Reliable Computing 1 (2) (1995), pp. 141–172

Applications of interval computations to earthquake-resistant engineering: How to compute derivatives of interval functions fast

Vladik Kreinovich, David Nemir, and Efren Gutierrez

One of the main sources of destruction during earthquake is resonance. Therefore, the following idea has been proposed. We design special control linkages between floors that are normally unattached to the building but can be attached if necessary. They are so designed that adding them changes the building's characteristic frequency. We continuously monitor displacements within the structure, and when they exceed specified limits, the linkages are engaged in a way to control structural motion. This idea can also be applied to avoid vibrational destruction of large aerospace structures.

Приложение интервальных вычислений к сейсмоустойчивой инженерии: как быстро вычислить производные интервальных функций

В. Крейнович, Д. Немир, Е. Гутиэррес

Один из основных источников разрушений при землетрясении — резонанс. В связи с этим предложена следующая идея. Мы разрабатываем специальные управляющие связи между этажами, которые в норме не соединены с самим зданием, но при необходимости могут быть присоединены. Они созданы так, что их добавление изменяет резонансную частоту здания Мы последовательно отслеживаем смещения в структуре, и, когда они превышают определенный предел, связи включаются для контроля движения системы Эта идея также может быть применена, чтобы предотвратить вибрационное разрушение в больших аэрокосмических системах

Brief introduction to an engineering problem

The problem of resonance destruction. One of the main sources of destruction during earthquake is resonance. The majority of the earthquake strikes are not powerful enough to destroy a building in one blow, but the earthquake usually contains a wide spectrum of vibrations with different frequencies. When the characteristic frequencies of the building lie inside this spectrum area, resonance and eventually destruction can occur.

The vibrations can also occur, when a bridge, an oil platform, or an aerospace construction encounters periodic waves. How can we diminish destruction? Before we enumerate different methods, let us describe (briefly) the corresponding mathematical problem.

Physical and mathematical description of a resonance: in brief. A building (bridge, aerospace construction, etc) consists of many basic parts (beams, etc). Earthquakes cause these parts to displace, thus (ultimately) destroying the building. The position of each part can be described by describing the positions of its major points (endpoints, sometimes a midpoint, etc). In order to describe the state of a building at any given moment of time t, we must describe the positions $\vec{x}_a(t), 1 \le a \le N$, of all these points. Since we are interested not in the engineering design itself, but only in the abnormalities, instead of the positions themselves, we will use *displacements* $\Delta\vec{x}_a(t) = \vec{x}_a(t) - \vec{x}_a^{(0)}$, i.e., the differences between the current and ideal positions of these points.

How to describe dynamics? According to Newton's law, to describe the dynamics, we need to know the total force that acts on each point. This force $\vec{F}_a$ consists of two parts: $\vec{F}_a = \vec{F}_a^e + \vec{F}_a^s$:

- the *external* force $\vec{F}_a^e$ that is caused by the earthquake itself;
- the *internal*, or *structural* force $\vec{F}_a^s$ that is caused by the connection between different parts of the building.

The structural force depends on the positions and velocities of all the points. Then, we can get the accelerations $d^2\vec{x}_a(t)/dt^2$ as $\vec{f}_a = \vec{F}_a/m_a$. Thus, we have a system of second-order differential equations:

$$d^2\vec{x}_a(t)/dt^2 = \vec{f}_a^s(\vec{x}_1,\ldots,\vec{x}_N,\dot{\vec{x}}_1,\ldots,\dot{\vec{x}}_n) + \vec{f}_a^e.$$

These equations can be reformulated in terms of displacements:

$$d^2\Delta\vec{x}_a(t)/dt^2 = \vec{f}_a^s(\Delta\vec{x}_1,\ldots,\Delta\vec{x}_N,\Delta\dot{\vec{x}}_1,\ldots,\Delta\dot{\vec{x}}_n) + \vec{f}_a^e.$$

Since we are talking about small and not so fast displacements (large or fast displacements will immediately ruin the building), we can expand $\vec{f}_a^s$ into Taylor series, and neglect the terms that are quadratic (or of higher order) in $\Delta\vec{x}_a$ and $\Delta\dot{\vec{x}}_a$. Thus, we arrive at a *linear* system of second-order differential equations with constant coefficients. It is well how to solve such a system:

- First, we reduce it to a first-order system by introducing new variables $\vec{v}_a = \dot{\vec{x}}_a$. In terms of Δx_a and v_a, we get the following first-order system:

  $$\dot{\vec{v}}_a = \vec{f}_a^{sl}(\Delta\vec{x}_1,\ldots,\Delta\vec{x}_n,\vec{v}_1,\ldots,\vec{v}_n) + \vec{f}_a^e,\ 1 \le a \le N;$$

  $$\Delta\dot{\vec{x}}_a = \vec{v}_a,\ 1 \le a \le N.$$

  Here, the additional index l in the expression for the structural acceleration $\vec{f}_a^{sl}$ indicates that we are restricting ourselves to linear terms only.

  These equations become easier to grasp and to solve if instead of $2N$ vector unknowns $\Delta\vec{x}_a$ and $\vec{v}_a$, we consider all $3 \cdot 2N = 6N$ components of these vectors as new scalar unknowns $q_1,\ldots,q_{6N}$. In terms of these variables, we get a system of first-order linear differential equations with constant coefficients:

  $$\dot{q}_a = \sum A_{ab}q_b + f_a.$$

- It is well known how to solve such a system (see, e.g., [2]). Namely, if $f_a = 0$, then a general solution of this system can be represented as linear combination of the functions $\exp(\lambda_\alpha t)$ and $t^m \exp(\lambda_\alpha t)$, where λ_α are eigenvalues of the matrix A_{ab} (terms $t^m \exp(\lambda_\alpha t)$ with $m > 0$ appear only if we have a degenerate eigenvalue).

These terms have different asymptotic behavior:

- If $\mathrm{Re}\,\lambda_\alpha > 0$, then the corresponding term becomes large as $t \to \infty$.
- If $\mathrm{Re}\,\lambda_\alpha < 0$, then the term tends to 0 as $t \to \infty$.
- If $\mathrm{Re}\,\lambda_\alpha = 0$ (i.e., if $\lambda_\alpha = i\omega_\alpha$ for some real value ω_α), then we get an infinitely oscillating term $\exp(i\omega_\alpha t) = \cos(\omega_\alpha t) + i\sin(\omega_\alpha t)$. This real value ω_α is called an *eigen frequency* of the system.

Buildings are usually stable, therefore $\mathrm{Re}\,\lambda_\alpha \le 0$.

All this was true for $f_a = 0$. What to do if $f_a \neq 0$? An arbitrary non-zero force f_a can be represented (by means of Fourier transformation) as a linear combination of the sinusoidal waves with different frequencies ω. Since our equations are linear, in order to get the response q_a to the initial force, it is sufficient to compute the system's reactions for different frequencies, and then add these reactions.

In particular, if one of the Fourier components of the force f_a has a frequency that coincides with one of the eigen frequencies ω_α, then the solution behaves as $t\exp(i\omega_\alpha t)$ when $t \to \infty$. As a result, displacements and/or frequencies become larger and larger, and the building can be destroyed. This phenomenon is called a *resonance*.

Strictly speaking, for real-life systems, there is always some friction, therefore, usually $\mathrm{Re}\,\lambda_\alpha < 0$. The role of this friction is different for small and large frequencies:

- For small frequencies (that correspond to relatively smooth motion), this friction is small and practically negligible (in physical terms: the materials have *elastic behavior* for such frequencies). Therefore, although for $t \to \infty$, the displacement will eventually tend to 0, but meanwhile, for reasonable t, its time dependency is indistinguishable from $t\exp(i\omega_\alpha t)$ and thus, the building may be destroyed.
- For bigger frequencies, that correspond to extremely fast changes of displacement and velocity, there is usually a very serious resistance.

  To give the reader who is not a specialist in mechanics an idea that this is really the case, let us imaging ourselves pushing a piece of furniture. It is possible (with some force) to move it to and fro slowly, but practically impossible to move to and fro fast.

  Because of that, for high eigen frequencies, the resonance displacement will quickly tend to 0, and therefore, such perturbations are of no serious threat to the building.

So, not all eigen frequencies cause destruction, but only small ones. How to avoid this resonance destruction?

Active control. The *brute-force* idea is to add a huge auxiliary mass to damp the vibrations, and to forcefully return the structures to their initial positions by applying computer-controlled force (see, e.g., [5, 10, 22, 31]; this idea is called *active control*).

In terms of our mathematical model, adding an additional force means adding an additional term to f_a that compensates for the unwelcome (resonant) components of f_a.

This method often works, but it has two main problems:

- this method requires lots of energy for control, and, e.g., spacecraft must be energy-efficient;
- if not precisely implemented, this method can pour lots of additional energy into the system in the wrong times, and thus cause additional destruction [17].

Semi-active control. There exist a modification of this method, called *semi-active control*, where instead of implementing an out-of-phase force, structural changes are made in the system to change its characteristic frequencies and thus avoid the resonance (see, e.g., [7, 10–14]).

A new approach. For semi-active applications, a new method was proposed in [23]. According to this method, we do not add any energy to the system at all. Instead, we design special control links that are normally unattached to the building but can be attached if necessary. They are so designed that adding them changes the building's characteristic frequency. We keep monitoring displacements, and when motion is detected, the control linkages are alternatively engaged and disengaged in such a way that energy in lower (more destructive) modes of vibration is shifted into higher modes, where the energy is quickly dissipated through passive dumping within the structure itself. This idea can be also applied to avoid vibrational destruction of large aerospace structures.

This idea has been thoroughly checked theoretically, and verified through computer simulation [23].

A brief mathematical description of the new approach. In terms of our above-given mathematical description of resonance destruction, the idea is as follows. When we engage an additional linkage, we thus change the way how the displacements and velocities of different points influence each other. In our terms, we change the structural force function $\vec{F}^s_a$. As a result, the coefficients of the linearized system of equations will also change: instead of

$$\dot{q}_a = \sum A_{ab} q_b + f_a$$

we will have

$$\dot{q}_a = \sum \tilde{A}_{ab} q_b + f_a$$

$\tilde{A}_{ab} \neq A_{ab}$.

The idea is as follows: if the frequency of one of the components of the external force coincides with one of the eigen frequencies of the system (i.e., with one of the eigen values of the matrix A_{ab}), then, we engage the control linkage; this changes the matrix and hence, changes the eigenvalues that are no more equal to the frequencies of the external force.

In order to implement this idea, we must have a way to decide when to switch. We have already mentioned that only small frequencies are potentially destructive. Therefore, it is reasonable to estimate the potential danger of the existing displacements and velocities by the total energy of all low-frequency components (i.e., of all the components whose frequency is $\leq \omega_0$ for some chosen ω_0), and to engage the control linkage if and only if this engagement decreases this total energy.

For a linear system, energy is a quadratic function of displacements and velocities (i.e., in our terms, of the variables q_a): terms that are quadratic in $\Delta\vec{x}_a$ correspond to the potential (elastic) energy, and terms that are quadratic in velocity represent kinetic energy. For this same reason, the total energy of low-frequency components is a quadratic function of q_a, say $E = \sum E_{ab}q_aq_b$. When we engage a control linkage, we thus change the coefficients of the linear system, and hence, we change the coefficients in the expression for its energy. The new expression for energy will be $\tilde{E} = \sum \tilde{E}_{ab}q_aq_b$. The coefficients E_{ab} and $\tilde{E}_{ab}$ can be determined before the control starts, so in course of the actual control, we know them.

So, for given q_a, the question of whether to engage the linkage or not reduces to checking a simple inequality: $E = \sum E_{ab}q_aq_b > \tilde{E} = \sum \tilde{E}_{ab}q_aq_b$, or, what is equivalent, $\sum(\tilde{E}_{ab}-E_{ab})q_aq_b > 0$.

In real life, we do not know the exact values of q_a (i.e., of displacements and velocities). If we substitute the imprecise values of q_a into this inequality, then we may end up with a wrong decision (and in these applications, wrong decisions can be fatal):

- We may *not* engage the linkage when it is necessary to, and thus fail to avoid destruction, or
- We may *engage* the linkage when there has been no potential damage to the building, and by engaging this linkage at the wrong time, actually worsen the situation and force destruction.

Since imprecision in q_a can lead to grave consequences, it is important to determine q_a as accurately as possible.

Related mathematical problem: brief informal description. To check for a resonance, one must know not only the displacements $x(t_i)$, but the rates $\dot{x}(t_i)$ with which they change. The existing velocity sensors are much more expensive than the displacement sensors. So, if we are designing a reasonably cost system, we cannot use velocity sensors. Instead, we must estimate the velocity from the measured (and hence, approximately known) values of the displacements $x(t)$.

We must make control decisions really fast (in milliseconds). Therefore, there is no time to process lots of data. So, when estimating $\dot{x}(t_i)$, we can take into consideration only the measurements in a few consequent points. Hence, all of them belong to a small time interval, and therefore, on this interval, the function $x(t)$ can be well approximated by its first few Taylor expansion terms. In case this interval is sufficiently small, linear approximation is sufficient, so we can assume that a function $x(t)$ is linear. If this is not enough, we must add second order terms, and consider the case when $x(t)$ is quadratic.

A reasonable system must rely on low-cost, reasonably priced sensors. Therefore, the resulting measurements have a non-negligible error. This error leads to an error in the resulting estimate for the derivative. In view of that, it is necessary to design methods of computing derivative that would have the smallest possible error.

What we are going to do. In the present paper, we find the optimal estimates for the derivatives for both cases: when the error is of statistical nature (in which case we know its statistical characteristics), and when this error is systematic (in which case we know only the interval of its possible values).

Interval estimates are considered in Part I, statistical estimates in Part II. Part III is reserved for proofs.

Part I. Interval estimates

1. Interval estimates for the derivative: formulation of a mathematical problem

1.1. Definitions

Definition 1. *Suppose that for some integer n, we are given n real numbers $t_1 < \cdots < t_n$, n real number $\varepsilon_i > 0$ and x_i, and n intervals $F_i = [x_i^+, x_i^-]$, $i = 1, \ldots, n$, where $x_i^- = x_i - \varepsilon_i$ and $x_i^+ = x_i + \varepsilon_i$. By a function interval we mean the set $\mathcal{F}$ of all continuous functions $x(t)$ such that for all $i = 1, \ldots, n$, $x(t_i) \in F_i$.*

Remark. This definition was, in effect, originally proposed by R. E. Moore (see, e.g., discussions of *discrete functions* in [20], Section 5.1, and [21], Section 2.5).

