Algorithm Development for a Low Cost Tire Localization Method for Tire Pressure Monitoring Systems (TPMS)

Master's thesis

Tifner Patrick

Institute of Microwave and Photonic Engineering
Graz University of Technology

Supervisor: Ao. Univ.-Prof. Dr. Erich Leitgeb

Graz, September 2013

Abstract

Tire Pressure Monitoring Systems (TPMS) became a standard safety feature in the automotive industry within the last years. In the classification of TPMS a distinction is drawn between systems which utilize available sensors in a car to determine the tire pressure, so called indirect TPMS, and systems which have a sensor directly mounted on the wheel. The latter one, the so called direct TPMS, will be topic of this thesis. Both systems measure and observe the tire pressure for all tires of a car and notify the driver when a pressure loss is detected. One challenge in the field of direct TPMS is the mapping of a TPMS module to its tire. As the tires might be interchanged among each other, e.g. car repairs, it is required to check the mapping at each driving cycle. In the course of this thesis an algorithm is developed, utilizing the acceleration sensor of a TPMS, to make this mapping possible. As a direct TPMS has to face the challenge of energy efficiency due to their limited battery to meet lifetime, the algorithm requires to have a short runtime. Several approaches and their basic principles, based on polynomial curve fitting, to solve the tire localisation problem are presented. A simulation framework is developed and used to define the parameter set for each approach - always considering a trade-off between short runtime and a sufficient level of accuracy. The impact of several optimisation methods, developed in the course of this thesis, are presented for the simulation as well as for an implementation on a TPMS. Two solution, a second- and a third-order polynomial approach, are choosen to be implemented on a TPMS and tested in a centrifuge. In conclusion, the comparison between both solutions comes up with a significant advantage of the second-order approach over the third-order approach.

Kurzfassung

Reifendruckkontrollsysteme (engl. „Tire Pressure Monitoring Systems“), kurz TPMS, sind mittlerweile zu einem Sicherheitsstandard der Automobilindustrie geworden. Es wird zwischen zwei Systemen unterschieden. Einerseits Systeme, welche sich die vorhandene Sensorik im Fahrzeug zunutze machen, um den Reifendruck zu überwachen, so genannte indirekte TPMS und andererseits die direkten TPMS, welche über einen Drucksensor im Inneren des Reifens den Reifendruck erfassen. Die vorliegende Arbeit beschäftigt sich mit den direkten TPMS. Beide Systeme verfolgen den Zweck, den Reifendruck am Fahrzeug zu überwachen und bei einem Druckabfall den Fahrer zu informieren. Eine der Herausforderungen bei direkten TPMS ist die Zuordnung der einzelnen TPMS-Module zu dem zugehörigen Reifen. Da die Reifen untereinander ausgetauscht werden können, z.B. bei Reparaturarbeiten am Auto, muss diese Zuordnung zu Beginn jedes Fahrzyklus durchgeführt werden. Im Verlauf dieser Arbeit wird ein Algorithmus entwickelt, welcher, mithilfe eines Beschleunigungssensors im TPMS, diese Zuordnung ermöglicht. Nachdem ein direktes TPMS darauf ausgerichtet ist, dass seine Batterie über den gesamten Lebenszyklus besteht, muss der Algorithmus eine möglichst kurze Laufzeit aufweisen. Verschiedene Ansätze, basierend auf polynomieller Kurvenanpassung und deren mathematischen Grundlagen, werden hierbei vorgestellt. Ein Framework zur Simulation dieser Ansätze und zur Bestimmung der einzelnen Parametersätze wird entwickelt, immer in Hinblick auf die Abwägung der Laufzeit und Genauigkeit. Die Auswirkungen verschiedener Optimierungsverfahren, welche im Verlauf erarbeitet werden, werden sowohl anhand von Simulationsergebnissen als auch mithilfe der Messergebnisse aus einer Implementierung in einem TPMS präsentiert. Zwei Ansätze, ein quadratisches und ein kubisches Polynom, wurden aufgrund der Simulationsergebnisse in die engere Auswahl genommen, auf einem TPMS implementiert und in einer Zentrifuge getestet. Letztendlich zeigte der Ansatz des quadratischen Polynoms signifikante Vorteile gegenüber dem des kubischen Polynoms.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……. ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……. ……..
date (signature)

Contents

List of Figures

1 Introduction

From 1 November 2012, 12 years after the US Department of Transportation legislated the TREAD Act [13], Tire pressure monitoring system (TPMS) became mandatory in Europe. EU Regulation No 64 states for the type-approval of new types of passenger cars that those need to be equipped with

> „[...] a system fitted on a vehicle, able to perform a function to evaluate the inflation pressure of the tyres or the variation of this inflation pressure over time and to transmit corresponding information to the user while the vehicle is running [4].“

From 1 November 2014 all new passenger cars in the EU must be equipped with a TPMS. TPMS aims to avoid under- and over-inflation of the cars tires and the risks that come along with this. The surface area of the tires contact patch is related to the pressure. The more under-inflated a tire is, the larger is it's contact patch and therefore the higher the tires rolling resistance. A pressure loss of 25% could cause an increase of the tire rolling resistance coefficient by about 16.4%. It is assumed that TPMS could cut down fuel costing by 8.1 billion litres in the EU. [15] The resulting increase in fuel consumption and CO2 emission is not the only threat. The cavity air temperature increase with pressure loss as well and leads to an increase of tire abrasion.
The target of TPMS is therefore

- avoiding traffic accidents caused by tire-malfunction
- reduction of CO_2 emission
- reduction of tire abrasion

A study requested by the European Parliament [17] pointed out that the majority of vehicles (70%) run on under-inflated tyres. Assuming that a TPMS could avoid an deflation of 0.5 bar the fuel efficiency could be improved by 3%.

1.1 Indirect TPMS

Indirect TPMS are software systems which utilize the wheel speed sensor of anti-lock brake system (ABS) or the traction control system (TCS). There exist two common methods: *wheel radius analysis* and *vibration analysis*. The first one makes use of the effect that a tires effective rolling radius r_e is proportional to this tires pressure (with respect to the speed). A decrease of the tire pressure thus leads to an increase of the wheel speed ω in case the car speed stays consistent ($\omega \sim r-1$). Equation 1.1 is an example for a static relation to detect under- or over-inflation [12].

$$r = \frac{\omega_{FL}}{\omega_{FR}} - \frac{\omega_{RL}}{\omega_{RR}} = \frac{r_{FR}}{r_{FL}} - \frac{r_{RR}}{r_{RL}} \tag{1.1}$$

The allocation of a wheel speed ω to a tire in equation 1.1 is denoted by its index. *FL* stands for front right, *RR* for rear right and so on. The same applies to the radii r. A pressure loss can be detected when r becomes non-zero.
In the *vibration analysis* the tire is modelled as a spring-damper system in vertical and torsional direction. The vertical model describes the effective rolling radius which is similar to the wheel radius analysis. A deviation from the nominal tire radius is equal to a variation of the spring constant. The other aspect described by the vertical spring-damper model is the vertical oscillation of the tire due to road conditions. These oscillation frequencies are usually in the range of 10 to 20 Hz but can, due to a variation of the spring constant respectively the tire pressure, be shifted to a higher frequency in case of over-inflation or a lower frequency in case of under-inflation. The torsional spring damper model is excited by road roughness and road irregularities. Its vibrations directly affect the wheel speed and the majority of the oscillation frequencies lie in the range of 40Hz to 50Hz. Vibration analysis independently evaluates each tire and unlike wheel radius analysis it enables to detect pressure losses in all tires [12] [11].

Limitations of indirect TPMS: The main disadvantage of the *wheel radius analysis* method is that an equal pressure loss, either on the same side/axle or on all tires, cannot be determined. The *vibration analysis* can detect under-inflation in up to all four tires simultaneously. For this purpose the software either needs a model of the tire stored in memory of the Engine control unit (ECU), or it has to compute one. Each of these methods can detect a pressure loss of 25% of the nominal pressure, a combination of both methods raise the precision so that a pressure loss of 15% of the nominal pressure can be detected [12].

1.2 Direct TPMS

Direct TPMS can physically measure the tire pressure and are either sticked to the tire or, which is common for aftermarket solutions, are part of the valve. As an active system direct TPMS has to face the challenge of energy efficiency due to their limited battery to meet lifetime. Taking the example of a typical TPMS module this system incorporates two separate blocks for either sending and receiving data. The LF Receiver, which meets the requirements for low power consumption, enables the vehicle to trigger pressure measurements or to update configuration data. On the other hand an RF Transmitter, which has a lower power consumption then an LF Transmitter would have, is used to send information to the vehicle. For the data transmission between TPMS and the ECU 2 methods exist. The bidirectional method the sensor module send the pressure information after an LF-Trigger from the ECU. For this purpose a LF-Transmitter need to be mounted in each tires wheel housing. In the second method, the unidirectional

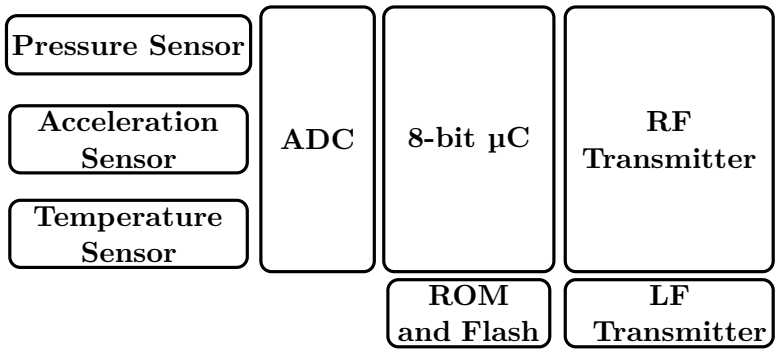


Figure 1.1: TPMS module - block diagram.

method, the sensor modules send autonomously. There is only one RF Receiver in the car. Thus, this method requires strategies in collision handling as well as a method for tire localisation. For the sake of completeness another method is to send messages more frequently when a pressure loss is detected in order to increase the probability that the message is received. While the bidirectional method is the common solution in high-price car's, the unidirectional method is used in the majority of vehicles which use direct TPMS due to its lower costs.

1.3 Tire localisation in Direct TMPS

The topic of tire localisation deals with the mapping of an TPMS ID to its tire. At the start of each driving cycle this process needs to be accomplished, since the system cannot assume the wheel arrangement hasn't changed. The TPMS wheel unit sends autonomously, but at a certain position of wheel, messages containing its ID and pressure information.

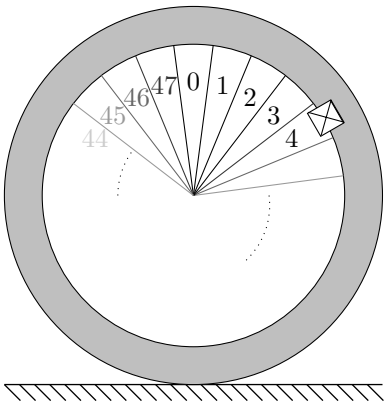


Figure 1.2: Angular positions of a wheel with its associated counter values.

In the following we consider the four wheels of a car, denoted as front left (FL), front right (FR), rear left (RL) and rear right (RR). The angular position of each wheel is represented by the counter value of ABS or ESC. For this purpose a tooth wheel is mounted on each wheel. The number of teeth define the count of angular positions that could be distinct. When the TPMS control unit receives a message from a wheel unit, it requests the counter values of all tires. The TPMS control unit must be able to read all counter values simultaneously. In the example in Figure 1.2 one revolution of the tire is split into 48 intervals. A certain position of the tire correspond to a certain - but unknown - counter value. In this example the angular position of the TPMS wheel unit correspond to counter value 4. After a number of trials the TPMS control unit has assembled a table with the counter values of all wheels for a specific ID. Exemplarily Table 1.1 shows the counter values of all tires for each message received from wheel unit A. Since each wheel unit transmit always at a certain angular position, only the counter values corresponding to wheel unit A will have similar values. This can easily be computed column wise by substraction of the mean and summation of the values in each column afterwards. TPMS wheel unit A belongs to the column which yields the smallest sum.

#	Wheel angle counter values				Wheel unit id
	FL	FR	RL	RR	
1	5	2	17	5	A
2	4	10	22	10	A
3	3	40	43	41	A
4	5	41	0	29	A
5	3	22	33	9	A
6	3	4	11	15	A

Table 1.1: Wheel angle counter table for TPMS ID A.

In this example this would be the front left tire.

The purpose of this thesis is to describe the development of a method to determine the transmission time of an TPMS wheel unit which correspond to a certain angular position of the wheel unit. As shown in Figure 1.1 the TPMS module has a build in acceleration sensor which is utilized to determine the angular position of the wheel unit. Due to the hardware limitation of the TPMS the designed algorithm needs to work efficient and with tread to the limited resources which are memory as well as energy respectively runtime.

1.4 Thesis Outline

In the course of this thesis we will develop and examine an algorithm to perform tire localisation. At the beginning of this thesis, in Chapter 2, the derivation of the acceleration sensor model is explained. This model is the basic principle for all further considerations in the topic of tire localization.
Chapter 3 covers the principles of polynomial curve fitting with focus on quadratic and cubic functions. Along with the fundamentals of curve fitting an iterative method, which form the basis for the algorithm to develop, is presented and some first improvements on this algorithm will be shown.
Chapter 4 is about the evaluation of the parameters for the curve fitting algorithm. Since there are many degrees of freedom, an evaluation concerning several aspects is required. The results of a simulation scenario, which enables to test the curve fitting algorithm as a function of these parameters, will be used to analyse the behaviour of the algorithm as subject to certain conditions. In addition an estimation on the computational effort of the algorithm for quadratic and cubic curve-fits is performed.
With the specified parameter set we examines the performance of the algorithm and compare it to a sinusoidal based approach of the tire localisation algorithm in Chapter 5. Based on real measurement data we can also simulate the expected behaviour on chip.
Chapter 6 deals with the implementation of the algorithm in hardware. Due to hardware limitations we have to give thoughts about an implementation in fixed point arithmetic, avoiding a loss in precision and achieving reasonable runtime performance.
Chapter 7 is about the optimisation of the algorithm. In the course of the evaluation of the algorithms performance three methods came up to optimize the algorithm in terms of runtime

and precision. One of them allows to abort computations in an early stage and therefore enables time-savings, the second method introduces threshold levels to gain certain degrees of precision and the last method increases the precision by comparatively small additional computational effort. At the end of this Chapter you find a flow chart of the algorithm considering these optimisation methods.
Chapter 8 deals with the performance evaluation of the algorithm on chip. Therefore a centrifuge is used which enables us to let the sensor rotate at a specified constant frequency and to measure the precision for certain scenarios. Beside the precision of the algorithm we will also examine the gain through the optimisation methods found in Chapter 7 and focus on the runtime performance for all these implementations. The last Chapter summarizes this thesis and gives an outlook on possible future developments of the developed algorithm.

2 Acceleration Sensor Model

As stated in the previous chapter, the TPMS has a build-in acceleration sensor to measures the acceleration in radial direction. Figure 2.1 illustrate the acceleration components which occur on a turning wheel. For sake of simplicity we consider constant angular velocity $\omega(t)$.

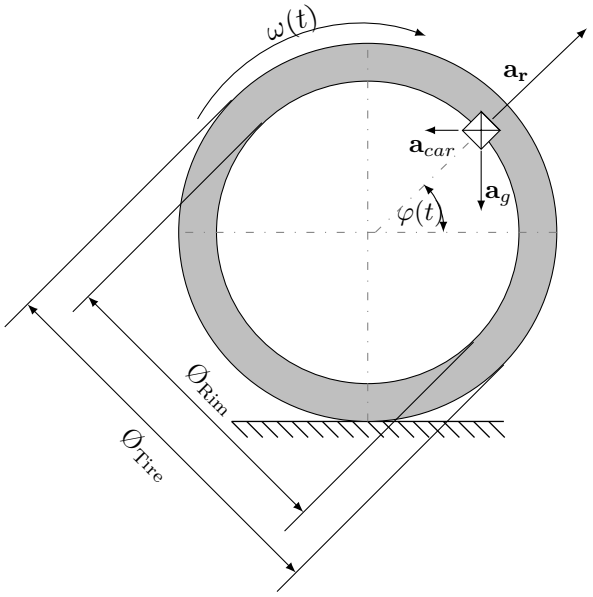


Figure 2.1: Components of acceleration.

The sensor output signal $\mathbf{a}$ is composed of the three acceleration components:

- Gravity component $\mathbf{a_g}$
- Car acceleration component $\mathbf{a_c}$
- Centrifugal acceleration component $\mathbf{a_r}$

The gravity component is the acceleration due to force of gravity and causes an oscillation of the sensor signal around ±1g. The frequency of this component is equal to the number of tire revolutions per second.

$$a_g = g \cdot sin(\varphi(t)) \tag{2.1}$$

If the car is accelerating or decelerating $\mathbf{a_{car}}$, which is orthogonal to the gravity component, is acting on the sensor. For constant velocity $\mathbf{a_{car}}$ is zero. The car acceleration component $\mathbf{a_c}$ describes this acceleration.

$$\mathbf{a_c} = \mathbf{a_{car}} \cdot cos(\varphi(t)) \tag{2.2}$$

The last component is the radial component, also known as centrifugal acceleration, which is calculated as follows:

$$a_r = \omega^2 \frac{\varnothing_{Rim}}{2} \tag{2.3}$$

Compared to the other components a_r is independent of the angle $\varphi(t)$ and therefore constant for a uniform circular motion. For constant velocity $v_{car} = const. \Rightarrow \omega(t) = const. = \omega$ the phase angle is given by $\varphi(t) = 2\pi ft + \varphi_0$.
The measured acceleration therefore add up to:

$$\begin{aligned} a(t) &= \underbrace{a_r}_{\text{const.}} + g \cdot \sin(2\pi ft + \varphi_0) + \underbrace{a_c \cdot \cos(2\pi ft + \varphi_0)}_{=0} + e(t) \\ a(t) &= a_r + g \cdot \sin(2\pi ft + \varphi_0) \; + e(t) \end{aligned}$$

The term $e(t)$ stands for a noise signal, super-imposing the sensor signal. In case of a non-uniform circular motion it can be shown, that the acceleration components are functions of time as well as velocity. Beginning with the definition of the velocity :

$$v_{car} = \frac{2\,\pi\,\varnothing_{Tire}}{2\,T} = 2\,\pi\,f\,\frac{\varnothing_{Tire}}{2} \qquad \xrightarrow{\text{with f = f(t)}} \qquad v_{car}(t) = 2\pi f(t)\frac{\varnothing_{Tire}}{2} = \omega(t)\frac{\varnothing_{Tire}}{2} \tag{2.4}$$

By definition the angular velocity is the derivative of $\varphi(t)$ with respect to time.

$$\omega(t) = \frac{d\varphi(t)}{dt} \tag{2.5}$$

From equation 2.4 and 2.5 it follows that the phase angle is a function of time and velocity as well:

$$\varphi(v,t) = \int \omega(t)\mathrm{d}t + \varphi_0 = 2 \cdot \int \frac{v(t)}{\varnothing_{Tire}}\mathrm{d}t + \varphi_0$$

Thus the acceleration a(t) for a non-uniform circular motion add up as follows:

$$a(t) = v_{car}(t)^2 \cdot \frac{\varnothing_{Tire}}{2} + g \cdot \sin(2 \cdot \int \frac{v(t)}{\varnothing_{Tire}}\mathrm{d}t + \varphi_0) + a_{car}(t) \cdot \cos(2 \cdot \int \frac{v(t)}{\varnothing_{Tire}}\mathrm{d}t + \varphi_0) + e(t) \tag{2.6}$$

Equation 2.6 form the basis for the simulation model introduced in Chapter 4. Summing up the sensor signal is varying in time as well as with respect to the cars velocity. For a non-uniform circular motion, that is to say an acceleration of the car, the magnitude of the measured signal as well as the frequency of the oscillation around this magnitude changes as well.

Based on the phase $\varphi(v,t)$ one can draw conclusion on the corresponding angular position of the TPMS wheel unit. E.g. the extremal points correspont with either the top position (90°) or the bottom position (270°) of the TPMS wheel unit.

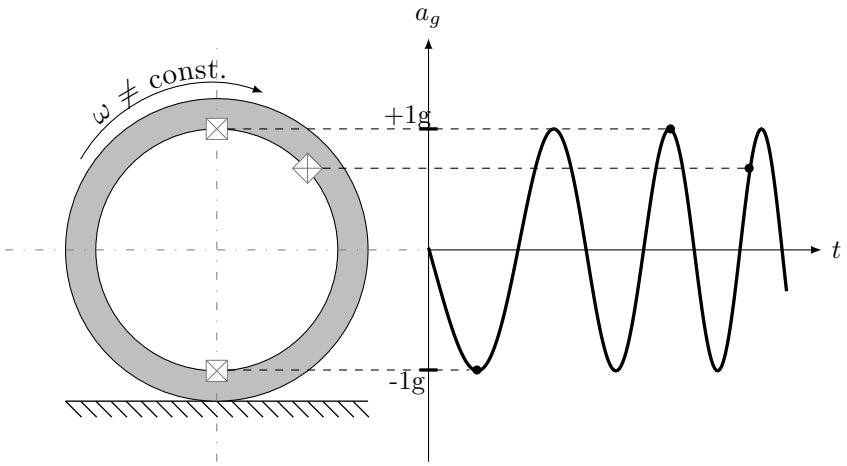


Figure 2.2: Correlation between phase angle and angular position of TPMS wheel unit.

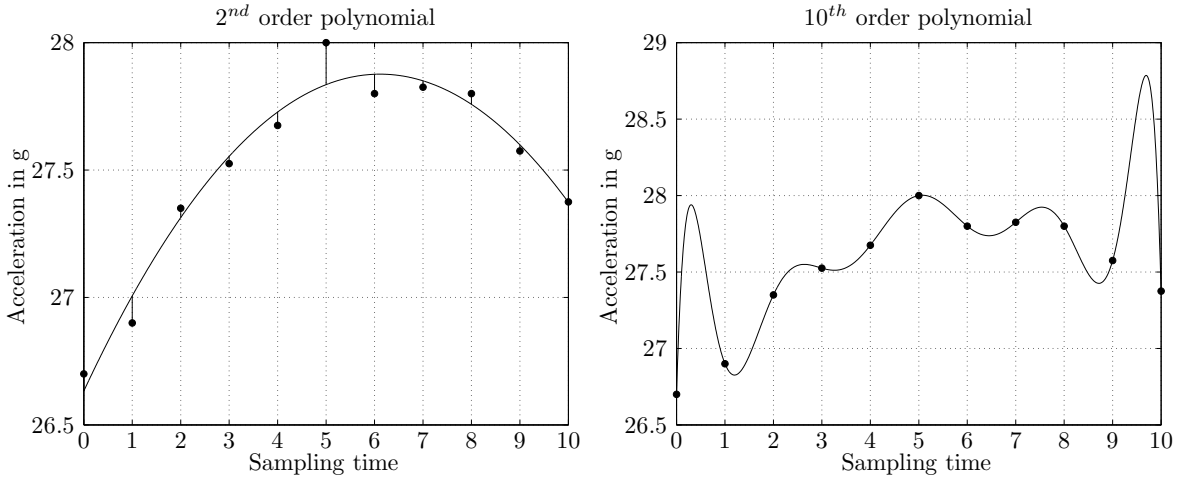
As depicted in Figure 2.2 the frequency of the gravity component a_g changes for non-uniform circular motion ($\omega \neq$ const). The tire localisation algorithm, in the following denoted as APS for angular positioning sensing, takes advantage of this correlation between phase angle and angular position of the wheel.

3 Curve fitting

In the following chapter the basic ideas behind curve fitting will be discussed. The importance of an appropriate fitting function as well as ways of optimisation will be shown. An iterative algorithm to solve a system of equations in sense of least squares is introduced and its performance is investigated.

