A Forecasting Algorithm for Latency Compensation in Indirect Human-Computer Interactions

Rosane Ushirobira, Denis Efimov, Géry Casiez, Nicolas Roussel, Wilfrid Perruquetti

Abstract—Human-computer interactions are greatly affected by the latency between the human input and the system visual response. The compensation of this latency is an important problem for the HCI (human-computer interaction) community. This work, a simple forecasting algorithm is developed for latency compensation in indirect interaction using a mouse, based on numerical differentiation. Several differentiators are compared, including a novel algebraic version. An optimized procedure is developed for tuning the parameters of the algorithm. The efficiency is demonstrated on real data, measured with a 1 ms sampling time.

I. INTRODUCTION

Human interactions with computing systems are largely affected by the latency between human gestures and system visual responses [1]. This problem remains relatively unexplored and it is only quite recently that some works studied the latency in this context [2], [3]. For example, it has been recognized that a latency as low as 2 ms can be perceived by a human [2] (by itself, average end-to-end latency estimation for human-computer interactions is a complex problem [4]). It is on direct touch systems, such as smartphones, tablets and touch-screens, that latency is the most perceivable. Furthermore, the compensation of latency is of great importance for some kind of interactions, e.g. drag and drop tasks for tablet computers or drawing using any type of input devices, from computer mouse till touch-based devices (touchpads, optical finger navigation sensors, touch screens, etc.). The solutions for latency compensation can be realized on the hardware level (by using more reactive and performing components), and that may significantly increase the price of the computer system, or on the software level (by code optimization or design of special prediction algorithms), and that may be a less expensive solution for end-users.

An end-to-end latency around 50 ms is known to affect performance in mouse-based pointing tasks. MacKenzie and Ware have shown the degradation of performance is particularly important from 75 ms latency [1]. Pavlovych and Stuerzlinger did not notice any drop of performance up to 58 ms of latency when acquiring targets as small as 14 pixels wide [3] while Teather et al. measured a 15% decrease of performance between 35 and 75 ms end-to-end latency [5]. More recently Deber et al. found a perception threshold for latency in indirect touch pointing tasks around 55 ms [6]. A recent work on the measure of end-to-end latency on modern operating systems, with no application-specific code running, found an average latency ranging from 46 to 83 ms with jitter ranging from 5 to 25 ms, depending on the operating system and toolkit [7]. This means that latency, close or above the 50 ms threshold, is still present in high-end machines with up-to-date operating systems and toolkit, even in the best case scenario.

Prediction or forecasting deals with the value estimation of some variable of interest at a specified future instant of time [8]. Many methods devoted to forecasting exist, whose selection depends on constraints and on the context of the addressed problem (i.e. whether process statistics or models are available, or its periodicity, or the available computational power for a prediction algorithm realization, etc.). If it is the prediction of human movements that must be accomplished, then statistical methods cannot be applied due to the diversity of possible gestures. Models based on famous Fitts’s or Accot-Zhai steering laws [9], [10] allow pointing time prediction. However, even though target prediction models have been studied in some works (e.g. [11], [12], [13]), the design of a bio-mechanical prediction model with an additional requirement of being sufficiently quick for an online implementation remains a difficult problem.

This paper presents the development of a latency compensation algorithm for a particular case of mouse-based interaction. It is supposed that the signal from a mouse-based sensor is received with a lag, and its future implementation and processing will also introduce an additional delay before a system reaction will be sent back to human. It is then necessary to predict the output of the sensor in a future instant of time to compensate the overall latency in the system. Since the model of the human gesture itself is assumed unavailable and there is no statistics on a human practice, then some basic signal-based prediction methods have to be applied. In this work a time-series approach based on differentiation from [14], [15], [16], [17], [18], [19], [20], [21] is employed.

The outline of this work is as follows. The computer system equipped with mouse sensors and the problem statement are presented in Section II. Applied differentiation algorithms are introduced in Section III, including a new algebraic differentiator. Our latency compensation algorithm is proposed in Section IV. Discussion is given in Section V.
scheme given in Fig. 2 with this mouse and Libpointing\(^2\) software, we were able to simulate a part of operating system responsible for interactions with a mouse-based sensor, then integrate and test a forecasting algorithm on this level.

