ORIGINAL PAPER

DOI: 10.26794/2587-5671-2022-26-3-196-225
UDC 330.42(045),51–77(045)
JEL C1, C15, C4, C5, C53

An Algorithm for Restoring a Function from Different Functionals for Predicting Rare Events in the Economy

Yu.A. Korablev
Financial University, Russia, Moscow

ABSTRACT

This paper aims to restore some parameters of functionals using cubic splines to forecast rare events in finance and economics. The article considers the mathematical method for recovering an unknown function from many different functionals, such as the value of a function, the value of its first derivative, second derivative, as well as a definite integral over a certain interval. Moreover, all observations can occur with an error. Therefore, the author uses a method of recovering a function from different functionals observed with an error. The function is restored in the form of a cubic spline, which has a value-second derivative representation. The optimization problem consists in minimizing several sums of squares of the deviation at once, for ordinary values, for the first derivatives, for the second derivatives, for integrals, and for roughness penalty. For greater flexibility, weights have been introduced both for each group of observations and for each individual observation separately. The article shows in detail how the elements of each corresponding matrix are filled in. The appendix provides an implementation of the method as a FunctionalSmoothingSpline function in R language. Examples of using the method for the analysis and forecasting of rare (discrete) events in the economy are given. Formulas for calculating the cross-validation score $CV(\alpha)$ for the automatic procedure for determining the smoothing parameter α from the data in our problem of recovering a function by many functionals are shown. The paper concludes that the presented method makes it possible to analyze and predict rare events, which will allow you to prepare for such future events, get some benefit from this, or reduce possible risks or losses.

Keywords: rare events; forecast; event analysis; recovery by functionals; smoothing spline; roughness penalty; R; FunctionalSmoothingSpline; cross-validation

For citation: **Korablev Yu.A. An algorithm for restoring a function from different functionals for predicting rare events in the economy. *Finance: Theory and Practice.* 2022;26(3):196-225. DOI: 10.26794/2587-5671-2022-26-3-196-225**

1. INTRODUCTION

Rare events in the economy have a special interest. The rarity of events from the point of view of information theory makes such events more significant [1]. At the same time, events can have significant costly consequences and the ability to predict them is an urgent task. In logistics, for example, there is a lot of interest in predicting rare/intermittent demand forecast when demand occurs at a large number of intervals at which there was no demand. Of course, absolutely random events cannot be predicted, but if there is some pattern in the occurrence of events, then prediction is possible. Croston's [2] and Willemaine's [3] most popular methods, as well as many of their modifications, can find statistical patterns. Recent reviews [4, 5] analyse existing publications on such methods of forecasting intermittent demand. But in all methods only statistical analysis takes place to some extent. From the available data (intermittent demand) either the parameters of distributions or the values of transition probabilities of simple models of the Markov process. Such approaches, if they can give the expected number of events per time interval, are not able to give a forecast of the moment of occurrence of a particular event (estimation of probability of an event in the next time period, which is some statistical "shamanism").

The author, unlike other researchers, offers an approach [6] based on the consideration of internal processes leading to the occurrence of events. So, the same rare demand should be analyzed from the point of view of the process of consumption, occurring on the side of a client outside our control. It turns out that it will not be difficult to restore the speed (intensity) of consumption of products at a particular customer. Of course, preliminary data about events (purchases) should be divided into different samples depending on the sources (customers) where they are generated, which may not always be possible due to imperfections in the data collection methodology. Viewing the data in the sample as a sequence of integrals from the consumption rate, there is a recovery of this very unknown consumption rate function by means of functional recovery methods. This approach can be applied not only in the analysis of intermittent demand, but in any area where there are processes similar to devastation (capacity) or the accumulation of some disturbance to a certain level, after which it is reset to the initial level (this method the author called "capacitive method"). In the financial sphere, this approach can also be applied, for example, when the client periodically requests support.

In a similar way, one might wonder what other processes might exist in an economy that generates rare events. Note that we are not interested in completely random events, which in principle is impossible (or at this point is not possible) to predict, we are interested in the events that arise from some process that we could formulate and then reproduce ourselves.

Here it is worth remembering, and what is the randomness itself. Randomness is only a measure of uncertainty, a measure of ignorance, an abstraction introduced to compact all the many unknown factors to the researcher. Of course, because of the universal study of statistics and theory of probability in institutions of higher education and already in schools, the notion of randomness and probability became almost a physical phenomenon. At the same time, there is a philosophical concept of cosmic determinism, when everything in the universe is predestined and a wise man, knowing the position and velocity of all particles at one time, can predict their position at any other time. This concept is opposed by the Heisenberg uncertainty principle, where in quantum mechanics it is impossible to determine both the position and the momentum of particles at the same time. We will not argue with either concept and argue whether there is a true randomness or not. The above was only necessary to ensure that senior colleagues

who have worked for many decades in the field of statistical research look at events or observations not as a statistical sample, and instead they tried to go deep into each event and they thought, what is the cause of this event, what is the mechanism of its formation. They looked at streams of events not as random streams, but as the result of some mechanism of formation of these events, or even the set of mechanisms from which the events are mixed when the implementation of data collection fails.

As mentioned above, many economic events are related to consumption processes (or accumulation of impact). Since we know how they are formed, the streams of events are no longer random, this knowledge brings in the information we need and allows us to better analyze them. If you do not mix events from different sources, in the simplest case it is possible to restore the speed of consumption on the side of the client, then construct a pattern model and calculate the next moment of consumption. For this it is enough to look at the values of volumes of purchases as integral. However, in the more complex case, other available information is required. This study focuses on how to recover unknown process parameters if different types of data are available, such as values at a certain point in time, values of the first and second derivatives at certain times, values of certain integrals at certain periods of time. And these data may be available all at once, and one thing may be available, but the sample sizes of the input data of the different characteristics may not be the same (or are zero if such data are not available).

2. FUNCTIONAL RECOVERY

2.1. Optimization problem

Need to restore dynamically changing value of some process parameter from available data. Assume that this parameter changes continuously as some unknown function $f(t)$, i.e. $f(t)$ — desired function.

The following data are known:

$$y_i = f(t_i) + \varepsilon_i,\ i = 1,\ldots,n_f;$$

$$y_j^{'} = f'(t_j) + \varepsilon_j^{'},\ j = 1,\ldots,n_{df};$$

$$y_l^{''} = f''(t_l) + \varepsilon_l^{''},\ l = 1,\ldots,n_{d^2f};$$

$$Y_u = \int_{t_u^a}^{t_u^b} f(t)dt + \varepsilon_u^{\text{int}},\ u = 1,\ldots,n_{\text{int}},$$

where t_i, t_j, t_l — the moment of observation of the values of the unknown function $f(t)$, its first and second derivatives; t_u^a and t_u^b — lower and upper integration range of the appropriate integral monitoring; ε_i, $\varepsilon_j^{'}, \varepsilon_l^{''}, \varepsilon_u^{\text{int}}$ — errors of observation at values, first derivative, second derivative and integral respectively (with zero mathematical expectation, variances can be different); n_f, n_{df}, n_{d^2f}, n_{int} — correspondingly the number of observations of values, first derivatives, second derivatives, integrals of the desired function.

Of course, it will not be possible to accurately restore the original function, there are an infinite number of ways to chart the function so that the corresponding values of this function comply with predetermined values at the specified points. You can roughly restore the function by imposing certain restrictions. For example, we can say that we will recover bits of polynomials (splines) of a certain degree (third degree is the most common). Will impose restrictions on the flexibility (roughness) of the function.

This class of tasks is called collocation tasks, inverse tasks, approximation tasks (can be considered as synonyms in this context). It is considered (in the English-language literature) that the fundamental work of approximating the restoration of functions is the work G. Kimeldorf and G. Wahba [7]. To date, this work has been quoted 797 times in international citation databases. Studying the whole list of citing works and familiarization with more than 100 full-text publications from this list (based on annotations) showed the lack of a ready solution to the problem, which is

solved in this article (even as presented by the author). For some reason, a lot of work is limited to interpolation splines, when errors are not taken into account, only interpolation conditions are used in the form of exact equations. In some works, it is possible to find a unique solution only on the basis of one functionality, not many, and mainly again with the help of interpolation splines [8, 9].

The decision presented by the author is based on a well written work P. J. Green and B. W. Silverman [10] and is based on cubic smoothing splines, which have a representation in the form of values and second derivatives. To restore the function for many functions, we will minimize several squares of deviation and nonlinearity penalty at once

$$S(f)=\sum_{i=1}^{n_f}\left(y_i-f(t_i)\right)^2+\sum_{j=1}^{n_{df}}\left(y_j^{'}-f'(t_j)\right)^2+\sum_{l=1}^{n_{d^2f}}\left(y_l^{''}-f''(t_l)\right)^2 + \sum_{u=1}^{n_{\text{int}}}\left(Y_u-\int_{t_u^a}^{t_u^b}f(t)dt\right)^2+\alpha\int_{t_{start}}^{t_{end}}\left(f''(t)\right)^2dt, \quad (1)$$

where the last summand — nonlinear penalty; α — smoothing ratio (regularization); t_{start} and t_{end} — the boundary at which the function is restored.

Since the values in which the values, derivatives and integrals are measured can vary greatly, add the appropriate coefficients so that the weight of each group of observations can be increased. In addition, we can change the weight of each individual observation. As a result, the optimization task will take the following form:

$$S(f)=\sum_{i=1}^{n_f}w_i^f\left(y_i-f(t_i)\right)^2+\mu\sum_{j=1}^{n_{df}}w_j^{df}\left(y_j^{'}-f'(t_j)\right)^2+\nu\sum_{l=1}^{n_{d^2f}}w_l^{d^2f}\left(y_l^{''}-f''(t_l)\right)^2 + \psi\sum_{u=1}^{n_{\text{int}}}w_u^{\text{int}}\left(Y_u-\int_{t_u^a}^{t_u^b}f(t)dt\right)^2+\alpha\int_{t_{start}}^{t_{end}}\left(f''(t)\right)^2dt, \quad (2)$$

where w_i^f, w_j^{df}, $w_l^{d^2f}$, w_u^{int} — individual weights of the respective observation groups;

μ — weight of the whole group of observations of the first derivatives; ν — weight of the whole group of observations of second derivatives; ψ — weight of the whole integral group of observations. Note that there is no weight for the whole group of observations of normal values, i.e. it is assumed to be equal to one, and all other coefficients then show weight compared to this first group of observations. Note also that all individual weights could be adjusted proportionally to change the group weight, but this is not very convenient, so we will use both individual and group weights.

Next, we will restore the unknown function $f(t)$ as a cubic spline $g(t)$ (articulation of pieces of third-degree polynomials).