Definition 2. *Suppose that a function interval $\mathcal{F}$ is given. We say that a class of functions S is consistent (or compatible) with $\mathcal{F}$ if $S \cap \mathcal{F} \neq \varphi$.*

Remark. In our case, this means that the measurement results (expressed by a function interval $\mathcal{F}$) are consistent with the supposition that the unknown function belongs to the class S.

Definition 3. *Suppose that we are given a function interval $\mathcal{F}$, a class S of differentiable functions that is compatible with $\mathcal{F}$, and a real number t_0. A real number d is called a possible value of the derivative $dx(t)/dt$ w.r.t. S if $d = \dot{x}(t_0)$ for some $x \in S \cap \mathcal{F}$. The set of all possible values will be called a derivative (or a derivative set) of $\mathcal{F}$ with respect to S in the point t_0, and denoted by $d\mathcal{F}/dt_{|S}(t_0)$.*

Remark. In other words, d is a possible value if there exists a function $x(t) \in S$ such that $dx(t_0)/dt = d$ and $|x_i - x(t_i)| \leq \varepsilon_i$ for $i = 1, \ldots, n$.

Proposition 1. *If S is a convex class of functions, then the derivative $d\mathcal{F}/dt_{|S}$ is convex, i.e., it is either an interval (open, closed, or semi-open), or a semi-line.*

(All the proofs are placed in Part III, for reader's convenience).

Remark. In this paper, we will consider only convex classes of functions, namely, the class of all linear functions, and the class of all quadratic functions. Therefore, the derivative sets will always be intervals (finite or infinite).

Denotations. Let us denote the set of all linear functions $x(t) = a + bt$ by L, and the set of all quadratic functions $x(t) = a + bt + ct^2$ by Q.

1.2. Main problem: first formulation

Given $\mathcal{F}$ and t_0, to check whether $\mathcal{F}$ is compatible with L or Q, and to compute the derivatives $d\mathcal{F}/dt_{|L}$ and $d\mathcal{F}/dt_{|Q}$

It is reasonable first to try L, and then, if L is incompatible with $\mathcal{F}$, to try Q.

1.3. How to solve this problem?

Traditional and interval methods of numerical differentiation are not applicable to this problem. There exist several methods of numerical differentiation, both in traditional numerical mathematics (see, e.g., [4]), and in interval mathematics (see, e.g., Chapter 8 of [1], Section 5.4 of

[3], [6, 16, 18], Chapter 11 of [19], [24–28]). However, we cannot directly apply these methods to our problem, because:

- methods of traditional numerical mathematics usually give a numerical estimate, and do not provide us with an interval of possible values; and we have already mentioned that for our applications, this is crucial;
- methods of *interval mathematics* are mainly designed for estimating the derivative of the functions that are given by analytical expressions [1, 3, 6, 18, 19, 24–26]; definitions from [16, 27, 28] are applicable to the case when we have finite interval estimates for $x(t)$ for all t, but in our case, we have such estimates only for $t = t_1, \dots, t = t_n$.

Comment. In general, there is no way to reconstruct a function for all t from the observations in finitely many points. In our problem, however, as we have already mentioned, high-frequency components of the displacement function go to 0 real fast. As a result, we consider only functions that are formed by low-frequency components and are therefore, very smooth. And if we have a very smooth function that is defined on a small interval, then its linear or quadratic terms provide a practically perfect approximation.

We can apply linear programming. These problems *can* be solved by reducing them to linear programming. For linear functions $x(t) = a + bt \in L$, $\dot{x}(t) = b$. Therefore, the upper limit d^+ is the solution of the problem $d \to \max$ under the condition that $|a + bt_i - x_i| \le \varepsilon_i$ for $1 \le i \le n$, or, what is equivalent, $-\varepsilon_i \le a + bt_i - x_i \le \varepsilon_i$. The lower limit d^- is the solution of the problem $d \to \min$ under the same constraints.

For quadratic functions $x(t) = a + bt + ct^2$, $d = b + 2ct_0$, therefore, the limits can be found by solving the following conditional optimization problems: $b + 2ct_0 \to \max$ and $b + 2ct_0 \to \min$ under the condition that $-\varepsilon_i \le a + bt_i + ct_i^2 - x_i \le \varepsilon_i$, $i = 1, \dots, n$.

In both cases, we have linear programming problems, and they can be solved using known polynomial-time linear programming techniques (e.g., Karmarkar's method [9]).

Why is linear programming not satisfactory. As we have already mentioned, we are in the area of real-time control, therefore, the computation time must be as small as possible. The computational time of an algorithm is roughly proportional to the total number of elementary computational steps (arithmetic operations, comparisons, etc). Karmarkar's method demands $Cn^{3.5}$ steps, where n is a number of equations (i.e., in our case, the number of measurements), and C is a rather big constant.

Even if we have several processors working in parallel, the general case of a linear programming program is unlikely to get a dramatic speed-up (more formally, it belongs to the class P that consists, crudely speaking, of problems with worst possible parallelization abilities, see, e.g., [8]).

Why cannot we apply known methods of solving interval linear equations? Our problem can be easily reformulated in interval terms. For example, for a linear case, the problem is to find all possible values of b for which for some a, we satisfy the system of interval inclusions: $a + bt_i - x_i \in [-\varepsilon_i, \varepsilon_i]$, $1 \le i \le n$. This system is called an *interval linear system* (for a general definition, see, e.g., [29, 30], and references therein), and the interval of all possible values of b is called an *optimal solution* of such a system. There exist numerous methods of finding optimal solutions of interval linear systems. These methods have been perfected for many years, and the latest algorithms are very ingenious and fast (for the latest survey, see [30]).

The majority of these methods share the same good property: they are *universal* in the sense that they are applicable to an arbitrary interval linear system. But this same good property is the cause of the common problem of these algorithms, the problem that leads us to the necessity to invent new ones. Namely, the problem of finding an optimal solution to an interval linear system has been proved to be NP-hard [15]. This term means that no matter how smart (and thus fast) an algorithm for solving such system can be, in some cases, its computation time will increase exponentially (i.e., as a^n for some $a > 1$). An exponential function grows so fast that for reasonable n, this time quickly exceeds the lifetime of the Universe. The fact that such "worst" cases exist for these algorithms does not invalidate their usage in economic problems, in some optimization problems, etc: if once in a while we do not get the optimal result, or it takes too long to get this result, no big deal.

In our problem (earthquake-resistant engineering) milliseconds do count, and 100% reliability (i.e., getting results in 100% of all the cases) *is* an issue. If once in a while, our system fails, the building (or the spaceship) may be destroyed. Because of that specific feature of our problem, we cannot use the existing universal algorithms. We have therefore to design new ones, that will be applicable to our problems only, but that will have a guaranteed (and small) running time.

1.4. Final formulation of the problem

To find algorithms that compute derivative sets faster than the general methods of linear programming.
Comment. If we achieve that, then our algorithm will be faster than $n^{3.5}$ and hence (for sufficiently large n), its running time will be smaller than an exponential function.

2. First result: How to compute derivative set if we know that $x(t)$ is linear

Theorem 1. *A function interval $\mathcal{F}$ is compatible with L if and only if $d^+ \geq d^-$, where $d^+ = \min_{i>j}\left((x_i^+ - x_j^-)/(t_i - t_j)\right)$ and $d^- = \max_{i>j}\left((x_i^- - x_j^+)/(t_i - t_j)\right)$. If $\mathcal{F}$ and L are compatible, then the derivative of $\mathcal{F}$ with respect to L is equal to $[d^-, d^+]$.*
Remark. The formulas from Theorem 1 prompt the following natural algorithm for computing d^+: set an auxiliary element where the current record will be stored to $s := +\infty$ (in Pascal, to *MaxInt*), and then make two embedded loops for i and for j, inside which we update the record values s by assigning $s := \min\left(s, (x_i^+ - x_j^-)/(t_i - t_j)\right)$. After the loops are over, s contains the desired value d^+. A similar natural algorithm can compute d^-.

Since we are interested in computing the estimates for the derivative as fast as possible, a crucial question is what is the running time of this algorithm. A running time is usually estimated by the total number of elementary computational steps (i.e., comparisons, and arithmetic operations) that we must perform to apply the algorithm. So, let us enumerate the number of computational steps for the above-described algorithm.

Before we proceed, let us make one remark. With one exception, throughout the whole paper, by a computational step, we will mean an arithmetic operation. There will be one exception: while computing the statistical estimate, it is impossible to avoid computing a square root, so, for that case (and for that case only), we will add computing the square root to the list of elementary operations. Let us now return to our algorithm. Since we are given x_i, t_i and ε_i, we first need to compute $x_i^+ = x_i + \varepsilon_i$ and $x_i^- = x_i - \varepsilon_i$. Each computation takes 1 step, so totally, we need $2n$ steps. After that, computing each pair of expressions $(x_i^- - x_j^+)/(t_i - t_j)$ and $(x_i^+ - x_j^-)/(t_i - t_j)$ takes 5 computational steps (3 subtractions and 2 divisions). Totally, we need to compute $n(n-1)/2$ such pairs (for all $i < j$), so it takes $5n(n-1)/2$ computational steps. Now, to compute d^+, we must compare $n(n-1)/2$ such expressions, and find the smallest one. We can find the smallest of $n(n-1)/2$ numbers in $n(n-1)/2 - 1$ steps. To compute d^-, we need the same number of comparisons. So, the total number of computational steps for computing d^+ and d^- is $2n + 5n(n-1)/2 + 2\big(n(n-1)/2 - 1\big) = 2n + 5/2n^2 - 5/2n + n^2 - n - 2 = 7/2n^2 - 3/2n - 2 < 7/2n^2$. So, we proved the following result:

Theorem 2. *For $S = L$, there exist an algorithm that given a function interval $\mathcal{F}$, checks whether it is compatible with L, and if it is compatible, computes the derivative of $\mathcal{F}$ in $< 3.5n^2$ computational steps.*

Remarks.

1) This estimate is much better than for Karmarkar's algorithm.

2) As we have already mentioned, in earthquake-resistant engineering, every millisecond counts. So, although our algorithm is faster than the known ones, it would be nice to make it still faster. Our hope that this can be done is based on the following argument. The algorithm that we have just described is based on the simplest possible idea of solving an interval linear system: namely, wherever in this system we have an interval (in our case, $x_i \in [x_i^-, x_i^+]$), we choose one of the possible endpoints. Thus, we get lots of different non-interval linear systems. For each of these systems, we find a solution. By comparing these solutions, we compute the biggest and the smallest values of the unknowns (i.e., compute an optimal solution). This idea is known to be too expensive time-wise: it turns out that not all possible linear systems have to be solved. By cutting down on the number of these systems, we can drastically decrease the running time of the algorithms that find optimal solutions of interval linear systems [29, 30]. These "cutting" ideas are not directly applicable to our case (at least we could not figure out how to apply them), but their existence makes us hope that our (rather primitive) algorithm can (probably) be further improved.

3) Another reasonable way to decrease the running time of an algorithm is to run it on a parallel computer (in which several processors can run in parallel, i.e., simultaneously). If we have several processors, then we can indeed decrease the computation time:

Theorem 3. *For $S = L$, there exists an algorithm that given a function interval $\mathcal{F}$, checks whether $\mathcal{F}$ is compatible with L, and if it is compatible, computes the derivative of $\mathcal{F}$ in parallel. The running time of this algorithm is smaller than the time of $2\log_2 n + 3$ computational steps.*

3. Second result: How to estimate the derivative in real time

3.1. Motivation of the following definitions

In the previous sections, we considered the situation when we have to apply the algorithm once. However, in the desired applications, we must monitor the derivative, i.e., with every new measurement, produce a new estimate. This new estimate must be obtained in *real time*, i.e., the computations must be done by the moment of the next measurement. Therefore, the computation that uses $x_1, \dots, x_n$, must produce the result before the moment t_{n+1}, and can, therefore, spend the computational time $\le t_{n+1} - t_n$.

Usually, the time intervals $t_{i+1} - t_i$ are either equal, or approximately equal. In both cases, the smallest m of these differences if positive, and the biggest M is finite: $0 < m \le M < \infty$. Therefore, to estimate the derivatives in the moment t_n, we must use time $\le M$. If we denote by Δt the time of one elementary computational step, then we conclude that for every n, we can use $\le C$ computational steps, where we denoted $C = M/\Delta t$.

For both algorithms from Section 3, however, the number of computational steps increases with n, so we cannot use them directly for monitoring

For a *sequential computer*, the fact that the computation time increases with n can be easily understood. Indeed, since we must process all n values $x_1, \dots, x_n$, to each value we must apply at least one elementary operation. The elementary operations that we considered (arithmetic, comparisons, etc) require at most 2 variables. So, we need at least $n/2$ elementary operations to process all the values $x_1, \dots, x_n$. Therefore, the number of computational steps is $\ge n/2$, and thus it tends to ∞ when $n \to \infty$.

We cannot use all the values $x_1, \dots, x_n$. How many of them can we use?

Since processing k numbers requires $\ge k/2$ computational steps, and the number of steps that we can use is limited by $M/\Delta t$, we can conclude that $k/2 \le M/\Delta t$, i.e., that $k \le b$, where we denoted $b = 2M/\Delta t$. Therefore, there exists a constant b such that in estimating the derivative, we can process only b values of x_i.

For a *parallel computer*, we can have similar estimates. Indeed, the final result of the computation is obtained by using some elementary operation. Each operation can handle only two numbers. Therefore, any computation that can be performed in the time of one computational step, can process at most 2 values x_i. If we take computations that take the time of 2 computational steps, then we can at best process 2 numbers, each of which is a result of processing at most 2 numbers, i.e., the result can depend on at most 4 different values x_i. In general, after the time that is necessary to perform k computational steps, we can process at most 2^k different values x_i. Therefore, since the time is limited by $\le M/\Delta t$ computational steps, we can process at most $b = 2^{M/\Delta t}$ values.

