A Derivative Estimation Toolbox based on Algebraic Methods – Theory and Practice

Josef Zehetner, Johann Reger and Martin Horn

***Abstract*— In this paper, the implementation and usage of a so-called Derivative Estimation Toolbox is demonstrated. By means of this toolbox, time derivatives of sampled, noisy time signals may be determined in realtime, all based on a recently presented algebraic derivative estimation method. The main contribution is a possible implementation on a prototyping control unit. The performance of the Derivative Estimation Toolbox is experimentally validated on a brake-testbench. In particular, the friction force and the drive torque are estimated in realtime.**

I. Introduction

The fast and robust estimation of derivatives of measured (or sampled) time signals is a very important issue in a variety of applications in control engineering, failure diagnostics, signal processing and transmission. Due to the fact that, in practice, signals are affected by measurement noise, filtering is inevitable. A common approach is based on linear time-invariant filters. More sophisticated approaches [1] are for example the Savitzky-Golay differentiation scheme [2], averaged finite difference methods [3] and wavelet differentiation schemes [4]. In [5]–[7] some of these approaches are used for state estimation of nonlinear systems.

A novel method on derivative estimation that is based on Laurent Schwarz' calculus of distributions and operational calculus, respectively, was introduced by Fliess and Sira-Ramírez [8], recently. Preliminary results of the so-called *algebraic derivative estimation* were presented at three international summer schools: the *Workshop on Identification, State Reconstruction, and Generalized PI-Control*, July 4–8, 2005 in Munich, the summer school *Fast Estimation Methods in Automatic Control and Signal Processing*, July 18–20, 2005 in Paris, and the summer school *Fast Estimation and Identification Methods in Control and Signal*, September 11–15, 2006, in Grenoble, and published in [9], [10]. Based on this new method, in [11] a realtime implementation was presented. This so-called *Derivative Estimation Toolbox* (DET) offers the possibility to use the presented derivative estimation technique in control and signal-processing applications in realtime for the first time.

The main contribution of this work is to show how the *DET* may beneficially be used for the estimation of the friction force and the drive torque on a laboratory brake-testbench. This will be done in realtime using a prototyping control unit.

The paper is organized as follows: in Section II the fundamentals of the derivative estimation method are recalled. A brief overview of a possible realtime implementation, the *DET*, is given in Section III. In Section IV we show experimental results of a friction force and drive torque estimation on a brake-testbench, based on the *DET*. Finally, Section V concludes the presented work and outlines future activities.

II. Algebraic Derivative Estimation based on a Receding Horizon Strategy

The basics of the receding horizon approach to algebraic derivative estimation shall be recalled from [9], [8] and [12]. An alternative approach was taken in [10], [13] where time-varying linear filters were used.

Consider a real-valued, analytic function of time, $y(t)$, which for time instants $t > 0$ shall be approximated by its Taylor-series expansion

$$y(t) \approx y_N(t) = \sum_{i=0}^{N} \frac{y^{(i)}(0)}{i!}\ t^i \tag{1}$$

of order N.

For simplicity of notation, we set

$$a_i := y^{(i)}(0),\ i = 0, 1, \ldots, N\,. \tag{2}$$

Referring to elementary operational calculus we transform $y_N(t) \ \circ\!\!-\!\!\bullet\ Y_N(s)$ and obtain

$$Y_N(s) = \sum_{i=0}^{N} \frac{a_i}{s^{i+1}}\,. \tag{3}$$

It is the key idea of the algebraic derivative estimation method to equivalently modify the above equation by a left-multiplication by an appropriate operator that helps isolate the j-th coefficient $a_j = y^{(j)}(0)$, $j = 0, 1, 2, \ldots, N$.