For a forecasting algorithm development, we have the following problem statement: the raw measurements 

\[ y(t_i) = x(t_i) + v(t_i) \]

at time instants \( t_i \) are available for \( i = 0, 1, \ldots, K \) (\( K > 0 \)), where \( x(t_i) \) corresponds to a coordinate \( x(t) \in \mathbb{R} \) provided by the sensor at \( t_i \), sampled irregularly with an average close to 1 ms and \( v(t_i) \) is a bounded noise of unknown nature. For a given lag \( L > 0 \), it is necessary to forecast the value \( \hat{x}(t_i + L) \) using previously measured data:

\[ \hat{x}(t_i + L) = F(y(t_i), \bar{t}), \]

where \( y(t_i) = [y(t_0), \ldots, y(t_i)] \), \( \bar{t} = [t_0, \ldots, t_j] \) and the map \( F(\cdot) \) represents the forecasting algorithm to be developed.

### iii. Numerical Differentiation Algorithms

Due to a rather general problem statement and the absence of a model for this human-computer interaction process, in this work we will use a simple Taylor series expansion for forecasting. If we would presume the human gesture as a smooth signal of time then

\[ x(t_i + L) = \sum_{j=0}^{+\infty} \frac{L!}{j!} x^{(j)}(t_i). \]

In practice, \( x(t) \) is not infinitely differentiable and it is difficult to calculate derivatives up to an arbitrary order, hence a constant \( M > 0 \) is selected such that

\[ \hat{x}(t_i + L) = \sum_{j=0}^{M} \frac{L!}{j!} \hat{x}^{(j)}(t_i), \]

(1)

where \( \hat{x}^{(j)}(t_i) \) are the estimates of \( x^{(j)}(t_i) \) calculated by a numerical differentiation algorithm. The error of Taylor series truncation has order \( L^{M+1} \) and the error of the most differentiators (the difference \( |x^{(j)}(t_i) - \hat{x}^{(j)}(t_i)| \)) is proportional to the amplitude of the noise \( v \) \((14), (15), (16), (18), (19), (21))\). Therefore, there exists \( M \) such that the forecasting algorithm (1) has a reasonable accuracy. In this work, we will apply and experimentally compare several numerical differentiators, their selection is motivated by possibilities of on-line implementation, required computational power, robustness with respect to noise and delay in the estimates.

### A. Algebraic Differentiator

The algebraic time derivative estimation is based on concepts of differential algebra and operational calculus. A more detailed description of the approach can be found in (19), (21), (17). A moving horizon version of this technique is summarized below adapted from the mentioned references. For a real-valued signal \( y(t) \), analytic on some time interval,
consider its approximating $N$th degree polynomial function, originated from its truncated Taylor expansion:

$$y(t) = \sum_{i=0}^{N} \frac{a_i}{i!} t^i,$$

where the terms $a_i$ are the unknown constant coefficients representing the derivatives of the signal. The aim is to estimate these time derivatives of $y(t)$ up to order $N$. Notice that in our case, we assume that $y(t)$ is the measured signal from a signal $x(t)$ with some negligible noise, so we may consider only $y(t)$. The errors from this assumption can be found for instance in [23]. This estimation can be done on a moving time horizon using integrals of $y(t)$. For any $T > 0$, the $j$th order time derivative estimate $\hat{y}^{(j)}(t)$, $j = 0, 1, 2, \cdots, N$, of the signal $y(t)$ defined in (2) satisfies the following convolution [19]:

$$\hat{y}^{(j)}(t) = \int_{0}^{T} H_j(T, \tau) y(t - \tau) d\tau, \quad j = 0, 1, \cdots, N,$$

for all $t \geq T$, where the convolution kernel,

$$H_j(T, \tau) = \frac{(N + j + 1)!(N + 1)!}{T^{N+j+1}} \times$$

$$\sum_{k_1=0}^{N-j} \sum_{k_2=0}^{N-j} \kappa_1 \kappa_2 \tau^{k_1} \tau^{k_2} \tau^{k_1+k_2}.$$

The kernel $H_j(T, \tau)$ depends on the order $j$ of the time derivative to be estimated and on an arbitrary constant length of time window $T > 0$. For example, if we are interested in calculating the first-order time derivative, a degree-one polynomial can be selected $y(t) = a_0 + a_1 t$. By applying the above result to this signal, we obtain the following first-order time derivative estimate

$$\hat{y}(t) = \frac{6}{T^3} \int_{0}^{T} (T - 2\tau) y(t - \tau) d\tau.$$  

The effect of the time integral, presented in equation (4), is obviously to dampen the impact of the measurement noise on the estimate. This noise dampening effect can also be used to filter out the noise from the original signal $\hat{y}(t)$. In that case, instead of calculating the first order time derivative, we have to estimate the zero order derivative (see [19] for details).

