Position and velocity estimation from incremental encoders

Citation for published version (APA):

Document status and date:
Published: 01/01/2003

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 20. Aug. 2019
Position and velocity estimation from incremental encoders

D.C.C. Verstraeten

DCT - 2003.107

Eindhoven University of Technology
Department of Mechanical Engineering
Control Systems Technology

Eindhoven, November 2003
## Contents

1 Introduction ................................................................. 1

2 Measurement set-up and software ..................................... 2
   2.1 Measurement set-up .................................................. 2
   2.2 DSP software ....................................................... 2
       2.2.1 Initialization .................................................. 2
       2.2.2 Buffer ......................................................... 3
       2.2.3 Scaling ......................................................... 4
       2.2.4 Look up table .................................................. 4
       2.2.5 Calculation of the interpolation extrapolation .......... 6
       2.2.6 Output ......................................................... 7
   2.3 Numerical problems ................................................ 7
       2.3.1 Floating point ................................................ 7

3 Experiments .................................................................... 10
   3.1 4 time-stamps polynomial fit ...................................... 11
       3.1.1 1 revolution per second and 1 Hz disturbance .......... 11
       3.1.2 1 revolution per second with 10 Hz disturbance ....... 12
       3.1.3 1 revolution per second with 20 Hz disturbance ....... 13
       3.1.4 1 revolution per second with 40 Hz disturbance ....... 14
       3.1.5 1 revolution per second with 60 Hz disturbance ....... 15
   3.2 6 time stamps .......................................................... 16
       3.2.1 1 revolution per second with 1 Hz disturbance ......... 16
       3.2.2 1 revolution per second with 10 Hz disturbance ...... 17
       3.2.3 1 revolution per second with 20 Hz disturbance ...... 18
       3.2.4 1 revolution per second with 40 Hz disturbance ...... 19
       3.2.5 1 revolution per second with 60 Hz disturbance ...... 20
       3.2.6 2 revolutions per second with 1 Hz disturbance ...... 21
       3.2.7 2 revolutions per second with 10 Hz disturbance ..... 22
       3.2.8 2 revolutions per second with 20 Hz disturbance ..... 23
       3.2.9 2 revolutions per second with 40 Hz disturbance ..... 24
       3.2.10 2 revolutions per second with 60 Hz disturbance ... 25
       3.2.11 8 revolutions per second with 1 Hz disturbance ..... 26
       3.2.12 8 revolutions per second with 10 Hz disturbance ... 27
       3.2.13 8 revolutions per second with 20 Hz disturbance ... 28
       3.2.14 8 revolutions per second with 40 Hz disturbance ... 29
       3.2.15 8 revolutions per second with 60 Hz disturbance ... 30
   3.3 8 time stamps .......................................................... 31
       3.3.1 2 revolutions per second with 1 Hz disturbance ..... 31
3.3.2 2 revolution per second with 10 Hz disturbance . . . . . 32
3.3.3 2 revolution per second with 20 Hz disturbance . . . . . 33
3.3.4 2 revolution per second with 40 Hz disturbance . . . . . 34
3.3.5 2 revolution per second with 60 Hz disturbance . . . . . 35

4 conclusions 36

5 recommendations 36
  5.1 Ways to measure the look up table ......................... 37
1 Introduction

Incremental position encoders are widely used for position and velocity measurements. In most of the applications the encoder pulses are counted and sampled by a fixed frequency. Incremental encoders have a finite resolution. The maximum deviation will be around a half of an encoder step. This noise is also called quantization noise. This quantization noise is increasing as the resolution of the encoders used decreases.

The velocity is commonly obtained by differentiating the position signal. Differentiating the quadrature signal of the encoder to enquire the velocity results also in undesired quantization noise. To suppress the quantization noise a low pass filter can be used. A disadvantage of filtering is that a phase lag occur in the velocity information.