To this end, Equation (3) is premultiplied by the operator s^{N+1}, thus

$$s^{N+1}\, Y_N(s) = \sum_{i=0}^{N} a_i\, s^{N-i}\,, \tag{4}$$

which may be differentiated $N - j$ times with respect to the operator s. In doing so, we have eliminated all coefficients $a_{j+1}, \ldots, a_N$, hence

$$\frac{d^{N-j}}{ds^{N-j}}\left(s^{N+1} Y_N(s)\right) = \sum_{i=0}^{j} a_i\, \frac{(N-i)!}{(j-i)!}\, s^{j-i}\,. \tag{5}$$

Josef Zehetner is with the Institute of Automation and Control, Graz University of Technology, 8010 Graz, Austria and with Magna Steyr, 8041 Graz, Austria josef.zehetner@tugraz.at

Johann Reger is with the EECS Control Laboratory at the University of Michigan, Ann Arbor, MI 48109-2122, USA reger@ieee.org

Martin Horn is with the Institute of Automation and Control, Graz University of Technology, 8010 Graz, Austria martin.horn@tugraz.at

This work was partially funded by the FIT-IT Project VehDynCtrl 809186 granted by bmvit.

The remaining coefficients $a_0, \ldots, a_{j-1}$ may be canceled, as well. To this end, we premultiply by the operator $1/s$

$$\frac{1}{s}\frac{d^{N-j}}{ds^{N-j}}\left(s^{N+1}Y_N(s)\right)=\frac{(N-j)!}{s}a_j+\sum_{i=0}^{j-1}a_i\frac{(N-i)!}{(j-i)!}s^{j-i-1} \tag{6}$$

and differentiate the latter equation j times with respect to the operator s in order to cancel the sum on the right hand side. This yields

$$\frac{d^j}{ds^j}\left(\frac{1}{s}\frac{d^{N-j}}{ds^{N-j}}\left(s^{N+1}Y_N(s)\right)\right)=\frac{(-1)^j\, j!\,(N-j)!}{s^{j+1}}a_j\,. \tag{7}$$

The expression on the left hand side of (7) contains s^N as operator monomial of maximal degree, which is equivalent to an N-fold derivation with respect to time. Thus, we premultiply the entire equation by $1/s^{N+\nu+1}$ with the consequence that any time signal is at least integrated once[1]. Therefore, we obtain

$$\frac{1}{s^{N+\nu+1}}\frac{d^j}{ds^j}\left(\frac{1}{s}\frac{d^{N-j}}{ds^{N-j}}\left(s^{N+1}Y_N(s)\right)\right)=\frac{(-1)^j j!\,(N-j)!}{s^{N+j+\nu+2}}\,a_j \tag{8}$$

which may be transferred back to the time domain employing Leibniz' formula for the differentiation of products. In light of this, we process the following steps:

$$\begin{aligned}
&\tfrac{1}{s^{N+\nu+1}}\left(\tfrac{d^j}{ds^j}\left(\tfrac{1}{s}\tfrac{d^{N-j}}{ds^{N-j}}\left(s^{N+1}\,Y_N(s)\right)\right)\right)=\\
&=\tfrac{1}{s^{N+\nu+1}}\left(\tfrac{d^j}{ds^j}\left(\sum_{\kappa_1=0}^{N-j}\tbinom{N-j}{\kappa_1}\tfrac{(N+1)!}{(N-\kappa_1+1)!}s^{N-\kappa_1}\tfrac{d^{N-j-\kappa_1}Y_N(s)}{ds^{N-j-\kappa_1}}\right)\right)\\
&=\tfrac{1}{s^{N+\nu+1}}\left(\sum_{\kappa_1=0}^{N-j}\tbinom{N-j}{\kappa_1}\tfrac{(N+1)!}{(N-\kappa_1+1)!}\tfrac{d^j}{ds^j}\left(s^{N-\kappa_1}\tfrac{d^{N-j-\kappa_1}Y_N(s)}{ds^{N-j-\kappa_1}}\right)\right)\\
&=\tfrac{1}{s^{N+\nu+1}}\sum_{\kappa_1=0}^{N-j}\sum_{\kappa_2=0}^{j}\tbinom{N-j}{\kappa_1}\tbinom{j}{\kappa_2}\tfrac{(N+1)!\,(N-\kappa_1)!}{(N-\kappa_1+1)!\,(N-\kappa_1-\kappa_2)!}\times\\
&\quad s^{N-\kappa_1-\kappa_2}\tfrac{d^{N-\kappa_1-\kappa_2}Y_N(s)}{ds^{N-\kappa_1-\kappa_2}}=\sum_{\kappa_1=0}^{N-j}\sum_{\kappa_2=0}^{j}\tbinom{N-j}{\kappa_1}\tbinom{j}{\kappa_2}\times\\
&\quad\tfrac{(N+1)!}{(N-\kappa_1-\kappa_2)!\,(N-\kappa_1+1)}\tfrac{1}{s^{\nu+\kappa_1+\kappa_2+1}}\tfrac{d^{N-\kappa_1-\kappa_2}Y_N(s)}{ds^{N-\kappa_1-\kappa_2}}\,.
\end{aligned}$$

