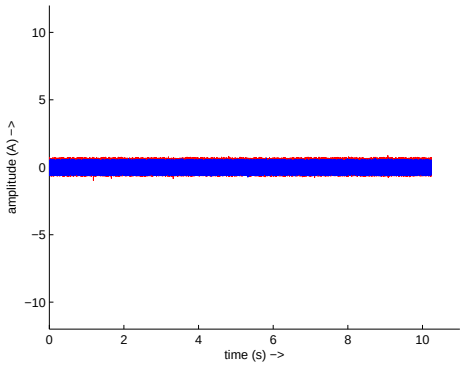


Figure 3.17: Current measured on a railway yard

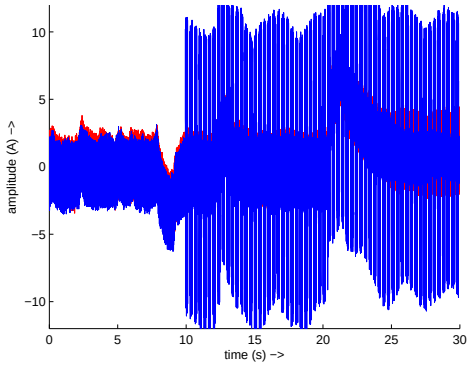


Figure 3.18: A train is accelerating further ahead. From the 10^{th} second, PSSSL is on the tracks

Railway yard

At a railway yard there is seldom ATB code installed on the tracks. More specific, ATB code is only present on tracks just passing through. We therefore only measure the train detection current originating from the supply (Figure 3.17). Railway yards, together with train stations make up for about 10% of the Dutch railway network. On these tracks we only measure the train detection current.

Other trains

Other trains can have an influence on what we measure in the axles of our train. When an electrical train ahead of us accelerates, it draws current from the overhead wire. This current passes through our section of the rails as a return current. This current is DC and in our measurements we see this as a translation of the signal either up or down, with its influence decreasing according to be^{-ax} for some a and b (Figure 3.18). Another possibility is a train passing on the tracks alongside ours, in the opposite direction. This results in irregular disturbances, caused by induction from the magnetic fields the other train generates around itself (Figure 3.19).

3.2.4 Phase shifts

We measured some remarkable occurings of phase shifts between the two axles of the train. This is only possible because the part of the tracks between the two sets of wheels has a nonzero resistance. In Figure 3.20(a) we see that in one axle an almost constant current is measured, the train detection current. In the other axle the measured signal is clearly containing ATB code. The signal containing ATB shifts phase

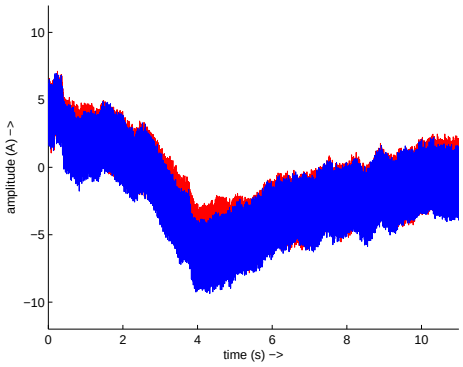
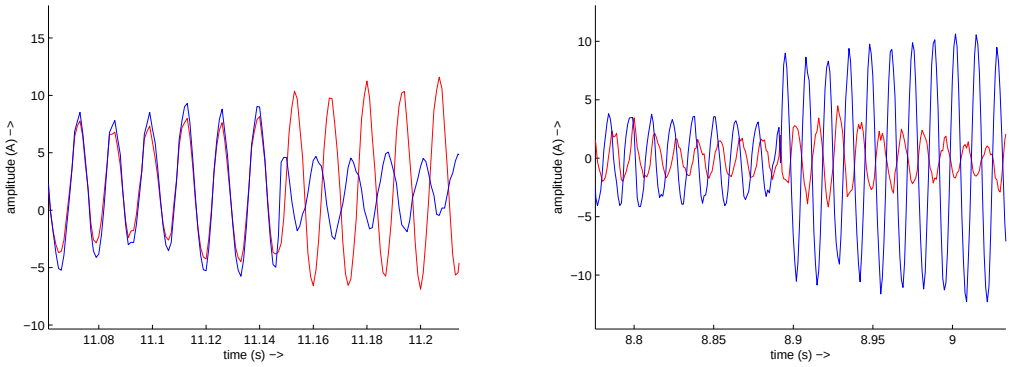


Figure 3.19: A train passes on the adjacent tracks



(a) A signal with 180 degrees phase shift every pulse (b) A signal with 90 degrees phase shift every pulse

Figure 3.20: Details of signals containing phase shifts

180 degrees every time it changes from the up- to the down-period or vice versa. In Figure 3.20(b) a data series is showed in which the phase shift is less than 180 degrees. A satisfying explanation for this behavior has not yet been found.

3.2.5 Noise

In literature ([4], specification ATBEG, aspect 10) we found a list of noise components originating from the return currents through the rails, see Figure 3.21.

We did not encouter large noise components with frequencies different from the ones mentioned above. This is to be expected, since everything in railway technology has to meet very strict conditions. Even screws have to be certified to use. We did find modulation of the 75 Hz basis frequency, for instance at 225 Hz. This can easily be mistaken for noise. Further we noticed that in almost all measurements noise with a small amplitude was present. This looks like white noise, so we will assume it is. In some measurements, this noise was well above the value of 3 A set by the train detection theory.

3.3 Overview of measured lab data

In this Section we will describe the measurements as carried out at RIO, the training facillity of the Dutch railway operator NS. For sections without ATB, we tried to determine the relation between the resistance of the train (the contact rail-wheel-axle-wheel-rail) and the magnitude of the train detection current through the axle and the relation between the length of a section and the current through the rails. We will make a

Origin	Frequency	Strenght
Return current of traction	DC	< 4000 A
Component of return current	50 Hz	250 A
Resistance of traction installations	modulation by motor frequency	3 A
Chopper traction installations	modulation by motor frequency	1 A
	$66\frac{2}{3}$, 100, 300, 315, 400 and 450 Hz	5 A
Rectifiers, when asymmetric	50, 100 Hz	
Rectifiers, 6 and (semi) 12 pulses	300 Hz	
Rectifiers, 12 and (semi) 24 pulses	600 Hz	
ATB code from another section	All code frequencies	< 1 A

Figure 3.21: Possible noise signals

distinction between single- and double-legged sections. For an explanation on the difference, please refer to Section 2.1.

3.3.1 Measurements

Results single-legged sections

For various values of the variable resistor, the 'train', the current through the resistor is measured. We also keep track of the movement of the TR, the train detection relay. In the plots, this can be seen from the dotted line. It shows the separation between measurements for which the train was detected and measurements for which it was not. In some plots we can see there is a difference in the results if we position the 'train' near the supply or near the relay. The plots can be found in Figure 3.22. For every measurement, the resistance of the ballast is given in the caption. This resistance has influence on the currents going through the train axle. If the resistance of the ballast is low, a lot of current will be lost through the ballast. The train detection relay will therefore stay up (not detect the train) for a lesser amount of current through the axle. Of course the voltage over the tracks is still the same, so the total amount of current from rail to rail is the same, but not all of it goes through the train axle. If the resistance is high, more current will be flowing through the axle, resulting in a higher current through the axle of the train. From these plots we create a new plot, displaying the current for which the train was not detected anymore, as a function of the section length, for various values of the ballast resistance. For the single-legged sections, this plot is in Figure 3.23.

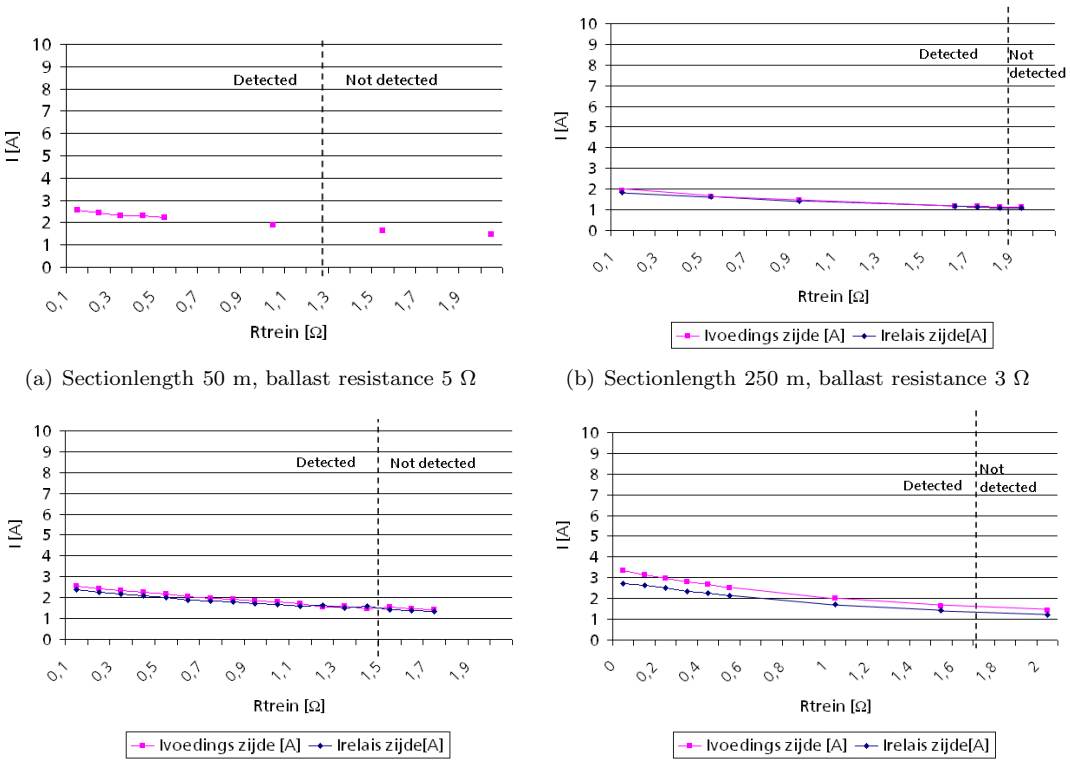
Results double-legged sections

Similar to the single-legged case, we carry out tests for double-legged sections. The same considerations hold as for the single-legged sections. The results of the tests are in Figure 3.24. Again we created an overview plot, showing the current through the axle as a function of the section length. Various values of the ballast resistance are taken into account.

Conclusions

We can conclude that for sections without ATB, there is a level of resistance of the train for which the train is no longer detected. There is a relation between this value and the current going through the axle of the train. Some factors of influence are single- or double-legged sections, the resistance of the ballast and the point in the section where the train is placed. In general we can say we found the following relations:

- The train detection current through the axle of the train decreases when the ballast resistance decreases. Ballast resistance can differ per section and per day, as temperature and moist have a large influence on it.
- The current at the supply side is larger than the current at the relay side.
- If the section length increases the current output at the supply increases. This is probably a result of the callibration of the equipment to provide enough power at the relay-end of the section, not a relation between the section length and the current.



(a) Sectionlength 50 m, ballast resistance 5 Ω

(b) Sectionlength 250 m, ballast resistance 3 Ω

(c) Sectionlength 500 m, ballast resistance 3 Ω

(d) Sectionlength 750 m, ballast resistance 3 Ω

Figure 3.22: The current through the train axle, single-legged sections

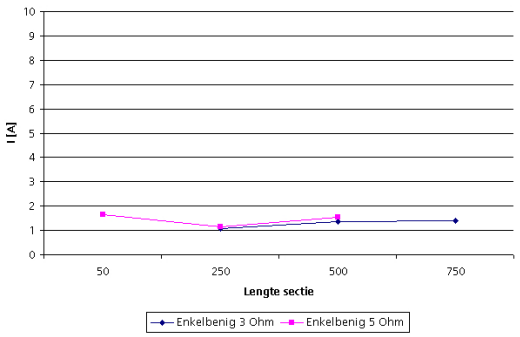
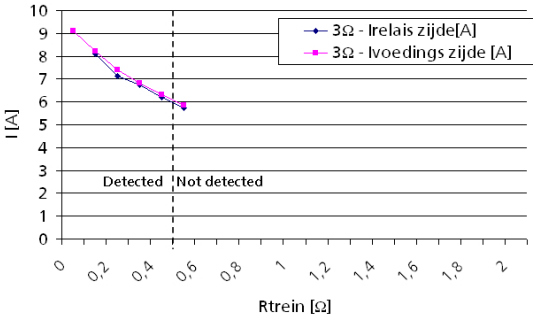
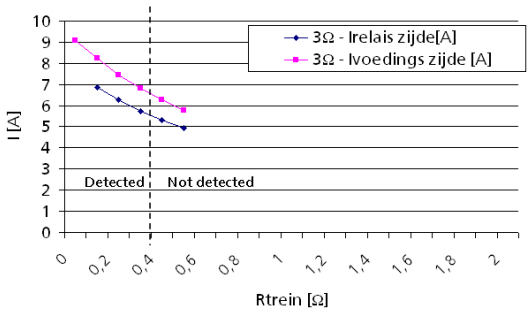


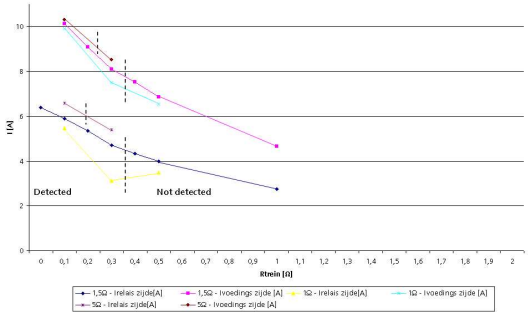
Figure 3.23: The current through the axle for which the train was not detected anymore



(a) Sectionlength 100 m, ballast resistance 3 Ω



(b) Sectionlength 400 m, ballast resistance 3 Ω



(c) Sectionlength 1000 m, ballast resistance 1.5 Ω

Figure 3.24: The current through the train axle, double-legged sections

Further, we can make the following remarks about the current through the axle of the train:

- For single-legged sections, the train detection current in case of detection is between 1 and 4 A. For double-legged sections, the train detection current for which the train is detected between 3 and 11 A.
- In several cases, the train was not detected for currents through the axle between 5 and 8.5 A. However, in the single-legged sections, the current was never more than 3.5 A.

3.3.2 Conclusions

We encouter difficulties when no ATB code is present in the tracks and we do want to say something about detection. Since we cannot find a code, we do not know which of the following we have at hand:

- (some of) the current of the train circuit current is going through the axles (no ATB code is installed on the tracks), its magnitude can vary from almost zero to 14.5 A,
- code 'no code' is supplied to the tracks, the train circuit is interrupted with no pulses per minute, this means the maximum speed is 40 km/h and is in fact the same as just the train circuit current,
- we have not been detected because the resistance between the wheels and the track is too large, resulting in almost no current through the axles of the train,
- we are driving on a part of the infrastructure where no train circuit is present, instead axle counters or pedals etc. are used. In this case if any current is measured it does not have the 75 Hz basis frequency.

In the first two cases it is likely a current of 1 A or more is going through the axles of the train, in the latter two cases it is likely the current through the axles is below 2 A.

We have tried to find a boudary value for the current through the axles in case we are not detected, so we could separate the cases 'detected' from 'not detected'. In other words, we search for a C such that if

$$\max(|\text{currenttroughaxles}|) > C$$

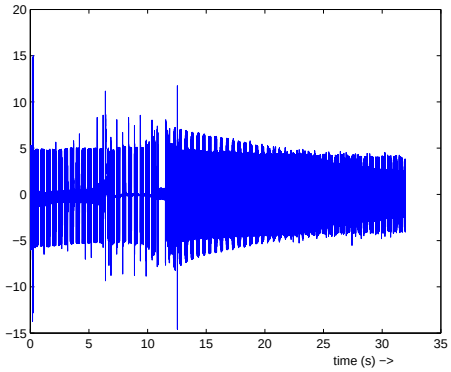


Figure 3.25: Worst dataset recorded

the train is detected. At RIO we simulated cases 1 and 2. We did find a large difference between single-legged and double-legged sections and a dependence on the resistance of the ballast. Unfortunately, we have no means (yet) to determine what section we are on, how large the actual resistance of the ballast is or if we are driving towards the supply or the relay. So we have to look for a general result. As we can see practically all currents from 1 to 11 A can occur signifying both 'detected' and 'not detected'. But we have a severe penalty on labeling 'detected' when we are not, since this results in an usafe situation. So if we would determine a boudary value with the current information, it would be really large ($C = 8.5$ or even larger) and it would result in always or almost always deciding 'not detected'. Thus it is of no practical use and this approach was put aside.

3.4 Theory and reality

In the measurements we see that the signal properties described in the theory match well with reality. All measurements have a basic frequency of 75 Hz $\pm$ 3 Hz. We encouter some data with a frequency of 71.92 Hz, but this is within the error margin imposed by the length of the measurements (see Chapter 4 and 8. Further, the code frequencies correspond to 1.25, 1.6, 2, 3 and $3\frac{2}{3}$ Hz, since the codes with frequency 2.45 and 4.5 Hz are currently not in use. The duty cycle is almost allways within the boundaries of 40 or 60 %. All code signals meet the requirement on the heigth of the current in the up period. However, we see several measurements where the low period is above 3 A. The worst example of this was recorded at 13.37h on a straight stretch (Figure 3.25).

Chapter 4

Analysis

In this chapter we will give a short outline of some mathematical tools needed to understand the measuring system developed in this thesis. It is assumed the reader is familiar with basic mathematical tools, to the level of a bachelor's student.

4.1 Continuous Fourier Transform

4.1.1 Fourier Integral Transform

The Fourier transform is a powerful tool in signal analysis. It is a transformation from the time domain to the frequency domain. We will denote the transformation by $\mathbb{F}$ or FT and the transformed signal by capital letters. Suppose $g(t)$ is absolutely integrable. The Fourier transform and its inverse are defined as:

$$\begin{aligned} \mathbb{F}[g(t)](\omega) = G(\omega) &= \int_{\infty}^{\infty} g(t)e^{-i\omega t}dt, \\ \mathbb{F}^{-1}[G(\omega)](t) = g(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} G(\omega)e^{i\omega t}d\omega. \end{aligned} \tag{4.1}$$

Here ω is the angular frequency, i is the imaginary unit $\sqrt{-1}$ and t is the time variable. We can think of the inverse transform as describing the continuous signal as a sum of infinitely many weighted harmonic functions. Angular frequency is measured in radians per second and is a variable often used for periodic phenomena. We can find a relation between ω and ordinary frequency f. Consider the scalar ω_0 in the expression $\cos(\omega_0 t)$. Hence one cycle takes $T = \frac{2\pi}{\omega_0}$ seconds, so $\frac{1}{T} = \frac{\omega_0}{2\pi} = f$ or:

$$\omega = 2\pi f. \tag{4.2}$$

The graphical representation of the absolute value of the Fourier Transform shows the frequency content or the spectrum: what frequencies are present in a signal and how strongly. Figure 4.1 shows a typical spectrum. If the spectrum is zero for all ω larger than some value B, then the smallest B with this property is called the bandwidth. It is the maximum frequency present in the signal.

$$|G(\omega)| = 0 \qquad \text{for } |\omega| > B \tag{4.3}$$

The absolute value of the Fourier Transform at a certain frequency is a measure for the power of the signal at that frequency. For example, if the absolute value of $|G(\omega_1)| = 0$ for $\omega_1 > B$, it means the vibration with frequency ω_1 has zero power. Another example is if the frequency 0 has some power, it indicates a constant component is present.

Example: Delta function

The Dirac-delta function $\delta(t-b)$ is the function which is large at b, zero elsewhere and has an area of one. We can view the delta function as a selector: if multiplied with another function, it 'selects' the value of the

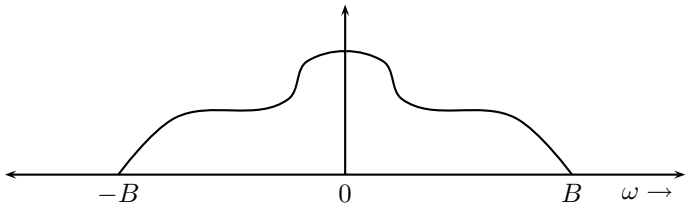


Figure 4.1: Frequency content of a signal.

other function at $t = b$. This is called the sifting property:

$$\begin{aligned} f(t)\delta(t-b) &= f(b)\delta(t-b), \\ \int_{-\infty}^{\infty} f(t)\delta(t-b)dt &= f(b), \end{aligned} \tag{4.4}$$

We can use this property to calculate $\mathbb{F}[\delta(t)]$ and the inverse Fourier Transform of $\delta(\omega)$:

$$\begin{aligned} \mathbb{F}[\delta(t)] &= \int_{-\infty}^{\infty} \delta(t)e^{-i\omega t}dt = e^{-i0t} = 1, \\ \mathbb{F}^{-1}[\delta(\omega)] &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \delta(\omega)e^{i\omega t}d\omega = e^{i\omega 0} = \frac{1}{2\pi}. \end{aligned} \tag{4.5}$$

With the help of the shift-rule, we can calculate the Fourier Transform of a shifted delta-function (in time or in frequency):

$$\begin{aligned} \mathbb{F}[\delta(t-t_0)] &= e^{-i\omega t_0}\cdot\mathbb{F}[\delta(t)] = e^{-i\omega t_0}, \\ \mathbb{F}^{-1}[\delta(\omega-\omega_0)] &= e^{-i\omega_0 t}\cdot\mathbb{F}[1] = \frac{1}{2\pi}e^{-i\omega_0 t}. \end{aligned} \tag{4.6}$$

Example: Cosine

A cosine has only one frequency present in its signal. The bandwidth of the signal is therefore equal to this frequency. We use the example about the delta function to calculate its Fourier transform:

$$\begin{aligned} \mathbb{F}[\cos(\omega_0 t)](\omega) &= \mathbb{F}[\frac{1}{2}(e^{i\omega_0 t} + e^{-i\omega_0 t})] \\ &= \frac{1}{2}\left(\mathbb{F}[e^{i\omega_0 t}] + \mathbb{F}[e^{-i\omega_0 t}]\right) \\ &= \frac{1}{2}\left(2\pi(\delta(\omega+\omega_0) + \delta(\omega-\omega_0))\right) \\ &= \pi\left(\delta(\omega+\omega_0) + \delta(\omega-\omega_0)\right). \end{aligned} \tag{4.7}$$

We see now that the Fourier transform of a sinusoid is, as expected, a scaled peak located at its frequency and a scaled peak located at minus its frequency (Figure 4.2).

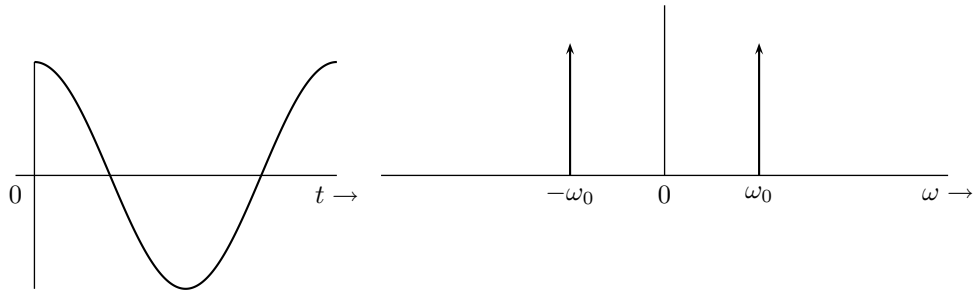
4.1.2 Convolution

The convolution of two signals $g(t)$ and $h(t)$ is defined as

$$(g * h)(t) = \int_{-\infty}^{\infty} g(\tau)h(t-\tau)d\tau. \tag{4.8}$$

The calculation of this integral is not alway straight-forward, but if the signals are Fourier-transformable we can simplify the calculations considerably:

$$\begin{aligned} \mathbb{F}[(g*h)(t)](\omega) &= \int_{-\infty}^{\infty} e^{-i\omega t}\left(\int_{-\infty}^{\infty} g(\tau)h(t-\tau)d\tau\right)dt \\ &= \int_{-\infty}^{\infty} g(\tau)\left(\int_{-\infty}^{\infty} e^{-i\omega t}h(t-\tau)dt\right)d\tau. \end{aligned} \tag{4.9}$$



(a) The graph of $\cos(\omega_0 t)$ (b) Absolute value of Fourier Transform

Figure 4.2: A sinusoid and its frequency content

We assumed we can interchange the order of integration. Since the Fourier transform of $h(t-\tau)$ equal is to the Fourier transform of $h(t)$ multiplied by $e^{-i\omega\tau}$ we can write (4.9) as:

$$\int_{-\infty}^{\infty} g(\tau)H(\omega)e^{-i\omega\tau}d\tau = G(\omega)H(\omega). \tag{4.10}$$

Now we see that convolution in the time-domain is equivalent to multiplication in the frequency-domain. The other way around we have that multiplication in the time-domain is equivalent to convolution in the frequency-domain:

$$\begin{aligned}
\mathbb{F}[g(t)h(t)](\omega) &= \int_{-\infty}^{\infty} g(t)h(t)e^{-i\omega t}dt \\
&= \int_{-\infty}^{\infty} \left(\frac{1}{2\pi}\int_{-\infty}^{\infty} G(v)e^{i\omega t}dv\right) h(t)e^{-i\omega t}dt \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} G(v)\int_{-\infty}^{\infty} h(t)e^{-i(\omega-v)t}dtdv \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} G(v)H(\omega-v)dv \\
&= \frac{1}{2\pi}(G*H)(\omega).
\end{aligned} \tag{4.11}$$

In summary:

Convolution in the time domain corresponds to multiplication in the frequency domain and vice versa:

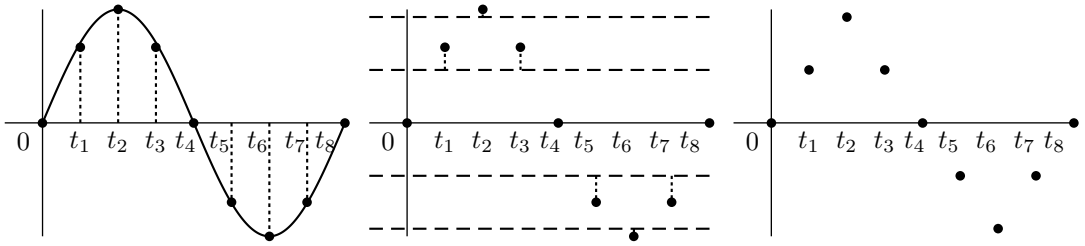
$$\begin{aligned}
\mathbb{F}[(g*h)(t)](\omega) &= G(\omega)H(\omega) \\
\mathbb{F}[g(t)h(t)](\omega) &= \frac{1}{2\pi}(G*H)(\omega).
\end{aligned} \tag{4.12}$$

4.2 The Sampling theorem

While all signals in the real world are continuous, computers are not. We will have to convert continuous signals (currents or voltages) to the discrete-time domain in order to use the computer for storage or numerical analysis. This process is called sampling. It is described in the first part of the Nyquist-Shannon Sampling theorem, also known as the 'Sampling Theorem'.