To improve the accuracy, an encoder with a higher resolution can be used, but this is more expensive. Another option is to collect the time stamps of the quadrature events and make a polynomial fit and extrapolate. These calculations can be done with a Digital Signal Processor (DSP). A DSP is especially made for "real time" processing of data. An experimental setup is made to test the feasibility of this idea. The experimental test set-up is thoroughly described in [1] and [2].

This report describes the practical implementation of polynomial fitting and extrapolation of an incremental position encoder signal. In chapter two a short introduction is given of the experimental set-up. Chapter two describes also the DSP software and the numerical problems which occur. In chapter three some result of the experiments are given. At last the conclusions and recommendations are given.
2 Measurement set-up and software

2.1 Measurement set-up

In report [1] and [2] an experimental test set-up is described which is used for the measurements. The test set-up consist of two encoders. The HEDS-5540 encoder with 100 slits and a Heidenhein ROD-426 encoder with 5000 slits. The Heidenhein ROD-426 encoder is given 20000 increments per revolution. The assumption is made that the Heidenhein generates the exact position. The Heidenhein encoder is called high resolution encoder in this report.

The HEDS-5540 encoder has a two 2+1 channel quadrature output. One channel is reserved for the index pulse. This index pulse is needed for applying the look-up table. The HEDS-5540 is called the low resolution encoder in this report. The HEDS-5540 encoder is given 400 increments per revolution. The encoders are driven by a motor.

The low resolution encoder is connected to a interface that is assigned with the acquisition of the time-stamps. This interface is called Special Rotational encoder Interface (SRI). This SRI consist of of a 32 bit quadrature counter and a 64 bit time stamp generator. Time stamps are generated with a resolution of 100 ns. The SRI records each change of the quadrature count value (either +1 or -1) and the time-stamp at which the corresponding change has occurred. The values of the time stamps and the quadrature count is stored in a Multi Purpose Memory bank. The values can be read from this memory by the Texas Instrument DSP (TMS320C40). The DSP calculates the polynomial fit. The results of the calculations can be read by the host. In the next section the software of the DSP is described.

2.2 DSP software

The DSP software can be divided in several procedures. Figure 1 shows all procedures of the DSP software. In the following sections a description is given of the several procedures.

2.2.1 Initialization

At the end the calculated data must be compared with the data from the high-resolution encoder. However the starting position is not know in advanced. Also to apply a look-up-table the starting position is needed. Therefor a special starting procedure is applied. This starting procedure rotates the axis until
the index (ref) pulse is detected. This indicates the absolute position of the encoder. The speed applied during this procedure is very slow because the exact position of the high-resolution encoder is needed. So without missing any increment. The position of the index pulse of the high-resolution is stored. The position of the low-resolution encoder is reset.

2.2.2 Buffer

The collected data is stored in the A/B memory. The data from this memory will be lost after each switching moment. There arise a problem when there are less time stamps in the A/B memory than used during the calculation (low speed or very high frequency of sampling). A buffer which stores the last time stamps solve this problem. This prevent that data is lost. The length of the buffer is depending on the used time stamps during the calculation.
2.2.3 Scaling

Normally the values of the time stamps will increase as function of the time. With a second order polynomial-fit these values are taken to the power of four. With a third order polynomial-fit these values are even taken to the power of six. The scaling procedure is needed to prevent overflow during these calculations. The buffer of time stamps is filled as follows. From the whole "time stamps buffer" the first value is subtracted. The value of the first time stamp is stored. The first value becomes zero. The values of the buffer are now relative to the first value. With the scaled data the polynomial-fit is calculated. Another advantage of this method is that the accuracy is improved. The following table shows the scaling procedure schematic.

<table>
<thead>
<tr>
<th>Time buffer</th>
<th>Time buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>0</td>
</tr>
<tr>
<td>130</td>
<td>28</td>
</tr>
<tr>
<td>162</td>
<td>60</td>
</tr>
<tr>
<td>178</td>
<td>76</td>
</tr>
<tr>
<td>190</td>
<td>88</td>
</tr>
<tr>
<td>205</td>
<td>103</td>
</tr>
<tr>
<td>222</td>
<td>120</td>
</tr>
<tr>
<td>238</td>
<td>136</td>
</tr>
<tr>
<td>251</td>
<td>149</td>
</tr>
</tbody>
</table>

The quadrature information is also scaled. To make a comparison between the high-resolution encoder and the low-resolution encoder the values of the low-resolution encoder is multiplied with a factor 50. This again is multiplied with a factor 100 to apply the look up table.

