Table 8.1: Enclosures of determinants from Example 8.20. Bounds of the enclosures are rounded off to 3 decimal digits. Fixed radii of intervals are denoted by r.

method	$r = 0.2$	$r = 0.1$	$r = 0.01$
`hull`	[-0.6, 21.72]	[4.06, 14.88]	[8.465, 9.545]
`ge`	[-29.25, 87.75]	[3.000, 21.857]	[8.275, 9.789]
`ge+inv`	$[-\infty, \infty]$	[3.6, 18]	[8.46, 9.56]
`ge+lu`	[-99.45, 134.55]	[1.44, 22.482]	[8.244, 9.791]
`cram`	$[-\infty, \infty]$	[3.01, 23.143]	[8.326, 9.722]
`cram+inv`	$[-\infty, \infty]$	[3.6, 17.067]	[8.46, 9.56]
`cram+lu`	$[-\infty, \infty]$	[1.44, 21.434]	[8.244, 9.79]
`had`	[-564.788, 564.788]	[-526.712, 526.712]	[-493.855, 493.855]
`had+inv`	[-30.048, 30.048]	[-16.801, 16.801]	[-9.563, 9.563]
`had+lu`	[-46.178, 46.178]	[-35.052, 35.052]	[-27.019, 27.019]
`gersch`	[-3371.016, 11543.176]	[-3132.927, 11089.567]	[-2926.485, 10691.619]
`gersch+inv`	[-81, 243]	[0, 72]	[6.561, 11.979]
`gersch+lu`	[-11543.176, 6435.576]	[-11089.567, 6116.667]	[-10691.619, 5838.41]

Example 8.20. To obtain a general idea how the methods work, we can use the following example. Let us take the midpoint matrix

$$A_c = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 6 & 7 \\ 5 & 9 & 8 \end{pmatrix},$$

and inflate it into an interval matrix using three fixed radii of intervals $0.2, 0.1$ and 0.01 respectively and test all the mentioned methods. The resulting enclosures of the determinants are shown in Table 8.1.

The previous example shows a case where the `lu` preconditioning gives better results for `ge` than the `inv` preconditioning. However when testing for larger matrices the determinant enclosure using the `lu` preconditioning tends to be infinite too. From the above example we see that for a general matrix preconditioning is favorable. That is why we later test only `ge+inv`, `cram+inv`, `had+inv` and `gersch+inv` methods.

We can perceive the method `ge+inv` used in [208] as the “state-of-the-art” method. Therefore, every other method will be compared to it.

All methods are tested on randomly generated matrices of sizes $n = 10, \dots, 60$. To generate an interval matrix a real midpoint matrix is randomly generated with coefficients selected independently and uniformly from $[-1, 1]$. Then, such a matrix is inflated into an interval matrix by wrapping the coefficients with intervals of a given fixed radius. Here we choose the radii ($r = 10^{-3}$ and $r = 10^{-5}$). For each size and radius we test on 100 matrices.

For each radius, size and method an average ratio of computed enclosures and average computation time are computed. We compute the ratios according to the

Table 8.2: Number of infinite enclosures returned by various method (out of 100) for fixed radii $r = 10^{-5}$ and $r = 10^{-3}$ respectively. The sizes of matrices are denoted by n.

n	`cram+inv`	`ge+inv`	`ge+inv`	`cram+inv`
10	0	0	4	2
20	0	0	15	15
30	0	0	10	10
40	0	0	34	31
50	0	0	38	38
60	2	2	54	51
r	10^{-5}		10^{-3}	

formula (3.8). If the average ratio is < 1 it means a methods returned narrower results than `ge+inv`. It can happen that an enclosure returned by a method is infinite. Such case is omitted from the computation of the average. The occurrence of such a phenomenon is captured in Table 8.2. We can see that for smaller radii it happens only rarely. The methods `had+inv` and `gersch+inv` never returned an infinite enclosure.

Average ratios of widths are presented in Table 8.3. When the ratio is a number less then 1000, it is displayed rounded off to 2 decimal digits. When it is greater, only the approximation 10^x is displayed. To accentuate the similarity of the results returned by `ge+inv` and `cram+inv`, their ratio of enclosures is rounded off to 6 decimal digits. With increasing size of a system (and also with increasing overestimation of `ge+inv` and `cram+inv`) the ratio difference of `had+inv` becomes less apparent.

Average computation times for $r = 10^{-5}$ are displayed in Table 8.4. Since the methods are basically direct (except for verified inverse computation in HBR method), the computation times for $r = 10^{-3}$ are very similar. The method `cram+inv` is significantly faster than `ge+inv`. To more clearly see the difference in computation times between the two most efficient methods `ge+inv` and `cram+inv` see Figure 8.6.

8.7.2 Symmetric matrices

We repeat the same test procedure for symmetric interval matrices. Symmetric matrices are generated in a similar way as before, only they are shaped to be symmetric (the lower triangle of a matrix is mirrored to the upper triangle). We again compare the `ge+inv`, `gersch+inv`, `had+inv` and `cram+inv`. We add one new method `eig` that is based on computation of enclosures of eigenvalues using the Rohn's simple enclosure (Proposition 8.14). The method `ge+inv` stays the reference method, i.e, we compare all methods with respect to this method.

The ratios of enclosures widths for symmetric matrices are displayed in Table 8.5 and Table 8.6. We can see that as in the general case the results of `cramer+inv` are very similar to `ge+inv`. When $r = 10^{-3}$ the overestimation by `had+inv` becomes smaller than on `eig` at a certain point ($n = 40$).