Due to (8) we conclude that

$$\begin{aligned}
&\tfrac{1}{s^{N+j+\nu+2}}\,a_j=\tfrac{(-1)^j}{j!\,(N-j)!}\sum_{\kappa_1=0}^{N-j}\sum_{\kappa_2=0}^{j}\tbinom{N-j}{\kappa_1}\tbinom{j}{\kappa_2}\times\\
&\quad\tfrac{(N+1)!}{(N-\kappa_1-\kappa_2)!\,(N-\kappa_1+1)}\tfrac{1}{s^{\nu+\kappa_1+\kappa_2+1}}\tfrac{d^{N-\kappa_1-\kappa_2}Y_N(s)}{ds^{N-\kappa_1-\kappa_2}}\,.
\end{aligned} \tag{9}$$

There is no difficulty now to carry out the backward transform to the time domain: The left hand side represents a polynomial in the time domain, the right hand side expression of s matches a respective convolutional integral; hence

$$a_j=\int_0^t \Pi_{jN\nu}(t,\tau)\,y_N(\tau)\,d\tau \tag{10}$$

with

$$\begin{aligned}
&\Pi_{jN\nu}(t,\tau)=\tfrac{(N+j+\nu+1)!\,(N+1)!\,(-1)^j}{t^{N+j+\nu+1}}\sum_{\kappa_1=0}^{N-j}\sum_{\kappa_2=0}^{j}\\
&\tfrac{(t-\tau)^{\nu+\kappa_1+\kappa_2}\,(-\tau)^{N-\kappa_1-\kappa_2}}{\kappa_1!\,\kappa_2!\,(N-j-\kappa_1)!\,(j-\kappa_2)!\,(N-\kappa_1-\kappa_2)!\,(\nu+\kappa_1+\kappa_2)!\,(N-\kappa_1+1)}\,.
\end{aligned} \tag{11}$$

[1]Instead of multiple integration with respect to time arbitrary filters of respective order may be used.

In this formula, instead of $y_N(t)$ we may resort naturally to the measured (noisy) signal $y(t)$. Furthermore, an arbitrary small time window T may be used for the determination of a_j. In course of time, the Taylor-series expansion (1) becomes inaccurate since it holds only in the vicinity of $t = 0$. On the contrary, usually one is interested in the derivative estimates at a time instant $t \gg 0$. Therefore, we have to adapt the method for arbitrary time instants. For this purpose, we flip the values of y, i. e. integrate backwards in time, so as to expand at the time instant of interest, that is t. Finally, we obtain

$$y^{(j)}(t)=(-1)^j\int_0^T \Pi_{jN\nu}(T,\tau)\,y(t-\tau)\,d\tau. \tag{12}$$

The interval of integration with length T can be interpreted as the window width of a receding horizon strategy. The window width T should be chosen small so as to calculate the derivative estimate within an acceptable short delay, it has to be chosen large enough in order to sustain the low pass filtering property for suppressing measurement noises on $y(t)$ (adjustable with ν) which improves the quality of the estimate.