2.2.4 Look up table

The look-up table is used to actively compensate the imperfections of an incremental encoder. The look up table is made with the assumption that the high-resolution encoder has no errors. To acquire the lookup table an experiment is done with a very low rotational speed. With such a low rotational speed it's possible to distinguish each pulse transition instant of the high-resolution encoder. Each transition instant of the low resolution encoder corresponds to one value of the high-resolution encoder. The experiment continues for 30 revolutions. From this experiment there are two different lookup tables generated. The first lookup table is acquired by taken the average of several revolutions. A array of 400 values is then derived which is shown in figure 2.

The graph shows that the error can be split up into two different parts. One part is the low frequency error, with a period equal to one revolution and a
sinusoidal-like shape. This is the error caused by the eccentricity and tilt of the code wheel. The second part is the high frequency error, with a period of one encoder cycle. This error is mainly caused by the misalignment of the pairs of light emitters-sensors and that the light emitters-sensors react different on rising and falling edge. To examine the influence of the low and high frequency there is made a second lookup table. This look-up table is acquired by filtering the first look-up table with a anti-causal filter. The filter window was set to 16 encoder increments. This look up table cancels only the high frequency error. In figure 2 the look up table is shown. To get an average value of the high frequency error the filtered look-up table is subtracted from the non filtered look-up table. The distribution of the errors are shown in figure 3. It shows that the high frequency has a gaussian distribution. With the different look-up tables it’s possible to test the influence of different encoders imperfections.

To apply the look-up-table the absolute position is needed. During the initialization, the value of the low-resolution encoder is reset at the index pulse. The assumption is made that never time-stamps are missed. The information of the look-up table is added. To calculate with integers, the collected look-up-table is multiplied with 100. By the scaling procedure the collected data was also multiplied with 100. The assumption is also made that the setup is always turning in one direction. If the set-up must be capable of turning two directions an another look-up table is needed. The electrical properties changes (rising edge become falling edge).
2.2.5 Calculation of the interpolation extrapolation

From the simulations of [1] it was proven that a second order polynomial fit was the best choice. An other advantage of a second order polynomial fit, is that also a velocity estimation can be evaluated at the same time instant. Before the calculation can performed, the number of time-stamps must be defined. A low order polynomial fit through these time-stamps is calculated with the least square method. If

\( n \) : number of time stamps
\( t_{1...n} \) : time information
\( x_{1...n} \) : position information

\[
A = \begin{bmatrix} t_1^2 & t_2 & 1 \\ t_2^2 & t_2 & 1 \\ t_3^2 & t_3 & 1 \\ \vdots & \vdots & 1 \\ t_n^2 & t_n & 1 \end{bmatrix}, \quad P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_n \end{bmatrix}, \quad B = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
\]

The following systems of linear equations has to be solved:

\[
AP = B \tag{2}
\]

and the least square method is formulated as follows:

\[
A_TAP = A_TB \Rightarrow (A_TA)^{-1}B \tag{3}
\]
To solve the above system, LU-factorization is used. Having the coefficients P of the fitted polynomial, the extrapolation is simply evaluating this polynomial at a certain time instant \( t_{\text{ext}} \). An estimation about the position \( \tilde{x} \) is then acquired:

\[
\tilde{x}|_{t=t_{\text{ext}}} = p_2 t_{\text{ext}}^2 + p_1 t_{\text{ext}} + p_0
\]  

An estimation about the velocity \( \tilde{v} \) is then acquired:

\[
\tilde{v}|_{t=t_{\text{ext}}} = 2p_2 t_{\text{ext}} + p_1
\]  

It is obvious that the time \( t_{\text{ext}} \) must be determined precisely. In practical situations the timing of the calculation will be very important. The time is depending on the calculation time of DSP. This calculation time is very depending on the number of time-stamps and the data type used during the calculation. In section 2.3 the different data types are explained.