Table 8.3: Average ratios of widths of enclosures returned by various methods for interval matrices with fixed radii 10^{-5} and 10^{-3} respectively. The methods are compared to `ge+inv`, the sizes of matrices are denoted by n.

n	`gersch+inv`	`had+inv`	`cram+inv`	`gersch+inv`	`had+inv`	`cram+inv`
10	35.34	10^3	1.000100	10^3	14.58	0.999974
20	50.16	10^3	1.000000	10^9	6.18	1.000911
30	10^9	257.45	1.000010	10^{18}	3.78	1.006991
40	10^4	178.02	0.999999	10^{24}	2.52	0.998934
50	10^{25}	117.85	1.000049	10^{29}	2.06	1.001731
60	10^{24}	101.19	0.999980	10^{40}	1.46	1.002089
r		10^{-5}			10^{-3}	

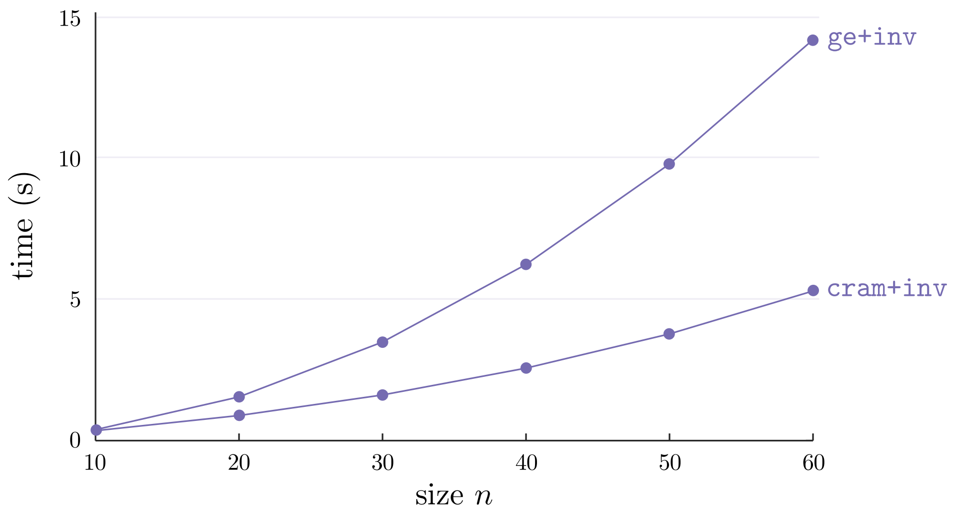

Figure 8.6: Visual comparison of average computation times (in seconds) of `ge+inv` and `cram+inv` for various matrix sizes n.

Table 8.4: Average computation times (in seconds) of various methods for fixed radii 10^{-5} and various sizes of matrices n.

n	`gersch+inv`	`had+inv`	`ge+inv`	`cram+inv`
10	0.04	0.01	0.39	0.36
20	0.06	0.01	1.58	0.90
30	0.09	0.02	3.56	1.64
40	0.12	0.02	6.34	2.59
50	0.15	0.04	9.95	3.80
60	0.19	0.05	14.37	5.32

Table 8.5: Average ratios of widths of enclosures returned by various methods for symmetric matrices with fixed radii $r = 10^{-5}$. The reference method is `ge+inv`, the sizes of matrices are denoted by n.

n	`gersch+inv`	`had+inv`	`cram+inv`	`eig`
10	18.50	10^3	1.000000	2.62
20	50.51	10^3	0.999999	3.07
30	108.66	10^3	1.000000	3.23
40	126.79	250.03	1.000000	3.59
50	10^9	166.60	1.000001	3.63
60	10^{11}	117.62	0.999999	3.63

Table 8.6: Average ratios of widths of enclosures returned by various methods for symmetric matrices with fixed radii $r = 10^{-3}$. The reference method is `ge+inv`, the sizes of matrices are denoted by n.

n	`gersch+inv`	`had+inv`	`cram+inv`	`eig`
10	242.93	19.48	1.000637	2.49
20	10^9	7.69	0.999870	2.88
30	10^{15}	4.16	0.998364	2.96
40	10^{22}	3.31	0.995946	3.56
50	10^{26}	2.56	1.000100	4.18
60	10^{33}	2.04	1.002171	4.99

Table 8.7: Average computation times (in seconds) of various methods for symmetric matrices with fixed radii 10^{-5}. The sizes of matrices are denoted by n.

n	`gersch+inv`	`had+inv`	`ge+inv`	`cram+inv`	`eig`
10	0.04	0.01	0.39	0.35	0.02
20	0.06	0.01	1.56	0.89	0.03
30	0.09	0.02	3.51	1.62	0.04
40	0.12	0.03	6.26	2.56	0.07
50	0.15	0.04	9.82	3.77	0.10
60	0.19	0.05	14.20	5.29	0.15

The average computation times are displayed in Table 8.7. We can see that `eig` shows slightly higher computational demands than `had`. In case of $r = 10^{-3}$ and $n \geq 40$ it pays off to use rather `had+inv` than `eig`. However, for $r = 10^{-5}$ "reasonable" overestimation in a fraction of `cram+inv` computation time is obtained. The method `eig` was based on a simple enclosure. That explains the low computational time. Of course, it is possible to use, e.g., filtering methods to obtain even tighter enclosures of eigenvalues. However, they work well in specific cases [79] and the filtering is much more time consuming.

8.7.3 Summary

It is always the question of the payoff between computation speed and quality of enclosure. Based on the tests from the previous subsections, we recommend to use `cram+inv` method, since it produces equivalent results to `ge+inv` in much less computational time.

9 Application of intervals to medical data

- ▶ Multiple breath washout test
- ▶ Finding breath ends
- ▶ Regression on interval data
- ▶ Interval regression with integer data matrix
- ▶ Application to medical data
- ▶ Hypotheses

This chapter is based on results from a joint research project of Department of Applied mathematics, Faculty of Mathematics and Physics and Department of Paediatrics, 2nd Faculty of Medicine at Charles University, Prague, especially, on collaboration with Václav Koucký. The author of this work was the head researcher of this project. First, we introduce the medical background of the project. Then, we discuss interval regressions and how to improve them for the sake of our problem. The conclusions from this project are still in a form of hypotheses that need to be further verified or rejected. However, this work might contribute to the ongoing discussions related to these topics. Some of our initial (rather too optimistic) ideas were published in [88]. More detailed results are contained in our unpublished work [90]. The algorithm for finding breath ends is published in [89].

9.1 Multiple Breath Washout test

First works concerning multiple breath washout test (MBW) date back to 40s or 50s [28]. In those days the method faced crucial limits. The precision of sensors was not satisfactory and also the computational power of digital computers was insufficient to handle problems described with too many parameters (much of mathematical work was still done manually). With increasing power of sensors and computers MBW received its rebirth in 90s.