In both cases, to estimate the derivative, we can process at most b different values of x_i. So, we must choose b values out of n. Since we consider a process that needs monitoring, the value of the derivative can change over time. So, to get the most precise estimate at the moment t_n, we must consider b latest values $x_n, x_{n-1}, \dots, x_{n+1-b}$ (of course, if $n \le b$, then we can process all the values x_i).

Let us formulate this situation in mathematical terms.

3.2. Definitions of real-time algorithms, and the complexity of such algorithms

Definition 4. *Suppose that an integer $b > 1$ is given. Suppose also that $t_1 < t_2 < \cdots < t_n < \cdots$, $x_1, x_2, \dots, x_n, \dots$, and $\varepsilon_1 > 0, \varepsilon_2 > 0, \dots, \varepsilon_n > 0, \dots$ are three potentially infinite sequences of real numbers. The values t_i will be called moments of time. For a moment t_n, by a b-bounded function interval $\mathcal{F}_b$ we mean the set of all continuous functions $x(t)$ such that $x(t_i) \in F_i = [x_i - \varepsilon_i, x_i + \varepsilon_i]$ for all i such that $n - b < i \le n$. By a b-bounded interval estimate for* *a derivative in the moment t_n, we mean the derivative set $d\mathcal{F}_{|b}/dt_{|S}$ of the b-bounded function interval.*

Remark. In particular, for $S = L$, the b-bounded interval estimate for a derivative is equal to $[d_n^-, d_n^+]$, where

$$d_n^+ = \min_{n-b<j<i\le n} \frac{x_i^+ - x_j^-}{t_i - t_j}$$

and

$$d_n^- = \max_{n-b<j<i\le n} \frac{x_i^- - x_j^+}{t_i - t_j}.$$

Definition 5. *Suppose that an integer $b > 1$ and a positive real number C are given. We say that an algorithm computes interval estimates in real time with $\le C$ computational steps per measurement if this algorithm works as follows: it reads x_1, t_1, x_2, t_2, then after $\le C$ computational steps generates d_2^- and d_3^+; then reads x_3 and t_3, and in $\le C$ computational steps generates d_3^- and d_3^+, etc.*

Remark. In principle, one can apply an algorithm from Theorem 2 to estimate d_n^- and d_n^+ for all n. Since we are processing $\le b$ numbers, this algorithm will require $C \approx 3.5b^2$ computational steps per measurement. The following theorem shows that we can do better.

Theorem 4. *For $S = L$, there exists an algorithm that computes interval estimates in real time, with $< b^2 + 4b$ computational steps per measurement.*

Remark. For parallel computers, if we apply an algorithm from Theorem 3 on every step, we get the following result:

Theorem 5. *For $S = L$, there exists an algorithm that computes interval estimates in parallel in real time, with $< 2\log_2 b + 3$ computational steps per measurement.*

4. Third result: How to compute derivative set if $x(t)$ is quadratic

Theorem 6. *Assume that t_0 is a real number.*

If $n \le 2$, then any function interval $\mathcal{F}$ is compatible with Q, and the derivative of $\mathcal{F}$ with respect to Q in the point t_0 coincides with $(-\infty, +\infty)$.

If $n > 2$, then a function interval $\mathcal{F}$ is compatible with Q if and only if the following three conditions are satisfied:

1) $(x_i^- - x_j^+)/(t_i - t_j) \le (x_k^+ - x_l^-)/(t_k - t_l)$ *for all* i, j, k, l *such that* $(t_i - t_0)^2 > (t_j - t_0)^2$, $(t_k - t_0)^2 > (t_l - t_0)^2$, *and* $t_i + t_j - 2t_0 = t_k + t_l - 2t_0 > 0$;

2) $(x_j^+ - x_i^-)/(t_j - t_i) \ge (x_l^- - x_k^+)/(t_l - t_k)$ *for all* i, j, k, l *such that* $(t_i - t_0)^2 > (t_j - t_0)^2$, $(t_k - t_0)^2 > (t_l - t_0)^2$, *and* $t_i + t_j - 2t_0 = t_k + t_l - 2t_0 < 0$;

3) $d^+ \ge d^-$, *where* $d^+ = \min(d_1^+, d_2^+)$, $d^- = \max(d_1^-, d_2^-)$,
$d_2^- = \max(N_{ijkl}^-/D_{ijkl})$, $d_2^+ = \min(N_{ijkl}^+/D_{ijkl})$,

$$N_{ijkl}^- = \frac{x_i^- - x_j^+}{(t_i - t_j)(t_i + t_j - 2t_0)} - \frac{x_k^+ - x_l^-}{(t_k - t_l)(t_k + t_l - 2t_0)},$$

$$N^+_{ijkl} = \frac{x_i^+ - x_j^-}{(t_i - t_j)(t_i + t_j - 2t_0)} - \frac{x_k^- - x_l^+}{(t_k - t_l)(t_k + t_l - 2t_0)},$$

$$D_{ijkl} = \frac{1}{t_k + t_l - 2t_0} - \frac{1}{t_i + t_j - 2t_0}$$

min *and* max *are taken over all quadruples* (i, j, k, l) *such that* $(t_i - t_0)^2 > (t_j - t_0)^2$, $(t_k - t_0)^2 > (t_l - t_0)^2$, *and* $t_i + t_j - 2t_0 > t_k + t_l - 2t_0$,

and $d_1^- = \max\big((x_i^- - x_j^+)/(t_i - t_j)\big)$, $d_1^+ = \min\big((x_i^+ - x_j^-)/(t_i - t_j)\big)$, *where* min *and* max *are taken over all* $i > j$ *such that* $(t_i - t_0)^2 = (t_j - t_0)^2$.

If $\mathcal{F}$ *and* Q *are compatible, then the derivative of* $\mathcal{F}$ *with respect to* Q *in the point* t_0 *is equal to* $[d^-, d^+]$.

Theorem 7. *For* $S = Q$, *there exist an algorithm that given a function interval* $\mathcal{F}$ *and a point* t_0, *checks whether* $\mathcal{F}$ *is compatible with* Q, *and, if it is compatible, computes the derivative set. This algorithm requires* $< n^4 + 12n^2$ *computational steps.*

Remark. For big n, this method is worse than Karmarkar's (that gives $Cn^{3.5}$). However, for small n, it is reasonable to use, because a constant C in Karmarkar's method is rather big.

Theorem 8. *For* $S = Q$, *there exists an algorithm that given a function interval and a value* t_0, *checks whether* $\mathcal{F}$ *is compatible with* Q, *and, if it is compatible, computes the derivative of* $\mathcal{F}$ *in parallel. The running time of this algorithm is less than or equal to the time necessary for* $4\log_2 n + 5$ *computational steps.*

We can apply these algorithms to the case of real-time estimates, and get the following results:

Theorem 9. *For* $S = Q$, *there exists an algorithm that computes interval estimates in real time with* $< b^4 + 12b^2$ *computational steps per measurement.*

Theorem 10. *For* $S = Q$, *there exists an algorithm that computes interval estimates in real time in parallel with* $\leq 4\log_2 b + 5$ *computational steps per measurement.*

Part II. Statistical estimates

5. Statistical estimates of the derivative: definitions and the main result

Remark. In this section, we will consider the case when all the measurements are performed by the same measuring device, and therefore, all the measurements have the same precision.

Denotation. For a random variable ξ, we will denote its mathematical expectation (average) by $E[\xi]$, and its standard deviation $\sqrt{E[(\xi - E[\xi])^2]}$ by $\sigma[\xi]$.

Definition 6. *Assume that we are given an integer* $n \geq 2$, *a real number* $\sigma > 0$, *and* n *real numbers* $t_1 < t_2 < \cdots < t_n$. *By a statistical estimate for an interval derivative we mean a linear function that transforms a sequence* $x_1, \ldots, x_n$ *into a value* $d = a_1x_1 + a_2x_2 + \cdots + a_nx_n$. *We say that a statistical estimate is non-biased if for arbitrary real numbers* a *and* b, *and for arbitrary* n *independent random variables* ξ_i *with average 0 and standard deviation* σ, *the mathematical expectation* $E[d] = E[a_1x_1 + a_2x_2 + \cdots + a_nx_n]$ *is equal to* b, *where* $x_i = a + bt_i + \xi_i$. *By an error of a statistical estimate we mean the standard deviation* $\sigma[b - d]$ *of the difference between*

b and d. We say that a statistical estimate is optimal if it is non-biased, and its error is the smallest possible (among non-biased estimates).

Remark. We are interested in statistical estimates for which the error is the smallest possible. We consider only linear functions as statistical estimates, because for the most common distribution (Gaussian) it is known that the estimates for which standard deviation is minimal are linear.

Theorem 11. *The optimal statistical estimate is $d = \sum_i a_i x_i$, where*

$$a_i = (t_i - \bar{t})/(n\sigma_t^2),\ \bar{t} = \sum_i t_i/n, \text{ and } \sigma_t^2 = \left(\sum_i (t_i - \bar{t})^2\right)/n.$$

Its error is equal to $\sigma/(\sqrt{n}\sigma_t)$.

6. Computational complexity of the optimal statistical estimate

Case when the times t_i are known beforehand. Let us first consider the case, when the times t_i are known before the measurements, so that the coefficients a_i can be precomputed.

Theorem 12. *If the coefficients a_i are known, then to compute the optimal statistical estimate for the derivative, we need $2n - 1$ computational steps.*

Theorem 13. *If the coefficients a_i are known, then we can compute the optimal statistical estimate for the derivative in parallel in the time that is necessary for $\le \log_2 n + 2$ computational steps.*

General case. Let us now consider the general case, when the values of t_i are not known beforehand.

Theorem 14. *In the general case, to compute the optimal statistical estimate for the derivative, we need $\le 7n - 2$ computational steps.*

Theorem 15. *If the general case, we can compute the optimal statistical estimate for the derivative in parallel in the time that is necessary for $\le 3\log_2 n + 8$ computational steps.*

7. Statistical estimates in real time

Definition 7. *Suppose that an integer b is given. Suppose also that $x_1, x_2, \ldots, x_n, \ldots$ and $t_1 < t_2 < \cdots < t_n < \cdots$ are two potentially infinite sequence of real numbers. By a statistical estimate for the derivative in the moment t_n, $n \ge b$, we mean the result of applying an optimal statistical estimate for the derivative to the values $x_n, x_{n-1}, \ldots, x_{n+1-b}$ and $t_n, t_{n-1}, \ldots, t_{n+1-b}$.*

Definition 8. *Suppose that an integer $b > 1$ and a positive real number C are given. We say that an algorithm computes statistical estimates for the derivative in real time, with $\le C$ computational steps per measurement if this algorithm works as follows: it reads x_1, t_1, then after $\le C$ computational steps reads x_2 and $t_2, \ldots$, reads x_b, t_b, after $\le C$ computational steps computes a statistical estimate for the derivative in the moment t_b and its error, then reads x_{b+1}, t_{b+1}, after $\le C$ computational steps computes a statistical estimate for the derivative in the moment t_{b+1} and its error, etc.*

Remark. Since we are interested also in computing the error, and the formula for the error contains a square root, we must add square root to the list of elementary computational steps. The results are as follows:

Theorem 16. *There exists an algorithm that computes statistical estimates for the derivative in real time, with ≤ 18 computational steps per measurement.*

Remark. The nice property of this algorithm is that its running time does not depend on b, so, we can use arbitrarily big b, and still get the estimates in real time. This running time can be further diminished if we have several processors at our disposal:

Theorem 17. *There exists an algorithm that computes statistical estimates for the derivative in parallel in real time, with ≤ 7 computational steps per measurement.*

Remark. In the next Section, we will consider a frequent case, when $t_{i+1} - t_i = \text{const}$. In this case, as we will see, the number of computational steps can be made even smaller.

8. Optimal statistical estimates for the case when measurements are made in consequent moments of time

Definition 9. *We say that the measurements are made in consequent moments of time if $t_{i+1} - t_i = \Delta t$ for some $\Delta t > 0$. In this case, $t_i = t_1 + (i-1)\Delta t$.*

For this case, we can simplify the formulas from Theorem 11. Let us first consider the case, when n is odd: $n = 2k+1$. To simplify the formulas, let us denote the midpoint (t_{k+1}) by s_0, and other points by $s_{-k}, \ldots, s_{-1}, s_0, \ldots, s_1, \ldots, s_k$, where $s_i = t_{i+k+1}$. Correspondingly, let us denote by y_i, $-k \leq i \leq k$, the values that correspond to the points s_i, i.e., $y_i = x_{i+k+1}$.

Proposition 2. *If $n = 2k+1$, then the optimal statistical estimate is*

$$d = \sum_{i=-k}^{k} \frac{3iy_i}{k(k+1)(2k+1)\Delta t}$$

and its error is equal to $\sigma / \left(\delta t \sqrt{nk(k+1)/3}\right)$.

In particular, for $n = 3$, the optimal statistical estimate is

$$d = \frac{y_1 - y_{-1}}{2\Delta t}$$

its error is equal to $\sigma/(\sqrt{2}\Delta t)$; for $n = 5$,

$$d = \frac{2y_2 + y_1 - y_{-1} - 2y_{-2}}{10\Delta t}$$

and $\sigma/(\sqrt{2}\Delta t)$; etc.

Let us now consider the case, when n is even: $n = 2k$ for some k. In this case, we will also denote the midpoint by s_0: $s_0 = (t_1 + t_n)/2 = t_1 + (n/2 - 1/2)\Delta t$. In this case, the distance between s_0 and t_i is not proportional to Δt, so it sounds reasonable to introduce the following denotations: we will denote t_i by s_j, where $j = (t_i - s_0)/\Delta t = i - (n+1)/2$, and correspondingly x_i by y_j for the same j. Then j runs from $-(n/2 - 1/2)$ to $+(n/2 - 1/2)$.

Proposition 3. *If $n = 2k$, then the optimal statistical estimate is*

$$d = \sum_{i=-(k-1/2)}^{k-1/2} \frac{12iy_i}{n(n^2-1)\Delta t}$$

and its error is equal to $\sigma/\left(\Delta t\sqrt{n(n^2-1)/12}\right)$.

In particular, for $n = 2$, the optimal statistical estimate is

$$d = \frac{y_{1/2} - y_{-1/2}}{\Delta t}$$

and error is equal to $\sigma/\Delta t$; for $n = 4$, the optimal statistical estimate is

$$d = \frac{(3/2)y_{3/2} + (1/2)y_{1/2} - (1/2)y_{-1/2} - (3/2)y_{-3/2}}{5\Delta t}$$

and error is equal to $\sigma/(\sqrt{5}\Delta t)$; etc.