2.2.6 Output

To export the data from the DSP must be converted to long data format. The calculated data is written to the C/D memory and is read by the host. The data can be further be examined with for e.g. Matlab. The results are given in section 3.

2.3 Numerical problems

The data of the DSP is organized into three fundamental types: integer, unsigned integer and floating point. The first two (also called fixed-point numbers) are limited in that they cannot simultaneously represent very large numbers or very small numbers using a reasonable word size. In the polynomial-fit calculation very large numbers are used. (e.g. time information to the power of four in a second order fit calculation). Therefore it's necessary to use the floating point format for a second order polynomial-fit calculation. In the next section a short description of a floating point is given.

2.3.1 Floating point

A floating point number can be represent with three bit parts. There is a sign bit. A sign bit of 0 represent a positive number and a sign bit 1 represent a negative number. There are some bits reserved for the mantissa and for the exponent. The total value can represented as \( \pm f \times 2^e \) where \( f \) is the mantissa
value; 2 is the radix or base (binary in this case); and e is the exponent of the radix.

The range of a number gives the limits of the representation while the precision gives the distance between successive numbers in the representation. If the numbers are not in the outer range it's called an overflow. If the numbers are not in the inner range it's called an underflow. Figure 4 shows this schematically.

The used Texas Instruments Digital Signal Processor (TMS320-C40) supports three floating-point formats:

- A short floating-point consisting of a 4-bit exponent, a sign bit, and an 11-bit mantissa.
- A single precision floating-point format by an 8-bit exponent field, a sign bit, and a 23-bit mantissa. (32-bit)
- An extended-precision floating-point format consisting of an 8-bit exponent field, a sign bit and a 31-bit mantissa. (40-bit)

If the accuracy of the calculation is still not sufficient it's possible to emulate a 96-bit float. A special emulator is developed by Texas Instruments. The disadvantage of the 96-bit floating point emulator is the calculation time. All most all values used for the calculation of the polynomial-fit has first to be converted to a 96 bit floating point. At the end result must be converted back to export to the host. The calculation time of a 96 bit value on a 32 bit DSP will be at least 3 times higher than an 32 bit value calculation. In the following table the range of the different floats of the used DSP are given.

<table>
<thead>
<tr>
<th>Float</th>
<th>Mantissa</th>
<th>Exponent</th>
<th>Low limit</th>
<th>High Limit</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>11</td>
<td>4</td>
<td>2&lt;sup&gt;-8&lt;/sup&gt;</td>
<td>2&lt;sup&gt;8&lt;/sup&gt;</td>
<td>2&lt;sup&gt;-11&lt;/sup&gt;</td>
</tr>
<tr>
<td>32</td>
<td>23</td>
<td>8</td>
<td>2&lt;sup&gt;-126&lt;/sup&gt;</td>
<td>2&lt;sup&gt;128&lt;/sup&gt;</td>
<td>2&lt;sup&gt;-23&lt;/sup&gt;</td>
</tr>
<tr>
<td>40</td>
<td>31</td>
<td>8</td>
<td>2&lt;sup&gt;-126&lt;/sup&gt;</td>
<td>2&lt;sup&gt;128&lt;/sup&gt;</td>
<td>2&lt;sup&gt;-31&lt;/sup&gt;</td>
</tr>
<tr>
<td>96</td>
<td>63</td>
<td>32</td>
<td>2&lt;sup&gt;-2147483646&lt;/sup&gt;</td>
<td>2&lt;sup&gt;2147483648&lt;/sup&gt;</td>
<td>2&lt;sup&gt;-63&lt;/sup&gt;</td>
</tr>
<tr>
<td>f + e + 1</td>
<td>f</td>
<td>e</td>
<td>2&lt;sup&gt;(-2&lt;sup&gt;e-1&lt;/sup&gt;)-2&lt;/sup&gt;</td>
<td>2&lt;sup&gt;-2&lt;sup&gt;e-1&lt;/sup&gt;&lt;/sup&gt;</td>
<td>2&lt;sup&gt;-f&lt;/sup&gt;</td>
</tr>
</tbody>
</table>
It is not possible to do the total calculation of the polynomials and extrapolation with fixed numbers (integer). In the calculation of the polynomials the time is taken to the power of four. This number is too large to represent in a fixed number. To prevent this overflow the calculations are using floating point numbers. There are 2 float data type which can be used for the polynomial-fit calculation. Namely 96.40/32 bit float. The advantage of the 96 bit float is the accuracy of the calculation. However the calculation time will be much higher. The calculation needed to be fast and accurate. However by using a the emulator the accuracy of the calculation improves but the calculation time become worse. There must be made a compromise between speed and the accuracy of the calculation. There are two different DSP programs written. One with a 40 bit float and the other with the 96 bit emulator. In the table in section 3 shows the calculation time of different types of floats.
3 Experiments