MBW is a very promising method since it does not require any specific breathing maneuvers. The only necessity is the ability to breathe normally with regular pattern, which makes it applicable to the variety of age scale. Small infants usually undergo this procedure in artificial sleep.

In contrast to classical methods (e.g., spirometry, bodypletysmography) MBW is

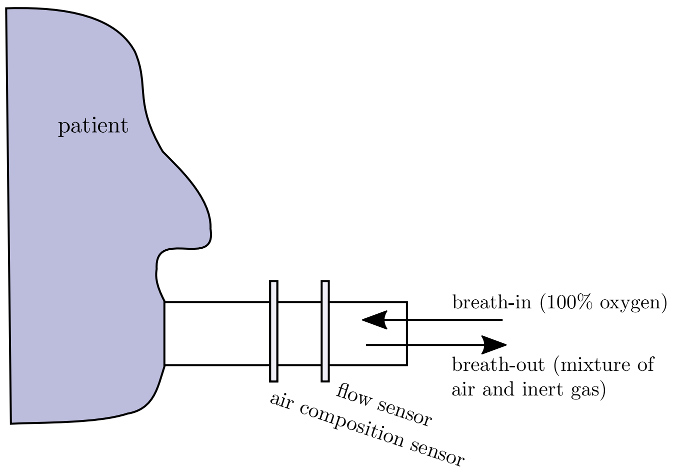

Figure 9.1: Schematic depiction of the washout phase.

able to evaluate even the most peripheral airway. The high sensitivity to the most peripheral airway changes has been shown in most of chronic lung diseases (e.g., bronchial asthma, cystic fibrosis, primary cilliary dyskinesia, etc.) [33, 56, 121].

The test consists of two phases – the washin and washout. During the first phase, lung is filled with an inert gas (sulphur hexafluoride SF_6, helium He or nitrogen N_2), during the second phase, the inert gas is washed out by air or by 100% oxygen (depending on the inert gas used). Concentration of the respective inert gas, volume of exhaled gas and flow are measured online. The measurement is stopped after reaching a certain concentration of innert gas within lung (usually 2.5%). The pattern of inert gas concentration decrease gives information about the homogeneity of ventilation and thus about the patency of airways. The washout phase is depicted in Figure 9.1.

In our work we focus on use of nitrogen (N_2) as inert gas. Although, the SF_6 has been historically used for much longer in practice, use of nitrogen has many advantageous properties:

- SF_6 can potentially have narcotic effects,
- SF_6 is not used in medicine, so it must be specially prefabricated, N_2 is naturally present in the surrounding air,
- N_2 is naturally present in the surrounding air and also in lung, that is why there is no need for washin phase,
- N_2 is present also in poorly ventilated areas of lung.

A small draw back is that there are questions whether N_2 is really an inert gas because of its absorbance and ocurence in tissues. During a measurement, N_2 returned back from tissues can influence the real measured concentrations of N_2, especially for infants. That is why the method is recommended for patients older than 6 months.

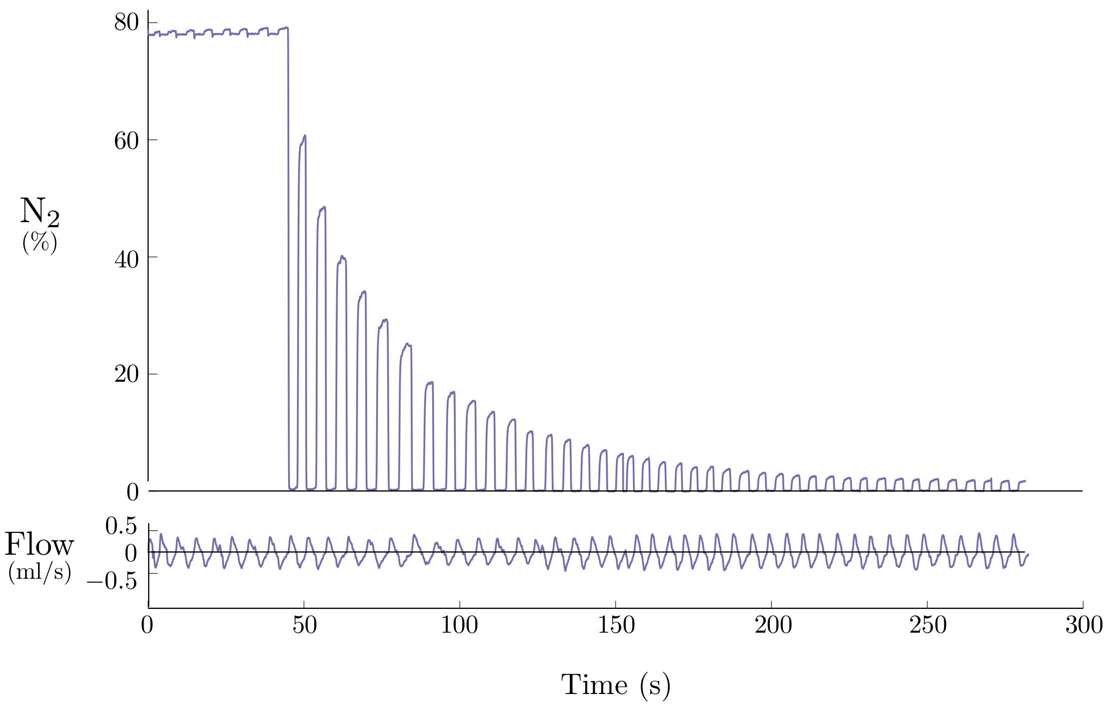

Figure 9.2: Nitrogen concentration (the top curve) and air flow (the bottom curve) in time measured during the nitrogen washout process.

The main output of the measurement is depicted in Figure 9.5. There are two main graphs – actual flow (the bottom curve) and decreasing nitrogen concentration (the top curve) measured in each *time-slice* (here it is 5ms). These data are further used for computing clinically significant indices (FRC, LCI, Scond, Sacin, etc.). Some of them will be mentioned later. The sensibility of MBW can detect a pathology in its early stages, which enables to start the cure early and with greater effect.