3.1 Curve types

Knowledge of the curve shape, represented by data points, helps to decide which type of fitting function to choose. This could either be trigonometric functions, conic sections or a polynomial. Only polynomial functions will be topic of this chapter. Figure 3.1 shows the impact on the order of the polynomial degree chosen. The graph of the higher degree polynomial shows a disturbing curve progression between the sampling points and gives a poor representation of the function. This effect is called *overfitting* and comes up when the fitting function aligns to the noise rather than the function itself. [1] Especially near the ends of the interval overshooting between the samples grows stronger. [5] This effect can be explained by *Runge's phenomenon* which states that for polynomials of higher order a equidistant interpolation fits quite well for a region near the center of an observation interval but *'is in some cases an ill-conditioned problem, especially in the outer parts of the interval.'*[3]. As a rule of thumb the polynomials degree should be choosen as $N < 2 \cdot \sqrt{M}$, where M is the number of samples, to get a well conditioned fit. [3]

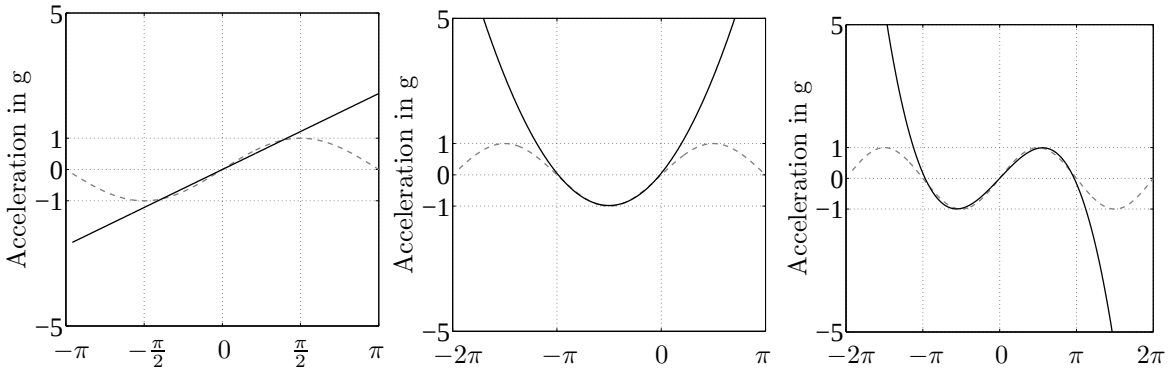


(a) 2^{nd} degree curve fit

(b) 10^{th} degree curve fit

Figure 3.1: Impact of polynomials degree on quality of curve fit.

In order to choose an adequate polynomial degree a study of the progression of the approximating curve is inevitable. As mentioned in Chapter 2 the sensor signal, in case of constant velocity of the car, is a sinusoidal curve with an amplitude of ±1g. For the sake of short runtime and reasonable computational complexity the number of samples as well as the polynomial degree of the fitting function shall be rather small. When examine a sinus curve and break the observation interval down starting with less then a complete period, the progression of the signal can be described by polynomials of different degree. Either, when there are more then one extremal points on the observation interval, a polynomial of third degree can describe the progression of the curve between them quite well or. When there is only one maximum or minimum in the observation interval, a quadratic polynomial can fit the sinus around them.



(a) Linear curve fit. (b) Quadratic curve fit. (c) Cubic curve fit.

Figure 3.2: Progression of curve fit for several polynomial degrees.

Figure 3.2 depict these described curve fit. The grey curve represents the gravity component a_g of the acceleration signal, as defined in 2.1. For the linear curve fit an observation interval T_{obs} between $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ was chosen. In case of the quadratic curve fit T_{obs} went from $-\pi$ to zero, comprising a vertex. For the quadratic curve fit it is mandatory to have an extremal point in the observation interval, otherwise the deviations of the fitted curve from the observed curve might be to intense. If, for a quadratic curve fit, the same observation interval as for the linear curve fit would have been chosen, the result of the curve fit would be a linear curve as well. For the cubic curve fit the observation interval covered a complete period, from $-\pi$ to π, comprising two extremal points.

3.2 General curve fitting

One usually obtains a set of M data-points $(n_1, y_1), ..., (n_M, y_M)$, where n denotes the time and y the amplitude, as result of a measurement. Curve fitting aims to find a mathematical function to approximate this dataset with the best accuracy. It has to be distinguished between interpolation, which deals with an exact fit of the dataset, or extrapolation, which deals with curve fitting beyond the observation interval. The latter is not topic of this chapter. Starting with a general polynomial function of N-th degree the measurement results can be handled as a system of linear equations.

$$\begin{array}{lcl} c_0 + n_1 \cdot c_1 + n_1^2 \cdot c_2 + ... + n_1^N \cdot c_N & = & y_1 \\ \vdots & \vdots & \vdots \\ c_0 + n_M \cdot c_1 + n_M^2 \cdot c_2 + ... + n_M^N \cdot c_N & = & y_M \end{array} \tag{3.1}$$

Equation 3.1 can be rewritten in general matrix-vector notation. Matrix $\mathbf{A}$ is a so called Vandermonde matrix.

$$\mathbf{A} \cdot \mathbf{c} = \mathbf{y} \tag{3.2}$$

$$A_{i,j} = n_i^{j-1}$$

$$\mathbf{A} = \begin{bmatrix} n_1 & n_1^2 & ... & n_1^N \\ \vdots & \vdots & ... & \vdots \\ n_M & n_M^2 & ... & n_M^N \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}$$

On the basis of the ratio between measured data-points and unknowns there can be distinguished between 3 different cases:

The Nonsingular Case: The number of equations equals the number of coefficients M = N. As long as non of the equations depend on another, which implies that $\mathbf{A}$ is square and non-singular, equation 3.2 can be rewritten to $\mathbf{c} = \mathbf{A^{-1}} \cdot \mathbf{y}$ and solved directly [16].

The Underdetermined Case: As there are less equations available then unknowns, no unique solution exists.[16]

The Overdetermined Case: In this case there are more equations then unknowns. This means the system is overdetermined and no unique solution exists. The system need to be solved in sense of a least squares (LS) solution, which intends to minimize a residual $\mathbf{r}$. The residual is the vertical distance between the observed dataset and the values provided by the fitting function.[16] and can be considered as a criteria for the accuracy of a fit.

$$r_i = y_i - F(n_i)$$

With $F(n)$ a general polynomial function of N_{th} degree.

$$F(n_i) = c_0 + c_1 \cdot n_i + c_2 \cdot n_i^2 + ... + c_N \cdot n_i^N$$

A least squares fit is obtained by selecting a coefficient vector $\mathbf{c}$ whereby the sum of the squares of the residual becomes a minimum.

$$\sum_{i=1}^{M} r_i^2 = min$$

In vector-matrix notation the solution can be stated as follows:

$$\begin{aligned} \|\mathbf{r}\|^2 =& \|\mathbf{A}\cdot\mathbf{c}-\mathbf{y}\|^2 = (\mathbf{A}\cdot\mathbf{c}-\mathbf{y})^T(\mathbf{A}\cdot\mathbf{c}-\mathbf{y}) \\ =& \mathbf{y^T y} - 2\cdot\mathbf{y^T A c} + \mathbf{c^T A^T A c} \end{aligned} \tag{3.3}$$

$$\begin{aligned} \frac{\partial \mathbf{r}}{\partial \mathbf{c}} =& \mathbf{A}^T(\mathbf{A c}-\mathbf{y}) \\ \mathbf{c} =& (\mathbf{A^T A})^{-1}\mathbf{A^T y} \end{aligned} \tag{3.4}$$

In the following this solution will be referred to as the *solution of the pseudo-inverse.*

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} \sum_{i=1}^{M} n[i]^2 & \sum_{i=1}^{M} n[i]^3 & \dots & \sum_{i=1}^{M} n[i]^2 \\ \sum_{i=1}^{M} n[i]^3 & \sum_{i=1}^{M} n[i]^4 & \dots & \sum_{i=1}^{M} n[i]^{N+1} \\ \vdots & & & \vdots \\ \sum_{i=1}^{M} n[i]^{N+1} & \sum_{i=1}^{M} n[i]^{N+2} & \dots & \sum_{i=1}^{M} n[i]^{N+N} \end{bmatrix} \tag{3.5}$$

3.3 Quadratic fit

The following section deals with the basic calculation of a curve fit for a quadratic polynomial. Furthermore a simple method to reduce computational effort will be shown.
Given the polynomial function for second degree and a general number of samples M.

$$F(n_i) = c_1 + c_2\cdot n_i + c_3\cdot n_i^2 \qquad \mathbf{A} = \begin{bmatrix} 1 & n_1 & n_1^2 \\ \vdots & \vdots & \vdots \\ 1 & n_M & n_M^2 \end{bmatrix}$$

Following Equation 3.4 the calculation of the matrix product $\mathbf{A^T A}$ is mandatory. As one can see in the equation below this matrix product is rather complicated and complexity grows by increase of M.

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} M & \sum_{i=1}^{M} n_i & \sum_{i=1}^{M} n_i^2 \\ \sum_{i=1}^{M} n_i & \sum_{i=1}^{M} n_i^2 & \sum_{i=1}^{M} n_i^3 \\ \sum_{i=1}^{M} n_i^2 & \sum_{i=1}^{M} n_i^3 & \sum_{i=1}^{M} n_i^4 \end{bmatrix} \tag{3.6}$$

Overall $4\cdot(M-1)$ additions and $5M$ multiplications are necessary for this matrix product.

Optimisation: In order to reduce the number of computations one way to do so is to move the sampling-time vector symmetric around zero which requires an odd number of samples (M = odd).

$$\mathbf{x} = \left[-\lfloor\frac{M}{2}\rfloor, -\lfloor\frac{M}{2}-1\rfloor, \ldots, 0, \ldots, \lfloor\frac{M}{2}-1\rfloor, \lfloor\frac{M}{2}\rfloor\right]^T$$

Equation 3.7 shows the optimized terms of the matrix product $\mathbf{A^T A}$ for an odd sampling-time vector placed symetric around zero.

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} M & 0 & 2\cdot\sum_{\lceil\frac{-M}{2}\rceil}^{-1} n_i^2 \\ 0 & 2\cdot\sum_{\lceil\frac{-M}{2}\rceil}^{-1} n_i^2 & 0 \\ 2\cdot\sum_{\lceil\frac{-M}{2}\rceil}^{-1} n_i^2 & 0 & 2\cdot\sum_{\lceil\frac{-M}{2}\rceil}^{-1} n_i^4 \end{bmatrix} \tag{3.7}$$

The number of computations has been brought down to $6\cdot\left(\lceil\frac{M}{2}\rceil\right)$ multiplications and $2\cdot\left(\lceil\frac{M}{2}\rceil-1\right)$ additions. Table 3.1 summarizes the numer of operations for the overall computation of the coefficient vector $\mathbf{c}$ for the optimized and the general solution as stated in 3.4. As in general the number of sampling values is predefined and does not change, most of this values can be precomputed and therefore do not affect the runtime.

$$\sum_{i=1}^{M} n_i = 0 \qquad \sum_{i=1}^{M} n_i^3 = 0$$

$$\sum_{i=1}^{M} n_i^2 = 2\cdot\sum_{\lceil\frac{-M}{2}\rceil}^{-1} n_i^2 \qquad \sum_{i=1}^{M} n_i^4 = 2\cdot\sum_{\lceil\frac{-M}{2}\rceil}^{-1} n_i^4$$

The sums of the negative and positive sampling times are always equal. Only the signs are opposite in case of an odd exponent.

	# multiplications	# additions
optimized	$6\cdot\left(\lceil\frac{M}{2}\rceil\right)$	$2\cdot\left(\lceil\frac{M}{2}\rceil-1\right)$
non-optimized	$5M$	$4\cdot(M-1)$

Table 3.1: comparison of number of operations

The times of the extremal points, computed with this optimisation, need to be transformed into the proper time domain afterwards. However, this transformation is only a simple addition.
The considerations above can be applied to a third order curve fit in the exact same manner.

3.4 Gauss-Seidel - an iterative approach

Gauss-Seidel is an iterative technique for solving a set of linear equations. An iterative algorithm always seeks for a form like $c^{k+1} = G(c^k)$ - the output depends on the last state.

$$c[n+1] = G(c[n])$$

In order to bring equation 3.2 into such a form matrix decomposition needs to be applied. We therefore assume the Vandermond Matrix to be represented in form of a square matrix $\mathbf{Q}$. *LDU decomposition* factorizes a square matrix into a diagonal matrix $\mathbf{D}$, a lower triangle matrix $\mathbf{L}$ and a upper triangle matrix $\mathbf{U}$.

$$\begin{aligned} \mathbf{Qc} &= \mathbf{y} \\ (\mathbf{U} + \mathbf{L} + \mathbf{D})\mathbf{c} &= \mathbf{y} \end{aligned} \tag{3.8}$$

Equation 3.8 can now be rewritten as follows:

$$\begin{aligned} (\mathbf{L} + \mathbf{D})\mathbf{c} &= -\mathbf{Uc} + \mathbf{y} \\ \mathbf{c} &= -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{Uc} + (\mathbf{L} + \mathbf{D})^{-1}\mathbf{y} \end{aligned} \tag{3.9}$$

The elements of the coefficient vector $\mathbf{c}$ are obtained sequentially by *forward substitution*.

$$\mathbf{c}^{k+1} = -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U}\mathbf{c}^k + (\mathbf{L} + \mathbf{D})^{-1}\mathbf{y} \tag{3.10}$$

In component form the equation looks the following:

$$c_i^{k+1} = \frac{y_i}{q_{i,i}} - \frac{1}{q_{i,i}} \sum_{j=1}^{i-1} q_{i,j} c_j^{k+1} - \frac{1}{q_{i,i}} \sum_{j=i+1}^{n} a_{i,j} c_j^k \tag{3.11}$$

In order to apply this algorithm on the curve fitting problem in Equation 3.2 a number of prerequisite conditions need to be met. The LDU decomposition requires a square matrix $\mathbf{Q}$.

$$\mathbf{Q} \in \mathbb{R}^{N \times N}$$

As the system is overdetermined, which means M > N, $\mathbf{A} \in \mathbb{R}^{M \times N}$ is a non-square matrix. By multiplication of both sides of equation 3.2 with $\mathbf{A}^T$ one gets a square-matrix $\overline{\mathbf{A}}$ and a vector $\mathbf{b} \in \mathbb{R}^{N \times 1}$.

$$\begin{aligned} \mathbf{A}^T \cdot \mathbf{y} &= \mathbf{A}^\mathbf{T}\mathbf{A} \cdot \mathbf{c} \\ \mathbf{b} &= \overline{\mathbf{A}} \cdot \mathbf{c} \end{aligned}$$

To guarantee convergence of the algorithm the matrix $\overline{\mathbf{A}}$ needs to be square as well as strictly diagonally dominant (SDD). SDD requires for each row that the sum of all off-diagonal elements in a row have a lower magnitude then the main diagonal element of that row:

$$\overline{a}_{jj} \overset{!}{>} \sum_{j=1, j \neq i}^{N} |\overline{a}_{ij}|$$

To comply $\overline{\mathbf{A}} = \mathbf{A^T A}$ with the rule of SDD, we need to calculate the euclidean norm of each column in $\mathbf{A}$

$$\|a_i\| = \sqrt{a_{1,i}^2 + \cdots + a_{M,i}^2}.$$

and devide each element in the belonging column of $\overline{\mathbf{A}}$ by the norm.

$$\overline{\mathbf{A}} \cdot \begin{bmatrix} \|a_1\| & 0 & 0 \\ 0 & \|a_2\| & 0 \\ 0 & 0 & \|a_3\| \end{bmatrix}^{-1} \tag{3.12}$$

The component form from equation 3.11 can therefore be rewritten as follows:

$$\begin{aligned} c_i^{k+1} =& b_i - \sum_{j=1}^{i-1} \overline{a}_{i,j} \|a_i\|^{-1} c_j^{k+1} - \sum_{j=i+1}^{N} \overline{a}_{i,j} \|a_i\|^{-1} c_j^k \\ =& \mathbf{a}_k^T \mathbf{y} - \sum_{j=1}^{i-1} \mathbf{a}_i^T \mathbf{a}_j \|a_i\|^{-1} c_j^{k+1} - \sum_{j=i+1}^{N} \mathbf{a}_i^T \mathbf{a}_j \|a_i\|^{-1} c_j^k \end{aligned} \tag{3.13}$$

By expansion of the seconds sum limits ...

$$c_i^{k+1} = c_i^k + \mathbf{a}_k^T \mathbf{y} - \sum_{j=1}^{i-1} \mathbf{a}_i^T \mathbf{a}_j \|a_i\|^{-1} c_j^{k+1} - \sum_{j=i}^{N} \mathbf{a}_i^T \mathbf{a}_j \|a_i\|^{-1} c_j^k \tag{3.14}$$

and introduction of a new vector $\widetilde{\mathbf{c}}_i^{k+1} = \left[c_1^{k+1}, \cdots, c_{i-1}^{k+1}, c_i^k, \cdots, c_N^k\right]^T$ equation 3.13 furthermore simplifies to[14]:

$$\begin{aligned} c_i^{k+1} =& c_i^k + \mathbf{a}_k^T \mathbf{y} - \mathbf{a}_k^T \left(\mathbf{A_N} \widetilde{\mathbf{c}}_i^{k+1}\right) \\ =& c_i^k + \mathbf{a}_k^T \left(\mathbf{y} - \mathbf{A_N} \widetilde{\mathbf{c}}_i^{k+1}\right) \end{aligned} \tag{3.15}$$

$\mathbf{A_N}$ denotes the normed matrix A, as stated in equation 3.12. As there only changes one element of $\widetilde{\mathbf{c}}_i^{k+1}$ in each iteration step, the last equation can be rewritten as [14]:

$$c_i^{k+1} = c_i^k + \mathbf{a}_k^T \mathbf{e}_i^{k+1} \tag{3.16}$$

where $\mathbf{e_i^{k+1}}$ is the current error vector. It is initialized with the magnitude of the data points $\mathbf{y}$ and computed recursively as:

$$\mathbf{e}_i^{k+1} = \begin{cases} \mathbf{e}_{i-1}^{k+1} - (c_{i-1}^{k+1} - c_{i-1}^k)\mathbf{a}_{i-1}, & i \neq 0 \\ \mathbf{e}_N^k, & i = 0 \end{cases} \tag{3.17}$$

To obtain the coefficients of the polynomial the vector $\mathbf{c}$ need to be devided by the norm of the columns of $\overline{\mathbf{A}}$, just as in equation 3.12.

3.4.1 Complexity Measure

In the following section an estimation of the number of operations necessary for the Gauss-Seidel algorithm is accomplished. The result of the considerations from the last section are summarized in form of pseudo-code algorithm below.

Algorithm 1 Iterative Gauss-Seidel Algorithm on the basis of [14]

1: $\mathbf{c} \leftarrow 0$
2: $\mathbf{e} \leftarrow \mathbf{y}$
3: **for all** iteration i=1..K **do**
4: **for all** coefficient k=1..N+1 **do**
5: $\Delta c \leftarrow \mathbf{a}_k^T \mathbf{e}$
6: $\mathbf{e} \leftarrow \mathbf{e} - \mathbf{a}_k \Delta c$
7: $c_k^{(i)} \leftarrow c_k^{(i-1)} + \Delta c$
8: **end for**
9: **end for**

We assume the columns of the Vandermonde matrix $\mathbf{A}$ to be arranged as stated below.

$$\mathbf{A} = \begin{bmatrix} 1 & n_1 & n_1^2 & n_1^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & n_M & n_M^2 & n_M^3 \end{bmatrix}$$

Because of the symmetric alignment of the sampling-time vector $\mathbf{n}$ around zero, as defined in 3.3, the number of computations for certain parts of the algorithm can be divided in half and even less. The division of each column by its norm, as stated in equation 3.12 does not affect this symmetry and is therefore disregarded in the following considerations.
In the first step we focus on line 5 of the algorithm above. The vector operation $\mathbf{a}_k^T \mathbf{e}$, with both vectors $\in \mathbb{R}^{1\times M}$, needs M additions and (M-1) multiplications in the conventional way of computation. We can now utilize the symmetric form of the sampling-time vector and therefore have to regard two cases:
If the column index k is **even**, the upper and lower half of the column vector $\mathbf{a_k}$ have opposite sign.

$$\begin{aligned} \mathbf{a}_2 &= \left[-\lfloor \tfrac{M}{2} \rfloor, -\lfloor \tfrac{M}{2} - 1 \rfloor, \ldots, 0, \ldots, \lfloor \tfrac{M}{2} - 1 \rfloor, \lfloor \tfrac{M}{2} \rfloor\right]^T \\ \mathbf{a}_4 &= \left[-\lfloor \tfrac{M}{2} \rfloor^3, -\lfloor \tfrac{M}{2} - 1 \rfloor^3, \ldots, 0, \ldots, \lfloor \tfrac{M}{2} - 1 \rfloor^3, \lfloor \tfrac{M}{2} \rfloor^3\right]^T \end{aligned}$$

The equation $\mathbf{a}_k^T \mathbf{e}$ can therefore be rewritten as:

$$\mathbf{a}_k^T \mathbf{e} = a_1 \cdot (e_1 - e_M) + a_2 \cdot (e_2 - e_{M-1}) + \ldots + a_{\lfloor \frac{M}{2} \rfloor} \cdot (e_{\lfloor \frac{M}{2} \rfloor} - e_{M-\lfloor \frac{M}{2} \rfloor+1}) \tag{3.18}$$

The number of operations has been cut down to $\lfloor \frac{M}{2} \rfloor$ multiplications and $M-2$ additions. If the column index k is **odd**, the sign of the upper and lower half of the column vector is the same. For this case one has to change the subtraction in the braces in equation 3.18 to an addition. The number of operations stays the same.

For the fist column vector $\mathbf{a}_1$ the computation simplifies even further. Since all element in $\mathbf{a}_1$ have the same value the computation can be rewritten as the sum of the elements of the error vector $\mathbf{e}$ multiplied by a constant.

$$\mathbf{a}_1^T\mathbf{e} = a_{1,1} \cdot \sum_{i=0}^{M} e_i \tag{3.19}$$

We therefore get 1 multiplication and $M-1$ additions.

In a similar way the number of computations for line 6 of algorithm 1 can be reduced. As the upper and lower half of the columns k=2,3 and 4 are symmetric it suffice to compute the product for the upper half of the vector and apply the result, with respect to the sign, to the lower half. The number of computations for line 6 therefore adds up to $\lfloor \frac{M}{2} \rfloor$ multiplications for the product $\mathbf{a}_k\Delta c$ and M additions. For the first columns $\mathbf{a}_1$ we only have one multiplication and M additions.

$$\mathbf{a}_k\Delta c = \begin{bmatrix} a_1 \cdot \Delta c \\ a_2 \cdot \Delta c \\ \vdots \\ a_{\lfloor \frac{M}{2} \rfloor} \cdot \Delta c \\ 0 \\ \pm a_{\lfloor \frac{M}{2} \rfloor} \cdot \Delta c \\ \vdots \\ \pm a_2 \cdot \Delta c \\ \pm a_1 \cdot \Delta c \end{bmatrix}$$

The overall number of computations for Algorithm 1 sums up as follows:

$$\begin{aligned} \#\text{multiplications} &= K \cdot \left(\underbrace{1}_{\mathbf{a}_1^T\mathbf{e}} + N \cdot \underbrace{\lfloor \frac{M}{2} \rfloor}_{\mathbf{a}_k^T\mathbf{e}|_{k=2,3,4}} + \underbrace{1}_{\mathbf{a}_1\Delta c} + N \cdot \underbrace{\lfloor \frac{M}{2} \rfloor}_{\mathbf{a}_k\Delta c|_{k=2,3,4}} \right) \\ &= 2 \cdot K \cdot \left(1 + N \cdot \lfloor \tfrac{M}{2} \rfloor \right) \end{aligned} \tag{3.20}$$

$$\#\text{additions} = K \cdot \left(\underbrace{M-1}_{\mathbf{a}_1^T\mathbf{e}} + N \cdot \underbrace{(M-2)}_{\mathbf{a}_k^T\mathbf{e}|_{k=2,3,4}} + (N+1) \cdot \underbrace{M}_{\mathbf{a}_k\Delta c} + \underbrace{N+1}_{c_k^{(i-1)}+\Delta c} \right) \tag{3.21}$$

K denotes the number of iterations, this parameter will be determine later in Chapter 4.5.