The first step was to get the exact time \( t_{ext} \) used for the extrapolation (equation 4). \( t_{ext} \) is the time at which the high resolution encoder is read by the QAD. This time is depending on the calculation time of the DSP. This value is gained by experiments. A simple experiment with applying a constant velocity is used. If the time \( t_{ext} \) is made to high the value of the extrapolation is to high. If value of the time is to low the value of the extrapolation is to low. It’s recommended to do this test with a high speed so the error made by a wrong time \( t_{ext} \) is high. The following table gives for different numbers of time stamps the estimated calculation time.

In the previous sections is mentioned that two different look-up tables are applied. The advantage of the filtered look up table that it’s possible to show the influence of low frequency error. There are several test performed to test the functions of the look up table. In [1] the results of the different look up tables are shown.

The next step was to get the limits of the system. Due to the maximum number which the DSP can represent by a 32 bit float a minimum speed is acquired to let the system properly work. This minimum speed is depending on the number of time stamps. To get a uniform representation it’s better to represent the speed as pulses/sec. (Note time stamps have a resolution of 100 ns.)

<table>
<thead>
<tr>
<th>Float</th>
<th>time stamps</th>
<th>min slits per second</th>
<th>estimated calculation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-40</td>
<td>4</td>
<td>±200</td>
<td>±250μs</td>
</tr>
<tr>
<td>32-40</td>
<td>6</td>
<td>±300</td>
<td>±400μs</td>
</tr>
<tr>
<td>32-40</td>
<td>8</td>
<td>±600</td>
<td>±510μs</td>
</tr>
<tr>
<td>32-40</td>
<td>10</td>
<td>±1000</td>
<td>±620μs</td>
</tr>
<tr>
<td>96 emu</td>
<td>6</td>
<td>-</td>
<td>±16000μs</td>
</tr>
</tbody>
</table>

With a 96 bits floating point calculation, the minimum slits per second can be very low. So for applications with very low velocity profiles it could be better to use the 96 bit floating point. At low velocity is the fault made by a very large calculation time smaller. However probably at low speed it’s better to reduce the accuracy of the time stamp generator.

In the next sections the results of the experiments are shown. The experiments were run for a nominal speed. The frequency of the disturbance used in the input is varied. The magnitude of the input disturbance was 10% of the nominal speed.
3.1 4 time-stamps polynomial fit

3.1.1 1 revolution per second and 1 Hz disturbance

Figure 5: Error using a 4 time-stamps polynomial fit

Figure 6: Speed profile using a 4 time-stamps polynomial fit
3.1.2 1 revolution per second with 10 Hz disturbance

Figure 7: Error using a 4 time-stamps polynomial fit

Figure 8: Speed profile using a 4 time-stamps polynomial fit
3.1.3 1 revolution per second with 20 Hz disturbance