2.2. Spline types

Instead of the usual representation of polynomials with 4 unknown coefficients a_0, a_1, a_2, a_3 for each slice of spline, between nodes s_k and s_{k+1}

$$g(t)=a_0+a_1(t-s_k)+a_2(t-s_k)^2+a_3(t-s_k)^3,\ s_k\le t\le s_{k+1}$$

we will use more convenient performance through spline values $g_k=g(s_k)$ and its second derivative values $\gamma_k=g''(s_k)$ at the ends of each interval (value-second derivative representation) [10, p. 12, 22, 23]

$$g(t)=\frac{(t-s_k)g_{k+1}+(s_{k+1}-t)g_k}{s_{k+1}-s_k}$$
$$-\frac{1}{6}(t-s_k)(s_{k+1}-t)\left[\left(1+\frac{t-s_k}{s_{k+1}-s_k}\right)\gamma_{k+1}+\left(1+\frac{s_{k+1}-t}{s_{k+1}-s_k}\right)\gamma_k\right], \quad (3)$$
$$s_k \le t \le s_{k+1}.$$

As before, each piece of spline is identified by 4 unknown g_k, g_{k+1}, γ_k, γ_{k+1}, but since the end of one piece of spline is the beginning of the next piece, it is enough to define only two parameters g_k, γ_k for each nodes s_k (note that the parameters g_k, γ_k contain more physical meaning than parameters a_0, a_1, a_2, a_3). To define spline in all m nodes $s_1 < s_2 < \ldots < s_m$ (the number of nodes m is usually given a priori by the researcher) it is necessary to specify a vector of values $g=(g_1,\ldots,g_m)^T$ and the vector of second derivatives $\gamma=(\gamma_2,\ldots,\gamma_{m-1})^T$ (the second derivative at the spline ends turns to zero $\gamma_1=\gamma_m=0$ — natural spline conditions).

This form of spline ensures continuity $g(t)$ and its second derivative $g''(t)$ in the articulation points (spline nodes s_k). However, for the continuity of the first derivative at the points of articulation $g'(s_k-0)=g'(s_k+0)$ must be done a system $m-2$ equations

$$\frac{g_{k+1}-g_k}{s_{k+1}-s_k}-\frac{g_k-g_{k-1}}{s_k-s_{k-1}}=\frac{1}{6}(s_{k+1}-s_k)(\gamma_{k+1}+2\gamma_k)+\frac{1}{6}(s_k-s_{k-1})(2\gamma_k+\gamma_{k-1}), \quad (4)$$
$$k=2,\ldots,m-1;$$

or in matrix form

$$Q^T g = R\gamma\,, \quad (5)$$

where Q — tridiagonal matrix of coefficients at unknown g_k dimension $m\times(m-2)$ (column written); R — three-diagonal matrix of coefficients at unknown γ dimension $(m-2)\times(m-2)$ ($h_k=s_{k+1}-s_k$ node distance for $k=1,\ldots,m-1$).

Q	2	3	…	$m-1$
1	h_1^{-1}	0	…	0
2	$-h_1^{-1}-h_2^{-1}$	h_2^{-1}	…	0
3	h_2^{-1}	$-h_2^{-1}-h_3^{-1}$	…	0
4	0	h_3^{-1}	…	0
…	…	…	…	…
$m-2$	0	0	…	h_{m-2}^{-1}
$m-1$	0	0	…	$-h_{m-2}^{-1}-h_{m-1}^{-1}$
m	0	0	…	h_{m-1}^{-1}

R	2	3	4	…	$m-1$
2	$(h_1+h_2)/3$	$h_2/6$	0	…	0
3	$h_2/6$	$(h_2+h_3)/3$	$h_3/6$	…	0
4	0	$h_3/6$	$(h_3+h_4)/3$	…	0
5	0	0	$h_4/6$	…	0
…	…	…		…	…
$m-2$	0	0		…	$h_{m-2}/6$
$m-1$	0	0		…	$(h_{m-2}+h_{m-1})/3$

Instead of including the continuity conditions of the first derivative $Q^T g = R\gamma$ as a constraint system in the optimization problem, from this system of equations express one of the unknown, for example $\gamma = R^{-1}Q^T g$, replace it and solve the optimization problem with only one of the unknowns.

Penalty of smoothness (roughness) $\int_{s_1}^{s_m}\left(g''(t)\right)^2 dt$ is simplified to operations with the same matrices, see [10, p. 24–35]:

$$\int_{s_1}^{s_m}\left(g''(t)\right)^2 dt = \gamma^T Q^T g = \gamma^T R\gamma = g^T\left[QR^{-1}Q^T\right]g = g^T Kg, \tag{6}$$

where $K = QR^{-1}Q^T$ symmetric matrix by dimension $m\times m$.

2.3. Recovery

As a result, we have the following task — to determine the parameters of spline $g(t)$, minimizing

$$\begin{aligned} S(g) = &\sum_{i=1}^{n_f} w_i^f\left(y_i - g(t_i)\right)^2 + \mu\sum_{j=1}^{n_{df}} w_j^{df}\left(y_j' - g'(t_j)\right)^2 + \nu\sum_{l=1}^{n_{d^2f}} w_l^{d^2f}\left(y_l'' - g''(t_l)\right)^2 \\ &+ \psi\sum_{u=1}^{n_{\text{int}}} w_u^{\text{int}}\left(Y_u - \int_{t_u^a}^{t_u^b} g(t)\,dt\right)^2 + \alpha\int_{t_{start}}^{t_{end}}\left(g''(t)\right)^2 dt, \end{aligned} \tag{7}$$

where the spline $g(t)$ has a form (3).

To simplify the record, it is convenient to record the notation $h_k = s_{k+1} - s_k$, $h_k^{-i} = t_i - s_k$, $h_k^{+i} = s_{k+1} - t_i$. The record of related functionalities will be as follows:

$$g(t_i) = \frac{h_k^{-i}}{h_k}g_{k+1} + \frac{h_k^{+i}}{h_k}g_k - \frac{h_k^{-i}h_k^{+i}\left(h_k + h_k^{-i}\right)}{6h_k}\gamma_{k+1} - \frac{h_k^{-i}h_k^{+i}\left(h_k + h_k^{+i}\right)}{6h_k}\gamma_k, \tag{8}$$
$$k : s_k \le t_i < s_{k+1};$$

$$g'(t_j) = \frac{g_{k+1}}{h_k} - \frac{g_k}{h_k} - \left(\frac{h_k}{6} - \frac{\left(h_k^{-j}\right)^2}{2h_k}\right)\gamma_{k+1} + \left(\frac{h_k}{6} - \frac{\left(h_k^{+j}\right)^2}{2h_k}\right)\gamma_k, \tag{9}$$
$$k : s_k \le t_j < s_{k+1};$$

$$g''(t_l)=\frac{h_k^{-l}}{h_k}\gamma_{k+1}+\frac{h_k^{+l}}{h_k}\gamma_k,$$
$$k: s_k \le t_l < s_{k+1}; \tag{10}$$

$$\int_{t_u^a}^{t_u^b} g(t)dt=\sum_{l=0}^{L}\int_{s_{k+l}}^{s_{k+l+1}} g(t)dt-\int_{s_k}^{t_u^a} g(t)dt-\int_{t_u^b}^{s_{k+L+1}} g(t)dt=$$
$$L: s_{k+L} < t_u^b \le s_{k+L+1},$$
$$k: s_k \le t_u^a < s_{k+1}. \tag{11}$$

$$=\sum_{l=0}^{L}\left[\frac{h_{k+l}}{2}g_{k+l+1}+\frac{h_{k+l}}{2}g_{k+l}-\frac{h_{k+l}^3}{24}\gamma_{k+l+1}-\frac{h_{k+l}^3}{24}\gamma_{k+l}\right]$$
$$-\frac{\left(h_k^{-a}\right)^2}{2h_k}g_{k+1}-\frac{h_k^2-\left(h_k^{+a}\right)^2}{2h_k}g_k-\frac{\left(h_k^{-a}\right)^2\left(\left(h_k^{-a}\right)^2-2h_k^2\right)}{24h_k}\gamma_{k+1}+\frac{\left(h_k^{-a}\right)^2\left(h_k^{+a}+h_k\right)^2}{24h_k}\gamma_k$$
$$-\frac{h_{k+L}^2-\left(h_{k+L}^{-b}\right)^2}{2h_{k+L}}g_{k+L+1}-\frac{\left(h_{k+L}^{+b}\right)^2}{2h_{k+L}}g_{k+L} \tag{12}$$
$$+\frac{\left(h_{k+L}^{+b}\right)^2\left(h_{k+L}^{-b}+h_{k+L}\right)^2}{24h_{k+L}}\gamma_{k+L+1}-\frac{\left(h_{k+L}^{+b}\right)^2\left(\left(h_{k+L}^{+b}\right)^2-2h_{k+L}^2\right)}{24h_{k+L}}\gamma_{k+L},$$
$$L: s_{k+L} < t_u^b \le s_{k+L+1},$$
$$k: s_k \le t_u^a < s_{k+1},$$

where in the last expression $h_k^{-a}=t_u^a-s_k$, $h_k^{+a}=s_{k+1}-t_u^a$, $h_k=s_{k+1}-s_k$, $h_{k+L}^{-b}=t_u^b-s_{k+L}$, $h_{k+L}^{+b}=s_{k+L+1}-t_u^b$, $h_{k+L}=s_{k+L+1}-s_{k+L}$.

In all these expressions, at the beginning it is determined at what interval k was observed. In the most recent expression for the integral, it is necessary to define at the beginning, at what interval k dropped out the lower limit of integration t_u^a and at what interval $k+L$ has dropped out the limit of integration t_u^b, where L — number of intervals between them (L can be equal to 0 if both are on the same interval).

All these expressions have a linear form relative to unknown spline parameters g_k and γ_k. Therefore, the optimization problem (7) can be expressed in the following matrix form:

$$S(g)=\left(Y_f-V_f g+P_f\gamma\right)^T W_f\left(Y_f-V_f g+P_f\gamma\right)+$$
$$+\mu\left(Y_{df}-V_{df}g+P_{df}\gamma\right)^T W_{df}\left(Y_{df}-V_{df}g+P_{df}\gamma\right)+$$
$$+\nu\left(Y_{d^2f}-0g+P_{d^2f}\gamma\right)^T W_{d^2f}\left(Y_{d^2f}-0g+P_{d^2f}\gamma\right)+ \tag{13}$$
$$+\psi\left(Y_{\text{int}}-V_{\text{int}}g+P_{\text{int}}\gamma\right)^T W_{\text{int}}\left(Y_{\text{int}}-V_{\text{int}}g+P_{\text{int}}\gamma\right)+$$
$$+\alpha g^T Kg\to\min,$$

where Y_f, Y_{df}, Y_{d^2f}, Y_{int} — observation column; matrix V_f, V_{df}, V_{int} — coefficient matrices at unknown g_k; P_f, P_{df}, P_{d^2f}, P_{int} — coefficient matrices at unknown γ_k; W_f, W_{df}, W_{d^2f}, W_{int} — diagonal weight matrices.