The differentiation error, caused by truncation in (2) and the integration over $T$, can be evaluated as a function of the second derivative amplitude.

**Theorem 1.** [23] If $\esssup_{t \geq 0} \{y(t)\} = \bar{y} < +\infty$, then

$$\esssup_{t \geq T} \{ y(t) - \hat{y}(t) \} \leq \bar{y} \frac{T}{T^2}.$$  

Since in our case only discrete observations are available, we need to discretize equation (4). Assume that the sampling is almost regular, i.e. $t_i = iT_s$ with $T_s$ is the sampling time, then for $y_k = y(kT_s)$ and $\hat{y}_k = \hat{y}(kT_s)$, the discretized approximation of equation (4) can be written as following:

$$\hat{y}_k = \frac{6}{m^2 T_s^2} \sum_{l=0}^{m} \left(1 - \frac{l}{m}\right) y_{k-l} \quad (5)$$

where $m > 0$ denotes the number of summation steps, i.e. $T = mT_s$.

**Remark 1.** One point to be noted here is the effect of delay on differentiation. This delay appears from the integration on the interval $[0, T]$ in equation (4). Again this integration helps to reduce the noise effect. Moreover, it was shown in [19] that delayed version of the differentiator gives better performance than the non-delayed version. So, in our case, we trade-off delay with robustness to noise and the quality of the differentiators. Finally, the algebraic differentiator (5) has only one tuning parameter $m$ (or $T$ in (4)).

Higher order time derivatives can also be computed in the same way using equation (3). However, in that case, higher order polynomials have to be considered. The new solution proposed in this paper is a consequent connection of the first differentiators (5). Since (5) is a discrete-time filter of order $m$, then to realize the second order differentiator, after self-product of the polynomial from (5), we obtain another discrete-time filter of order $2m + 1$ (in Matlab, the function conv can be used for this purpose). Repeating these procedure we can obtain filters for estimation of $j^{th}$ derivative of order $im + 1$. For $T_s = 0.001$, the amplitude Bode plots of these filters together with the corresponding plots of “ideal” differentiating filters $s^j |_{s^j = \frac{z_1 - z^{-1}}{z_m + 1}}$ are shown in Fig. 3. As we can conclude from these plots, below the frequency 10 Hz all algebraic differentiators up to order four are rather close to the “ideal” ones.

**B. Homogeneous finite-time differentiator**

Consider a sufficiently smooth signal $y(t)$. The aim of this differentiator is to estimate $\hat{y}(t)$, $\cdots$, $\hat{y}^{(n-1)}(t)$. Assume that $y^{(n)}(t) = \theta(t)$ and set $z = [y \ y \ \cdots \ y^{(n-1)}]^T$. Then,

$$\dot{z} = Az + \Theta(t), \quad y = Cz,$$  

with $A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$, $C = (1 \ 0 \ \cdots \ 0)$ and $\Theta(t) = (0 \ \cdots \ 0 \ \theta(t))^T \in \mathbb{R}^n$. The following homogeneous
finite-time differentiator can be proposed [18]:

\[ \dot{\hat{z}}_1 = \hat{z}_2 - k_1 [\hat{z}_1 - y]^\alpha, \]
\[ \dot{\hat{z}}_i = \hat{z}_{i+1} - k_i [\hat{z}_1 - y]^\alpha - (i-1), \quad i = 2, \ldots, n - 1 (6) \]
\[ \dot{\hat{z}}_n = -k_n [\hat{z}_1 - y]^\alpha - (n-1) \]

where \([x] = |x| \text{sign}(x)\) for all \(x \in \mathbb{R}\) and \(\gamma > 0\), \(\hat{z}\in\mathbb{R}^n\) is the estimate of \(z\), \(k_1, \ldots, k_n\) form a Hurwitz polynomial and a suitable choice of \((\alpha_1, \ldots, \alpha_n) \in \mathbb{R}_{>0}^n\) ensuring the system homogeneity. Then \(\hat{y}^{(j)}(t) = \hat{z}_j(t), \quad j = 1, \ldots, n\) after some finite time of transients. The differentiation error dynamics takes the form:

\[ \dot{e}_1 = e_2 - k_1 [e_1]^\alpha, \]
\[ \dot{e}_i = e_{i+1} - k_i [e_1]^\alpha - (i-1), \quad i = 2, \ldots, n - 1, (7) \]
\[ \dot{e}_n = \theta(t) - k_n [e_1]^\alpha - (n-1). \]