4.2.1 Sampling

If we sample a signal, we only measure it a finite number of times per second, thus at discrete points in time. Suppose we measure the signal with T seconds between two samples ($T > 0$). From the sampling period T



(a) Sampling (b) Quantization (c) Resulting signal

Figure 4.3: Converting a continuous to a discrete signal

the angular sampling frequency ω_s is defined as:

$$\omega_s = \frac{2\pi}{T}. \tag{4.13}$$

This sampling is often done by a Data Acquisition Unit, a DAC, containing an A/D-converter, an analogue/digital converter. Since this is a physical and not a mathematical instrument, the sampling *instants* do not have width zero, although they are very small compared to T. We will assume these intervals are negligibly small. Besides sampling, the signal is rounded off towards the values the A/D converter can handle. This is called quantization. The more precise values a converter can handle, the smaller the quantization error is. See Figure 4.3 for an illustration of both sampling and quantization.

4.2.2 Reconstruction

An important question is now, can we reconstruct the original signal from the samples? Under certain conditions on the sampling frequency, the answer is yes, we can. This is the conclusion from the second part of the Sampling Theorem.

Consider a signal $g(t)$ with bandwidth B. We sample this signal at sampling frequency $f_s = 1/T$. To obtain a continuous function, we do the sampling by multiplying the signal with a Dirac-comb $\Delta_T(t)$, with the teeth T seconds apart:

$$g_s(t) = g(t) \cdot T \cdot \Delta_T(t) \tag{4.14}$$

Here g_s represents a function only depending on values of $g(t)$ at discrete instants in time, the sample values. Since the delta-Dirac function is periodic with period T, it equals its Fourier series. A Fourier series is the representation of a signal by an infinite summation of harmonics:

$$\begin{aligned} f(t) &= \sum_{k=-\infty}^{\infty} c_k e^{ik2\pi t/T} \\ \text{with } c_k &= \frac{1}{T}\int_{t_0+T}^{t_0} f(t)e^{-ik2\pi t/T} \end{aligned} \tag{4.15}$$

The c_k are called the Fourier constants. For the Dirac comb, these constants are

$$\begin{aligned} c_k &= \frac{1}{T}\int_{t_0}^{t_0+T} \Delta_T(t)e^{-ik2\pi t/T} \\ &= \frac{1}{T}\int_{-T/2}^{T/2} \Delta_T(t)e^{-ik2\pi t/T} \end{aligned} \tag{4.16}$$

Since there is only one tooth of the comb present in this integration interval

$$\begin{aligned} c_k &= \frac{1}{T}\int_{-T/2}^{T/2} \delta(t) e^{-ik2\pi t/T} \\ &= \frac{1}{T} e^{-ik2\pi 0/T} \\ &= \frac{1}{T} \end{aligned} \tag{4.17}$$

So the Fourier series of the Dirac comb is equal to

$$f(t) = \frac{1}{T}\sum_{k=-\infty}^{\infty} e^{ik2\pi t/T}. \tag{4.18}$$

Now we can write (4.14) as (see [15]):

$$\begin{aligned} g_s(t) &= g(t)T\sum_{k=-\infty}^{\infty}\delta(t-kT) \\ &= T\sum_{k=-\infty}^{\infty} g(kT)\delta(t-kT) \\ &= T\cdot\frac{1}{T}\sum_{k=-\infty}^{\infty} g(kT)e^{i2\pi kt/T} \\ &= \sum_{k=-\infty}^{\infty} g(kT)e^{i\omega_s kt}. \end{aligned} \tag{4.19}$$

Use the shifting property:

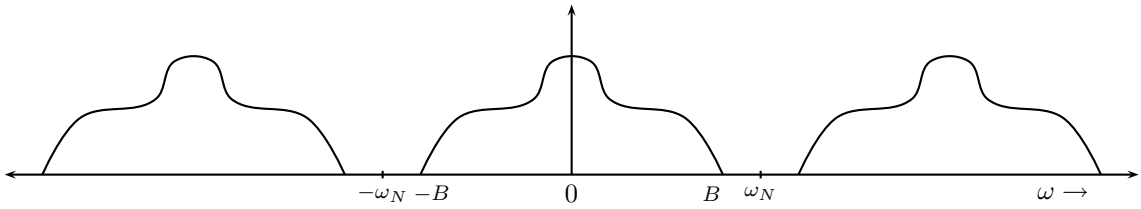
$$\begin{aligned} \mathbb{F}[g_s(t)] &= \mathbb{F}\left[\sum_{k=-\infty}^{\infty} g(kT)e^{i\omega_s kt}\right] \\ &= \sum_{k=-\infty}^{\infty}\mathbb{F}\left[g(kT)e^{i\omega_s kt}\right] \\ &= \sum_{k=-\infty}^{\infty} G(\omega - k\omega_s) \end{aligned} \tag{4.20}$$

Note that with using only the sample values, we obtain an infinite number of copies of the spectrum, called images, ω_s apart. The spectra are $2B$ wide but spaced ω_s, Figure 4.4(a). Thus there is no overlap and hence perfect reconstruction possible if the sampling frequency is larger than twice the bandwidth of the signal:

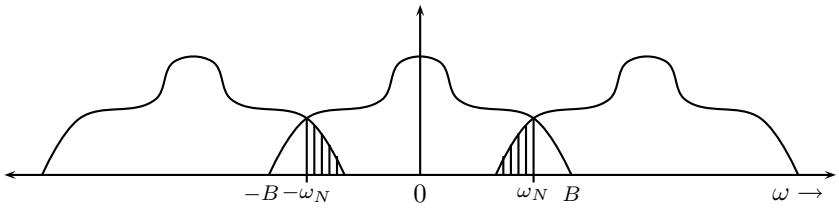
$$\omega_s > 2B. \tag{4.21}$$

The frequency $\frac{1}{2}\omega_s$ is commonly referred to as the Nyquist frequency ω_N, so we can write (4.21) as $\omega_N > B$. ω_N is a property of the sampled signal. The original signal can be retrieved by filtering the infinite spectrum with a filter (see Section 4.4) to separate the first spectrum from the images and then using the inverse Fourier transform. The filtering should select all values up to some frequency between B and ω_N. In that interval the spectrum is zero, since $B < \frac{1}{2}\omega_s$. Note that this is also the maximal frequency that can be captured by the sampled and reconstructed signal. For the filtering we multiply the infinite spectrum by the function

$$\begin{cases} 1 & |\omega| < \omega_s \\ 0 & |\omega| > \omega_s \end{cases}. \tag{4.22}$$



(a) No overlap if $\omega_N > B$



(b) Overlap if $\omega_N < B$

This is a rectanglar function we will denote by $\text{rect}_{[-B,B]}$. Its inverse Fourier transform equals

$$\begin{aligned}
\mathbb{F}^{-1}[\text{rect}_{[-\omega_s/2,\omega_s/2]}] &= h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \text{rect}_{[-\omega_s/2,\omega_s/2]}(\omega) e^{i\omega t} d\omega \\
&= -\frac{1}{2\pi i\omega t}\left[e^{i\omega t}\right]_{-\omega_s/2}^{\omega_s/2} \\
&= \frac{1}{\pi t}\frac{e^{i\omega_s t/2} - e^{-i\omega_s t/2}}{2i} \\
&= \frac{\sin(\omega_s t/2)}{\pi t}.
\end{aligned} \tag{4.23}$$

So the reconstruction of $g(t)$ from its sampled signal $g_s(t)$ equals the iFT of the rectangle convoluted with $g_s(t)$:

$$\begin{aligned}
g(t) &= h(t) * \sum_{k=-\infty}^{\infty} g(kT) T\delta(t - kT) \\
&= \sum_{k=-\infty}^{\infty} g(kT) T h(t) * \delta(t - kT) \\
&= \sum_{k=-\infty}^{\infty} g(kT) T \frac{\sin(\omega_s(t - kT)/2)}{\pi(t - kT)} \\
&\overset{(\omega_s = 2\pi/T)}{=} \sum_{k=-\infty}^{\infty} g(kT) \frac{\sin(\omega_s(t - kT)/2)}{\omega_s(t - kT)/2}.
\end{aligned} \tag{4.24}$$

We have now arrived at Shannon's Sampling Theorem. For completeness we give the full theorem.

Shannon's Sampling Theorem

Let $g(t) \overset{\mathbb{F}}{\leftrightarrow} G(\omega)$ and suppose $g(t)$ is bandlimited with bandwidth B. Let $g(kT)$, $k \in \mathbb{Z}$ and T the sampling time, be the sampled signal of $g(t)$ with angular sampling frequency $\omega_s = 2\pi/T$. If $\omega_s > 2B$ then $g(t)$ is uniquely determined by its samples $g(kT)$ by:

$$g(t) = \sum_{k=-\infty}^{\infty} g(kT) \frac{\sin(\omega_s(t - kT)/2)}{\omega_s(t - kT)/2} \tag{4.25}$$

An alternate proof of the Sampling Theorem starts with the Fourier series of the band limited frequency signal. Consider the signal as periodic with period ω_s:

$$G(\omega) = \sum_{n=-\infty}^{\infty} G_n e^{in\omega T} \quad \omega \in (-\omega_s/2, \omega_s/2). \tag{4.26}$$

Since $G(\omega) = 0$ for $\omega > \omega_s/2$, we can multiply with a rectangle function:

$$G(\omega) = \sum_{n=-\infty}^{\infty} G_n e^{in\omega T} \text{rect}_{[-\omega_s/2,\omega_s/2]}. \tag{4.27}$$

We calculate the Fourier constants according to Equation 4.15

$$\begin{aligned} G_n &= \frac{1}{\omega_s}\int_{-\omega_s/2}^{\omega_s/2} G(\omega)e^{-in\omega T} \\ &\overset{bandlimited}{=} \frac{1}{\omega_s}\int_{-\infty}^{\infty} G(\omega)e^{-in\omega T} \\ &= \frac{2\pi}{\omega_s}\underbrace{\frac{1}{2\pi}\int_{-\infty}^{\infty} G(\omega)e^{i\omega(-nT)}}_{FT^{-1}} \\ &= \frac{2\pi}{\omega_s} g(-nT) \\ &= Tg[-n], \end{aligned} \tag{4.28}$$

where $g[n]$ is the sample at time $-n$. Now substitute k for -n. Equation 4.27 becomes

$$G(\omega) = T\sum_{k=-\infty}^{\infty} g[k]e^{ik\omega T} \text{rect}_{[-\omega_s/2,\omega_s/2]}. \tag{4.29}$$

We can already see we can express $G(\omega)$ in terms of the samples only, without loss of information. From the previous proof we know

$$\frac{1}{T}\text{sinc}(\omega_s t/2) \overset{\mathbb{F}}{\leftrightarrow} \text{rect}_{[-\omega_s/2,\omega_s/2]}. \tag{4.30}$$

By the shift-rule we have

$$\frac{1}{T}\text{sinc}(\omega_s(t-nT)/2) \overset{\mathbb{F}}{\leftrightarrow} e^{-in\omega T}\text{rect}_{[-\omega_s/2,\omega_s/2]}. \tag{4.31}$$

Using this last equation, we back Fourier Transform $G(\omega)$ and find the reconstruction of the time-signal using just the sample values:

$$\begin{aligned} g(t) &= T\sum_{k=-\infty}^{\infty} g[k]\frac{1}{T}\text{sinc}(\omega_s(t-kT)/2) \\ &= \sum_{k=-\infty}^{\infty} g[k]\text{sinc}(\pi(t/T-k)) \end{aligned} \tag{4.32}$$

4.2.3 Aliasing

The overlap of the original spectrum with the images causes a phenomenon called aliasing. It causes frequencies higher than the Nyquist frequency to be 'aliased' (mapped) into the interval $(-\omega_N, \omega_N)$. In other words, fast vibrations appear to be slow vibrations. An example is a car with spoked wheels: at certain speeds the wheels appear to be turning forward slowly (or even backwards). For sinusoids, this is illustrated

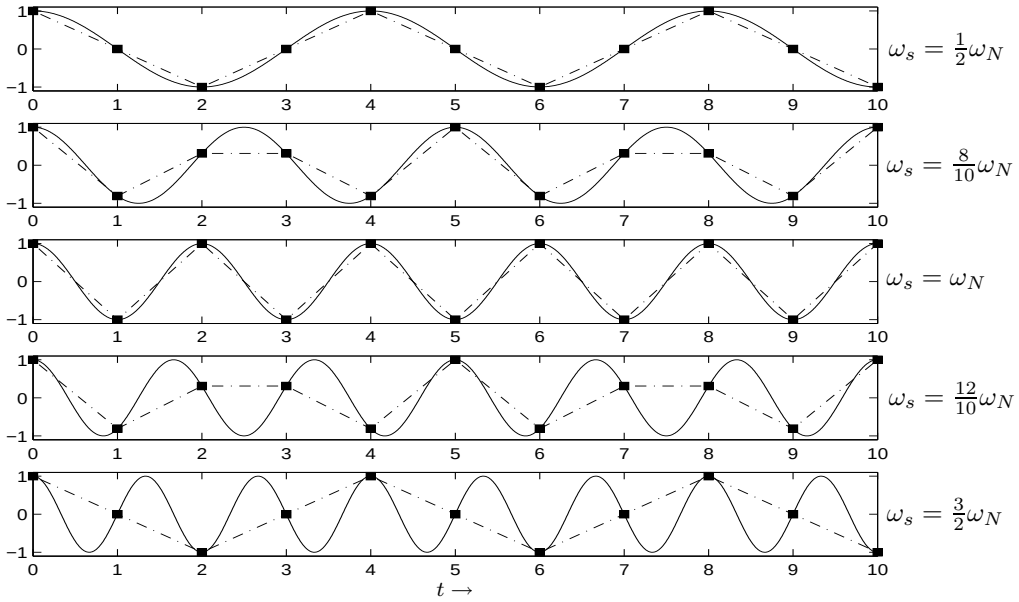


Figure 4.4: Example of aliasing

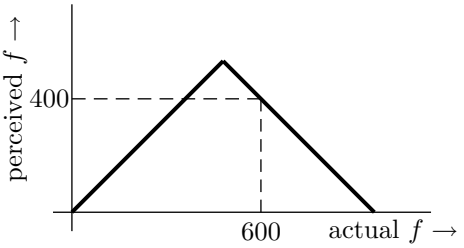


Figure 4.5: Appearance of aliased signals at sampling frequency f_s 500 Hz

in Figure 4.4. The sine appears to have different frequencies than its own for low sampling frequencies. Note that for $\omega_s = \omega_N$ we can shift the sample instants in such a way we only sample where the signal equals zero.

There is a relation between the actual frequency to the perceived or aliased frequency. Figure 4.5 shows this rule of thumb for the appearance of aliased frequencies. Aliasing can be a serious problem in signal analysis, since it causes us to perceive frequencies which are not there. This can lead us to wrong conclusions. A solution would be to increase the sampling rate, or filter the data with a lowpass filter to exclude high-frequency components before sampling.

4.3 Discrete Fourier Transform

Discrete Fourier Transform

As we saw, in most practical cases we do not have continuous signals. Instead, we are dealing with not only sampled but also finite versions of the infinite mathematical signals. These are discrete signals stored in arrays with finite length. Say we have the finite set of samples $g_0, g_1, \cdots, g_{N-1}$. For this there is a finite, discrete version of the theorem, the Discrete Fourier Transform (DFT, see for instance [16]) and its inverse,

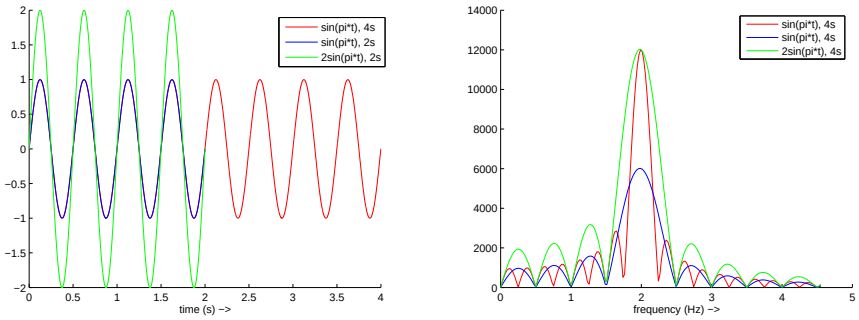


Figure 4.6: Three finite sines (left) and the magnitude of their Fourier transforms

defined as:

$$G(2\pi k/N) = \frac{1}{T}\sum_{n=0}^{N-1} g_n e^{-\frac{i2\pi kn}{N}} \qquad k = 1, \ldots, N-1 \tag{4.33}$$

$$DFT^{-1}[G(k)] = \frac{1}{M}\sum_{k=0}^{M-1} G(2\pi k/N) e^{\frac{2\pi i}{M}nk} \qquad n = 1, \ldots, M-1.$$

Here $2\pi k/N$ is the discrete angular frequency, N is the number of samples (in both time and frequency) and g_n are the sample values. Note that (in this case) n/N are the times the samples are taken. The DFT is an estimation of the FT of the analogue signal. In the DFT we sum over a finite number of time samples, introducing an uncertainty: the resolution in frequency depends on the length of the interval the DFT is taken over. As a rule of thumb, we can say the frequency resolution equals $\frac{1}{T_f}$ where T_f is the length (in seconds) of the interval over which the DFT is calculated. In formula, where R_f is the resolution in frequency:

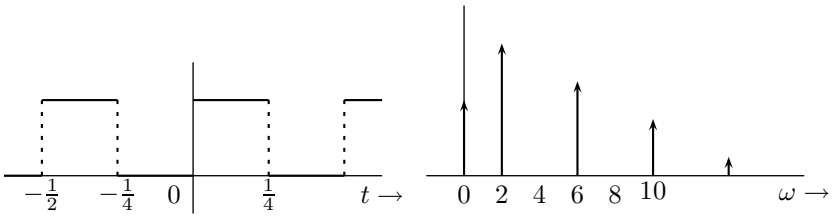
$$R_f = \frac{1}{T_f}. \tag{4.34}$$

Fast Fourier Transform

The Fast Fourier Transform is a collective term for several efficient algorithms to calculate the DFT [17]. It can also be used for the inverse DFT because of the similarity it has to the DFT. The most efficient FFT algoritm uses a base 2 of Cooley-Turkey [17]. It uses factorisation of N and is fastest if L is a integral power of 2, say $N = 2^M$. The main principle of the FFT is that for $N = N_1 N_2$ we can express the FFT in terms of two smaller FFT's with lengths N_1 and N_2. We can repeat this, thus using the fact that N can be decomposed onto all factors two. It may be shown that instead of calculating N times N terms, we only have to calculate $M = \log N$ times N terms. So we reduced N^2 computations to $N \log N$ computations.

Example: Finite sinusoid

Suppose we want to calculate the transform of a finite length sinusoid. From Equation (4.34) we know we need a certain duration to obtain certainty about the frequency. We will see that the longer the signal, the higher and narrower the peak around its frequency, see Figure 4.6. The red sine is twice as long as the blue and green sine. When we look closely, we see that the main lobe of the red sine is about 1/2 Hz wide, the main lobes of the blue and green sine are 1 Hz wide. So the location of the peak for 4s of data may deviate 1/4 Hz to either side, for the blue and green sine the deviation can be 1/2 Hz, which agrees with the rule of thumb (4.34). Notice that an increased amplitude increases the magnitude of all of the spectrum, but does not decrease the uncertainty in frequency.



(a) A square wave (b) Absolute value of Fourier Transform

Figure 4.7: A square wave and its frequency content

Example: Square wave

A somewhat more difficult example is the infinite length square wave $s(t)$ with period $\frac{1}{2}$.

$$\begin{aligned} s(t) &= \sum_{-\infty}^{\infty} \mathbb{I}_{A_k}(t) \\ A_k &= \left[\frac{2k}{4}, \frac{2k+1}{4}\right), \quad k \in \mathbb{Z} \end{aligned} \tag{4.35}$$

Where $\mathbb{I}_{A_k}(t)$ is the indicator function which is 1 where $t \in A_k$ and 0 elsewhere (Figure 4.7(a)). The Fourier series of a signal is an approximation by an infinite sum of weighted harmonics, [16], Chapter 4. It is known that the Fourier series of a square wave with period $\frac{1}{2}$ is:

$$s(t) = \sum s_k e^{-ik\omega_0 t}, \quad k = 0, \pm 1, \pm 3, \pm 5, \ldots \tag{4.36}$$

where

$$\begin{aligned} \omega_0 &= 4\pi, \\ s_k &= \begin{cases} 0 & \text{if k is even;} \\ \frac{2}{ik\pi} & \text{if k is odd.} \end{cases} \end{aligned} \tag{4.37}$$

Now if we take the Fourier Transform of the expression above and use what we know about the shifted delta function, we obtain:

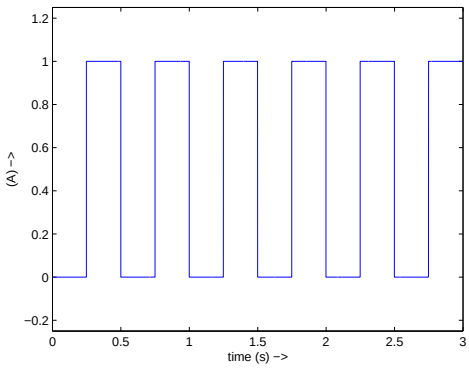
$$\mathbb{F}[s(t)] = \sum s_k \delta(\omega - \omega_0 k) \tag{4.38}$$

If we plot the magnitude of the Fourier Transform, it looks like a train of pulses, with peaks located at $t = 0$, at $t = 2$, $t = 6$, $t = 10$ and so on (Figure 4.7(b)).

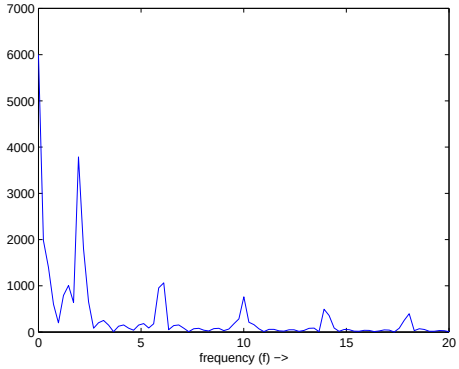
Example: Finite rectangular pulse, convolved with a finite sinus

Now consider our two examples: the finite sine and the finite blokpulse. What happens if we multiply these, or convolve them in the frequency domain (Figure 4.8)? First we look at the time domain. If we multiply a rectangular pulse with a cosine, we get a signal that is zero wherever the blokpulse is zero, and sinusoidal where the blokpulse equals one. In the frequency domain we see something called modulation: appearance of small peaks around a larger one. In this case the large peak is a result of the cosine, since we have more periods of it in view. The small peaks result from the rectangular pulsen, we only have 6 of those in our timespan. Now look at the spectra of the two original signals. Convolution can be seen as flipping one of the graphs and 'sliding it over' the other graph from left to right and calculate the area which is beneath both (and not one) graphs. We obtain Figure 4.8(f).