3.4.2 Rate of convergence

Simulations showed that the order of the columns in the Vandermonde matrix $\mathbf{A}$ impact the rate of convergence of Gauss-Seidel algorithm.

$$\mathbf{A} = \begin{bmatrix} 1 & -\lfloor \frac{M}{2} \rfloor & \lfloor \frac{M}{2} \rfloor^2 \\ 1 & -(\lfloor \frac{M}{2} \rfloor - 1) & (\lfloor \frac{M}{2} \rfloor - 1)^2 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & (\lfloor \frac{M}{2} \rfloor - 1) & (\lfloor \frac{M}{2} \rfloor - 1)^2 \\ 1 & \lfloor \frac{M}{2} \rfloor & \lfloor \frac{M}{2} \rfloor^2 \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{n} & \mathbf{n} \circ \mathbf{n} \end{bmatrix} \tag{3.22}$$

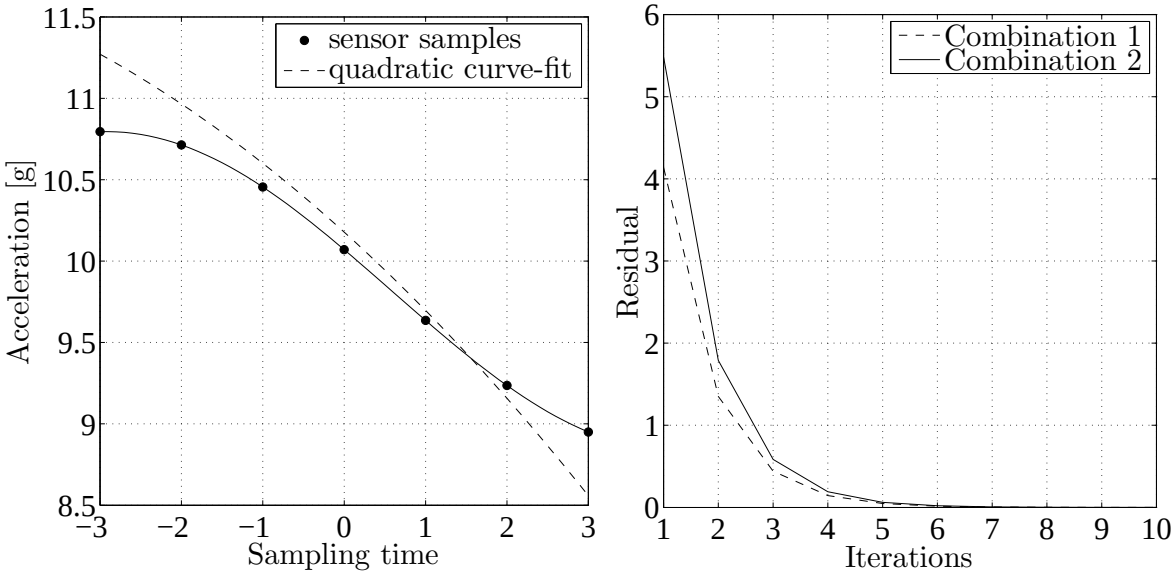
Recapitulating equation 3.3 in combination with the improvement of rearranging the sampling-time vector leads to the general form of $\mathbf{A}$ for a number of samples M as shown in equation 3.22. The term $\mathbf{n} \circ \mathbf{n}$ in equation 3.22 denote the Hadamard product of the vector $\mathbf{n}$, which simply states an entry-wise multiplication [10]. As the columns in $\mathbf{A}$ are orthogonal only two possible combinations for the alignment of the columns arise.

$$\text{Combination 1: } \mathbf{A} = \begin{bmatrix} 1 \\ \mathbf{n} \\ \mathbf{n} \circ \mathbf{n} \end{bmatrix}^T \text{ or } \mathbf{A} = \begin{bmatrix} 1 \\ \mathbf{n} \circ \mathbf{n} \\ \mathbf{n} \end{bmatrix}^T$$

$$\text{Combination 2: } \mathbf{A} = \begin{bmatrix} \mathbf{n} \circ \mathbf{n} \\ \mathbf{n} \\ 1 \end{bmatrix}^T \text{ or } \mathbf{A} = \begin{bmatrix} \mathbf{n} \circ \mathbf{n} \\ 1 \\ \mathbf{n} \end{bmatrix}^T$$

Which of these combinations result in a faster convergence of Gauss-Seidel depends on the shape of the curve which the algorithm aims to fit. In the following the convergence behaviour of both combinations will be studied with the help of two examples.

Figure 3.3a shows the distribution of the samples on the observation interval, denoted as black dots. The observation interval in this example encloses no extremal point. The result of Gauss-Seidel algorithm after 10 iterations, which is represented by the dashed line, is almost a straight line.



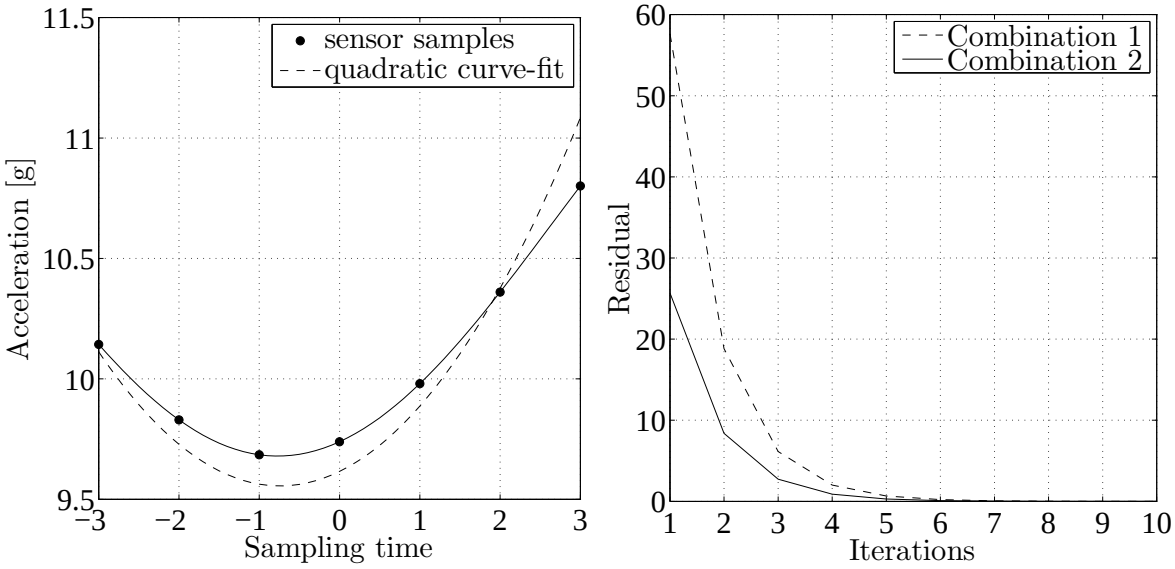
(a) 2^{nd} degree curve fit.

(b) Rate of convergence.

Figure 3.3: Curve fit and rate of convergence for first example.

Taking a look at the polynomial coefficients $\mathbf{c} = \begin{bmatrix} -0.2936 & -4.5212 & 1.7883 \end{bmatrix}^T$ of this curve fit, one can see that the quadratic term is rather small which leads to a slow decrease of the function from the vertex. Figure 3.3b compares the residual - in this case the difference

between the standard deviation of the phase error for the iterative solutions and the solution of the pseudoinverse - in each iteration step for both combinations. The dashed line in Figure 3.3b corresponds to the first combination with the signal with lowest energy in the first column, while the solid line relates to the second combination respectively the signal with the highest energy in the first column. For this example the first combination shows an advantage in the convergence speed over the second one, whereat attention should be paid that the curve on the observation interval does not provide an information about the extremal point and therefore is not usable for our application



(a) 2^{nd} degree curve fit.

(b) Rate of convergence.

Figure 3.4: Curve fit and rate of convergence for second example.

In the second example, depicted in Figure 3.4, the curve fit on the left - after 10 iterations - estimates the sensor signal with sufficient precision. Taking a look on the polynomial coefficient for this curve fit $\mathbf{c} = \begin{bmatrix} 1.0958 & 1.6206 & -3.8710 \end{bmatrix}^T$ one can see the difference in the magnitude of the quadratic term compared to the first example. The difference in the rate of convergence shows also a significant difference. In this example the second combination with the signal with the highest energy in the first column is the faster one, although the overall residual for both combinations in the first iterations is many times higher then the residual in the first example.

As the informations about the extremal points in the first example are too imprecise and therefore not suitable for the tire localisation algorithm, the rate of convergence for this case is irrelevant. For the implementation of the polynomial APS algorithm we will use the second combination.

4 Parameter Evaluation

In the previous chapters the properties of the sensor signal and methods to perform a curve fit were proposed. The following chapter deals with the parameter evaluation for the *polynomial APS* algorithm, which incorporates the curve-fit algorithm and the computation of the time of the extremal points. For lack of knowledge of these design parameters further investigation of the performance of the polynomial APS under certain conditions are required. The following chapter summarizes these parameters and explains the methods for their evaluation.

4.1 Degrees of freedom

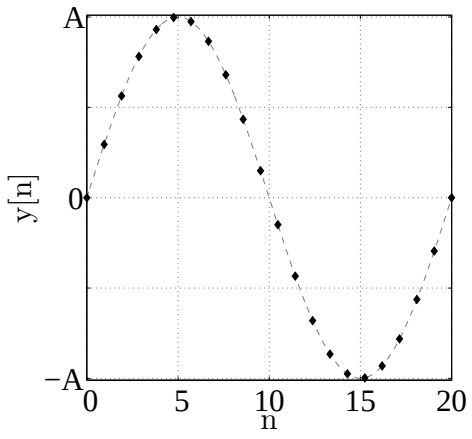
As mentioned in Chapter 3 polynomials of higher degree should be avoided by reason of stability as well as computational effort. Therefore in the following only quadratic and cubic polynomials will be considered $N \in [2, 3]$. For each polynomial degree a set of parameters needs to be defined :

- observation interval T_{OBS}
- number of samples M
- number of iterations for Gauss-Seidel k algorithm

4.2 Simulation Framework

As the most of these parameters interdepend on each other, modification and their effects need to be assessed by computer simulations. The simulation framework computes the sensor signal, based on the considerations from Chapter 2, for certain scenarios.

$$a(t) = v_{car}(t)^2 \cdot \frac{\varnothing_{Tire}}{2} + g \cdot \sin(2 \cdot \int \frac{v(t)}{\varnothing_{Tire}} \mathrm{d}t + \varphi_0) + a_{car}(t) \cdot \cos(2 \cdot \int \frac{v(t)}{\varnothing_{Tire}} \mathrm{d}t + \varphi_0) + e(t)$$



As stated in Chapter 3 this sensor model can, for a certain interval, be described by a polynomial:

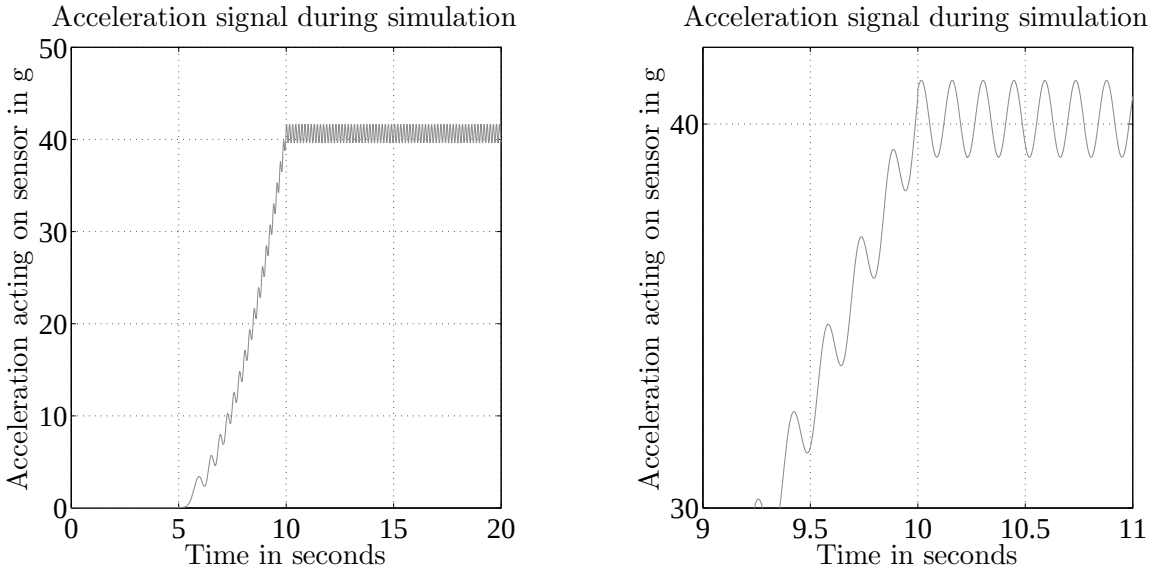
$$y[n] = c_0 + c_1 \cdot n + c_2 \cdot n^2 + e[n]$$

The term $e[n]$ is representative for sensor defects like sensitivity error and offset error. The sensitivity of a sensor defines the *minimum input of physical parameter that will create a detectable output change* [6]. The sensitivity error describes the deviation from the ideal slope of the sensors characteristic curve. The offset error describes the difference between the measured

magnitude of the sensor signal compared to its specified value. As a consequence of these sensor defectives the sensor signal must exceed a certain threshold before any computations can be executed, in order to ensure that the car is moving. In the further simulations we will either consider an ideal sensor with no defects $e(t) = 0$ or use well-known experienced data. Additionally the sensor signal can be superimposed by normally distributed noise signal. The signal to noise ration adds up as follows.

$$\mathrm{SNR} = 10 \log_{10} \frac{A^2}{\mathrm{Var(e[n])}} \tag{4.1}$$

The term A denotes the magnitude of the sensor signal and can, based on the considerations from Chapter 2, be assumed as 1g.
Figure 4.1 depicts the sensor signal for one trial of the simulation framework. A complete driving cycle is simulated, which includes the acceleration of the car from a standstill position for a time slot of 5 seconds followed by a phase of 10 seconds where the car either drives with constant speed or a variable acceleration between $\pm 2 \frac{m}{s^2}$.



(a) Sensor signal for one driving cycle.

(b) Detailed view.

Figure 4.1: Sensor signal computed by simulation framework.

The curve fitting algorithm is applied in the last phase, in the time slot between 10 and 20 seconds.
Figure 4.2 describes the individual steps of the APS algorithm. In the following these steps will be describes in in short.

Coarse frequency estimation: On the basis of the DC component of the acceleration sensor signal the frequency of the wheel can be determined.

$$a = \omega^2 \cdot r = (2\pi)^2 f^2 \cdot r$$

$$f_{est} = \sqrt{\frac{a}{4\pi^2 r}} \qquad \text{hence} \qquad T = \frac{1}{f_{est}}$$

On the basis of the frequency estimate f_{est} respectively the rotation period T the sampling period is determined.

Determine sampling period: The sampling period defines the distance in time between the samples used for the polynomial APS algorithm. In order to determine this value the number of samples M on the observation interval T_{OBS} need to be known. If, for example, we assume an observation interval of a half period of the sinusoidal sensor signal $T_{OBS} = \frac{T}{2}$ and M = 11 as number of samples the sampling period adds up as follows:

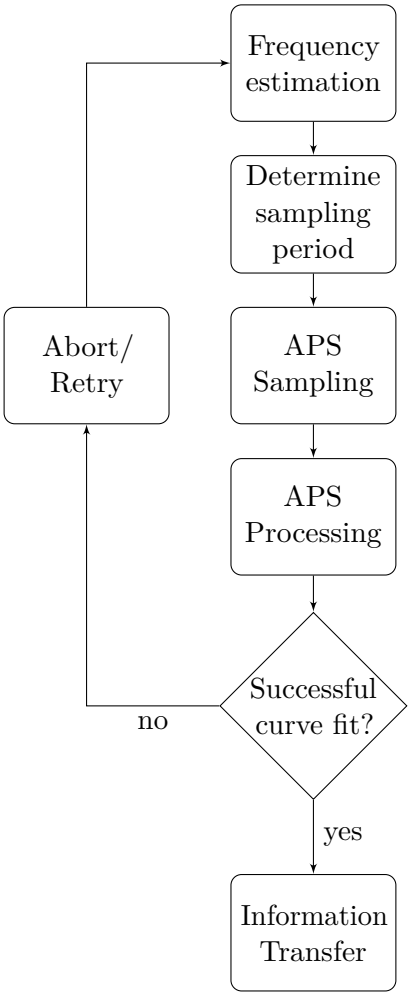


Figure 4.2: Flow Chart of the APS Algorithm.

Based on the number of samples M and the ratio between the rotation period T and the observation interval T_{OBS} we can determine the number of samples per rotation period M_0.

$$M_0 = \frac{T}{T_{OBS}} \cdot M$$

Therefore the sampling period can be defined as the rotation period T divided by the number of samples per period M_0.

$$T_s = \frac{T}{M_0} = \frac{1}{M_0 \cdot f_{est}}$$

APS Sampling: During sampling M acceleration sample points are acquired, one per sampling period. To reduce noise and therefore improve the performance of the polynomial APS, an oversampling technique is used. Each sample is assemble from the sum of a certain number of consecutive measurement values. The number of values is denoted as OSF and defined as OSF = 8 samples for this application. Thus, the SNR improves by $10\log_{10}(OSF) = 9.0309$ dB. The oversampling is not uniform over time. Every T_s the TPMS wheel unit acquires 8 measurement values in rapid succession.

APS Processing: In this step the polynomial curve fit on the basis of the Gauss-Seidel algorithm from Chapter 3.4. The result is a polynomial of N-th degree as well as the time value of an extremal point.

Successful curve fit: Whether the time value of the extremal point lies within the limits of the observation interval the curve fit was successful otherwise the curve fit is considered as unsuccessful and the algorithm needs to start again.

Information transfer: For a certain range the sensor signal can be assumed as periodic. The frequency of the wheel won't change drastically between two or more consecutive periods. Therefore the information transfer takes place at the computed next occurrence of the extremal point.

4.3 Observation interval

The observation interval defines a range on a period of the sensor signal which is used by the polynomial APS algorithm as input data. Its position, respectively the phase angle of the signal $\varphi(v, t)$ when the observation interval begins, is always arbitrary since the point in time when the algorithm starts to record the sensor samples is chosen randomly. A curve fit is regarded as successful when the computed time of the polynomials extremal point lies within the limits of the observation interval.
As shown in Figure 4.3 the observation interval, on which the curve fit is applied, affect the precision of the result. If for example a quadratic curve fit would be applied over a complete period of the sensor signal, the result would be a straight line. On the other hand an observation interval covering only a small parts of the sensor signal, e.g. less then a quarter of a period, would result in a good fit when applied on an area around an extremal point but may fail more often, since the observation interval would more often fall into a region with no extremal point. The fail rate defines the total number of unsuccessful curve fits in percent of a specified amount of trials of the observation interval.

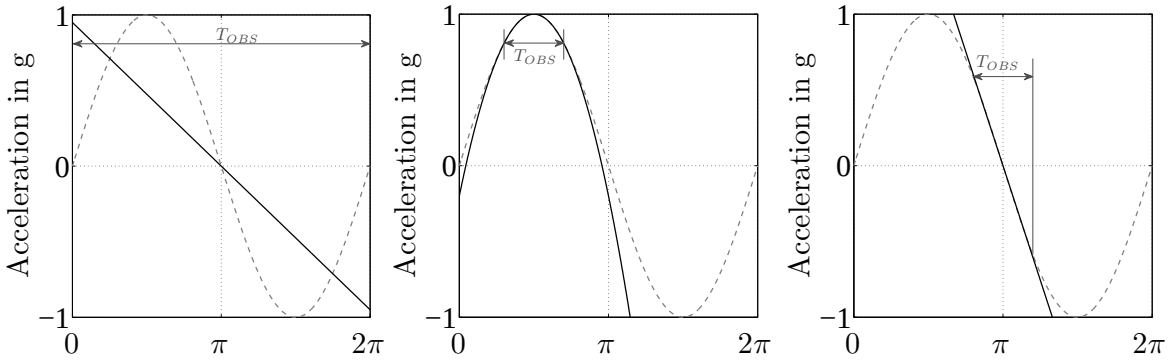


Figure 4.3: Impact of observation interval on curve fit.

The TPMS wheel unit has to estimate the rotation speed of wheel, which is equivalent to the frequency of the sinusoidal sensor signal, in order to determine the observation interval. Deviation of the measured frequency from the true frequency affect the further computations. This variations need to be recognised in the definition of the observation interval.

Frequency error: As stated in the previous section, the frequency of the wheel can be estimated by the measured acceleration signal a as follows:

$$a = \omega^2 \cdot r = (2\pi)^2 f^2 \cdot r$$

$$f_{est} = \sqrt{\frac{a}{4\pi^2 r}} \tag{4.2}$$

As the measured sensor signal is expected to be erroneous due to offset and sensitivity errors of the sensor, a deviation from the true tire frequency is expected. In consideration of this deviation in the parameter evaluation we assume the true sensor signal to be afflicted with an offset error of $\pm 6g$ and a sensitivity error of ± 10 percent.

$$\overline{a} = a_{true} \pm 6g \pm \frac{a_{true}}{10} \tag{4.3}$$

Because of these inaccuracy a threshold of 10g was introduced to determine a movement of the car. Computations will only be executed when the measured sensor signal exceeds this threshold. Considering this we can now define the range within the estimated observation interval can shift. As stated earlier, the sampling period is defined as $T_S = \frac{T}{M_0}$ at which M_0 denotes the number of samples per rotation period. The observation interval can be derived from this relation.

$$T_{OBS} = M \cdot T_S = M \cdot \frac{T}{M_0} = \frac{M}{f_{est} \cdot M_0}$$

By applying the frequency estimate from equation 4.2:

$$T_{OBS} = \frac{M}{M_0} \cdot \sqrt{4\pi^2 r} \cdot \frac{1}{\sqrt{a}}$$

To determine the region within the estimated frequency f_{est} respectively the observation interval T_{OBS} can shift, two cases have to be considered:
For the first, we assume a true acceleration $a_{true} = 4g$, a desired observation interval $T_{OBS} = \frac{T}{2}$ and a measured sensor signal $\overline{a} = 10.4g$

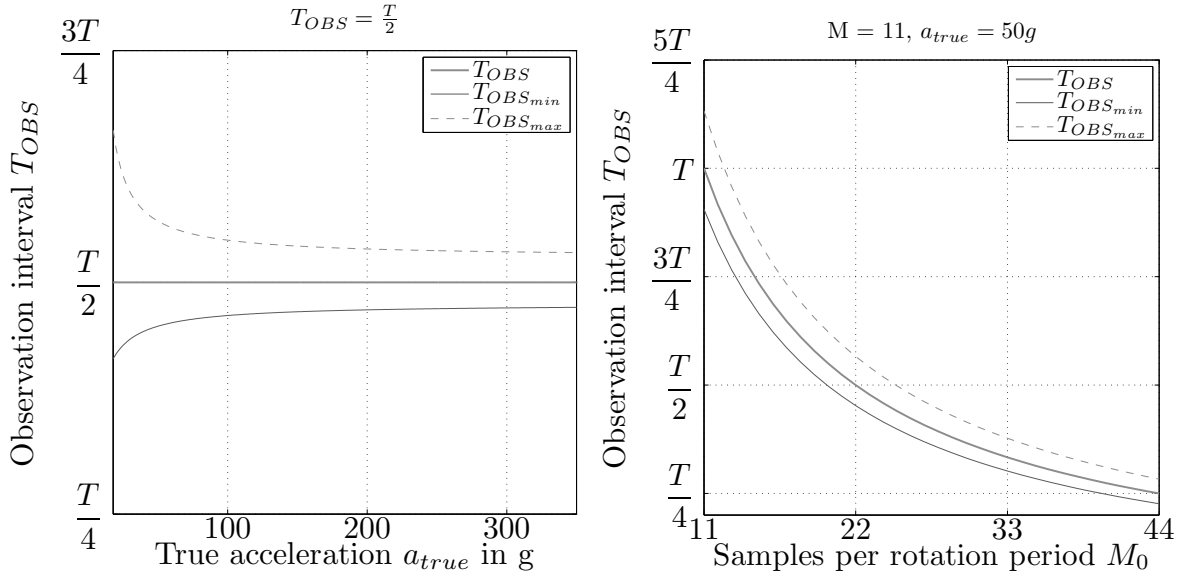
$$\begin{aligned} T_{OBS} &= \underbrace{\frac{M}{M_0} \cdot \sqrt{4\pi^2 r}}_{const} \cdot \frac{1}{\sqrt{a}} \hat{=} \frac{T}{2} \\ T_{OBS_{min}} &= \overbrace{\frac{M}{M_0} \cdot \sqrt{4\pi^2 r}} \cdot \frac{1}{\sqrt{\overline{a}}} \hat{=} \frac{T}{3.224} \approx \frac{T}{3} \end{aligned} \tag{4.4}$$

This gives us an upper limit for the deviation of the observation interval. To determine the lower limit we assume a measured acceleration of $\overline{a} = 10g$ and a true acceleration of $a_{true} = 17g$ with the same desired observation interval from above.

$$\begin{aligned} T_{OBS} &= \underbrace{\frac{M}{M_0} \cdot \sqrt{4\pi^2 r}}_{const} \cdot \frac{1}{\sqrt{a}} \hat{=} \frac{T}{2} \\ T_{OBS_{max}} &= \overbrace{\frac{M}{M_0} \cdot \sqrt{4\pi^2 r}} \cdot \frac{1}{\sqrt{\overline{a}}} \hat{=} \frac{T}{0.6393} \approx \frac{2T}{3} \end{aligned} \tag{4.5}$$

This means that through uncertainness of the frequency estimation an observation interval defined at $\frac{T}{2}$ can be somewhere between $\frac{T}{3}$ and $\frac{2T}{3}$.
Figure 4.4 depicts the maximum deviation of the estimated observation interval $T_{OBS_{max}}$ and

$T_{OBS_{min}}$ from the true observation interval T_{OBS}. In Figure 4.4a the impact of the sensor defects are shown. The true acceleration is varied from 18g to 350g. $T_{OBS_{max}}$ and $T_{OBS_{min}}$ are determined based on the erroneous sensor signal as described in Equation 4.3. Apparently, for lower acceleration values the offset error mostly characterizes the deviation from T_{OBS}. With increasing acceleration the impact of the offset error becomes minor while the sensitivity error becomes dominant.
Figure 4.4b depicts the impact of a misestimation of the number of samples per rotation period M_0 for a define acceleration of $a_{true} = 50g$ and a number of samples M = 11.