V_f dimension $n_f \times m$, and each i-line looks like

1	…	$k-1$	k	$k+1$	$k+2$	…	m
0	…	0	h_k^{+i}/h_k	h_k^{-i}/h_k	0	…	0

P_f dimension $n_f \times (m-2)$, and each i-line looks like

2	…	$k-1$	k	$k+1$	$k+2$	…	$m-1$
0	…	0	$h_k^{-i}h_k^{+i}\left(h_k+h_k^{+i}\right)/6h_k$	$h_k^{-i}h_k^{+i}\left(h_k+h_k^{-i}\right)/6h_k$	0	…	0

V_{df} dimension $n_{df} \times m$, and each j-line looks like

1	…	$k-1$	k	$k+1$	$k+2$	…	m
0	…	0	$-1/h_k$	$1/h_k$	0	…	0

P_{df} dimension $n_{df} \times (m-2)$, and each j-line looks like

2	…	$k-1$	k	$k+1$	$k+2$	…	$m-1$
0	…	0	$-h_k/6+\left(h_k^{+j}\right)^2/2h_k$	$h_k/6+\left(h_k^{-j}\right)^2/2h_k$	0	…	0

P_{d^2f} dimension $n_{d^2f} \times (m-2)$, and each l-line looks like

2	…	$k-1$	k	$k+1$	$k+2$	…	$m-1$
0	…	0	$-h_k^{+l}/h_k$	$-h_k^{-l}/h_k$	0	…	0

V_{int} dimension $n_{int} \times m$, and each u-line is filled in as follows:

$$V_{u,k}=\frac{\left(h_k^{+a}\right)^2}{2h_k};V_{u,k+l}=\frac{h_{k+l-1}+h_{k+l}}{2},l=1,\ldots,L;V_{u,k+1}=V_{u,k+1}-\frac{\left(h_k^{-a}\right)^2}{2h_k};$$
$$V_{u,k+L}=V_{u,k+L}-\frac{\left(h_{k+L}^{+b}\right)^2}{2h_{k+L}};V_{u,k+L+1}=\frac{\left(h_{k+L}^{-b}\right)^2}{2h_{k+L}}. \quad (14)$$

Note that depending on L, some expressions may change twice (for example, if $L=0$, that k-item is changed by two expressions $V_{i,k}$ and $V_{i,k+L}$). In the case $L>2$ line u will be

$k-1$	k	$k+1$	$k+2$	$\ldots k+l \ldots$	$k+L$	$k+L+1$	$k+L+2$
0	$\frac{\left(h_k^{+a}\right)^2}{2h_k}$	$\left(\frac{h_k+h_{k+1}}{2}-\frac{\left(h_k^{-a}\right)^2}{2h_k}\right)$	$\frac{h_{k+1}+h_{k+2}}{2}$	$\frac{h_{k+l-1}+h_{k+l}}{2}$	$\left(\frac{h_{k+L-1}+h_{k+L}}{2}-\frac{\left(h_{k+L}^{+b}\right)^2}{2h_{k+L}}\right)$	$\frac{\left(h_{k+L}^{-b}\right)^2}{2h_{k+L}}.$	0

In the case $L=0$ line u will be

$k-1$	k	$k+1$	$k+2$
0	$\begin{pmatrix} \frac{\left(h_k^{+a}\right)^2}{2h_k} \\ -\frac{\left(h_{k+L}^{+b}\right)^2}{2h_{k+L}} \end{pmatrix}$	$\begin{pmatrix} \frac{\left(h_{k+L}^{-b}\right)^2}{2h_{k+L}} \\ -\frac{\left(h_k^{-a}\right)^2}{2h_k} \end{pmatrix}$	0

P_{int} dimension $n_{int}\times(m-2)$, each u-line is filled in as follows:

$$P_{u,k}=\frac{h_k^3}{24}-\frac{\left(h_k^{-a}\right)^2\left(h_k^{+a}+h_k\right)^2}{24h_k};P_{u,k+l}=\frac{h_{k+l-1}^3+h_{k+l}^3}{24},l=1,\ldots,L;$$
$$P_{u,k+1}=P_{u,k+1}+\frac{\left(h_k^{-a}\right)^2\left(\left(h_k^{-a}\right)^2-2h_k^2\right)}{24h_k};P_{u,k+L}=P_{u,k+L}+\frac{\left(h_{k+L}^{+b}\right)^2\left(\left(h_{k+L}^{+b}\right)^2-2h_{k+L}^2\right)}{24h_{k+L}};\quad(15)$$
$$P_{u,k+L+1}=\frac{h_{k+L}^3}{24}-\frac{\left(h_{k+L}^{+b}\right)^2\left(h_{k+L}^{-b}+h_{k+L}\right)^2}{24h_{k+L}}.$$

Here too, depending on L, some values may change several times. In the case $L>2$ line u will be

$k-1$	k	$k+1$	$k+2$
0	$\begin{pmatrix} \frac{h_k^3}{24}- \\ \frac{\left(h_k^{-a}\right)^2\left(h_k^{+a}+h_k\right)^2}{24h_k} \end{pmatrix}$	$\begin{pmatrix} \frac{h_k^3+h_{k+1}^3}{24}+ \\ \frac{\left(h_k^{-a}\right)^2\left(\left(h_k^{-a}\right)^2-2h_k^2\right)}{24h_k} \end{pmatrix}$	$\frac{h_{k+1}^3+h_{k+2}^3}{24}$
$\ldots k+l\ldots$	$k+L$	$k+L+1$	$k+L+2$
$\frac{h_{k+l-1}^3+h_{k+l}^3}{24}$	$\begin{pmatrix} \frac{h_{k+L-1}^3+h_{k+L}^3}{24}+ \\ \frac{\left(h_{k+L}^{+b}\right)^2\left(\left(h_{k+L}^{+b}\right)^2-2h_{k+L}^2\right)}{24h_{k+L}} \end{pmatrix}$	$\begin{pmatrix} \frac{h_{k+L}^3}{24}- \\ \frac{\left(h_{k+L}^{+b}\right)^2\left(h_{k+L}^{-b}+h_{k+L}\right)^2}{24h_{k+L}} \end{pmatrix}$	0

Then, due to the continuity conditions of the first derivative, (5) $Q^Tg=R\gamma$, that can express $\gamma=R^{-1}Q^Tg$, optimization task (13) can be written more compactly only through one unknown g:

$$S(g)=\left(Y_f-C_fg\right)^TW_f\left(Y_f-C_fg\right)+\mu\left(Y_{df}-C_{df}g\right)^TW_{df}\left(Y_{df}-C_{df}g\right)+$$
$$+\nu\left(Y_{d^2f}-C_{d^2f}g\right)^TW_{d^2f}\left(Y_{d^2f}-C_{d^2f}g\right)+\quad(16)$$
$$+\psi\left(Y_{\text{int}}-C_{\text{int}}g\right)^TW_{\text{int}}\left(Y_{\text{int}}-C_{\text{int}}g\right)+\alpha g^TKg\to\min,$$

where $C_f=V_f-P_fR^{-1}Q^T$, $C_{df}=V_{df}-P_{df}R^{-1}Q^T$, $C_{d^2f}=0-P_{d^2f}R^{-1}Q^T$, $C_{int}=V_{int}-P_{int}R^{-1}Q^T$.

Finally find the column of unknown parameters g, setting the derivatives equal from $S(g)$ to g (rules for taking matrix derivatives

$\frac{d\left(x^T b\right)}{dx} = b, \frac{d\left(bx\right)}{dx} = b^T, \frac{d\left(x^T Ax\right)}{dx} = \left(A + A^T\right)x = 2Ax$, the latter is true if A — symmetric matrix).

The expression for the parameters g is as follows:

$$g = \left(C_f^T W_f C_f + \mu C_{df}^T W_{df} C_{df} + \nu C_{d^2 f}^T W_{d^2 f} C_{d^2 f} + \psi C_{\text{int}}^T W_{\text{int}} C_{\text{int}} + \alpha K\right)^{-1} \times \\ \times \left(C_f^T W_f Y_f + \mu C_{df}^T W_{df} Y_{df} + \nu C_{d^2 f}^T W_{d^2 f} Y_{d^2 f} + \psi C_{\text{int}}^T W_{\text{int}} Y_{\text{int}}\right). \tag{17}$$

Knowing g, is calculated $\gamma = R^{-1} Q^T g$, after which you can build spline $g(t)$ at any point t on the (3).

2.4. Select a smoothing option

In all similar function recovery (smoothing) tasks, the selection of the smoothing parameter is discussed separately α. The procedure for automatically selecting this option for recovery tasks could not be found across functions. Classic cross-validation or L-curve procedures will not work. No other work to automatically select the smoothing parameter for many different functionalities. The author had to independently obtain modified formulas to estimate cross-validation.

Recall that the basic idea of cross-validation is to choose such an anti-aliasing parameter α, to recover a function $g(t,\alpha)$ was effective in predicting, i.e. that the function has the least variance when predicting the following values. Therefore, to calculate cross-validation exclude one observation, build spline $g^{(-i)}(t,\alpha)$ and observe the quadratic error this excluded observation is determined, and so do all observations. As a result, the cross validation estimate gives some estimate of the variance of observations as if they were predicted for spline sampled with the element-by-element exception of these observations. Formulas for calculating this estimate are as follows (without intermediate calculations):

$$CV(\alpha) = n_f^{-1} \sum_{i=1}^{n_f} w_i^f \left(y_i - g^{(-i)}(t_i,\alpha)\right)^2 + n_{df}^{-1} \mu \sum_{j=1}^{n_{df}} w_j^{df} \left(y_j' - g'^{(-j)}(t_j,\alpha)\right)^2 + \\ + n_{d^2 f}^{-1} \nu \sum_{l=1}^{n_{d^2 f}} w_l^{d^2 f} \left(y_l'' - g''^{(-l)}(t_l,\alpha)\right)^2 + n_{\text{int}}^{-1} \psi \sum_{u=1}^{n_{\text{int}}} w_u^{\text{int}} \left(Y_u - \int_{t_u^a}^{t_u^b} g^{-u}(t,\alpha)\,dt\right)^2 = \tag{18}$$

$$= n_f^{-1} \sum_{i=1}^{n_f} w_i^f \left(\frac{y_i - g(t_i,\alpha)}{1 - \sum_{k=1}^{m} C_{ik}^f A_{ki}^f(\alpha)}\right)^2 + n_{df}^{-1} \sum_{j=1}^{n_{df}} w_j^{df} \left(\frac{y_j' - g'(t_j,\alpha)}{1 - \sum_{k=1}^{m} C_{jk}^{df} A_{kj}^{df}(\alpha)}\right)^2 + \\ + n_{d^2 f}^{-1} \sum_{l=1}^{n_{d^2 f}} w_l^{d^2 f} \left(\frac{y_l'' - g''(t_l,\alpha)}{1 - \sum_{k=1}^{m} C_{lk}^{d^2 f} A_{kl}^{d^2 f}(\alpha)}\right)^2 + n_{\text{int}}^{-1} \sum_{u=1}^{n_{\text{int}}} w_u^{\text{int}} \left(\frac{Y_u - \int_{t_u^a}^{t_u^b} g(t,\alpha)\,dt}{1 - \sum_{k=1}^{m} C_{uk}^{\text{int}} A_{ku}^{\text{int}}(\alpha)}\right)^2 = \tag{19}$$

$$= n_f^{-1} \sum_{i=1}^{n_f} w_i^f \left(\frac{y_i - \sum_{k=1}^{m} C_{ik}^f g_k}{1 - \sum_{k=1}^{m} C_{ik}^f A_{ki}^f(\alpha)}\right)^2 + n_{df}^{-1} \sum_{j=1}^{n_{df}} w_j^{df} \left(\frac{y_j' - \sum_{k=1}^{m} C_{jk}^{df} g_k}{1 - \sum_{k=1}^{m} C_{jk}^{df} A_{kj}^{df}(\alpha)}\right)^2 + \\ + n_{d^2 f}^{-1} \sum_{l=1}^{n_{d^2 f}} w_l^{d^2 f} \left(\frac{y_l'' - \sum_{k=1}^{m} C_{lk}^{d^2 f} g_k}{1 - \sum_{k=1}^{m} C_{lk}^{d^2 f} A_{kl}^{d^2 f}(\alpha)}\right)^2 + n_{\text{int}}^{-1} \sum_{u=1}^{n_{\text{int}}} w_u^{\text{int}} \left(\frac{Y_u - \sum_{k=1}^{m} C_{uk}^{\text{int}} g_k}{1 - \sum_{k=1}^{m} C_{uk}^{\text{int}} A_{ku}^{\text{int}}(\alpha)}\right)^2. \tag{20}$$