Let us now consider real-time algorithms. The values of t_i are known beforehand and therefore, definitions from Section 7 can be simplified:

Definition 10. *Suppose that an integer b and real numbers t_1 and $\Delta t > 0$ are given. Suppose also that $x_1, x_2, \ldots, x_n, \ldots$ is a potentially infinite sequence of real numbers. By a statistical estimate for the derivative in the moment t_n, $n \geq b$, we mean the result of applying an optimal statistical estimate for the derivative to the values $x_n, x_{n-1}, \ldots, x_{n+1-b}$ and $t_n, t_{n-1}, \ldots, t_{n+1-b}$, where $t_i = t_1 + (i-1)\Delta t$.*

Definition 11. *Suppose that an integer $b > 1$ and a positive real number C are given. We say that an algorithm computes statistical estimates for the derivative in real time, with $\leq C$ computational steps per measurement if this algorithm works as follows: it reads x_1, then after $\leq C$ computational steps reads x_2,..., reads x_b, after $\leq C$ computational steps computes a statistical estimate for the derivative in the moment t_b, then reads x_{b+1}, after $\leq C$ computational steps computes a statistical estimate for the derivative in the moment t_{b+1}, etc.*

Remark. In case $t_{i+1} - t_i = \Delta t =$ const, the error does not depend on i or x_i, so it can be computed before any measurements are known. Therefore, we do not have to consider computations of the error if we are talking about real time algorithms.

Theorem 18. *There exists an algorithm that computes statistical estimates for the derivative in real time, with ≤ 10 computational steps per measurement.*

Remark. According to Propositions 2 and 3, we can precompute the coefficients a_i of the optimal statistical estimate, and thus computing this estimate would take at most $2b-1$ computational steps: b multiplications and $b-1$ additions. Therefore, if $b \leq 5$, and $2b-1 < 10$, it is better to use the direct formulas described after Propositions 2 and 3. If $b > 6$, then $2b-1 > 10$, and it is better to use the algorithm from Theorem 18.

Theorem 19. *There exists an algorithm that computes statistical estimates for the derivative in parallel in real time, with ≤ 5 computational steps per measurement.*

Remark. The gain in running time is even bigger than we can conclude from comparing these Theorems from the ones from the previous section. The reason is that in Section 7 we counted computation of a square root as 1 computational step, while in the present Section, we need only arithmetic operations, and arithmetic operations are much faster than computing square root. So, not only we need fewer computational steps, but the steps are shorter.

9. Optimal statistical estimates in case the function $x(t)$ is quadratic

Definition 12. *Assume that we are given an integer $n \geq 2$, a real number $\sigma > 0$, and $n+1$ real numbers t_0 and $t_1 < t_2 < \cdots < t_n$. By a linear estimate for an interval derivative at* *the point t_0, we mean a linear function that transforms a sequence $x_1, \dots, x_n$ into a value $d = a_1x_1 + a_2x_2 + \dots + a_nx_n$. We say that a statistical estimate is non-biased if for arbitrary real numbers a, b and c, and for arbitrary n independent random variables ξ_i with average 0 and standard deviation σ, the mathematical expectation $E[d] = E[a_1x_1 + a_2x_2 + \dots + a_nx_n]$ is equal to b, where $x_i = a + b(t_i - t_0) + c(t_i - t_0)^2 + \xi_i$. By an error of a statistical estimate we mean the standard deviation $\sigma[b - d]$ of the difference between b and d. We say that a statistical estimate is optimal if it is non-biased, and its error is the smallest possible (among non-biased estimates).*

Theorem 20. *If $x(t)$ can be quadratic, then the optimal statistical estimate is equal to $d = \sum_i a_ix_i$, where $a_i = \alpha + \beta(t_i - t_0) + \gamma(t_i - t_0)^2$, where α, β and γ are the solutions of the following system of linear equations:*

$$\begin{array}{lllllll}
\alpha \sum_i 1 & + & \beta \sum_i (t_i - t_0) & + & \gamma \sum_i (t_i - t_0)^2 & = & 0; \\
\alpha \sum_i (t_i - t_0) & + & \beta \sum_i (t_i - t_0)^2 & + & \gamma \sum_i (t_i - t_0)^3 & = & 1; \\
\alpha \sum_i (t_i - t_0)^2 & + & \beta \sum_i (t_i - t_0)^3 & + & \gamma \sum_i (t_i - t_0)^4 & = & 0.
\end{array}$$

Remark. In this case, we can repeat Definitions 7 and 8 to define real-time algorithms and their complexity. The result of this application is as follows:

Theorem 21 ($x(t)$ is quadratic). *There exists a constant $C > 0$ such that for every b, there exists an algorithm that computes statistical estimates for the derivative in real time, with $\leq C$ computational steps per measurement.*

Remark. In other words, this algorithm has the same nice property as an algorithm from Theorem 16: its running time does not depend on b, so, we can use arbitrarily big b, and still get the estimates in real time.

10. Linear formulas are not helpful for interval estimates of derivatives

Linear formulas are very simple to compute, so let's try them. In Section 6, we considered formulas that describe an estimate for the derivative as a linear combination of x_i: $d = \sum_i a_ix_i$. Computing a linear formula is computationally very easy, so it is reasonable to ask: can we use linear estimates in the interval case as well? We will prove that even in the simplest case, when the function is linear (i.e., $S = L$), and the precision is fixed ($\varepsilon_i = \varepsilon$), linear formulas are not helpful for interval estimates.

Definition 13. *Assume that we are given an integer $n \geq 2$, a real number $\varepsilon > 0$, and n real numbers $t_1 < t_2 < \dots < t_n$. By a linear estimate for an interval derivative we mean a linear function that transforms a sequence $x_1, \dots, x_n$ into a value $d = a_1x_1 + a_2x_2 + \dots + a_nx_n$. By a precision of a linear estimate we mean the maximum possible value of the difference $|b - d|$ for all a, b and x_i such that $|x_i - (a + bt_i)| \leq \varepsilon$.*

Remark. So, if the precision is 0.1, it means that the real values b of the derivative and the estimate d can differ by ≤ 0.1, and therefore, the possible values of the derivative belong to the interval $[d - 0.1, d + 0.1]$. Naturally, we are interested in the estimates with the best (i.e., the smallest) value of the precision.

Theorem 22. *For given n, $t_1, \dots, t_n$, a linear estimate has the smallest possible precision when $a_n = -a_1 = 1/(t_n - t_1)$, and $a_2 = a_3 = \dots = a_{n-1} = 0$.*

Remark. So, the best estimates for the linear formulas are obtained when we use only 2 measurements, and the resulting estimate is the simplest numerical differentiation $(x_n-x_1)/(t_n-t_1)$. In other words, if we restrict ourselves to linear estimates, then there is no way to use additional measurements to improve the precision of our estimates of the derivative. This means that in case the error of the simple numerical estimate is too big, linear formulas cannot help in diminishing this error. In other words, *linear formulas are not helpful for interval estimates of derivatives.*

Part III. Proofs

Proof of Proposition 1. Suppose that S is convex, and d_1 and d_2 are possible values of the derivative. This means that $d_1 = \dot{x}_1(t_0)$ and $d_2 = \dot{x}_2(t_0)$ for some functions $x_i \in S \cap \mathcal{F}$. Let us prove that for every $\alpha \in [0,1]$, $\alpha d_1 + (1-\alpha)d_2$ is also a possible value. Indeed, since S is convex, and (as is easy to prove) $\mathcal{F}$ is convex, the function $x(t) = \alpha x_1(t) + (1-\alpha)x_2(t)$ also belongs to $S \cap \mathcal{F}$, and therefore its derivative $dx(t_0)/dt = \alpha\, dx_1(t_0)/dt + (1-\alpha)\, dx_2(t_0)/dt = \alpha d_1 + (1-\alpha)d_2$ is a possible value. □

Proof of Theorem 1. A linear function $x(t) = a + bt$ belongs to a function interval $\mathcal{F}$ if and only if $x_i^- \le a + bt_i \le x_i^+$ for all i from 1 to n. We are interested in the values of $dx(t+0)/dt = b$, so let us eliminate a from these inequalities. To do that, suppose that b is given, and let us find whether there exists an a for which the function $x(t) = a + bt$ satisfies all these inequalities. If we move bt_i into the other side of the inequalities, we get the following equivalent inequalities: $x_i^- - bt_i \le a \le x_i^+ - bt_i$. So, a must be not bigger than all the numbers $x_i^+ - bt_i$, and not smaller than all the numbers $x_i^- - bt_i$. In other words, a must lie between $\max_j(x_j^- - bt_j)$ and $\min_i(x_i^+ - bt_i)$. Such a number exists if and only if $\min_i(x_i^+ - bt_i) \ge \max_j(x_j^- - bt_j)$, i.e., if and only if $x_i^+ - bt_i \ge x_j^- - bt_j$ for every i and j.

So, the problem is reduced to the following: find the interval of all values of b for which $x_i^+ - bt_i \ge x_j^- - bt_j$ for every i and j. By moving all terms with b to the right-hand side, we can transform this inequality into the following: $b(t_i - t_j) \le x_i^+ - x_j^-$. This inequality must be true for every pair (i,j). If $i = j$, it degenerates into $0 \le 0$, and is thus trivially true. If $i > j$, then $t_i > t_j$, and this inequality turns into $b \le (x_i^+ - x_j^-)/(t_i - t_j)$. When $i < j$, then $t_i < t_j$, and the above inequality turns into $b \ge (x_j^- - x_i^+)/(t_j - t_i)$. So, b must be not smaller than all the numbers $(x_j^- - x_i^+)/(t_j - t_i)$, and not greater than all the numbers $(x_i^+ - x_j^-)/(t_i - t_j)$. These inequalities are equivalent to the condition that b is not smaller than the biggest of its lower bounds $\max_{j>i}\left((x_j^- - x_i^+)/(t_j - t_i)\right)$, and not bigger than the smallest of its upper bounds $\min_{i>j}\left((x_i^+ - x_j^-)/(t_i - t_j)\right)$.

In other words, b is a possible value of the derivative if and only if $d^- \le b \le d^+$. Such values exist if and only if $d^- \le d^+$, and form the interval $[d^-, d^+]$. □

Theorem 2 was proved in the main text.

Proof of Theorem 3. Suppose that we have an unlimited number of processors (later on we will compute how many processors we actually need). First, let us compute $x_i^+ = x_i + \varepsilon_i$ and $x_i^- = x_i - \varepsilon_i$ for all i. This can be done in parallel on $2n$ processors. Since they are all working in parallel, we spend the time that is equivalent to only one computational step.

Now, we compute the values $(x_i^+ - x_j^-)$ and $(t_i - t_j)$ for $i > j$, and the values $(x_j^- - x_i^+)$ and $(t_j - t_i)$ for $j > i$. All these subtractions can also be done in parallel, and for that we need $2n(n-1)$ processors (two for each pair (i,j) except for $i=j$). The amount of time that we spend here is also equivalent to one computational step.

Next, we compute the ratios $(x_i^+ - x_j^-)/(t_i - t_j)$ and $(x_j^- - x_i^+)/(t_j - t_i)$. This can also be done in parallel, on $n(n-1)$ processors, and takes the time of one computational step.

After that, we have two sets of $n(n-1)/2$ numbers each; for the first set, we must find the biggest, for the second set, we must find the smallest. It is known (see, e.g., [8]), that for every N, the best method of finding the maximum of N numbers $p_1, \ldots, p_N$ in parallel takes $\leq N$ processors, and can be done in $\leq \log_2 N + 1$ computational steps. For completeness, let us add a description of this algorithm:

$N=2$. If $N=2$, then we simply compare the two numbers p_1 and p_2.

$N=4$. If $N=4$, then we divide the numbers into pairs (p_1, p_2) and (p_3, p_4). Then, first, we find the maximum for each pair: i.e., $\max(p_1,p_2)$ and $\max(p_3,p_4)$ (in parallel), and second, compare the two results to find the maximum of all four numbers p_i.

$N=8$. If $N=8$, then we divide the eight numbers into 4 pairs: (p_1,p_2), (p_3,p_4), (p_5,p_6) and (p_7,p_8). Then, for each pair, we compute its maximum: $q_1 = \max(p_1,p_2)$, $q_2 = \max(p_3,p_4)$, $q_3 = \max(p_5,p_6)$ and $q_4 = \max(p_7,p_8)$. This can be done in parallel, and therefore takes the times of one computational step. After that, we find the maximum of the resulting 4 numbers q_1, q_2, q_3, q_4 (we already know that this can be done in 4 steps).

$N=2^k$. If $N=2^k$ for some k, then we have the following recursive procedure:

> If $k=0$, then the only number present is the biggest;
>
> If $k>0$, then we divide N numbers into 2^{k-1} pairs, in one step find the biggest of all pairs, and then apply the same algorithm to find the biggest of 2^{k-1} results.
>
> After each step, we halve the set of numbers that we still need to compare, so in k steps we get the maximum.

$N \neq 2^k$. Let us now consider the remaining case, when $N \neq 2^k$. In this case, we take the smallest number 2^k that is $\geq N$, add $N - 2^k$ numbers that are equal to p_1, and apply the same algorithm.

Since k is the smallest for which $N \leq 2^k$, we have $N > 2^{k-1}$, hence $\log_2 N > k-1$, and $k < \log_2 +1$. So, we need $\leq N$ processors and $\leq \log_2 N + 1$ computational steps. We can compute the minimum (to compute d^+) and the maximum (to compute d^-) in parallel. Each computation takes $\leq \log_2 \big(n(n-1)/2\big) + 1 < \log_2(n^2/2) + 1 = 2\log_2 n - 1 + 1 = 2\log_2 n$ computational steps, and takes $\leq n(n-1)/2$ processors. To compute both estimates in parallel, we need $\leq n(n-1)$ processors.

After we computed d^+ and d^-, we must compare them to check whether $d^+ \geq d^-$ and thus whether $\mathcal{F}$ is compatible with L. This takes one more step.