(a) Deviation of T_{OBS} as function of the acceleration.

(b) Deviation of T_{OBS} as function of M_0.

Figure 4.4: Impact of number of samples on second order polynomial APS.

To determine the optimal observation interval regarding phase error and fail rate, quadratic and cubic curve-fits were simulated on a set of parameters. In the simulation the range of the observation interval got altered after one thousand trials and the standard deviation (STD) for each observation interval was computed. Starting with an observation interval covering a complete period of the sensor signal it was successive reduced down to a quarter period. The number of samples were distributed equally spaced over the observation interval and stayed constant during the simulation. This simulation scenario was repeated for all odd number of samples M, from 7 up to 17 samples.
Figure 4.5 depicts the progression of the phase error over different observation intervals for quadratic and cubic curve-fits with M=11 samples. The sensor signal in this simulation was superimposed by a noise with an SNR of 10dB. The gray area labelled with $f_{estimate}$ marks the possible range of deviation from an observation interval $T_{OBS} = \frac{T}{2}$ due to errors in the frequency estimate, derived in equation 4.4 and 4.5.

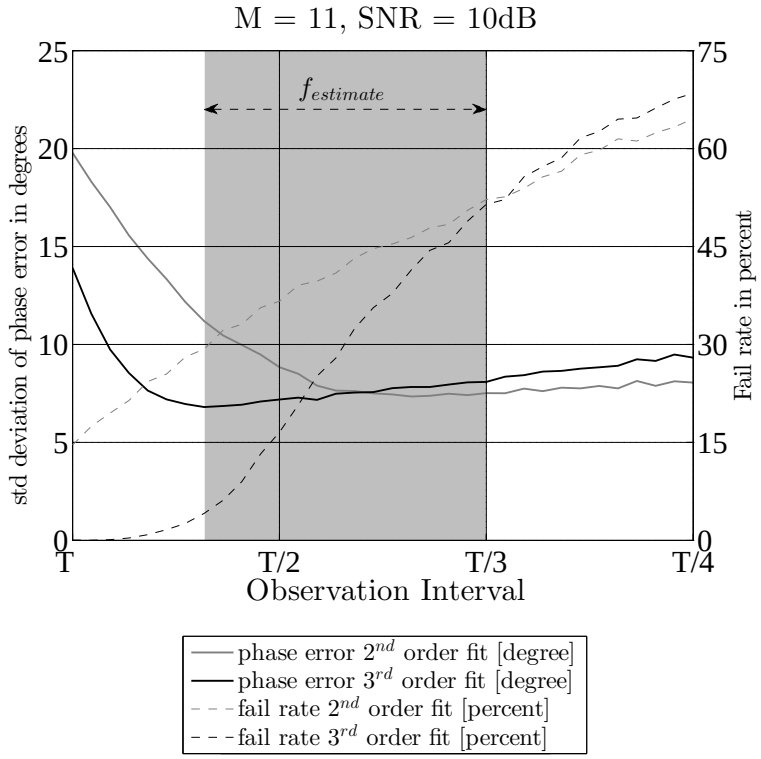


Figure 4.5: Impact of observation interval on curve fit.

As one can see in the Figure, the STD of the phase error decreases for both polynomial fits by decrease of the observation interval until a certain point is reach. For the second order curve fit this point is at $T_{OBS} \approx \frac{5T}{12}$ and for third order at $T_{OBS} \approx \frac{3T}{4}$. From this point the STD of the phase error remain static for the quadratic curve fit. In contrast the STD of the fail rate increases continuously. On the contrary the performance of the cubic curve fit slightly worsen by further shortening of the observation interval. As already stated the fail rate increases by decrease of the observation intervals length, since the probability increases that the observation interval covers an area of the sensor signal which involves no extremal point. In the simulation depicted in Figure 4.5 the phase error slightly increases from $T_{OBS} \approx \frac{5T}{12}$ for the second order curve fit respectively $T_{OBS} \approx \frac{3T}{4}$ for the third order curve fit, which can be attributed to the superimposed noise signal. In a noise free simulation scenario the phase error would continuously decline by decrease of the length of the observation interval. This gain in precision goes hand in hand with and increase of the fail rate.

Due to the uncertainty in the frequency estimate the observation interval for a quadratic curve fit is defined at $\frac{T}{2}$ and for cubic curve fit at about $\frac{3T}{4}$ to obtain reasonable results with acceptably fail rate.

Table 4.1 summarizes the STD of phase error and fail rate for different numbers of samples at the specified observation interval ($T_{OBS} = \frac{T}{2}$ respectively $\frac{3T}{4}$).

(a) cubic curve fit

M	fail rate percent	STD of phase error degree
7	2.59	8.20
9	0.89	8.18
11	0.62	7.74
13	0.47	8.03
15	0.36	7.67
17	0.13	8.03

(b) quadratic curve fit

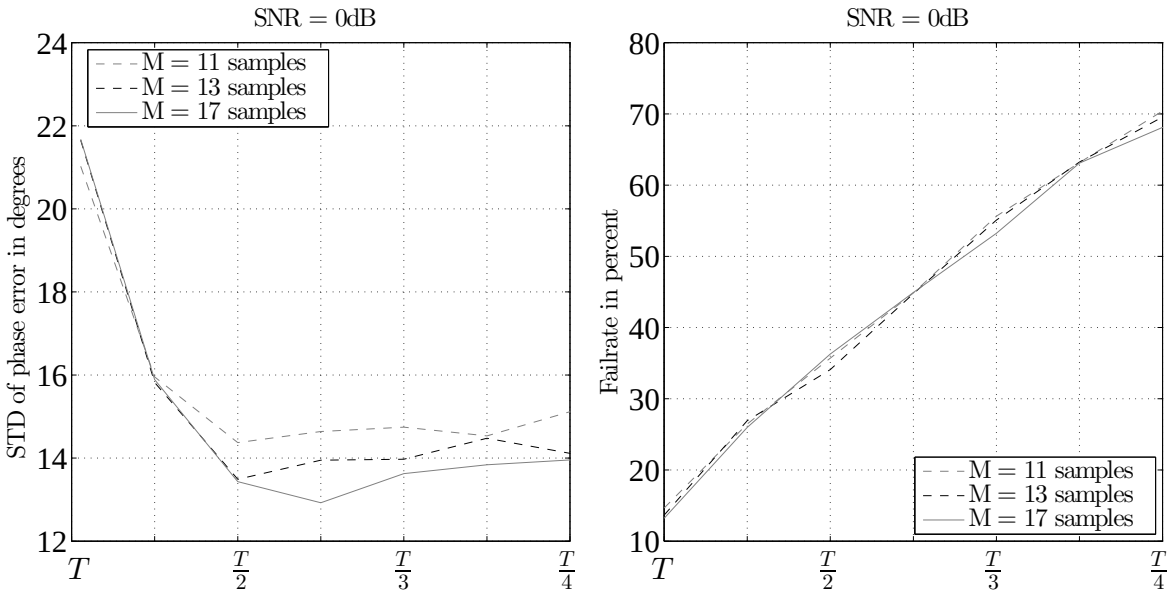
M	fail rate percent	STD of phase error degree
7	38.06	8.52
9	37.53	8.72
11	36.84	8.73
13	36.39	8.81
15	35.50	9.03
17	35.46	8.73

Table 4.1: STD of phase error and fail rate for different number of samples M.

4.4 Number of samples M

Due to the optimisation in reduction of computations in Chapter 3.3 an odd number of samples became a constraint. For reasons of computational effort and runtime constraints the number of observed samples shall be minimized. In order to determine the impact of the number of samples, the STD of the phase error was computed for several parameters. On the one hand the number of samples on the observation interval was varied from M = 9 to M = 17 samples and on the other hand the length of the observation interval was altered, beginning with the observation of a complete period of the sensor signal down to an eighth period. The simulations were performed with signals with imperfections. Otherwise there would have been only slight deviations between the solutions without providing any information. Therefore the simulated sensor signal was superimpose by a noise signal with zero mean and an SNR of 0dB. Furthermore quantization errors were added.

Figure 4.6 shows the trend of the STD of the phase error as function of the observation interval for a quadratic curve fit as well as the increase of the fail rate by reduction of the length of the observation interval. For clarity reasons only three different parameter sets (M=11, 13 and 17 samples) were plotted. The results for the remaining simulations at an observation interval of $\frac{T}{2}$ can be found in Table 4.4. Each simulation result in the graph as well as the table is computed from one thousand trials of the simulation scenario for the given parameter set. Compared to the simulations from the previous section, the fail rates for the different numbers of samples M in case of the quadratic curve fit in Figure 4.6 only slightly differ from those depicted in the simulation results in Figure 4.5. What becomes apparent is that STD of the phase error almost doubled due to the increased noise.



(a) Phase error over observation interval

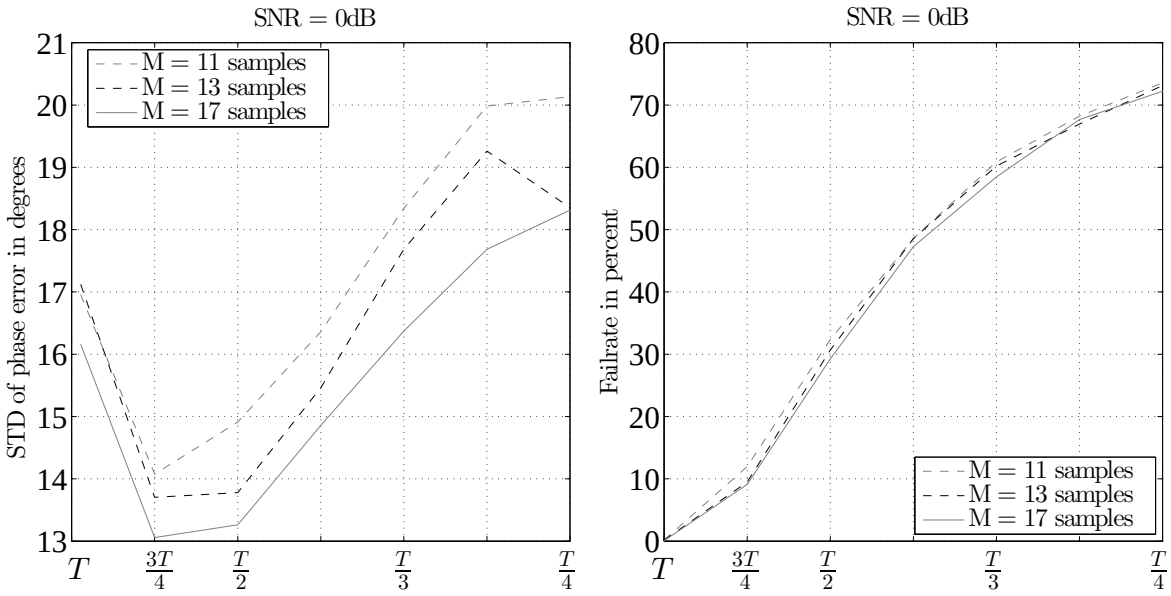
(b) Fail rate over observation interval

Figure 4.6: Impact of number of samples on second order polynomial APS.

M	STD of phase error degrees	Fail rate percent
7	15,99	38,70
9	15,08	35,74
11	14,37	35,70
13	13,49	34,10
15	13,43	36,28
17	13,04	35,10

Table 4.2: Simulation results for second order polynomial APS at $T_{OBS} = \frac{T}{2}$

For the cubic curve-fit depicted in Figure 4.7 the progression of the STD of phase error and fail rate are quite similar as for the quadratic curve-fit. For both simulations it becomes apparent that for consecutive observed number of samples M, e.g. M=7 and M=9, the difference in the standard deviation is at most about 1 degree. This difference decreases the higher the number of samples is. For observation intervals less than $T_{OBS} = \frac{3T}{4}$, respectively $T_{OBS} = \frac{T}{2}$ for the quadratic curve-fit, the number of samples show more impact on the phase error. By contrast the fail rate hardly changes for different M.



(a) Phase error over observation interval

(b) Fail rate over observation interval

Figure 4.7: Impact of number of samples for third order curve fit.

As apparent from these results the difference in the phase error and the fail rate for different M are rather small and would not justify an increase of the number of observed samples. Based on the consideration on the number of computations from Section 4.5 an increase from M=11 to M=17 samples for a quadratic curve-fit would result in additional computational expenses of $K \cdot 35$ more additions and, which is more time-consuming, $K \cdot 6$ more multiplications. K denotes the number of iterations which are determined in the next section. Therefore it was decided to choose M=11 as parameter for the polynomial APS.

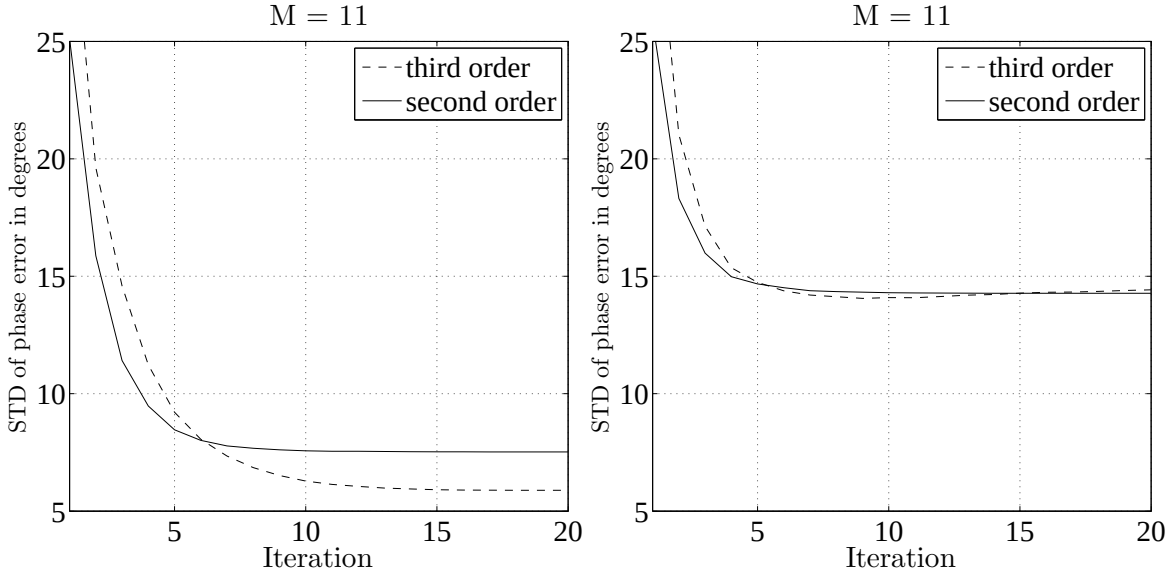
M	STD of phase error degrees	Fail rate percent
7	16,91	17,10
9	15,12	14,12
11	14,07	11,98
13	13,70	9,50
15	13,05	9,12
17	12,59	7,48

Table 4.3: Simulation results cubic curve-fit at $T_{OBS} = \frac{3T}{4}$

4.5 Number of iterations k

Beside the parameters discussed until now, the degree of accuracy of the polynomial APS mainly depends on the number of iterations of the Gauss-Seidel algorithm. In the following section the required number of iterations to reach a reasonable precision are determined. Therefore the result of the iterative Gauss-Seidel algorithm for several numbers of iterations is compared with the solution of the pseudo-inverse from equation 3.4. To ensure equal conditions the same data set from several thousand trials of the simulation framework is used for both solutions. The exact time values and therefore the phase angles of the extremal points on the observation interval are known from the simulation data. In the following the number of iterations of the Gauss-Seidel algorithm is successively increased. For each iteration step the STD of the phase error for both solutions is computed.

Figure 4.8 depicts the STD of these phase errors for quadratic and cubic curve fits as function of the iterations count. The Figure on the left depict the result for a noise-free sensor signal, while in the Figure on the right the sensor signal was superimposed by 0dB SNR. In the noise free scenario the steady state for the quadratic curve fit arise after about k = 5 iterations, after k = 10 iterations the cubic curve fit can be assumed to be in a steady state. The difference in the performance of both polynomial fits is about 2.2 degrees. For the simulation with disturbance both polynomial fits gain almost the same performance. An increase of the number of iterations beyond 5 iterations hardly affect the performance. Simulation have shown that a further increase of the noise level cause an offset toward higher standard deviation of the phase error. However the progression of the curve over the iteration count remains the same. Just as in the evaluation of the other parameters, it is required to trade of the gain in reduction of phase error due to increase of iterations against the additional computational effort.

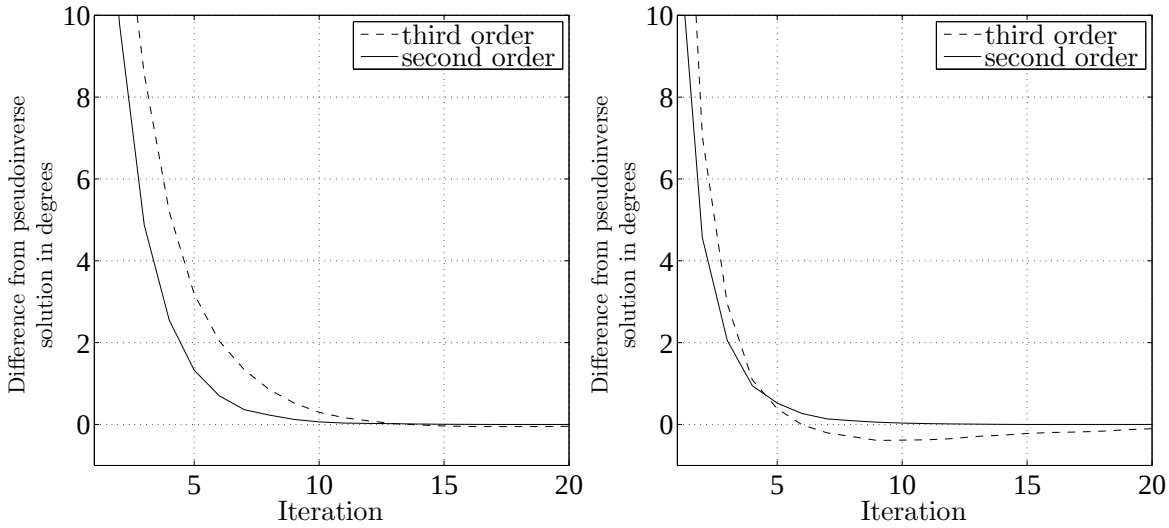


(a) No noise

(b) Superimposed by 0dB SNR

Figure 4.8: STD of phase error over iterations.

The decay of the phase error becomes flat-angled the higher the number of iterations are, so that the decrease in phase error becomes disproportional compared to the additional computational effort. For the quadratic curve fit the slope of the curve continuously decreases. To the point of 5 iterations the slope took of from initially 10 degrees per iteration down to 1 degree. As number of iterations we choose the point when booth solutions are in its steady state. This means k=5 for the quadratic and k=10 for the cubic curve fit. In the performance evaluation in Section 5.3.2 we will observe the impact of the number of iterations applied on data from real measurements.



(a) No noise

(b) Superimposed by 0dB SNR

Figure 4.9: Difference between iterative curve-fit and the solution of the pseudo-inverse.

Figure 4.9 depicts STD of the difference between the performance of the iterative curve-fits, as function of the number of iterations, and the solution of the pseudo-inverse. Again, this simulations were performed once in a noise-free scenario and once superimposed with 0dB SNR.

Recapitulating the computational effort for the coefficient estimation of the polynomial APS, derived in Section 3.4.1, the number of computations sums up as follows:

2nd order polynomial APS:

$$\begin{aligned}\#\text{multiplications} &= 2 \cdot K \cdot \left(1 + N \cdot \lfloor \frac{M}{2} \rfloor\right) \\ &= 2 \cdot 5 \cdot \left(1 + 2 \cdot \lfloor \frac{11}{2} \rfloor\right) \\ &= 110\end{aligned}$$

$$\begin{aligned}\#\text{additions} &= K \cdot (M - 1 + \ N \cdot (M - 2) + \ (N + 1) \cdot M + N + 1) \\ &= 5 \cdot (10 + 2 \cdot 9 + 3 \cdot 11 + 3) \\ &= 320\end{aligned}$$

3rd order polynomial APS:

$$\begin{aligned}\#\text{multiplications} &= 2 \cdot K \cdot \left(1 + N \cdot \lfloor \frac{M}{2} \rfloor\right) \\ &= 2 \cdot 10 \cdot \left(1 + 3 \cdot \lfloor \frac{11}{2} \rfloor\right) \\ &= 320\end{aligned}$$

$$\begin{aligned}\#\text{additions} &= K \cdot (M - 1 + \ N \cdot (M - 2) + \ (N + 1) \cdot M + N + 1) \\ &= 10 \cdot (10 + 3 \cdot 9 + 4 \cdot 11 + 4) \\ &= 850\end{aligned}$$

4.6 Final Parameter set

In this Chapter we have determined all the parameters required for the polynomial APS. In summary we have defined the observation interval and its impact on the success of a curve fit. Sources of error like the frequency estimate were defined and also integrated in the decision making. The other two parameters, the number of samples M and the number of iterations k, were both an agreement between performance and computational effort. Naturally an increase of both would gain an increase in the performance. On the other hand this is only possible to a certain extent, beyond that the additional computational effort becomes disproportional to the realised result. In principle the polynomial APS is required to work with respect to its hardware resources and therefore the parameters have always been chosen as trade-off between effort and gain. In the next chapter the performance of the polynomial APS with will be analysed under different conditions.