9.2 LCI and FRC

Currently, the most important indices calculated from MBW data are FRC and LCI. If we omit the deadspace correction, the *functional residual capacity* (FRC) is calculated as

$$\mathrm{FRC} = \frac{\mathrm{N}_{out}}{\mathrm{N}_{start} - \mathrm{N}_{end}},$$

where N_{out} is the total volume of expired N_2; N_{start} and N_{end} are concentrations of nitrogen at initial and terminal peak respectively. The FRC relates to the size of lung. The *lung clearance index* (LCI) is calculated simply as

$$\mathrm{LCI} = \frac{\mathrm{V}_{out}}{\mathrm{FRC}},$$

where V_{out} is the total volume of expired air. It is necessary to specify how we decide the terminal breath. Usually, a measurement is stopped when the concentration of nitrogen in peaks decreases below 2.5% of the initial nitrogen concentration. This level is chosen historically. The terminal peak is defined to be the first of three consequent peaks with concentration below 2.5% of the initial nitrogen concentration. The corresponding LCI index is then marked LCI2.5. It states how many air volumes (equal to FCR) exchanges are necessary to clean the lung from the inert gas (more specifically to reach the level of 2.5% of initial inert gas concentration). LCI index seems to be very useful to evaluate the homogeneity of lung ventilation (the most peripheral airways included).

Completing the washout process up to 2.5% might be too time consuming, which makes it difficult for uncooperative patients to finish the MBW test properly. That is why there are being discussions about use of the level 5%.

9.3 Our data

We collected the data according to proper conditions for valid measurement defined in [101]. The three necessary conditions are:

- a patient has sufficiently regular breathing pattern during measurement,
- there is no leakage during the measurement,
- washout phase is finished (nitrogen is washed out at least to a given level).

The measurements were approved by the ethical committee of Motol University Hospital, Prague, Czech republic. Patients (or their parents in case of infants) were informed about the measurement before the tests.

In all data files the peaks of the nitrogen concentrations have been identified using our own algorithm described in the next section.

9.4 Finding breath ends

Breath detection (i.e., finding the spot where an expiration ends and an inspiration starts) is a crucial step in pulmonary function testing (PFT). It is a starting point for computing various clinically significant indices, performing regression analyses or making predictions. With the increasing importance of PFT as a diagnostic tool, new methods of PFT and approaches to data analysis are required especially in infants and toddlers (i.e., uncooperative children). In this age category, precise raw data analysis is of utmost importance, as the PFT is very prone to technical errors. Based on our clinical experience, the current PFT algorithms suffer from severe imprecisions, which may lead to difficult and time-consuming interpretation of results or even raw data rejection.

Although breath detection is a relatively easy task for a physician as a human being, automated detection by computer remains a challenge, especially in cases of severely distorted data (e.g., as a result of patients' insufficient cooperation, severe volume drift, etc.). An approach to the breath detection analysis is primarily determined by the signals being measured. Usually, a time-flow signal is captured. In this situation, two basic algorithms for breath detection have been proposed – threshold and smoothing approach, each with numerous modifications and extensions increasing their reliability and accuracy [13]. The threshold approach outputs any breath having parameters above a pre-set threshold. On the other hand, the smoothing approach smooths the signal to eliminate spurious breath endings. Despite the significant progress done in this field, clinicians are still facing situations in which the measured signal is too distorted to be automatically analyzed.

In comparison with the "conventional" methods that are based solely on flow, volume and pressure measurements and estimated primarily airway resistance (e.g., bodypletysmography, tidal breath analysis, etc.), MBW brings a new dimension to raw data – the gas concentration signal (O_2, CO_2 , inert gas). A current commercial software (Spiroware, Ecomedics, Duernten, Switzerland) uses concentrations only for constructing washout curves. However, this information may be also used for breath detection. The aim of our study [89] was to design and justify a new and robust algorithm for breath detection using not only time-flow data but also gas concentration signal. Such a breath detection algorithm can significantly outperform the current threshold-based algorithms. Moreover, its key ideas have the potential to contribute to the general design of the medical algorithms.

9.4.1 Our algorithm

Our algorithm (`Alg-OUR`) was programmed in the free software GNU Octave, version 4.0.0. and works in several steps, which are outlined below. A depiction of each step can be found in Figure 9.3.

Algorithm 9.1 (Breath end detection (`Alg-OUR`))**.** The input is raw flow and CO_2 concentration data in time. The algorithm outputs integer intervals containing numbers of zero-crossings corresponding to one breath end.

1. Load raw data.

2. Zero-crossings detection – a zero-crossing is defined as a time spot, where the air flow changes its direction from minus to plus (see a comment on general physiology of respiratory tract in Subsection 9.4.4). All the zero-crossings in flow raw data are detected and numbered from 1 to N, where N is the total number of zero-crossings. They form a set of potential breath ends.

3. For each $-/+$ zero-crossing at time T, the nearest peak of CO_2 curve (i.e., local maximum) is found and attributed to the time T.

4. The volume of each inhalation and exhalation (V_{in}, V_{out}) corresponding to the time T is calculated by integration of the flow curve (using simple trapezoidal rule, similarly as in Example 2.4).

5. The zero-crossings with corresponding CO_2 peaks of insufficient concentration (i.e., less than 2% – see a comment in Subsection 9.4.4) are discarded; the numbering of zero-crossings is preserved. Next, our goal is to discard zero-crossings

that do not form a breath end or capture intervals of zero-crossings that belong to the same breath. Initially, we view each zero-crossing to be a singleton interval ($[b, b]$). Next, the algorithm is going to discard or merge some of these intervals (steps 6 to 8).

6. Two intervals of zero-crossings $[a, b]$ and $[c, d]$ are merged if the CO_2 concentration between b and c does not drop below 0.5%. Consequently, a new interval $[a, d]$ instead of the previous two is created. This process is repeated until there exists no such a pair of intervals. Note that in an interval $[a, b]$, a can be equal to b.