Here the top indexes $(-i),(-j),(-l),(-u)$ mean, that spline $g(t,\alpha)$ estimated from data without this specific observation. Matrices $C^f, C^{df}, C^{d^2f}, C^{int}$ — same as in formulas (16), (17). Matrices $A^f(\alpha)=A(\alpha)C_f^TW_f$, $A^{df}(\alpha)=A(\alpha)\mu C_{df}^TW_{df}$, $A^{d^2f}(\alpha)=A(\alpha)\nu C_{d^2f}^TW_{d^2f}$, $A^{\text{int}}(\alpha)=A(\alpha)\psi C_{\text{int}}^TW_{\text{int}}$, where $A(\alpha)=\left(C_f^TW_fC_f+\mu C_{df}^TW_{df}C_{df}+\nu C_{d^2f}^TW_{d^2f}C_{d^2f}+\psi C_{\text{int}}^TW_{\text{int}}C_{\text{int}}+\alpha K\right)^{-1}$. Minimization $CV(\alpha)$ any known manner relative to α gives the desired smoothing parameter value α.

Cross validation works well on conventional splines. However, we have an unusual case, we are recovering also on the first/second derivatives and integrals. As a result, the cross-validation assessment shows not the usual variance of observations, but the variance of observations of both values and derivatives with integral (as you probably understand, mixing of different variances occurs, but in the construction of the spline we also mixed squares of errors of different observations). But the problem is that if you exclude some observations, you can dramatically change the type of the restored function. In the following examples (especially in the second one) it will be obvious that the exclusion of any observation will result in the function being very imprecise, with the forecast error calculated for the excluded observation. When each observation brings much needed information to restore the function, excluding the survey from the sample will result in very, very large errors. In this case cross-validation is not suitable. But if we have very many observations, some observations only statistically supplement the information of other observations, and the exclusion of one of the observations does not lead to significant changes in the restored function, cross validation method can be a good solution to determine the smoothing parameter.

3. R-LANGUAGE SOFTWARE IMPLEMENTATION

The described function recovery method is implemented in R for many different functional as a function FunctionalSmoothingSpline (see *Appendix 1*). Existing ready-made features and packages in R or other languages implementing similar capabilities could not be found.

4. USE CASE TO PREDICT RARE EVENTS IN AN ECONOMY

4.1. Forecasting future customer purchases

Let there be some customer who buys products from us (for example, an ordinary buyer buys in some trading network, or some wholesaler buys products from the manufacturer). We know nothing about the client, except the dates and volumes of his purchases. For simplicity, we model the replenishment process as in classic stock management models. Let the purchase data will be as follows (*table. 1*).

Table 1

Purchases data of a client not controlled by us

Date t_i	Volume y_i	Date t_i	Volume y_i	Date t_i	Volume y_i	Date t_i	Volume y_i	Date t_i	Volume y_i
03.01.2020	2170	06.06.2020	1976	09.10.2020	2093	10.04.2021	2257	07.08.2021	1968
25.01.2020	2281	19.06.2020	2205	24.10.2020	2141	28.04.2021	2189	19.08.2021	2136
22.02.2020	2242	03.07.2020	2096	12.11.2020	2273	12.05.2021	2026	02.09.2021	2145
11.03.2020	2206	17.07.2020	2125	10.12.2020	2217	24.05.2021	2072	20.09.2021	2235
26.03.2020	2142	29.07.2020	2034	31.12.2020	2218	04.06.2021	1983	07.10.2021	2186
12.04.2020	2210	09.08.2020	1980	21.01.2021	2252	16.06.2021	2059	22.10.2021	2141
30.04.2020	2215	21.08.2020	2098	18.02.2021	2211	30.06.2021	2146	09.11.2021	2256
14.05.2020	2102	05.09.2020	2222	09.03.2021	2218	14.07.2021	2082	07.12.2021	2264
26.05.2020	2115	23.09.2020	2191	24.03.2021	2137	27.07.2021	2177	29.12.2021	2241

Source: compiled by the author.

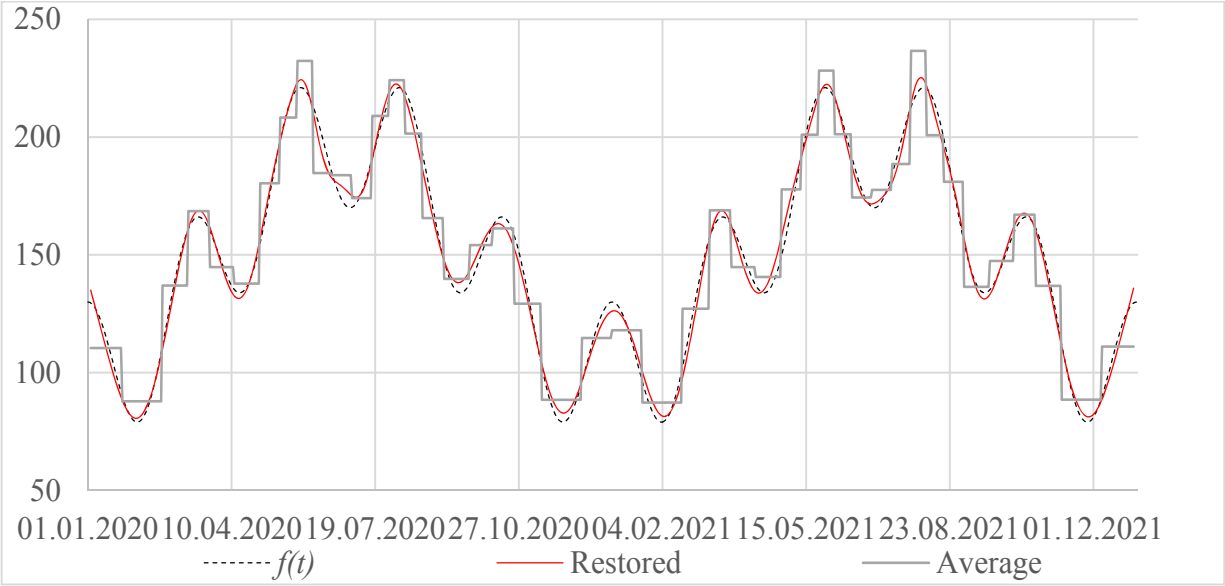

Fig. 1. **The restored intensity of product consumption on the side of the uncontrolled customer**

Source: compiled by the author.

Note: solid line – restored consumption intensity; dashed line – original consumption rate; stepped line – average intensity $y_i / (t_{i+1} - t_i)$ below each step means the purchase volume y_i.

Perceiving this data as a sequence of integrals from an unknown consumption intensity function from the current event to the next, reconstruct this unknown function using the mathematical method described above (*Fig. 1*). *Appendix 2* presents the implementation in R, shows how to prepare the data from the csv file and how to get the function FunctionalSmoothingSpline. For the current data set, the unknown consumption intensity function of the client beyond our control was restored very well.

The next step is to determine the pattern in the identified function, construct the model and extrapolate. Any known method may be used. The very dependence of the restored function can only be built on time, and it is possible to determine the dependence on some known external factors. Here the complete freedom and responsibility of the researcher is assumed. In our case, we will build a model and extrapolate values for the future as the sum of harmonic functions (harmonic function was laid down in the data modelling). Quinn-Fernandes algorithm well-suited to this [11, 12], which represents the sum of a limited number of harmonic functions. The result of this extrapolation is presented in *Fig. 2*. Since the restored function contained some deviations from the true function assumed in the simulation, the parameters of the extrapolated function were determined with some inaccuracy, as a result of which, at some sites, the extrapolated function is noticeably different from the true

The final step is to start the process of determining future events itself. In our case, the consumption process is restarted as in inventory management systems, where extrapolated values are involved as a consumption function. Starting with the most recent observation, it is possible to determine when the stock will run out and thus to give a forecast of the next circulation. If the data determine a further maximum reserve (or build a model for the volume of purchases), it will be possible to predict the entire sequence of future events. *Fig. 2* shows a prediction of future events on the horizontal axis with crosses. And the first few events are determined with an error of only 0–2 days, further the error increases to 6 days, but then again reduced to 2 days (and this is with the time between events 15–28 days). That is, the first events can be defined very well, but when you increase the planning horizon the

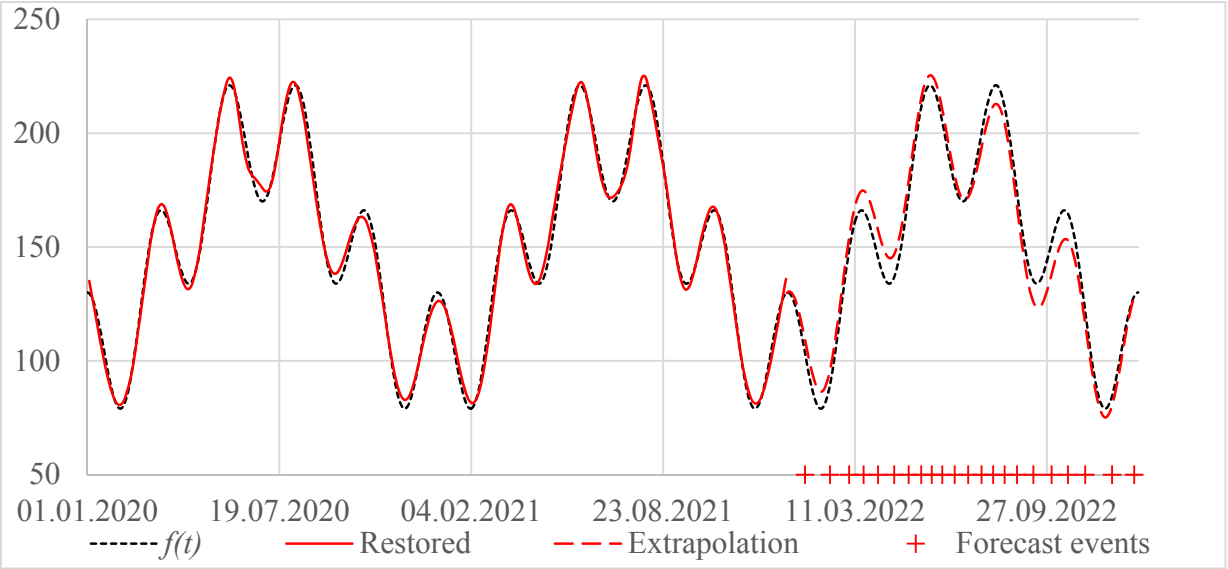

Fig. 2. **Extrapolation of the restored function and forecast of future events**

Source: compiled by the author.

accuracy drops (in this example then improves again), which is logical.