Figure 9: Error using a 4 time-stamps polynomial fit

Figure 10: Speed profile using a 4 time-stamps polynomial fit
3.1.4  1 revolution per second with 40 Hz disturbance

Figure 11: Error using a 4 time-stamps polynomial fit

Figure 12: Speed profile using a 4 time-stamps polynomial fit
3.1.5 1 revolution per second with 60 Hz disturbance

Figure 13: Error using a 4 time-stamps polynomial fit

Figure 14: Speed profile using a 4 time-stamps polynomial fit
3.2 6 time stamps

3.2.1 1 revolution per second with 1 Hz disturbance

Figure 15: Error using a 6 time-stamps polynomial fit

Figure 16: Speed profile using a 6 time-stamps polynomial fit
3.2.2 1 revolution per second with 10 Hz disturbance

Figure 17: Error using a 6 time-stamps polynomial fit

Figure 18: Speed profile using a 6 time-stamps polynomial fit
3.2.3 1 revolution per second with 20 Hz disturbance

Figure 19: Error using a 6 time-stamps polynomial fit

Figure 20: Speed profile using a 6 time-stamps polynomial fit
3.2.4 1 revolution per second with 40 Hz disturbance

Figure 21: Error using a 6 time-stamps polynomial fit

Figure 22: Speed profile using a 6 time-stamps polynomial fit
3.2.5 1 revolution per second with 60 Hz disturbance

Figure 23: Error using a 6 time-stamps polynomial fit

Figure 24: Speed profile using a 6 time-stamps polynomial fit
3.2.6 2 revolutions per second with 1 Hz disturbance

Figure 25: Error using a 6 time-stamps polynomial fit

Figure 26: Speed profile using a 6 time-stamps polynomial fit
3.2.7 2 revolutions per second with 10 Hz disturbance

Figure 27: Error using a 6 time-stamps polynomial fit

Figure 28: Speed profile using a 6 time-stamps polynomial fit
3.2.8 2 revolutions per second with 20 Hz disturbance

Figure 29: Error using a 5 time-stamps polynomial fit

Figure 30: Speed profile using a 6 time-stamps polynomial fit
3.2.9 2 revolutions per second with 40 Hz disturbance

Figure 31: Error using a 6 time-stamps polynomial fit

Figure 32: Speed profile using a 6 time-stamps polynomial fit
3.2.10  2 revolutions per second with 60 Hz disturbance

Figure 33: Error using a 6 time-stamps polynomial fit

Figure 34: Speed profile using a 6 time-stamps polynomial fit
3.2.11 8 revolutions per second with 1 Hz disturbance

Figure 35: Error using a 6 time-stamps polynomial fit

Figure 36: Speed profile using a 6 time-stamps polynomial fit
3.2.12 8 revolutions per second with 10 Hz disturbance

![Figure 37: Error using a 6 time-stamps polynomial fit](image1)

![Figure 38: Speed profile using a 6 time-stamps polynomial fit](image2)
3.2.13 8 revolutions per second with 20 Hz disturbance

Figure 39: Error using a 6 time-stamps polynomial fit

Figure 40: Speed profile using a 6 time-stamps polynomial fit
3.2.14 8 revolutions per second with 40 Hz disturbance

Figure 41: Error using a 6 time-stamps polynomial fit

Figure 42: Speed profile using a 6 time-stamps polynomial fit
3.2.15 8 revolutions per second with 60 Hz disturbance

Figure 43: Error using a 6 time-stamps polynomial fit

Figure 44: Speed profile using a 6 time-stamps polynomial fit
3.3 8 time stamps

3.3.1 2 revolutions per second with 1 Hz disturbance

![Figure 45: Error using a 8 time-stamps polynomial fit](image1)

![Figure 46: Speed profile using a 8 time-stamps polynomial fit](image2)
3.3.2 2 revolution per second with 10 Hz disturbance

Figure 47: Error using a 8 time-stamps polynomial fit

Figure 48: Speed profile using a 8 time-stamps polynomial fit
3.3.3 2 revolution per second with 20 Hz disturbance