In Section IV, the following polynomials are used for estimating the derivatives in a realtime application:

$N = 1$, $j = 0$, $\nu = 0$:

$$y^{(0)}(t)=\int_0^T \frac{-6\tau+4T}{T^2}y(t-\tau)\,d\tau \tag{13}$$

$N = 1$, $j = 1$, $\nu = 0$:

$$y^{(1)}(t)=\int_0^T \frac{-12\tau+6T}{T^3}y(t-\tau)\,d\tau \tag{14}$$

$N = 2$, $j = 1$, $\nu = 0$:

$$y^{(1)}(t)=\int_0^T \frac{180\tau^2-192\tau T+36T^2}{T^4}y(t-\tau)\,d\tau \tag{15}$$

The value of the integral within (12) and (13), (14), (15), respectively, may be approximated by a trapezoidal numerical integration. For discrete time values of $y_k = y(kT_s)$ and $y_k^{(j)} = y^{(j)}(kT_s)$ we obtain the approximation

$$y_k^{(j)}\approx(-1)^j\,\frac{T_s}{2}\sum_{i=1}^{M}(\Pi_{i-1}\,y_{k-i+1}+\Pi_i\,y_{k-i}), \tag{16}$$

where $t_{k-i} = (k-i)\,T_s$ and $t_i = i\,T_s$, $y_{k-i} = y(t_{k-i})$ and $\Pi_i = \Pi_{jN\nu}(T, t_i)$. T_s is the sample time and M denotes the number of summation steps, it holds $M = T/T_s$.

III. DERIVATIVE ESTIMATION TOOLBOX

The ideas presented in Section II can be used for *offline* analysis, i.e. after a measurement has been finished. For this purpose, a vast literature on different filter algorithms is available. More challenging is the *online* calculation of derivatives, in particular, in a realtime setting during measurement for control tasks. Based on (16) a toolbox for Simulink from Mathworks was developed. The main idea is to solve the *moving integral* for every simulation step using a so-called *S-function* [14]. The S-function is written in C. Therefore it can be used for both simulation and realtime applications. A detailed overview of the implementation and first results are presented in [11].

A. Data structure

The S-function has to be parameterized using the following parameters:

- j: order of derivative for the input signal
- T: interval of integration in [sec]
- N: order of the Taylor-series expansion
- ν: number of additional integrals
- T_s: sample time in [sec]

The $M = T/T_s$ last sample values of the time *window* $(t-T, t]$ are used to calculate the actual estimated derivative $y_k^{(j)}$ at time t. This window is *moving* at every time step by adding the actual sample value y_k and removing the oldest value y_{k-M}. For this purpose the data can be stored in a so called *ring-buffer* in a very efficient manner. A ring-buffer is a linear array of size M. A circular memory-structure is created by using *modulo*-operations[2] for every read- and write-access.

B. Code structure

The flow diagram of the S-function which has to be evaluated for every time step is shown in Figure 1. The critical section concerning the computing time of the S-function is the loop calculating the integral. In Figure 2 the pseudo-code representation is shown. The ring-buffer of the size `RBSZ` is stored in the array `RB`. The sample value y_k at time t is written to the position `RBPOS` in `RB`. The user-parameters `PARAM` are fetched and checked at every time step (line `2`), i.e. they can be adjusted online without resetting the application. Lines `5` to `11` form the loop to calculate the integral. In `6` and `7` the sample values y_{k-i} and y_{k-i+1} are read from `RB`. In line `8` and `9` the function `derivate` evaluates the appropriate equation for `PARAM` using `i` and `RB1`, `RB2`, respectively. In `10` the trapezoidal numerical integration is carried out and summed up.

C. Remarks

Some interesting features of the presented code structure concerning realtime applications should be mentioned. Multiple time derivatives (of different order) from one signal may be calculated using the same S-function. The overhead generated by the `for`-loop is produced only once for all results. In the same manner, the time derivatives of multiple input signals may be calculated once, e.g. the first derivative of four wheel-speed-sensor signals.