4.2. Determination of the hidden dynamics of the company on communications containing mainly qualitative information

Speculating what other processes of formation of events are in the economy, first of all come to mind those that can be somehow reduced to processes similar to the processes of consumption or the accumulation of disturbances (the depletion of stocks or the accumulation of disturbances to a critical level). Just a reminder that we don't care about random events, where there's no pattern, there's nothing to find. Try to think about what else you think about, whose education process is different from processes of consumption or accumulation of impact. Not immediately, but after a long period of reflection, it was possible to formulate the process of producing rare events in the economy, different from those mentioned, the following will be about one such process. It is worth saying that the process under consideration demonstrates more the possibility of applying the proposed approach to the analysis of rare events than to solve specific practical problems, although there will certainly be (we are interested in the possibility of analyzing rare events formed by processes other than consumption processes).

Let there be some organization beyond our control, from which at certain discrete moments of time there are signals, carrying mainly qualitative (rather than quantitative) information. And in each signal, this qualitative information may relate to different characteristics (not necessarily that each signal reports all the characteristics). For example, imagine a situation where the following signals come from a certain company:

a) We started to sink — we have to do something now!
b) We are still sinking heavily!
c) The decline seems to be slowing down.
d) We started to surface.
e) It's good to go, if we always go like this.
f) Oh no, we're sinking again!
g) We have passed the point of no return...
h) We're at the bottom.

What can be learned from this set of signals? It is possible to guess that events are formed within such an organization by observing some internal indicator that changes dynamically over time (possibly due to some controlling effects). The occurrence of events is related to the operation of comparing this indicator with some critical

values. If we use the proposed method, restore the dynamics of the non-observed internal indicator of the organization, construct a model of change of this dynamic, extrapolate and restart the process of formation of events, it is possible to predict future events (and perhaps even controlling influences).

What do we know from this type of signal that will help to restore the dynamics of the change of the non-observed indicator directly. It turns out that the following data can be extracted from the available signals: moments of time as points of receipt (possibly adjusted for delay in receipt); values of first derivative or position of extremity points; values of second derivative or position of inflection points; values of the most variable in some points. For greater completeness, also assume that integrals from the function of changing the same unobserved indicator can be observed (not included in the above example, but it can be assumed that if the indicator of interest indicates funds in the accounts, the accrued interest on these accounts over some time periods will be these integrals). It should be noted that all observations may be inaccurate, and both the observations themselves and the significance of those observations may be inaccurate. However, the error in the timing of the observation can be reduced in one way or another to an error in the values of the same observations (for example, if the true position of the extremum is slightly shifted from the observation point at which the value of the derivative is shifted from zero).

Denote the unknown search function of hidden dynamics as $f(t)$, which in the future will be restored according to available data in the form of spline. For simplicity let function $f(t)$ will be dimensionless and at the beginning of times t_0 the value of this function will be assumed as the reference value, and all other values will be expressed as a percentage of this reference value (i.e. assume that $f(t_0)=100$). Next, for example, let the original unknown function behave as shown in *Fig. 3*.

Initially it was said that we are dealing with primarily qualitative data, but this qualitative data will not be difficult to give a

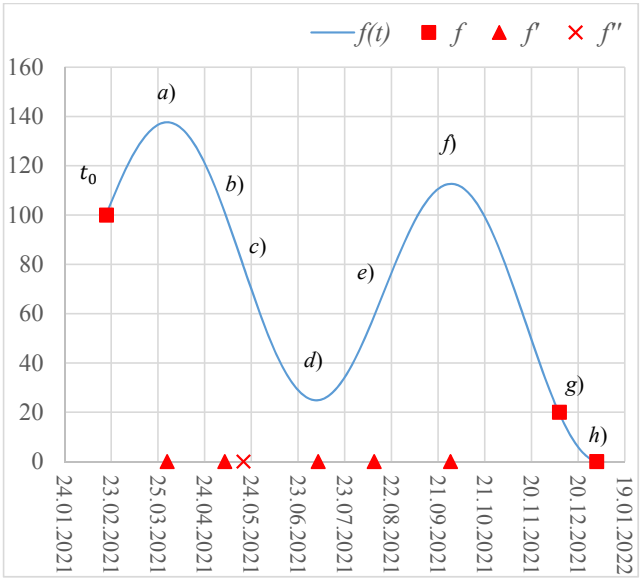

Fig. 3. **An example of the initial function $f(t)$ and the initial data**

Source: compiled by the author.

Note: Function $f(t)$ is dimensionless, shows the number (%) of the initial level (the value at the moment t_0 is taken as the initial). The function value data (events) are marked on the graph itself, the derivatives data are plotted on the horizontal axis and indicate their position.

quantitative estimate, and it is enough to give an approximate estimate (or the values could be initially approximated). So, in events *a*, *c*, *d*, *f* say that there are zero values of the first or second derivatives. For events *b* and *e* suppose we know the tangent of the angle of inclination, for event *g* suppose (or know) that the non-return rate starts at 20%. We gave an approximate estimate of values, and moments of time of events are observed and known. As a result, the available data may be as follows (*Table 2*).

Notice, the sample size can be very different, for the second derivative we have only one observation. Implementing the described function recovery method by functional (all coefficients μ, ν, ψ, α are equal 1), from the available data we will get the following result (*Fig. 4*). Code in R is presented in *Appendix 1*. The task proved poorly conditioned, there was insufficient information for a good function recovery (it was not specified that the function should have increased in the beginning).

If you add more information, for example, as integrals of the desired function, (*Fig. 5*),

Table 2

Available approximate data

t_f	y_f	t_{df}	y_{df}	t_{d^2f}	y_{d^2f}
20.02.2021	100	31.03.2021	0	19.05.2021	0
08.12.2021	20	07.05.2021	–1.75	–	–
01.01.2022	0	06.07.2021	0	–	–
–	–	11.08.2021	1.55	–	–
–	–	29.09.2021	0	–	–

Source: compiled by the author.

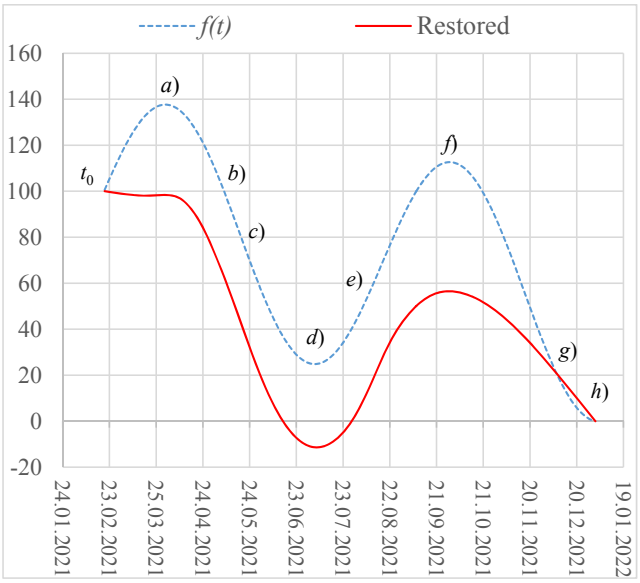

Fig. 4. **Poorly conditioned task. There was not enough information**

Source: compiled by the author.

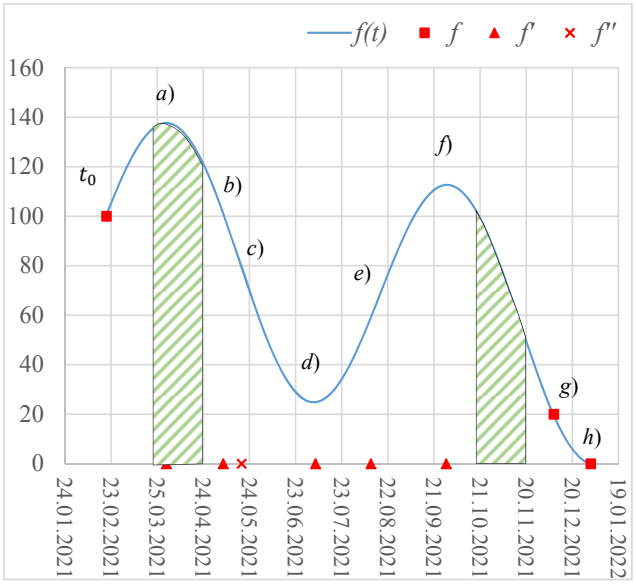

Fig. 5. **The original function $f(t)$ and the initial data with the additional information about the integrals**

Source: compiled by the author.

(*Table. 3*), then restore the original function gets much more accurate (*Fig. 6*).

Of course, recovery is still not perfect, but it is due to a lack of information, not a lack of mathematical method. If you add more observations, you can get a well recovery. Note that by adjusting the weights of both observations and observation groups, it is possible to adjust very flexibly which information should be paid more attention and which less.

Further forecasting of future events follows the same pattern as in the previous example (in this example we may not expect any future events).

5. AN APPROACH SCHEME FOR PREDICTING RARE EVENTS

Once again, separately, we give the idea of the approach of predicting rare events. If from rare events it is possible to restore the parameters of the process (dynamic), then further actions to predict rare events will be to identify the regularity of changes in these parameters of the process (their dynamics) over time. And to do so, all possible information should be used, patterns can be determined depending on changes in external observable factors, such as GDP, inflation, unemployment and other factors. Any existing mathematical methods can be used for this purpose. The purpose of

Table 3

Data with added information about integrals

t_f	y_f	t_{df}	y_{df}	t_{d^2f}	y_{d^2f}	t^a_{int}	t^b_{int}	Y_{int}
20.02.2021	100	31.03.2021	0	19.05.2021	0	25.03.2021	24.04.2021	4000
08.12.2021	20	07.05.2021	–1,75	–	–	21.10.2021	20.11.2021	2282
01.01.2022	0	06.07.2021	0	–	–	–	–	–
–	–	11.08.2021	1,55	–	–	–	–	–
–	–	29.09.2021	0	–	–	–	–	–

Source: compiled by the author.

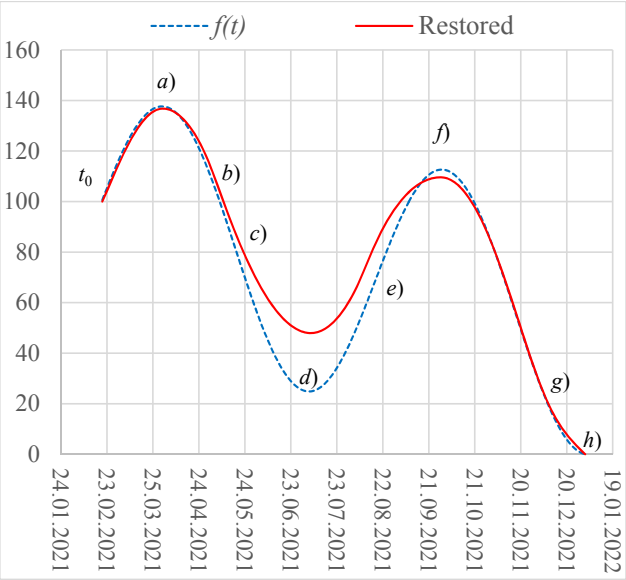

Fig. 6. **Restoration of a function from data with the additional information about integrals**

Source: compiled by the author.

this step will be to construct an appropriate model of changing the internal parameters of the process. The next step is to extrapolate process parameters for the future. If previously a model of dependence on external factors is built, it is necessary to extrapolate the values of external factors beforehand. Any known mathematical method can be used again for the extrapolation phase. Finally, if we have an idea of how the parameters of the process of making events will change in the future, and we can reproduce the mechanism of this process, it will be easy to get a forecast of future events by running the process itself with set parameters. The scheme of obtaining a forecast of future events is presented at *Fig. 7*.