Now we have described all the stages of our computation. In total, we need $\leq 1 + 1 + 2\log_2 n + 1 = 2\log_2 n + 3$ computational steps, and $\leq \max\big(2n, 2n(n-1), n(n-1)\big) = 2n(n-1)$ processors. □

Proof of Theorem 4. In the algorithm from Theorem 2, the biggest part of computational time was spent on computing the expressions $(x_i^- - x_j^+)/(t_i - t_j)$ and $(x_i^+ - x_j^-)/(t_i - t_j)$. In our case, we need these expressions for $n-b<j<i\le n$. After we computed the estimates d_n^+ and d_n^-, we read the new values x_{n+1} and t_{n+1}. To compute d_{n+1}^+ and d_{n+1}^-, we must use the similar expressions for $n+1-b<j<i\le n+1$. But the values of these expressions for $i<n+1$ have already been computed on the previous stage, so we do not need to compute them again. The only new values that we have to compute correspond to $i=n+1$.

So, the new algorithm is as follows: for every n, we follow the same steps as in algorithm from Theorem 2, with the only difference that the values of x_i^-, x_j^+, and the expressions $(x_i^- - x_j^+)/(t_i-t_j)$ and $(x_i^+ - x_j^-)/(t_i-t_j)$ that we have already computed on the previous stage (for d_n^+ and d_n^-) we can use for d_{n+1}^+ and d_{n+1}^- as well.

Let us estimate the number of computational steps per measurement for this algorithm. For a new value t_{n+1}, we must compute $(b-2)$ new pairs of expressions (one pair for each j that lie between $n+1-b$ and $n+1$). To compute each pair, we need 5 operations, so totally, we need $5(b-2)$ computational steps.

We must add to this amount 2 computational steps to compute $x_{n+1}^+ = x_{n+1}+\varepsilon_{n+1}$ and $x_{n+1}^- = x_{n+1}-\varepsilon_{n+1}$, and $2\big(b(b-1)/2-1\big)$ steps to find the smallest and the biggest values (d_{n+1}^+ and d_{n+1}^-). As a result, we get $2+5(b-2)+2\big(n(n-1)/2-1\big) = 2+5b-10+b^2-b-2 = b^2+4b-10.$ □

Theorem 5 is proved in the main text.

Proof of Theorem 6. We consider the case when a function $x(t)$ is quadratic, i.e., when $x(t)=a+b(t-t_0)+c(t-t_0)^2$. In such a representation, the value of the derivative $dx(t)/dt$ in the point t_0 equals b.

If $n=2$, then for every b and arbitrary values x_1, x_2, $\varepsilon_1>0$ and $\varepsilon_2>0$, we can easily find a function $x(t)$ that belongs to both Q and $\mathcal{F}$: indeed, it is sufficient to find a and c for which $a+b(t_1-t_0)+c(t_1-t_0)^2=x_1$ and $a+b(t_2-t_0)+c(t_2-t_0)^2=x_2$. These two equalities form a system of 2 linear equations with 2 unknowns a and b whose determinant is different from 0, therefore this system always has a solution.

The same is true for $n=1$. So, if $n\le 2$, then $\mathcal{F}$ is always consistent with Q, and any value of the derivative is possible, i.e., the derivative set is indeed equal to $(-\infty,+\infty)$.

Let us now consider the case $n>2$. A quadratic function $x(t)$ belongs to a function interval $\mathcal{F}$ if and only if $x_i^- \le a+b(t_i-t_0)+c(t_i-t_0)^2 \le x_i^+$ for all i from 1 to n. We are interested in the value of b, so let us reformulate these inequalities in an equivalent form that does not contain a and c. In other words, let us eliminate a and c from these inequalities.

First, let us eliminate a. This can be done in the same manner as in the proof of Theorem 1. Indeed, suppose that b and c are given. When does there exist an a for which the above inequalities are true? If we move terms containing b and c to the other sides of the inequalities, we will obtain equivalent restrictions on a:

$$x_i^- - b(t_i-t_0) - c(t_i-t_0)^2 \le a \le x_i^+ - b(t_i-t_0) - c(t_i-t_0)^2.$$

So, a must be not smaller than all the left-hand side expressions, and not bigger than all the right-hand side expressions. For such an a to exist, every left-hand side expression must be not be greater than the right-hand side one, i.e., for every i and j the following inequality must be true:

$$x_i^- - b(t_i-t_0) - c(t_i-t_0)^2 \le x_j^+ - b(t_j-t_0) - c(t_j-t_0)^2.$$

Since the new set of inequalities is equivalent to the old set, the problem of finding the derivative set can be now reformulated as follows: we must find all b for which there exists a

c such that the inequalities

$$x_i^- - b(t_i - t_0) - c(t_i - t_0)^2 \le x_j^+ - b(t_j - t_0) - c(t_j - t_0)^2$$

are true for all i and j.

Let us now eliminate c from this new set of inequalities. If we move all terms with c into one side, and all other terms into another side, we get the following inequality:

$$(x_i^- - x_j^+) - \big(b(t_i - t_0) - b(t_j - t_0)\big) \le c(t_i - t_0)^2 - c(t_j - t_0)^2.$$

Since

$$b(t_i - t_0) - b(t_j - t_0) = b(t_i - t_0 - t_j + t_0) = b(t_i - t_j)$$

and

$$c(t_i - t_0)^2 - c(t_j - t_0)^2 = c\big((t_i - t_0)^2 - (t_j - t_0)^2\big)$$

$$= c\big((t_i - t_0) + (t_j - t_0)\big)\big((t_i - t_0) - (t_j - t_0)\big) = c(t_i + t_j - 2t_0)(t_i - t_j)$$

we can transform this inequality into the following simplified equivalent one:

$$(x_i^- - x_j^+) - b(t_i - t_j) \le c(t_i + t_j - 2t_0)(t_i - t_j).$$

For every pair (i, j), we have this inequality, and we also have the inequality that is obtained from this one by changing i and j.

The resulting inequality for c depends on whether the coefficient at c (equal to $(t_i - t_0)^2 - (t_j - t_0)^2$) is negative, positive, or equal to 0.

Let us first consider the case when $(t_i - t_0)^2 = (t_j - t_0)^2$. Suppose for certainty that $i > j$. Then, we have two inequalities $(x_i^- - x_j^+) - b(t_i - t_j) \le 0$, and $(x_j^- - x_i^+) - b(t_j - t_i) \le 0$. Since $i > j$, we can reformulate these inequality as explicit restrictions on b: $(x_i^- - x_j^+)/(t_i - t_j) \le b \le (x_i^+ - x_j^-)/(t_i - t_j)$.

Now, let us consider the remaining case, when $(t_i - t_0)^2 \ne (t_j - t_0)^2$. For certainty (and without losing generality), let us assume that $(t_i - t_0)^2 > (t_j - t_0)^2$. Then the i, j inequality is equivalent to the following:

$$\frac{(x_i^- - x_j^+) - b(t_i - t_j)}{(t_i + t_j - 2t_0)(t_i - t_j)} \le c$$

and the j, i inequality is equivalent to the following one:

$$c \le \frac{(x_i^+ - x_j^-) - b(t_i - t_j)}{(t_i + t_j - 2t_0)(t_i - t_j)}.$$

So, for a given b, the value of c for which all the desired inequalities are true, exists if and only if first, all the inequalities that do not contain c are true, and, second, any lower bound for c is not greater than any upper bound for c, i.e.,

$$\frac{(x_i^- - x_j^+) - b(t_i - t_j)}{(t_i + t_j - 2t_0)(t_i - t_j)} \le \frac{(x_k^+ - x_l^-) - b(t_k - t_l)}{(t_k + t_l - 2t_0)(t_k - t_l)}$$

for each quadruple (i, j, k, l) such that $(t_i - t_0)^2 > (t_j - t_0)^2$ and $(t_k - t_0)^2 > (t_l - t_0)^2$.

We can express this inequality directly in terms of b by moving all terms with b into one side of the equation:

$$\frac{x_i^- - x_j^+}{(t_i+t_j-2t_0)(t_i-t_j)} - \frac{x_k^+ - x_l^-}{(t_k+t_l-2t_0)(t_k-t_l)}$$
$$\le \frac{b(t_i-t_j)}{(t_i+t_j-2t_0)(t_i-t_j)} - \frac{b(t_k-t_l)}{(t_k+t_l-2t_0)(t_k-t_l)}.$$

Canceling similar terms in the numerator and the denominator of the fractions that are coefficients at b, we arrive at the following simplified inequality:

$$\frac{x_i^- - x_j^+}{(t_i+t_j-2t_0)(t_i-t_j)} - \frac{x_k^+ - x_l^-}{(t_k+t_l-2t_0)(t_k-t_l)} \le b\left[\frac{1}{t_i+t_j-2t_0} - \frac{1}{t_k+t_l-2t_0}\right].$$

If $t_i+t_j-2t_0 = t_k+t_l-2t_0$, then the coefficient at b is equal to 0, and therefore, this inequality turns into the following one:

$$\frac{x_i^- - x_j^+}{(t_i+t_j-2t_0)(t_i-t_j)} - \frac{x_k^+ - x_l^-}{(t_k+t_l-2t_0)(t_k-t_l)} \le 0.$$

If $t_i+t_j-2t_0 = t_k+t_l-2t_0 > 0$, then by multiplying both sides of the inequality by this number we conclude that $(x_i^- - x_j^+)/(t_i-t_j) - (x_k^+ - x_l^-)/(t_k-t_l) \le 0$, or $(x_i^- - x_j^+)/(t_i-t_j) \le (x_k^+ - x_l^-)/(t_k-t_l)$. If $t_i+t_j-2t_0 = t_k+t_l-2t_0 < 0$, then we get the inequality $(x_i^- - x_j^+)/(t_i-t_j) \ge (x_k^+ - x_l^-)/(t_k-t_l)$.

Let us now consider the remaining case, when $t_i+t_j-2t_0 \ne t_k+t_l-2t_0$. In this case, we actually have two inequalities: the one that we gave above, and the one that is obtained by changing i,j to k,l and vice versa. So, without losing generality, we can assume that $t_i+t_j-2t_0 > t_k+t_l-2t_0$. In this case, $1/(t_i+t_j-2t_0) < 1/(t_k+t_l-2t_0)$, the coefficient at b is negative, and the inequality turns into the following:

$$\left[\frac{x_i^- - x_j^+}{(t_i+t_j-2t_0)(t_i-t_j)} - \frac{x_k^+ - x_l^-}{(t_k+t_l-2t_0)(t_k-t_l)}\right] / \left[\frac{1}{t_i+t_j-2t_0} - \frac{1}{t_k+t_l-2t_0}\right] \ge b.$$

For the inequality that is obtained by changing i,j to k,l, the coefficient at b is positive, so we get the following inequality:

$$\left[\frac{x_i^+ - x_j^-}{(t_i+t_j-2t_0)(t_i-t_j)} - \frac{x_k^- - x_l^+}{(t_k+t_l-2t_0)(t_k-t_l)}\right] / \left[\frac{1}{t_i+t_j-2t_0} - \frac{1}{t_k+t_l-2t_0}\right] \le b.$$

We also have the bounds for b from the cases when $(t_i-t_0)^2 = (t_j-t_0)^2$:

$$(x_i^- - x_j^+)/(t_i-t_j) \le b \le (x_i^+ - x_j^-)/(t_i-t_j).$$

So, we reduced the problem to the inequalities that do not depend on b at all, and inequalities of the type $X \le b$ and $b \le Y$ for some expressions X and Y. The inequalities that contain b are consistent if and only if the maximum of the lower bounds X is not bigger than the minimum of all upper bounds, and, if they are consistent, then the interval of possible values of b coincides with the set of all possible values between these maximum and minimum. □

Proof of Theorem 7. The formulas from Theorem 6 prompt the following natural algorithm:

1) First, we compute the values $x_i^+ = x_i + \varepsilon_i$, $x_i^- = x_i - \varepsilon$, $t_i - t_0$ and $(t_i - t_0)^2$ for all i from 1 to n.

2) For every $i > j$, we check whether $(t_i - t_0)^2 = (t_j - t_0)^2$.

2a) For all $i > j$, for which this equality is true, we compute the values $t_i - t_j$, $x_i^+ - x_j^-$, $x_i^- - x_j^+$, $(x_i^+ - x_j^-)/(t_i - t_j)$, and $(x_i^- - x_j^+)/(t_i - t_j)$.

2b) If $(t_i - t_0)^2 \neq (t_j - t_0)^2$, we permute i and j if necessary $\left(\text{so that } (t_i - t_0)^2 > (t_j - t_0)^2\right)$ and compute the values $t_i - t_j$, $(t_i - t_0) + (t_j - t_0) = t_i + t_j - 2t_0$, $1/(t_i + t_j - 2t_0)$, $x_i^+ - x_j^-$, $x_i^- - x_j^+$, $(x_i^+ - x_j^-)/(t_i - t_j)$, $(x_i^- - x_j^+)/(t_i - t_j)$, $(x_i^+ - x_j^-)/\left((t_i - t_j)(t_i + t_j - 2t_0)\right)$, and $(x_i^- - x_j^+)/\left((t_i - t_j)(t_i + t_j - 2t_0)\right)$.

3) Enumerate all the pairs i, j and k, l such that $(t_i - t_0)^2 > (t_j - t_0)^2$ and $(t_k - t_0)^2 > (t_l - t_0)^2$. For each such quadruple, compare $t_i + t_j - 2t_0$ with $t_k + t_l - 2t_0$.

3a) If $t_i + t_j - 2t_0 = t_k + t_l - 2t_0 > 0$, then check whether $(x_k^+ - x_l^-)/(t_i - t_j) \geq (x_i^- - x_j^+)/(t_i - t_j)$. If this inequality is not true, then $\mathcal{F}$ is inconsistent with Q.

3b) If $t_i + t_j - 2t_0 = t_k + t_l - 2t_0 < 0$, then check whether $(x_j^+ - x_i^-)/(t_j - t_i) \geq (x_l^- - x_k^+)/(t_l - t_k)$. If this inequality is not true, then $\mathcal{F}$ is inconsistent with Q.

3c) If $t_i + t_j - 2t_0 < t_k + t_l - 2t_0$, change i, j to k, l and vice versa, so that after that change we will have $t_i + t_j - 2t_0 > t_k + t_l - 2t_0$.

