INTERNATIONAL STANDARD

**ISO 532-1**

First edition
2017-06

Corrected version
2017-11

# Acoustics — Methods for calculating loudness —

## Part 1: Zwicker method

*Acoustique — Méthode de calcul du niveau d'isosonie —*

*Partie 1: Méthode de Zwicker*



Reference number
ISO 532-1:2017(E)



ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

# Contents

Page

# Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html

This document was prepared by Technical Committee ISO/TC 43, *Acoustics*.

This first edition cancels and replaces ISO 532:1975, which has been technically revised.

A list of all parts in the ISO 532 series can be found on the ISO website.

This corrected version of ISO 532-1:2017 incorporates the following correction:

— Table A.9 has been corrected.



# Introduction

Loudness and loudness level are two perceptual attributes of sound, describing absolute and relative sensations of sound strength perceived by a person under specific listening conditions. Due to inherent individual differences among people, both loudness and loudness level have the nature of statistical estimators characterized by their respective measures of central tendency and dispersion determined for a specific sample of the general population.

The object of the ISO 532 series is to specify calculation procedures based on physical properties of sound for estimating loudness and loudness level of sound as perceived by persons with otologically normal hearing under specific listening conditions. Each procedure provides single numbers that can be used in many scientific and technical applications to estimate the perceived loudness and loudness level of sound, without conducting separate human observer studies for each application. Because loudness is a perceived quantity, the perception of which may vary among people, any calculated loudness value represents only an estimate of the average loudness as perceived by a group of individuals with otologically normal hearing.

ISO 532-1 and ISO 532-2 specify two different methods for calculating loudness which may yield different results for given sounds. Since no general preference for one or the other method can presently be stated, it is up to the user to select the method which appears most appropriate for the given situation. Some major features of each of the methods are described below to facilitate the choice.

The first method of this document describes the calculation of loudness and loudness level of stationary sounds and is based on DIN 45631:1991. The second method of this document covers the procedures for calculation of loudness and loudness level of arbitrary non-stationary (time-varying) sounds, including stationary sounds as a special case, and is based on DIN 45631/A1:2010.

This document also includes a program code for both methods leading to estimates of loudness and loudness level for stationary and time-varying sounds. An executable computer program is also provided for both methods. The applied software is normative for calculating loudness values, against which other implementations can be checked subject to stated tolerances, and provides additional functionality for the convenience of the user.

The method for stationary sounds in this document differs slightly from the methods included in the previous ISO 532:1975, method B, by specifying corrections for low frequencies and by restricting the description of the approach to numerical instructions only, thus allowing a unique software description. For reasons of continuity, the method given in this document is in accordance with ISO 226:1987 instead of the later revised version, ISO 226:2003.

Based on the general concept of the method for stationary sounds, the method for time-varying sounds incorporates a generalization of the Zwicker approach to arbitrary, non-stationary sounds. Of course, this generalization is compatible with the method for stationary sounds in that it gives the same loudness values as the method for stationary sounds if applied to stationary sounds.

The Moore-Glasberg method as implemented in ISO 532-2 is limited to stationary sounds and can be applied to tones, broadband noises and complex sounds with sharp line spectral components. The method in ISO 532-2 differs from those in ISO 532:1975. ISO 532:1975, method A (Stevens loudness), was removed as this method was not often used and its predictions were not accurate for sounds with strong tonal components. The method described in ISO 532-2 also improves the precision of calculated loudness in the low frequency range and allows for calculation of loudness under conditions where the sound differs at the two ears. It has been shown that this method provides a good match to the contours of equal loudness level as defined in ISO 226:2003 and the reference threshold of hearing as defined in ISO 389-7:2005.

NOTE Equipment or machinery noise emissions/immissions can also be judged by other quantities defined in various International Standards (see e.g. ISO 1996-1, ISO 3740, ISO 9612 and ISO 11200).

# Acoustics — Methods for calculating loudness —

# Part 1: Zwicker method

## 1 Scope

This document specifies two methods for estimating the loudness and loudness level of sounds as perceived by otologically normal persons under specific listening conditions. The first method is intended for stationary sounds and the second method for arbitrary non-stationary (time-varying) sounds, including stationary sounds as a special case.

The methods can be applied to any sound recorded as single-channel measurements using a microphone, or as multi-channel measurements, for example by means of a head and torso simulator (see Annex D). Since most important technical sounds are time-varying, a model of time-varying loudness is preferable.

The methods are based on the Zwicker algorithm.[14] The method for stationary sounds is provided for reasons of continuity and also offers the use of measured one-third-octave-band levels as input. The more general method for arbitrary sounds calculates the specific loudness pattern based on measured time signals by applying a signal processing model that is directly related to physiological and psychological characteristics of the human hearing system. Loudness is calculated from the specific loudness pattern. It has been shown that this method provides a good match to the results of many loudness experiments using synthetic and technical sounds.

No prior knowledge about the properties of the sound (e.g. broadband or narrowband noise, tonal content) and no user interactions are required for the fully automated application of the method.

The evaluation of the harmful effect of sound events is outside the scope of this document.

## 2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61260-1:2014, *Electroacoustics — Octave-band and fractional-octave-band filters — Part 1: Specifications*

IEC 61672-1:2013, *Electroacoustics — Sound level meters — Part 1: Specifications*

## 3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at http://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

**3.1**
**otologically normal person**
person in a normal state of health who is free from all signs or symptoms of ear disease and from obstructing wax in the ear canals, and who has no history of undue exposure to noise, exposure to potentially ototoxic drugs or familial hearing loss

[SOURCE: ISO 226:2003, 3.1]

**3.2**
**sound pressure level**
$L_p$
ten times the logarithm to the base 10 of the ratio of the square of the sound pressure, $p$, to the square of a reference value, $p_0$, expressed in decibels

$$L_p = 10 \lg \frac{p^2}{p_0^2} \text{ dB}$$

where the reference value, $p_0$, is 20 µPa

Note 1 to entry: Because of practical limitations of the measuring instruments, $p^2$ is always understood to denote the square of a frequency-weighted, frequency-band-limited or time-weighted sound pressure. If specific frequency and time weightings as specified in IEC 61672-1 and/or specific *frequency bands* (3.3) are applied, this should be indicated by appropriate subscripts, for example, $L_{p,\mathrm{AS}}$ denotes the A-weighted sound pressure level with time weighting S (slow). Frequency weightings such as A-weighting should not be used when specifying sound pressure levels for the purpose of *loudness* (3.18) calculation using the current procedure.

Note 2 to entry: This definition is technically in accordance with ISO 80000-8:2007, 8-22.

**3.3**
**frequency band**
continuous set of frequencies in the range of two specified limiting frequencies

Note 1 to entry: A frequency band is characterized by two values that define its position in the frequency spectrum, for instance its lower and upper cut-off frequencies.

Note 2 to entry: Frequency is expressed in Hz.

[SOURCE: IEC 60050-702:1992, 702-01-02]

**3.4**
**filter**
device or mathematical operation that, when applied to a complex signal, passes energy of signal components of certain frequencies while substantially attenuating energy of signal components of all other frequencies

**3.5**
**cut-off frequency**
lowest ($f_\mathrm{l}$) or the highest ($f_\mathrm{h}$) frequency beyond which the response of the *filter* (3.4) to a sinusoidal signal does not exceed −3 dB relative to the maximum response measured between $f_\mathrm{l}$ and $f_\mathrm{h}$

**3.6**
**band-pass filter**
*filter* (3.4) that passes signal energy within a certain *frequency band* (3.3) and rejects most of the signal energy outside of this frequency band

**3.7**
**filter bandwidth**
$\Delta f$
difference between $f_\mathrm{h}$ and $f_\mathrm{l}$ for a *band-pass filter* (3.6)

**3.8**
**one-third-octave band**
*frequency band* (3.3) with the centre frequency $f_T$ and the width of one-third of an octave

Note 1 to entry: The subscript T instead of c is used to specify the centre frequency in the special case of a one-third-octave band.

Note 2 to entry: Width of one-third of an octave as specified in IEC 61260-1.

**3.9**
**band-reject filter**
*filter* (3.4) that rejects signal energy within a certain *frequency band* (3.3) and passes most of the signal energy outside of this frequency band

Note 1 to entry: A narrow band-reject filter is also called a notch filter.

**3.10**
**one-third-octave-band level**
$L_T$
*sound pressure level* (3.2) of sound contained within a *frequency band* (3.3) with the width of one-third of an octave

**3.11**
**sound spectrum**
representation of the magnitudes (and sometimes of the phases) of the components of a complex sound as a function of frequency

**3.12**
**critical band**
**auditory filter**
*filter* (3.4) within the human cochlea describing the frequency resolution of the auditory system with characteristics that are usually estimated from the results of masking experiments

**3.13**
**critical bandwidth**
**auditory filter bandwidth**
bandwidth of a *critical band* (3.12)

Note 1 to entry: The critical bandwidth values in Hz are as specified in Reference [22].

Note 2 to entry: Each critical bandwidth has a width of one unit on the *critical band rate scale* (3.14).

**3.14**
**critical band rate scale**
transformation of the frequency scale, constructed so that an increase in frequency equal to one *critical bandwidth* (3.13) leads to an increase of one unit on the critical band rate scale

Note 1 to entry: Frequencies on the critical band rate scale are expressed in Bark.