Second order polynomial APS	
Number of samples M	11
Number of samples per rotation M_0	22
Number of iterations k	5
Observation interval T_{OBS}	$\frac{T}{2}$
Third order polynomial APS	
Number of samples M	11
Number of samples per rotation M_0	15
Number of iterations k	10
Observation interval T_{OBS}	$\frac{3T}{4}$

Table 4.4: Parameter set for the polynomial APS.

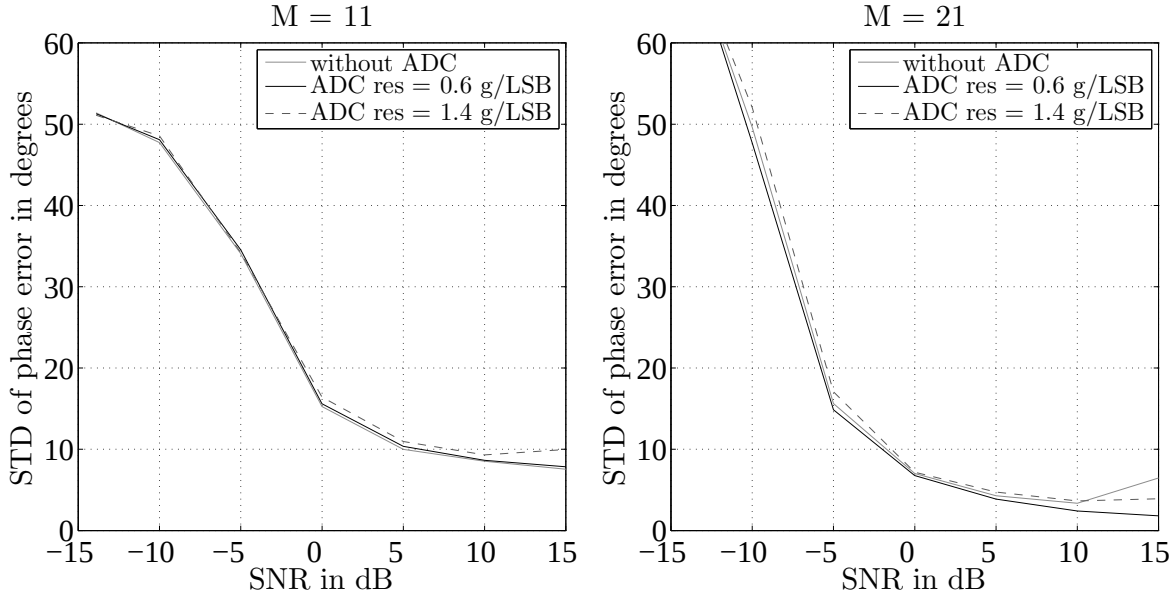
5 Simulation Results

In the following Chapter the performance of the polynomial APS algorithm is compared to the performance of a sinusoidal APS algorithm. The simulations are executed with imperfections of the sensor signal and the parameter set specified in the previous chapter.

Sinusoidal APS: The sinusoidal APS algorithm is based on a parameter estimation of a sinusoidal signal model describing the sensor signal. For this, it is required to have an exact estimate of the tire frequency. The computation of the parameters themselves can be solved in sense of least squares in an iterative solution, just as for the polynomial APS. The sinusoidal APS provides the possibility to determine the time of an arbitrary phase angle of the sensor signal and is not restricted to the extremal points. However, due to its complexity its runtime performance is inferior by a factor of 2 compared to the polynomial APS.

5.1 Performance over SNR

In the previous chapter the worst conditions for simulations were at most with a noise level of SNR = 0dB. This means that the power of the sensor signal was equal to the power of the superimposed noise signal. In the following the SNR is altered from -15dB to +15dB.



(a) 2^{nd} order polynomial APS

(b) sinusodial APS

Figure 5.1: Performance of polynomial and sinusodial APS over SNR.

In real world situation the noise signal consists of an internal noise signal, which arise from the acceleration sensor and the measurement interface, and an external noise signal, which is caused by disturbances due to road conditions and vibrations between tire and road, which result from this. The internal noise of TPMS can be measured while the sensor remain in a steady state condition, e.g. lying on a table. The random variations in steady state are in the range from 0.25 to 0.5g standard deviation, which corresponds to SNR values of 12dB to 6dB. In real world situations conditions could come up which cause an SNR of -10dB or even worse.
Figure 5.1 depict the STD of the phase error as function of the SNR for the polynomial APS on the left and the sinusoidal APS on the right. Again, each value of the simulation is calculated from the STD of one thousand trials for the a parameter set. Since the number of samples M for the sinusoidal APS is almost twice as much as for polynomial APS, +3dB would need to be added to the values of the x-axis of the sinusoidal APS for direct comparsion of both simulations. Furthermore the impact of the ADC resolution is observed, as 2 different quantization levels were used. Beside the sensor signal without quantization two different ADC resolutions, an average resolution of $0.6\frac{g}{LSB}$ and a worst case resolution of $1.4\frac{g}{LSB}$, were applied to the sensor signal. While the progression of the simulation result itself is as expected - an increase of the STD of the phase error by decrease of the SNR - it is surprising that the ADC resolution shows almost no effect on the performance.
Both simulations in Figure 5.1 as well as the simulation for the third order polynomial APS in Figure 5.2 show an increase of the STD of the phase error increases between 10dB and 15dB SNR for the sensor signal with an ADC resolution of 0.6 $\frac{g}{LSB}$. This is due to the fact that the quantization noise in this range is prevailing over the superimposed noise signal.

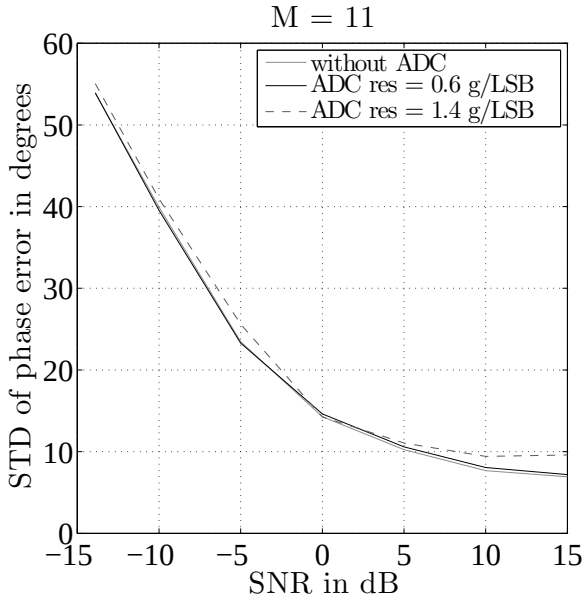


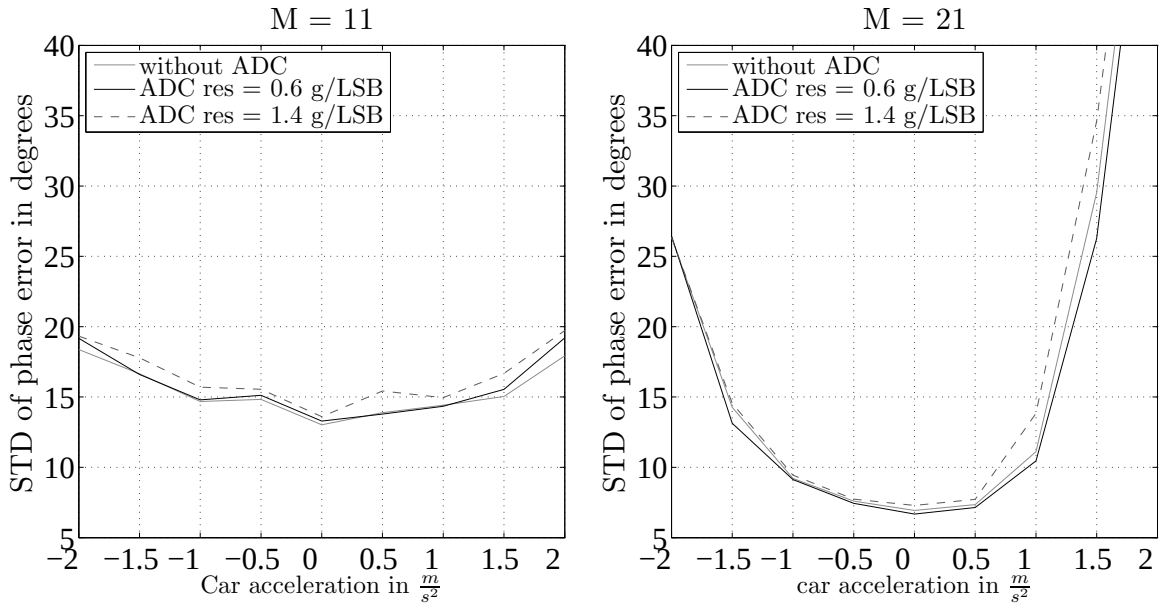
Figure 5.2: Performance of 3^{rd} order polynomial APS over SNR.

For the sake of completeness the performance of the cubic curve fit is depicted in Figure 5.2.

The progression of the phase error shows minor changes compared to the polynomial curve fit in Figure 5.1 and has a slightly better performance.

5.2 Performance over acceleration

In the following simulation the performance of both algorithms as function of the car acceleration is examined. The simulation is performed with an SNR of 0dB and the car acceleration is altered between -2$\frac{m}{s^2}$ and +2$\frac{m}{s^2}$. Again, to point out the impact of the quantization, the performance for the sensor signal as floating point and for two different ADC resolutions, same as in the last section, is displayed.



(a) 2^{nd} order polynomial APS

(b) sinusodial APS

Figure 5.3: Performance of polynomial and sinusodial APS over car acceleration.

For the polynomial APS the STD of the phase error stays constant to a certain extent over the car acceleration. In the range of constant velocity up to an acceleration of about 1$\frac{m}{s^2}$ the sinusoidal APS has a better performance then the 2^{nd} order polynomial APS. The break even point for the polynomial APS is between 1 and 1.5$\frac{m}{s^2}$ for positive acceleration and at about -1$\frac{m}{s^2}$ for deceleration. Beyond this points the polynomial algorithm illustrates its strength compared to the sinusoidal APS. While the STD of the phase error abrupt rise in case of the sinusoidal APS, the performance of the second order polynomial APS slowly degrades. For the third order polynomial APS, depicted in Figure 5.4, this effect is even far more distinctive, since the performance remains constant over the complete range. Again it becomes apparent that the ADC resolution has only a slight impact on the performance.

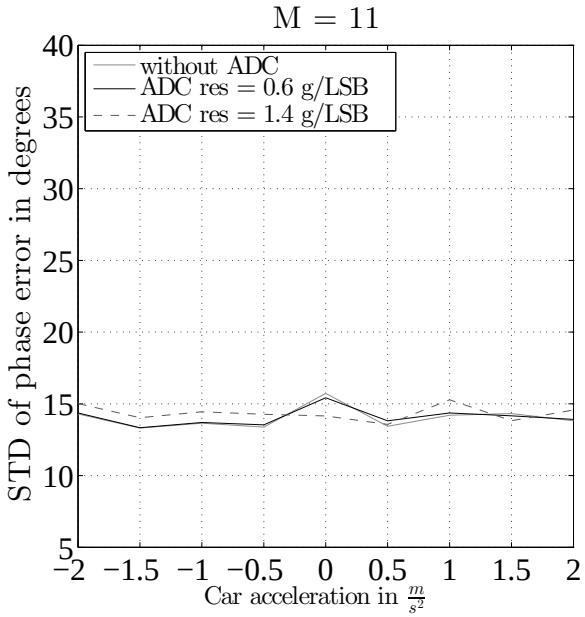


Figure 5.4: Performance of 3^{rd} order polynomial APS over car acceleration.

Simulations up to now have shown that the performance of the polynomial approach mainly depend on the SNR of input signal, the polynomial APS can therefore be stated as a function of SNR.

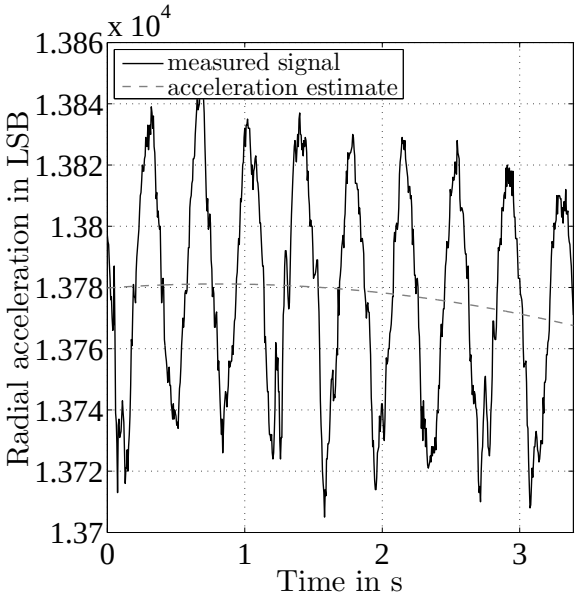
5.3 Performance on real measured data

Up to now the sensor signal in all simulations were generated by the simulation framework. Thus, the SNR as well as the phase angel $\varphi(v,t)$ of the data set were known from the outset. In the following chapter the performance of the polynomial APS on real measurement data is evaluated.

5.3.1 True value estimate

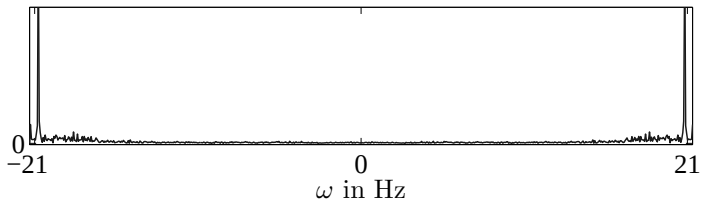
In order to rate the performance of the polynomial APS the real signal needs to be reconstructed from the noisy measurement data. In a first step the radial acceleration for a certain interval is determined. This interval covers about $300 \cdot \mathrm{OSF}$ samples, several periods of the sensor signal in order to to make a statement about the progression of the radial acceleration. A 2^{nd} order polynomial curve fit is applied on this data to determine the progression of the acceleration. Figure 5.3.1 depicts the measured signal on the observation interval and the estimated radial acceleration. By simple subtraction of the estimate from the measured signal one gets a straighten sensor signal a_{measured}. On the basis of the the original measurement data we can estimate the frequency of the wheel.

$$f_{est} = \sqrt{\frac{a}{4\pi^2 r}} \tag{5.1}$$

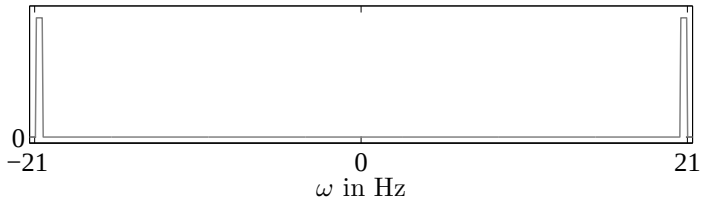


In order to remove the noise we transform the signal from time domain to frequency domain by applying Fourier transformation on the samples.

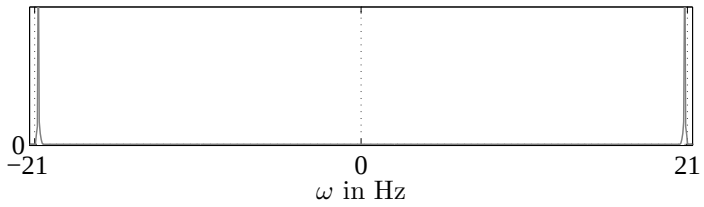
$$a[n] \Leftrightarrow A(j\omega)$$



(a) Noisy sensor signal in frequency domain.



(b) Band-pass filter.



(c) Filtered sensor signal in frequency domain.

Figure 5.5: Sensor signal before and after band-pass filtering in the frequency domain.

In the frequency domain we use a band-pass filter to remove the portion of noise. This means we simply multiply the spectrum of the signal with a certain pattern $\hat{h}(\omega)$ which sets all spectral sequences to zero, except those at the frequency of the sensor signal. Due to the uncertainty of the frequency estimation we have to set limits of ± 30 percent (ω_{min},ω_{max}) for the pattern in order to avoid a loss of information.

$$\hat{h}(\omega) = \begin{cases} 1 & \text{if } \omega_{max} \geq \omega \geq \omega_{min} \\ (3n+1)/2 & \text{else} \end{cases}$$

By applying reverse transformation onto the filtered sensor signal back into time-domain, we get the estimation of the measured gravity component a_g. This estimation serves as reference for the phase angle information.

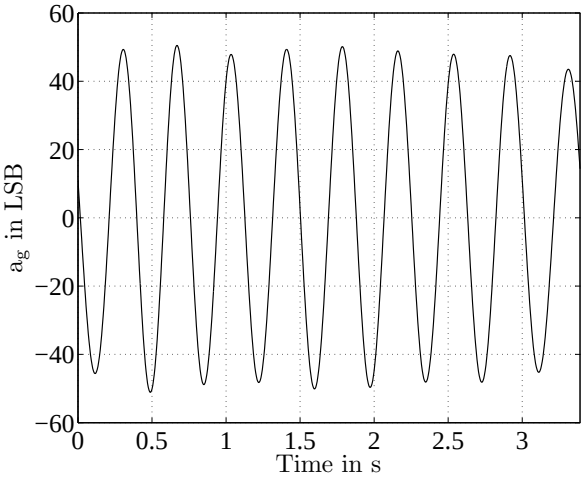


Figure 5.6: Estimation of gravity component a_g of the sensor signal.

To determine the SNR further computations are necessary. In the next step we downsize our observation window and examine about one period of the sensor signal. Through transformation into the frequency domain with increased resolution, we can now improve our first frequency estimation by selecting the spectral component with the highest magnitude. On the basis of this improved estimate we can now determine the sinusoidal parameter of the signal by applying a least squares estimate as described in [14]. We therefore propose a linear sinusoidal model that considers amplitude and frequency modulation:

$$y(n) \approx (A + n\dot{A}) \cdot \cos((\theta + \Delta\theta)n + \phi) \tag{5.2}$$

$y(n)$ denote the magnitude of the n-th samples in the observation windows, θ the phase estimate derived from our frequency estimate, $\Delta\theta$ denotes the deviation from this initial phase estimate θ, A the amplitude and $\dot{A}$ the amplitude modulation parameter. This equation can be linearised by introducing 4 basic functions

$$y(n) \approx c\cos\theta n + s\sin\theta n + dn\cos\theta n + tn\sin\theta n \tag{5.3}$$

with the 4 basic functions defined as follows:

$$\begin{aligned} c &= A\cos\phi \\ s &= -A\sin\phi \\ d &= \dot{A}\cos\phi - A\Delta\theta\sin\phi \\ t &= -\dot{A}\sin\phi - A\Delta\theta\cos\phi \end{aligned} \tag{5.4}$$

Equation 5.3 can be rewritten in vector-matrix notation.

$$\mathbf{y} \approx \mathbf{A}\mathbf{w}$$

Consequential the coefficient vector $\mathbf{w}$ can be expressed as

$$\mathbf{w} = [c, s, d, t]^T$$

The matrix $\mathbf{A}$ is composed of column-vectors $\mathbf{A} = \left[\mathbf{a}^c, \mathbf{a}^s, \mathbf{a}^d, \mathbf{a}^t\right]$. Their elements are computed according the following calculation rules:

$$\begin{aligned} a_n^c &= \cos\theta n \\ a_n^s &= \sin\theta n \\ a_n^d &= n\cos\theta n \\ a_n^t &= n\cos\theta n \end{aligned}$$

Just as in equation 3.4 from Chapter 3, the solution for the coefficient vector $\mathbf{w}$ stated as follows:

$$\mathbf{w} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{y}$$

Once the equation is solved, the sinusoidal parameters can be derived from the values of the coefficient vector.

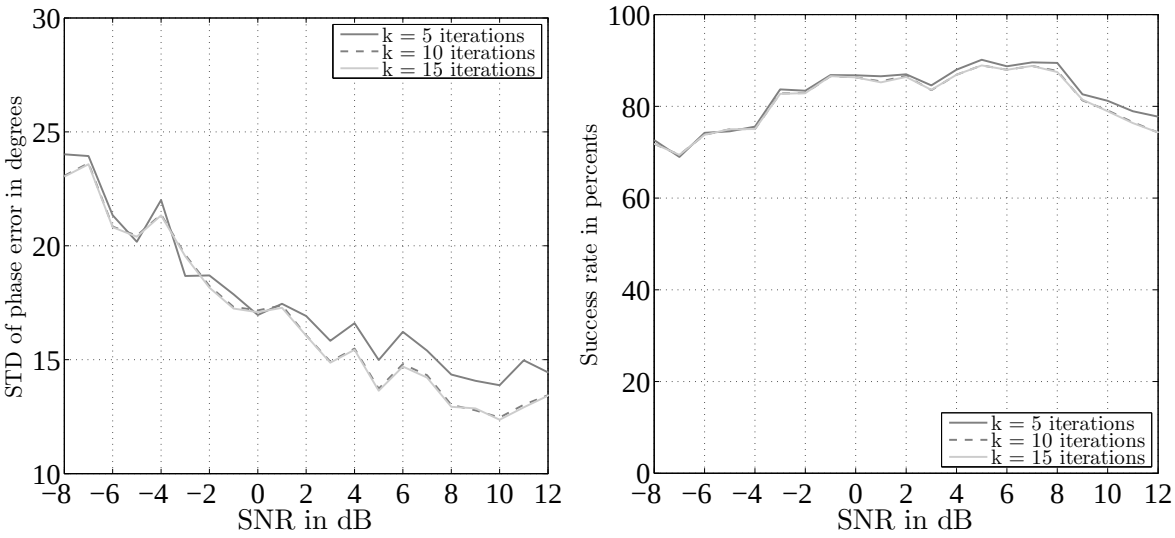
$$\begin{aligned} A &= \sqrt{c^2 + s^2} \\ &= \sqrt{A^2\cos^2\phi + A^2\sin^2\phi} \\ &= \sqrt{1 + \underbrace{A^2(\cos^2\phi + \sin^2\phi}_{=1}} \end{aligned}$$

$$\begin{aligned} \Delta\theta &= \frac{d\cdot s + t\cdot c}{A^2} \\ &= \frac{(\dot{A}\cos\phi - A\Delta\theta\sin\phi)\cdot -A\sin\phi + (-\dot{A}\sin\phi - A\Delta\theta\cos\phi)\cdot A\cos\phi}{A^2} \\ &= \frac{A^2\Delta\theta\sin^2\phi + A^2\Delta\theta\cos^2\phi}{A^2} \\ &= \frac{A^2\Delta\theta(\sin^2\phi + \cos^2\phi)}{A^2} \end{aligned}$$

With these parameters we can reconstruct the sensor signal a_{est} and determine the noise level based on Equation 4.1.

5.3.2 Simulation with real measured data

The sensor samples for this simulations stem from a TPMS wheel unit which performed a driving cycles of several minutes. SNR and phase angle of the data were computed afterwards according to the considerations we made in this section. Figure 5.7 depicts the performance of the second order polynomial APS and the success rate, both as function of the SNR . As some uncertainty in the estimation of the reference values (SNR and phase angle) still remain, we define our confidence interval between -5dB and 5dB SNR. Results beyond that shall be assumed as rough estimate of the performance of the polynomial APS. In addition the impact of the iteration count was observed.