On realtime systems only a limited number of calculations per time step are possible, i.e. for higher sample rates the integration interval T has to be chosen rather small. In this case the interval can be widened in two different ways to improve the suppression of noise. On the one hand, only every L-th $(L \geq 2)$ sample value may be taken into account, i.e. for a constant number of summation steps M, the integration interval T is multiplied by L. On the other hand, the number of summation steps M can be partitioned into P time steps, i.e. the number of summations per time step is reduced to $\tilde{M} = M/P$.

[2]The *modulo*-operation gives the remainder of the division of two integer numbers, e.g. $p = 34\, modulo\, 20 = 14$.

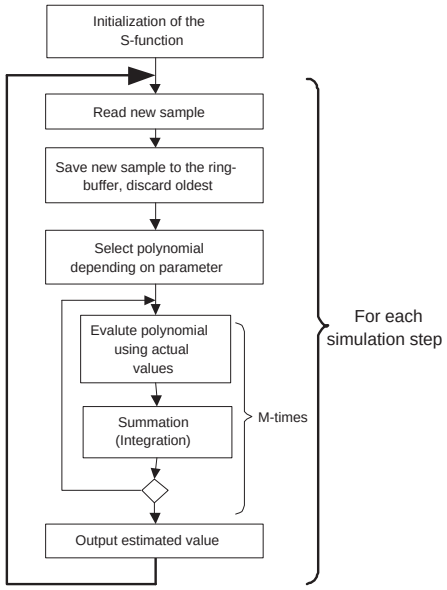

Fig. 1. Flow diagram of the S-Function

```
 1: get SAMPLE
 2: get and check PARAM
 3: increment RBPOS
 4: save SAMPLE to RB
 5: for i = 1 to RBSZ
 6:   RB1 = RB((RBPOS-i+1) modulo RBSZ)
 7:   RB2 = RB((RBPOS-i) modulo RBSZ)
 8:   TRAP1 = derivate(RB1,i-1,PARAM)
 9:   TRAP2 = derivate(RB2,i,PARAM)
10:   OUTPUT ++= trap(TRAP1,TRAP2)
11: endfor
12: write OUTPUT
```

Fig. 2. Pseudocode

IV. APPLICATIONS

The *DET* is used for estimating time derivatives on a brake-testbench. A detailed description of the testbench is

given in [15]. For the sake of completeness a short explanation of the testbench is given here.

A. Brake-testbench

The testbench is located at the *Institute of Automation and Control* at Graz University of Technology, Austria. It is used to implement and verify control strategies for antiblock and anti-spin systems for passenger cars. The testbench consists of two main parts, a wheel with a rubber tire and a bearing-supported solid steel roll. A schematic is given in Figure 3. The wheel is connected to an electric motor and to

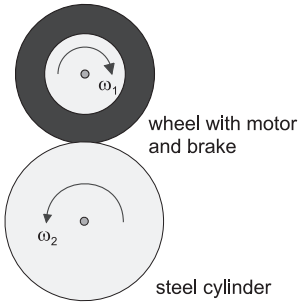

Fig. 3. Schematic of the brake-testbench

a conventional automotive brake system. By accelerating or braking the wheel the roll is accelerated and braked, respectively, due to the friction forces between the rubber tire and the steel roll.

Index "1" marks variables associated to the wheel, index "2" denotes variables with respect to the roll. The equations of motion for both parts of the brake-testbench are given as

$$J_1 \frac{d\omega_1}{dt} = T_d - T_b - T_{f,1} + Fr_1 \tag{17}$$

for the wheel and

$$J_2 \frac{d\omega_2}{dt} = -T_{f,2} - Fr_2 \tag{18}$$

for the roll. The moments of inertia with respect to the axes of rotation are given as J_1 and J_2, ω_1 and ω_2 are the angular velocities. The drive torque generated by the motor is denoted by T_d and the brake torque applied is T_b. $T_{f,1}$ and $T_{f,2}$ represent respective friction torques. F is the longitudinal friction force and r_1, r_2 are the radii of the wheel and the roll, respectively. The differential equations (17) and (18) are coupled via the friction force F which is a nonlinear function of the angular velocities ω_1 and ω_2, hence, renders the entire dynamic system nonlinear.