EXAMPLE The value of the critical bandwidth for a centre frequency of 1 000 Hz is approximately 160 Hz, so an increase in frequency from 920 Hz to 1 080 Hz corresponds to a step of one Bark.

**3.15**
**critical band level**
$L_{CB}$
*sound pressure level* (3.2) of sound contained within a *critical band* (3.12)

**3.16**
**loudness level**
*sound pressure level* (3.2) of a frontally incident, sinusoidal plane progressive wave, presented binaurally at a frequency of 1 000 Hz that is judged by *otologically normal persons* (3.1) as being as loud as the given sound

Note 1 to entry: Loudness level is expressed in phons.

**3.17**
**calculated loudness level**
$L_N$
*loudness level* (3.16) calculated following the procedure of a predictive model

**3.18**
**loudness**
perceived magnitude of a sound, which depends on the acoustic properties of the sound and the specific listening conditions, as estimated by *otologically normal persons* (3.1)

Note 1 to entry: Loudness is expressed in sones.

Note 2 to entry: Loudness depends primarily upon the *sound pressure level* (3.2), although it also depends upon the frequency, waveform, bandwidth, and duration of the sound.

Note 3 to entry: One sone is the loudness of a sound with a *loudness level* (3.16) of 40 phon.

Note 4 to entry: A sound that is twice as loud as another sound is characterized by doubling the number of sones.

**3.19**
**calculated loudness**
$N$
*loudness* (3.18) calculated following the procedure of a predictive model

Note 1 to entry: The calculated loudness is denoted $N_F$ or $N_D$, expressed in sones. The letters F and D signify that the calculation assumes either free field frontal sound incidence (F) or diffuse field incidence (D).

Note 2 to entry: The calculated loudness corresponds to the loudness that would be experienced by an average of a group of persons with otologically normal hearing whose heads are centred at the position of the microphone. This is equivalent to diotic listening (same sound at each ear).

**3.20**
**specific loudness**
$N'$
*loudness* (3.18) evoked over a *frequency band* (3.3) with a bandwidth of a *critical band* (3.12) centred at the frequency of interest

Note 1 to entry: Specific loudness is expressed in sones/Bark.

Note 2 to entry: The definition together with the stated unit are different from those in ISO 532-2.

**3.21**
**percentile loudness**
$N_X$
*loudness* (3.18) that is reached or exceeded in $X$ % of the measuring time intervals

Note 1 to entry: The percentile loudness is expressed in sones.

## 4 Specification of input signal and instrumentation

The input signal is measured using a sound acquisition system consisting of at least a microphone, pre-amplifier and amplifier. The instrumentation shall meet the requirements of IEC 61672-1:2013, class 1, as far as applicable. The waveform of the time signal is sampled with an A/D-converter. The test

implementation in A.4 requires a sampling rate of 48 kHz. Signals with other sampling rates shall be resampled to 48 kHz when using this implementation. For the convenience of the user an additional source file containing an algorithm for up-sampling signals with sampling rates of 32 kHz or 44,1 kHz is provided (see '**// BLOCK Resampling**').

For reasons of convenience, the program code is extended to allow the use of time signals in the form of WAVE files for the calculation algorithm of loudness. The data format can be 16-bit integer or 32-bit float format (correct sound pressure values, no normalized data). For a 16-bit integer format, an appropriate calibration file and the corresponding calibration value shall be given. The corresponding algorithms are also contained in the additional source file.

The type of sound field [free (F) or diffuse (D)] shall be specified.

The one-third-octave-band sound pressure levels are determined using one-third-octave-band filters that conform to IEC 61260-1:2014, class 1 with centre frequencies spanning the range between 25 Hz and 12,5 kHz. In order to reduce uncertainties of loudness calculation the filter coefficients are provided (A.2). The one-third-octave-band filters are designed as 6th order Chebyshev filters using three 2nd order sections. These filters are optimized to provide a damping of 20 dB at the centre frequencies of the adjacent bands (except for 12,5 kHz only the lower band is considered). This means that, for example, a 1 kHz tone with a sound pressure level of 70 dB produces the following levels at different centre frequencies: 50 dB at 800 Hz, 70 dB at 1 kHz and 50 dB at 1,25 kHz.

The method for stationary sounds also offers the use of one-third-octave-band levels measured with a sound level meter according to IEC 61672-1:2013, class 1, and a filter according to IEC 61260-1:2014, class 1, as input.

NOTE A software package including the listing of the program code can be freely downloaded from http://standards.iso.org/iso/532/-1/ed-1/en



## 5 Method for stationary sounds

### 5.1 General

This specifies a method for calculating loudness and loudness level of stationary sounds using objective measurement data and a calculation procedure developed by E. Zwicker and his colleagues. [14] The calculated data allow the loudness to be specified and thus different stationary sounds to be quantitatively evaluated according to their individual expected perceived loudness. The procedure applies to all stationary sounds.

The method for stationary sounds given in this document is an updated version of method B as given in ISO 532:1975, providing a modified treatment of low-frequency energy. The calculation is based on an algorithmic approach as opposed to a graphical method. The modified treatment for frequencies below 300 Hz strictly follows the approach given by DIN 45631:1991, introduced to improve the accuracy of the calculated loudness. Since then, this modified version has achieved widespread use and thus frequently replaced previous applications of ISO 532:1975, method B. It therefore can be seen as the direct continuation of the frequently used Zwicker method.

It is this aim of maximum continuity which has prevented any other changes because the benefit of some smaller adoptions was seen to be smaller than the loss of continuity and comparability of data. For reasons of continuity, the method given in this document is in accordance with ISO 226:1987 instead of the later revised version ISO 226:2003.

The procedure involves a sequence of steps. For a precise definition of each step the user is referred to a detailed description in 5.2 and the program code in A.4. Both descriptions of the procedure are provided in order to facilitate the comprehension of the procedure. It is envisaged that those wishing to calculate loudness using the method for stationary sounds will use an implementation of the algorithm given by the computer program of A.4 or other implementations in conformance with the tolerances given in the paragraph below.

The implementation given in Annex A shall be used as the test method against which other implementations shall be tested to determine compliance with this document.

Any specific implementation is permitted if, for all stationary test signals given in B.2 and B.3,

— their calculated specific loudness values differ by not more than ± 5 % of the specific loudness values calculated by the test implementation or ± 0,1 sone/Bark, and if

— the deviation for total loudness is not more than ± 5 % or ± 0,1 sone.

The supplier of any specific implementation shall provide a declaration of conformance with this document, indicating the results achieved using the test signals given in Annex B, e.g. as part of the solution's user manual. For convenience, the test signals and the tables of the results from the test implementation (including tolerances) are supplied electronically (as WAVE and EXCEL files). Modifying the test signals is not allowed.

NOTE A software package including the WAVE and EXCEL files can be freely downloaded from http://standards.iso.org/iso/532/-1/ed-1/en

## 5.2 Description of the method

The method for calculating loudness consists of three steps. These steps provide a means of combining and converting the one-third-octave-band levels to give the total loudness level. These steps are described below and illustrated by a block diagram in Figure 1. An example for factory noise in a diffuse sound field is also given for further explanation (see Figure 2). Even though the abscissa gives cut-off or centre frequencies, Figure 2 is scaled according to critical band rate. At each one-third-octave band, "ladders" can be seen, the rungs of which represent possible values of the respective one-third-octave-band levels.

**Step 1**

This step accounts for the fact that the human hearing system is less sensitive at low frequencies below 300 Hz than at higher frequencies. This is done by appropriately decreasing the respective one-third-octave- band levels before entering them into the diagram of Figure 2. The details of these corrections can be taken from Table A.3.

**Step 2**

The approximation of critical bands by one-third-octave bands is only acceptable for frequencies above about 300 Hz. For lower frequencies, one-third-octave bands are smaller than the critical bands, so two or more one-third-octave bands shall be added in order to approximate critical bands. This is the case for all evaluated one-third-octave bands between 20 Hz and 90 Hz ($L_{CB'1}$), for the three one-third-octave bands between 90 Hz and 180 Hz ($L_{CB'2}$), and for the two one-third-octave bands between 180 Hz and 280 Hz ($L_{CB'3}$). In these cases, the critical band level shall be approximated by power summation in the given one-third-octave bands. The thick horizontal lines shown in Figure 2 with the width of about a critical band at low centre frequencies were produced in this way from the smaller horizontal thin bars which correspond to the measured one-third-octave-band levels.

**Step 3**

Before entering the corrected one-third-octave-band levels into the diagram (in order to transform the levels into their corresponding core loudness, see A.3), it shall be ascertained whether the spectrum was obtained under diffuse or free field conditions. In the example of Figure 2, this was done by using the appropriate ladder structure assigning spectral values to specific loudness values. Graphical representations of different ladder structures for a wide range of one-third-octave-band levels in diffuse or free field were given in ISO 532:1975 and DIN 45631:1991. In practice, however, the appropriate values of the ladder structure currently are defined within numerical tables. For reasons of simplicity and convenience, this document uses a C-implementation of the algorithm instead of particular tables to specify all numerical allocations. The letters "D" for diffuse and "F" for free field shall be used to specify the algorithm applied.