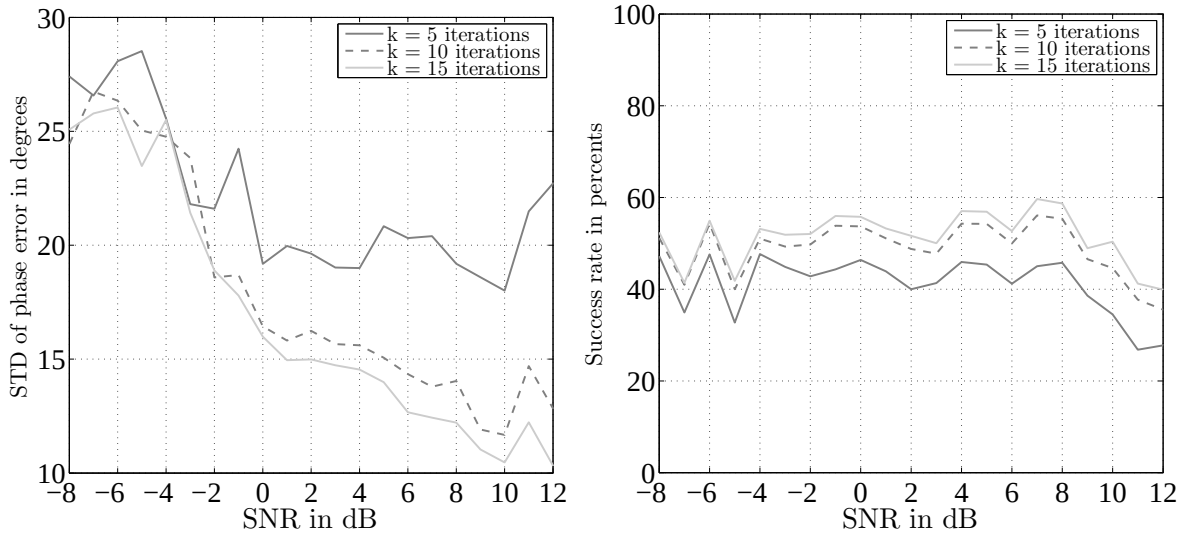


(a) Performance over SNR

(b) Success rate over SNR

Figure 5.7: Performance of second order polynomial APS over SNR.

In our confidence interval the simulation shows a better performance for negative SNR then the simulation depicted in Figure 5.1. For instance at -5dB SNR the performance is about 15 degree better as expected from the simulations. In contrast at 5dB SNR the performance is 5 degree worse then expected. As already noticed in Chapter 4.5, the difference between the STD of the phase error for k=5 and k=10 iterations increases the better the SNR ratio gets. This effect is noticeable from 0dB upwards and has already been shown in the parameter evaluation in Figure 4.8. In contrast the impact of the number of iterations on the success rate is minimal.



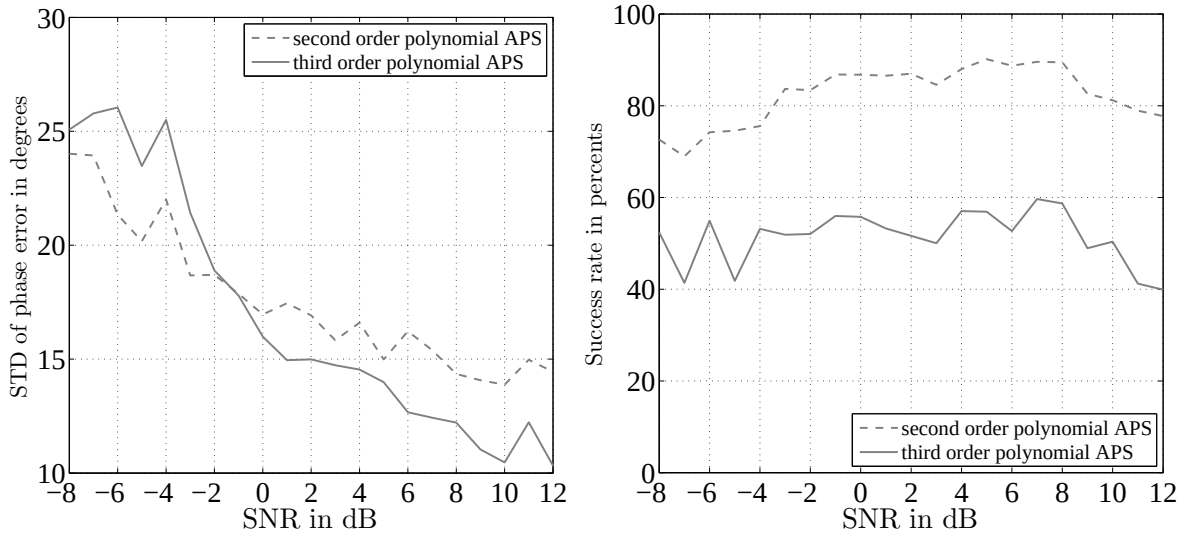
(a) Quadratic curve-fit

(b) Cubic curve-fit

Figure 5.8: Performance of third order polynomial APS over SNR.

Figure 5.8 depicts the performance and success rate for the third order polynomial APS applied on the measurement data. By contrast to the simulation with the second order polynomial APS, the impact of the number of iterations is way higher. This trend already became apparent in the parameter evaluation in Figure 5.1 although not to that extent. The progression of the success rate also shows dependency on the number of iterations. For k=10 and k=15 iterations the simulation result is in accordance with the expected performance based on the previous simulation results.

In the following we examine the performance of second order polynomial APS by contrast with the third order algorithm. Based on the results up to now the simulations were performed with k=5 iterations in case of the second order polynomial APS an k=15 iterations for the third order polynomial APS. Beforehand it must be said that the acceleration for the simulation depicted in Figer 5.10 was determined by the measured sensor samples and therefore only depicts an estimation.
Figure 5.9 shows the performance and success rate for both degrees of polynomial APS over the SNR. Like the simulations in Section 5.1 had already shown, the performance of the third order polynomial APS - relating to our confidence interval - is slightly better. An increase of the number of iterations for the second order algorithm would be able to flatten this difference for positive SNR. Regarding the success rate the second order polynomial APS shows, with up to 90 percent, a far better performance then the cubic algorithm.



(a) Performance over SNR.

(b) Successrate over SNR

Figure 5.9: Comparison of second and third order polynomial APS over SNR.

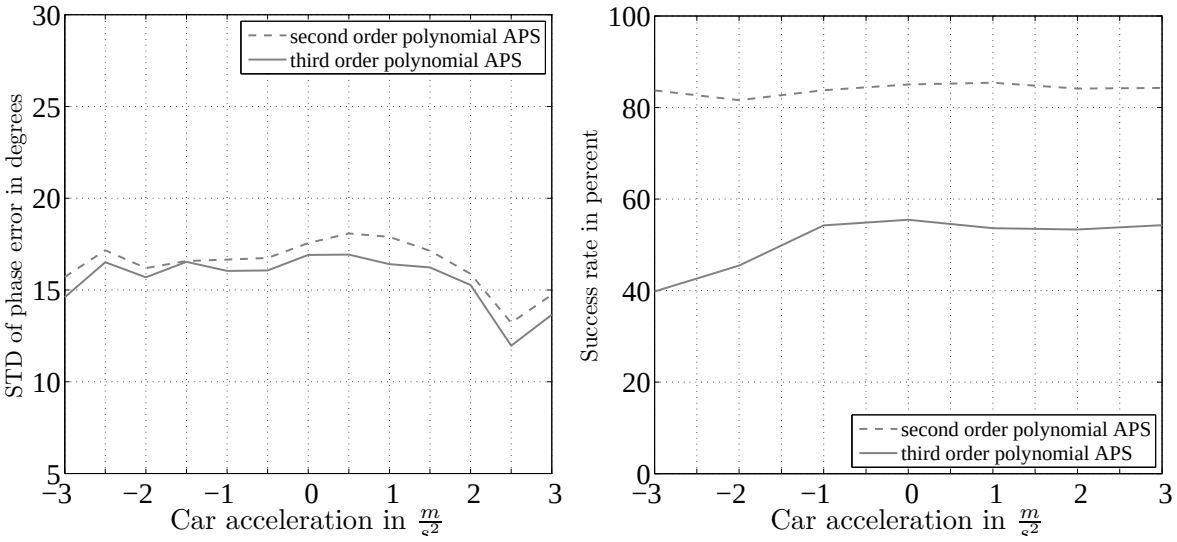
As last simulation in this section we consider the performance of the polynomial APS as function of the car acceleration. For the estimation of the car acceleration over a certain interval we compute a second order polynomial, just like at the beginning in Section 5.3.1. Based on this values we can determine a frequency estimate, as stated in equation 5.1, for the beginning and the end of the interval. with the help of this estimate we can compute the cars velocity.

$$v_{car} = 2\pi\, r_{tire} f$$

Since the length of the interval respectively its duration Δt is known, we can calculate the acceleration as a linear estimate:

$$a_{est} = (v_{car_{end}} - v_{car_{start}}) \cdot \Delta t \tag{5.6}$$

Figure 5.10 depict the performance for both polynomial APS over the acceleration. Naturally the acceleration values represent only estimates. However the progression of both STD is similar to the simulation results in Section 5.2, although the difference between both polynomial APS is mean and the performance of the second order polynomial APS is better then expected. In spite of the uncertainty due to the estimates, the constant performance over the acceleration assumed in Section 5.2 is confirmed.



(a) Performance over car acceleration.

(b) Cubic curve-fit

Figure 5.10: Comparison of second and third order polynomial APS over car acceleration.

6 Fixed point implementation

Because of the hardware limitations of the 8051-based TPMS the implementation of the tire-localisation algorithm has to meet special demands. As the hardware platform is a 8 bit microcontroller the smallest possible data type should always be favoured for computation. Moreover the lack of a floating point unit (FPU) requires an implementation of fractional numbers with fixed point numbers.
In fixed point representation the implied decimal point, also referred to as radix point, is always at the same location and the number of digits left and right to it is fixed.

$$x = \underbrace{\overbrace{b_N b_{N-1} b_{N-2} b_{N-3} \cdots}^{\text{integer part}} . \overbrace{\cdots b_3 b_2 b_1 b_0}^{\text{fractional part}}}_{\text{N bit}}$$

The digits of a N-bit number representing the integer part are denoted as a while the digits of the fractional part are denoted as b. Therefore a fractional format is described in the following way: **a.b**. This notation is based on the format used in [9].

In the following the difference between integer and fixed point representation shall be pointed out by an example. Considering the number of bits N = 16 and the partitioning between integer and fractional part, according to the notation above, with 3.13 . The range of numbers representable by an signed 16 bit integer goes from 2^{-15} to $2^{15} - 1$. For the fixed point representation the integer and the fractional part need to be considered separately. The numerically largest number which can be represented by the fractional part is $2^{13} - 1$ multiplied by the scale factor $\frac{1}{2^b} = 2^{-13}$. The numerically smallest number for the fractional part is zero. As a result from this the resolution for this fixed point representation can be derived. This is the ‚smallest non-zero magnitude representable' [9], in other words the value of one least significant bit (LSB). Hence the resolution for this example is $\frac{1}{2^{13}}$.
On the other hand the integer part is treated as is usually the case. The most significant bit (MSB) serves as a sign bit. Thus the range of signed numbers representable by the integer part is from -4 to 3.

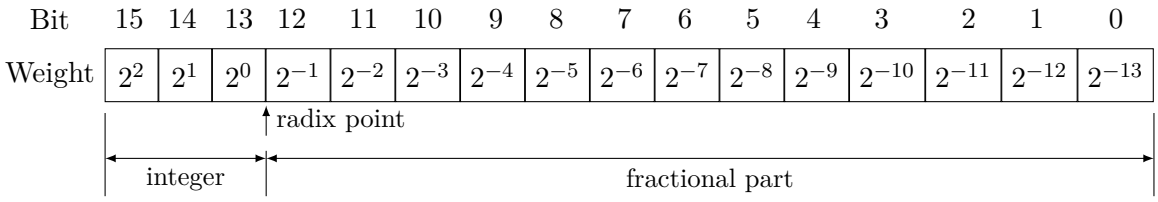


Figure 6.1: Bit weighting for fixed point format 3.13 .

Aware of this the conversion between float value and binary representation can now be deduced. Taking a decimal number 0.03125 and transforming it into a fixed point 3.13 number is done by simply scaling this value by 2^{-13}. In this case the result is 256 or, in binary representation, 0b0000 0001 0000 0000. For this example the conversion was easy since 0.03125 is an integer multiple of the resolution. From this it follows that the conversion from fractional to fixed point representation and vice versa is nothing else then a simple shift operation. Since the conversion from a fraction number into a fixed point representation is of no interest, this issue will not be addressed in-depth any further.

6.1 Arithmetic rules

Arithmetic operations on fixed point numbers differ from those for decimal numbers. Therefore the rules for the operations used for the fixed point implementation of the tire-localisation algorithm will be explained briefly in the following.

Addition: All operands have to be represented by in the same fixed point format - their fractional points need to be aligned. One only have to take care that the sum of the operands fit in the underlying data type.

Multiplication: Both Operands can directly by multiplied, regardless of the position of their fractional point. The format of the result is composed of the sum of fractional part and the integer part digits. For example the multiplication of a 4.4 and a 2.6 number result in a 6.10 number((4+2).(4+6)).

Division: The division operation is the opposite of the multiplication operation. Therefore it is based on the same rules, except that the format of the result is not the sum but the product of the fractional part and the integer part digits.

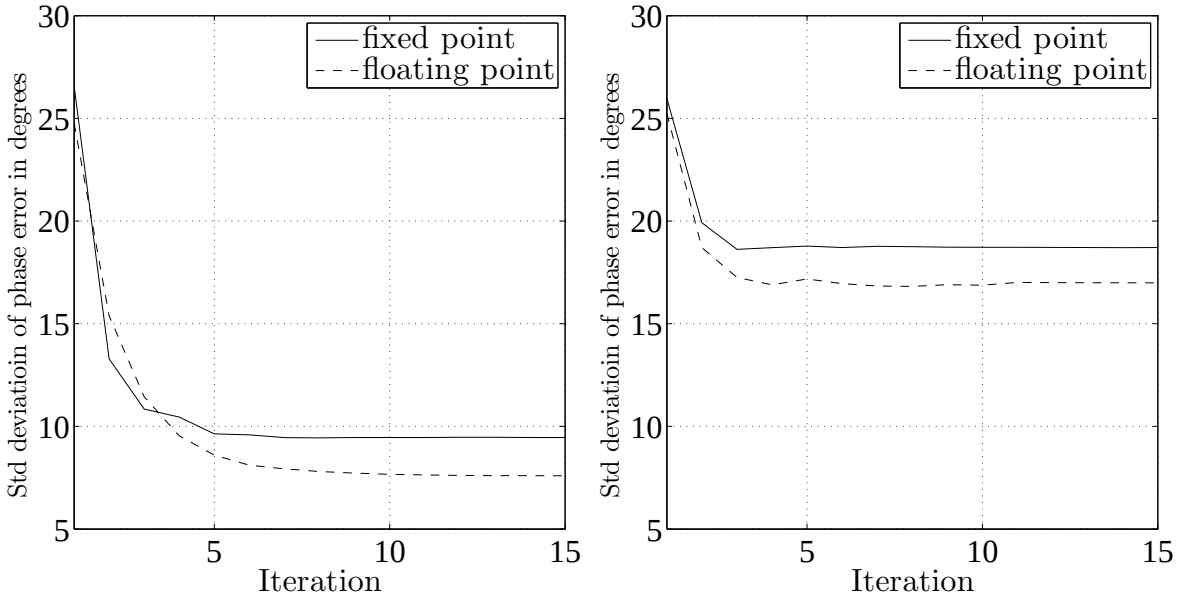
6.2 Scale factor

To minimize a loss in precision the operands need to be scaled. The precision that could be achieved depends on the one hand on the data type, respectively the number of available bits, and on the other hand on the magnitude of the integer part. This means always a trade-off between loss in accuracy and the risk of an overflow. To choose an appropriate scale factor the range of the magnitude first need to be identified. As stated in Chapter 2 the input samples of the pressure sensor oscillate with ±1g around a certain mean. By zero-mean normalization the magnitude get scaled down without any loss of information.
Assuming a ADC resolution of 0.6 $\frac{g}{lsb}$ and a SNR of 0dB the magnitude of a single ADC sample y_{1x} is composed as follows:

$$y_{1\mathrm{x}} = \pm\, 1.5 \;+\; \underbrace{\text{noise}}_{\text{of the same order}} \;+ \text{safety margin}$$

$$y_{1\mathrm{x}} = \pm\, 4\,\mathrm{lsb}$$

As each sample is composed by the sum of OSF=8 consecutive samples, the magnitude increases to $y_{8x} = \pm$ 32 LSB. The samples supplied by the ADC are 16 bit integer values in the format 16.0 . At first glance one might assume that 6 bit for the integer part - 5 bit for the magnitude and 1 bit for the sign - would be sufficient and therefore the fractional part may be 10 bit width. For test purpose a high level model of the fixed point implementation of Gauss-Seidel was developed, providing the possibility of an overflow detection. As input data real measurement data from the setup described in Chapter 8 was used. It turned out that the standard deviation of a single samples magnitude $y_{8x}[i]$, calculated from a data set of 600 trials, is in the range of $\pm$34.5 LSB. The integer part of the fixed point representation therefore needed to be extended by 1 bit to the format 7.10. The input samples as well as the constant values therefore need to be scaled by 2^7, which is equivalent to left arithmetic shift by 7 digits. Simulations have shown that this scale factor caused no overflow on the evaluated data.
Figure 6.2 depict the difference in the precision between the iterative floating point and the fixed point curve fitting algorithm. The left figure represents the simulation result with a clean sensor signal. After 8 iterations the standard deviation for both implementations becomes steady. One can see that precision loss of the fixed point implementation cause a difference of about 1.7 degrees.
The Figure on the right depicts the simulation results with an erroneous sensor signal, superimposed with a noise signal at an SNR of 0dB. After 10 iterations both curves become steady. The difference between both curves is about the same as in the first simulation, even though the overall phase error is increased by 9 degrees.



(a) Clean sensor signal

(b) Erroneous sensor signal

Figure 6.2: Difference in precision between fixed point and floating point implementation.

```
1   /*
2     scale sensor samples by 2^7
3   */
4   for (n = 0; n < M; n++)
5   {
6     y[n] = y[n] << 7;
7   }
8
9   /*  compute error vector;
10    initial state: error vector = sensor samples
11  */
12  sum_error = 0;
13  for (n = 0; n < M; n++)
14  {
15    sum_error = sum_error + y[n];
16  }
17
18  /*
19    deltaW = a' * e;
20    for first column of A-Matrix: a(:,1) = const. = NORM[0]
21  */
22  delta_w = ( ( (signed long) NORM[0] * (signed long) sum_error ) >> 7 );
```

The code listing above shows an extract of the fixed point implementation of the polynomial APS algorithm.

6.3 Iterative square root approximation

For the calculation of the extremal points of a third order polynomial it is required to solve a square root. For these purpose an iterative method with low computational effort is required. The Babylonian Method, which is a special case of the Newton-Raphson method, meet this criteria. For our scope of application the values those square root are sought are always positive and real valued. Under this conditions converge of the algorithm is guaranteed. Computing the square root of a real valued number $\sqrt{a}$ can also be considered as computing the root of a function $f(x) = x^2 - a$. Beginning with the Newton-Raphson formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_i)}$$

and substituting the function $f(x_n) = x_n^2 - a$ and $f'(x_n) = 2x_n$

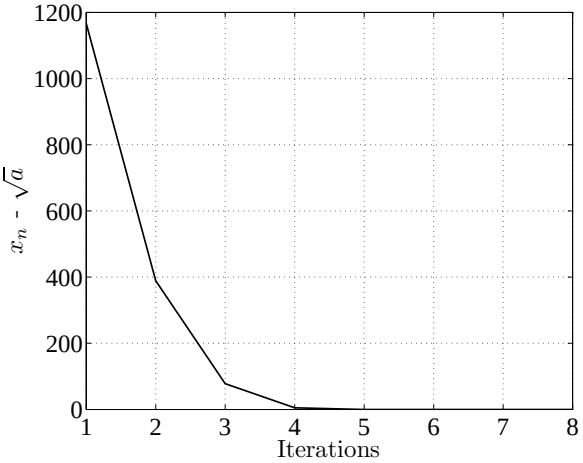
$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n}$$

leads to the representation of the Babylonian method [8] [2].

$$x_{n+1} = \frac{1}{2}(x_n - \frac{a}{x_n})$$

In the following the rate of convergence was analysed for different initial seeds x_0. Evaluation showed that, when a seed was chosen which is near the solution, the gain in speed of

convergence is marginal. Generally speaking the algorithm can be consider as converged after 6 iterations. As an example the square root from the calculation of the extremal points of a cubic polynomial, defined by $\mathbf{c} = \begin{bmatrix} 24 & 49 & -1146 & -405 \end{bmatrix}^T$, is examined. Figure 6.3 depict the error in each iteration step. As initial seed $x_0 = 100$ was chosen.



Simulations for different square roots and different seeds have all shown to be similar to the curve progression above. In general it can be stated that the result after 5 iterations yields sufficient accuracy.

7 Optimisation

In the following Chapter methods to estimate and improve the quality of the result of a quadratic curve fit are introduced. The first one enables to evaluate the quality of the curve fit after the first iteration and therefore provides the opportunity to abort the computation in an early stage. The second method introduces several threshold levels for the polynomial coefficients to filter poor estimates and to raise the precision. Finally a moving average filter is introduced which enables, with little additional computational effort, a gain of the performance.

7.1 Termination condition for Gauss-Seidel

In the analysis of the simulation results a criteria to decide on success or failure of the Gauss-Seidel algorithm in the first iteration stage was found. Based on the value of the quadratic coefficient $c_2^{(1)}$, where the superscript index denotes the iteration count of the Gauss-Seidel algorithm, we can decide whether the algorithm will lead to a successful curve fit or not. A successful curve fit means that an extremal point within the observation interval T_{OBS} is found.
Figure 7.1 illustrates the values of the quadratic coefficient $c_2^{(1)}$ after the first iteration of Gauss-Seidel and its probability of occurrence. Coefficients of the gray curve led to a successful curve fit while the coefficients in the black curve didn't.

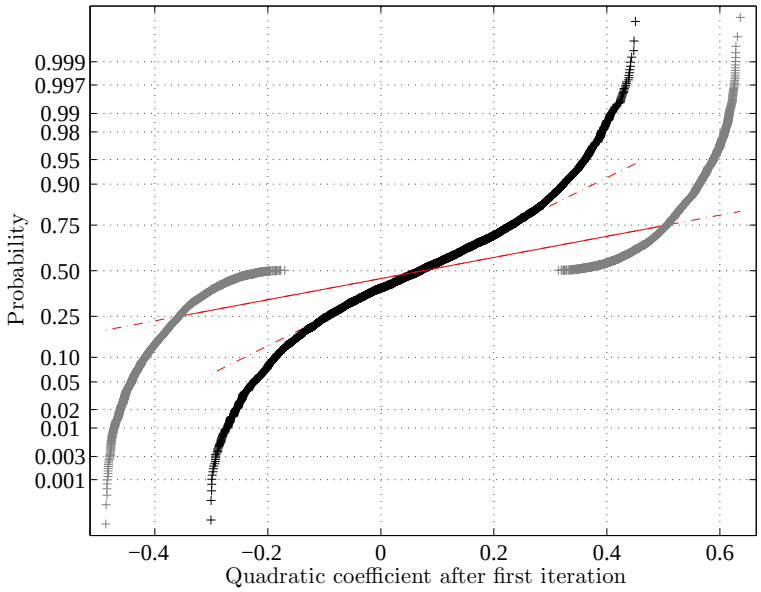


Figure 7.1: Distribution of quadratic coefficients.

Obviously for quadratic coefficients in the range of $0.35 > c_2^{(1)} > -0.2$ the iterative procedure can be aborted which means a saving of iterations and time. The asymmetrical distribution of the curve respective the thresholds mentioned above are result of the alignment of the columns in matrix **A** defined in Section 3.4.2 and a characteristic behaviour of Gauss-Seidel.