The angular velocities ω_1 and ω_2 are measured with inductive wheel-speed sensors. The parameters J_1, J_2, $T_{f,1}$, $T_{f,2}$, r_1 and r_2 are supposed to be known. The estimation of these parameters can be found in [15].

The brake-testbench is equipped with a *MicroAutoBox DS 1401/1501* (MABX) from *dSpace*. This is a prototyping control unit specially designed for automotive applications. The MABX is based an a PowerPC 750 FX processor and provides numerous I/O-ports, e.g. two CAN controllers, four

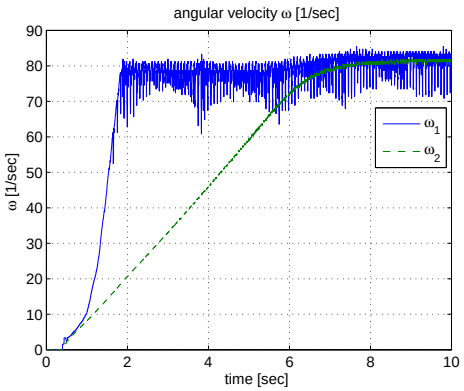

Fig. 4. Acceleration of wheel without anti-spin control

input channels for frequency measurement or 16 analogue and up to 32 digital inputs. A target compiler for Simulink is available. Therefore, applications for the MABX may be built directly from Matlab/Simulink. A detailed description can be found in [16].

A target wheel speed can be set to control the testbench. The angular velocities of the wheel and the roll are measured. The measured signals and the results from the *DET* are transferred to a PC via a highspeed serial interface in realtime.

For the following experiments a simple test cycle was generated. Therefore a target angular velocity of $\omega_1 = 80\,[1/\sec]$ is set. The anti-spin control is disabled, i.e. the wheel-slip can reach values close to 1. The measured angular velocities ω_1 and ω_2 for this maneuver are depicted in Figure 4.

The longitudinal friction force F is a function of the wheel-slip

$$\lambda := \frac{\omega_1 r_1 - \omega_2 r_2}{max(\omega_1 r_1, \omega_2 r_2)},$$

where the wheel-slip λ is calculated by estimating the zero-order-derivative for ω_1 and ω_2 using the *DET* as a zero-order-derivative filter. To this end, polynomial (13) is used with parameters $N = 1$, $j = 0$, $\nu = 0$. For the estimation of ω_1 an integration interval of $T = 0.2$ sec is used, concerning ω_2 the window size is reduced to $T = 0.1$ sec since there is less noise on the measurement signal for ω_2. In Figure 5 the algebraically estimated wheel-slip λ is compared to the wheel-slip calculated from the unfiltered signals.

B. Friction force estimator

Based on (18) a simple estimator for the friction force F can be derived as

$$F = \frac{1}{r_2}\left(-T_{f,2} - J_2 \frac{d\omega_2}{dt}\right). \tag{19}$$

Values for r_2, $T_{f,2}$ and J_2 are well known in our case. ω_2 is measured by an inductive wheel-speed sensor mounted to the roll axle. Due to measurement and quantization noise