7. The two consecutive intervals of zero-crossings $[a, b]$ and $[c, d]$ where $c = b + 1$ are merged if the ratio of volumes V_{in}/V_{out} for zero-crossing b is greater than 5 (see the comment in section Discussion). This process is repeated until there exists no such a pair of intervals.

8. The upper bounds of the remaining intervals (even tight ones - $[a, a]$) are marked as the breath ends (i.e., from $[a, b]$, it is b, from $[a, a]$, it is a).

(!) Note that the order of the steps 5, 6 and 7 cannot be changed; otherwise the algorithm produces incorrect results.

For the sake of comparison, the most commonly used flow threshold algorithm (originally described in [216]) were implemented in our software. Two different thresholds (`Alg3-0.01` and `Alg3-0.25`) according to the age of the patient and an additional plausibility check were used as specified in [13], [198] and [216].

9.4.2 Test data characteristics

To test the clinical usefulness and accuracy of our newly developed algorithm, we compared it with representatives of the currently used algorithms on real patient data. We intentionally selected severely distorted measurements, which are, in our experience, very difficult to be automatically analyzed by the current software. In total, 47 anonymized traces (A-files) coming from 19 patients were enrolled. Such an approach was in general approved by the local ethics committee. Patients' characteristics are stated in Table 9.1. The rationale for intentional selection of severely distorted data was the fact, that only severely distorted data offer the possibility to test the performance of breath detection algorithms properly. Analysis of regular breathing is no challenge for current breath detection approaches anymore.

9.4.3 Comparison of algorithms

The raw data were analysed in four different ways:

1. analysis performed by our algorithm described above (`Alg-OUR`),

2. analysis performed by the previously described algorithms (`Alg3-0,01` and `Alg3-0,25`) that are implemented in our software,

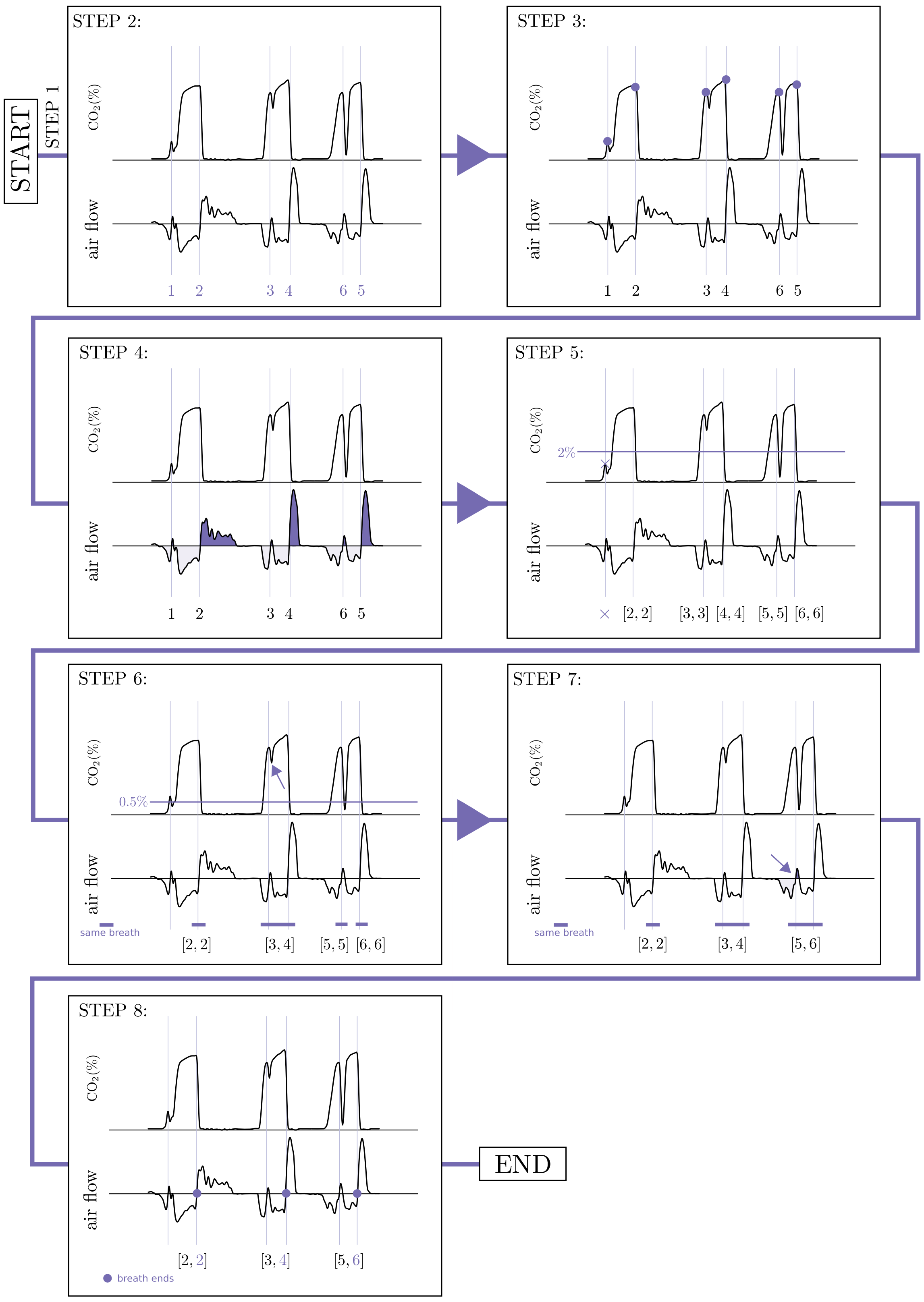

Figure 9.3: Flow diagram and depiction of each step of our breath detection algorithm (`Alg-OUR`).

Table 9.1: Characterization of the patients. Their A-files were used for the sake of the comparison of breath detection algorithms.

General information	Number of patients (male)	19(9)
	Number of A-files	47
	Age (mean ± SD) [years]	6.6 ± 5.6
	Weight (mean ± SD) [kg]	26.6 ± 19.0
	Weight z-score (mean ± SD)	0.02 ± 1.2
	Height (mean ± SD) [cm]	114.6 ± 33.2
	Height z-score (mean ± SD)	-0.4 ± 1.3
Diagnoses	nonrespiratory problematic	4
	cystic fibrosis	7
	primary ciliary dyskinesia	2
	repeated obstructive bronchitis	4
	miscellaneous	2

3. analysis performed by the commercial package Spiroware 3.2.0 (`Alg-Spi`),
4. manual analysis performed by two specialists experienced in PFT.