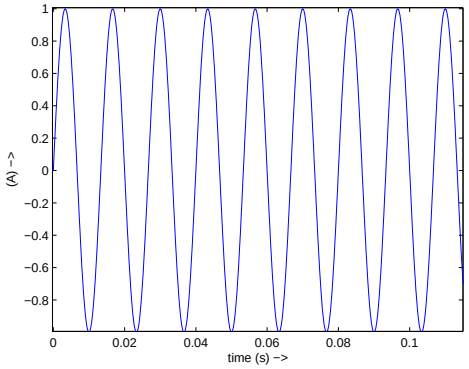
We will see that we can identify spectra like in Figure 4.8(f) in our measurements, resulting from modulation of a cosine by a rectangular pulse.



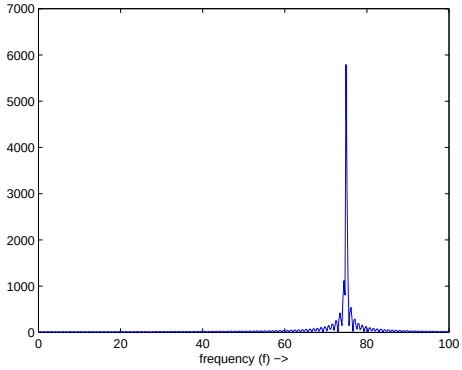
(a) Finite sequence of rectangular pulses



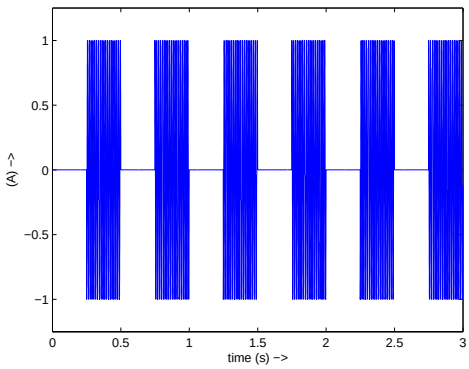
(b) Fourier transform blok



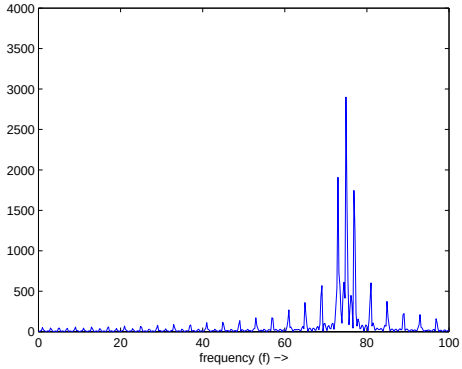
(c) Finite sine (detail)



(d) Fourier transform sine



(e) Product of the signals



(f) Fourier transform product

Figure 4.8: Multiplication of two finite signals

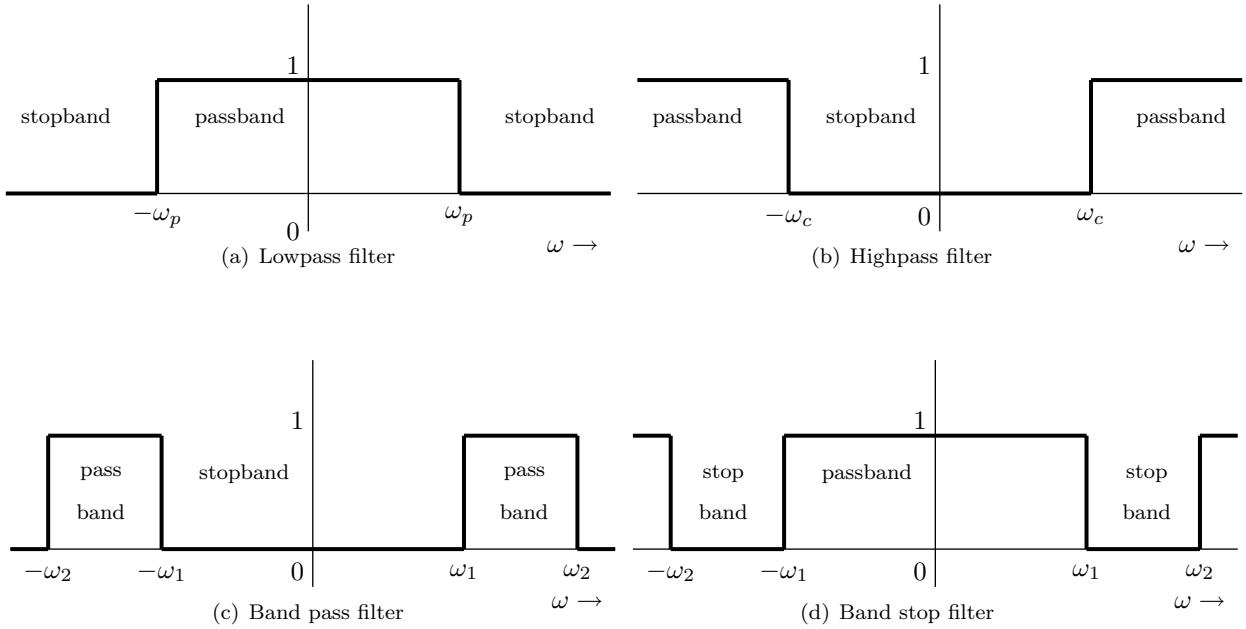


Figure 4.9: Four types of ideal filters

4.4 Filtering

Filtering is the general name for methods used to select certain information, remove certain frequencies from or increase gain or delay in a (noisy) signal. It is used to improve speech transfer in mobile communication, to percieve the heartbeat of a pregnant woman's child, to enhance music files, to monitor the condition of mechanical systems, brighten pictures, and so on. In this section we will discuss the mathematical mechanism behind it. A more elaborated description can be found in [16].

4.4.1 Ideal filters

In Section 4.1.1 we described what the spectrum of a signal is: what frequencies are present and how strongly. We also stated it can be obtained by taking the absolute value of the Fourier Transform. Further more, we have seen in Section 4.3 how two signals can be convoluted in time to obtain a new signal. This is also how most filters works. In Figure 4.9 the spectra of four types of ideal filters are shown. By multiplying these with the Fourier Transform of some signal, they pass or stop certain frequencies unchanged resp. perfectly. This is why they are called ideal filters. They are not physically realizable, since they require information on the signal from times in the future (the filters are non-causal).

4.4.2 Non-ideal Butterworth filters

We cannot build an ideal filter except in a computer, but we can approximate one. In this thesis we use the Butterworth filter. It can be built from resistors and capacitors. One of the biggest advantages is that the magnitude of its FT $|H(i\omega)|$ is monotone (it has no ripples) so it does not amplify frequencies in the stopband irregularly. Moreover, the Butterworth filters are 'maximally flat'. The expression for the Butterworth filter is quite complicated, but the absolute value of its Fourier Transform is quite simple:

$$|H(\omega)| = \frac{1}{\sqrt{1 + (\omega/\omega_c)^{2N}}} \tag{4.39}$$

Here ω_c is the cut-off frequency, ω/ω_c is the normalized angular frequency, such that the (low)passband is $[-1, 1]$. N is called the order of the filter. Figure 4.10 shows the Bode plot of some Butterworth filters with different order. A Bode magnitude plot is a displaying $|H(i\omega)|$ as a function of ω. A Bode phase plot

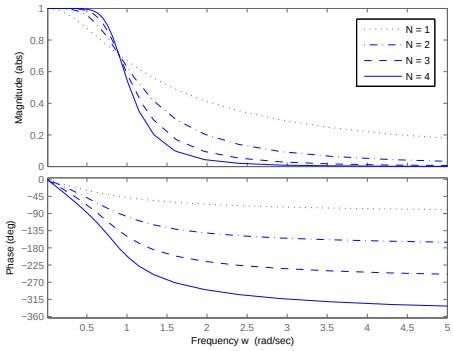


Figure 4.10: Bode magnitude and phase plots for 1^{st} till 4^{th} order Butterworth filter for $\omega_c = 1$

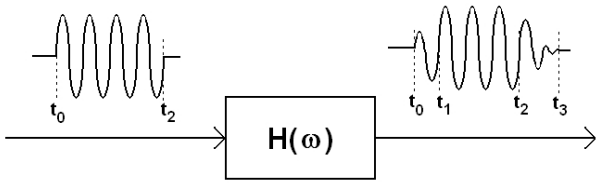


Figure 4.11: A signal passing through a filter

displayes the phase of $H(i\omega)$ as a function of ω. From such a plot we can easily see the steepnes of the filter, its passband and its stopband. An important property of a filter is its speed. With this, we mean the time the output of the filter needs to follow the input. This is illustrated in Figure 4.11. The speed of the filter is equal to $t_1 - t_0$ or $t_3 - t_2$. The order of the filter is an important factor in the speed of the filter. As we can easily see from the Bode phase plot in Figure 4.10, the higher the order, the larger the phase shift. Phase shift is in fact a delay on the original signal: the first input point with its phase appears later in time. So there is an obvious trade off between the speed and the steepness of the filter. A steep filter needs to be a high order, but this makes it a slow filter.

Ofter the poles of a filter can also be a good indication of the filter speed, especially in the long term. The decrease and increase in power of the output signal can, in case of a discrete filter, be described by λ^n, where λ are the poles of the transfer function of the filter and n are the samples. So the filter is faster if the poles are further from the unit circle and closer to the origin. The positions of the poles on their turn are influenced by the steepness, flatness and order of the filter: the steeper and flatter, the higher the order needed. All three factors give poles closer to the unit circle.

Chapter 5

Development

In this Chapter one discarded approach and the (development of the) preliminary Matlab script, used to analyse the data, will be described. This was scipt used as a starting point to develop the final model. The purpose of the program is to determine whether or not the train has been detected. We assume the train is not detected if we do not measure any code. Thus we label 'unsafe' in all four cases described in Section 3.3.

5.1 Matched filtering

We are trying to recognize a well-defined current from a noisy signal. In general, the optimal way to do this is by matched filtering. For this type of filtering, we define a template (the current we are trying to detect), reverse it in time and convolute it with the noisy signal. Then we can divide by the norm of the template to scale the filtered signal to range [0,1]. In discrete time this is

$$y(n) = \frac{1}{\|t\|} \sum_{\infty}^{k=-\infty} t(n-k)g(k) \tag{5.1}$$

where y is the filtered signal, t is the template and g is the noisy signal. When the template is detected in the noisy signal, the filtered signal is near 1. This seems like the perfect approach to our problem. There are two problems, however. First, the basic frequency can vary from 72 to 78 Hz, so we cannot construct the perfect template without knowing the frequency more precise, which takes time (see also Section 5.2). The second problem is more important: the matched filtering approach does not respond well to phase shifts. But as we saw in Chapter 3, phase shifts occur all the time: at section transitions, somtimes at every pulse. Since the shift at every pulse does not occur in all data and is not always by the same amount, we cannot include it in our template. So we conclude the matched filtering approach is not suited for this particular detection problem.

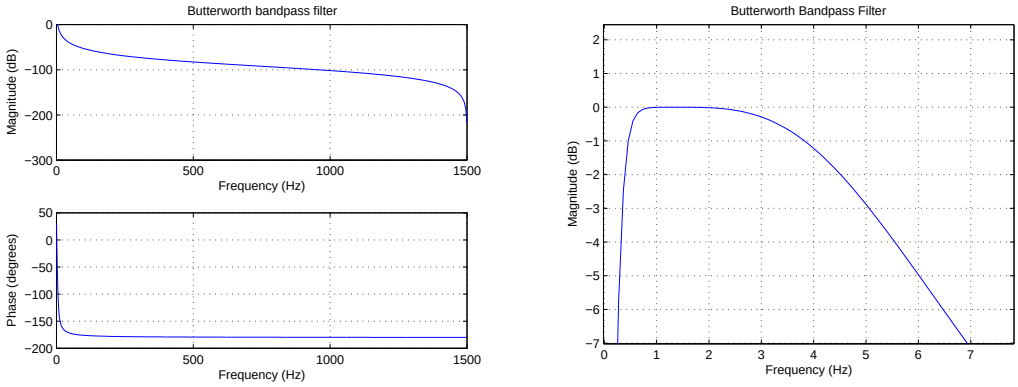
5.2 ATB - Lowpass Butterworth filter and FFT analysis

In our first approach, we based our program on the analysis of the Fourier Transform of the data. We should say: the approximation of the Fourier Transform by the Fast Fourier Transform with a base 2 Cooley-Turkey algorithm (see Matlab help for 'fft'). Now remember the properties of the signal we are trying to detect:

- Basic frequency 75 Hz, +/- 3 Hz,
- The duty cycle should be between 40 and 60 %,
- The high period of the ATB current should be above 6.5 A, the down-period should be below 3 A,
- Code frequencies are by switching current on and of with 1.25, 1.6, 2, 2.45, 3, $3\frac{2}{3}$ or 4.5 Hz.

Passband (normalized)	$[.5, 3.5]/(f_s/2)$
Stopband (normalized)	$[0, 0.001]/(f_s/2)$
Noise attenuation in crossover	20 dB
Transfer function	$10^{-4} \cdot \frac{0.2425-0.4850z^{-2}+0.2425z^{-4}}{1-3.9860z^{-1}+5.9581z^{-2}-3.9583z^{-3}+0.9861z^{-4}}$

Figure 5.1: Specification of Butterworth filter



(a) Bode plot of used filter

(b) Detail of Bode magnitude plot

Figure 5.2: Bode magnitude and Bode phase plots for the bandpass filter design

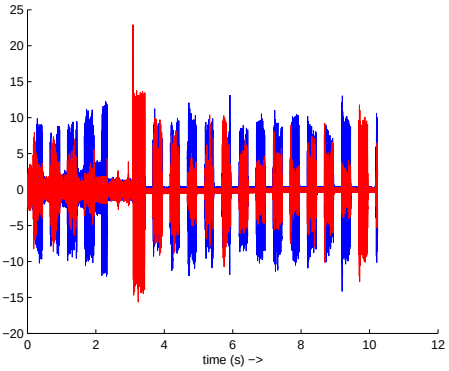
We check for the code by filtering the absolute value of the data with a lowpass Butterworth filter. The absolute value is taken to ensure a peak in the Fourier Transform at zero, this causes modulation with the code-frequency about zero and thus increases the visibility. A Butterworth filter does not have the steepest roll-off possible in the crossover frequencies, but it has no ripples in the pass- and stopband (Section 4.4). The specifications of the designed filter are described in Figure 5.1 and the Bode plot is in Figure 5.2. After the filtering we calculate the FFT of the filtered data and visually check if there is a peak at one of the code frequencies.

Now we apply this method to an example, Figure 5.3. Figure 5.3(a) shows the raw data. First we remark the maximum of the current is over 4 A, since both data series are above this value. Then we notice the duty cycle is about 60 %, that is close to the error margin. Now consider Figure 5.3(b) and Figure 5.3(c). We see here the Fast Fourier Transform of our data series. A large peak is situated at 72 Hz. This is in the interval (72,78), so we conclude this is the peak resulting from the strong presence of the basis frequency. There is not much high-frequency noise. One small peak can be seen at 216, which is resulting from the third harmonic (multiple) of 72 Hz. Now we checked the three preconditions and we can apply the Butterworth filter to the data.

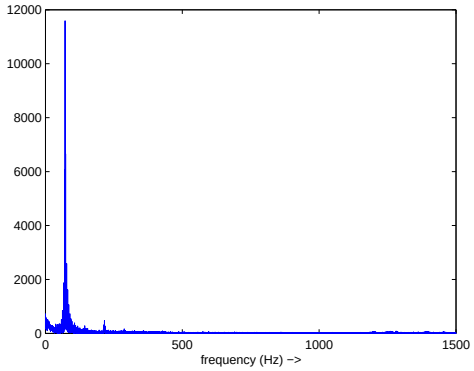
Figure 5.3(d) shows the data after filtering all frequencies above 4 Hz away. Here we can already see if there is some low-frequency signal present in the data, even though it is noisy. Now we calculate the FFT of this signal, Figure 5.3(e). By visual inspection or comparison to a reference value, we conclude code 120 (corresponding to 2 Hz) is present in this signal.

5.2.1 Disadvantages

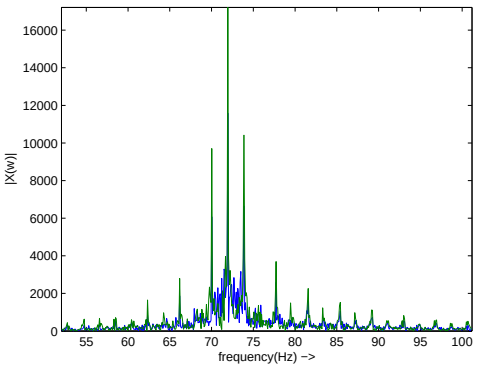
This method has several disadvantages. The most prominent is that the we need a certain timespan to obtain sufficient resolution in frequency. A rule of thumb states that, to distinguish two frequencies lying 0.35 Hz apart (code 75 and code 96), we need a timespan of $\frac{1}{0.35/2} \approx 5.7$s. This is a very long time, a train driving at 120 km/h would travel almost 200 meters is that timespan. This is unacceptable. A second disadvantage is that the use of a Fourier Transform in a product requires a computer or at least a chip. Since the purpose of this research is producing a small and cheap system for installation on a train, it is useful to investigate a different method of determining the code. Thirdly, Fourier analysis has very low time resolution. In Example 3.2.1, we saw an example of changing code (Figure 3.10). In Figure 5.4 the result of



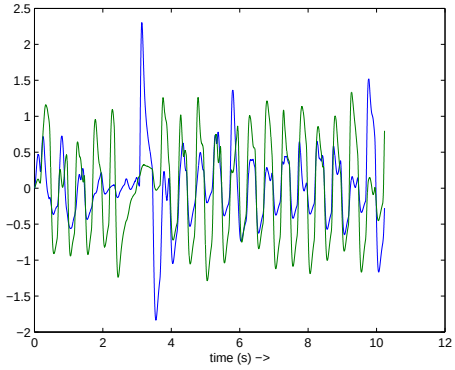
(a) Raw data, red is first axle, blue is second axle



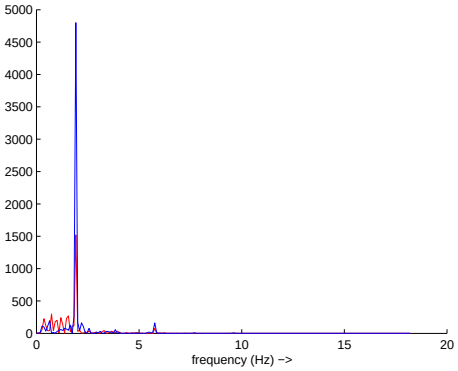
(b) A typical (Fast) Fourier Transform. Peak resulting from basis frequency at 72 Hz, small peak at harmonic frequency $3 \cdot 72 = 216$ Hz



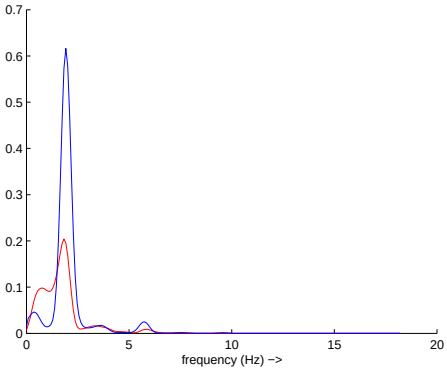
(c) Detail of FFT



(d) Filtered data



(e) Fourier Transform after filtering



(f) Smoothed FFT after filtering

Figure 5.3: Example of data filtered with a bandpass Butterworth filter

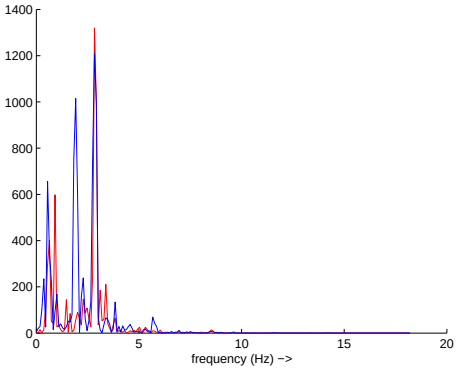


Figure 5.4: FFT of change in code after filtering

filtering with our Butterworth filter is shown. From it, we cannot determine at what time what code was present. Moreover, the two codes in the same measurement result in less certain detection of both. We want to be able to detect a change in code. This result also implies that when in a measurement of 10 seconds the train was detected for 5 seconds and not detected for the other 5 seconds, we would not notice that. One could consider using the Short Time FT, which can show a change in time. Unfortunately with this algoritmh, we have to choose between precision in time and frequency. We need both, so the STFT is no solution either. So this method of analysis does not suffice for our purposes.

5.3 ATB - Multiple filters and energy content

We now develop a program which does not necessarily has to use the Fourier Transform. For that reason we do not consider the magnitude of the Fourier Transform, but the power of the filtered signal. For if the filtered signal has low power, the frequency the filter can pass through was apperently not (strongly) present in the signal. This program will not be real time, it will be able to analyse stretches of measuring data. It is designed with the intention to rewrite it to real-time (or close to real-time).

5.3.1 Code filtering

We design seven Butterworth filters: one for every code, alltough only five codes are currently in use. To distinguish between this type of filtering and the previous, we will refer to this filtering for every code separately as 'code filtering'. We designed these filters to be very tight; only a small interval around every code frequency is passed through. Since all codes are allowed to deviate just 0.05 Hz from their nominal value, we take 2*0.05 Hz as the width of the passband. To achieve this steepness, we need 4^{th} order filters. The specifications are summarized in Figure 5.5. We filter the absolute value of the data of N samples long with all of these filters separately, to obtain 7 new data series. For each series holds: if the frequency of the applied filter was present, the resulting signal x will oscillate with some amplitude. If the frequency was not or barely present, the resulting signal will have almost no power. We therefore calculate an estimation of the power of the signal, the 'Root Mean Square':

$$RMS(x) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

Since we do not yet need a lot of accuracy in this (that only takes time), we calculate the rms once per second. Then the signal with the largest RMS is the current code, if it is strong enough and meets the other requirements on the basic frequency, the current level and the duty cycle. We now statistically determine a boundary value for the power of a filtered signal. This boudary value services to distinguish between actual

Passband (normalized)	$[f_c - 0.05, f_c + 0.05]/(f_s/2)$
Stopband (normalized)	$[f_c - 0.25, 0.001]/(f_s/2)$
Noise attenuation in crossover	12 dB
Transfer functions:	
code 75	$10^{-6} \cdot \frac{0.0602 - 0.1204z^{-2} + 0.0602z^{-4}}{1 - 3.9993z^{-1} + 5.9979z^{-2} - 3.9979z^{-3} + 0.9993z^{-4}}$
code 96	$10^{-6} \cdot \frac{0.0622 - 0.1244z^{-2} + 0.0622z^{-4}}{1 - 3.9993z^{-1} + 5.9978z^{-2} - 3.9979z^{-3} + 0.9993z^{-4}}$
code 120	$10^{-6} \cdot \frac{0.0637 - 0.1275z^{-2} + 0.0637z^{-4}}{1 - 3.9993z^{-1} + 5.9978z^{-2} - 3.9978z^{-3} + 0.9993z^{-4}}$
code 140	
code 180	$10^{-6} \cdot \frac{0.0660 - 0.1319z^{-2} + 0.0660z^{-4}}{1 - 3.9992z^{-1} + 5.9977z^{-2} - 3.9977z^{-3} + 0.9993z^{-4}}$
code 220	$10^{-6} \cdot \frac{0.0668 - 0.1336z^{-2} + 0.0668z^{-4}}{1 - 3.9992z^{-1} + 5.9976z^{-2} - 3.9977z^{-3} + 0.9993z^{-4}}$
code 270	

Figure 5.5: Specification of code filters, with f_c code frequency and f_s sampling frequency

Passband (normalized)	$[72, 78]/(f_s/2)$
Stopband (normalized)	$[68, 82]/(f_s/2)$
Noise attenuation in crossover	10 dB
Ripple in passband	1 dB
Transfer function	$10^{-4} \cdot \frac{0.2426 - 0.7278z^{-2} + 0.7278z^{-4} - 0.2426z^{-6}}{1 - 5.2418z^{-1} + 12.0425z^{-2} - 15.4123z^{-3} + 11.5777z^{-4} - 4.8449z^{-5} + 0.8886z^{-6}}$

Figure 5.6: Specification of Butterworth filter, with f_s sampling frequency

code and distubances for example due to the passing of a train or a switch. It also serves to exclude a signal which is low-power code from an adjacent section.

5.3.2 Check 75 Hz

The first check we have to perform concerns the presence of the correct basic frequency. We know the basic frequency is required to be 75 Hz +/- 3 Hz. To check if this is the case, we design a 6^{rd} order Butterworth filter with a passband from 72 to 78 Hz, specifications can be found in Figure 5.6. After this, we do the same as we did with the code-filtering: we calculate the RMS value of the signal and compare it to a threshold value.