Due to the term \(\theta(t)\), it is impossible to get the convergence of the error to zero without having any additional knowledge about the signal \(\theta(t)\). This problem can be overcome by assuming that \(y(t)\) is locally polynomial and that on a small time interval, \(\theta(t) = 0\). In this case, it is possible to recover the time derivatives.

C. Higher order sliding mode and linear high-gain differentiators

A way to compensate exactly \(\theta(t)\) in the last equation of (7) is to select \(\alpha = 1 - \frac{1}{n}\). Then a homogeneous observer can be obtained, and in the last equation

\[ [e_1]^{n\alpha - (n-1)} = [e_1]^0 = \text{sign}(e_1). \]

Therefore, if \(|\theta(t)| < k_n\) then the sign function can compensate exactly the influence of \(\theta\). Such a differentiation algorithm is called Higher Order Sliding Mode (HOSM) differentiator [16]. For \(n = 2\), its very popular version, super-twisting differentiator, has been proposed in [15]:

\[ \dot{z}_1 = z_2 - k_1 [z_1 - y]^{0.5}, \quad \dot{z}_2 = -k_2 \text{sign}(z_1 - y). \]

Its non-homogeneous modification has been given in [20]. However, the sign function is discontinuous, then a chattering phenomenon may appear in the numerical realization due to measurement noise [15], [16]. According to [16], the system (7) is globally finite-time stable for this selection of \(\alpha\) and for \(k_1, \ldots, k_n\) forming a Hurwitz polynomial.

Another limiting case is by selecting \(\alpha = 1\), then the system (6) is reduced to a linear one \((\alpha_i = r_i = 1\) for all \(i = 1, \ldots, n)\) and for sufficiently high values of gains \(k_1, \ldots, k_n\) (forming a Hurwitz polynomial) a differentiator from [14] can be obtained. In this case, the error system is globally exponentially stable. As for the case \(\alpha = 1\) the influence of \(\theta\) is not rejected but just attenuated by high gains \(k_1, \ldots, k_n\).

Remark 2. To switch between a HOSM differentiator, a high-gain linear differentiator or homogeneous one, it is necessary to fix the gains \(k_1, \ldots, k_n\) forming a Hurwitz polynomial and take \(\alpha = 1 - \frac{1}{n}\), \(\alpha = 1\) or \(\alpha \in \left[1 - \frac{1}{n-1}, 1\right]\), respectively. Thus, by tuning only \(\alpha\), it is possible to switch between differentiators. To select the gains \(k_1, \ldots, k_n\) the following polynomials can be considered

\[ (s + \lambda)^{n+1} = s^n + \sum_{j=1}^{n} k_j s^{n-j} \]

parametrized by \(\lambda > 0\). Consequently, two parameters, \(\alpha\) and \(\lambda\), can be used for tuning these three differentiators.

IV. LATENCY COMPENSATION

Let us consider (1). By selecting some \(M > 0\) and by taking estimates of derivatives \(\hat{x}^{(j)}(t_i)\) calculated for \(j = 1, \ldots, M\) by one of numerical differentiators given in Section III, we obtain a forecasting algorithm.