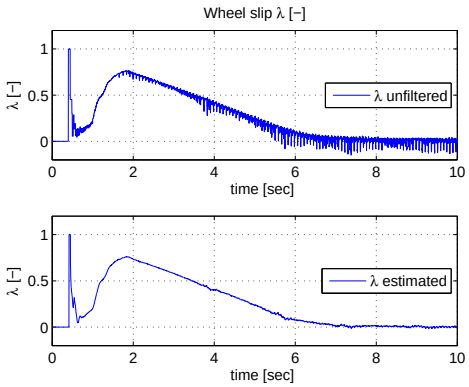

Fig. 5. Unfiltered wheel-slip λ vs. algebraically estimated wheel-slip λ

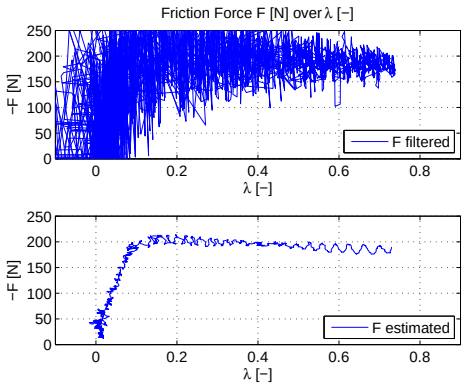

Fig. 7. Comparison of filtered and algebraically estimated F over λ

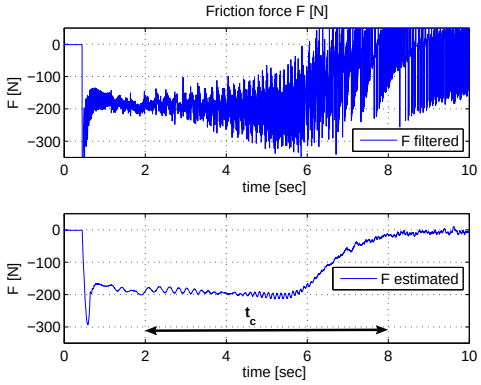

Fig. 6. Comparison of filtered and algebraically estimated F over time

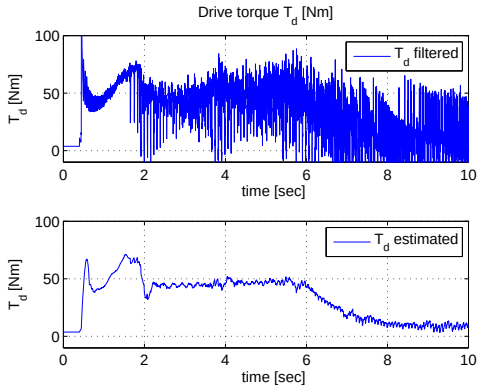

Fig. 8. Comparison of filtered and algebraically estimated drive torque T_d

the direct expression $d\omega_2/dt$ yields no acceptable result. As a consequence, the *DET* shall be used to calculate the first derivative with respect to time of ω_2 in *realtime*. Equation (14) is used for this estimation. The parameters are $j = 1$, $T = 0.2$ sec, $N = 1$, $\nu = 0$ and $T_s = 0.002$ sec.

For a comparison of the results obtained with the *DET* a filtered "dirty" first derivative of ω_2 with respect to time is calculated by the transfer function

$$P(s) = \frac{s}{0.05s + 1}. \tag{20}$$

Figure 6 shows the filtered friction force compared to the algebraically estimated friction force.

In Figure 7, the algebraically estimated and the filtered friction forces are plotted with respect to the estimated and the unfiltered wheel-slips. It has to be mentioned that only values from the time-interval t_c (see Figure 6) are used for this plots. Note that the friction forces are plotted with negative sign.

C. Drive torque estimator

Using (17) and the results from above an estimator for the drive torque T_d (the brake torque T_b) follows straight forward as

$$T_d - T_b = J_1 \frac{d\omega_1}{dt} + T_{f,1} - Fr_1. \tag{21}$$

In the case of acceleration, the brake torque T_b is assumed to be zero. Again the determination of $d\omega_1/dt$ is the crucial part. To this end, again, the first derivative of ω_1 with respect to time is algebraically estimated referring to Equation (15) with parameters $j = 1$, $T = 0.4$ sec, $N = 2$, $\nu = 0$ and $T_s = 0.002$ sec. Again a filtered dirty derivative $d\omega_1/dt$ is calculated with transfer function (20). In Figure 8 the results of the corresponding laboratory experiment are shown.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this work, the mathematical framework recalled in Section II is tested in a realworld *and* realtime environment. It was shown that the *Derivative Estimation Toolbox* offers the possibility to design simple and considerably robust estimators for realtime applications. The results indicate the potential of the method in an impressive way. The implementation on a prototyping electronic control unit closes the gap between theoretic brilliance and applied sciences.