The proposed approach cannot be directly compared with other methods. The fact is that none of the existing methods can predict when a future event will occur.

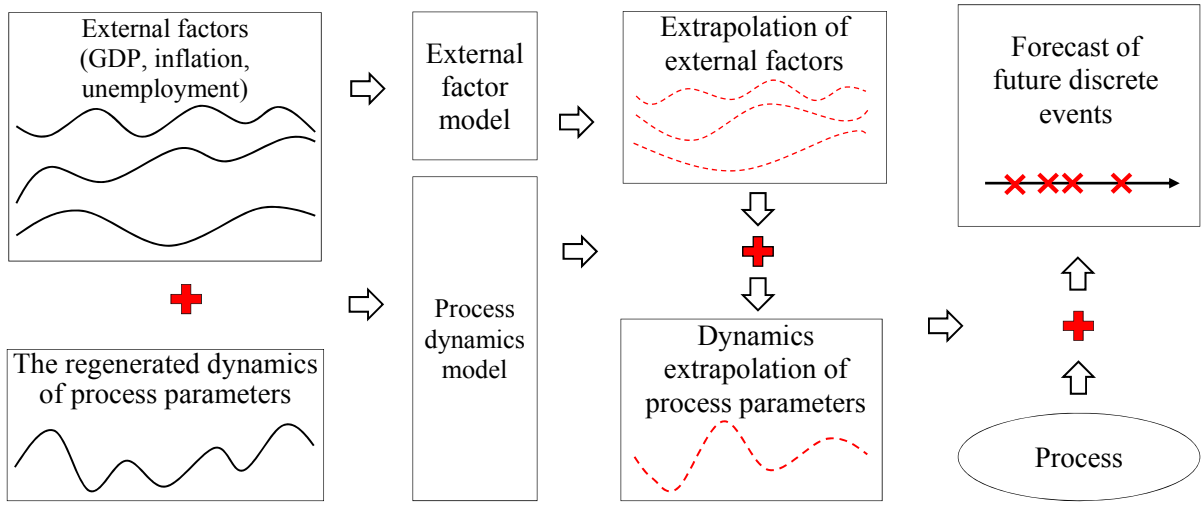

Fig. 7. **Scheme of the approach for predicting rare events in the economy**

Source: compiled by the author.

Some methods predict the occurrence of a given number of events in a certain amount of time or determine only the probability of one event in the next time interval (statistical methods based on Poisson and other flows of events, methods of analysis of irregular demand). Other methods measure the probability of an event if a sequence is observed among observed external factors or lag variables (all possible classification methods). Moreover, in the described approach it is proposed to build a model and extrapolate the restored dependencies of internal parameters over time, when in other methods all estimates are obtained static. In order to compare the proposed approach of prediction of events with other methods it will be necessary to adapt it to specific tasks of other methods. That is, with the help of the method it is necessary to give a forecast at once to a set of future events in an interval of time, and immediately from a multitude of sources (from different clients). This mixed prediction of a group of events can be compared with the prediction of other methods that work with events presented in the form of time series. However, it should be borne in mind that the proposed approach, in addition to the phase of recovery of internal parameters of the process, there are stages of identification of patterns and extrapolation of dynamics of the process, carried out by any known mathematical method. Depending on which methods the researcher chooses, the effectiveness of the proposed approach can be evaluated in different ways. In order to compare performance with other methods, a separate volumetric study is necessary, and maybe even not one.

CONCLUSION

As a result of this study, it was possible to develop a mathematical method of function recovery simultaneously for many different functionalities, taking into account the error in the observation of these functionalities. In *Appendix 1* the software implementation of the method in R language is presented. Due to the described method it is possible to analyze and predict rare events that are caused by some processes. In case it is processes of consumption, it is enough to consider data (purchases, credits, etc.) as integral and restore dependencies from this sequence of integrals. If these are more complex processes, some data can also be considered as first or second derivatives. If we can reconstruct the dynamics of the process of formation of rare events from the data available on these events, the next step will be to determine the pattern and extrapolate this dynamics to the future. And at this stage the researcher is not limited and can use any suitable mathematical method. After the extrapolation stage, it is possible to start the process of formation of events with set values of internal parameters and to get a forecast of future events in the economy. The described approach can be applied in various areas, for example, by analyzing the data on the haircut of some regular customer at the hairdresser, you can restore the function of the rate of accumulation of desire to cut [13], and analyzing the historical data of the Russo-Turkish wars, you can get the speed of the build-up of disagreements or the speed of preparation for another war [14] (the latter has a more demonstrative character). The analysis and prediction of rare events is very important. This will allow to prepare for such events, gain some benefit from it or reduce possible risks or losses.

ACKNOWLEDGEMENTS

The study was funded by the Russian Foundation for Basic Research (RFBR), project No. 19–010–00154. Financial University, Moscow, Russia.

REFERENCES

1. Shannon C. Works on information theory and cybernetics. Transl. from Eng. Moscow: Foreign Literature Publ.; 1963. 258 p. (In Russ.).
2. Croston J. D. Forecasting and stock control for intermittent demands. *Operational Research Quarterly*. 1972;23(3):289–303. DOI: 10.1057/jors.1972.50
3. Willemain T.R., Smart C.N., Schwarz H.F. A new approach to forecasting intermittent demand for service parts inventories. *International Journal of Forecasting*. 2004;20(3):375–387. DOI: 10.1016/S 0169–2070(03)00013-X
4. Kaya G.O., Sahin M., Demirel O.F. Intermittent demand forecasting: A guideline for method selection. *Sādhanā: Academy Proceedings in Engineering Sciences*. 2020;45(1):51. DOI: 10.1007/s12046–020–1285–8
5. Pinçe Ç., Turrini L., Meissner J. Intermittent demand forecasting for spare parts: A critical review. *Omega*. 2021;105:102513. DOI: 10.1016/j.omega.2021.102513
6. Korablev Yu.A. The function restoration method by integrals for analysis and forecasting of rare events in the economy. *Ekonomika i matematicheskie metody = Economics and Mathematical Methods*. 2020;56(3):113–124. (In Russ.). DOI: 10.31857/S 042473880010485–2
7. Kimeldorf G., Wahba G. Some results on Tchebychefian spline functions. *Journal of Mathematical Analysis and Applications*. 1971;33(1):82–95. DOI: 10.1016/0022–247X(71)90184–3
8. Lee C.H. A phase space spline smoother for fitting trajectories. *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*. 2004;34(1):346–356. DOI: 10.1109/tsmcb.2003.817027
9. Xian J., Li Y., Lin W. Reconstruction of signal from samples of its integral in spline subspaces. In: Bubak M., van Albada G.D., Sloot P.M.A., Dongarra J., eds. Int. conf. on computational science: Computational science (ICCS 2004). Berlin, Heidelberg: Springer-Verlag; 2004:574–577. (Lecture Notes in Computer Science. Vol. 3037). DOI: 10.1007/978–3–540–24687–9_75
10. Green P.J., Silverman B.W. Nonparametric regression and generalized linear models. A roughness penalty approach. Boca Raton, London: Chapman & Hall/CRC; 1994. 194 p. (Monographs on Statistics and Applied Probability. No. 58).
11. Quinn B.G., Fernandes J.M. A fast efficient technique for the estimation of frequency. *Biometrika.* 1991;78(3):489–497. DOI: 10.1093/biomet/78.3.489
12. Quinn B.G., Hannan E.J. The estimation and tracking of frequency. Cambridge, New York: Cambridge University Press; 2001. 278 p. (Cambridge Series in Statistical and Probabilistic Mathematics. No. 9).
13. Korablev Yu.A., Golovanova P.S., Kostritsa T.A. Capacity method of rare events analysis in the area of services. *Ekonomicheskaya nauka sovremennoi Rossii = Economics of Contemporary Russia*. 2020;(3):132–142. (In Russ.). DOI: 10.33293/1609–1442–2020–3(90)-132–142
14. Korablev Yu.A., Golovanova P.S., Kostritsa T.A. Using the capacity method to analyze historical events. *KANT*. 2021;(1):27–32. (In Russ.). DOI: 10.24923/2222–243X.2021–38.6