5.3.3 Check duty cycle

The check if the duty cycle is correct is the most tricky. A lot of the measurements do not seem to obey the requirement about the forbidden area: the down period should be below 3 A and the up period should be above 6.5 A. We therefore need some means to separate the up and down period more sophisticated than comparing the value of the signal to for instance 4 A. First we smoothe the absolute value of the data. This removes most sudden peaks. We choose a moving average scheme for this: every sample is assigned the average of the 80 samples around it. This could also be done by a lowpass filter. After this we subtract the mean from the data series. Now we determine for every sample if it is more or less than zero using an if/else statement and construct a new signal which is one when the originar signal is high and zero when the original signal is low. We are not quite done yet, though. We cannot smooth the original 75 Hz vibrations entirely without damaging the up-down behavior, so we left some of the vibrations for what they are. This can result in unwanted phenomena when we are testing whether the signal is above or below zero. For example: the signal is greater than zero for some time, then one sample is smaller, one greater and the signal becomes smaller than zero for some time. So to smooth this out, we use another moving average scheme, this time of 10 wide. Now we can determine the length of the up and the down period, t_1 and t_2 respectively, using a while loop. This whole process is illustrated in Figure 5.7. Lastly we check whether the duty cycle $\frac{t_1}{t_1+t_2}$ is between 80 and 20 % and if the length of the cycle $t_1 + t_2$ is approximately equal to the period of the current code.

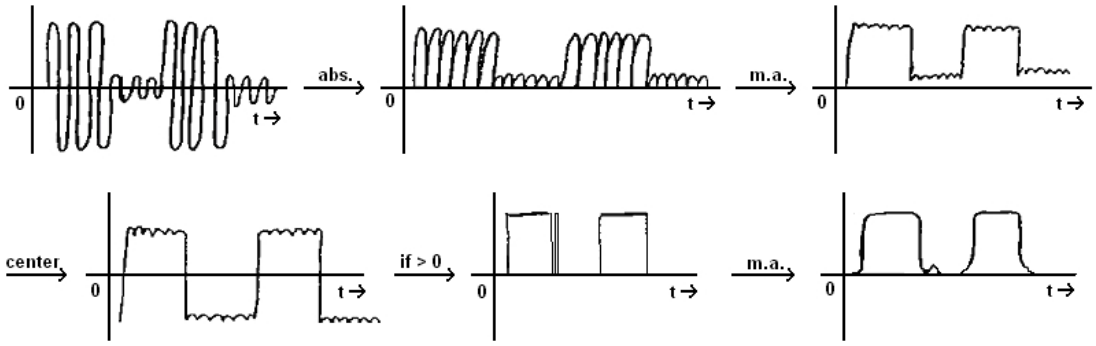


Figure 5.7: Extract the square wave from the data signal

5.3.4 Check current level

For the current level we implement something fairly easy. Since we cannot not use filtering, for it reduces the power of the signal, we just take the maximum value x per second of the sum of the absolute values of the signals per axle. We then compare these with reference values:

$$\begin{cases} 5.5 > x & \text{Current too low;} \\ 5.5 < x < 20 & \text{Current level ok;} \\ x > 20 & \text{Extremum, no conclusion.} \end{cases}$$

The last condition is added to detect extrema due to section transitions and the like.

5.3.5 Example

We now apply the code filtering to the same example we used for demonstrating the Fourier analysis approach. The results are in Figure 5.8. We can see code 96 has some power, allthough this was not the code present in the tracks. Notice that the filter is not bothered by the discontinuity at the 3^{rd} second. It is a slow filter though; its delay is over 2 seconds.

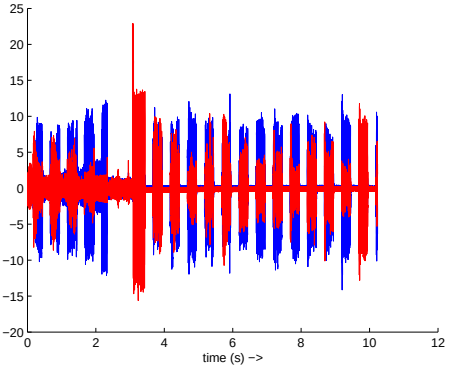
We apply the code-filtering method to the data with a change in code (Figure 3.10). The plot of the energy content of the filtered signals is in Figure 5.9. Notice the transition can be detected after the 2 seconds delay we mentioned before. So this filtering method solves the problem of changing codes.

5.3.6 Disadvantages

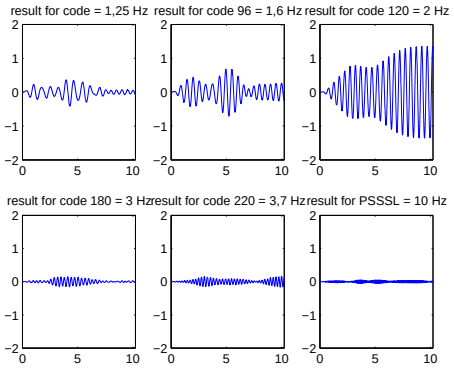
Something that immediately becomes clear is that this methods has a long start-up and return-to-rest delay. This is both a property of the filter and a result of the kind of information we are trying to obtain. Since one period of the slowest code, code 75, is 0.85 seconds long, we will need at least that time to detect if it was present and preferably twice as long.

5.3.7 Performance / Reliability

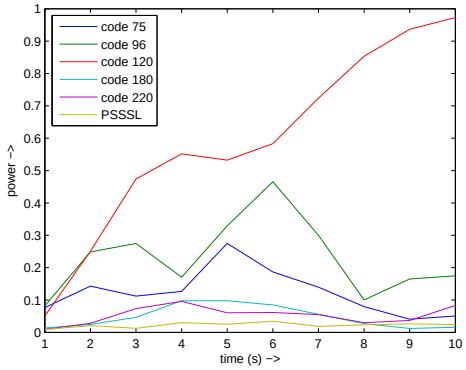
Our program does not give a continuous output, but gives a conclusion every second. In 3.21 % of the seconds the program concluded 'no code present' while we decided there was, and in 0.19% of the seconds it concluded there was a code present, while we decided there was not. There is a severe penalty on deciding 'a code is present' while in reality there is not. This is equivalent with telling the driver of the train: 'You are safe' when he is not. The three seconds contributing to the 0.19% error all occured while passing a section transition or a switch in an area without ATB code. Such a passing results in a high current for a short period of time, possibly equal to half the period of one of the ATB codes. So if we would introduce an on-delay of one period of the slowest ATB code (that is 0.8 s) we can remove these cases. On-delay means



(a) raw data, red is first axle, blue is second axle



(b) Code-filtered data



(c) Energy content of filtered data

Figure 5.8: Example of code filtering

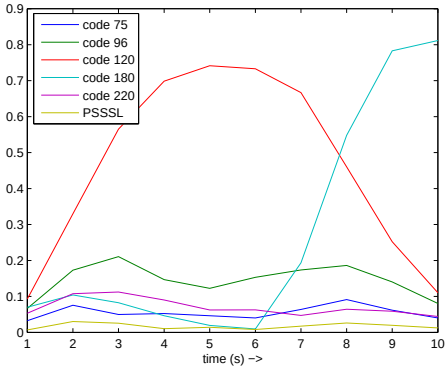


Figure 5.9: Energy content for a changing code

a signal is only set to 'high' if the signal becomes high and stays high for more than the delay-time.

Chapter 6

Final model

In this chapter we will examine the final model in detail. First we will describe the overall idea and how it works, after that we will discuss the different parts of the model in detail. In these subsections we will also treat the chosen model constants. We will end the chapter with a discussion of the performance of the system and its reliability.

6.1 Overview of final model

6.1.1 Soft- and hardware

The m-file we described in the previous chapter is not suited for running analysis, only for analysis in batches. We therefore create a new model in the simulation environment of Matlab, called Simulink. This will enable us to perform analysis while measuring, allthough it will not be real-time in the strict sense of the word. The new model will be based on the principles used in the m-file, where possible improved by new insights. We can also use ideas from a model for ATB detection created in the department of Stukton Systems in Hendrik-Ido-Ambacht (HIA). This model is based on simulated data and does not work for our messy, real world signals. However, we will be able to recycle parts of it. This department also developed a printed circuit board (PCB) with DAQ functionality and a digital signal processor (DSP). This DSP can be programmed with C code automatically generated from a Simulink model. The PCB has 8 input ports and a glassfiber output port which can be configured to be used for data logging. We will be using this PCB to realize the model on a train.

6.1.2 Time requirements

We want a model that can do analysis while time is running. This will be nessecary in order to inform the driver of the train whether or not he is detected. Preferably, the driver gets this information real-time, so he will know immediately if he is not detected and thus not safe. Real-time is a bit much to ask for however. It is sufficient if we know for sure the analysis is finished before the next sample is taken. In this margin we allow the analysis time to vary. We will have to accept some delay in the output of the model. This is due to the fact that we are using filters for low frequencies. We use a Butterworth filter, as we did before. If a sinusoid with a certain effective value is put through a filter with corresponding frequency, the resulting sinusoid goes from zero to original power in a certain amount of time, depending on the filter order and poles (see also Section 4.4). We decided that we will accept a delay of two seconds for the filter to reach its power and one second for the filter returning to no power. This means a delay of two seconds for a 'safe' notification and a delay of 1 second for a 'not safe' notification. Thus the model introduces one risky second.

6.1.3 Overview of model

We created two mostly identical models, one for sampling frequency $f_s = 3000$ Hz and one for sampling frequency $f_s = 1000$ Hz. The models differ only in the implementation of the filters, since these depend on

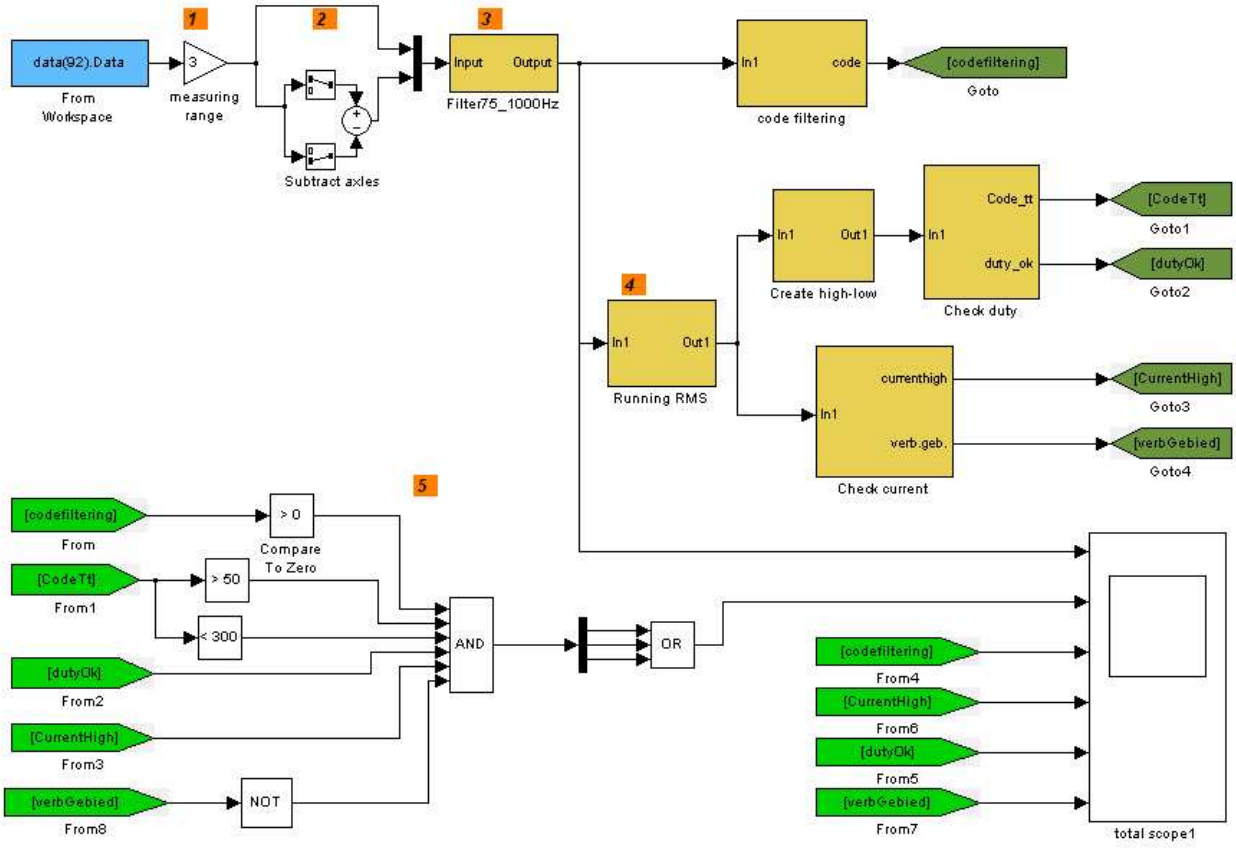


Figure 6.1: The final Simulink model

the sampling frequency. We will therefore discuss these models as if they are one.

The model is roughly divided into four sections: a general part (containing some preliminary operations, some logic and a plotting device), the actual code filtering, the check if the duty cycle meets the requirements and the check if the current is high enough. Notice there is no explicit check if 75 Hz is the basic frequency of the signal. This check is implicitly done while checking the current level, see Section 6.2.4. The model runs the analysis for three configurations: the signal of just axle one, of just axle two and the signal of axle one minus axle two. The latter case is in fact the total current going from rail one to rail two, since the measuring coils have opposite phase. For these three configurations the model applies code-filtering, a duty-cycle check and a current check. The order of these is left to calculate for Simulink, which contains optimalisation routines for determining the best order, reusing as many memory and calculations as possible. Currently all of the model is calculated every sampling instant. If more speed is needed, we could adapt the model to break as soon as it becomes clear the result will be negative. For instance is the current check fails for all three configurations. After the result of these submodels is known, the model does a few final comparisions. Then if all checks are ok and the code is in the correct range for one or more of the configurations, a positive result (one) is passed. If for all of the configurations one or more checks are false or the code is out of range, a zero is passed. The model is shown in Figure 6.1.

6.2 Details

In this section we will describe the working of the various components used in our Simulink model. We will start with the components visible in the top layer of the model, as can be seen in Figure 6.1. Then we will

describe 4 submodels visible in the top layer: codefiltering (Figure 6.6), construct high-low (Figure 6.13), check dutycycle(Figure 6.15) and check current (Figure 6.18). The orange numbers in the figures coorespond to the numbered list of components in each subsection. Also we will discuss the influence a component has on the signal it processes, such as loss in power or increase of noise. We can later use this information to determine the reliability of the model.

6.2.1 General model parts

Here we will describe the workings and components not directly related to codefiltering, duty cycle calculations or current level calculations. The numbers correspond to the orange numbers in Figure 6.1.

Components

1. Gain A gain amplifies the signal. The purpose of this block is to compensate for the resistors used in the measuring arrangement to convert current into voltage. This block amplifies all of the signal, including its noise. If there was noise added after passing resistors in the measuring arrangement, this noise is magnified by the amplifier. In practice, this particular source of noise is negligibly small. So this block has no notable influence on the signal.

2. Subtract axles This set of blocks creates a column vector containing the current sample of axle 1, axle 2 and axle 1 - axle 2. The top two positions are signals which are passed unchanged. The third position however, is influenced by the presence of phase shift between the two signals we subtract. In theory, the signals are in anti-phase, or one signal switches between zero phase shift and 180 degrees phase shift, see Section 3.2.4. In the former case we find the total current from rail 1 to rail 2 by subtracting the signals of both axles, which is what we are looking for. In the latter case, the signal after subtraction has less power, but it does exhibit all properties of ATB code, except maybe the current level. Moreover, all code is in one of the axles and we will detect this in the signal containing only that axle. In practice there is a third possibility. The two signal can have the same phase. This is a rare phenomenon, but if it occurs, subtracting gives a signal with no power. We will conclude there is no code present, warning the driver he is not safe, while he is safe. In this case, it is possible one of the two separate signals containes strong enough code.

3. 75 Hz Butterworth filter The schematic model of the filter is shown in Figure 6.2. Filtering can be seen as multiplying the Fourier Transform of a signal with a certain filter function. If this function is near zero for a certain stopband, that part of the signal (and the power it contains) is set to zero. Therefore the resulting signal can have less power than the original signal. However, we can be sure the remaining signal is all due to frequencies around 75 Hz, hence all due to ATB code or train detection currents. There can be another cause for loss of power after filtering. Whenever we want a certain passband of frequencies to pass the filter unchanged, we need to make the filter function equal to one for that interval. But this cannot always be realized for a continuous function with the desired order and steepness. In Figure 6.3 we plotted the Bode plots of the filters we used for the models with sample frequencies $f_s = 1000$ Hz and 3000 Hz. Also, in Table 6.4, we show the passband, stopband, poles and zeros of the filters. From the plots we can see the 75 Hz filters decrease the power of the signal by at most 1.8 %. We may have to consider this in the current level check.

Now we investigate the speed of the filter after the signal has stopped. So suppose we have a proper 75 Hz signal, sampled at 3000 Hz and with an amplitude of 20 A. We want the result of the filtering to drop to a peak value less than 0.5 A whithin one second after the original signal has stopped. This is a reduction with a factor 0.025 in 3000 samples. We already know the energy of the filtered signal after the original signal stops decreases according to λ^n, with λ the poles of the filter and n the samples. The sign of λ is not relevant, as long as its absolute value is smaller than one. Now we can calculate a boundary for the absolute value of the filter poles, needed to achieve the desired speed. We will use the following result from calculus:

$$\lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n = e^x. \tag{6.1}$$

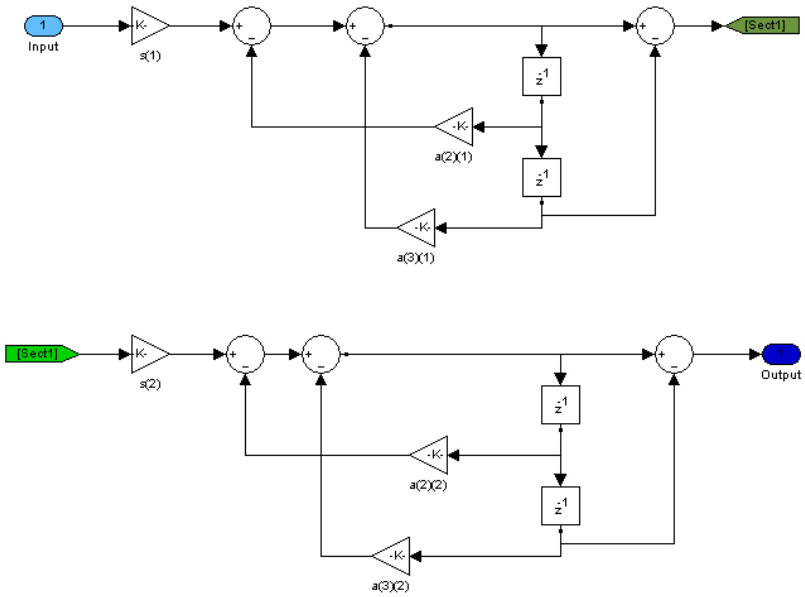
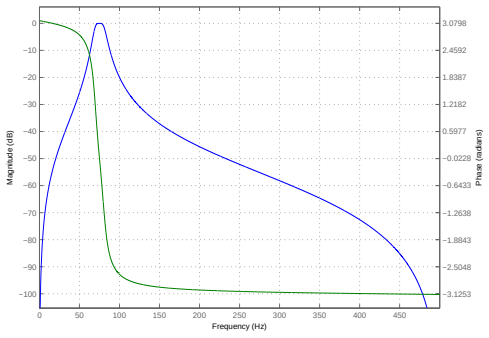
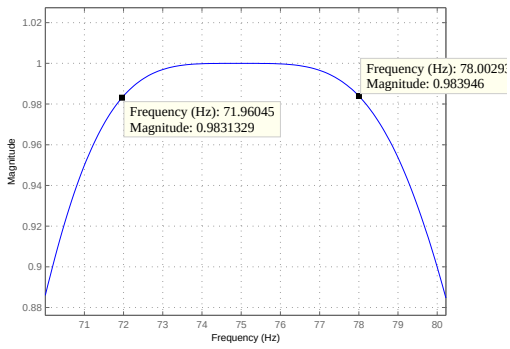


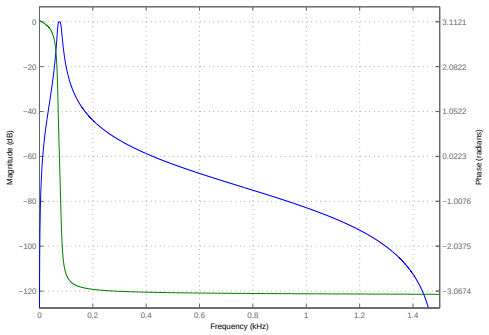
Figure 6.2: The Simulink model of a 75 Hz filter



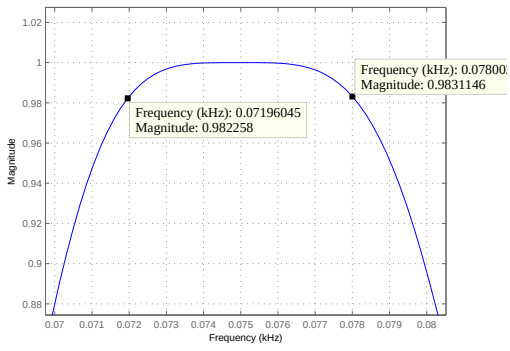
(a) Bode plot of 75 Hz filter for $f_s = 1000$



(b) Detail of Bode plot of 75 Hz filter for $f_s = 1000$



(c) Bode plot of 75 Hz filter for $f_s = 3000$



(d) Detail of Bode plot of 75 Hz filter for $f_s = 3000$

Figure 6.3: Bode plots of the magnitude of the 75 Hz filters

Sample frequency	3000 Hz	1000 Hz
Passband	$[72, 78]$	$[72, 78]$
Stopband	$[50, 100]$	$[50, 100]$
Order	4 in 2 2^{nd} order sections	4 in 2 2^{nd} order sections
Fall-of in stopband	20 dB	20 dB
Ripple in passband	1 dB	1 dB
Poles	$\{0.9752778 \pm 0.1646684i,$ $0.97978 \pm 0.1447217i\}$	$\{0.8478498 \pm 0.4656587i,$ $0.8787433 \pm 0.4132086i\}$
Zeros	-1, multiplicity 2	-1, multiplicity 2

Figure 6.4: Specification of 75 Hz Butterworth filter

We know λ is just smaller than one, so for large N we can write it as

$$\lambda := \left(1 - \frac{x}{N}\right). \tag{6.2}$$

Now if we need $\lambda^N < 0.025$ we can first solve for x:

$$\begin{aligned} \lambda^N &= \left(1 - \frac{x}{N}\right)^N \approx e^{-x} = 0.025 \\ x &= -\ln(0.025) = 3.6889. \end{aligned} \tag{6.3}$$

For every N we can calculate λ:

$$\lambda = \left(1 - \frac{3.6889}{N}\right). \tag{6.4}$$

For our two filters at sampling frequencies 3000 and 1000 Hz respectively we find

$$\begin{aligned} |\lambda| &< \left(1 - \frac{3.6889}{3000}\right) = 0.9988 \text{ and} \\ |\lambda| &< \left(1 - \frac{3.6889}{1000}\right) = 0.9963 \end{aligned} \tag{6.5}$$

So we have the desired speed iff the absolute value of the poles is smaller than 0.9988 or 0.9963, depending on the sampling frequency. Now if we consider the filters we constructed, we can calculate the absolute value of each pole and see if it fit the requirements.

$$\begin{aligned} f_s = \text{ 3000 Hz: } |0.9752778 \pm 0.1646684i| &= 0.9891 \\ |0.97978 \pm 0.1447217i| &= 0.9904 \\ f_s = \text{ 1000 Hz: } |0.8478498 \pm 0.4656587i| &= 0.9673 \\ |0.8787433 \pm 0.4132086i| &= 0.9710 \end{aligned} \tag{6.6}$$

So the designed 75 Hz filters are fast enough.

4. Running RMS RMS is short for Root Mean Square, a widely used estimate for the power of a signal x which is N samples long:

$$RMS(x) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2} \tag{6.7}$$

The implementation is shown in Figure 6.5. In case of an alternating current, it is an estimate of the effective value of the signal. In case the signal is a sine, the effective value can also be calculated by multiplying the

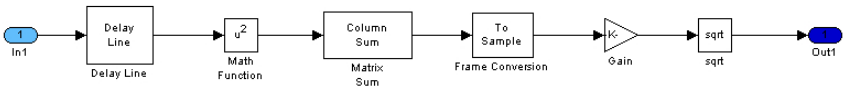


Figure 6.5: Simulink model for running RMS

peak value of the current by $1/\sqrt{2}$. By running we mean that we always calculate the RMS over the last N samples. In this case we choose $N = 2/75 \cdot f_s$, so we calculate the RMS over 2 periods of the 75 Hz signal. This results in an average effective value per 2 periods of 75 Hz, for every sampling instant. This way, we filter out (allmost) all vibrations due to the 75 Hz signal. The effect is stronger than if we would use one period of 75 Hz and it still only introduces a delay of 0.0267 s. We are left with the low frequency blokpulse indicating code, with some noise around 75 Hz and some rounding. We cannot remove the 75 Hz noise without slowing down the system considerably or damaging the up-down behavior of the blokpulse, so we will cope with this noise later on in the analysis. How large is the rounding effect of the running RMS? As said before, we use a window of $2/75 = 0.0267$ seconds wide. This is very small compared to the width of the blokpulse, which is at least 0.22 seconds long and more often 0.5 seconds long. Further, to construct a true blokpulse from this signal for duty cycle calculations we will be looking at whether the signal is above or below the equilibrium, which is not influenced by the rounding. We can neglect the influence of rounding.