3d) If $t_i + t_j - 2t_0 > t_k + t_l - 2t_0$, then compute N_{ijkl}^+, N_{ijkl}^-, D_{ijkl}, N_{ijkl}^+/D_{ijkl}, and N_{ijkl}^-/D_{ijkl} according to the formulas from the formulation of Theorem 6.

4) After we have done the previous steps, we can compute $d_1^- = \max\left((x_i^- - x_j^+)/(t_i - t_j)\right)$, $d_1^+ = \min\left((x_i^+ - x_j^-)/(t_i - t_j)\right)$, $d_2^- = \max(N_{ijkl}^-/D_{ijkl})$, $d_2^+ = \min(N_{ijkl}^+/D_{ijkl})$, $d^+ = \min(d_1^+, d_2^+)$, and $d^- = \max(d_1^-, d_2^-)$. If $d^+ < d^-$, then $\mathcal{F}$ is not compatible with Q, else $[d^-, d^+]$ is the desired derivative set.

Let us now estimate the number of computational steps for this algorithm:

1) The first part of the algorithm requires 4 steps for every i from 1 to n, i.e., overall, $4n$ steps;

2) For every pair $i > j$, we make 1 comparison and then (depending on the result of this comparison) either 5 or 9 computational steps. Therefore, for each pair, we need at most 10 computational steps. Since there are $n(n-1)/2$ pairs with $i > j$, we need at most $10n(n-1)/2 = 5n(n-1)$ computational steps for this part of the algorithm.

3) For each quadruple (i, j, k, l), we need 1 comparison, and then, depending on the result of this comparison, either 1 more comparison (in cases 3a and 3b), or 5 computational steps (in cases 3c and 3d). So, for every quadruple, we need at most 6 computational steps.

To estimate the total number of steps for this part of the algorithm, we must multiply 6 by the total number of quadruples. A quadruple (i,j,k,l) is a pair of different pairs (i,j) and (k,l). So, the total number of quadruples is equal to $P(P-1)/2$, where P is the total number of pairs with the property $(t_i-t_0)^2 > (t_j-t^0)^2$. This number P is not greater than the total number of pairs $n(n-1)/2$, so the number of quadruples is $\le P^2/2 \le \big(n(n-1)/2\big)^2/2$. Hence, the total number of computational steps for this part of the algorithm is $\le 6\big(n(n-1)/2\big)^2/2 = 3\big(n(n-1)/2\big)^2$.

4) To compute the minimum d_2^+ of $N \le 1/2\big(n(n-1)/2\big)^2$ numbers, we need $\le N$ computational steps. Likewise, the number of steps for computing the maximum d_2^- is $\le 1/2\big(n(n-1)/2\big)^2$. To compute the values d_1^+ and d_1^-, we need $\le n(n-1)/2$ computational steps. Then, we need two steps to compute d^+ and d^-. So, totally, we need $\le \big(n(n-1)/2\big)^2 + n(n-1)/2$ computational steps for this final part of the algorithm.

By adding all these numbers we can now get the estimate for the total amount of computational steps for the entire algorithm: it is $\le S$, where $S = 4n + 5n(n-1) + 3\big(n(n-1)/2\big)^2 + \big(n(n-1)/2\big)^2 + n(n-1)/2 = 4n + 11n(n-1)/2 + 4\big(n(n-1)/2\big)^2$. If we compute all the expression in the parentheses, we arrive at the following result: $S = 4n + 11n^2 - 11n + n^4 - 2n^3 + n^2$. Adding all the coefficients at 1, n, n^2, n^3, and n^4, we arrive at the following formula: $S = n^4 - 2n^3 + 12n^2 - 7n < n^4 + 12n^2$. $\square$

Proof of Theorem 8. We can compute all the values from parts 2), 3) of the previous proof in parallel: for that we need $\le n^4$ processors (one for each quadruple). Totally, to compute all these expression, we need the time that is necessary for 6 computational steps:

1) On 1st step, we compute $x_i^+ = x_i + \varepsilon_i$, $x_i^- = x_i - \varepsilon$, $t_i - t_0$, and $t_i - t_j$ for all i and j.

2) On the 2nd step, we compute $(t_i-t_0)^2$, $(t_i-t_0)+(t_j-t_0) = t_i + t_j - 2t_0$, $x_i^+ - x_j^-$, and $x_i^- - x_j^+$ for all i and j.

3) On the 3rd step, for every $i > j$, we check whether $(t_i-t_0)^2 = (t_j-t^0)^2$. For all $i > j$, for which this equality is true, we compute the values $(x_i^+ - x_j^-)/(t_i-t_j)$ and $(x_i^- - x_j^+)/(t_i-t_j)$. If $(t_i-t_0)^2 \ne (t_j-t^0)^2$, we permute i and j if necessary (so that $(t_i-t_0)^2 > (t_j-t^0)^2$) and compute the values $1/(t_i+t_j-2t_0)$, $(x_i^+ - x_j^-)/(t_i-t_j)$, $(x_i^- - x_j^+)/(t_i-t_j)$.

4) On the 4th step, we compute $(x_i^+ - x_j^-)/\big((t_i-t_j)(t_i+t_j-2t_0)\big)$ and $(x_i^- - x_j^+)/\big((t_i-t_j)(t_i+t_j-2t_0)\big)$ for all i,j.

5) On the 5th step, we compute the values $N_{ijkl}^{\pm}$ and D_{ijkl} for all i,j,k,l.

6) On the 6th step, we compute all the fractions $N_{ijkl}^{\pm}/D_{ijkl}$.

Then, we must find the maximum and the minimum of $\le \big(n(n-1)\big)^2 + n(n-1)/2$ numbers. This amount is $\le 1/4(n^4 - 2n^3 + n^2) + (1/2)n^2 - (1/2)n = (1/4)n^4 - (1/2)n^3 + (1/4)n^2 + (1/2)n^2 - (1/2)n = (1/4)n^4 - (1/2)n^3 + (3/4)n^2 - (1/2)n$. Since $n \ge 3$, $n^3 \ge 3n^2$, so

$(1/2)n^3 \geq (3/2)n^2 > (3/4)n^2$, so $(1/4)n^4 - (1/2)n^3 + (3/4)n^2 - (1/2)n = (1/4)n^4 - \left((1/2)n^3 - (3/4)n^2\right) - (1/2)n < (1/4)n^4$.

Computing maximum and minimum can be done in parallel, so it is sufficient to estimate the time that is necessary to compute one of them, and then add 1 step for comparison of d^+ and d^-. To find the smallest of $\leq (1/4)n^4$ numbers, we need $\leq \log_2\left((1/4)n^4\right) + 1 = 4\log_2 n - 2 + 1 = 4\log_2 n - 1$ steps.

Totally, we need $\leq 6 + (4\log_2 n - 1) = 4\log_2 n + 5$ computational steps. □

Theorems 9 and 10 were proved in the main text.

Proof of Theorem 11. Let us first figure out what conditions on a_i are equivalent to the demand that a statistical estimate is non-biased. This demand means that $E[d] = E[a_1x_1 + \cdots + a_nx_n] = \sum_i a_iE[x_i] = b$ for all a, b and $x_i = a + bt_i + \xi_i$. Since $E[\xi_i] = 0$, we conclude that $E[x_i] = a + bt_i$, hence $E[d] = \sum_i a_i(a + bt_i) = a\left(\sum_i a_i\right) + b\left(\sum_i t_i\right)$. So, the condition that $E[d] = b$ for all a and b means that $a\left(\sum_i a_i\right) + b\left(\sum_i t_i\right) = b$ for all a and b. Two linear function are identical if and only if their coefficients coincide, therefore, this condition is true if $\sum_i a_i = 0$ and $\sum_i a_it_i = 1$. So, these two equalities are necessary and sufficient for an estimate to be non-biased.

In the following we will consider only non-biased estimates. Let us compute the error for such estimates. Since ξ_i are assumed to be independent, we can conclude that $\sigma^2[d] = \sigma^2\left[\sum_i a_ix_i\right] = \sum_i \sigma^2[a_ix_i]$. But $x_i = a + bt_i + \xi_i$, therefore, $E[x_i] = a + bt_i$, $x_i - E[x_i] = \xi_i$, $\sigma^2[x_i] = E\left[(x_i - E[x_i])^2\right] = E[\xi_i^2] = \sigma^2$, hence, $\sigma^2[a_ix_i] = a_i^2\sigma^2$ and $\sigma^2[d] = \sum_i \sigma^2[a_ix_i] = \sigma^2\left(\sum_i a_i^2\right)$.

So, the error is proportional to $\sqrt{\sum_i a_i^2}$. Hence, the error is the smallest possible if and only if the sum $\sum_i a_i^2$ is the smallest possible.

So, the problem of finding the non-biased statistical estimate with the smallest possible error, is equivalent to the following mathematical problem: to minimize a function $\sum_i a_i^2$ under the conditions $\sum_i a_i = 0$ and $\sum_i a_it_i = 1$.

The Lagrange multipliers method allows us to reduce this problem to the unconditional optimization problem $F = \sum_i a_i^2 + \lambda_1\sum_i a_i + \lambda_2\left(\sum_i a_it_i - 1\right) \rightarrow \min$ for some constants λ_i. This problem can be easily solved by equating the derivative $\partial F/\partial a_i$ of the minimized function F to 0: $2a_i + \lambda_1 + \lambda_2t_i = 0$. As a result, we get the expression $a_i = \alpha + \beta t_i$, where $\alpha = -1/2\lambda_1$ and $\beta = -1/2\lambda_2$. To find α and β, let us substitute these expressions into the conditions $\sum_i a_i = 0$ and $\sum_i a_it_i = 1$. As a result, we get the following system of equations:

$$\alpha\sum_i 1 + \beta\sum_i t_i = 0,$$

$$\alpha\sum_i t_i + \beta\sum_i t_i^2 = 1.$$

From the first equation, we conclude that $\alpha = -\beta\left(\sum_i t_i\right)/\left(\sum_i 1\right) = -\beta\left(\sum_i t_i\right)/n = -\beta\bar{t}$. If we substitute this expression into the second equation, we conclude that

$$\beta\left(\sum_i t_i^2 - \left(\sum_i t_i\right)\bar{t}/n\right) = 1.$$

The coefficient at β can be easily represented as $\sum_i(t_i - \bar{t})^2 = n\sigma_t^2$, therefore, $\beta = 1/(n\sigma_t^2)$. Hence, $\alpha = -\bar{t}/(n\sigma_t^2)$, and $a_i = \alpha + \beta t_i = (t_i - \bar{t})/(n\sigma_t^2)$.

So, we have deduced the expressions for a_i that correspond to the optimal statistical estimate. Let us now find the error of this estimate. This error is equal to $\sqrt{\sigma^2 \sum_i a_i^2} = \sigma\sqrt{\sum_i a_i^2}$. Substituting the above expression for a_i into $\sum_i a_i^2$, we conclude that $\sum_i a_i^2 = \sum_i \left((t_i - \bar{t})^2\right)^2/(\sigma_t^2 n)^2 = \left(\sum_i (t_i - \bar{t})^2\right)^2/(\sigma_t^2 n)^2$. But $\left(\sum_i (t_i - \bar{t})^2\right)^2 = n\sigma_t^2$, therefore $\sum_i a_i^2 = 1/(\sigma_t^2 n^2)$, and the error is equal to $\sigma\sqrt{\sum_i a_i^2} = \sigma/(\sigma_t n)$. □

Proof of Theorem 12. If we know a_i, then after measuring x_i, we need only $2n - 1$ computational steps to compute the estimate $d = \sum_i a_i x_i$: n multiplications and $n-1$ additions. □

Proof of Theorem 13. If we have several computers working in parallel, then we can make all the multiplications in parallel (taking the time of only 1 computational step), and then compute the sum of the resulting n products in the time of $\le \log_2 n + 1$ computational steps (such an algorithm is given, e.g., in [8]) The idea of such an algorithm that parallelizes addition is very close to the idea of finding maximum in $\log_2 n + 1$ steps (see the proof of Theorem 3): if $n = 2^k$ for some k, then we divide n products $p_1 = a_1 x_1, p_2 = a_2 x_2, \ldots$ into pairs; on 1st step, we add each pair, getting the results $p_1 + p_2, p_3 + p_4, \ldots, p_{n-1} + p_n$; on 2nd step, we divide these $n/2$ results into pairs, and compute the sum of each pair (getting $p_1 + p_2 + p_3 + p_4$, $p_5 + p_6 + p_6 + p_7 + p_8, \ldots$), etc. □

Proof of Theorem 14. In this case, for a sequential algorithm, we need $n - 1$ computational steps to compute the sum $\sum_i t_i$, 1 to compute $\bar{t}$, $2n$ to compute $t_i - \bar{t}$ and $(t_i - \bar{t})^2$, $n - 1$ to compute the sum $\sum_i (t_i - \bar{t})^2$, n to compute $a_i = (t_i - \bar{t})/\left(\sum_i (t_i - \bar{t})^2\right)$, and $2n - 1$ to compute $d = \sum_i a_i x_i$: totally, we need $7n - 2$ computational steps. □

Proof of Theorem 15. For a parallel computer, we need $\le \log_2 n + 1$ steps to compute $\sum_i t_i$, 1 step to compute $\bar{t}$, 2 steps to compute $t_i - \bar{t}$ and $(t_i - \bar{t})^2$ (because computations for different i can be done in parallel), $\log_2 n + 1$ steps to compute the sum $\sum_i (t_i - \bar{t})^2$, 1 step to compute $a_i = (t_i - \bar{t})/\left(\sum_i (t_i - \bar{t})^2\right)$ for all i, and $\log_2 n + 2$ steps to compute $d = \sum_i a_i x_i$. Totally, we need $3\log_2 n + 8$ computational steps. □

Proof of Theorem 16. Let us first reformulate the formula for d from Theorem 11, so that is will become easier to compute in real time. According to Theorem 11, the optimal statistical estimate is equal to $d = \sum_i a_i x_i$, where $a_i = (t_i - \bar{t})/(n\sigma_t^2)$ and $\sigma_t^2 = \left(\sum_i (t_i - \bar{t})^2\right)/n$. If we substitute the expression $\sigma_t^2 = \left(\sum_i (t_i - \bar{t})^2\right)/n$ into the formula for a_i, we conclude that $a_i = (t_i - \bar{t})/\left(\sum_i (t_i - \bar{t})^2\right)$. If we use the easily verifiable fact that $\sum_i (t_i - \bar{t})^2 = \sum_i t_i^2/n - n\bar{t}^2$, then we arrive at a formula $a_i = (t_i - \bar{t})/\left(\sum_i t_i^2 - n\bar{t}^2\right)$. Finally, if we substitute the expression for $\bar{t}$, we conclude that

$$a_i = \left(t_i - \left(\sum_i t_i\right)/n\right)/\left(\sum_i t_i^2 - n\left(\sum_i t_i\right)^2\right).$$