Then, for each band a slope towards the higher critical band is added, and the area below the distribution of specific loudness is summed. The specific value of the slope to be added depends on the respective one-third-octave-band levels and centre frequencies. Again, detailed information can be found in the above mentioned graphical representations or in the tables of A.3, respectively. Having entered the corrected one-third-octave-band levels into the diagram, the shape of the specific loudness pattern starts with a vertical rise to the one-third-octave-band level measured, stays at the main value corresponding to the one-third-octave-band level in question and then falls with a slope unless the level is higher in the next one-third-octave band, in which case the pattern rises vertically to the level appropriate for the next one-third-octave band. Both the one-third-octave-band spectrum and the loudness pattern are highlighted by solid curves in the diagram of Figure 2.

If the next one-third-octave-band level is lower, the decrease of the specific loudness towards higher centre frequencies follows the broken lines, corresponding to the upper slope. In this way, the final specific loudness versus critical band rate pattern, shortened to "loudness pattern", is determined and indicated by the highest thick solid lines in Figure 2. For narrow-band sounds, this upper slope contributes strongly to the total loudness, i.e. to the total area below the curve. Therefore, it contributes especially to the total loudness of pure tones. An example is given in Figure 2 by the dotted line for a 1 kHz tone with a sound pressure level of 70 dB. Generally, one-third-octave-band filters show a leakage towards neighbouring filters of about −20 dB. This means that a 1 kHz tone with a sound pressure level of 70 dB produces the following levels at different centre frequencies: 50 dB at 800 Hz, 70 dB at 1 kHz and 50 dB at 1,25 kHz. Therefore, the lower slope of the loudness pattern becomes less steep.

The solid curve in Figure 2 shows the loudness pattern of a factory noise. An area is formed extending from low to high frequencies. It is bordered by the straight line upwards at the left and right sides of the overall diagram, and also by the horizontal lower abscissa. The area within these boundaries is marked by hatching. To calculate the area quantitatively, a rectangular surface of equal area is drawn, which has the width of the diagram as a basis. The height of this rectangle is a measure of the total area, which is marked by shading from upper left to lower right. Using this height (the dashed-dotted line), the loudness or the loudness level can be read from the scales on the right or the left of the diagram. In the diffuse field example shown in Figure 2, a calculated loudness $N_D$ of 24 sone and a corresponding loudness level $L_{N,D}$ of 86 phon is found. The sound under test has a relatively broad spectrum. Therefore there is quite a large difference between the measured sound pressure level of 73 dB or the A-weighted sound pressure level of 68 dB on the one hand, and the calculated loudness level of 86 phon on the other hand.

The graphical procedure which finally leads to a loudness pattern has the advantage that partial areas in the diagram correspond to specific parts of the loudness. Therefore, in many cases the diagram clearly shows which partial area is dominant or which part contributes strongly to the total loudness. In many applications it is often very important to first reduce that part of the noise which produces the largest area in the loudness pattern. On the other hand, the diagram shows which parts of the spectrum are so small in relation to the neighbouring parts that they are partially or even totally masked. In Figure 2, for example, the one-third-octave-band level of 51 dB at the centre frequency of 630 Hz does not contribute to loudness because it is totally masked, as indicated by the fact that this one-third-octave-band level lies below the shaded curve limiting the total area and arising, at this frequency, from the one-third-octave-band level at 500 Hz.

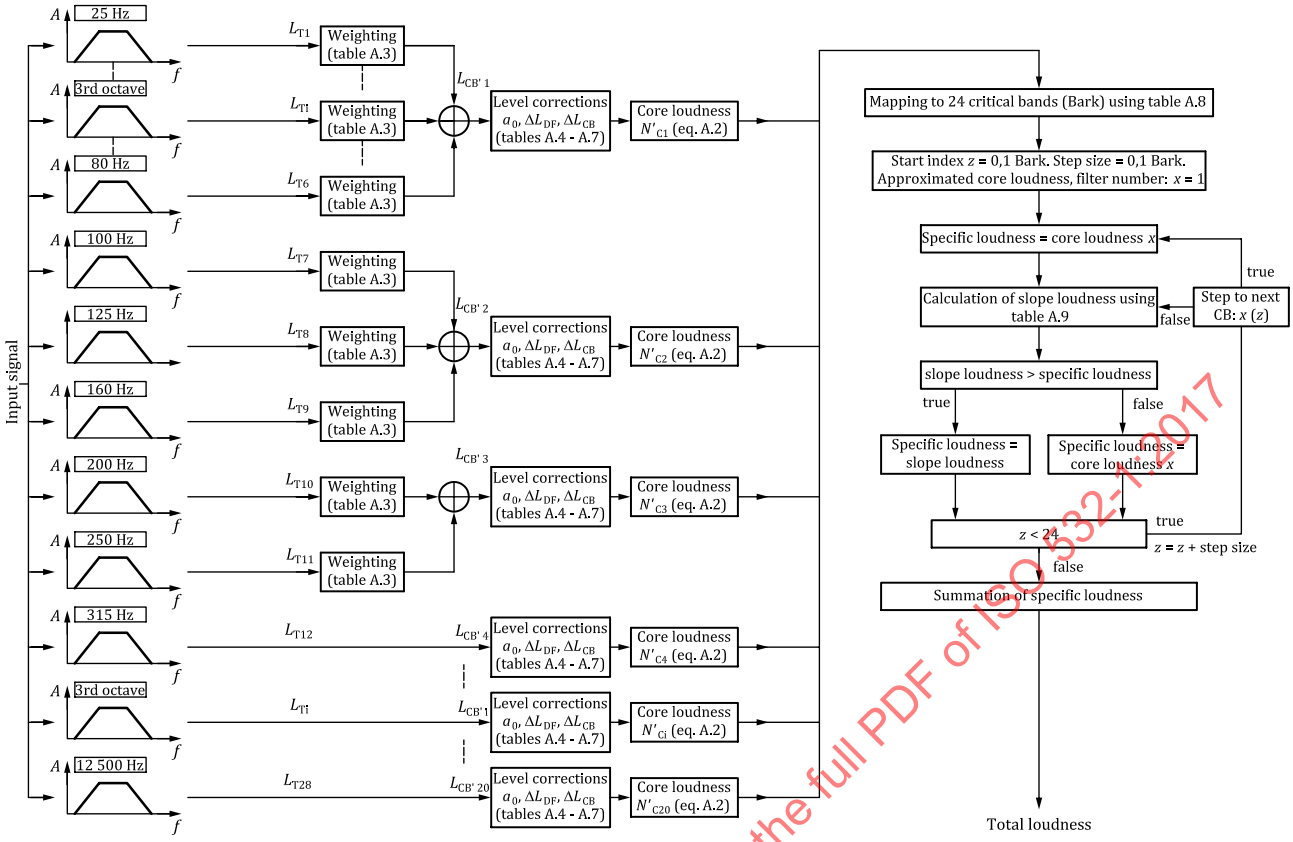
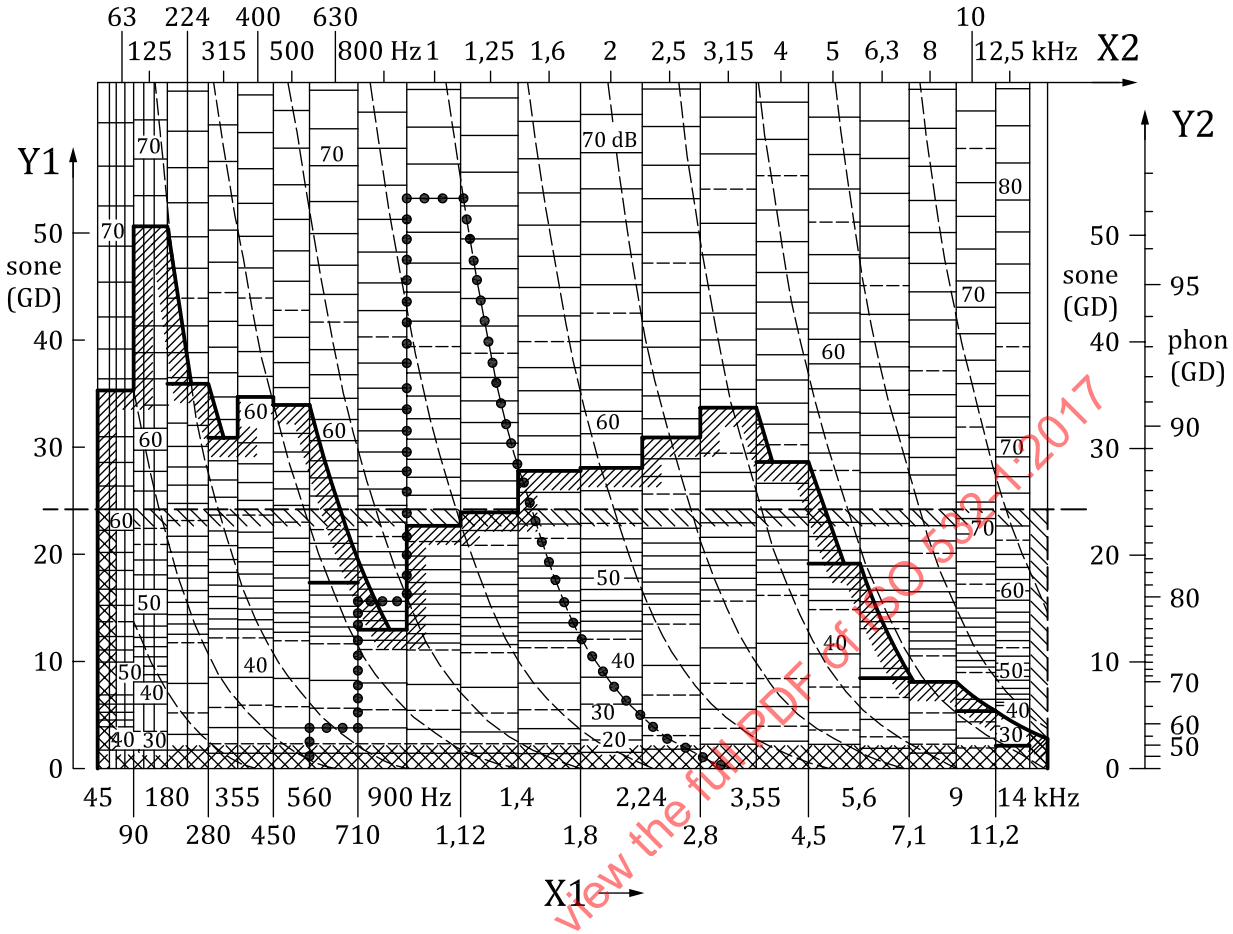