B. Future Works

In light of the high quality of the results the *Derivative Estimation Toolbox* will be tested in more complex applications, e.g. in prototype vehicles. The mathematical theory still offers potential for optimization concerning exactness

and calculation effort. This will lead to even better results and a faster online calculation.

ACKNOWLEDGEMENTS

The authors would like to thank Cédric Join and Hebertt Sira-Ramírez for their invaluable discussions during the summer schools in Munich, Paris and Grenoble.

REFERENCES

[1] S. Diop, J. Grizzle, and F. Chaplais, "On numerical differentiation algorithms for nonlinear estimation," in *Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 2, 2000, pp. 1133–1138.

[2] W. Gander and U. von Matt, "Smoothing filters," in *Solving Problems in Scientific Computating Using Maple and Matlab*, 3rd ed., W. Gander and J. Hrebicek, Eds. Berlin, Germany: Springer, 1997, pp. 121–139.

[3] R. Anderssen, F. de Hoog, and M. Hegland, "A stable finite difference ansatz for higher order differentiation of non-exact data," *Bull. Austral. Math. Soc.*, vol. 58, pp. 223–232, 1998.

[4] C. Burrus, R. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms. A Primer*. New Jersey: Prentice-Hall, 1998.

[5] S. Diop, V. Fromion, and J. Grizzle, "A global exponential observer based on numerical differentiation," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 4, 2001, pp. 3344–3349.

[6] S. Diop, J. Grizzle, P. Moraal, and A. Stefanopoulou, "Interpolation and numerical differentiation for observer design," in *American Control Conference*, vol. 2, 1994, pp. 1329–1333.

[7] F. Plestan and J. Grizzle, "Synthesis of nonlinear observers via structural analysis and numerical differentiation," in *Proceedings of the 5th European Control Conference*, Sept. 1999.

[8] M. Fliess and H. J. Sira-Ramírez, "Control via state-estimation of some nonlinear systems," in *IFAC Symp. on Nonlinear Control Systems (NOLCOS)*, Stuttgart, Germany, 2004.

[9] M. Fliess, C. Join, M. Mboup, and H. Sira-Ramírez, "Analyse et représentation de signaux transitoires: application à la compression, au débruitage, et à la détection de ruptures," in *GRETSI 2005*, Louvain-la-Neuve, Belgium, 2005.

[10] J. Reger, H. J. Sira-Ramírez, and M. Fliess, "On non-asymptotic observation of nonlinear systems," in *IEEE Int. Conf. on Decision and Control (CDC)*, Seville, Spain, 2005.

[11] J. Zehetner, J. Reger, and M. Horn, "Echtzeitimplementierung einer Ableitungsschätzung mit Methoden der differentiellen Algebra," *at - Automatisierungstechnik*, submitted for publication.

[12] J. Jouffroy and J. Reger, "An algebraic perspective to single-transponder underwater navigation," in *IEEE Conf. on Control Applications (CCA)*, Munich, Germany, 2006.

[13] J. Reger, M. Mai, and H. J. Sira-Ramírez, "Robust algebraic state estimation of chaotic systems," in *IEEE Conf. on Control Applications (CCA)*, Munich, Germany, 2006.

[14] *Simulink Help: Writing S-Functions*, The MathWorks Inc., 1994–2006.

[15] M. Horn and J. Zehetner, "A brake-testbench for research and education," in *Proc. IEEE Conference on Control Applications (CCA) 2007*, Singapore, submitted for publication.

[16] *MicroAutoBox Features, Release 5.1*, dSpace GmbH, Paderborn, Germany, May 2006.