Figure 49: Error using a 8 time-stamps polynomial fit

Figure 50: Speed profile using a 8 time-stamps polynomial fit
3.3.4 2 revolution per second with 40 Hz disturbance

Figure 51: Error using a 8 time-stamps polynomial fit

Figure 52: Speed profile using a 8 time-stamps polynomial fit
3.3.5 2 revolution per second with 60 Hz disturbance

Figure 53: Error using a 8 time-stamps polynomial fit

Figure 54: Speed profile using a 8 time-stamps polynomial fit
4 conclusions

Examine the experiments it's obvious that the accuracy of the encoder is improved. The accuracy was improved by a factor 5. The result is better with higher nominal velocities because the difference between the time stamps are smaller. A big advantage of second order polynomial fitting is that the velocity can also be evaluated. The result of the velocity estimations were very promising. The estimated velocities are much smoother than the produced velocities by differentiating and filtering. This can be a great advantage to choose for a system with polynomial fitting.

The performance of the system depends on a lot of parameters which can be changed. To optimize the system these parameters must be tuned depending on the application.

There must be always made a compromise between the number of time stamps and the duration of the calculation. The system can be work better with more time-stamps but the calculation time will introduce larger errors. The performance of the polynomial fitting was improved by an active compensating of the encoder imperfections. The active compensating is done by measuring the encoder imperfections of the low resolution encoder with a higher resolution encoder. From these measurements a so called look-up table is generated. There were two look-up tables generated to examine the influence of different encoder imperfections.

Numerical problems were also an issue. Polynomial fitting through time-stamps requires a calculation with large numbers. Therefore the use of floating point data types is required. The currently used DSP support maximum a 40 bit float data type. However there is a 96 bit float emulator supported by Texas Instruments. The disadvantage of the 96 bit float emulator is that the calculation time is at least 3 times higher. The calculation time will be too long for most of the applications. It's recommended to use the emulator only for applications with very low velocities. However for low velocities probably it's better to reduce the resolution of the time stamp generator.

5 recommendations

There are several design questions still not answered. There are still parameters which can be changed. Most of the parameters will be depending on the application where the encoder is mounted. Some parameters must still be optimized.

To prevent overflow of the DSP there can be chosen to use a DSP with 64 bit
floating point or emulator. However it's also possible to reduce the resolution of the time stamps.
To increase the sampling frequency for a controller it's possible to use the calculated polynomials more than one time.
To reduce the influences of the high frequency error it can be better to use only one channel edge of the encoder. This can be accomplished by just take every fourth time sample. The

5.1 Ways to measure the look up table

Another topic which still must be investigated is how to generate the look up table in a practical situation. The main problem is that if a encoder is re-mounted on an axis the error will change. The encoder must be mounted on the axis before the look up table can be measured. There are several options to make a look-up table. One method is used in this report. Just take an higher resolution encoder and measure the difference. An other option is to let rotate the encoder with a constant speed and use the time information. To ensure a constant speed, a another encoder must be used or a set-up with a big fly wheel.
A last option is to redesign the physical properties of the encoder. A special encoder with an extra sensor which is 180 degrees rotated. In figure 55 a schematic overview of this concept is given. The error will be probably in

![Figure 55: redesign incremental encoder](image)

opposite phase in respect to the other sensor(s). If the setup is controlled by the information of one sensor the lookup table can be measured with the
other sensor. The disadvantage of this concept is the cost to produce such an encoder. The cost will be probably higher than a encoder with a higher resolution. It will be depending on the cost of the disc in respect to extra sensors and a different housing. But the advantage of this method is that the look up table can be gained when the encoder already is mounted into the system.
References

[1] C. Vottis
Extracting more accurate position and velocity information from optical incremental encoders
Eindverslagen Stan Ackermans Insituut 2003/069
ISBN 90-444-0335-4
Eindhoven 2003

Extracting a more accurate position from a quadrature signal using the SRI
DCT-nr: 2002.40
Eindhoven 2002