**Figure 1 — Block diagram of the calculation method by this document, method for stationary sounds**

Figure 2 shows a schematic diagram of the graphical calculation algorithm as implemented in this document, method for stationary sounds, according to DIN 45631:1991.




**Key**

X1 cut-off frequency of third-octave bands
X2 centre frequency
Y1 scale of total loudness
Y2 scale of corresponding loudness level

**Figure 2 — Example of the loudness calculation procedure using charts indicating the measured one-third-octave-band levels of a factory noise in a diffuse field (see Annex A)**

Explanation of Figure 2: Specific loudness is on the ordinate while the critical band rate expressed as cut-off frequencies of the one-third-octave bands is on the abscissa. The area surrounded by the *thick solid line* and hatched from *lower left* to *upper right* indicates the total loudness of the noise. This area is approximated by a rectangular area of the same width but with a height indicated by the area hatched from *upper left* to *lower right.* The height of this rectangular area marks the total loudness on the left scale and the corresponding loudness level on the right scale. The dotted curve represents the loudness pattern of a 1 kHz tone with a sound pressure level of 70 dB.

## 5.3 Calculation of loudness and loudness level

The loudness calculation can be performed from provided one-third-octave-band sound pressure levels (function 'f_loudness_from_levels' in A.4) or from calculated one-third-octave-band levels based on time signals (function 'f_loudness_from_signal' in A.4). The one-third-octave-band levels or the time signal (as WAVE file) can be easily input using the command line or the graphical user interface of the

software described in Annex C. The appropriate sound field (F or D) shall be specified. For the method for stationary sounds, loudness is given in sones and loudness level in phons [see Formulae (1) to (3)].

$$N = 2^{0,1\left(\frac{L_N}{\text{phon}} - 40\right)} \text{ sone} \tag{1}$$

where

$$L_N \geq 40 \text{ phon}$$

and

$$L_N = \left[40 + 33{,}22 \lg\left(\frac{N}{\text{sone}}\right)\right] \text{ phon} \tag{2}$$

where

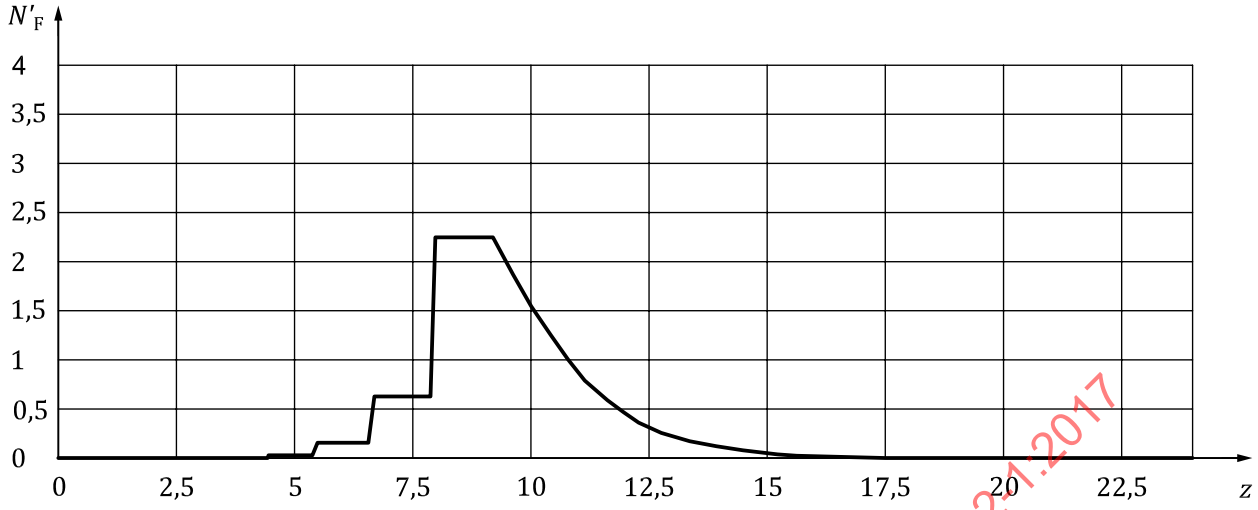
$$N \geq 1 \text{ sone}$$

respectively.

The relationship for $N < 1$ sone is

$$L_N = 40\left(\frac{N}{\text{sone}} + 0{,}0005\right)^{0,35} \text{ phon} \tag{3}$$

The example in Figure 3 shows the specific loudness versus critical band rate pattern for a single tone with the frequency $f$ = 1 kHz and with the one-third-octave-band sound pressure level $L_T$ = 70 dB in a free field. The specific loudness $N'_F$ in sones/Bark is plotted on the critical band rate scale $z$. The scale of the critical band rate is subdivided in the unit Bark; each of the 24 critical bands has a width of 1 Bark. The input for the calculation program is based on a one-third-octave-band level analysis, providing levels that are reduced by a typical side attenuation of 20 dB/one-third-octave band for $f_T < 1$ kHz and $f_T > 1$ kHz. The contribution of one-third-octave-band for $f_T < 1$ kHz to the total area remains comparably low. For $f_T > 1$ kHz, the one-third-octave-bands are masked by the upper slope of $f_T$ = 1 kHz. The total calculated loudness of the single tone results in $N_F$ = 8 sone with the calculated loudness level $L_{N,F}$ = 70,0 phon.



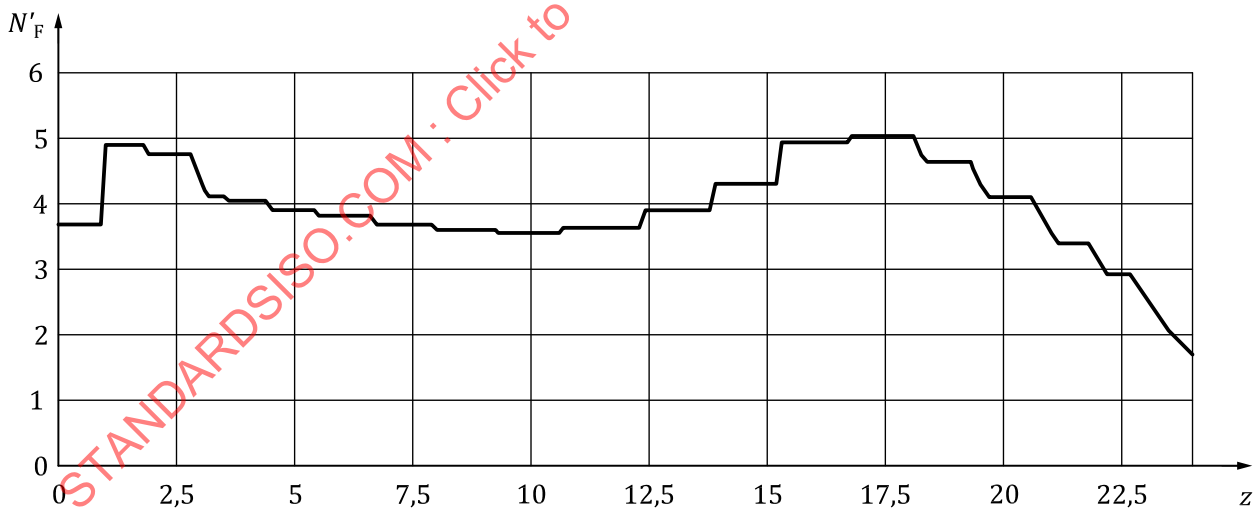

**Key**

$N'_F$ specific loudness, in sones/Bark

$z$ critical band rate, in Bark

**Figure 3 — Example of the specific loudness versus critical band rate pattern for a single tone in a free field**

The example in Figure 4 shows the specific loudness versus critical band rate pattern for pink noise in a free field with a constant third-octave-band level $L_T$ = 78 dB. For the loudness of this broad-band noise the calculation program gives a value of $N_F$ = 95,0 sone and a loudness level of $L_{N,F}$ = 105,7 phon. For comparison, the A-weighted sound pressure level in this example is $L_A$ = 90 dB.