A. Noise and overshoot attenuation

One problem is that the estimates \(\hat{x}^{(j)}(t_i)\), \(j = 1, \ldots, M\) (independently on used differentiation algorithm from Section III) are sensitive to the measurement noise \(v\) (imported by the mouse) and sampling/quantization errors (imported by software), especially for higher values of \(j\). Thus, the predicted values \(\hat{x}(t_i + L)\) contain these errors/noises, and it is desirable to decrease the dependence of \(\hat{x}(t_i + L)\) on them. The influence of the noise is most important for small values of derivatives (when only noise is present in \(\hat{x}^{(j)}(t_i)\)) and for high amplitudes of the signal \(\hat{x}^{(j)}(t_i)\) (these big deviations of \(\hat{x}^{(j)}(t_i)\) may also be originated by the noise). Of course, some preliminary filtering of \(y(t)\) can be used to decrease the noise influence in the differentiated signal. However, any additional filtering will induce an additional delay in \(y(t)\) to be compensated. Thus, in this work we applied a static nonlinear transformation \(\chi_j : \mathbb{R} \rightarrow \mathbb{R}\), \(j = 1, \ldots, M\) of \(\hat{x}^{(j)}(t_i)\), that combines saturation and dead-zone, in order to reduce noise influence on \(\hat{x}(t_i + L)\) for big and small values of \(\hat{x}^{(j)}(t_i)\) respectively. The form of the transformation \(\chi_j\) is given in Fig. 4, where the parameter \(\sigma_j > 0\) determines the dead-zone width and \(\sigma_j > 0\) is responsible for saturation amplitude. The forecasting algorithm (1) takes the form:

\[
\hat{x}(t_i + L) = \sum_{j=0}^{M} \frac{L}{j!} \chi_j [\hat{x}^{(j)}(t_i)].
\]
right hand-side of (8) depending on derivative value:

$$\tilde{L}(\gamma) = \max \left\{ 0, L - \sum_{j=1}^{M} \gamma_j |x_j[\tilde{z}^{(j)}(t_i)]| \right\},$$

where $\gamma = [\gamma_1, \ldots, \gamma_M]^T \in \mathbb{R}^M$ are tuning parameters. The idea is also that if the values of some derivatives are high, then to avoid overshooting, it is desirable to decrease the interval of prediction $L$. Then (8) can be rewritten as:

$$\tilde{x}(t_i + L) = \sum_{j=0}^{M} \tilde{L}_j \chi_j[\tilde{z}^{(j)}(t_i)]. \quad (9)$$

To realize (9), we must select (or tune) the parameters $\sigma_j$, $\sigma$, $\gamma_j$ for $j = 1, \ldots, M$, and the type of differentiator, e.g., algebraic (just one tuning parameter $m$), HOSM, linear high-gain or homogeneous one (two tuning parameters $\alpha$ and $\lambda$).

B. Parameter tuning

To tune the parameters and select the type of differentiator, the following criteria have been chosen for $e(t_i) = x(t_i + L) - \tilde{x}(t_i + L)$:

$$J_2 = \sum_{i=k}^{K} e^2(t_i), \quad J_\infty = \max_{k \leq i \leq K} |e(t_i)|,$$

where $0 < k < K$ is selected to avoid the influence of transients at initial instants of time. Notice that $J_\infty$ helps to minimize the overshooting in forecasting, and $J_2$ is responsible for overall closeness of $x(t_i + L)$ and $\tilde{x}(t_i + L)$.

It is difficult to tune all these parameters simultaneously due to a high nonlinearity and interrelations among them, then an iterative tuning procedure can be applied.

First, for parameters $\sigma_j$, $\sigma$, $\gamma$, $\gamma_j$ and $\alpha$, $\lambda$, it is not possible to calculate a gradient with respect to $J_2$ or $J_\infty$, therefore, a heuristic minimization algorithm has to be applied [24]. The simplest one consists in generating a grid of admissible values for these parameters and calculating next the corresponding values of $J_2$ and $J_\infty$. For instance, if $\Theta \subset \mathbb{R}^{2M+2}$ where $\theta = [\sigma_1, \sigma_1, \ldots, \sigma_M, \sigma_M, \alpha, \lambda] \in \mathbb{R}^{2M+2}$ and $\Theta$ defines the set of admissible values, then

$$\theta^* = \arg \min_{\theta \in \Theta} \left( \kappa_2 J_2(\theta) + \kappa_\infty J_\infty(\theta) \right)$$

with $\kappa_2 > 0$ and $\kappa_\infty > 0$ are weighting coefficients. To evaluate the shape of $\Theta$, some preliminary statistics on measurements of $x(t_j)$ and derivatives $\tilde{z}^{(j)}(t_i)$ can be used, e.g., histograms of these signals with taking $\sigma_j$, $\sigma$ in order to cover a desired percentage of values for $\tilde{z}^{(j)}$.