7.2 Coefficient Threshold

A glance on the coefficient of a quadratic polynomial provide information of the fitting curve progression.

$$y = c_0 + c_1 \cdot x + c_2 \cdot x^2$$

While the linear and the constant coefficients c_0 and c_1 define the offset on the x- and y-axis, the quadratic coefficient c_2 provide information about speed of decrease of the parabola from its maximum.

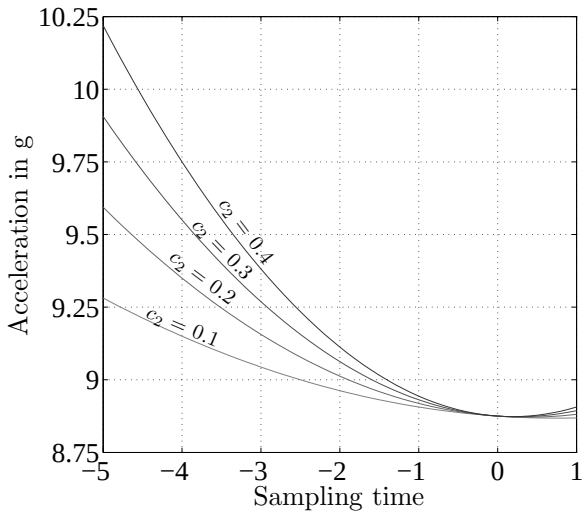


Figure 7.2: Progression of a polynomial by variation of its quadratic coefficient.

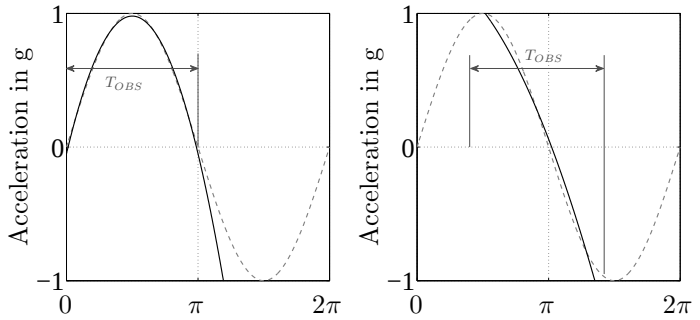
As shown in Figure 7.2 the higher the magnitude of the quadratic coefficient the sharper the curve declines, the lower the magnitude of the quadratic coefficient the flatter is the progression of the curve and the higher the probability that the observation interval fell into a range of the signal which involves no extremal point.
Simulation results have shown that the quadratic coefficient is a good criteria to evaluate the quality of a curve fit. Figure 7.3 shows the impact of the position of the observation interval on a second degree curve fit. In Figure 7.3a the observation interval encloses an extremal point and the curve fit follows the sensor signal precisely, whereas the observation interval in Figure 7.3b is located between two extremal points and the curve fit follows the sensor signal imprecisely. In addition the curve fit depicted in Figure 7.3b provides no information about the time of any of the extremal points. In turn, based upon the quadratic coefficient one can conclude the quality of a curve fit. The higher magnitude of the quadratic coefficient of the second degree curve fit in Figure 7.3a

$$y = -0.0499 + 1.3114 \cdot x + -0.4174 \cdot x^2$$

compared with the curve fit in Figure 7.3b

$$y = 1.3511 + -0.0060 \cdot x + -0.1290 \cdot x^2$$

illustrates this.



(a) $c_2 = -0.4174$ (b) $c_2 = -0.1290$

Figure 7.3: Progression of second degree curve fit for different positions of the observation interval.

Figure 7.4 depicts the distribution of the quadratic coefficients as function of the phase error. As one can see, the smaller the magnitude of the quadratic coefficient is the greater is the variation of the phase error. Three different threshold levels are marked in this Figure to illustrate the variation of the phase error below these limits.

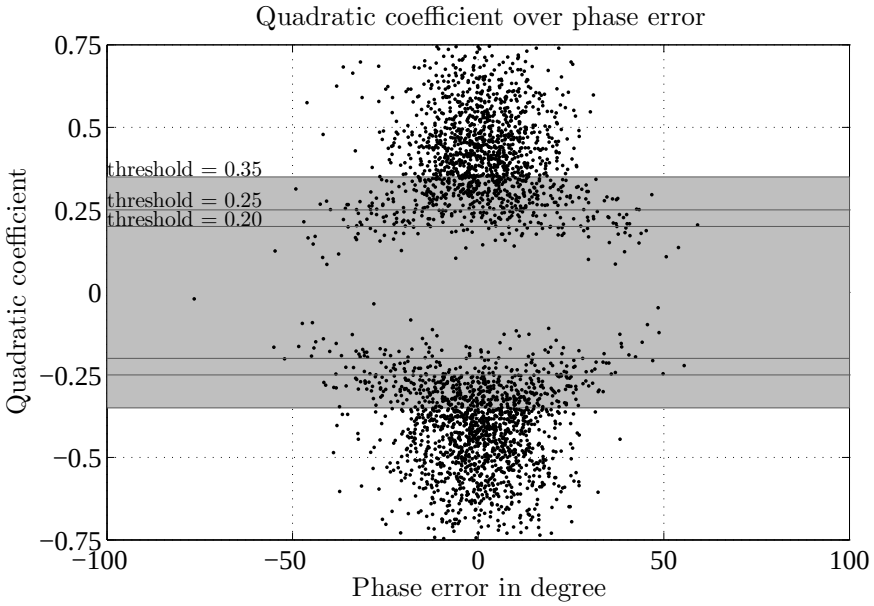


Figure 7.4: Scatter plot of quadratic coefficient over phase error.

Table 7.1 and Table 7.2 summarize an improvement in the standard deviation of the phase error for the different threshold levels depicted in Figure 7.4. The simulations these values were taken from were performed with an SNR of 0dB, an ADC resolution of $0.6\frac{lsb}{g}$ and M = 11 samples.

M = 11, SNR = 0dB

threshold	fail rate percent	STD of phase error degree
without offset and quantization error		
0	36.6000	14.1656
0.20	39.3200	13.0995
0.25	43.4400	12.2880
0.35	58.68	11.4105

Table 7.1: STD of phase error and fail rate for different threshold levels.

In addition the simulation were executed both, with and without quantization errors and an offset error of 6g. Due to the application of threshold levels to the simulation results were discarded and therefore the fail rate increases.
For the cubic curve fit no correlation between the values of the coefficient and the phase error were found.

M = 11, SNR = 0dB

threshold	fail rate percent	STD of phase error degree
with offset and quantization error		
0	38.4000	13.9261
0.20	41.0200	12.9060
0.25	44.8400	12.0186
0.35	59.9400	11.0691

Table 7.2: STD of phase error and fail rate for different threshold levels.

7.3 Moving Average

Moving average is a common technique to smooth out fluctuations from a data series. In our case this fluctuations arise from the noise superimposing the sensor signal.
Figure 7.5 illustrates the structure of a 3-point moving average filter which will be applied on the sensor data. The input signal x[n] is passed through a series of N buffers and multiplied with a certain filter coefficient h. The number of buffers define the order of the filter, not to be mistaken for the order of a polynomial.

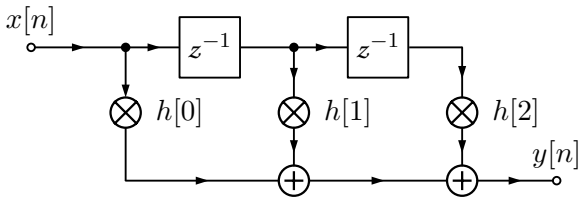


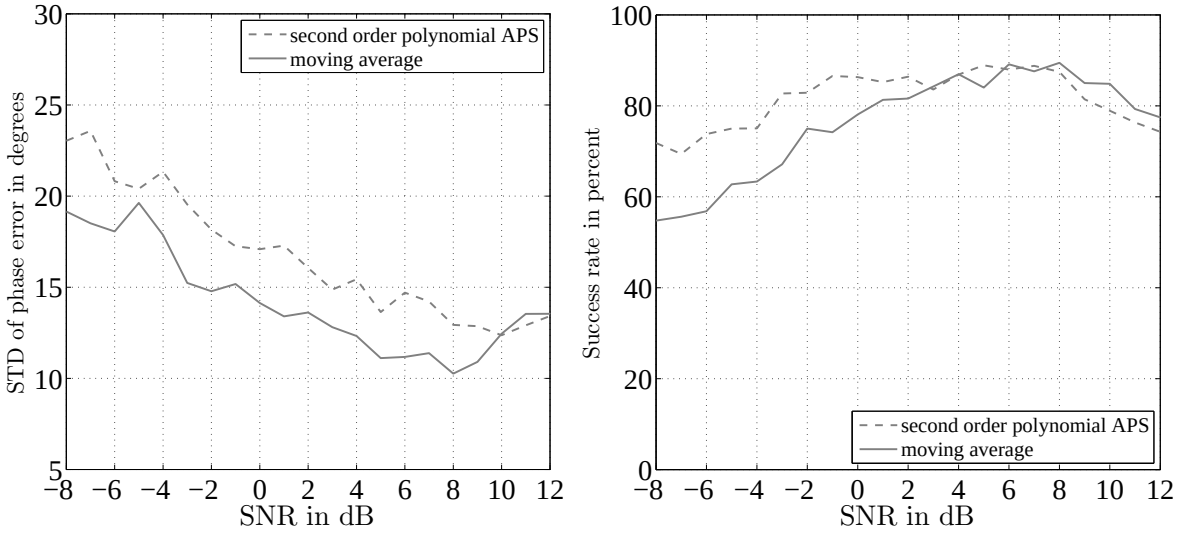
Figure 7.5: Structure of moving average filter.

The output signal y[n] is therefore composed of the weighted sum of the current input sample and N previous input samples.[7]

$$y[n] = \sum_{i=0}^{N} h[i] \cdot x[i]$$

The values of the filter coefficient are usually chosen so as to give a sum of one. In our case the filter coefficient vector has been chosen as $h = \begin{bmatrix} 0.25 & 0.5 & 0.25 \end{bmatrix}^T$, in order to attach greater significance to the sample in the middle of the filter window. The output is not valid as long as all buffers are filled. As a consequence two additional samples than for the usual second order polynomial APS are necessary.

Figure 7.6 depicts the gain in performance of the moving average filter applied on second order polynomial APS compared to the unoptimized polynomial APS. The simulations were performed with k=10 iterations for both algorithm. On our confidence interval, between -5dB and 5dB, this optimisation shows a gain in performance of upto 3 degree. As a drop of bitterness the success rate decreased by upto 10 percent for negative SNR.

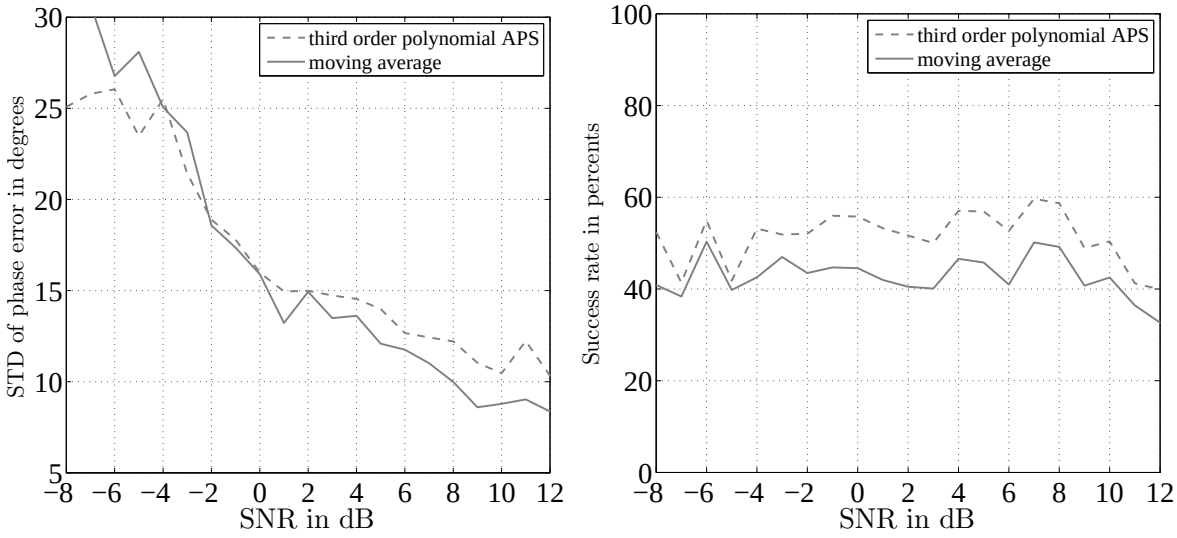


(a) Performance over SNR.

(b) Successrate over SNR.

Figure 7.6: Impact of moving average optimisation on second order polynomial APS.

Figure 7.7 depicts the progression of the filtered third order polynomial APS compared to the conventional polynomial APS. In contrast to the second order APS, the performance on the confidence interval remained constant for negative SNR and enhances at only for about 1 degree between 3dB and 5dB. The impact of the decline of the success rate is yet more serious since the success rate of the third order polynomial APS is innately more then 20 percent lower then the success rate of the second order polynomial APS. Thus, this optimisation should only be applied on the second order implementation. Having a look at Figure 5.9 clarifies that, with this optimisation applied on the second order implementation, the difference between the performance of third and second order polynomial APS becomes negligibly small despite the lower computational effort of the latter one.



(a) Performance over SNR.

(b) Successrate over SNR.

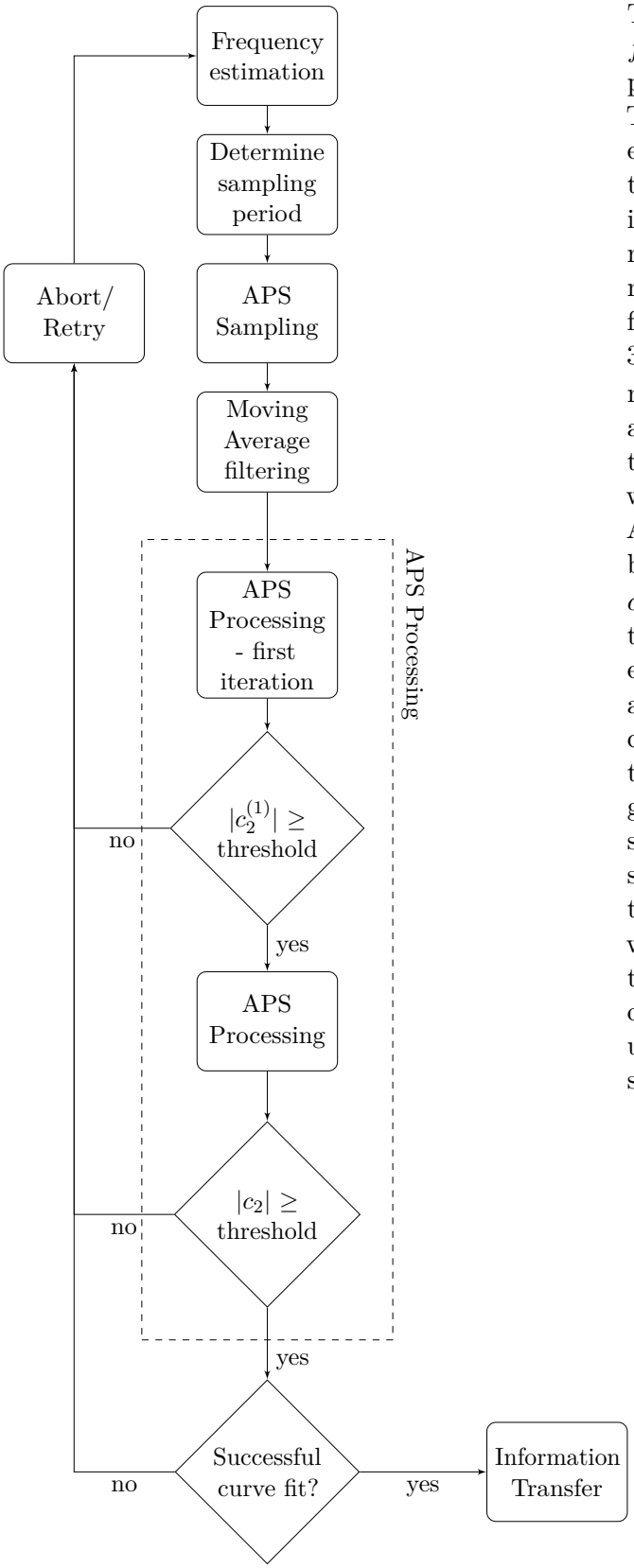
Figure 7.7: Impact of moving average optimisation on third order polynomial APS.

The additional computational effort for moving average filtering keeps within reasonable limits. Multiplication of the input samples x[n] by the filter coefficients h can be replaced by simple shift operations. The remaining operations limit to $2 \cdot (M+1)$ additions. The overall number of additions, as derived in Section 4.5, therefore add up to:

$$\begin{aligned} \#\text{additions} &= \underbrace{2 \cdot (M+1)}_{\text{moving average}} + \underbrace{K \cdot (M-1+\ N \cdot (M-2) + \ (N+1) \cdot M + N + 1)}_{\text{general second order polynomial APS}} \\ &= 24 + 5 \cdot (10 + 2 \cdot 9 + 3 \cdot 11 + 4) \\ &= 349 \end{aligned}$$

7.4 Final Comments

The flow chart for the second order polynomial APS described in Figure 4.2 changes by consideration of the optimisation methods discussed in this Chapter as follows:



The steps for the frequency estimation f_{est} and the determination of the sampling period T_{OBS} remain unchanged. The first modification in the program execution flow it the moving average filtering of the sensor samples. As stated in Section 7.3 M + 2 sensor samples are required for this optimization. In the next step the first iteration of the curve fitting algorithm, as described in Section 3.4.1, is executed. Based on the magnitude of the quadratic coefficient $|c_2^{(1)}|$ after the first iteration of the curve fitting algorithm the decision is reached on whether the execution of the polynomial APS is resumed or the algorithm will be restarted. If the quadratic coefficient $c_2^{(1)}$ meets the requirement to resume the program execution the remaining iterations of the curve fitting algorithm are executed. Based on the quadratic coefficient after the N-th iteration, again the decision is reached whether the algorithm shall be aborted or not. The step to decide whether the curve fit was successful or not remains unchanged. If the time value of the extremal point lies within the limits of the observation interval T_{OBS} the curve fit was successful otherwise the curve fit is considered as unsuccessful and the algorithm needs to start again.

8 Performance evaluation on chip

In the last Chapter of this thesis we will examine the performance of the polynomial APS on a TPMS wheel unit.

8.1 Measurement setup

With the help of a centrifuge we can let a TPMS wheel unit rotate at a specified constant frequency. The frequency range within the centrifuge can operate enables us to generate centrifugal accelerations up to 360g. In addition a climate chamber ensures constant ambient temperature.
The current angular position of the centrifuge is displayed by an increment signal. This increment signal has a saw-tooth waveform and encodes the angular position of the centrifuge as function of time. A rotation of 360 degree is equal to the time between two edges of the saw-tooth signal. The TPMS wheel unit triggers an *angle-reached-event*, that is a short pulse on a specified output pin when the point in time computed by the polynomial APS arises.

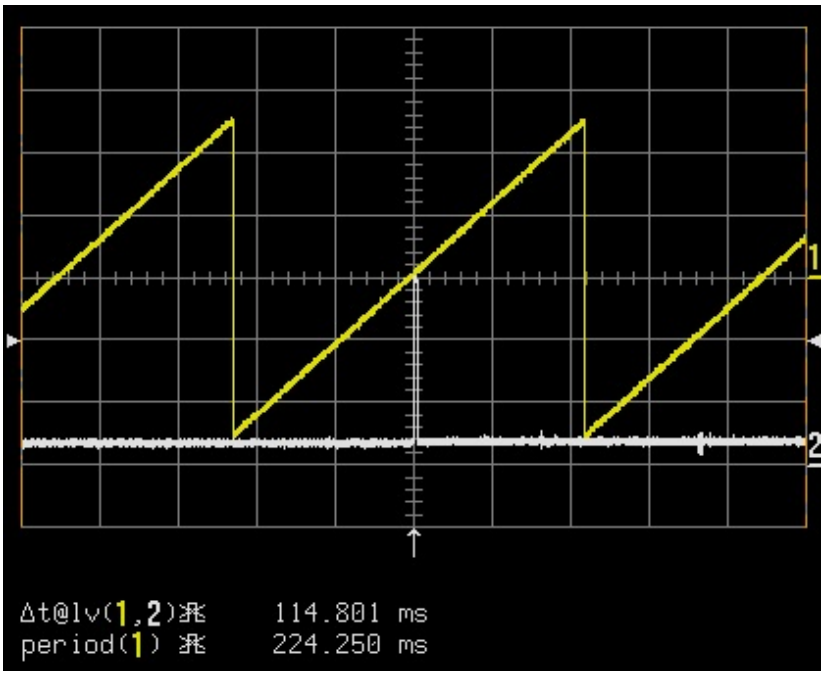


Figure 8.1: Typical trigger event displayed on an DSO.

Figure 8.1 depict the progression of the increment signal of the centrifuge as yellow curve and

the trigger pulse of the angle reached event as white curve on a digital sampling oscilloscope (DSO). In addition the period duration - the time between two negative edges of the increment signal - is measured by the DSO and the time difference between a negative pulse of the increment signal and the positive edge of the angle-reached-event, denoted by Δt in the Figure, is measured. With the exemplary values in Figure 8.1 we can calculate the angular position of the angle-reached-event.

$$\frac{360^\circ}{T} \cdot \Delta t = \frac{360^\circ}{224.25\text{ms}} \cdot 114.801\text{ms} = 184.3^\circ \tag{8.1}$$

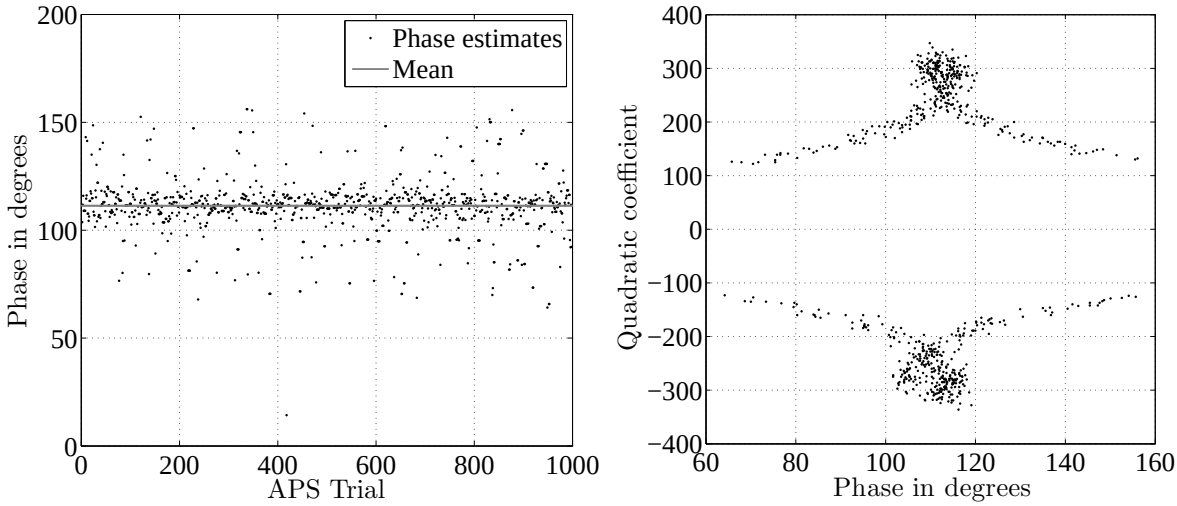
For this example we assume the negative edge of the increment signal to arise at 270 degree and the polynomial APS to be configured to trigger the angle-reached-event at the top position (90 degree), cf. Figure 2.2. Therefore the true angular position of the TPMS wheel unit is at $184.3^\circ - 90^\circ = 94.3^\circ$. Hence, the polynomial APS for this particular example has a phase error of $94.3^\circ - 90^\circ = 4.3^\circ$.
The centrifuge as well as the DSO are connected via General Purpose Interface Bus (GPIB) with a measurement PC. The measurement software on the PC stores the times values of the DSO and additional informations that can be transmitted by the TPMS wheel unit for a series of measurement.