**Key**

$N'_F$ specific loudness, in sones/Bark

$z$ critical band rate, in Bark

**Figure 4 — Example of the specific loudness versus critical band rate pattern of pink noise in a free field**

# 6 Method for time-varying sounds

## 6.1 General

This method specifies the requirements and the procedure to determine the loudness of arbitrary sounds. When using the procedure for stationary sounds to calculate the loudness of time-varying sounds, the values obtained are too low. Therefore, a procedure is described in this clause that can be used to simulate the time processing of sound by the human hearing system when evaluating the loudness of time-varying sounds.

The procedure involves a sequence of additional steps, as illustrated in Figure 5. For a precise definition of each step the user is referred to a detailed description in 6.2, 6.3 and the program code in A.4. The description of the procedure is provided in order to facilitate the comprehension of the procedure. However, it is envisaged that those wishing to calculate loudness using the method for time-varying sounds will use the algorithm given by the computer program of A.4 or the executable software given in Annex C.

The implementation given in Annex A shall be used as the test method against which other implementations shall be tested to determine compliance with this document.

Any specific implementation is permitted if

— for all time-varying test signals given in B.4, their calculated specific loudness vs. time function differs by not more than ± 5 % of the specific loudness vs. time function calculated by the test implementation or ± 0,1 sone/Bark, within a temporal tolerance of ± 2 ms. The tolerance can be extended to ± 10 % of the specific loudness vs. time function calculated by the test implementation or ± 0,2 sone/Bark, within a temporal tolerance of ± 2 ms, but only for maximum 1 % of the sampled specific loudness vs. time function using a time resolution of 2 ms; and if

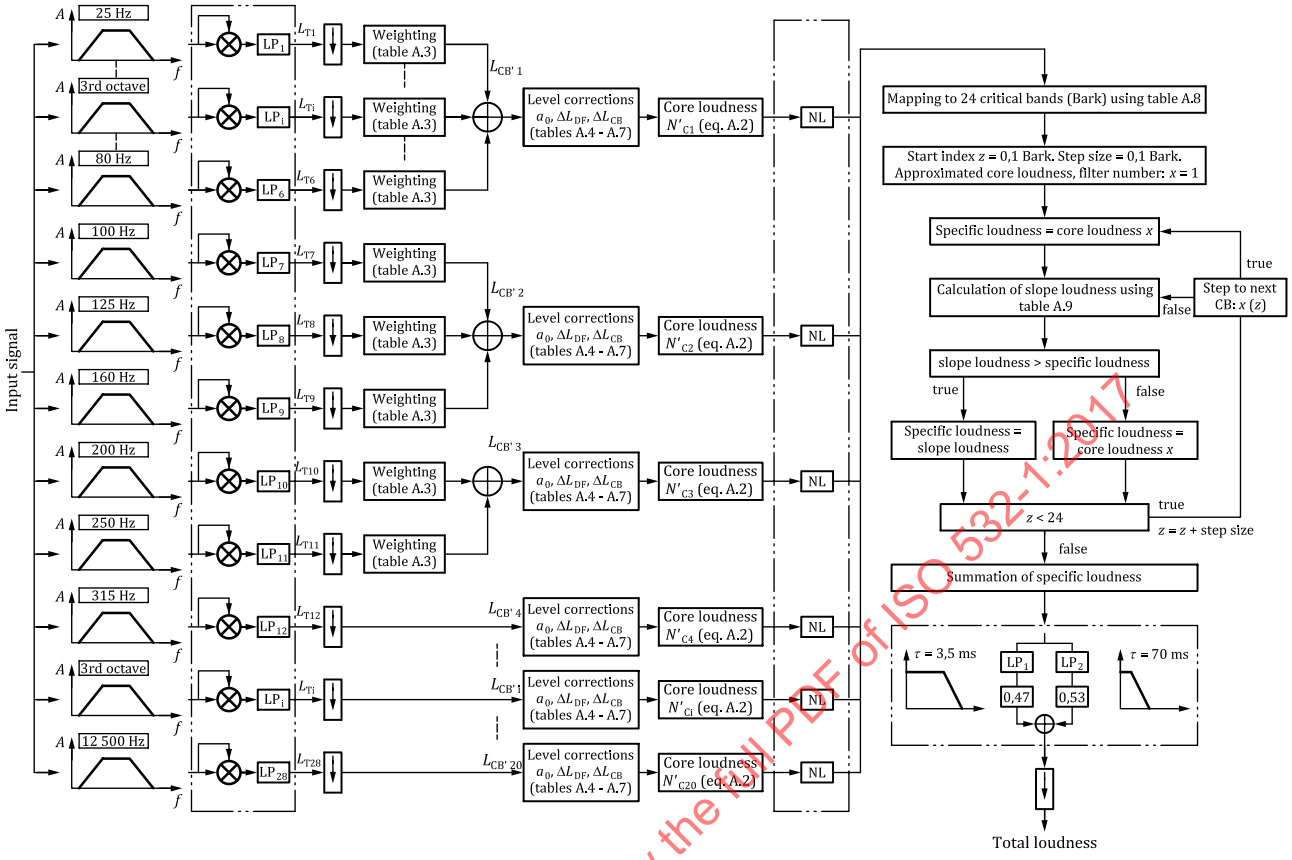
— for all time-varying test signals given in B.4 and B.5, the deviation for the total loudness vs. time function is not more than ± 5 % or ± 0,1 sone, within a temporal tolerance of ± 2 ms. The tolerance can be extended to ± 10 % of the loudness vs. time function calculated by the test implementation or ± 0,2 sone, within a temporal tolerance of ± 2 ms, but only for maximum 1 % of the sampled loudness vs. time function using a time resolution of 2 ms.

The supplier of any specific implementation shall provide a declaration of conformance with this document, indicating the results achieved using the test signals given in Annex B, for example as part of the solution's user manual. For convenience, the test signals and the tables of the results from the test implementation (including tolerances) are supplied electronically (as WAVE and EXCEL files). Modifying the test signals (e.g. appending zeros) is not allowed.

NOTE A software package including the WAVE and EXCEL files can be freely downloaded from http://standards.iso.org/iso/532/-1/ed-1/en






NOTE Additional elements with respect to Figure 1 are shown within the boxes with chain dotted lines.

**Figure 5 — Block diagram of the calculation method by this document, method for arbitrary sounds**

## 6.2 Description of the method

The calculation of the loudness of arbitrary sounds is based on the calculation of the loudness of stationary sounds. The spectral analyses using one-third-octave-band filters are common to both of these procedures. Although these calculations follow standards (IEC 61260-1) with defined tolerance bands, the loudness calculation procedure requires the filters to be defined more precisely. An important prerequisite is a damping of 20 dB at the centre frequencies of the adjacent bands. Thus, this document intends to reduce the uncertainties of the existing standards by defining all mathematical facts of the algorithms, starting with the waveform of the time signal and ending with specific and total loudness vs. time functions.[15] 16-bit integer data together with a reference signal of known sound pressure level (for calibration) or 32-bit float data (input as sound pressure in Pascal) can be used. For processing, a common sampling rate $f_S$ of 48 kHz is chosen in the test implementation given in Annex A. Signals with other sampling rates shall be resampled to this common sampling rate when using the test implementation. For the convenience of the user, an additional source file (see '**// BLOCK Resampling**') containing an algorithm for up-sampling signals with sampling rates of 32 kHz or 44,1 kHz is provided.

The temporal evaluation of the methods for stationary and time-varying sounds shows significant differences; the nonlinear decay of the human hearing system needs to be modelled in detail. Furthermore, a number of very complex hearing characteristics, such as effects of the temporal summation and forward-masking, shall be taken into account.

The procedure is in principle suitable for calculating the loudness of all time-varying sounds. Sounds with distinct time structures, such as sounds caused by helicopters and diesel motors as well as spoken language, yield results which can be used in various fields of application.

## 6.3 Calculation algorithm

The input signal is filtered by an array of 28 one-third-octave-band filters according to IEC 61260-1:2014, class 1, with centre frequencies $f_T$ from 25 Hz to 12 500 Hz, designed as 6th order Chebyshev filters using three 2nd order sections. The filter coefficients are given in Table A.1 and Table A.2 for a sampling rate $f_s$ of 48 kHz with 6 significant digits. A damping of 20 dB at the centre frequencies of the adjacent bands (except for $f_T$ = 12 500 Hz only the lower band is considered) was a parameter for optimizing the filter coefficients.

The output signals of the filter array are squared and smoothed by third-order low-pass-filters. These low-pass-filters are constructed using three identical low-pass-filters of first order in series. The corresponding digital filter design is shown in Figure 6.