5. Logic operations The goal of the logic operators it to determine if the signal meets all requirements of ATB code. These operations have a boolean as output: either a zero for 'false' or a one for 'true'. So the original input is not recognizable in this output. Now lets take a look at the operations themselves. First, the code must be nonnegative. A high current which is not pulsating is no guarantee for detection, only code 75 or higher is. Second, the length of a period calculated through counting all samples between an upward and downward flank (see Section 6.2.3) should resemble code (between 50 and 300 pulses per minute). This requirement is not very stringent, because this method is not very accurate. It is, however, suited to eliminate extreme results. Thirdly, ATB code should meet *all* posed requirements per configuration: the duty cycle should be in a certain region and the current should be high enough. We do not require the down-period of the signal to be low enough, since a large part of the measurements with code violate this rule. Lasty, it is enough if in one of the configurations axle 1, axle 2 or axle 1 - axle 2 all requirements are true. Now we have ensured the model gives a positive result if and only if there is at least one configuration for which all requirements are met.

6.2.2 Code filtering

The submodel Codefiltering (Figure 6.6) determines what code was dominant in the original data. Again this is done for three configurations. Since blocks are used which cannot process all three configurations at the same time, the signal is almost immediately splitted in three (Figure 6.7). The idea is that the absolute value of the signal is fed through seven very narrow and disjunct bandpass filters around the seven code frequencies. For every resulting signal we calculate an estimate for the power. Since the passbands of the code filters do not overlap, the power of each signal is an indication for the presence of the code corresponding to the passband of the filter. So at every sampling instant we select the signal with the largest power and give the according code in pulses per minute as output. We will now discuss why we filter the absolute value of the signal. Suppose we want to filter a modulation peak that the code causes around the basic frequency. We would then filter for the first and largest peak, located at 'basic frequency $\pm$ code frequency' Hz. The filters we would use would be 0.05 Hz wide, since this is the tolerance in the code frequencies. We would now have to determine the basic frequency very precisely, say a resolution of 0.01 Hz, to enable filtering at the precise location of the modulation peak. In Section 4.3 we found the relation between the resolution in frequency R_f and (time) length of the measurement T_f can be approximated by

$$R_f = \frac{1}{T_f}. \tag{6.8}$$

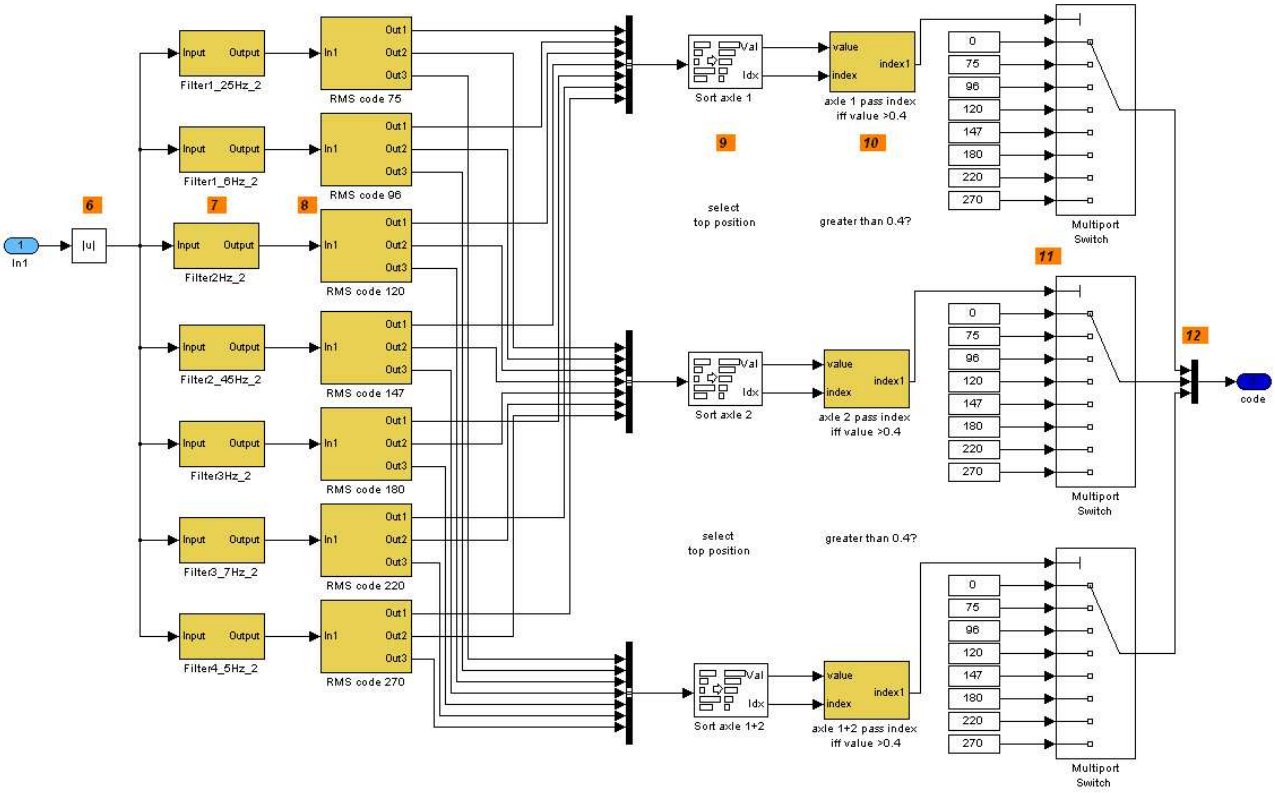


Figure 6.6: Simulink submodel codefiltering

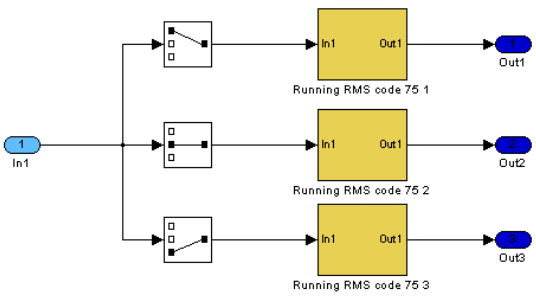


Figure 6.7: Simulink submodel RMS code 75: splitting the configurations

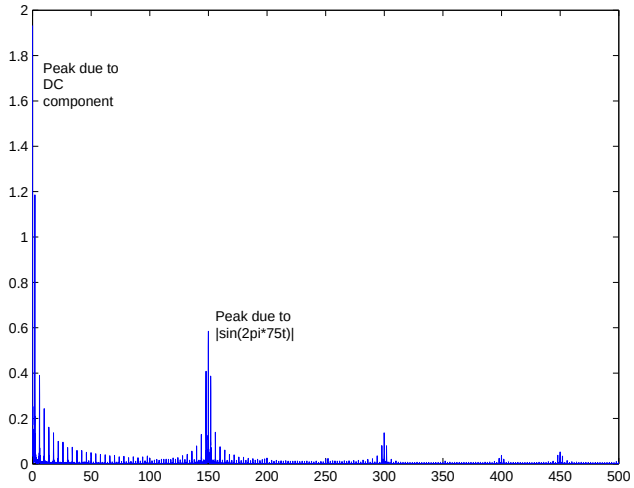


Figure 6.8: The Fourier Transform of the absolute value of a test signal

In this case we would need 100 seconds of measurements to reach enough certainty on the basic frequency. In this amount of time, the basic frequency probably changes 2 or more times. So this method is way too slow. Besides this, we would have to redesign our filters every time the basic frequency changes. Any method with fixed filters is more computationally effective. To eliminate the need for knowledge of the exact basic frequency, we take the absolute value of the signal. Now we have a signal with a large DC component: the mean is well above zero. In the Fourier Transform, this causes a large peak at zero (Figure 6.8). This peak acts as a second basic frequency, including modulation around it with the code frequency. For these modulation peaks we do know the exact basic frequency, namely zero. We can therefore design filters before analysis and be sure they always filter the correct frequencies.

Components

The numbers in the numbered list correspond to the orange numbers in Figure 6.6.

6. Absolute value The absolute value is taken to facilitate filtering, as explained above. It does not add noise to the signal, but it does double frequencies of all sines present. This is not a problem, since we are not inrerested in them, but just in the square wave.

7. Code filters These filters are 4^{th} order Butterworth filters, just like the 75 Hz filter we discussed before. They have the same components in the same order as the 75 Hz filters, allthough the gains have different values. All filters have a passband 0.1 Hz wide, this is precisely the tolerance in frequency for codes. The crossover region extends 0.15 Hz to each side of the code frequency. It turns out that all filters have two zeros with multiplicity two, namely -1 and 1. The poles of these filters can be found in Table 6.9.

From this set of poles (all have absolute value 0.9997) we can conclude the filters are not fast enough; $|\lambda|$ is not smaller than 0.9988 ($f_s = 3000$ Hz) or 0.9963 ($f_s = 1000$ Hz). In this model the problem is solved by creating a fast duty-cycle check, but we rather have fast filters. We could make them faster by reducing the steepness of the filters, widening the crossover- or passbands or allowing more ripple. However, all of these options were shown by testing to reduce the performance of the system. Acceptable is a sampling frequency of 1000 Hz, passband of 0.05 Hz to each side, 0.15 Hz crossover to each side and a fall-off of 10 dB in the crossover region. Unfortunately, reducing or increasing the sampling frequency has no notable effect on the speed. So for now we will have to rely on the duty cycle check.
As we described for the 75 Hz filter, not all power is contained in the frequency for which we are filtering. When we are filtering for code, this effect is much stronger, since there is no actual sine with the code

code frequency	$f_s = 3000$ Hz	$f_s = 1000$ Hz
1.25 Hz	$\{0.9998264 \pm 0.002775362i,$ $0.9998468 \pm 0.00245563i\}$	$\{0.9994563 \pm 0.008323255i,$ $0.9995225 \pm 0.007364528i\}$
1.6 Hz	$\{0.9998239 \pm 0.003510747i,$ $0.9998407 \pm 0.0003186978i\}$	$\{0.9994916 \pm 0.009557339i,$ $0.9994348 \pm 0.01052896i\}$
2 Hz	$\{0.9998204 \pm 0.0043504i,$ $0.9998346 \pm 0.0004023178i\}$	$\{0.9994046 \pm 0.01304659i,$ $0.9994553 \pm 0.01206531i\}$
2.45 Hz	$\{0.9998157 \pm 0.005294219i,$ $0.999828 \pm 0.004964447i\}$	$\{0.9993632 \pm 0.0158767i,$ $0.9994103 \pm 0.01488805i\}$
3 Hz	$\{0.9998088 \pm 0.00644713i,$ $0.9998197 \pm 0.00611533i\}$	$\{0.9994982 \pm 0.01915232i,$ $0.9995209 \pm 0.01852754i\}$
$3\frac{2}{3}$ Hz	$\{0.9997988 \pm 0.007844204i,$ $0.9998086 \pm 0.007510694i\}$	$\{0.9992118 \pm 0.02352279i,$ $0.9992566 \pm 0.02252289i\}$
4.5 Hz	$\{0.9990746 \pm 0.009590019i,$ $0.9997926 \pm 0.009254936i\}$	$\{0.9990746 \pm 0.02875669i,$ $0.991209 \pm 0.02775252i\}$

Figure 6.9: Poles of the code filters for $f_s = 3000$ and 1000 Hz

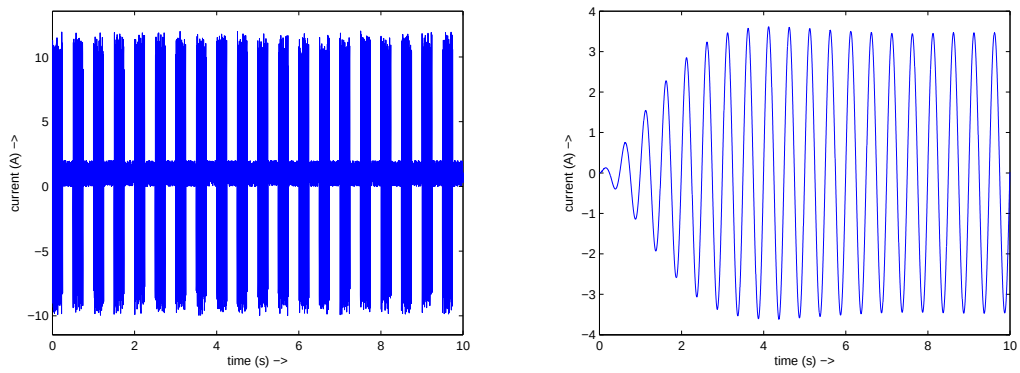


Figure 6.10: A 75 Hz signal modulated by a 2 Hz square wave and filtered with a 2 Hz bandpass filter

frequency present, but the code frequency is formed by high and low amplitudes of the 75 Hz signal. We created a test signal by multiplying a 75 Hz sine with a 2 Hz block puls and adding 1 A white noise. We used this testsignal to determine the maximal output when a typical measurement with code is filtered with a narrow 2 Hz filter. The result can be found in Figure 6.10.

8. RMS per code For each code a running RMS is calculated over the width of one period of the code. The submodel is shown in (Figure 6.11). This way we filter out all vibrations due to the specific code and we obtain a measure for the strenght of the filtered signals we can compare. We described this component in the previous section.

9. Sort axle This block receives as input a vector containing one sampling instant of all code filtered signals for one configuration, ordered from lowest to highest code frequency. The block sorts the input vector from high to low values and keeps track of which index ends up where in the sorted list. It then ouputs two vectors: the vector of ordered values and a vector containing the mapping that maps the original indices to the sorted indices. So if the highest value 2.5 was the 3^{rd} signal and the lowest value 0.1 was the first, the resulting value vector will have 2.5 at the top and 0.1 at the bottom position. The resulting index vector will have 3 at the top and 1 at the bottom position. Since we are interested in what signal was maximal, it would be no problem if we threw out all other information. However, to exclude code due to noise, we require the RMS of the filtered signal to be above a certain value, so we will need the value vector a little while longer.

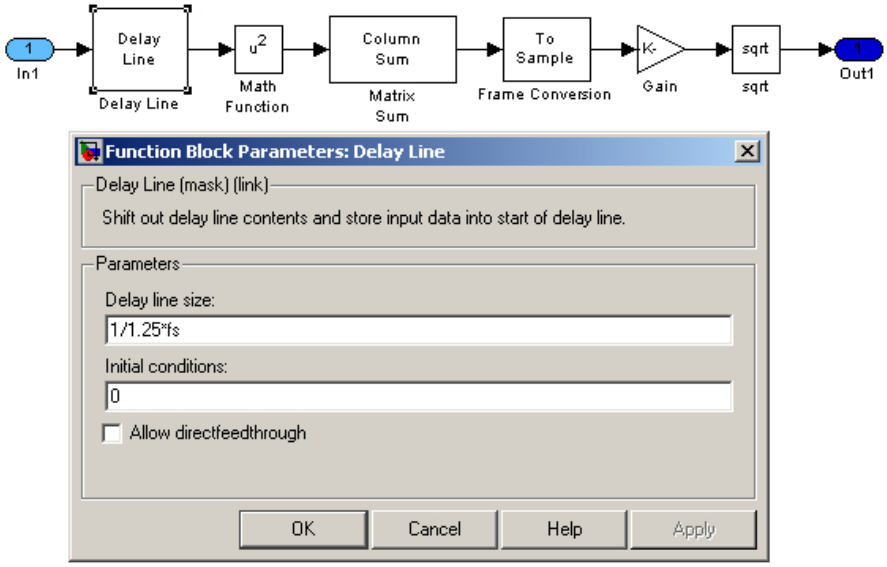


Figure 6.11: Simulink model of a running RMS for code 75

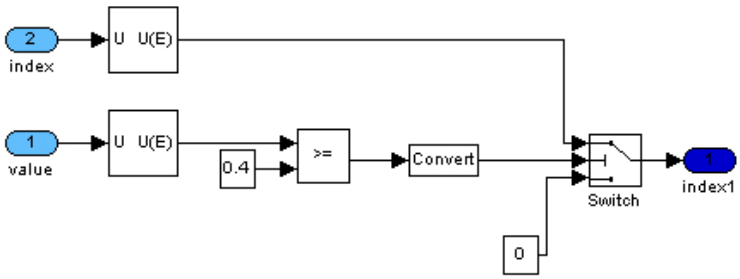


Figure 6.12: Simulink submodel pass index iff value > 0.4

10. Pass index iff value > 0.4 The title for this set of blocks speaks for itself: we pass the index of the dominant code if and only if its RMS value is above 0.4 (Figure 6.12). We choose this value to exclude small vibrations mistaken for code by one of the code filters. The value of 0.4 is empirically chosen. By visual inspection we noticed that the actual code was also the maximal code when it had reached a RMS value of 0.4. Moreover, if a code was present, this value was always reached within 1.5 seconds after the start of the filtering and therefore gives a conclusion fast enough. Now the switch block works as follows: if the controlling input is larger than one, the top input is passed, if not, the bottom input is passed. In this case, the control input equals one if the RMS value of the largest filtered signal is above 0.4, and zero if not. The upper input is the index belonging to the maximal value, the lower input is the constant value zero. So if the RMS value of the largest filtered signal is not above 0.4, the index is set to zero.

11. Multiport switch This block passes one of its inputs depending on the value of the control input. In our case, the control input is the index of the maximal code and the other inputs of the switch are zero and the amount of pulses per minute for each code. In this way, the output of the switch block is either zero or the number of pulses per minute for the dominant code.

12. Create a vector Lastly, the result for each configuration is put into a vector at the correct position: $[(\text{axle 1}) \ (\text{axle 2}) \ (\text{axle 1 - axle 2})]^T$.

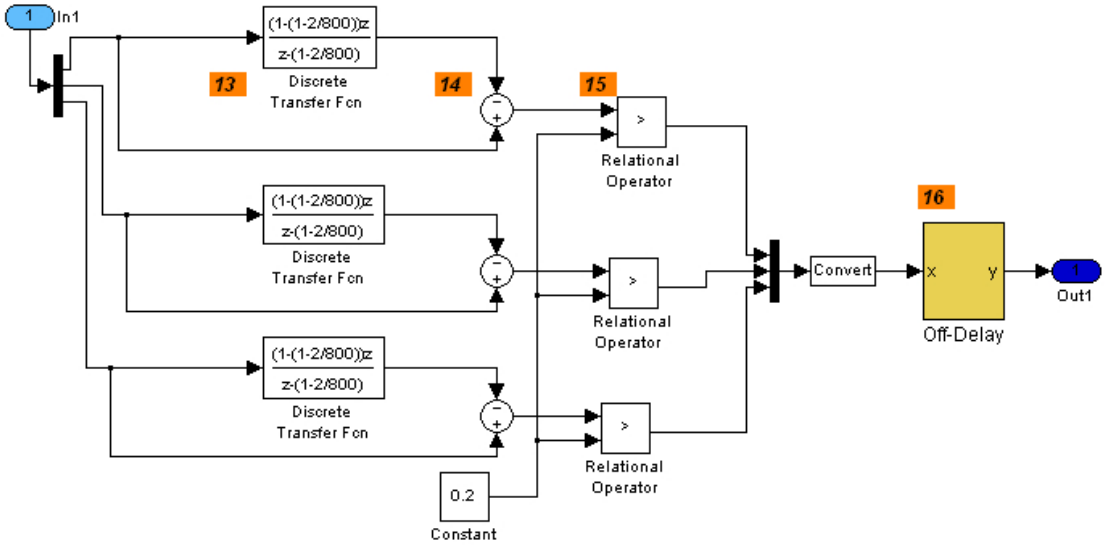


Figure 6.13: Simulink submodel 'create high-low'

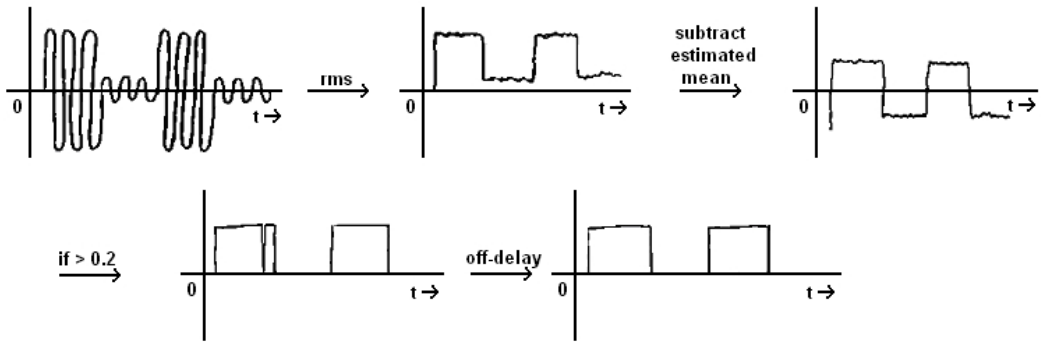


Figure 6.14: Extract the square wave from the data signal

6.2.3 Duty cycle check

The part of the model checking the duty cycle consists of two parts. In the first part (Figure 6.13), called 'create high-low', a blokpulse is constructed from the pre-processed data signal by subtracting its mean and comparing it to a constant. This is illustrated in Figure 6.14. Then in the second part (Figure 6.15), called 'Check duty', it is checked if the dutycycle is okay by using a finite state machine. This is modeled as a flow chart which keeps track of elapsed time by counting the number of samples until a certain event occurs. In this part of the model it is also checked if the signal is still pulsating as it should. It has two outputs: an estimate for the code present in the signal, which can be used to eliminate outliers, and it will output a boolean describing if the duty cycle is okay and the system is still pulsating.

Components - part one: create high-low

The numbers in the numbered list below correspond to the orange number in Figure 6.13.

13. Discrete Transfer Function The input to this block is a signal resembling a square wave, with some noise around 75 Hz. The discrete transfer function is a first order filter estimating the mean of the signal over the last 800 samples. We choose 800 samples, since this is the length in samples of one period of

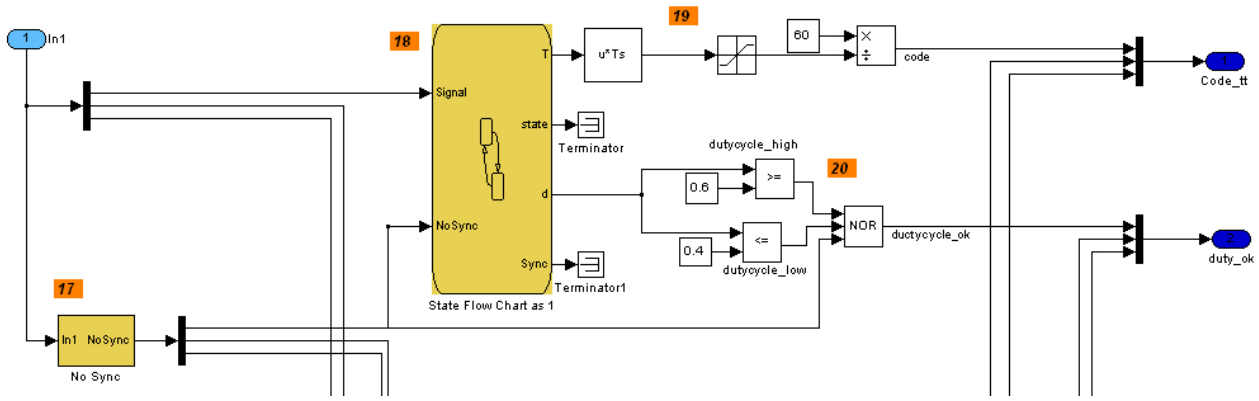


Figure 6.15: Simulink submodel 'Check duty'

the slowest code (75 pulses per minute or 1.25 Hz).

14. Addition or subtraction Here we subtract the estimated mean value from the signal to obtain a slightly noisy square wave centered around the time axis.

15. Relational operator We still need to construct a signal which only takes the values one and zero and follows the original signal closely. For this purpose we compare the square wave we constructed to the constant value 0.2. If it is larger, a 1 is passed, if it is smaller, a 0 is passed. Indeed we should be comparing the square wave to zero. The reason we take 0.2 is that if no code is present, the input to this block is a low energy signal with some noise which is not pulsating. Would we compare to zero, the result could look like '1101001010' for some set of ten subsequent samples. The result should be all zeros, since the original signal is in fact 'constantly low'. Assuming the noise at this stage of the analysis is alway smaller than 0.2 A, all zeros for a low power signal is enforced by comparing to 0.2. This approach decreases the length of the up-period by the time it takes for the signal to increase from 0 to 0.2 and the time in takes to decrease from 0.2 to 0. Since the transition from high (over 6.5 A) to low (under 3 A) in the original signal should take no time at all, we assume the decrease in time introduced by comparing to 0.2 is negligibly small.