Therefore, $d = \sum_i a_i x_i = \left(\sum_i x_i t_i - (\sum_i x_i)(\sum_i t_i)/n\right)/\left(\sum_i t_i^2 - (\sum_i t_i)^2/n\right)$. We can also express it in the following form: $d = (S_1 - S_2 S_3/n)/(S_4 - S_3^2/n)$, where we denoted $S_1 = \sum_i x_i t_i$, $S_2 = \sum_i x_i$, $S_3 = \sum_i t_i$, and $S_4 = \sum_i t_i^2$. In these terms, the error is equal to $e = \sigma/\sqrt{\sum_i t_i^2 - n\bar{t}^2} = \sigma/\sqrt{S_4 - S_3^2/n}$

For our case, a statistical estimate d_n for a derivative in the moment t_n, is equal to $d_n = (S_{1,n} - S_{2,n}S_{3,n}/b)/(S_{4\,n} - S_{3,n}^2/b)$, where

$$\begin{aligned} S_{1,n} &= \sum_{i=n+1-b}^{n} x_i t_i, & S_{2,n} &= \sum_{i=n+1-b}^{n} x_i, \\ S_{3,n} &= \sum_{i=n+1-b}^{n} t_i, & S_{4,n} &= \sum_{i=n+1-b}^{n} t_i^2. \end{aligned}$$

It is easy to check that $S_{1,n+1} = S_{1,n} + x_{n+1}t_{n+1} - x_{n+1-b}t_{n+1-b}$, $S_{2,n+1} = S_{2,n} + x_{n+1} - x_{n+1-b}$, $S_{3,n+1} = S_{3,n} + t_{n+1} - t_{n+1-b}$, and $S_{4,n+1} = S_{4,n} + t_{n+1}^2 - t_{n+1-b}^2$. So, we can apply the following algorithm:

1) We reserve 4 variables S_1, S_2, S_3, S_4 that will contain the current values of the sums $S_{1,n}$. For every moment t_n, we also keep the values x_i, $x_i t_i$, t_i, and t_i^2 for all i such that $n - b < i \le n$. So, we must reserve $4 + 4b$ variables.

2) The initial values of $S_{i,n}$ are 0. When $n < b$ then after reading x_n and t_n, we compute (and remember) the values $x_n t_n$ and t_n^2, and update the values S_i in the following manner: $S_1 := S_1 + x_n t_n$, $S_2 := S_2 + x_n$, $S_3 := S_3 + t_n$, and $S_4 := S_4 + t_n^2$.

3) For $n = b$, we do this update, and also compute d_n: $\bar t := S_3/b$, $S := S_4 - S_3\bar t$, $d_n := (S_1 - S_2\bar t)/S$, and $e := \sigma/\sqrt{S}$.

4) For $n > b$, we must compute (and remember) the values $x_n t_n$ and t_n^2, and then update S_i according to the following formulas: $S_1 := S_1 + x_n t_n - x_{n-b}t_{n-b}$, $S_2 := S_2 + x_n - x_{n-b}$, $S_3 = S_3 + t_n - t_{n-b}$, and $S_4 := S_4 + t_n^2 - t_{n-b}^2$. After that we compute d_n and e using the same formulas as for $n = b$.

For $n < b$, we need 6 computational steps; for $n = b$, we need $6+6 = 12$ steps to compute d_n, and one division + one square root to compute e. For $n > b$, we need 10 computational steps for an update, 6 to compute d_n, and 1 division + 1 square root to compute e. In all cases, we need at most 17 arithmetic operations + 1 square root per measurement. □

Remark. The algorithm described in this proof is vulnerable to errors: if accidentally we input a wrong value of t_i for some i, then, we spoil S_1, S_3, S_4, and thus spoil the resulting estimates d_n for arbitrary big n. To make this algorithm more error-prone, we can periodically check its results by applying the formulas from Theorem 11 directly. This checking will practically not slow down the computations, since we do not need to do it for every measurement, just periodically (e.g., for every tenth or every hundredth measurement).

Proof of Theorem 17. In this case, we can use an algorithm similar to the one from the proof of Theorem 16. Namely, if we have several processors that can work in parallel, then we can compute $x_n t_n$ and t_n^2 in parallel (thus, in the time that is necessary for 1 computational step), update the values S_i in parallel on 4 processors (thus spending the time of 2 computational steps), and then compute the value d_n as follows: first, compute $\bar t := S_3/b$ (1 step), then $S := S_4 - S_3\bar t$, $s := \sqrt{N}$, and $N := S_1 - S_2\bar t$ in parallel (2 steps), and $d_n := N/S$ and $e := \sigma/s$ (1 step). Totally, we need the time of 7 computational steps. □

Proof of Proposition 2. In this case, $\bar t = s_0$, and

$$\sigma_t^2 = 1/(2k+1)\sum_i (t_i - \bar t)^2 = \big(1/(2k+1)\big)\sum_{i=1}^{k}(i\Delta t)^2 + \sum_{i=1}^{k}(i\Delta t)^2 = \big(2\Delta t^2/(2k+1)\big)\sum_{i=1}^{k} i^2.$$

The value of the sum $\sum_{i=1}^{k} i^2$ is known to be $k(k+1)(2k+1)/6$ (this formula can be easily checked by mathematical induction). Therefore, $\sigma_t^2 = 2\big(1/(2k+1)\big)\big(k(k+1)(2k+1)/6\big)\Delta t^2 =$

$\big(k(k+1)/3\big)\Delta t^2$. Substituting these expressions into the formulas from Theorem 11 completes the proof. □

Proof of Proposition 3. In this case, $\bar{t} = s_0$, and

$$\sigma_t^2 = \frac{1}{n}\sum_i (t_i - \bar{t})^2 = \frac{\Delta t^2}{n} 2 \sum_{i=1/2}^{n-1/2} i^2.$$

The values $i = 1/2,\ 3/2,\ 5/2, \ldots, (n-1)/2$ can be represented as $(2j-1)/2$ for $j = 1, 2, \ldots, k$. Here, $i^2 = 1/4(2j-1)^2$, hence

$$\sum_{i=1/2}^{n-1/2} i^2 = 1/4\left(\sum_{j=1}^{k}(2j-1)^2\right).$$

The sum $S = \sum_{j=1}^{k}(2j-1)^2$ of the square of all odd numbers from 1 to n can be represented as the difference between the sum of all numbers from 1 to n and all even numbers from 1 to n, i.e., as $S = \sum_{j=1}^{n} j^2 - \sum_{j=1}^{n/2}(2j)^2$. The first sum is equal to $n(n+1)(2n+1)/6$, the second sum $\sum_{j=1}^{n/2}(2j)^2$ is equal to 4 times the sum $\sum_{j=1}^{n/2} j^2$, i.e., to $4(n/2)(n/2+1)(2n/2+1)/6 = n(n+2)(n+1)/6$. Therefore, the difference between these two sums is equal to $n(n+1)(2n+1)/6 - n(n+2)(n+1)/6 = n(n+1)\big((2n+1)-(n+2)\big)/6 = n(n+1)(n-1)/6 = n(n^2-1)/6$. So, $\sigma_t^2 = (2/n)(1/4)\big(n(n^2-1)/6\big) = (n^2-1)/12$. Substituting these expressions into the formulas from Theorem 11 completes the proof. □

Proof of Theorem 18. Here, $d = \left(\sum_i x_i t_i - (\sum_i x_i)(\bar{t})\right)/(n\sigma_t^2)$ The value $n\sigma_t^2$ is computed once (before the computations), so we can assume that it is already stored in the computer as some value S. So here, we have only three expressions to update: $S_1 = \sum_i x_i t_i$, $S_2 = \sum_i x_i$, and $\bar{t}$.

The value $\bar{t}$ is (as one can easily see) equal to the average of the first and the last values: $\bar{t} = 1/2(t_n + t_{n+1-b}) = 1/2\big(t_1 + t_1 + \Delta t n + \Delta t(n+1-b)\big) = t_1 + \Delta t\big(n - (b-1)/2\big)$, so its update can be done according to a simple formula: $\bar{t} := \bar{t} + \Delta t$.

The update of S_1 and S_3 can be done by the same formulas as above, with the only difference that we cannot read t_n, we must compute it. This computation is also simple: $t_n := t_{n-1} + \Delta t$.

So, we arrive at the following algorithm: at every moment of time, we keep the values S_1, $S_2\bar{t}$, and the values x_i, t_i and $x_i t_i$ for all i such that $n - b < i \leq n$. Initially, S_1 and S_2 are set to 0, $\bar{t}$ to $t_1 - (b-1)/2\Delta t$, and t_1 to the given value.

Then, while $n < b$, we read x_n, compute $t_n := t_{n-1} + \Delta t$ and $x_n t_n$, update S_1 and S_2. $S_1 := S_1 + x_n t_n$ and $S_2 := S_2 + x_n$, and update $\bar{t} := \bar{t} + \Delta t$.

For $n = b$, in addition to all those steps, we compute the first estimate d as $(S_1 - S_2\bar{t})/S$.

For $n > b$, we read x_n, compute $t_n := t_{n-1} + \Delta t$ and $x_n t_n$, update $\bar{t}$: $\bar{t} := \bar{t} + \Delta t$, update S_1 and S_2 using more complicated formulas: $S_1 := S_1 + x_n t_n - x_{n-b} t_{n-b}$, $S_2 := S_2 + x_n - x_{n-b}$, and compute d as $(S_1 - S_2\bar{t})/S$.

For $n > b$, we need 10 computational steps; for $n \leq b$ we need less. So, our algorithm uses ≤ 10 computational steps par measurement. □

Proof of Theorem 19. Let us describe what can be parallelized in the algorithm from the previous proof: first, we can compute the new value of t_n, update $\bar{t}$ and start updating S_2 (1 step), then, on the 2nd step, we end updating S_2, start updating S_1. On the 3rd step, we the error is the smallest if and only if the sum $\sum_i a_i^2$ is the smallest possible. Applying Lagrange multipliers method to the resulting conditional optimization problem, we conclude that $a_i = \alpha + \beta(t_i - t_0) + \gamma(t_i - t_0)^2$. The linear equations for α, β, and γ are obtained, if we substitute the above expression for a_i into the conditions $\sum_i a_i = 0$, $\sum_i a_i(t_i - t_0) = 1$, and $\sum_i a_i(t_i - t_0)^2 = 0$. □

Proof of Theorem 21. According to Theorem 18, the optimal estimate d is equal to $\alpha\left(\sum_i x_i\right) + \beta\left(\sum_i x_i(t_i - t_0)\right) + \gamma\left(\sum_i (t_i - t_0)^2\right)$. So, if we already know how to compute α, β, and γ, we will be able to compute d in 5 steps (3 multiplications and 2 additions) if we update the three sums $\sum_i x_i$, $\sum_i x_i(t_i - t_0)$, and $\sum_i (t_i - t_0)^2$ (just like we did it in the proof of Theorem 16) The number of computational steps that is necessary to update a sum, does not depend on b.

To compute α, β, and γ, we need to solve a system of 3 linear equations with 3 unknowns. If we know the coefficients, then the number of computational steps that we need to solve this system, does not depend on b.

All the coefficients are of the type $\sum_i (t_i - t_0)^k$ for $k = 0, 1, 2, 3, 4$. The number of steps that we need to update these sums does not depend on b.

Adding all these numbers, we get the desired upper bound C for the number of computational steps that does not depend on b. □

Proof of Theorem 22. A linear estimate is uniquely determined by a sequence of coefficients $a_1, \ldots, a_n$. A precision p of a linear estimate is defined as $p = \max |\sum_i a_i x_i - b|$ for all a, b and x_i such that $|x_i - (a + bt_i)| \le \varepsilon$.

To simplify out proofs, let us introduce a new denotation: namely, let us denote $\Delta_i = x_i - (a + bt_i)$. Then, $|\Delta_i| \le \varepsilon$, $x_i = (a + bt_i) + \Delta_i$, and $d - b = \sum_i a_i(a + bt_i) + \sum_i a_i \Delta_i - b$. So, in these denotations, the definition of a precision p can be rewritten as follows:

$$p = \max_{a,b,|\Delta_i| \le \varepsilon} \left| \sum_i a_i(a + bt_i) + \sum_i a_i \Delta_i - b \right|$$

Let us first prove that if $\sum_i a_i \ne 0$, then $p = \infty$. Indeed, suppose that $\sum_i a_i \ne 0$. Let us fix some b and Δ_i such that $|\Delta_i| \le \varepsilon$, and consider the dependency of $s = \sum_i a_i(a + bt_i) + \sum_i a_i \Delta_i - b$ on a. Since $s = (\sum_i a_i)\, a + (\sum_i a_i t_i - 1)\, b + \sum_i a_i \Delta_i$, s is a linear function of a with a non-zero coefficient at 0. Therefore, $|s| \to \infty$ as $a \to \infty$ and hence, $p = \max s = \infty$.

So, if $p < \infty$, then $\sum_i a_i = 0$. Likewise, we can prove that if $p < \infty$, then $\sum a_i t_i = 1$. Indeed, suppose that $\sum a_i t_i \ne 1$. Then, we can fix a, Δ_i, and consider the dependency of s on b. Here, s is a linear function of b with a non-zero coefficient, so $|s| \to \infty$ as $b \to \infty$ and hence, $p = \max s = \infty$.

So, if there is an estimate with finite precision p, then for this estimate, $\sum_i a_i = 0$ and $\sum_i a_i t_i = 1$. Let us consider any such estimate. For it $s = \sum_i a_i \Delta_i$. So,

$$p = \max_{|\Delta_i| \le \varepsilon} \left| \sum_i a_i \Delta_i \right|.$$

Let us compute this maximum. Since Δ_i lies between $-\varepsilon$ and ε, the maximum possible value of $a_i \Delta_i$ is attained when $\Delta_i = \varepsilon$ if $a_i > 0$, and when $\Delta_i = -\varepsilon$ if $a_i < 0$. In both cases, this maximum value is equal to $|a_i|\varepsilon$. Therefore, the maximum possible value p of the sum is equal to $(\sum_i |a_i|)\,\varepsilon$.