Appendix 1

Implementation in R

Functionality recovery is implemented as a FunctionalSmoothingSpline function, at the input of which the available data is supplied.

```
FunctionalSmoothingSpline = function (
                        t_f = NULL,          # array of observation moments
                        values_f = NULL,     # array of observation values
                        weights_f = NULL,    # array of  weights
                        t_df = NULL,         # array of first derivative moments
                        values_df = NULL,    # array of first derivative values
                        weights_df = NULL,   # array of first derivative weights
                        coef_df = 1,         # coefficient of first derivative sum of squares
                        t_d2f = NULL,        # array of second derivative moments
                        values_d2f = NULL,   # array of second derivative values
                        weights_d2f = NULL,  # array of second derivative weights
                        coef_d2f = 1,        # coefficient of second derivative sum of squares
                        t_int_a = NULL,      # array of interals start moments
                        t_int_b = NULL,      # array of interals end moments
                        values_int = NULL,   # array of interal values
                        weights_int = NULL,  # array of interal weights
                        coef_int = 1,        # coefficient of integral sum of squares
                        knots = NULL,        # knots
                        knots_number = NULL, # number of knots
                        alpha = 1,           # smoothing parameter
                        x = NULL,            # output moments
                        info = FALSE)        # need info?
{

        if (is.null(knots_number) & is.null(knots))
        {
                knots_number = 0
                if (!is.null(t_f) & length(t_f)>0)
                        knots_number = length(t_f)
                if (!is.null(t_df) & length(t_df)>0 & knots_number<length(t_df))
                        knots_number = length(t_df)
                if (!is.null(t_d2f) & length(t_d2f)>0 & knots_number<length(t_d2f))
                        knots_number = length(t_d2f)
                if (!is.null(t_int_a) & length(t_int_a)>0 & knots_number<length(t_int_a))
                        knots_number = length(t_int_a)
        }

        if (knots_number<2)
                stop('knots_number or observations should not be less than 2')

        m = knots_number # for short

        # in case knots is not defined
        if (m != length(knots)) # when knots_number defined, but knots not defined
        {
                start_knot = + Inf
                end_knot = - Inf
                if (!is.null(t_f) & length(t_f)>0 )
                {
```

```
                if (start_knot>min(t_f))
                        start_knot = min(t_f)
                if (end_knot<max(t_f))
                        end_knot = max(t_f)
        }
        if (!is.null(t_df) & length(t_df)>0)
        {
                if (start_knot>min(t_df))
                        start_knot = min(t_df)
                if (end_knot<max(t_df))
                        end_knot = max(t_df)
        }
        if (!is.null(t_d2f) & length(t_d2f)>0)
        {
                if (start_knot>min(t_d2f))
                        start_knot = min(t_d2f)
                if (end_knot<max(t_d2f))
                        end_knot = max(t_d2f)
        }
        if (!is.null(t_int_a) & length(t_int_a)>0 & length(t_int_b)>0)
        {
                if (start_knot>min(t_int_a))
                        start_knot = min(t_int_a)
                if (end_knot<max(t_int_b))
                        end_knot = max(t_int_b)
        }
        knots=seq(start_knot,end_knot,length = m)
}

h = array(0,dim = m - 1) #array of distance between knots
h[1:(m - 1)] = knots[2:m] - knots[1:(m - 1)]

#Matrix Q
Q=matrix(0, nrow = m, ncol = m - 2)
for (i in 1:(m - 2))
{
        Q[i,i] = 1/h[i];
        Q[i + 1,i] = - 1/h[i] - 1/h[i + 1];
        Q[i + 2,i] = 1/h[i + 1]
}

#Matrix R
R = matrix(0, nrow = m - 2, ncol = m - 2)
for (i in 1:(m - 2))
{
        R[i,i] = 1/3*(h[i] + h[i + 1]);
        if (i<m - 2)
        {
                R[i + 1,i] = 1/6*h[i + 1];
                R[i,i + 1] = 1/6*h[i + 1];
        }
}
        #Matrix K calculation
```

```
inv_R = solve(R)
t_Q = t(Q)
K = Q %*% inv_R %*% t_Q

# ========= 1. Observation (t_f, values_f)  ===========

if (! is.null(t_f) & length(t_f)>0)
{
        nf = length(t_f) #number of observation coordinates
        if (length(values_f) != nf)
                stop('length of values_f and t_f must be same')
        if (is.null(weights_f))
                weights_f = rep(1,nf)
        if (length(weights_f) != nf)
                stop('length of weights_f and t_f must be same')
        Wf = diag(weights_f)

        #reorder observations (t_f, values_f) by appear time t_f
        ord = order(t_f,values_f)
        t_f = t_f[ord]
        values_f = values_f[ord]

        #Filling in Vf and Pf matrices
        Vf = matrix(0,nrow = nf, ncol = m)
        Pf = matrix(0,nrow = nf, ncol = m)
        k = 1 # start knot
        for (i in 1:nf)
        {
                while( (knots[k]<=t_f[i]) & (knots[k+1]<t_f[i]) & (k<knots_number)) #find first k, that
knots[k+1]>t_f[i]
                        k = k+1
                hk_m = t_f[i] - knots[k]
                hk_p = knots[k+1] - t_f[i]
                Vf[i,k] = hk_p/h[k]
                Vf[i,k + 1] = hk_m/h[k]
                Pf[i,k] = hk_m*hk_p*(h[k] + hk_p)/(6*h[k])
                Pf[i,k + 1] = hk_m*hk_p*(h[k] + hk_m)/(6*h[k])
        }
        Pf = Pf[1:nf,2:(m - 1)] #don't need first and last column

        #Matrix Cf calculation
        Cf = Vf - Pf %*% inv_R %*% t_Q
        t_Cf = t(Cf)
}

# ========= 2. Observation (t_df, values_df)  ===========

if (! is.null(t_df) & length(t_df)>0)
{

        ndf=length(t_df) #number of observation
        if (length(values_df) != ndf)
                stop('length of values_df and t_df must be same')
```

```
        if (is.null(weights_df))
                weights_df = rep(1,ndf)
        if (length(weights_df)!=ndf)
                stop('length of weights_df and t_df must be same')
        Wdf = diag(weights_df)

        ord = order(t_df,values_df)   #reorder observations (t_df, values_df)  by appear time t_df
        t_df = t_df[ord]
        values_df = values_df[ord]

        #Filling in Vdf and Pdf matrices
        Vdf = matrix(0, nrow = ndf, ncol = m)
        Pdf = matrix(0, nrow = ndf, ncol = m)
        k = 1 # start knot
        for (i in 1:ndf)
        {
                while( (knots[k]<=t_df[i]) & (knots[k+1]<t_df[i]) & (k<m)) #find first k, that knots[k +
1]>t_df[i]
                        k = k + 1
                hk_m = t_df[i] - knots[k]
                hk_p = knots[k + 1] - t_df[i]

                Vdf[i,k] = - 1/h[k]
                Vdf[i,k + 1] = 1/h[k]
                Pdf[i,k] = - h[k]/6+(hk_p)^2/(2*h[k])
                Pdf[i,k + 1] = h[k]/6-(hk_m)^2/(2*h[k])
        }
        Pdf = Pdf[1:ndf,2:(m-1)] #don't need first and last column

        #Matrix Cdf calculation
        Cdf = Vdf - Pdf %*% inv_R %*% t_Q
        t_Cdf = t(Cdf)

}

# ========= 3. Observation (t_d2f, values_d2f)  ===========

if (! is.null(t_d2f) & length(t_d2f)>0)
{

        nd2f = length(t_d2f) #number of observation
        if (length(values_d2f) != nd2f)
                stop('length of values_d2f and t_d2f must be same')
        if (is.null(weights_d2f))
                weights_d2f = rep(1,nd2f)
        if (length(weights_d2f) != nd2f)
                stop('length of weights_d2f and t_d2f must be same')
        Wd2f = diag(weights_d2f)

        #reorder observations (t_d2f, values_d2f)  by appear time t_d2f
        ord = order(t_d2f,values_d2f)
        t_d2f = t_d2f[ord]
        values_d2f = values_d2f[ord]
```

```
                #Filling in Pd2f matrices
                Pd2f=matrix(0, nrow = nd2f,ncol = m)
                k = 1 # start knot
                for (i in 1:nd2f)
                {
                        while( (knots[k]<=t_d2f[i]) & (knots[k+1]<t_d2f[i]) & (k<m)) #find first k, that knots[k+1]>t_d2f[i]
                                k = k + 1
                        hk_m = t_d2f[i] - knots[k]
                        hk_p = knots[k + 1] - t_d2f[i]

                        Pd2f[i,k] = - hk_p/h[k]
                        Pd2f[i,k+1] = - hk_m/h[k]
                }
                Pd2f = Pd2f[1:nd2f,2:(m - 1)] #don't need first and last column

                #Matrix Cd2f calculation
                Cd2f = - Pd2f %*% inv_R %*% t_Q
                t_Cd2f = t(Cd2f)

        }

        # ========= 4. Observation (t_int_a, t_int_b, values_int)  ===========

        if (! is.null(t_int_a) & length(t_int_a)>0)
        {
                nint=length(t_int_a) #number of observation
                if (length(t_int_b) != nint)
                        stop('length of t_int_b and t_int_a must be same')
                #if (length(values_int) != nint)
                        stop('length of values_int and t_int_a must be same')

                if (is.null(weights_int))
                        weights_int = rep(1,nint)
                if (length(weights_int) != nint)
                        stop('length of weights_int and t_int_a must be same')
                Wint=diag(weights_int)

                #reorder observations (t_int_a, t_int_b, values_int) by appear time t_int_a
                ord = order(t_int_a, t_int_b,values_int)
                t_int_a = t_int_a[ord]
                t_int_b = t_int_b[ord]
                values_int = values_int[ord]

                #Filling in Vint and Pint matrices
                Vint = matrix(0, nrow = nint, ncol = m)
                Pint = matrix(0, nrow = nint, ncol = m)
                k = 1 # start knot
                for (i in 1:nint)
                {
                        while( (knots[k]<=t_int_a[i]) & (knots[k + 1]<t_int_a[i]) & (k<m)) #find first k, that knots[k + 1]>t_int_a[i]
                                k = k + 1
```

```
                #finding L, it can be 0
                for (L in 0:(m - k - 1))
                        if (t_int_b[i] <= knots[k + L + 1])
                                break;
                l = 1;
                hk_m = t_int_a[i] - knots[k]
                hk_p=knots[k + 1] - t_int_a[i]
                hkL_m = t_int_b[i] - knots[k + L]
                hkL_p = knots[k + L + 1] - t_int_b[i]

                Vint[i,k] = (hk_p)^2/h[k]/2
                Pint[i,k] = h[k]^3/24 - (hk_m)^2*(hk_p + h[k])^2/h[k]/24
                while (l<=L)
                {
                        Vint[i, k + l] = (h[k + l - 1] + h[k + l])/2
                        Pint[i, k + l] = (h[k + l - 1]^3 + h[k + l]^3)/24
                        l = l + 1;
                }
                Vint[i, k + 1] = Vint[i, k + 1] - (hk_m)^2/h[k]/2
                Pint[i, k + 1] = Pint[i, k + 1] + (hk_m)^2*((hk_m)^2 - 2*h[k]^2)/h[k]/24
                Vint[i, k + L] = Vint[i, k + L] - (hkL_p)^2/h[k + L]/2
                Pint[i, k + L] = Pint[i, k + L] + (hkL_p)^2*((hkL_p)^2 - 2*h[k + L]^2)/h[k + L]/24
                Vint[i, k + L + 1] = (hkL_m)^2/h[k + L]/2
                Pint[i, k + L + 1] = h[k + L]^3/24 - (hkL_p)^2*(hkL_m + h[k + L])^2/h[k + L]/24

        }
        Pint=Pint[1:nint,2:(m - 1)] #don't need first and last column

        #Matrix Cint calculation
        Cint = Vint - Pint %*% inv_R %*% t_Q
        t_Cint = t(Cint)

}

# ============ Calculation =============

# matrix A
A = alpha * K
if (! is.null(t_f) & length(t_f)>0)
        A = A + t_Cf %*% Wf %*% Cf
if (! is.null(t_df) & length(t_df)>0)
        A = A + coef_df * t_Cdf %*% Wdf %*% Cdf
if (! is.null(t_d2f) & length(t_d2f)>0)
        A = A + coef_d2f * t_Cd2f %*% Wd2f %*% Cd2f
if (! is.null(t_int_a) & length(t_int_a)>0)
        A = A + coef_int * t_Cint %*% Wint %*% Cint

# matrix D
D = matrix(0, nrow = m, ncol = 1)
if (! is.null(t_f) & length(t_f)>0)
        D = D + t_Cf %*% Wf %*% values_f
if (! is.null(t_df) & length(t_df)>0)
```

```
            D = D + coef_df * t_Cdf %*% Wdf %*% values_df
        if (! is.null(t_d2f) & length(t_d2f)>0)
            D = D + coef_d2f * t_Cd2f %*% Wd2f %*% values_d2f
        if (! is.null(t_int_a) & length(t_int_a)>0)
            D = D + coef_int * t_Cint %*% Wint %*% values_int

        #Calculation of g and gamma
        g = solve(A , D)
        gamma = inv_R %*% t_Q %*% g   #After that spline is completely defined via g and gamma


        # ===== Calculating and returning spline values at x coordinates  =====

        g2 = c(0,gamma,0) #Second derivative on the edges was zero

        if (is.null(x))
            x = seq(knots[1],knots[m],by=1)

        y = rep(0,length(x))

        k = 1; #index of interval
        for (j in (1:length(x)))
        {
            while (x[j]>knots[k]+h[k] & k<m)
                k = k + 1;
            hk_m = x[j] - knots[k]
            hk_p = knots[k + 1] - x[j]
            y[j] = (hk_m*g[k + 1] + hk_p*g[k])/h[k] - 1/6*hk_m*hk_p*(g2[k + 1]*(1 + hk_m/h[k]) + g2[k]*(1 +
hk_p/h[k]) )
        }

        if (info)
        {
            error_total = 0
            error_f = 0
            error_df = 0
            error_d2f = 0
            error_int = 0
            error_penalty = 0
            fraction_error_f = 0
            fractio_error_df = 0
            fractio_error_d2f = 0
            fractio_error_int = 0
            fractio_penalty = 0
            relative_sqr_error_f = 0
            relative_sqr_error_df = 0
            relative_sqr_error_d2f = 0
            relative_sqr_error_int = 0
            relative_abs_error_f = 0
            relative_abs_error_df = 0
            relative_abs_error_d2f = 0
            relative_abs_error_int = 0
```

```
if (! is.null(t_f) & length(t_f)>0)
{
        V = values_f - Cf %*% g
        error_f = t(V) %*% Wf %*% V
        V = abs(V / values_f)
        relative_abs_error_f = (t(V) %*% Wf %*% V) / nf
        V = V^2
        relative_sqr_error_f = sqrt((t(V) %*% Wf %*% V) / nf)
}
if (! is.null(t_df) & length(t_df)>0)
{
        V = values_df - Cdf %*% g
        error_df = t(V) %*% Wdf %*% V
        V = abs(V / values_df)
        relative_abs_error_df = (t(V) %*% Wdf %*% V) / ndf
        V = V^2
        relative_sqr_error_df = sqrt((t(V) %*% Wdf %*% V) / ndf)
}
if (! is.null(t_d2f) & length(t_d2f)>0)
{
        V = values_d2f - Cd2f %*% g
        error_d2f = t(V) %*% Wd2f %*% V
        V = abs(V / values_d2f)
        relative_abs_error_d2f = (t(V) %*% Wd2f %*% V) / nd2f
        V = V^2
        relative_sqr_error_d2f = sqrt((t(V) %*% Wd2f %*% V) / nd2f)
}
if (! is.null(t_int_a) & length(t_int_a)>0)
{
        V = values_int - Cint %*% g
        error_int = t(V) %*% Wint %*% V
        V = abs(V / values_int)
        relative_abs_error_int = (t(V) %*% Wint %*% V) / nint
        V = V^2
        relative_sqr_error_int = sqrt((t(V) %*% Wint %*% V) / nint)
}
error_penalty = t(g) %*% K %*% g

error_total = error_f + coef_df*error_df + coef_d2f*error_d2f + coef_int*error_int + alpha*error_
penalty

fraction_error_f = error_f/error_total
fraction_error_df = coef_df*error_df/error_total
fraction_error_d2f = coef_d2f*error_d2f/error_total
fraction_error_int = coef_int*error_int/error_total
fraction_penalty = alpha*error_penalty/error_total

result = list(      x = x,
                    y = y,
                    error_total = error_total,
                    error_f = error_f,
                    error_df = error_df,
                    error_d2f = error_d2f,
                    error_int = error_int,
```

```
                                    error_penalty = error_penalty,
                                    fraction_error_f = fraction_error_f,
                                    fraction_error_df = fraction_error_df,
                                    fraction_error_d2f = fraction_error_d2f,
                                    fraction_error_int = fraction_error_int,
                                    fraction_penalty = fraction_penalty,
                                    relative_sqr_error_f = relative_sqr_error_f,
                                    relative_sqr_error_df = relative_sqr_error_df,
                                    relative_sqr_error_d2f = relative_sqr_error_d2f,
                                    relative_sqr_error_int = relative_sqr_error_int,
                                    relative_abs_error_f = relative_abs_error_f,
                                    relative_abs_error_df = relative_abs_error_df,
                                    relative_abs_error_d2f = relative_abs_error_d2f,
                                    relative_abs_error_int = relative_abs_error_int
                            )
        }
        else
                result = y
        return (result) }
```