It is possible to calculate the gradient of $J_2$ with respect to $\gamma = [\gamma_1, \ldots, \gamma_M]^T$

$$\nabla_\gamma J_2(\gamma) = \begin{cases} \sum_{i=k}^{K} e(t_i) \sum_{j=1}^{M} \frac{\tilde{L}^{j-1}(\gamma)}{(j-1)!} \chi_j[\tilde{z}^{(j)}(t_i)] d(t_i) & \text{if } L > \gamma^T d(t_i), \\ 0 & \text{otherwise}, \end{cases}$$

where $d(t_i) = [\chi_1[\tilde{z}^{(1)}(t_i)], \ldots, \chi_M[\tilde{z}^{(M)}(t_i)]]^T$. Then a least square algorithm can be applied to optimize the values of these parameters. The same for the parameter $m$ in (5).

### C. Experimental results

To test our proposed forecasting algorithms, two datasets have been collected using Logitech G500 mouse with sampling period 1 ms and Libpointing software, with lags 100 ms and 50 ms. More precisely, to produce a dataset, we record human movements of a mouse with Libpointing and add a predefined artificial lag. Optimization has been performed for each dataset separately either for homogeneous or for algebraic differentiators.

The results of application of the developed forecasting algorithm are shown in Fig. 5 for the lag 100 ms and in Fig. 6 for the lag 50 ms. These results are obtained by algebraic and homogeneous differentiators with $M = 4$. The values of criteria for these scenarios are given in Tab. I. As we can conclude from these results, homogeneous differentiators provide a worse quality of forecasting (both criteria $J_2$ and $J_\infty$ take higher values) and the predicted signal has more chattering due to noise and quantization, while the new algebraic differentiators give smoother curves. For the case of latency 50 ms the results of forecasting have less overshoots and errors than for 100 ms.

Another set of simulation results is devoted to analysis

---

**Table I**

<table>
<thead>
<tr>
<th>Lag</th>
<th>Differentiation</th>
<th>$J_2$</th>
<th>$J_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Algebraic</td>
<td>3541.41</td>
<td>157.60</td>
</tr>
<tr>
<td>100</td>
<td>Homogeneous</td>
<td>4134.70</td>
<td>166.52</td>
</tr>
<tr>
<td>50</td>
<td>Algebraic</td>
<td>2146.22</td>
<td>106.00</td>
</tr>
<tr>
<td>50</td>
<td>Homogeneous</td>
<td>2264.94</td>
<td>118.48</td>
</tr>
</tbody>
</table>

---

**Fig. 5.** Forecasting for the lag 100 ms

**Fig. 6.** Forecasting for the lag 50 ms
of impact of the value $M$. For the case of the lag 100 ms, forecasting algorithm with algebraic differentiators has been applied for different values of $M$. The values of criteria are given in Tab. II and the corresponding predicted values are shown in Fig. 7. As we can conclude, the second derivative improves significantly the quality of forecasting, the third and fourth derivatives also influence positively on the quality of prediction (the predicted signal becomes more oscillatory with these derivatives), but with less impact.

V. Conclusion

The problem of latency compensation for human-computer interaction through a mouse device has been introduced. A simple forecasting algorithm has been developed, based on truncated Taylor series and numerical differentiators, including a novel algebraic technique. To decrease the dependence of predicted values on measurement and numerical noises, a nonlinear transformation has been applied to derivatives (since filtering would introduce a delay). Nonlinear optimization tools have been used for parameter tuning. The results of the developed forecasting algorithm applied on real data have been included to demonstrate efficiency of the approach. The implementation of this algorithm in the software of touch-based devices is planned. In future works, a comparison with other existing methods will be studied.

REFERENCES


<table>
<thead>
<tr>
<th>$M$</th>
<th>$J_2$</th>
<th>$J_{\infty}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4933.18</td>
<td>192.89</td>
</tr>
<tr>
<td>2</td>
<td>3821.61</td>
<td>149.30</td>
</tr>
<tr>
<td>3</td>
<td>3720.08</td>
<td>150.47</td>
</tr>
<tr>
<td>4</td>
<td>3662.73</td>
<td>150.23</td>
</tr>
</tbody>
</table>

Fig. 7. Forecasting using different derivatives