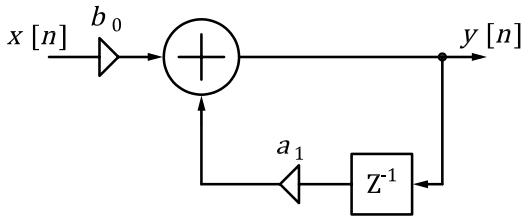



**Figure 6 — Filter structure of a first order low-pass filter**

The filter output is calculated using the recursive Formula (4) with the given filter coefficients.

$$\begin{aligned} y[n] &= b_0 x[n] + a_1 y[n-1] \\ b_0 &= 1 - \exp\left(-\frac{1}{f_s \tau}\right) \\ a_1 &= \exp\left(-\frac{1}{f_s \tau}\right) \end{aligned} \tag{4}$$

The frequency-dependent time constant is given in Formula (5).

$$\tau = \begin{cases} \frac{2}{3} \cdot \frac{1}{f_T} & \text{for} \quad f_T \leq 1\,\text{kHz} \\ \frac{2}{3} \cdot \frac{1}{1\,\text{kHz}} = \frac{2}{3}\,\text{ms} & \text{for} \quad f_T > 1\,\text{kHz} \end{cases} \tag{5}$$

The smoothed signals are down-sampled for decreased computation time by a decimation factor of 24 (temporal resolution: 0,5 ms, i.e. sampling rate: 2 kHz).

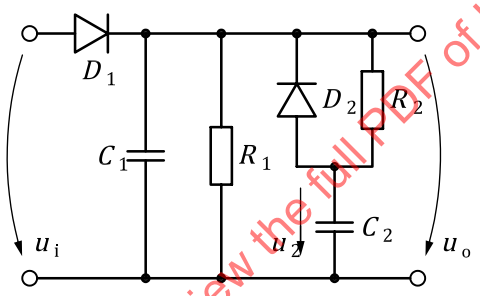
The calculated one-third-octave-band level $L_T$ of the time-varying sound are processed in the same way as for stationary sounds up to the calculation of the core loudness values. Block $N'$ includes the

transformation from the sound level into the specific loudness as described in the method for stationary sounds.

The nonlinear temporal decay of the hearing system is simulated in block NL. The drop after short signals is steeper than the one following longer signals.

For the signal processing carried out in block NL, Figure 7 shows an example of a network simulation. A calculation program (in C) is given in A.4 starting with '**// BLOCK NL**'. The calculation is performed internally at a sampling rate of 48 kHz (linear interpolation of the input signals, sampled at 2 kHz) in order to better simulate the times of conduction of the diodes.

For temporal weighting of total loudness, two first-order low-pass filters (time constants 3,5 ms and 70 ms) are applied to the sum of the specific loudness values in order to simulate the duration dependent behaviour of loudness perception for short impulses. A signal with a duration of 10 ms is perceived as about half as loud as a signal with a duration of 100 ms. The low-pass filtering is performed at a sampling rate of 48 kHz (linear interpolation of the specific loudness values, originally calculated every 0,5 ms) in order to better match the characteristic of the low-pass filter. The total loudness is the weighted sum of the low-pass filtered signals (factors 0,47, respectively 0,53) which is down-sampled to 500 Hz (temporal resolution: 2 ms).




**Key**

$\tau_{short} = R_1 \cdot C_1 = 5$ ms

$\tau_{long} = R_1 \cdot (C_1 + C_2) = 15$ ms

$\tau_{var} = R_2 \cdot C_2 = 75$ ms

NOTE See block NL in Figure 5.

**Figure 7 — Equivalent circuit diagram for the simulation of the temporal decay of the hearing system**

## 6.4 Guidance for determining the loudness of time-varying sounds

The sound pressure signals are detected by microphones. The measurements can either be done as single-channel measurements using a microphone, or as multi-channel measurements, for example by means of a head and torso simulator (see Annex D), in order to take direction-dependent effects into account. As a result, the calculation provides a temporal sequence of specific loudness spectra $N'(z,t)$ or summed loudness $N(t)$.

The time interval between the spectra or the individual values, respectively, shall be 2 ms. Extending this interval in the representation by skipping single spectra or values, respectively, can lead to inaccuracies in the further evaluation if the signals examined show a high temporal variability.

Due to the fact that the statistical mean of time-varying loudness leads, in general, to results that are too low in comparison to the evaluated loudness, the percentile loudness $N_5$ shall be given when stating the overall loudness perceived, for example for sound emissions. In addition, other percentile values of loudness may be stated.

$N_5$ is not suited for impulses because the percentile loudness $N_5$ strongly depends on the measurement time. At least the measurement time should be mentioned, and should be equal for all considered impulses.

NOTE Recent studies showed higher correlation between the overall loudness perceived and other single values than $N_5$, like the energy mean of loudness level versus time function[16] and the root mean cubed (cubic mean) loudness.[17] Note that the energy mean of the loudness level versus time function corresponds (for loudness values exceeding 1 sone) to an average of $N$ raised to the power of 1/lg(2), approximately 3,322 with a subsequent application of a power law with the exponent lg(2).

## 7 Reporting data

The following information shall be reported:

a) the sound under consideration;

b) a reference to this document, i.e. ISO 532-1;

c) the calculation method used [method for stationary sounds (Clause 5) or method for time-varying sounds (Clause 6)];

d) the sound field selected for the calculation [free (F) or diffuse (D)];

e) when using the method for stationary sounds:

   1) the type of input data [one-third-octave-band levels or time signals];

   2) the loudness in sones;

   3) if required, the specific loudness in sones/Bark;

   4) if required, the loudness level in phons;

f) when using the method for time-varying sounds:

   1) the loudness time function in sones;

   2) if required, the specific loudness time function in sones/Bark;

   3) if required, the loudness ($N_{max}$) and loudness ($N_5$) in sones.

NOTE Considerations on uncertainties are given in Annex E



# Annex A
(normative)

# Numerical details and program code for the calculation of loudness of stationary and time-varying sounds (test implementation)

## A.1 Background

An implementation for the calculation of loudness of stationary and time-varying sounds is given in this annex. It is intended to facilitate the comprehension of the procedure. It is also intended as a test implementation against which other implementations shall be tested. For compliance requirements, see 5.1 or 6.1.

NOTE A software package including the listing of the program code can be freely downloaded from http://standards.iso.org/iso/532/-1/ed-1/en

## A.2 Filter coefficients of one-third-octave-band filters

Each one-third-octave-band filter consists of three second order sections [Figure A.1 and Formula (A.1)]. Their filter coefficients are given in tables as the difference from a reference table (Tables A.1 and A.2). Each gain factor is applied once to the corresponding one-third-octave-band filter.

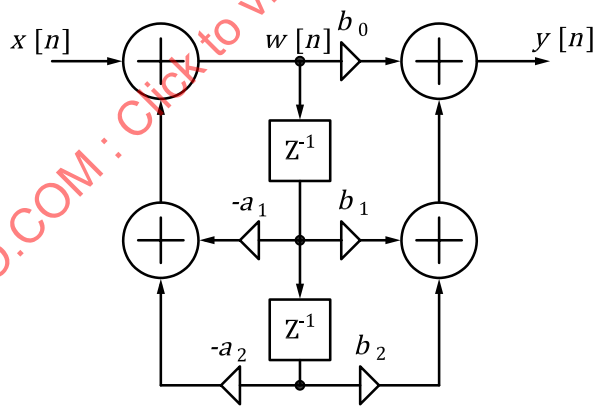



**Figure A.1 — Filter structure of a 2nd order filter**

$$
\begin{aligned}
w[n] &= a_0 x[n] - a_1 w[n-1] - a_2 w[n-2] \\
y[n] &= b_0 w[n] + b_1 w[n-1] + b_2 w[n-2] \\
a_i &= a_{ir} - a_{id}, \; i = 0,1,2 \\
b_i &= b_{ir} - b_{id}, \; i = 0,1,2
\end{aligned}
\tag{A.1}
$$

**Table A.1 — Filter coefficients of one-third-octave-band filters (reference table)**

| reference | $b_{0r}$ | $b_{1r}$ | $b_{2r}$ | $a_{0r}$ | $a_{1r}$ | $a_{2r}$ |
|---|---|---|---|---|---|---|
| 1st section | 1 | 2 | 1 | 1 | −2 | 1 |
| 2nd section | 1 | 0 | −1 | 1 | −2 | 1 |
| 3rd section | 1 | −2 | 1 | 1 | −2 | 1 |

**Table A.2 — Filter coefficients of one-third-octave-band filters (difference to reference table for 28 one-third-octave-band filters)**

| 25 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −6,70260e-4 | 6,59453e-4 |
| 2nd | 0 | 0 | 0 | 0 | −3,75071e-4 | 3,61926e-4 |
| 3rd | 0 | 0 | 0 | 0 | −3,06523e-4 | 2,97634e-4 |
| gain | 4,30764e-11 | | | | | |

| 31,5 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −8,47258e-4 | 8,30131e-4 |
| 2nd | 0 | 0 | 0 | 0 | −4,76448e-4 | 4,55616e-4 |
| 3rd | 0 | 0 | 0 | 0 | −3,88773e-4 | 3,74685e-4 |
| gain | 8,59340e-11 | | | | | |

| 40 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −1,07210e-3 | 1,04496e-3 |
| 2nd | 0 | 0 | 0 | 0 | −6,06567e-4 | 5,73553e-4 |
| 3rd | 0 | 0 | 0 | 0 | −4,94004e-4 | 4,71677e-4 |
| gain | 1,71424e-10 | | | | | |

| 50 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −1,35836e-3 | 1,31535e-3 |
| 2nd | 0 | 0 | 0 | 0 | −7,74327e-4 | 7,22007e-4 |
| 3rd | 0 | 0 | 0 | 0 | −6,29154e-4 | 5,93771e-4 |
| gain | 3,41944e-10 | | | | | |