In particular, if we take a_i from the formulation of the theorem (i.e., $a_n = -a_1 = 1/(t_n - t_1)$ and $a_2 = a_3 = \cdots = a_{n-1} = 0$), we can easily check that $\sum_i a_i = 0$ and $\sum_i a_i t_i = 1$. So, for this estimate, the precision equals $(\sum_i |a_i|)\,\varepsilon = \big(2/(t_n - t_1)\big)\varepsilon$. This value is finite, so, there always exists an estimate with finite precision.

Since ε is fixed, the precision $p = (\sum_i |a_i|)\,\varepsilon$ is the smallest possible if and only if the sum $\sum_i |a_i|$ attains the smallest possible value. So, the problem of finding an estimate with the smallest possible precision is equivalent to the problem of finding a sequence of real numbers $a_1, \ldots, a_n$ that satisfies the following conditional optimization problem: $\sum_i |a_i| \to \min$ under the condition that $\sum_i a_i = 0$ and $\sum_i a_i t_i = 1$.

Let us first prove that this problem has a solution. Indeed, we already know a sequence a_i that satisfies both conditions. For this sequence, $\sum_i |a_i| = 2/(t_n - t_1)$. Therefore, while looking for the most precise estimate, we can restrict ourselves by the sequences for which $\sum_i |a_i| \le 2/(t_n - t_1)$. For such sequences, for every i, $|a_i| \le \sum_i |a_i| \le 2/(t_n - t_1)$. The set of the sequences for which $\sum_i |a_i| \le 2/(t_n - t_1)$ is bounded, and it is also closed. Hence, it is compact. The conditions $\sum_i a_i = 0$ and $\sum_i a_i t_i = 1$ also determine closed sets. Therefore, the set S of all sequences, for which $\sum_i |a_i| \le 2/(t_n - t_1)$, $\sum_i a_i = 0$ and $\sum_i a_i t_i = 1$, is an intersection of a compact set and a closed set, and is therefore a compact. A function $\sum_i |a_i|$ is continuous on this compact set S, therefore, it attains its minimum in some point. So, there exists a sequence, for which the sum $\sum_i |a_i|$ takes the smallest possible value. As we have already mentioned, this means that this sequence has the smallest possible value of p.

Let us now find the sequence with the smallest possible value of p (or, what is equivalent, the smallest possible value of $\sum_i |a_i|$). Let a_i be such sequence. Let us prove that at most two of its elements are different from 0. We will prove it by showing that if three different elements of a_i are different from 0, then this sequence cannot be the one for which $\sum_i |a_i|$ is the smallest possible. Indeed, suppose that $a_i \ne 0$ for at least 3 different indices i. Since $\sum_i a_i = 0$, the values of a_i cannot all be of one sign. So, we can choose the three of them that are not all of one sign (i.e., either one is positive and two other are negative, or one is negative, and two other are positive). Let us denote these three values by j, and l. So, $a_j \ne 0$, $a_k \ne 0$, and $a_l \ne 0$ for some $j < k < l$.

Let us prove that by changing these three values we can diminish the value of $\sum_i |a_i|$, while still retaining the conditions $\sum_i a_i = 0$ and $\sum_i a_i t_i = 1$. We will try to take a sequence a'_i that is defined by the formulas $a'_j = a_j + \alpha b_j$, $a'_k = a_k + \alpha b_k$, and $a'_l = a_l + \alpha b_l$ for some real numbers $\alpha \ne 0$, b_j, b_k, b_l, and $a'_i = a_i$ for all i that are different from j, k, or l.

We want to guarantee that $\sum_i a'_i = 0$ and $\sum_i a'_i t_i = 1$. We assumed that the sequence a_i satisfies these conditions. So, if we substitute the above expressions for a'_i into these formulas, and use the conditions that $\sum_i a_i = 0$ and $\sum_i a_i t_i = 1$, we can conclude that a'_i satisfies these conditions if and only if $b_j + b_k + b_l = 0$ and $b_j t_j + b_k t_k + b_l t_l = 0$. The first equation is satisfied if $b_l = -(b_j + b_k)$. Substituting this value into the second equation, we obtain the equation $b_j(t_j - t_l) + b_k(t_k - t_l) = 0$. Therefore, the second condition is satisfied, if we take an arbitrary b_j, and $b_k = -b_j(t_l - t_j)/(t_l - t_k)$. From these two expression, we conclude that $b_l = -(b_j + b_k) = b_j(t_k - t_j)/(t_l - t_k)$.

So, if we take an arbitrary b_j, $b_k = -b_j(t_l - t_j)/(t_l - t_k)$, and $b_l = b_j(t_k - t_j)/(t_l - t_k)$, then the resulting values a'_i satisfy both conditions $\sum_i a'_i = 0$ and $\sum_i a'_i t_i = 1$.

Let us now estimate $\sum_i |a'_i|$ for these a'_i. Since $a_j + \alpha b_j \to a_j$ as $\alpha \to 0$, we can conclude that for sufficiently small α, the expression $a_j + \alpha b_j$ has the same sign as a_j. Therefore, for such α, $|a_j + \alpha b_j| = \text{sign}(a_j + \alpha b_j)(a_j + \alpha b_j) = \text{sign}(a_j)(a_j + \alpha b_j) = |a_j| + \text{sign}(a_j)\alpha b_j$. Similar expressions are true for k and l, therefore, for sufficiently small α, $\sum_i |a'_i| = \sum_i |a_i| + \alpha\Sigma$, where $\Sigma = \text{sign}(a_j)b_j + \text{sign}(a_k)b_k + \text{sign}(a_l)b_l$. Let us prove that $\Sigma \neq 0$. Indeed, we chose j, k and l in such a way that two of the values a_j, a_k, a_l are of one sign, and the third value is of different sign. Suppose, for example that $a_j > 0$, $a_k > 0$ and $a_l < 0$. Then, $\Sigma = b_j + b_k - b_l$. We have already proved that $b_j + b_k + b_l = 0$, therefore, $\Sigma = (b_j + b_k + b_l) - 2b_l = -2b_l$. According to the above expression for b_l, it is $\neq 0$ if $b_j \neq 0$. So, in this cases, $\Sigma \neq 0$. Likewise, we can consider all other combinations of signs and prove that in all cases, $\Sigma \neq 0$.

Since $\Sigma \neq 0$, we can take α of the opposite sign with Σ, and find a sequence a'_i, for which $\sum_i |a'_i| < \sum_i |a_i|$, and thus, a contradiction to our assumption that $\sum_i |a_i|$ attained its minimum in the given sequence a_i. This contradiction proves that the assumption that a_i is non-zero for at least 3 different i, is false.

So, a_i is different from 0 for at most 2 different values of i. It cannot be different from 0 only for one i, because from $\sum_i a_i = 0$ we would then conclude that $a_i \equiv 0$, and hence $\sum_i a_i t_i = 0 \neq 1$. Therefore, for optimal sequence a_i, a_i is different from 0 precisely for 2 different values $j < k$.

In this case, the conditions $\sum_i a_i = 0$ and $\sum_i a_i t_i = 1$ turn into $a_j + a_k = 0$ (hence $a_j = -a_k$) and $a_j t_j + a_k t_k = 1$. Substituting $a_j = -a_k$ into the second equation, we conclude that $a_k = 1/(t_k - t_j)$ and $a_j = -1/(t_k - t_j)$. For this sequence, $\sum_i |a_i| = 2/(t_k - t_j)$.

So, in order to find a sequence a_i with the smallest possible precision p, we must find a pair $j < k$, for which the expression $2/(t_k - t_j)$ takes the smallest possible value. This expression is the smallest possible if and only if the difference $t_k - t_j$ is the biggest possible. This is attained if t_k is the biggest possible (i.e., $k = n$), and t_j is the smallest possible (i.e., $j = 1$). Therefore, the sequence a_i for which p is the smallest coincides with the one given in the formulation of the theorem. □

Acknowledgements

This work was supported by a NSF Grant No. CDA–9015006, NASA Research Grant No. 9–482, and a grant from GSA, administered by Materials Research Institute and the Institute for Manufacturing and Materials Management. The authors are thankful to Misha Koshelev who tested our sequential algorithms by programming them in Pascal, to Sergey Shary and Vyacheslav Nesterov for valuable discussions and important preprints, and to the anonymous referees for their thoroughness and help.

References

[1] Aberth, O. *Precise numerical analysis.* Wm. C. Brown Publ., Dubuque, Iowa, 1988.

[2] Bellman, R. *Introduction to matrix analysis.* McGraw Hill, N.Y., 1970.

[3] Bohlender, G., Ulrich, C., Wolff von Gudenberg, J., and Rall, L. B. *Pascal–SC. A computer language for scientific computations.* Academic Press, N.Y., 1987.

[4] Burden, R. L. and Faires, J. D. *Numerical analysis.* Prindle, Weber & Schmidt, Boston, MA, 1985.

[5] Chung, L. L., Lin, R. C., Soong, T. T., and Reinhorn, A. M. *Experimental study of active control for MDOF seismic structures.* Journal of Engineering Mechanics **115** (8) (1989), pp. 1609–1627.

[6] Corliss, G. F. *Applications of differentiation arithmetic.* In: Moore, R. E. (ed.) "Reliability in computing", Academic Press, N.Y., 1988, pp. 127–148.

[7] Dehghanyar, T. J., Masri, S. F., Miller, R. K., and Caughey, T. K. *On-line parameter control of nonlinear flexible structures.* In: "Proceedings of the 2nd International Symposium on Structural Control, Martinus-Nijhoff, Boston, MA, 1987", pp. 141–159.

[8] Jaja, J. *An introduction to parallel algorithms.* Addison-Wesley, Reading, MA, 1992.

[9] Karmarkar, N. *A new polynomial-time algorithm for linear programming.* Combinatorica **4** (1984), pp. 373–396.

[10] Kobori, T. *State-of-the-art of structural control research in Japan.* In: "Proceedings of the US National Workshop on Structural Control, October 25–26, 1990", pp. p1–p21.

[11] Kobori, T. et al. *U.S. Patent No. 5,036,633.*

[12] Kobori, T., Kanayama, H., and Kamagata, S. *A proposal of new anti-seismic structure with active seismic response control system - dynamic intelligent building.* In: "Proceedings of the 9th World Conference on Earthquake Engineering, Tokyo—Kyoto, Japan, **VIII**", pp. 465–470.

[13] Kobori, T., Kanayama, H., and Kamagata, S. *Active seismic response control systems for nuclear power plant equipment facilities.* Nuclear Engineering and Design **111** (1989), pp. 351–356.

[14] Kobori, T., Yamada, S., and Kamagata, S. *U.S. Patent No. 4,922,667.*

[15] Kreinovich, V., Lakeyev, A., and Noskov, S. *Optimal solution of interval linear systems is intractable (NP-hard).* Interval Computations (1) (1993).

[16] Markov, S. *Interval differential equations.* In: Nickel, K. E. (ed.) "Interval Mathematics 1980", Academic Press, N.Y., 1980, pp. 145–164.

[17] Meirovitch, L., Baruch, H., and Oz, H. *A comparison of control techniques for large flexible structures.* Journal of Guidance **6** (4) (1983), pp. 302–310.

[18] Moore, R. E. *Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solution of ordinary differential equations.* In: Rall, L. B. (ed.) "Errors in Digital Computations, Proceedings of a Symposium", John Wiley & Sons, N.Y., 1965, pp. 103–140.

[19] Moore, R. E. *Interval analysis.* Prentice Hall, Englewood Cliffs, N.J., 1966.

[20] Moore, R. E. *Mathematical elements of scientific computing.* Holt, Rinehart and Winston, N.Y., 1975.

[21] Moore, R. E. *Methods and applications of interval analysis.* SIAM, Philadelphia, 1979.

[22] Nemir, D. C., Koivo, A. J., and Kashyap, R. L. *Pseudolinks and the self-tuning control of a non-rigid link mechanism.* IEEE Transactions on Systems, Man and Cybernetics **18** (1) (1988), pp. 40–48.

[23] Osegueda, R. A., Nemir, D. C., and Lin, Y. J. *On-line adaptive stiffness control to tailor modal energy contents in structures.* In: "ADPA/AIAA/ASME/SPIE Conference on Active Materials and Adaptive Structures, Alexandria, VA, November 5–8, 1991".

[24] Rall, L. B. *Applications of software for automatic differentiation in numerical computation.* In: Alefeld, G. and Grigorieff, R. D. (eds) "Fundamentals of Numerical Computations (Computer-Oriented Numerical Analysis)", Springer-Verlag, Wien, N.Y., 1980, pp. 141–156.

[25] Rall, L. B. *Automatic differentiation: techniques and applications.* Lecture Notes in Computer Science **120**, Springer-Verlag, Berlin-Heidelberg-N.Y., 1981.

[26] Rall, L. B. *Differentiation and generation of Taylor coefficients in Pascal–SC.* In: Kulisch, U. W. and Miranker, W. L. (eds) "A New Approach to Scientific Computation", Academic Press, N.Y., 1983, pp. 291–309.

[27] Ratschek, H. and Schröder, G. *Über die Ableitung von intervallwertigen Funktionen.* Computing **7** (1971), pp. 172–187.

[28] Sendov, B. *Some topics of segment analysis.* In: Nickel, K. E. (ed.) "Interval Mathematics 1980", Academic Press, N.Y., 1980, pp. 203–222.

[29] Shary, S. P. *Solution of "outer" and "inner" problems for an interval system of linear algebraic equations.* Ph.D. Dissertation, Kransoyarsk—Ekaterinburg, 1991 (in Russian).

[30] Shary, S. P. *A new class of algorithms for optimal solution of interval linear systems.* Interval Computations (2) (1992), pp. 18–29.

[31] Soong, T. T. *Active structural control: theory and practice.* Longman Scientific, Essex, 1990.

Received: May 25, 1992
Revised version: January 12, 1994

Electrical Engineering Department and
Computer Science Department
University of Texas at El Paso
El Paso, TX 79968
USA