16. Off-delay This block was developed by the department of Strukton Systems in Hendrik-Ido-Ambacht (HIA). We eliminate errors due to the 75 Hz noise present in the square wave that entered the submodel. These errors most often consist of one or two samples that are zero near the end (for example 50 samples before the end) of the up-period of the code. The off-delay repairs the short dip and keeps the signal high untill the end of the up-period. It introduces the specified amount of delay for the signal taking a downward slope. In other words: the signal is only then set to zero if it has been zero for more than x seconds. We choose $x = 1/75 \cdot 0.5$ seconds. Now we do not disrupt the up-down behavior of the code, which has a period of at least $1/4.5 = 0.22$ seconds. It lengthens the up-period by $x = 1/75 \cdot 0.5 \cdot f_s$ samples, but we will correct for this in the next part of the duty cycle check.

Components - part two: Check duty

The numbers in the numbered list below correspond to the orange numbers in Figure 6.15

17. NoSync In this subsystem (Figure 6.16) it is checked if the signal is still pulsating. The basic idea of this check was developed in HIA. For this we use a so-called triggered subsystem. In this case, the model starts running and resets when the input signal has an upward slope: when it changes from zero to one. At this trigger it starts integrating the constant function 1 and outputs the result. This output thus is the time elapsed since the last upward slope. If this value is larger than 0.85 seconds (duration of slowest code period plus margin) the signal 'NoSync' is set to one. So if the signal stops pulsating, we will know about it in 0.85 seconds.

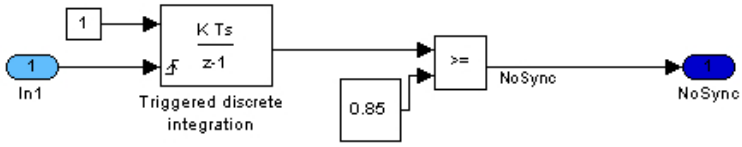


Figure 6.16: Simulink submodel 'NoSync'

18. State flow chart In Figure 6.17 the State Flow diagram used to model a finite state machine is shown. This machine checks for each sampling instant whether the square wave constructed before is one or zero and executes a specified action depending on the answer and the current state. We will use an example to explain the workings of this machine. Suppose we have a signal with $f_s = 1000$ Hz which is equal to one for 250 samples, then is zero for 1250 samples. This is the same as one complete pulse of code 120 (2 Hz) and then a whole second of no code at all. At the start t_1, t_2 and T are initalated and set to zero. We enter the 'idle entry State'. The first transition is made because Signal==1 (we are in the up-period of the duty cycle). We enter State 1. We make another transition to State 2, since Signal ==1. Now T is set to $t_1 + t_2$, d is set to $(t_1 - 7)/((t_1) + t_2)$, t_1 is set to 0 and the function Sync is called. 7 is subtracted from t_1 to correct for the delay introduced by the off-delay block. From State 2, there is again one option open to us at this time: back to State 2, t_1 is increased by one. This last step we repeat while Signal ==1, in our example this is for 250 samples and finally we have $t_1 = 250$, $t_2 = 0$, $T = 250$, $d = 1$. Now Signal == 0, so we make the transition to State 1. We make the transition back to State 1 250 times. Now we have processed 1 period of ATB code 120, $t_1 = 250$, $t_2 = 250$, $T = 500$, $d = 0.5$. The signal now continues to be zero. Now remember the 'NoSync' submodel. This sets the NoSync signal equal to one if an upward flank was more than 0.85 seconds ago, which triggers a transition. This corresponds to 850 samples. In our examle, we reach this point after sample 849. Before transition we have $t_1 = 250$, $t_2 = 600$, $T = 850$, $d = 0.29$, after transition to the idle State all is set to zero.
The state flow outputs the length of one period it counted, the state, the duty cycle and a call to the function Sync at the moment an upward slope is encountered. We do not use the output state or the function call to Sync. The output containing the length of one period was used by HIA to determine the code. However, this turns out to only work for generated, not measured, code signals. We therefore use this output only to detect extreme outliers, corresponding to less than 50 or more than 300 pulses per minute. The output $d = \frac{t_1}{t_1+t_2}$ is the duty cycle, for which we will check if it is within the margins.

19. Calculate pulses per minute This set of blocks calculates the number of pulses per minute from the length of one period in samples. For this we first multiply the amount of samples in one period with the sampling time to obtain the length of one period in seconds. Then we multiply the result by 60, to get the pulses per minute.

20. Check duty cycle Now we do the actual check: is the duty cycle within the correct boundaries? In theory, the duty cycle should be between 60 and 40 percent. However, we check if it is between 70 and 30 percent, to allow some deviation in the hardware along the rail. If we would comply to the official requirements, we would have to reject about 2 percent of all measured code, which is quite a lot. Lhe last block in the set of block could use some explanation. This is a logical operator, a NOR. It only puts out a one if input 1, nor input 2, not input 3 is true. In this case, a one is passed only if ['NoSync' equals zero, the duty cycle is not smaller than 0.3 and not larger than 0.7]. In other words, if the duty cycle is okay and the signal is still pulsating.

6.2.4 Current level check

This submodel was designed to conclude if the current level of the measured data is above 6.5 A, which is the requirement for ATB code, Figure 6.18. The input to this block is the three configurations, filtered for

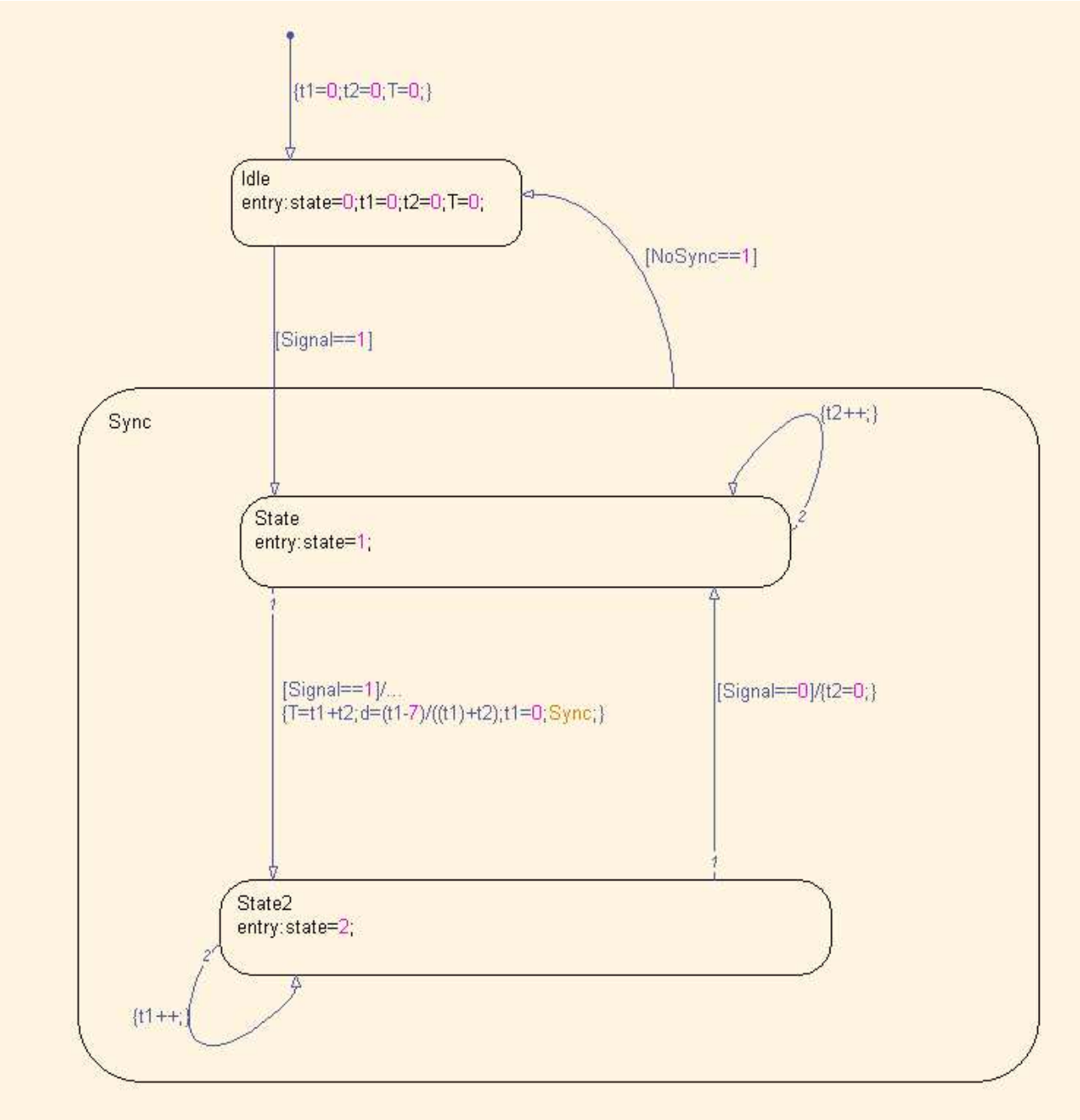


Figure 6.17: Simulink State flow chart

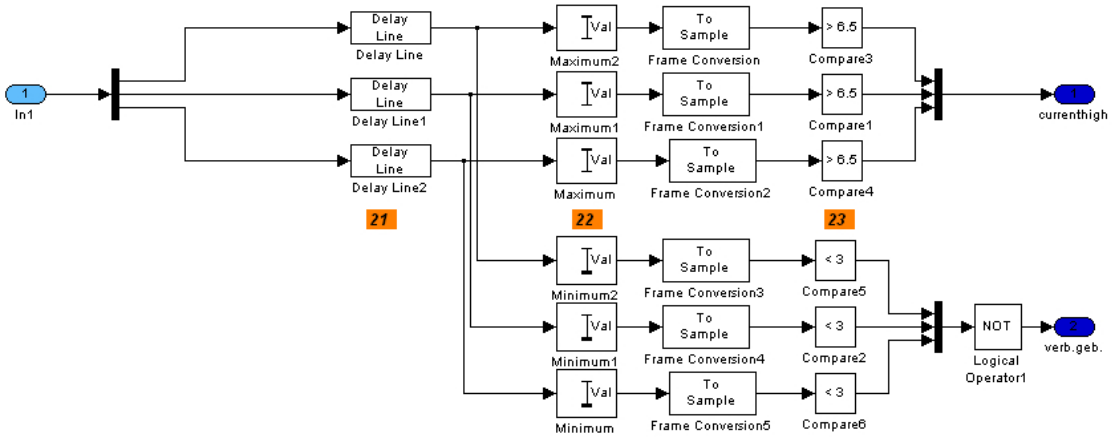


Figure 6.18: The Simulink submodel 'check current'

75 Hz and with a 2/75 seconds wide running RMS taken. This corresponds to the effective value of the 75 Hz signal. Now we take the maximal value in a window one period wide and check if it is larger than the specified minimum.

Components

The numbers in the numbered list below correspond to the orange number in Figure 6.18.

21. Delay line A delay line creates a vector of specified length, in this case $0.8 \cdot f_s$ samples, and fills it with data samples from the immediate past. At initialisation, the vector is filled with zeros.

22. Maximum and minimum These blocks calculate the maximum resp. minimum over the whole input vector, that is over the window $0.8 \cdot f_s$ samples wide, created by the delay line block.

23. Compare to constant According to the requirements for ATB, the current must have an effective value over 6.5 A in the up-period and under 3 A in the down-period. Remember from Section 6.2.1 we lost at most 1.8% of power. In the up-period this means we should check for $6.5 \cdot 0.018 = 6.4$ A. However, if we would encounter a data series with exactly 75 Hz as a basis and 6.4 A effective current, the 75 Hz filtering would not diminish the power of the signal. So if we would approve this series with a maximum of 6.4 A, we would approve a code which is not actually ATB code. So we check if the maximum is over 6.5 A, just to be safe. The same story goes for the minimum value; we check if it is below 3 A.

6.3 Reliability and Performance

6.3.1 Reliability

We tried to make this model as reliable as possible. A large sidenote is that we only had a limited set of data. Moreover, for this data we had no information on the state of traindetection and only little information on the particular stretch of tracks. We therefore made the assumption that the locomotive 'Ploeg 14' we did our testing on was detected in all cases. This is a relatively safe assumption. Although this locomotive has not yet been approved for driving on the tracks, it is ecpected that it will be soon. Next, we assumed our model had to label everything we judged 'no code' as unsafe, and everything we judged code it had to label as safe. We were quite tolerant on what was code, since there is no information available on to what degree the hardware along the tracks meets the requirements set by ProRail. This assumption can be disputed and it can only be verified by making more test runs and ensuring the status of train detection is known throughout these tests.

Model configuration	Error type I	Error type II	Total % correct
As proposed	14.54 s $\Leftrightarrow$ 0.68 %	1.85 s $\Leftrightarrow$ 0.09 %	99.23 %
Duty cycle between 30 and 70 %	7.23 s $\Leftrightarrow$ 0.34 %	4.2 s $\Leftrightarrow$ 0.20 %	99.46 %
Rejecting if effective value of down-period $>$ 3 A	54.05 s $\Leftrightarrow$ 2.54 %	0.85 s $\Leftrightarrow$ 0.04 %	97.42 %

Figure 6.19: Error percentages

All measured signals which are approved by our program meet the following requirements:

- The maximum of the effective value of the up-period of the current is above 6.5 A.
- The square wave in the signal has a duty cycle between 40 and 60 %.
- A component with one of the code frequencies (within their tolerance of 0.05 Hz) is present in the signal. It is strong enough to state that it is not due to noise.

We dropped the requirement that the effective value of the current in the down-period of the square wave should be below 3 A. This compromise made it possible to accept more signals as code. A test has been done to monitor the effect of this compromise. We also tested what would happen if we would widen the requirement on the duty cycle to allow signals with a duty cycle between 30 and 70 % instead of just between 40 and 60 %.

6.3.2 Performance

We tested the performance of the system. In Figure 6.19 a table can be found with the errors in percentage of the total amount of time in the measuring data. There is a distinction between two types of errors: the first type are errors resulting in a 'not safe' notification, a red light, while in reality the train was detected and thus safe. This is the type of error we try to keep small, but it is no big deal if it occurs every now and then. Mostly this is caused by some irregularity, such as a switch. The second type of errors occors when the model decides code is present, allthough in reality there is not. This type of error is much worse, since it makes the train driver feel safe, allthough he is not. In the worst case, a collision takes place. So we really want this error to approach zero. To some degree, this can be done by changing model parameters in such a way the percentage errors of type II decreases. This almost immediately causes the percentage errors of type I to increase. So we have a trade off between the two types of error. We believe this model with the duty cycle between 40 and 60 % provides the best balance between the two.

Chapter 7

Product

This project resulted in a PCB: a Printed Circuit Board. This board has a component programmed in C code automatically generated from the Simulink program described in Chapter 6. The board was specifically designed for application in the railway branch by the Strukton Systems department in Hendrik-Ido-Ambacht.

Chapter 8

Frequency Resolution

In this chapter we will look into the frequency resolution after Fourier transforming a signal of finite time length. We take the following approach. Consider the continuous time domain. If we apply the Fourier transform to a finite length signal, we will obtain a spectrum with some uncertainty or error, so we can only determine the frequency with a certain frequency resolution R_f. This phenomenon is called leakage and caused by finite time. There is a technique to reduce leakage, called windowing, which we will explain below. Several windows are known from literature [18], such as the Hann and the Hamming window. In this chapter we will, for a certain R_f, generate an optimal window.

8.1 Windowing

In Figure 4.6 in Section 4.2 we saw the effects of Fourier Transformation of a finite sine. In this section we will consider an approach which tries to minimalize the effects of finite time on the Fourier Transform of a signal. First we consider the situation in Figure 4.6. We can regard a sine g of finite length T and period $1/T$ as the result of a multiplication of the same sine of infinite length and a signal which is 1 on the interval where g is nonzero and 0 elsewhere (Figure 8.1). We call this a rectangular time window. In a formula this multiplication would be:

$$g(t) = \sin(2\pi t/T) \cdot \text{rect}_{[-\frac{T}{2},\frac{T}{2}]} \tag{8.1}$$

We want to explain Figure 4.6, therefore we compute the FT of the signal g by separately transforming the infinite sine and the time window. Now we know the Fourier Transform of a sine is a single peak located at its frequency, in this case 1 Hz: $\delta(\omega - 1/(2\pi))$. The Fourier Transform of the rectangular window equals:

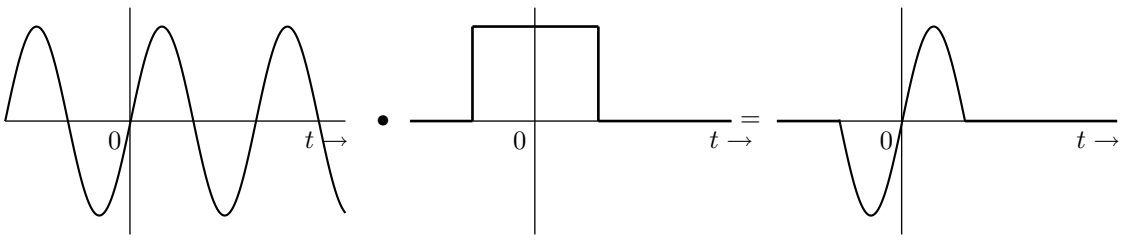


Figure 8.1: The multiplication of a sine and a rectangular window

$$\begin{aligned}
\mathbb{F}[\text{rect}_{[-\frac{T}{2},\frac{T}{2}]}] &= \int_{-\infty}^{\infty} \text{rect}_{[-\frac{T}{2},\frac{T}{2}]}(t)e^{-i\omega t}dt \\
&= \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-i\omega t}dt \\
&= -\frac{1}{i\omega}\left[e^{-i\omega t}\right]_{-T/2}^{T/2} \\
&= \frac{e^{iT\omega/2} - e^{-iT\omega/2}}{2i\omega/2} = T\frac{\sin(T\omega/2)}{T\omega/2} \\
&= T\ \text{sinc}(T\omega/2).
\end{aligned} \tag{8.2}$$

We convolve the sine and the window in the frequency domain:

$$T\text{sinc}(T\omega/2) * \delta(\omega - 1/(2\pi)) = T\ \text{sinc}(\frac{T\omega - 1/\pi}{2}) = T\text{sinc}(T\pi f - 1), \tag{8.3}$$

which is precisely a sinc function centered around the frequency of the finite sine, with height and width scaled by T. Indeed we see in Figure 4.6 the absolute values of sinc-functions. Now we see that the rectangular window used to create a finite-time signal is the cause of leakage in the frequency domain.

One will immediately see the sinc function (rectangular window) is not the most optimal to use as a window. It is wide, causing a large spread of power, and it has large sidelobes, causing leakage around other frequencies. Rather we have a window with a Fourier transform which is narrow and has very small sidelobes. This way we can minimalize the error due to the sidelobes, the leakage. A lot of mathematicians have tried to find such a window. In Figure 8.2 we plot the rectangular window and two very famous windows with small sidelobes: the Hann and the Hamming window.

8.2 Find optimal window

Consider an infinite time-length signal g. We multiply this signal by a real window w, still to be determined, which is zero except on $[-\frac{T}{2}, \frac{T}{2}]$ and scaled such that $\|w\|^2_{L_2[-\frac{T}{2},\frac{T}{2}]} = 1$. Denote its Fourier Transform by $W(\omega)$.

For a given a we will calculate

$$\min_W \int_{|\omega|>a} |W(\omega)|^2\, d\omega. \tag{8.4}$$

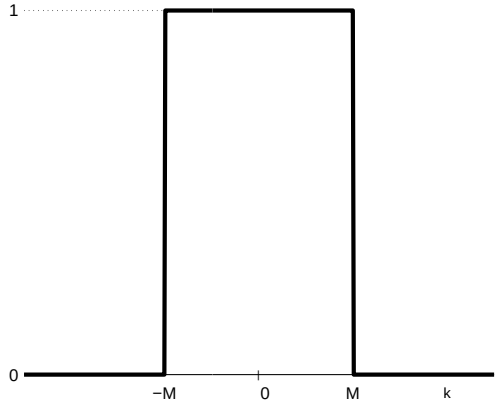
First we remark that Parseval's theorem immediately gives that

$$\|w\|^2_{L_2[-\frac{T}{2},\frac{T}{2}]} = \|W\|^2_{L_2} = 1. \tag{8.5}$$

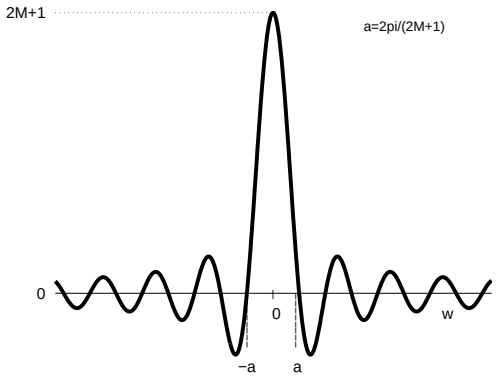
Now if we have a bandlimited signal, we can write it as a sum of sinc-functions in the time-domain, according to the Sampling Theorem (Section 4.2). We are interested in the parallels between a finite bandwidth signal and a finite time width signal. We could interpret the frequnecy resolution as some sort of sampling: the information in between discrete points h_k is deemed unreliable and we ignore it. This is similar to sampling. What would happen if we interchange t and ω in the Sampling Theorem?

Modified Sampling Theorem

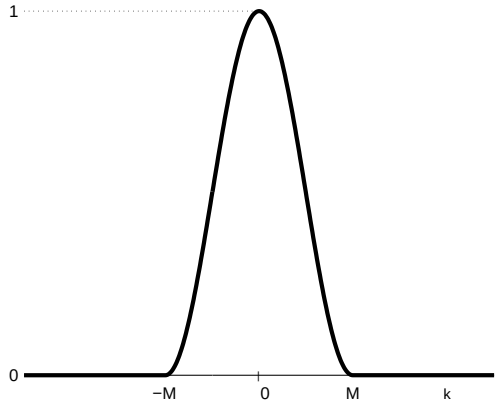
Let $g(\omega) \overset{\mathbb{F}}{\leftrightarrow} G(t)$ and suppose $g(\omega)$ is timelimited with length T. Let $g(k2a)$, $k \in \mathbb{Z}$ and a the frequency resolution, be the 'sampled' signal of window $g(\omega)$ with angular sampling frequency $\omega_s = 2\pi/(2a)$. If $\omega_s > 2T$



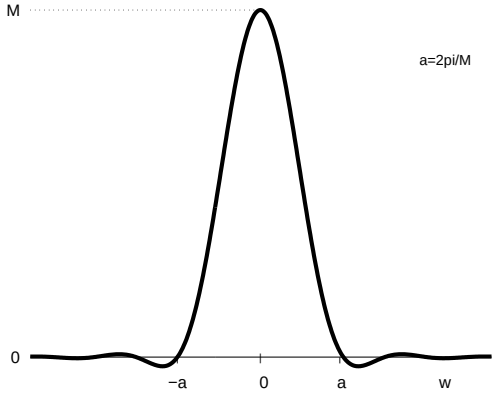
(a) Rectangular window in time domain



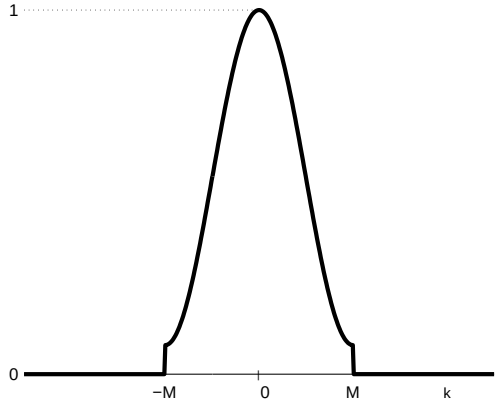
(b) Rectangular window in frequency domain



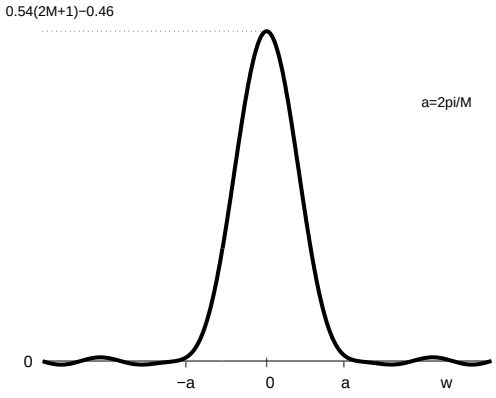
(c) Hann window in time domain



(d) Hann window in frequency domain



(e) Hamming window in time domain



(f) Hamming window in frequency domain

Figure 8.2: Three windows in time (left) and frequency

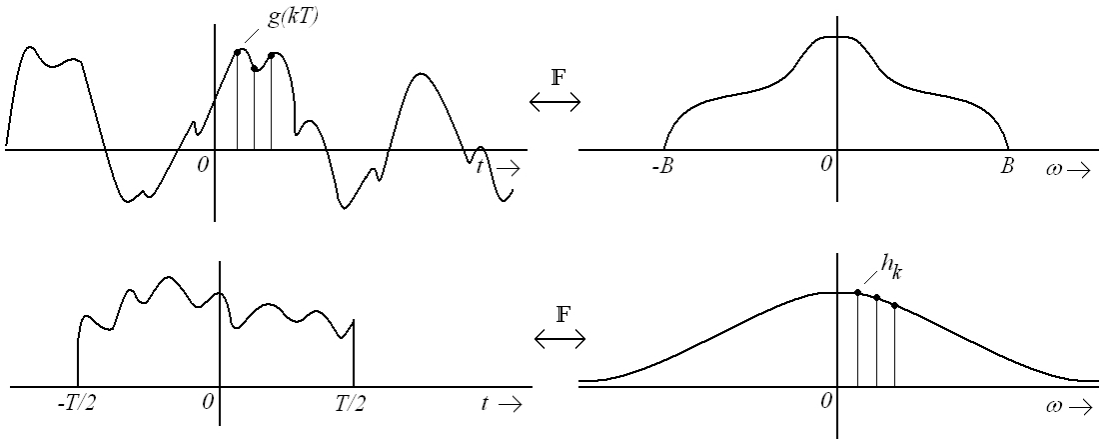


Figure 8.3: A parallel between a finite band and finite time signal

then $g(\omega)$ is uniquely determined by its samples $g(k2a)$ by:

$$g(\omega) = \sum_{k=-\infty}^{\infty} g(k2a) \frac{\sin(\omega_s(\omega - k2a)/2)}{\omega_s(\omega - k2a)/2} \tag{8.6}$$

This idea is illustrated in Figure 8.3 and we have the following parallels:

finite bandwidth B	$\longleftrightarrow$	finite timewidth T
sampled signal	$\longleftrightarrow$	frequency resolution
sampling time T	$\longleftrightarrow$	frequency 'steps' $2a$
represented by sinc functions in time	$\longleftrightarrow$	represented by sinc functions in frequency

We can now extract the rule of thumb we use for the relation between the time length T and the frequency resolution a. In the modified Sampling Theorem the requirement for reconstruction from 'samples' is $2\pi/(2a) > 2T$, or $1/a > T$ for the 'normal' frequency f. Here we have the rule of thumb the minimal frequency resolution is approximately equal to $1/T$, with T the time length.