After loading an A-file into our software, the number of breaths detected by `Alg-OUR`, `Alg3-0,01` and `Alg3-0,25` were calculated. The A-file was also loaded into Spiroware and the number of breaths was estimated using the functionality of this commercial software. Afterwards, two PFT specialists inspected the data from each A-file independently. The inspection was done in the interface of our software, created for this purpose. It enables visualization of flow, volume and CO_2 concentration, while at the same time visualisation of breath ends found by the respective algorithms. Such visualization enables both the estimation of the number of true breaths (reference number of breaths – RNB) and simultaneously the localization of falsely positive/negative breaths as analyzed by different algorithms.

All the A-files included in our testing could be successfully analysed by all the implemented algorithms. The analysis time was longer for `Alg-OUR` than for the threshold algorithms (1.35 ± 0.23s vs. 0.12 ± 0.01s, $p < 0.001$). The manual analysis took much longer; the average analysis time was roughly estimated to be between 100 and 180s.

The two specialists in PFT working independently detected the same number of breaths in 35 out of 47 A-files (74%). In the remaining cases, differences were not larger than two breaths. These cases were reanalyzed by the two specialists jointly in order to reach consensus and the "reference number of breaths" (RNB) was assigned to each A-file. Finally, 2861 true breaths in 47 A files were included.

The agreement between the algorithm `Alg-OUR` and RNB was in 70.2% files (33 out of 47 A-files), the maximal difference between the result of `Alg-OUR` and RNB was 7 breaths. The falsely positive breaths (i.e., zero-crossings misinterpreted as breath

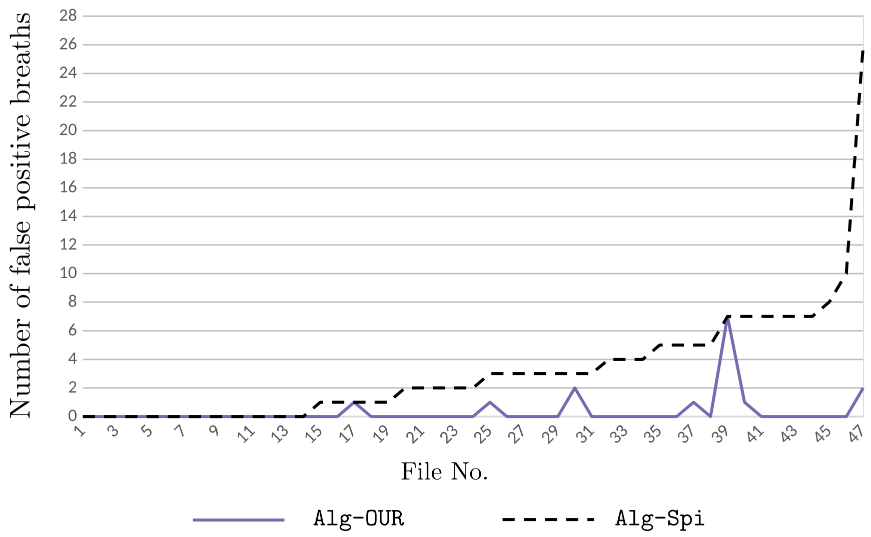

Figure 9.4: Number of false positive breaths detected in each A-file by **Alg-OUR** and **Alg-Spi**. The dashed line displays number of false positive breaths for **Alg-Spi**, the solid line displays the false positive breaths for **Alg-OUR**. The A-files are ordered (numbered) according to the increasing amount of false positive breaths detected by **Alg-Spi**.

ends) were the most prominent issue of this automated breath end detection. On the other hand, its sensitivity was high enough not to miss any true breath (number of falsely negative breath ends was equal to zero in the whole analysis). All other algorithms were clearly less effective. The agreement between Alg-Spi and RNB was only 17.0% files (8 A-files); the maximum difference in breaths detected was 26 breaths, no falsely negative breath was detected. Alg3-0,01 suffered severely from the false positive breath detection, even in the youngest age category (toddlers under 3 years). Agreement between Alg3-0,01 and RNB was reached only in 4 cases (8.5% files), no false negative breath was detected. On the other hand, Alg3-0,25 showed tendency to miss true breaths (so called false negative breaths), especially in the youngest age category. In adolescents older than 15 years, the agreement with RNB was much higher (55.6% files).

In total, there was 2861 reference breaths. Our algorithm successfully detected all of them (100%). It detected no false negative breaths (0%). Our algorithm returned 2876 breath ends, hence it returned 15 false positive breaths (0.52 %). Later, we are going to use these numbers to compare our algorithm with other published methods for finding breath ends, since it is a commonly used measure in most of the cited papers.

Related to Alg-Spi, the higher effectiveness of Alg-OUR in comparison with Alg-Spi is clearly demonstrated in Figure 9.4. Note that there was no A-file for which the Alg-OUR was less effective than Alg-Spi. Additionally, the performance of the two algorithms (Alg-OUR and Alg-Spi) was compared with RNB using the Wilcoxon paired test. Alg-OUR did not detect significantly different numbers of breaths in comparison with RNB ($p = 0.789$), while Alg-Spi did ($p < 0.001$).

9.4.4 Final thoughts on our algorithm

We proposed an innovative algorithm for breath detection that has similar accuracy to that of human experts. In comparison with the existing threshold-based algorithms and commercial software algorithm, it exhibits significantly higher success rate in recognition of true breaths, especially in severely distorted data. The algorithm addresses both the problems of false negative and positive breath detection. We are convinced that the higher performance is caused by a simultaneous use of more types of information obtained from the measurement and by respecting the basic facts of respiratory tract physiology. The main characteristics taken into account when designing our algorithm were:

1. The breath end corresponds to the time spot, when the inspiration starts and expiration ends or vice versa. Consequently, the direction of flow must change. This is the crucial presumption that we implemented it in step 2 of Algorithm 9.1.