| 63 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −1,72380e-3 | 1,65564e-3 |
| 2nd | 0 | 0 | 0 | 0 | −9,91780e-4 | 9,08866e-4 |
| 3rd | 0 | 0 | 0 | 0 | −8,03529e-4 | 7,47455e-4 |
| gain | 6,82035e-10 | | | | | |

| 80 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −2,19188e-3 | 2,08388e-3 |
| 2nd | 0 | 0 | 0 | 0 | −1,27545e-3 | 1,14406e-3 |
| 3rd | 0 | 0 | 0 | 0 | −1,02976e-3 | 9,40900e-4 |
| gain | 1,36026e-9 | | | | | |

| 100 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −2,79386e-3 | 2,62274e-3 |
| 2nd | 0 | 0 | 0 | 0 | −1,64828e-3 | 1,44006e-3 |
| 3rd | 0 | 0 | 0 | 0 | −1,32520e-3 | 1,18438e-3 |
| gain | 2,71261e-9 | | | | | |

| 125 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −3,57182e-3 | 3,30071e-3 |
| 2nd | 0 | 0 | 0 | 0 | −2,14252e-3 | 1,81258e-3 |
| 3rd | 0 | 0 | 0 | 0 | −1,71397e-3 | 1,49082e-3 |
| gain | 5,40870e-9 | | | | | |

| 160 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −4,58305e-3 | 4,15355e-3 |
| 2nd | 0 | 0 | 0 | 0 | −2,80413e-3 | 2,28135e-3 |
| 3rd | 0 | 0 | 0 | 0 | −2,23006e-3 | 1,87646e-3 |
| gain | 1,07826e-8 | | | | | |

| 200 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −5,90655e-3 | 5,22622e-3 |
| 2nd | 0 | 0 | 0 | 0 | −3,69947e-3 | 2,87118e-3 |
| 3rd | 0 | 0 | 0 | 0 | −2,92205e-3 | 2,36178e-3 |
| gain | 2,14910e-8 | | | | | |

| 250 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −7,65243e-3 | 6,57493e-3 |
| 2nd | 0 | 0 | 0 | 0 | −4,92540e-3 | 3,61318e-3 |
| 3rd | 0 | 0 | 0 | 0 | −3,86007e-3 | 2,97240e-3 |
| gain | 4,28228e-8 | | | | | |

| 315 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −1,00023e-2 | 8,29610e-3 |
| 2nd | 0 | 0 | 0 | 0 | −6,63788e-3 | 4,55999e-3 |
| 3rd | 0 | 0 | 0 | 0 | −5,15982e-3 | 3,75306e-3 |
| gain | 8,54316e-8 | | | | | |

| 400 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −1,31230e-2 | 1,04220e-2 |
| 2nd | 0 | 0 | 0 | 0 | −9,02274e-3 | 5,73132e-3 |
| 3rd | 0 | 0 | 0 | 0 | −6,94543e-3 | 4,71734e-3 |
| gain | 1,70009e-7 | | | | | |

| 500 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −1,73693e-2 | 1,30947e-2 |
| 2nd | 0 | 0 | 0 | 0 | −1,24176e-2 | 7,20526e-3 |
| 3rd | 0 | 0 | 0 | 0 | −9,46002e-3 | 5,93145e-3 |
| gain | 3,38215e-7 | | | | | |

| 630 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −2,31934e-2 | 1,64308e-2 |
| 2nd | 0 | 0 | 0 | 0 | −1,73009e-2 | 9,04761e-3 |
| 3rd | 0 | 0 | 0 | 0 | −1,30358e-2 | 7,44926e-3 |
| gain | 6,71990e-7 | | | | | |

| 800 Hz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −3,13292e-2 | 2,06370e-2 |
| 2nd | 0 | 0 | 0 | 0 | −2,44342e-2 | 1,13731e-2 |
| 3rd | 0 | 0 | 0 | 0 | −1,82108e-2 | 9,36778e-3 |
| gain | 1,33531e-6 | | | | | |

| 1 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −4,28261e-2 | 2,59325e-2 |
| 2nd | 0 | 0 | 0 | 0 | −3,49619e-2 | 1,43046e-2 |
| 3rd | 0 | 0 | 0 | 0 | −2,57855e-2 | 1,17912e-2 |
| gain | 2,65172e-6 | | | | | |

| 1,25 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −5,91733e-2 | 3,25054e-2 |
| 2nd | 0 | 0 | 0 | 0 | −5,06072e-2 | 1,79513e-2 |
| 3rd | 0 | 0 | 0 | 0 | −3,69401e-2 | 1,48094e-2 |
| gain | 5,25477e-6 | | | | | |

| 1,6 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −8,26348e-2 | 4,05894e-2 |
| 2nd | 0 | 0 | 0 | 0 | −7,40348e-2 | 2,24476e-2 |
| 3rd | 0 | 0 | 0 | 0 | −5,34977e-2 | 1,85371e-2 |
| gain | 1,03780e-5 | | | | | |

| 2 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | −1,17018e-1 | 5,08116e-2 |
| 2nd | 0 | 0 | 0 | 0 | −1,09516e-1 | 2,81387e-2 |
| 3rd | 0 | 0 | 0 | 0 | −7,85097e-2 | 2,32872e-2 |
| gain | 2,04870e-5 | | | | | |

| 2,5 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | –1,67714e-1 | 6,37872e-2 |
| 2nd | 0 | 0 | 0 | 0 | –1,63378e-1 | 3,53729e-2 |
| 3rd | 0 | 0 | 0 | 0 | –1,16419e-1 | 2,93723e-2 |
| gain | 4,05198e-5 | | | | | |

| 3,15 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | –2,42528e-1 | 7,98576e-2 |
| 2nd | 0 | 0 | 0 | 0 | –2,45161e-1 | 4,43370e-2 |
| 3rd | 0 | 0 | 0 | 0 | –1,73972e-1 | 3,70015e-2 |
| gain | 7,97914e-5 | | | | | |

| 4 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | –3,53142e-1 | 9,96330e-2 |
| 2nd | 0 | 0 | 0 | 0 | –3,69163e-1 | 5,53535e-2 |
| 3rd | 0 | 0 | 0 | 0 | –2,61399e-1 | 4,65428e-2 |
| gain | 1,56511e-4 | | | | | |

| 5 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | –5,16316e-1 | 1,24177e-1 |
| 2nd | 0 | 0 | 0 | 0 | –5,55473e-1 | 6,89403e-2 |
| 3rd | 0 | 0 | 0 | 0 | –3,93998e-1 | 5,86715e-2 |
| gain | 3,04954e-4 | | | | | |

| 6,3 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | –7,56635e-1 | 1,55023e-1 |
| 2nd | 0 | 0 | 0 | 0 | –8,34281e-1 | 8,58123e-2 |
| 3rd | 0 | 0 | 0 | 0 | –5,94547e-1 | 7,43960e-2 |
| gain | 5,99157e-4 | | | | | |

| 8 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | –1,10165e0 | 1,91713e-1 |
| 2nd | 0 | 0 | 0 | 0 | –1,23939e0 | 1,05243e-1 |
| 3rd | 0 | 0 | 0 | 0 | –8,91666e-1 | 9,40354e-2 |
| gain | 1,16544e-3 | | | | | |

| 10 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 | –1,58477e0 | 2,39049e-1 |
| 2nd | 0 | 0 | 0 | 0 | –1,80505e0 | 1,28794e-1 |
| 3rd | 0 | 0 | 0 | 0 | –1,32500e0 | 1,21333e-1 |
| gain | 2,27488e-3 | | | | | |

| 12,5 kHz | $b_{0d}$ | $b_{1d}$ | $b_{2d}$ | $a_{0d}$ | $a_{1d}$ | $a_{2d}$ |
|---|---|---|---|---|---|---|
| **1st** | 0 | 0 | 0 | 0 | –2,50630e0 | 1,42308e-1 |
| **2nd** | 0 | 0 | 0 | 0 | –2,19464e0 | 2,76470e-1 |
| **3rd** | 0 | 0 | 0 | 0 | –1,90231e0 | 1,47304e-1 |
| **gain** | 3,91006e-3 | | | | | |

## A.3 Details of the Zwicker method

This annex gives all remaining details which are necessary to independently evaluate this loudness. In addition, the program code given in A.4 allows for independent implementations of the Zwicker method in accordance with DIN 45631:1991.

While the graphical approach used to describe the method in this document may be very valuable for promoting awareness of the basic hearing effects and in interpreting the results, concrete calculations require numerical values. Therefore, applying the approach to one-third-octave-band spectral data essentially means using numerical tables. These values are given in the context of the program code and for reasons of clarity also in the tables below.

However, because the low frequency corrections described in this document mean a modification to the previous method B of ISO 532:1975, these corrections are explicitly specified in detail in Table A.3. These corrections strictly follow DIN 45631:1991 where they were introduced in 1991.

As described in 5.3, step 1, the fact that the hearing system is less sensitive for frequencies below 300 Hz shall be taken into account. This is accomplished by means of weightings listed in Table A.3. Eight level ranges with weighted one-third-octave-band levels between ≤ 45 dB and ≤ 120 dB are considered for frequencies between 25 Hz and 250 Hz.