8.2 Evaluation

In the following section we evaluate the performance of the second and third order polynomial APS on chip and compare the results with the simulations from the previous chapters. In the evaluation we examine the deviation of the phase estimates from their mean, not the phase error itself. Recalling the example from Section 1.3, for the mapping of the TPMS wheel unit it is only important that the transmission occur at a constant angle, as long as the offset from the true angular position of the extremal point remain constant.

8.2.1 Second order polynomial APS

For the following measurements the parameter set defined in Chapter 4 was used. The device under test (DUT) experienced an acceleration of 50g in the centrifuge. Figure 8.2 depict the measurement result of 1000 trials. On the left the estimated phase for each trial is shown, on the right the quadratic coefficient c_2 of the second order fit as function of the phase estimate. One can see that there is a tendency toward a phase angle around 111 degrees. In the distribution of the quadratic coefficient in Figure 8.2b a distinct tendency is also noticeable. Coefficients of the phase estimate below $|c_2| = 200$ scatter over a wide range, above $|c_2| = 200$ the phase estimate are located around 115 degrees with deviations of about $\pm$ 5 degrees. Compared to the assumptions based on the simulations in Chapter 7.2, especially the scatter plot in Figure 7.4, this tendency is far more distinctive than expected and relieves the filtering by certain coefficient thresholds.



(a) Deviation of estimated phase.

(b) Quadratic coefficient over estimated phase.

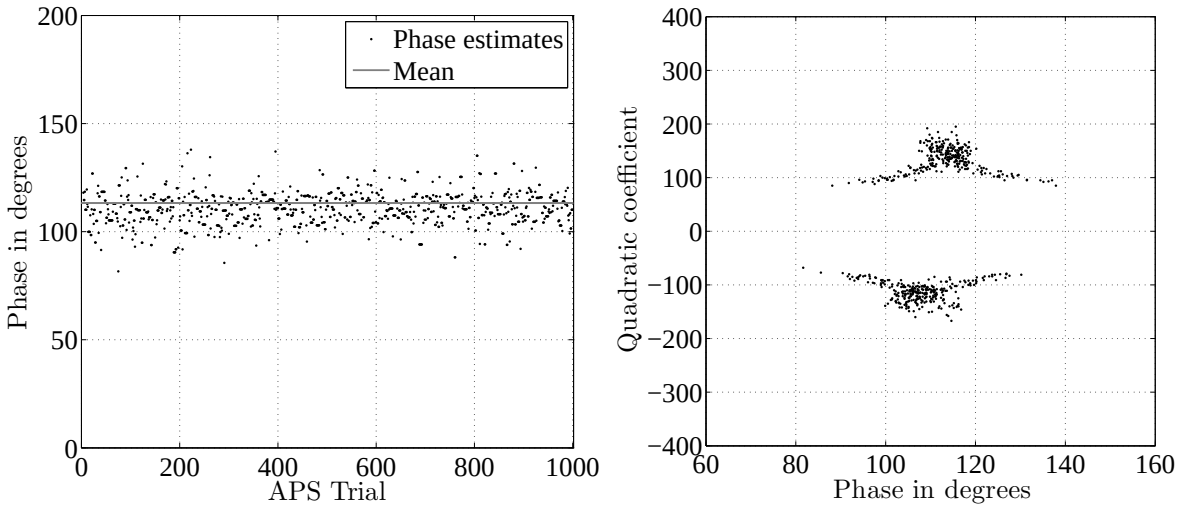
Figure 8.2: Measurement result of second order polynomial APS at $T_{OBS} = \frac{T}{2}$.

Table 8.1 summarizes the fail rate and STD of the phase estimate for several threshold levels. For the case of no threshold filtering the achieved fail rate is worse than the expectations based on the simulation results from Chapter 5.

threshold	fail rate percent	STD of phase estimate degree
0	29.00	12.4192
150	34.20	8.1354
175	39.90	5.4851
200	47.20	3.9307
250	62.10	3.6526

Table 8.1: Performance of second order polynomial APS for certain threshold levels.

In the next measurement a moving average filter was applied on the sensor data. Based on the simulations in Section 7.3 it is expected that the deviation of the phase becomes lower due to this optimisation, on the other hand the fail rate is expected to increases. The measurement approved this assumption as apparent from the results in Table 8.2. Figure 8.3a depict the estimated phase for each trial. By contrast to the measurement of the unoptimized polynomial APS in Figure 8.2 the extent of the scattering of the estimates declined. The distribution of the quadratic coefficient in Figure 8.3b shows a lower magnitudes of $|c_2|$ by contrast with the measurement before. Comparing the results of both measurements - with and without moving average - in case of no threshold filtering the deviation of the phase estimate is almost 5 degrees lower than for unoptimized second order polynomial APS. However, the fail rate increased by almost 15 percent. This corresponds with the assumptions from the simulation results from Figure 7.6.



(a) Deviation of estimated phase.

(b) Quadratic coefficient over estimated phase.

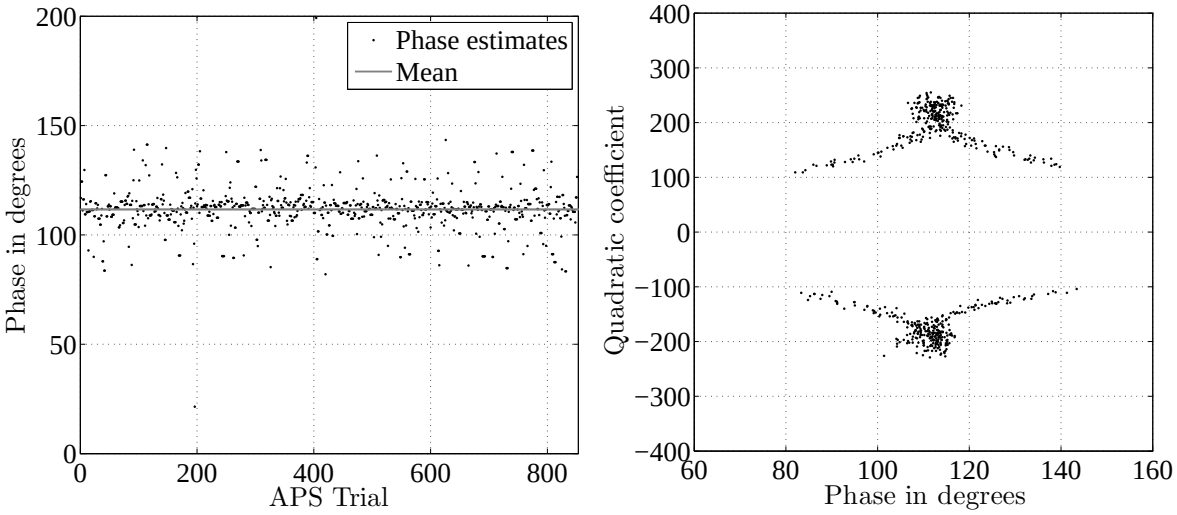
Figure 8.3: Measurement result of second order polynomial APS with moving average filtering at $T_{OBS} = \frac{T}{2}$.

threshold	fail rate percent	STD of phase estimate degree
0	43.30	7.8089
90	47.3	6.7335
100	53.5	5.4929
125	75.2	4.3111

Table 8.2: Performance of second order polynomial APS with moving average filtering.

Table 8.2 summarizes the fail rate and the deviation of the phase estimate for different threshold levels. In comparison with the results of the unoptimized polynomial APS from Table 8.1 it becomes apparent that threshold filtering is sufficient to gain better results with lower fail rate then by applying moving average filtering.

The measurement setup allows us to use the exact frequency of the centrifuge for the computation of T_{OBS}. From the parameter evaluation in Chapter 4 it became apparent that the best phase estimate, regardless of the fail rate, can be achieved at an observation interval of $T_{OBS} = \frac{2T}{3}$. Figure 8.4a depict the phase estimates for a measurement of 850 trials and an observation interval of $\frac{2T}{3}$. The lower variation of the estimates compared to the measurements before is noticeable instantly. From the results in Table 8.3 one can see that the STD of the phase estimates has decreased by about 3 degree, compared to the measurement at $T_{OBS} = \frac{T}{2}$. On the other hand the fail rate also increased by 3 percent, just as in the simulation in Figure 4.5. By using several threshold levels the precision can be increased with comparatively small increase of the fail rate. This result can be considered as the best accomplishable concerning the precision for the second order polynomial APS.



(a) Deviation of estimated phase.

(b) Quadratic coefficient over estimated phase.

Figure 8.4: Measurement result of second order polynomial APS at $T_{OBS} = \frac{2T}{3}$.

threshold	fail rate percent	STD of phase estimate degree
0	32.35	9.13
125	36.81	6.95
150	45.54	3.86
175	59.67	2.72
200	75.85	2.64

Table 8.3: Performance of second order polynomial APS at $T_{OBS} = \frac{2T}{3}$.

Runtime performance: A measurement of the runtime of the second order polynomial APS yielded in 34.2ms. With the optimisation method described in Section 7.1 we are able to terminate the polynomial APS after one iteration. One iteration lasts about 6.1ms. Regarding the unoptimized second order polynomial APS without threshold filtering, c.f. Table 8.1, this means an average runtime for a successful phase estimate of:

$$\underbrace{34.2\text{ms}}_{\text{runtime of polynomial APS}} + \underbrace{(\frac{1}{1-0.29} - 1)}_{\text{number of unsuccessful runs}} \cdot \underbrace{6.1\text{ms}}_{\text{time for one iteration}} = 36.7\text{ms}$$

The optimisation of moving average filtering of the sensor data extends the runtime of the second order polynomial APS by 2ms to 36.2ms. In the same circumstances, c.f. Table 8.2, the average runtime for the algorithm therefore adds up to:

$$36.2\text{ms} + (\frac{1}{1-0.43} - 1) \cdot 6.1\text{ms} = 40.8\text{ms}$$

8.2.2 Third order polynomial APS

As last measurement we examine the performance of the third order polynomial APS. Figure 8.5 shows the phase estimates over 600 trials. Similar to the measurement of the second order polynomial APS with moving average filtering in Figure 8.3, the measurement shows only few outliers. The STD of the phase estimate for this measurement came up with 5.6 degrees. Together with a success rate of 37 percent the performance is better than expected from the simulation in Figure 7.7.

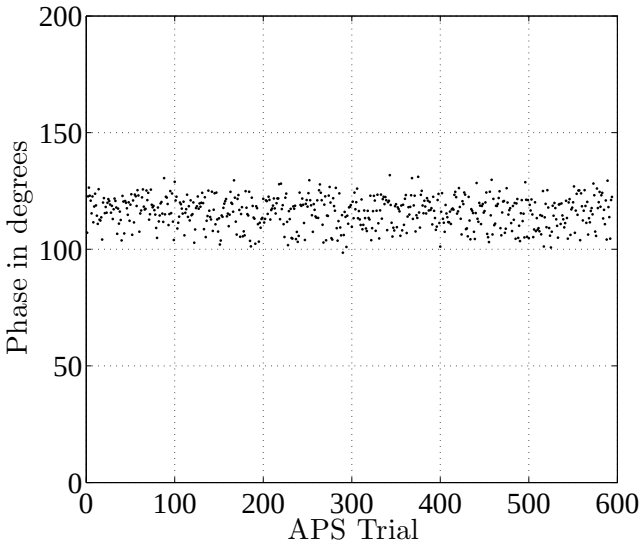


Figure 8.5: Deviation of estimated phase for third order polynomial APS.

Nevertheless, the third order polynomial APS has not proven to be suitable for the tire localisation. The time for one run of the algorithm takes 116.2ms. Based on the measurement result from above, the average runtime adds up to:

$$116.2\text{ms} + (\frac{1}{1-0.37} - 1) \cdot 116.2\text{ms} = 184.4\text{ms}$$

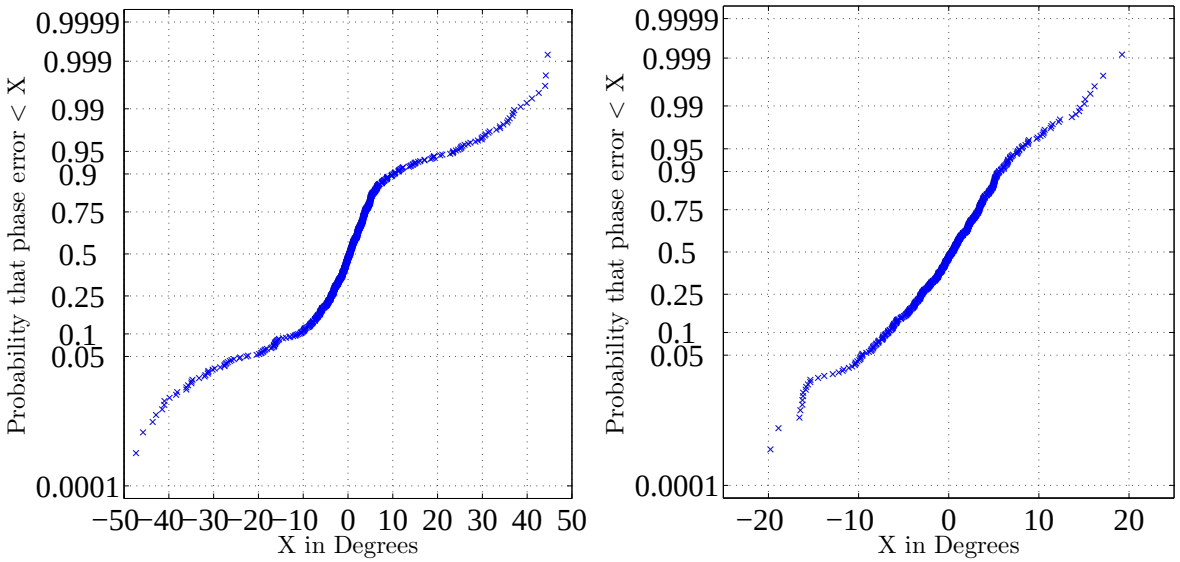
Despite the precision, the same result can be achieved with the second order polynomial APS in less time, amongst others because of the possibility to terminate computations in an early stage. Consider the measurement results from Table 8.1, with a threshold of 175 a similar precision can be achieved. For this case, the average runtime adds up to

$$34.2\text{ms} + (\frac{1}{1-0.39} - 1) \cdot 6.1\text{ms} = 38.1\text{ms}$$

Thus the second order polynomial APS is almost 5 times faster than the third order algorithm.

8.3 Resumé

The measurement results of this chapter have mainly approved the assumption suggested from the simulations in the previous chapters. As already have been shown in the parameter evaluation of the observation interval in Figure 4.5, the best results regarding the precision are achieved at $T_{OBS} \approx \frac{2T}{3}$. The difference in the measurements summarized in Table 8.1 and Table 8.3 verified this assumption. Due to constraint from the fixed point implementation, the overall results differ from the results in Figure 4.5.
Alike described in Chapter 7, the impact of the moving average filter indeed brings an increase in the performance but also yields in an increase of the fail rate. The difference between the measurement results from Table 8.2 and Table 8.1 points this out.
The analysis of the runtime of both algorithm made clear that - at this juncture - the implementation of the second order polynomial APS should be preferred to the third order algorithm, not least because of the optimisation methods applicable to the second order polynomial APS.



(a) Unfiltered measurement data, see Table 8.1.

(b) Measurement data with threshold of 175, see Table 8.1.

Figure 8.6: Probability plots of second order polynomial APS phase error.

Figure 8.6 shows the phase errors of all trials of the measurement in summarized in Table 8.1. On the left the phase errors without threshold filtering are visualized as probability plot and on the right with a threshold filter of $|c_0| \geq 175$. In comparison to the probability plot of the phase errors of the sinusoidal APS one can see that the majority of the samples in Figure 8.6b follow the same probability distribution as the sinusoidal APS in Figure 8.7.

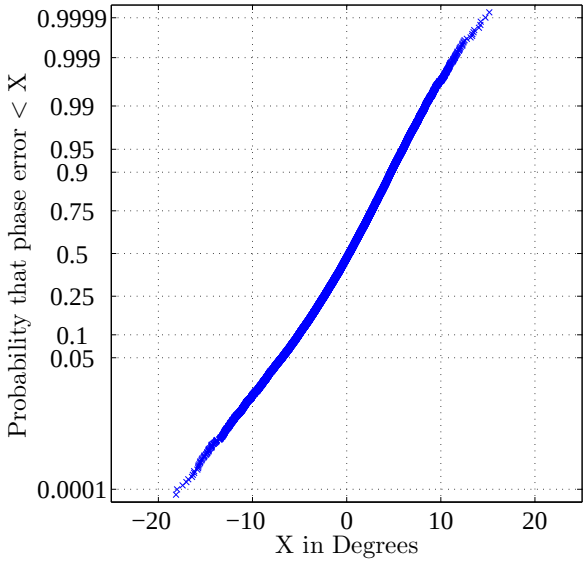


Figure 8.7: Probability plots of sinusoidal APS phase error.

9 Conclusion and further research

9.1 Summary

In the process of this work it has been shown that the challenge of tire localisation can be resolved with a polynomial curve fitting algorithm, in the following denoted by polynomial APS which stands for angular positioning sensing, with reasonable complexity.
The topic of tire localisation concerns with the mapping of a TPMS wheel unit ID to a tire. Therefore the signal characteristics of an built-in acceleration sensor is utilized to estimate the angular position of a wheel. The TPMS control unit in the car can draw conclusions on the arrangement of the tires based on their angular position. For simulation purpose a model of the sensor signal was establish considering deficiencies of the sensor and external effects such as noise. Based on this sensor signal simulations were performed to determine the parameters of the polynomial APS algorithm. In addition to the polynomial complexity, which was limited to second and third order polynomials, the impact of the observation interval on which the polynomial APS is applied, was investigated. Since the wave form of the acceleration sensor signal is sinusoidal, we can speak in terms of multiple of a period time T. For second order polynomials an observation interval of a half period has shown to be suitable, while the observation interval for third order polynomial fits is at about $\frac{3T}{4}$, both with respect to inaccuracy in the parameter estimation on chip. To make the algorithm applicable on a hardware platform, the performance with limited resources was investigated. Beside the number of samples on the observation interval the number of iterations of the curve fitting algorithm is limited. The decision of the values of these parameters was always a trade-off between precision and complexity respectively runtime. Neither of these requirements was allowed to miss out. With these parameters and the sensor signal model the performance of the polynomial APS algorithm was investigated under various conditions - different noise levels as well as car acceleration - and compared to another APS based on a sinusoidal approach. The simulations came up with the conclusion that the performance is mainly influenced by the signal-to-noise ration. Comparison with the simulation result of the sinusoidal APS revealed the strength of the polynomial APS. Over the range of various acceleration values the performance remained constant to a certain extent. Especially for high acceleration or deceleration the precision of the polynomial APS is above the precision of the sinusoidal APS. In a first step, a simulation with real measurement data of a TMPS wheel unit, the performance expected from the simulation was validated. In addition the impact of the number of iterations was examined and compared with the simulation results. Before an implementation in hardware was possible thoughts on an implementation of the algorithm in fixed point decimal notation had been given. To avoid loss in precision the sensor samples need to be scaled - with respect to the underlying data type - before being applied on the algorithm. To increase the performance, several method to optimize the polynomial APS were found. In simulations and later in measurements on-chip their usability for the polynomial

APS was evaluated. In general, the measurements on chip confirmed the simulation results to a certain extent. Due to the measurement results regarding precision, fail rate and runtime it became apparent that the second order polynomial APS full fills all the requirements with an advantage in runtime and complexity over the third order polynomial APS.

9.2 Further research

Before the polynomial APS is ready to be used in an tire localization application further research in some field are required.

As the most of the optimization methods identified in this thesis are limited to the second order polynomial APS it might be worth to be considered if further investigation in the optimization of the third order polynomial APS should be applied. Simulations and the evaluation on chip have proven an advantage of the third order polynomial approach over the second order one regarding the precision. However, the great disadvantage of this method is its comparatively long runtime. If it could be managed to reduce the runtime, down to some reasonable limit, the third order polynomial APS might be the better option.

Measurements of the second order polynomial APS have shown, that the magnitude of the quadratic coefficient $|c_2|$ varies for different observation intervals. It may be assumed that the magnitude also depends on the angular velocity $\omega(v, t)$. To ensure proper application of the optimisation of threshold filtering it will be necessary to determine the relationship between the magnitude of the coefficient and $\omega(v, t)$.

Since both algorithm, the sinusoidal APS and the polynomial APS, reveal their performance in different fields of application a cooperation of both algorithms should be intended . In case of constant velocity up to low acceleration, the sinusoidal APS should be preferred to the polynomial APS. For higher acceleration of the car the polynomial APS shows it advantages.

List of Abbreviations

TPMS	Tire pressure monitoring system
STD	Standard deviation
APS	Angular positioning sensing
OSF	Oversampling factor
ABS	Anti-lock brake system
TCS	Traction control system
ECU	Engine control unit
LS	Least squares
SDD	Strictly diagonally dominant
FPU	Floating point unit
LSB	Least significant bit
FFT	Fast Fourier transformation
DSO	Digital sampling oscilloscope

Bibliography

[1] C.M. Bishop, Pattern recognition and machine learning, Information Science and Statistics, Springer, 2006.

[2] Steven C Chapra and Raymond P Canale, Numerical methods for engineers, McGraw-Hill Higher Education, 2010.

[3] Germund Dahlquist and Å ke Björck, Numerical methods in scientific computing. vol. 1., Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2008.

[4] EU, Regulation (ec) no 64 of the economic commission for europe of the united nations (un/ece) - uniform provisions concerning the approval of vehicles with regard to their equipment which may include: a temporary-use spare unit, run-flat tyres and/or a run-flat system, and/or a tyre pressure monitoring system, Official Journal of the European Union (2010), 21,29.

[5] Åke Björck Germund Dahlquist, Applied numerical methods using matlab, Courier Dover Publications, 1974.

[6] John M. Brown Joseph J. Carr, Introduction to biomedical equipment technology, third edition, Prentice Hall, 1998.

[7] Walt Kester et al., Mixed-signal and dsp design techniques, Newnes, 2003.

[8] Olga Kosheleva, Babylonian method of computing the square root: Justifications based on fuzzy techniques and on computational complexity, Fuzzy Information Processing Society, 2009. NAFIPS 2009. Annual Meeting of the North American, IEEE, 2009, pp. 1–6.

[9] Hazarathaiah Malepati, Digital media processing: Dsp algorithms using c, Newnes, 2010.

[10] Elizabeth Million, The hadamard product, 2007.

[11] N. Persson, Event based sampling with application to spectral estimation, Division of Control & Communication, Department of Electrical Engineering, Linköpings universitet, 2002.

[12] N. Persson, F. Gustafsson, and M. Drevö, Indirect tire pressure monitoring using sensor fusion, SAE paper **1** (2002), 1.

[13] US DEPARTMENT OF TRANSPORTATION, Federal motor vehicle safety standards; tire pressure monitoring systems; controls and displays, RIN 2127-AI33 (2005), 9–13.

[14] Jean-Marc Valin, Daniel V Smith, Christopher Montgomery, and Timothy B Terriberry, An iterative linearised solution to the sinusoidal parameter estimation problem, Computers & Electrical Engineering **36** (2010), no. 4, 603–616.

[15] Walter Waddell, Inflation pressure retention effects on tire rolling resistance, vehicle fuel economy and co2 emissions presented to the california air resources board, 2008.

[16] Yang Won Young, Wenwu Cao, Tae-Sang Chung, and John Morris, Applied numerical methods using matlab, Wiley, Hoboken, NJ, 2005.

[17] Ph.D. Esteban Cañibano Álvarez, European parlament - type approval requirements for the general safety of motor vehicles, IP/A/IMCO/IC/2008-112 (2008), 49,51.