2. During the expiration, carbon dioxide, which is being produced by body metabolism, is eliminated from the alveoli. Consequently, CO_2 concentration in exhaled air increases up to 6%. Its concentration during the expiration needs to be at least 2%, otherwise CO_2 will cumulate in the body, which will lead to respiratory failure. This characteristic is reflected in step 5 of Algorithm 9.1. It allows for the elimination of false breaths like breath (A) in Figure 9.5.

3. Carbon dioxide concentration in atmospheric (inhaled) air is approximately 0.04%. Consequently its concentration between two subsequent zero-crossings that both correspond to the true breath ends must drop close to this level. The level of 0.5% was chosen to safely allow for minor technical issues such as time shift of signals. This characteristic is reflected in step 6 Algorithm 9.1. It discards the zero-crossing (C) or earlier discarded zero-crossing (A) in Figure 9.5.

4. Volume of inhaled air must be approximately the same as the volume of exhaled air. In case when these volumes differ by more than 5 times, severe hyperinflation or detrimental changes to residual volume would occur. This is not attributable to physiologic tidal breathing. This characteristic is reflected in step 7 of Algorithm 9.1) and would discard the zero-crossing (E) in Figure 9.5.

These are the only theoretical assumptions used in our algorithm. No standalone assumption is universal, i.e., it is not sufficient to eliminate all false breath ends. However, appropriate combination and sequence of these conditions does the job very well.

In comparison with the previously published algorithms [13], [216] and [198] and with the threshold one implemented in Exhalyzer D, `Alg-OUR` introduced several unique features:

- **No preset thresholds** – as the algorithm is based only on generally valid assumptions from respiratory tract physiology, it does not require any pre-set threshold or other patient specific limitations. Our algorithm is applicable in all

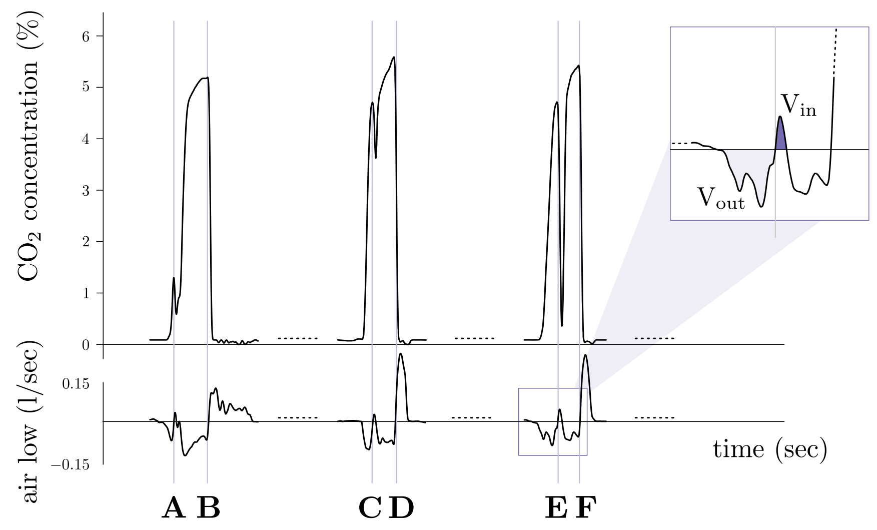

Figure 9.5: Possible cases of the shape of flow and CO$_2$ curves in real data. Vertical bars correspond to zero-crossings. The zero-crossings **B**, **D** and **F** correspond to true breath ends. The zero-crossings **A**, **C** and **E** form false breath ends and need to be filtered out. In the top right corner, there is a zoom-in of part of the curve below. It shows the ratios of volumes of exhaled and inhaled air.

types of respiratory diseases including restrictive, obstructive and mixed ventilatory disorders. We were able to use it successfully in patients with variety of obstructions (cystic fibrosis, primary ciliary dyskinesia, obstructive bronchitis, etc.).

- **Robustness** – the algorithm is capable to detect breath ends even in severely distorted data, there is no need of strict adherence to the tidal breathing.
- **No false negatives** – the algorithm was able to detect all breaths that human experts detected.
- **Simplicity** – the algorithm is easy to describe and implement in software.
- **Generalizability** – the principles of our algorithm can be translated to other types of gases (oxygen, nitrogen, sulphur hexafluoride, . . .).
- **Grouping of zero crossings** – the algorithm groups together the zero-crossings corresponding to one breath.

To the best of our knowledge, there exist only two previously published algorithms using CO$_2$ concentration signal to detect breath ends. An algorithm presented by Brunner et al [23] was developed in the 1980s and was intended for patients from intensive care units. In contrast to our algorithm, it does not include calculation and comparison of tidal volume of the consecutive breaths to filter out false positive breaths. Moreover, their algorithm does not use grouping of zero-crossings. Their validation was performed only on healthy subjects with intentionally introduced artefacts. The validation in children and on severely distorted data was missing. They reported there was no apparent algorithm failure during its clinical use on 100 patients, however precise specification of the testing conditions are not transparent. They provide test results from only one patient (150 breaths in total). Govindajaran and Prakash [55] proposed an algorithm for breath detection during different modes of artificial ventilation (volume and pressure controlled, patient triggered modes). They used mainly the flow and airway pressure signals; CO_2 data were only an additional input to confirm a computed delineation of detected breaths. They did not report the accuracy of the algorithm and no validation was performed. Because the algorithm is designed for artificial ventilation, it is of limited applicability in lung function testing.

Besides the algorithms based on flow and gas concentration signal analysis, another approaches to breath detection were proposed. Recently, Nguyen C. D. et al developed a breath detection algorithm based on finding inflexion points in flow or epiglottic pressure signal [142]. The validation was performed in healthy individuals and in patients with sleep obstructive apnoea syndrome using continuous positive airway pressure therapy (CPAP). Their algorithm correctly identified 97.6% of reference breaths. They do not mention false positives. If we assume there are no false negatives it makes 2.4% false negative detections (for the sake of comparison, our algorithm returned no false negatives and only 0.5% false positives). Moreover, their approach needs pressure measurement during CPAP therapy and relies on tight face mask.