The duality between the frequency and time domain and the FT and inverse FT inclines us to investigate whether a timelimited signal can be written in the form of a sum of sinc-functions in the frequency-domain:

$$W(\omega) = c_1 \sum h_n \ \mathrm{sinc}(c_2(c_3\omega - an)), \quad n = 0, \pm 1, \pm 2, \pm 3, \ldots \tag{8.7}$$

with c_i some constants.

Properties of the h_k

First consider the inverse Fourier Transform of $W(\omega)$, which should give the unknown real function $w(t)$. The inverse FT of the sinc function is a real rectangular, as we see from the duality of the FT and the inverse FT. Since the h_k's are constants and the Fourier Transform is linear (Appendix A), we need to choose the h_n real, in order to have $w(t)$ real.
Secondly, consider the terms h_1 and $\overline{h_{-1}}$. These two terms of the summation should be real, when added. There is no other way to force these two specific terms to be real, since the functions $\mathrm{sinc}(\omega - k)$, $k \geq 0$ form an orthonormal basis. We know

$$h_0 \mathrm{sinc}(\omega) \overset{\mathbb{F}}{\leftrightarrow} h_0 \mathrm{rect}_{[-\frac{T}{2}, \frac{T}{2}]},$$

so by the shift rule, for the h_1 and h_{-1} terms hold:

$$h_1\text{sinc}(\omega-1)+h_{-1}\text{sinc}(\omega+1) \overset{\mathbb{F}}{\leftrightarrow} h_1 e^{it}\text{rect}[-\frac{T}{2},\frac{T}{2}]+h_{-1}e^{-it}rect[-\frac{T}{2},\frac{T}{2}]. \tag{8.8}$$

Rewrite the two terms in the frequency domain.

$$\begin{aligned} h_1 e^{it} &= h_1(\cos(t)+i\sin(t)) \\ h_{-1}e^{-it} &= h_{-1}(\cos(t)-i\sin(t)) \end{aligned} \tag{8.9}$$

Since we need h_1+h_{-1} real, we know

$$h_1 = \overline{h_{-1}} \tag{8.10}$$

and we only need to determine the h_k for $k \geq 0$.

Derivation of optimal window

Consider the time-limited window w as a window of infinite length w_∞ multiplied by a rectangular window $\eta(t)$ which is zero outside $[-T/2, T/2]$. Now since we want to know if we can write w in terms of constants we can see as samples, we follow the idea behind the Sampling Theorem and assume W_∞ is sampled in frequency with sampling time $2a$, the desired frequency resolution. In the frequency domain, W_∞ can be written as a train of modulated pulsed:

$$W_\infty(\omega) = \sum_{n=-\infty}^{\infty} h_n \delta(\omega/(2\pi)-2an) \tag{8.11}$$

with h_n the sampling instants - or the constants we will have to choose. Now we Fourier transform the rectangular time-window $\eta(t)$ (see Equation 8.2):

$$\mathbb{F}[\eta(t)] \quad = \quad T\ \text{sinc}(T\omega/2). \tag{8.12}$$

Since multiplication in the time domain is the same as convolution in the frequency domain, we convolve W_∞ and $\mathbb{F}[\eta(t)]$ to obtain $W(\omega)$:

$$\begin{aligned} W(\omega) &= T\ \text{sinc}(T\omega/2) * \sum_{n=-\infty}^{\infty} h_n \delta(\omega/(2\pi)-2an) \\ &= \sum_{n=-\infty}^{\infty} h_n T\ \text{sinc}(T\omega/2) * \delta(\omega/(2\pi)-2an) \\ &= \sum_{n=-\infty}^{\infty} h_n T\ \text{sinc}(T\frac{\omega/(2\pi)-2an}{2}) \end{aligned} \tag{8.13}$$

and for $\omega = 2\pi f$ and $a \approx 1/T$ this is approximately equal to

$$\sum_{n=-\infty}^{\infty} h_n T\ \text{sinc}(f/(2a)-n) \tag{8.14}$$

Note that h_n are the constants completely determining the window, for a given a. We now switch to normal frequency f, since it is more intuitive. The window W can be seen as the inner product between the vecors h and s, where $s_n = T\text{sinc}(f/(2a)-n)$, for $n = 1, 2, 3, \ldots$. We will later discuss the truncation of this summation. These sinc functions are real, so they are equal to their complex conjugates. Like the sinc

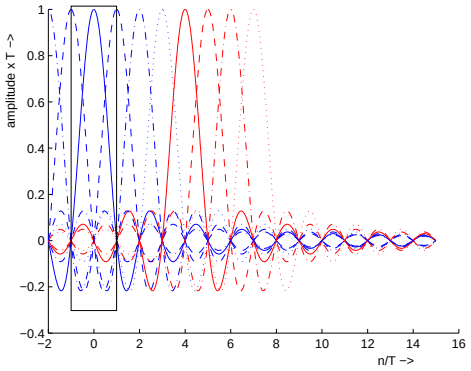


Figure 8.4: The functions $\text{sinc}(\omega - 2an)$

functions, the constants h_n are real. Now we can write (8.4) in terms of vectors:

$$
\begin{aligned}
(8.4) &= \min_W \int_{|f|>a} \overline{W}^T W df \qquad (8.15)\\
&= \min_W \int_{|f|>a} \overline{h}^T \overline{s}^T s h df \\
&= \min_W \int_{|f|>a} \begin{bmatrix} h_0 & h_1 & \dots \end{bmatrix} \underbrace{\begin{bmatrix} s_0 s_0 & s_0 s_1 & \dots \\ s_1 s_0 & \ddots & \\ \vdots & & \ddots \end{bmatrix}}_{S} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \end{bmatrix} df \\
&= \min_W \begin{bmatrix} h_0 & h_1 & \dots \end{bmatrix} \left(I - \int_{|f|>a} S df \right) \begin{bmatrix} h_0 \\ h_1 \\ \vdots \end{bmatrix}.
\end{aligned}
$$

Since I is a constant matrix, it has no role in the minimalisation over W, so we can pull it out of the expression and drop it. To get rid of the minus sign, we maximalize the expression multiplied by -1:

$$\max_W \begin{bmatrix} h_0 & h_1 & \dots \end{bmatrix} \underbrace{\int_{|f|<a} S df}_{X} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \end{bmatrix} = \overline{h} X h \qquad (8.16)$$

We approach the infinite matrix S by a finite one, and consequently, also truncate h_n. Consider the case $h_n = 1$. As n increases, the sinc functions shift to infinity with steps of n, see Figure 8.4. If we truncate the matrix S we discard integrals around zero of sinc functions with their peak far from zero. Their contribution is therefore small. However, it is not trivial the sum of all discarded tail-pieces is finite, since the sinc function approaces zero as $1/x$, whose integral over $[1, \infty)$ diverges. But since all sinc functions have the same period, we can regard all integrals as taken over a different interval of the same function. So instead of shifting the functions, we shift the interval of integration. This idea is illustrated in Figure 8.5, were each arrow indicates an interval. So we write

$$\sum \int_{|f|<a} T\text{sinc}(Tf/(2a) - n) df = 2T \int_{-\infty}^{\infty} \text{sinc}(Tf/(2a)) df. \qquad (8.17)$$

This integral is finite and equal to $2\pi T$. If we truncate the integral at the window corresponding to $n = k$, then the integral over the tail, or error, equals

$$2T \int_{|f|>ak} \text{sinc}(Tf/(2a)) df \qquad (8.18)$$

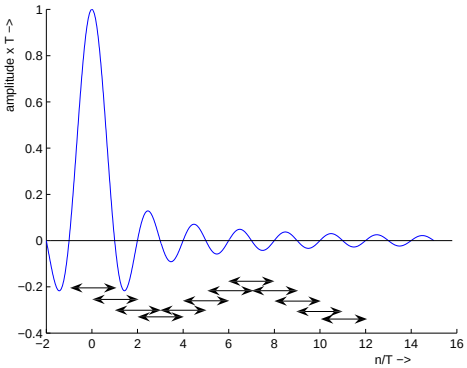


Figure 8.5: Shift the windows, not the functions

This error can be made as small as desired by choosing k larger, even if the h_n are larger than one. So we replace the infinite matrix S by the same matrix truncated at k. So our problem becomes:

$$\max_W \begin{bmatrix} h_0 & h_1 & \dots & h_k \end{bmatrix} \underbrace{\int_{|f|<a} Sdf}_{X} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_k \end{bmatrix} = \overline{h}Xh \tag{8.19}$$

with X a $k \times k$ matrix.

Decomposition

We can decompose the square matrix X. For this, we will use the singular value decomposition.

Theorem 8.2.1 (Singular value decomposition)**.** *If A is a real $n \times m$ matrix with $m > n$, the A can be rewritten using a singular value decomposition, of the form*

$$A = UDV^T, \tag{8.20}$$

where U and V are uniform matrices, so

$$\begin{aligned} U^TU &= I \\ V^TV &= I \end{aligned} \tag{8.21}$$

and D a diagonal matrix contaning the so called singular values. These are defined as the eigenvalues of A^TA, completely determined by A and always real. U and V consist of the eigenvectors of A^TA and AA^T, respectively. Therefore, they are not uniquely determined.

If matrix A is symmetric, A^TA equals AA^T. Then $D > 0$ and $U = V$. The singular values are often orderen from highest to lowest, the matrices U and V can be constructed to be real instead of each others complex conjugate. So if we apply the singular value decomposition to the matrix X we constructed before, we obtain a square matrix D with positive entries on the diagonal, ordered from high to low, and $U = V$:

$$X = UDU^T. \tag{8.22}$$

Now maximizing the product

$$(8.16) = hUDU^Th \tag{8.23}$$

can be done by selecting the largest singular value from D, which is D_{11}. Let u_1 be the first row vector of U. Since U is a unitary matrix

$$u_1 U = [100 \ldots 0] \tag{8.24}$$

and

$$U u_1^T = [100 \ldots 0]^T. \tag{8.25}$$

So to maximize (8.16) we choose $h = u_1$.

8.3 Calculate windows with Matlab

As explained in the previous section, the optimal window that minimizes leakage outside of a given frequency resolution is approximately equal to (in terms of 'normal' frequency f)

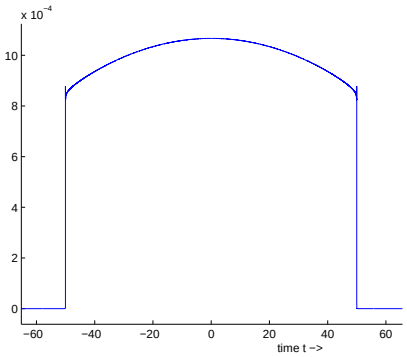
$$\sum_{n=1}^{\infty} h * \mathrm{sinc}(\pi f - an) \tag{8.26}$$

where we calculate h using singular value decomposition. We use Matlab to do the numerical calcutations of the integrals over the sinc functions and to carry out the singular value decompostion. For both the integrals and the svd there exist very good numerical algorithms. The program code is below.

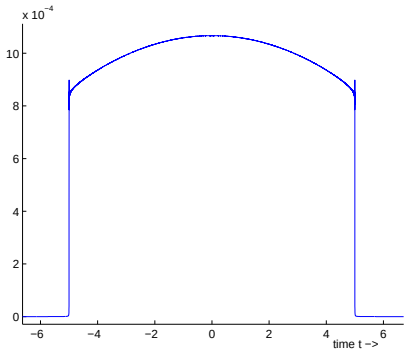
```
clear all; close all;
r = [0.01 0.1 .5 1 2 4];
for l = 1:6

%%This script calculates the optimal window


k = 400;            %number of sinc functions
X = zeros(k-1);
a = r(l);           %Choose some a = frequency resolution in Hz
T=1/a;

%calculate the integral over the product of sinc functies
for i=0:(k-1)
   for j=0:(k-1)
       %compute s_conj_transp * s
       S = @(f)T.*(sinc(f/(2*a)-i)).*T.*(sinc(f/(2*a)-j));
       %compute finite integral
       X(i+1,j+1) = quad(S,-a,a);
   end
end

%Decompose matrix X into UDV, with U and V unitary matrices and D a
%diagonal matrix containing the eigenvalues of X
[U,D,V] = svd(X);

%%
%svd organizes eigenvalues from lowest to highest, so optimal h is the
%first vector of U

h = V(1:k,1);
```

```
%%
%calculate and plot the resulting window, W=T*Sum[h_n*sinc(f/2/a-n)]
f = -1000:0.01:1000;
W = 0;

for i=1:k
    W = W + h(i)*T.*sinc(f/2/a-(i-1));
end

%calculate the time-window by inverse Fourier transformation
w_t = ifft(W);
t = f./25;

figure(1)
hold on
plot(t(100001:200001),abs(w_t(1:100001)))
plot(t(1:100001),(abs(w_t(100001:200001))))

end
```

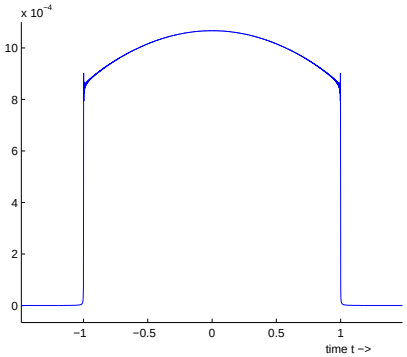
With this code, we generated 6 windows, for $a = 0.1$, 0.5, 1, 2, 4 and 6 Hz. These correspond to time-length signals of 100, 10, 2, 1, 0.5 and 0.25 seconds long. The generated windows are indeed close to zero outside of [-T/2, T/2]. The shape is also to be expected: the Hamming window, which is an improvement of the Hann window, has the same shape as the Hann window, but with straight sides. The generated windows have straight sides too, so it is imagineable they improve the Hamming window. They can be found in Figure 8.6. Two close-ups of the disconituity can be found in the next Figure, 8.7, one taken from the generated window for $a = 0.01$ Hz, one from the window for $a = 4$ Hz.



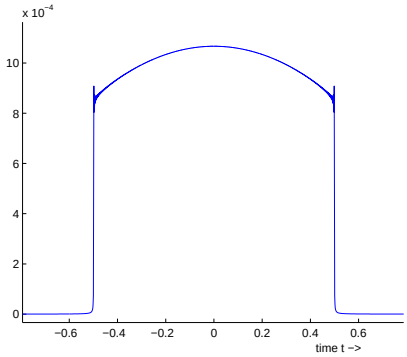
(a) Generated window for $a = 0.01$ Hz, $T = 100$ s



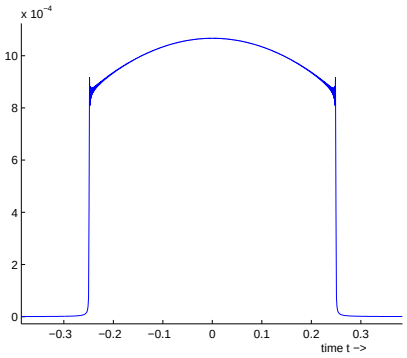
(b) Generated window for $a = 0.1$ Hz, $T = 10$ s



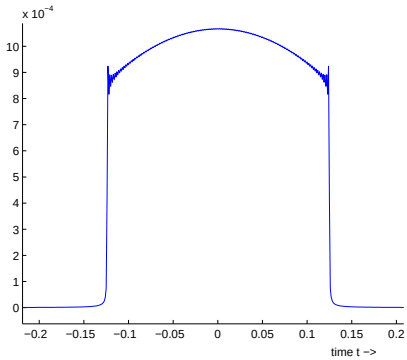
(c) Generated window for $a = 0.5$ Hz, $T = 2$ s



(d) Generated window for $a = 1$ Hz, $T = 1$ s

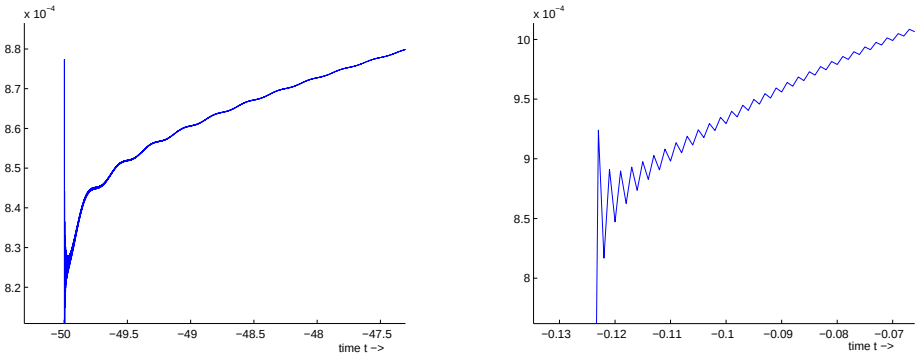


(e) Generated window for $a = 2$ Hz, $T = 0.5$ s



(f) Generated window for $a = 4$ Hz, $T = 0.25$ s

Figure 8.6: Six generated windows, with k = 400



(a) Close-up of generated window for $a = 0.01$ Hz, $T = 100$ s

(b) Close-up of generated window for $a = 4$ Hz, $T = 10$ s

Figure 8.7: Two close-ups of generated windows, with k = 400

Chapter 9

Conclusions and recommendations

9.1 Conclusions

9.1.1 Train detection measuring system

This project resulted in a system indicating the status of detection of the train on which it is installed. This is achieved by a set of measuring clamps, a PCB (Printed Circuit Board) with DAQ functionality, a DSP and a Simulink program programmed on it. Connected to the PCB are two LED-lights, a green and a red one, which signify 'detected' and 'not (surely) detected', respectively.

The system can only measure 'detected' on sections with ATB-EG installed on the tracks and a speed limit above 40 km/h. This is due to the fact that code '40 km/h or slower' and no code have no discerning properties distinghuishing detected from not detected. For these sections a totally different system will have to be developed. For sections with ATB-EG the system detects if the ATB-signal is present in the axle of the train. It checks several properties: a basic frequency of 75 Hz $\pm$ 3 Hz, modulated by a square wave with one of seven code frequencies: 1.25, 1.6, 2, 2.45, 3, $3\frac{2}{3}$ or 4.5 Hz. This square wave should have a duty cycle between 40 and 60 %. Its up- and down-period are bound: the effective value of the up-period should be above 6.5 A and the effective value of the down-period should be below 3 A. Currently the down-period below 3 A is no requirement for code. We would have to reject too much data which appear to be code, except for the down-period requirement. So we assume the high down-period is an error in the circuit next to the tracks and not very important. If desired, the system can give a warning when the down-period is too high.

The overall performance of the system was tested by comparing its conclusions to a human conclusion reached by visual inspection of the measuring data. We assumed the human was always right. Two types of errors are present: we say a red light even though the train is detected is error Type I and green light even though the train is not detected we call error Type II. Now error Type II is heavily punisched, since it can result in a collision. Thus the system was designed for a smaal error Type II. Based on this, the system achieves a reliabitity of 99.23 %, with 0.68 % error Type I and 0.09 % error Type II. Thus the system is not flawless, but it can be a very good addition to the safety system of a work train.

There is one large sidenote, we cannot leave unmentioned. When the data was recorded, it was not registered if the train was actually detected or not. This information could have been gained from looking at the signals, or via telephone contact with the train traffic controllers. So we had to base our program on what we deemed code, which was of course an educated guess. Basically we assumed the train was allways detected, so up to today we are not sure what 'not detected' on a section with ATB looks like except from theory. To solve this, we need to do more test drives, which is not possible at the moment due to lack of funds. But since theory tells us 'not detected' on a section with ATB means no ATB-signal at all, we still think this system will be confirmed by the extra tests.

9.1.2 Frequency resolution

We derived the optimal window to minimize leakaze. This was done by rewriting the window by following the proof of the Shannon Sampling theorem with t and ω interchanged. The the optimalization was done by approaching an infinite matrix by a finite matrix. On this finite matrix we applied the singular value decomposition, which orders the singular values from high to low. Thanks to this ordening and the properties of the unitary matrices calculated by the svd, we could define the optimal window, without using any search algorithm. A Matlab program was written to calculated the optimal windows for a given frequency resolution.

9.2 Recommendations

9.2.1 Testing

We advise testing of the system to validate its conclusions. The train should drive on different sections: single and double legged, with all codes and on sections without ATB to test the error Type II. It is important that during the test drives the status of train detection is monitored, so the results of the system can be compared to reality.

9.2.2 No ATB

In case no ATB is installed on the tracks, a train can still be detected, but we cannot use the convenient ATB current. Unfortunately this means we do not have sufficient information left to distinguish detected from not-detected. Two approaches can be taken to tackle this. We could use information not accesible from the train by allowing radio communication or the use of a database and GPS for more information on the current section. We migth be able to find a different current level for single and double legged sections above which the train is detected. This will probably render a system with a very large Type II error.
The other alternative is to use a totally different approach. Forget about the ATB and train detection currents and create a test to check the resistance of the rail-wheel contact. A system is under development at Strukton Systems whith uses a pulse generator on one axle and a matched filter on another axle to deduct the resistance between the wheel and the rail. If the resistance is low enough, the work train meets the requirements set by ProRail and we assume the train is detected. Since this approach will probably use a lot of energy, it could be combined with the system developed in this thesis. Then it will be applied whenever the red light switches on.

Bibliography

[1] Strukton Systems. ATB-e in werkmaterieel / decoder. Technical report, Strukton Systems, 2006.

[2] R. Mersman and E. van Zuijlen. Gegarandeerde detectie van rail/wegvoertuigen. Technical report, Strukton Systems, 2005.

[3] R. M. Hensen. Treindetectie op voertuigen - Opzet vooronderzoek ontwikkeling van een treindetectie-waarschuwingssysteem op het voertuig. Technical report, Strukton Systems, 2005.

[4] J. Schouten. Decoderen ATB en specificatie ATB. Technical report, Strukton Systems, 2006.

[5] G. S. Bakker, H. C. de Recht, and Dms Zwolle. *Geisoleerde sporen, module 5431.* Nederlandse Spoorwegen, 1990.

[6] F. Walenberg. Treindetectie - Inventarisatie van materieeleigenschappen en mogelijke methoden en systemen voor treindetectie verbetering. Technical report, KEMA Rail Transport Certification, 2005.

[7] D. A. W. Baars. div. vooronderzoeken treindetectie. Technical report, Eleq, 2005.

[8] A. Vloerbergh, A. de Keyser, M. Olthof, and A. Bekkenutte. Theorie treindetectie. Technical report, Strukton Systems, 2006. 1,2 pp.

[9] A. J. Visser and H. Steenkamp. *Spoorstroomlopen, theoriedeel 5462.* Nederlandse spoorwegen, 1981.

[10] ProRail. Meet- en instelvoorschrift C5527/I-2, MS GE 54/3, uitgave C. Technical report, ProRail, 03-07-2001.

[11] ProRail. Meet- en instelvoorschrift C5527/I-2, MS GE 54/7, uitgave B. Technical report, ProRail, 03-07-2001.

[12] Prorail. Richtlijn RLN00018, versie 6. Technical report, ProRail, 20-06-2006.