Appendix 2

Computing in R

Example 1

Read data from a CSV file (which contains all corresponding columns):

```
#================ Data input ===================
library(lubridate)
filename = « F:/DIR/Sales.csv»;

#if CSV file was generated by Excel
MyData <- read.csv(file = filename, header = TRUE, sep = «;», stringsAsFactors = FALSE, dec =»,»)
t = as.numeric(dmy(MyData[[1]]))

#if CSV file was generated by R
#MyData <- read.csv(file = filename, header = TRUE, sep = «,», stringsAsFactors = FALSE, dec =».»)
#t = as.numeric(ymd(MyData[[1]]))
```

Deleting missing values:

```
# ================= Remove NA ====================
t = t[!is.na(t)]
n = length(t)
Y = MyData[[2]]
Y = Y[1:(n - 1)] # Last value not used
origin = dmy(MyData[[1]][1]) - t[1] #will need time origin for x-axis labes
```

Specify the number of nodes, you can take several times more than the number of all observations:

```
m = round(n*3)
```

Function set, result is saved to variable r. If Info = FALSE, the calculated values will be displayed immediately y.

```
#=========== Calculating spline =============
r = FunctionalSmoothingSpline(#t_f = t_f,
                #values_f = y_f,
```

```
                #weights_f = NULL,
                #t_df = t_df,
                #values_df = y_df,
                #weights_df = NULL,
                #coef_df = 1,
                #t_d2f = t_d2f,
                #values_d2f = y_d2f,
                #weights_d2f = NULL,
                #coef_d2f = 1,
                t_int_a = t[1:(n-1)],
                t_int_b = t[2:n],
                values_int = Y,
                weights_int = W,
                #coef_int = 1,
                #knots = NULL,
                knots_number = m,
                alpha = 10^(4),
                info = TRUE)
#r
y = r$y
```

Rendering a chart:

```
#================= Plotting spline with graph of average values ==============
x = r$x
#x = seq(t[1], t[n], by = 1) # In case Info = FALSE
x2 = t[1]:t[n] #for graph of average values
y2 = rep(0, length(x2))
for (i in 1:(n - 1))
  for (j in t[i]:t[i + 1])
    y2[j-t[1] + 1] = Y[i]/(t[i + 1] - t[i])


plot(x, y, col = “red”, type = “l”, lwd = “1”, lty = 1, xaxt = “n”, ylim = range(c(y,y2)), xlim = range(c(x,x2)))
axis.Date(1, at=seq(min(dmy(MyData[[1]])), max(dmy(MyData[[1]])), by = “months”), format = “%m-%Y”)
lines(x2, y2, col = “black”, type = “l”, lwd = “1”, lty = 1)
```

File output:

```
#================ Data output ===================
MyWriteData = data.frame(t = x + origin, Value = y, x2 = x2 + origin, y2 = y2)
s2 = “F:/DIR/f_spline_out.csv”;
write.csv(MyWriteData, file = s2,row.names=FALSE)
```

Example 2

Read data from a CSV file (which contains all corresponding columns):

```
#================ Data input ===================

Library (lubridate)
filename = “F:/Dir/DiscrSignals.csv”;

#if CSV file was generated by Excel
```

```
MyData < – read.csv(file = filename, header = TRUE, sep = ";", stringsAsFactors = FALSE, dec = ",")
t_f = as.numeric(dmy(MyData[[1]]))
y_f = MyData[[2]]
t_df = as.numeric(dmy(MyData[[3]]))
y_df = MyData[[4]]
t_d2f = as.numeric(dmy(MyData[[5]]))
y_d2f = MyData[[6]]
t_int_a = as.numeric(dmy(MyData[[7]]))
t_int_b = as.numeric(dmy(MyData[[8]]))
y_int = MyData[[9]]

#if CSV file was generated by R
#MyData < – read.csv(file = filename, header = TRUE, sep =",", stringsAsFactors = FALSE, dec =".")
#t_f = as.numeric(ymd(MyData[[1]]))
#t_df = as.numeric(ymd(MyData[[3]]))
#t_d2f = as.numeric(ymd(MyData[[5]]))
#t_int_a = as.numeric(ymd(MyData[[7]]))
#t_int_b = as.numeric(ymd(MyData[[8]]))
```

Delete missing values at the very end (CSV file had different column lengths, but missing values still read):

```
# ================= Remove NA ====================
t_f = t_f[!is.na(t_f)]
nf = length(t_f)
y_f = y_f[1:nf]
t_df = t_df[!is.na(t_df)]
ndf = length(t_df)
y_df = y_df[1:ndf]
t_d2f = t_d2f[!is.na(t_d2f)]
nd2f = length(t_d2f)
y_d2f = y_df[1:nd2f]
t_int_a = t_int_a[!is.na(t_int_a)]
nint = length(t_int_a)
t_int_b = t_int_b[!is.na(t_int_b)]
y_int = y_int[1:nint]

#will need time origin for x-axis labes
origin = dmy(MyData[[1]][1]) – t_f[1]
#origin = ymd(MyData[[1]][1]) – t_f[1]
```

Specify the number of nodes, you can take several times more than the number of all observations:

```
m = round(3*(nf + ndf + nd2f + nint))
```

Function set, result is saved to variable r. If Info = FALSE, the calculated values will be displayed immediately y:

```
#========== Calculating spline =============
r = FunctionalSmoothingSpline( t_f = t_f,
                               values_f = y_f,
                               t_df = t_df,
                               values_df = y_df,
```

```
                                    t_d2f = t_d2f,
                                    values_d2f = y_d2f,
                                    t_int_a = t_int_a,
                                    t_int_b = t_int_b,
                                    values_int = y_int,
                                    knots_number = m,
                                    alpha = 10^(0),
                                    info = TRUE)
y = r$y
```

Rendering a chart:

```
#============================ Plotting spline ============================
x = r$x
#x = seq (min(t_f, t_df, t_d2f, t_int_a), max (t_f, t_df, t_d2f, t_int_b), by = 1) # In case Info = FALSE
xrange = range (x)
yrange = range (y)
plot(x, y, col = "red", type = "l", lwd = "1", lty = 1,  xaxt = "n", ylim = yrange, xlim = xrange)
axis (1, at = c (t_f,t_df,t_d2f,t_int_a,t_int_b), labels = as. Date (c(t_f,t_df,t_d2f,t_int_a,t_int_b), origin=origin))
#lines(x,y2,col = "black", type = "l", lwd="2", lty = 1) # In case you want add some precalculated original function y2
```

File ouput:

```
#================ Data output ===================
#write to the csv file
MyWriteData = data.frame(x + origin, y)
s2 = «F:/DIR/f_spline_out.csv»
write.csv (MyWriteData, file = s2,row.names = FALSE
```

ABOUT THE AUTHOR

Yurii A. Korablev — Cand. Sci. (Econ.), Assoc. Prof., Department of System Analysis in Economics, Financial University, Moscow, Russia
http://orcid.org/0000-0001-5752-4866
yura-korablyov@yandex.ru

Conflicts of Interest Statement: The author has no conflicts of interest to declare.

The article was submitted on 21.10.2021; revised on 08.11.2021 and accepted for publication on 25.02.2022.
The author read and approved the final version of the manuscript.