There is also an approach using neural networks [197] on respiratory volume data. They tested it on three young healthy volunteers and six healthy infants. Their algorithm shows similar or better results than other existing algorithms using volume information [27] and [220]. The accuracy of the algorithm was 98% of the reference number of breaths with 2% false negatives and 5% false positives.

Another approaches use body image processing techniques analyzing body position and movements [10], [213] and photopletysmographic approach [120]. Nevertheless, such algorithms are more suitable for monitoring of vital functions rather than for further clinical processing.

We acknowledge several limitations of our algorithm. Although it outperforms the currently existing algorithms in their accuracy, it still suffers from false breath detection on severely distorted data. This only proves the difficulty of the task of automated processing. Even two independent human experts might not agree on what is the proper breath identification for a given dataset. That explains the small chance of having this problem fully solved by a computer. Moreover, in our study we did not include a comparison of `Alg-OUR` to the breath detection algorithms based on neural networks or sound analysis. However, we primarily focused on lung function testing, which relies on flow and gas concentration signal. The other algorithms have their application in other fields of medicine (e.g., sleep medicine). There also exist various possibilities to extend our algorithm, which we did not investigate in greater detail. One of the next steps might be creating a database of documented patterns of breathing curve behavior and its combination with breath end detection.

9.5 Nitrogen concentration at peaks

After localization of the breath ends the imprecision of machine sensors must be incorporated. We used machine Exhalyzer D, that does not measure nitrogen concentration directly. It computes the nitrogen concentration (in %) according to the formula [101]

$$100 = N_2\% + O_2\% + CO_2\% + Ar\%,$$

where $Ar\% = N_2\% \times 0.0093/0.7881$ and where the concentrations of nitrogen, oxygen, carbon dioxide and argon in inspired and expired air are supposed to sum up to 100 %. The argon concentration is fixed. Together it gives

$$N_2\% = \frac{1}{1.0118}(100 - O_2\% - CO_2\%),$$

where all parameters are in percents.

According to the manufacturer, the O_2 sensor has 0.3% accuracy and the CO_2 sensor has 5% accuracy. From that we can derive a interval bounds for the nitrogen concentration in each time slice $\boldsymbol{n_i}$

$$\underline{n}_i = \frac{1}{1.0118}(100 - 1.003 * O_2\% - 1.05 * CO_2\%),$$
$$\overline{n}_i = \frac{1}{1.0118}(100 - 0.997 * O_2\% - 0.95 * CO_2\%).$$

We subtracted the minimal possible value from 100 to obtain upper bound and the maximal possible value from 100 to obtain lower bound. In the MBW procedure there are many sources of errors:

- Imprecision of sensors
- Changing viscosity and humidity of air
- Time shift of signals
- Interaction with deadspace air
- Physiological noise (heart pulse, hick-ups, leaks)
- Irregular breathing pattern, apnea
- Computer and machine rounding errors
- etc.

Unknown distributions and interplay of the mentioned uncertain variables will result in intervals with unknown distribution. Hence it is necessary to work with only lower and upper bounds. Here the interval analysis can be viewed as a tool for dealing with such uncertainties algebraically (using the means of interval linear algebra). We further view the data as interval data as depicted in Figure 9.6.

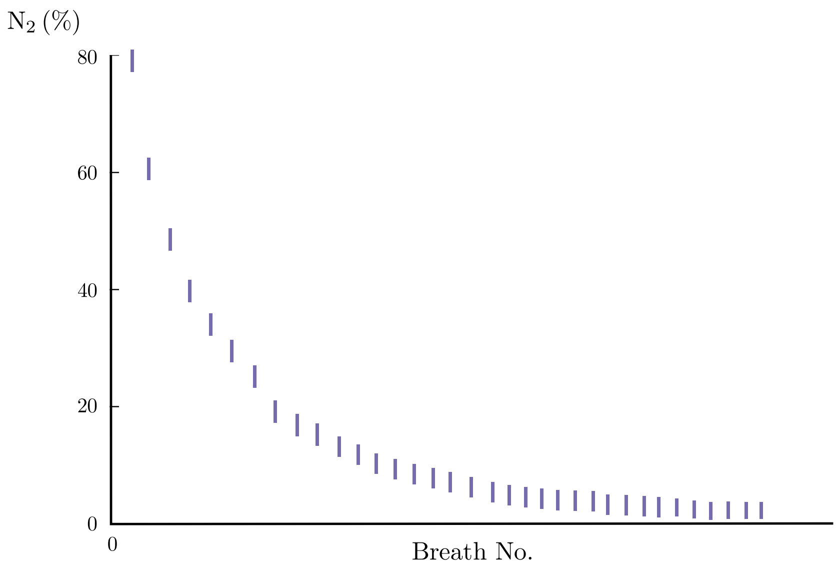

Figure 9.6: Illustration of decreasing concentration of nitrogen in peaks bounded with intervals.

9.6 Questions we asked

After long discussions we stated a few questions that are interesting from both clinical and mathematical point of view. The important and still discussed question is the behavior of the nitrogen washout in time. There is an observable difference between a healthy and diseased person, however the objective description is still missing. The long duration of washout (especially in severely affected patients) limits the feasibility of the test especially in small children (toddlers and pre-schoolers). Currently, the premature cessation of the washout (before reaching 2.5% of the starting nitrogen concentration) prevents us from analyzing the data. The possibility to derive substitute indices computable from an incomplete washout curve would be of great benefit.

9.7 Regression on interval data

Various authors approached the topic of regression on interval data, e.g, [24, 34, 77, 211]. Behind an interval regression or interval estimation the following general definition can be seen.

Definition 9.2. A result of a multi-linear interval regression on (interval) data tuples

$$(\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \ldots, \boldsymbol{x}_n^i, \boldsymbol{y^i}),$$

is generally

$$\boldsymbol{r}(x_1, x_2, \ldots, x_n) = \boldsymbol{p}_1 x_1 + \boldsymbol{p_2} x_2 + \cdots + \boldsymbol{p_n} x_n,$$

where $\boldsymbol{p} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)^T$ are interval parameters.