[13] P. M. Bourdon. Device for use in operating track relays in railway signaling systems. Patent nr 2,024,845, December 17$^{\text{th}}$, 1935.

[14] http://en.wikipedia.org/wiki/Clamp_meter.

[15] http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem.

[16] C. L. Phillips and J. M. Parr. *Signals, Systems and Transforms.* Prentice Hall, 1995.

[17] http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm.

[18] H. Kwakernaak and G. Meinsma. *Time Series Analysis.* Universiteit Twente, Department of Applied Mathematics, 2004.

[19] H. G. ter Morse and G. Meinsma. *Signalen en Transformaties.* Universiteit Twente, Faculteit der toegepaste wiskunde, 2002.

List of Figures

Appendix A

Properties of the Fourier Transform

In this section we discuss some properties of the Fourier Transform, most taken from [19]. Let $g(t)$ be an absolutely integrable signal and $G(\omega)$ its Fourier Transform:

$$\begin{aligned} g(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} G(\omega)e^{i\omega t}d\omega \\ G(\omega) &= \int_{-\infty}^{\infty} g(t)e^{-i\omega t}dt \end{aligned} \tag{A.1}$$

Linearity: $a_1g_1(t) + a_2g_2(t) \overset{\mathbb{F}}{\leftrightarrow} a_1G_1(\omega) + a_2G_2(\omega)$**,** $\forall a_1, a_2 \in \mathbb{C}$

$$\begin{aligned} \mathbb{F}[a_1g_1(t) + a_2g_2(t)] &\Leftrightarrow \int_{-\infty}^{\infty}(a_1g_1(t) + a_2g_2(t))e^{-i\omega t}dt \\ &\Leftrightarrow \int_{-\infty}^{\infty} a_1g_1(t)e^{-i\omega t}dt + \int_{-\infty}^{\infty} a_2g_2(t)e^{-i\omega t}dt \\ &\Leftrightarrow a_1G_1(\omega) + a_2G_2(\omega) \end{aligned} \tag{A.2}$$

Duality: $G(t) \overset{\mathbb{F}}{\leftrightarrow} 2\pi f(-\omega)$ In formulas (A.1) we interchange t and ω and then we replace ω by $-\omega$:

$$\begin{aligned} (\text{A.1}) \quad \overset{t\leftrightarrow\omega}{\Leftrightarrow} & \begin{cases} g(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} G(t)e^{it\omega}dt \\ G(t) = \int_{-\infty}^{\infty} g(\omega)e^{-it\omega}d\omega. \end{cases} \\ \overset{\omega\leftrightarrow-\omega}{\Leftrightarrow} & \begin{cases} 2\pi g(-\omega) = \int_{-\infty}^{\infty} G(t)e^{-it\omega}dt \\ G(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} 2\pi g(-\omega)e^{it\omega}d\omega. \end{cases} \\ \Leftrightarrow & \begin{cases} 2\pi g(-\omega) = \mathbb{F}[G(t)] \\ G(t) = \mathbb{F}[2\pi g(-\omega)] \end{cases} \end{aligned} \tag{A.3}$$

Conjugation: $g^*(t) \overset{\mathbb{F}}{\leftrightarrow} G^*(-\omega)$

$$\begin{aligned} \mathbb{F}[g^*(t)] &= \int_{-\infty}^{\infty} g^*(t)e^{-i\omega t}dt \\ &= \left(\int_{-\infty}^{\infty} g(t)e^{i\omega t}dt\right)^* \\ &= F^*(-\omega) \end{aligned} \tag{A.4}$$

Time scaling: $g(at) \overset{\mathbb{F}}{\leftrightarrow} \frac{1}{|a|}G(\frac{\omega}{a})$**,** $\forall (a \in \mathbb{R}, a \neq 0)$

If $a < 0$, then

$$\begin{aligned} \mathbb{F}[g(at)] &= \int_{-\infty}^{\infty} g(at)e^{i\omega t}dt \\ &\overset{at=\tau}{=} \frac{1}{a}\int_{-\infty}^{\infty} g(\tau)e^{-i\omega\tau/a}d\tau \\ &= \frac{1}{a}G(\frac{\omega}{a} \end{aligned} \tag{A.5}$$

if $a < 0$, then the integral gains a minus sign and we find $\mathbb{F}[g(at)] = \frac{1}{-a}G(\frac{\omega}{a})$
Time shift: $g(t-\tau) \overset{\mathbb{F}}{\leftrightarrow} G(\omega)e^{-i\omega\tau}$

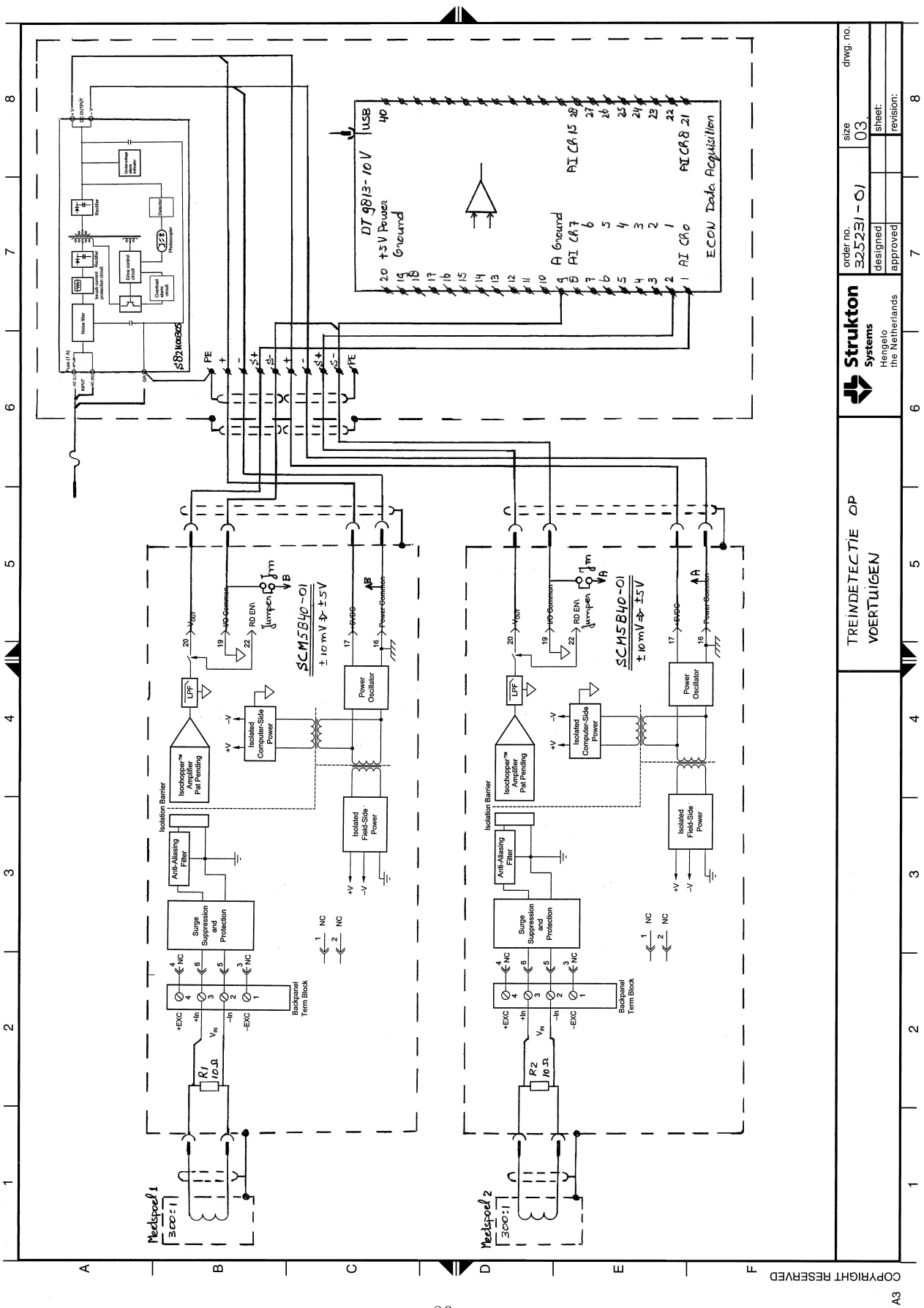
$$\begin{aligned} \mathbb{F}[g(t-\tau)] &= \int_{-\infty}^{\infty} g(t-\tau)e^{-i\omega t}dt \\ &\overset{v=t-\tau}{=} \int_{-\infty}^{\infty} g(v)e^{i\omega(v+\tau}dt \\ &= G(\omega)e^{-i\omega\tau} \end{aligned} \tag{A.6}$$

Frequency shift: $f(t)e^{i\omega_0} \overset{\mathbb{F}}{\leftrightarrow} F(\omega-\omega_0)$

$$\begin{aligned} \mathbb{F}[f(t)e^{i\omega_0}] &= \int_{-\infty}^{\infty} g(t)e^{-i(\omega-\omega_0)t}dt \\ &= F(\omega-\omega_0) \end{aligned} \tag{A.7}$$

Appendix B

Measuring arrangements


TREINDETECTIE OP VOERTUIGEN
Strukton Systems
Hengelo the Netherlands
order no. 325231-01
size 03
drwg. no.
designed
approved
sheet:
revision:
DT 9813-10 V
+5 V Power
Ground
USB
ECON Data Acquisition
AI CH0
AI CH8
AI CH15
SCM5B40-01
±10 mV ⇒ ±5V
Isochopper™ Amplifier Pat Pending
Isolated Computer-Side Power
Isolated Field-Side Power
Power Oscillator
Anti-Aliasing Filter
Surge Suppression and Protection
Backpanel Term Block
Isolation Barrier
Medspoel 1
Medspoel 2
300:1
R1 10Ω
R2 10Ω
COPYRIGHT RESERVED
A3

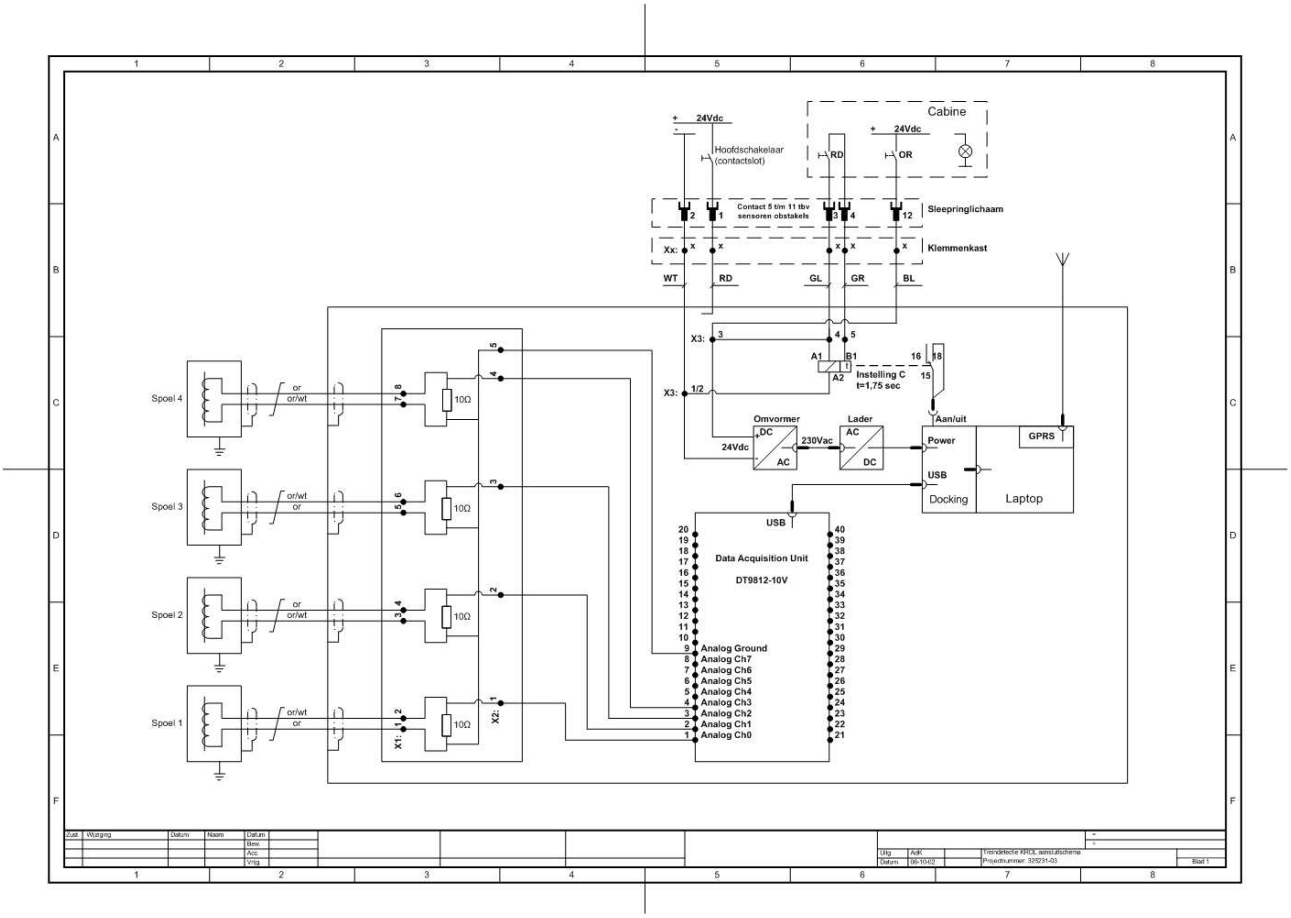


Figure B.2: Schematic representation of measuring arrangement on the KROL

Appendix C

Matlab m-file Lowpass Butterworth filter and FFT analysis

```
close all
clear all

%############### read data
%meetdata = xlsread('1207 emplacement stilstaand voor wissel.xls', 'Sheet18', 'B2:D31992');
%meetdata = xlsread('1209 rijden vooruit emplacement Almelo.xls', 'Sheet24', 'B2:D31992');
%...
%meetdata = xlsread('1245 vrije baan net na railspoel', 'Sheet76', 'B2:D31992');

%begin 1000 Hz
%meetdata = xlsread('1246 sectie wissel en overgang 85km','Sheet77','B2:D31992');
%meetdata = xlsread('1247 rechtuit vrije baan 2x sectiewissel','Sheet78','B2:D31992');
%...
%meetdata = xlsread('1355_1 emplacement almelo langzaam rijden', 'Sheet154', 'B2:D31992');

%############### define some variables
t = meetdata(:,1);           %select time data
m = meetdata(:,2:3);         %select measuring data
fs = 1000;                   %sampling frequency
Ts = 1/fs;                   %sample time
Tf = meetdata(end,1);        %final time
N = length(meetdata(:,2));   %nr of samples
p = 2^(nextpow2(N));         %the next number > nr of samples which is a power of 2

%############### plot data
figure(1)  hold on
plot(t,3*m(:,1), 'r')        %plot the 2 series of measuring data,
plot(t,3*m(:,2), 'b')        %multiplied by 3 for correct amplitude scaling
title('raw data, red = 1st axis, blue = 2nd axis')
xlabel('time (s) ->')
hold off

%############### plot sum data
figure(2)
plot(t,(m(:,2)-m(:,1))*3, 'b')
title('sum of data')
xlabel('time (s) ->')
```

```
%############### calculate and plot messy and/or smoothed fft
M = fft(m(:,1:2),p);                %frequency spectrum, p is power of 2 (for increased speed)
absM = abs(M);                      %absolute value of fft
f = fs.*[0:(p-1)]./p;               %right frequency scale
%[Pm,fm] = pwelch(m,5000,[],p,fs);  %smooth the fft (optional)
figure(3)
plot(f(1:ceil(p/2)),absM(1:ceil(p/2)))     %plot the absolute value of the fft,
title('freqency content of the data')      %for positive frequencies
xlabel('frequency (Hz) ->')
%plot(fm,Pm)                        %plot the smoothed fft (optional)

%############### actual basis frequency
[a,hz] = max(M(:,1));
hz = fs*hz/p

%############## design a butterworth filter
Wp = [.5 3.5]./(fs/2);              %passband from 0.5 Hz to 3.5 Hz
Ws = [0.001 15]./(fs/2);            %crossover region till 0.001 and 15, in which
[n,Wn] = buttord(Wp,Ws,1,20);       %20 dB attenuation is achieved
[b,a] = butter(n,Wn);
figure(3)
freqz(b,a,8192,fs); title('n=2 Butterworth Lowpass Filter')

%############## filter the high frequencies out of the data
y = filter(b,a,abs(m));             %Where abs(m) is your input signal and y is
figure(4)                           %the filtered signal
plot(t,y)
title('gefilterde data (2de orde Butterworth passband 0,1 tot 3 Hz)')
y2 = filter(b,a,abs(m(:,1)-m(:,2))); %filter the absolute value of the sum of the data

%############# plot a smoothed, filtered fft
[Pxx1,f1] = pwelch(y(:,1),[],[],p,fs);  %pwelch windows the data with 8 Hamming windows
[Pxx2,f2] = pwelch(y(:,2),[],[],p,fs);
figure(6)  hold on
plot(f1(1:200),Pxx1(1:200),'r')
plot(f2(1:200),Pxx2(1:200),'b')
title('smoothed filtered fft')
hold off

[Pxx3,f3] = pwelch(y2,[],[],p,fs);  %do the same for the sum of the data
figure(8)
plot(f3(1:200),Pxx3(1:200))
title('smoothed and filtered frequency content of sum')

%############# plot a messy periodogram of data
figure(7)
y3 = (1/p)*(abs(fft(y,p))).^2;
hold on
plot(f(1:200),y3(1:200,1), 'r')
plot(f(1:200),y3(1:200,2), 'b')
title('messy filtered periodogram')
xlabel('frequency (Hz) ->')
hold off
```

Appendix D

Matlab m-file Code filtering

```
close all
clear all

%read data
%begin 3000 Hz
%meetdata = xlsread('1207 emplacement stilstaand voor wissel.xls', 'Sheet18', 'B2:D31992');
%meetdata = xlsread('1209 rijden vooruit emplacement Almelo.xls', 'Sheet24', 'B2:D31992');
%...
%meetdata = xlsread('1245 vrije baan net na railspoel', 'Sheet76', 'B2:D31992');

%begin 1000 Hz
%meetdata = xlsread('1246 sectie wissel en overgang 85km','Sheet77','B2:D31992');
%meetdata = xlsread('1247 rechtuit vrije baan 2x sectiewissel','Sheet78','B2:D31992');
%...
%meetdata = xlsread('1355_1 emplacement almelo langzaam rijden', 'Sheet154', 'B2:D31992');

%############### define some variables
t = meetdata(:,1);
m = 3.*meetdata(:,(2:3));
fs = 1000;                      %sampling frequency (3000 of 1000 Hz)
Ts = 1/fs;                      %sample time
Tf = meetdata(end,1);           %final time
N = length(meetdata(:,2));      %nr of samples
p = 2^(nextpow2(N));            %the next int > N which is a power of 2
%the result of FFT will have 3000 (=fs) frequencies, and these divided
%into p iterations, so one array-step in fequency will be fs/p hz
fstep = fs/p;
codes = [75 96 120 180 220 600];
fcodes = [1.25 1.6 2 3 (3+2/3) 10];
Tcodes = [.8 .625 .5 (1/3) (3/11) .1];

%-----------------------------plot!-----------------------------------------
%############### plot raw data
figure(1)  hold on
plot(t,m(:,1), 'r')
plot(t,m(:,2), 'b')
title('raw data, red = 1st axis, blue = 2nd axis')
xlabel('time (s) ->')
ylabel('amplitude (A) ->')
```

```
hold off

%############### plot sum data
figure(2)
plot(t,(m(:,2)-m(:,1)), 'b')
title('sum of data')
xlabel('time (s) ->')

%############### plot messy fft
M = (1/N)*fft(m(:,1:2),p);          %frequency plot
absM = abs(M);                      %absolute value of fft
f = fstep.*[0:(p-1)];               %right frequency scale
final_v = max(max(absM));

figure(3)
plot(f,absM(:,1:2))                 %plot the absolute value of the fft
axis([0 f(end)/2 0 final_v])
title('freqency content of the data')
xlabel('frequency(Hz) ->')

%-----------------------------check 75 Hz---------------------------------
%############# check basis frequency - using power
%design Butterworth filter
Wp = [72 78]./(fs/2);               %3 Hz passband to each side
Ws = [68 82]./(fs/2);               %2 Hz crossover, then stopband
Rp = 1; Rs = 10;                    %only a little ripple in Wp,
                                    %at least 10 dB reduction in stopband
[n,Wn] = buttord(Wp,Ws,Rp,Rs);
[b,a] = butter(n,Wn);

y_75 = filter(b,a,m(:,1:2));        %filter the data around 75 Hz
figure(4)                           %plot the filtered data
plot(t,y_75)

%calculate and check the rms value
check_rms = rms(y_75(:,1))+rms(y_75(:,2));
if (check_rms > 0.4)
    fprintf('75 Hz is the basic frequency\n\n')
else
    fprintf('NO basic frequency\n\n')
end

%-----------------------code filtering-------------------------------------
%############## filter the code frequencies out of the data
y = zeros(length(m),6);
 % Where m is your input signal and y is the filtered signal
 for j = 1:1:6
     y(:,j) = codefilter(codes(j), abs((m(:,2)-m(:,1))),fs );
     %designs filter and filters sum
 end

%############## root mean square per second
sec = (ceil(Tf)-1);          %number of whole seconds in measurements
rms_ps = zeros(sec,6);
```

```
for i = 1:6
    for k =1:sec
        rms_ps(k,i) = rms( y( (fs*(k-1)+1):(fs*k),i));
    end
end

figure(5)
plot(rms_ps)
legend('code 75','code 96','code 120','code 180','code 220','PSSSL')

%-----------determine code per second & current check & duty-check--------
max_period_s = zeros(sec,2);
max_s = zeros(sec,1);

%Use this for current level check once you know the code
absm1 = abs(m(:,1))+abs(m(:,2));

%use this for duty cycle check once you know the code
absm = abs((m(:,2)-m(:,1)));
a=1; b = 1/80.*ones(1,80);
y_duty = filter(b,a,absm);
y_duty = y_duty - mean(y_duty);

figure(6) hold on
plot(t,y_duty)
hold off

high_low = zeros(1,N);
for s = 1:N
    if y_duty(s) > 0
        high_low(s) = 1;
    end
end

%filter some more to get 1 or two deviating samples out
a=1; b = 1/10.*ones(1,10);  high_low = filter(b,a,high_low);

%########################Dominant code per second:
[C,I] = max( rms_ps');
for j = 1:sec

    %every second, the code with highest RMS is the present code iff the
    %current level and the duty cycle are good.
    if C(j)>.4
       fprintf('t = %g..%g, code %g is dominant\n', j-1, j, codes(I(j)))

       %check current level: maximum value of abs value of sum of currents in one second.
       begin_sec = 1+(j-1)*fs;
       max_s(j) = max( absm1( begin_sec : begin_sec + fs));

       if (max_s(j) > 5.5) && (max_s(j,1) < 20)
            fprintf('High enough current.\n')
       elseif (max_s(j,1) >= 20)
           fprintf('Extremum, no conclusion.\n')
```

```
        else
            fprintf('Current too low - no code.\n')
        end

        %extract duty cycle:
        %now chop high_low into pieces: one period long, at the beginning of each second.
        cycle_sec = high_low( begin_sec: (begin_sec+ceil(fs*(Tcodes(I(j))))));
        start_high_low = cycle_sec(1);

        %check 50% duty-cycle
        i1 = 1; i2 = 1; i3 = 1;
        if start_high_low <= 0.5
                while ((i1 < length(cycle_sec)) && (cycle_sec(i1) < 0.5))
                i1 = i1 + 1;
                end
                while ((i1+i2) < length(cycle_sec))&&(cycle_sec(i1 + i2) > 0.5)
                i2 = i2 + 1;
                end
                while ((i1+i2+i3)<length(cycle_sec))&&(cycle_sec(i1+i2+i3) < 0.5)
                i3 = i3 + 1;
                end
         end

         if start_high_low >0.5
                while (i1<length(cycle_sec))&&(cycle_sec(i1) > 0.5)
                i1 = i1 + 1;
                end
                while ((i1+i2)<length(cycle_sec))&&(cycle_sec(i1 + i2) < 0.5)
                i2 = i2 + 1;
                end
                while ((i1+i2+i3)<length(cycle_sec))&&(cycle_sec(i1+i2+i3) > 0.5)
                i3 = i3 + 1;
                end
        end

        %now check if the three times meet the requirements:
        i123 = i1 + i2 + i3;
        eis_1 = 0; eis_2 = 0; eis_3 = 0;
          if (i123) > (.8*length(cycle_sec))
            eis_1 = 1;
          end
          if ( i2 > (.2 * i123) && ( i2 < (.8*i123) ) )
            eis_2 = 1;
          end
          if (eis_1 == 1) && (eis_2 == 1)
            fprintf('Duty cycle ok\n\n')
          else
            fprintf('NO duty\n\n')
          end
    else
        fprintf('t = %g..%g, no dominant code\n\n', j-1,j)
        C(j)=0;
    end
end
```