Table A.4 contains the level corrections in accordance with the ear's transmission characteristics $a_0$ in decibels that shall be subtracted from the approximated critical band levels $L_{CB'}$. In the case of a diffuse sound field the level difference $\Delta L_{DF}$ (Table A.5) shall be added. If the resulting level exceeds the critical band level at the threshold in quiet, $L_{TQ}$ (Table A.6), then the corrections caused by critical bandwidth approximations, $\Delta L_{CB}$ (Table A.7) shall be subtracted and the core loudness values $N'_C$ are calculated according to Formula (A.2):

$$N'_C = 0{,}0635 \times 10^{0{,}025 L_{TQ}} \left( \left( 1 - s + s \times 10^{0{,}1 \left( L_{CB'} - L_{TQ} \right)} \right)^{0{,}25} - 1 \right) \tag{A.2}$$

with the threshold factor $s = 0{,}25$.

Table A.8 shows the mapping of centre frequencies $f_T$ of one-third-octave-band filters to Bark and Table A.9 the steepness of upper slopes for the specific loudness in dependence of the value range of the specific loudness and critical band number. The calculation of the specific loudness is an iterative process starting from 0,1 Bark to 24 Bark in steps of 0,1 Bark, which is illustrated in Figure 1 (right part of the block diagram). Total loudness is achieved by the integration of specific loudness values.

**Table A.3 — Weighting of one-third-octave-band sound pressure levels, $L_T$, for centre frequencies, $f_T$, below 300 Hz; corrections, $\Delta L$, in decibels**

| Range | $L_T + \Delta L$ | $f_T$, Hz | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **25** | **32** | **40** | **50** | **63** | **80** | **100** | **125** | **160** | **200** | **250** |
| **I** | ≤ 45 dB | –32 | –24 | –16 | –10 | –5 | 0 | –7 | –3 | 0 | –2 | 0 |
| **II** | ≤ 55 dB | –29 | –22 | –15 | –10 | –4 | 0 | –7 | –2 | 0 | –2 | 0 |
| **III** | ≤ 65 dB | –27 | –19 | –14 | –9 | –4 | 0 | –6 | –2 | 0 | –2 | 0 |
| **IV** | ≤ 71 dB | –25 | –17 | –12 | –9 | –3 | 0 | –5 | –2 | 0 | –2 | 0 |
| **V** | ≤ 80 dB | –23 | –16 | –11 | –7 | –3 | 0 | –4 | –1 | 0 | –1 | 0 |

**Table A.3** *(continued)*

| Range | $L_T + \Delta L$ | $f_T$, Hz | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **25** | **32** | **40** | **50** | **63** | **80** | **100** | **125** | **160** | **200** | **250** |
| **VI** | ≤ 90 dB | −20 | −14 | −10 | −6 | −3 | 0 | −4 | −1 | 0 | −1 | 0 |
| **VII** | ≤ 100 dB | −18 | −12 | −9 | −6 | −2 | 0 | −3 | −1 | 0 | −1 | 0 |
| **VIII** | ≤ 120 dB | −15 | −10 | −8 | −4 | −2 | 0 | −3 | −1 | 0 | −1 | 0 |

In order to illustrate the use of Table A.3, the following example is given.

EXAMPLE For the one-third-octave band with $f_T$ = 40 Hz the one-third-octave-band level is $L_T$ = 78 dB. The evaluation is:

in the range I: $L_T + \Delta L$ = (78 - 16) dB = 62 dB > 45 dB

in the range II: $L_T + \Delta L$ = (78 - 15) dB = 63 dB > 55 dB

in the range III: $L_T + \Delta L$ = (78 - 14) dB = 64 dB < 65 dB

Therefore, the weighted one-third-octave-band level amounts to $L_{T,W}$ = 64 dB.

**Table A.4 — Level correction in accordance with the ear's transmission characteristics, $a_0$, in decibels, for centre frequencies, $f_T$**

| $f_T$ Hz | $a_0$ dB |
|---|---|
| 25,0 | 0,0 |
| 31,5 | |
| 40,0 | |
| 50,0 | |
| 63,0 | |
| 80,0 | |
| 100,0 | 0,0 |
| 125,0 | |
| 160,0 | |
| 200,0 | 0,0 |
| 250,0 | |
| 315,0 | 0,0 |
| 400,0 | 0,0 |
| 500,0 | 0,0 |
| 630,0 | 0,0 |
| 800,0 | 0,0 |
| 1 000,0 | 0,0 |
| 1 250,0 | 0,0 |
| 1 600,0 | −0,5 |
| 2 000,0 | −1,6 |
| 2 500,0 | −3,2 |
| 3 150,0 | −5,4 |
| 4 000,0 | −5,6 |
| 5 000,0 | −4,0 |
| 6 300,0 | −1,5 |

**Table A.4** *(continued)*

| $f_T$<br>Hz | $a_0$<br>dB |
|---|---|
| 8 000,0 | 2,0 |
| 10 000,0 | 5,0 |
| 12 500,0 | 12,0 |

**Table A.5 — Level difference for diffuse sound field, $\Delta L_{DF}$, in decibels, for centre frequencies, $f_T$**

| $f_T$<br>Hz | $\Delta L_{DF}$<br>dB |
|---|---|
| 25,0 | 0,0 |
| 31,5 | |
| 40,0 | |
| 50,0 | |
| 63,0 | |
| 80,0 | |
| 100,0 | 0,0 |
| 125,0 | |
| 160,0 | |
| 200,0 | 0,5 |
| 250,0 | |
| 315,0 | 0,9 |
| 400,0 | 1,2 |
| 500,0 | 1,6 |
| 630,0 | 2,3 |
| 800,0 | 2,8 |
| 1 000,0 | 3,0 |
| 1 250,0 | 2,0 |
| 1 600,0 | 0,0 |
| 2 000,0 | −1,4 |
| 2 500,0 | −2,0 |
| 3 150,0 | −1,9 |
| 4 000,0 | −1,0 |
| 5 000,0 | 0,5 |
| 6 300,0 | 3,0 |
| 8 000,0 | 4,0 |
| 10 000,0 | 4,3 |
| 12 500,0 | 4,0 |

**Table A.6 — Critical band level at the threshold in quiet, $L_{TQ}$, in decibels, without consideration of the ear's transmission characteristics, $a_0$, for centre frequencies, $f_T$**

| $f_T$<br>Hz | $L_{TQ}$<br>dB |
|---|---|
| 25,0 | 30 |
| 31,5 | |
| 40,0 | |
| 50,0 | |
| 63,0 | |
| 80,0 | |
| 100,0 | 18 |
| 125,0 | |
| 160,0 | |
| 200,0 | 12 |
| 250,0 | |
| 315,0 | 8 |
| 400,0 | 7 |
| 500,0 | 6 |
| 630,0 | 5 |
| 800,0 | 4 |
| 1 000,0 | 3 |
| 1 250,0 | 3 |
| 1 600,0 | 3 |
| 2 000,0 | 3 |
| 2 500,0 | 3 |
| 3 150,0 | 3 |
| 4 000,0 | 3 |
| 5 000,0 | 3 |
| 6 300,0 | 3 |
| 8 000,0 | 3 |
| 10 000,0 | 3 |
| 12 500,0 | 3 |

**Table A.7 — Adaptation of the one-third-octave-band levels to the corresponding critical band levels due to different bandwidth for centre frequencies, $f_T$; corrections, $\Delta L_{CB}$, in decibels**

| $f_T$<br>Hz | $\Delta L_{CB}$<br>dB |
|---|---|
| 25,0 | −0,25 |
| 31,5 | |
| 40,0 | |
| 50,0 | |
| 63,0 | |
| 80,0 | |
| 100,0 | −0,60 |
| 125,0 | |
| 160,0 | |

**Table A.7** *(continued)*

| $f_T$<br>Hz | $\Delta L_{CB}$<br>dB |
|---|---|
| 200,0 | −0,80 |
| 250,0 | |
| 315,0 | −0,80 |
| 400,0 | −0,50 |
| 500,0 | 0,00 |
| 630,0 | 0,50 |
| 800,0 | 1,10 |
| 1 000,0 | 1,50 |
| 1 250,0 | 1,70 |
| 1 600,0 | 1,80 |
| 2 000,0 | 1,80 |
| 2 500,0 | 1,70 |
| 3 150,0 | 1,60 |
| 4 000,0 | 1,40 |
| 5 000,0 | 1,20 |
| 6 300,0 | 0,80 |
| 8 000,0 | 0,50 |
| 10 000,0 | 0,00 |
| 12 500,0 | −0,50 |

**Table A.8 — Mapping of centre frequencies, $f_T$, of one-third-octave-band filters to Bark**

| $f_T$<br>Hz | **Approximated critical band number**<br>dB | **Critical band number**<br>Bark |
|---|---|---|
| 25,0 | 1 | 0,9 |
| 31,5 | | |
| 40,0 | | |
| 50,0 | | |
| 63,0 | | |
| 80,0 | | |
| 100,0 | 2 | 1,8 |
| 125,0 | | |
| 160,0 | | |
| 200,0 | 3 | 2,8 |
| 250,0 | | |
| 315,0 | 4 | 3,5 |
| 400,0 | 5 | 4,4 |
| 500,0 | 6 | 5,4 |
| 630,0 | 7 | 6,6 |
| 800,0 | 8 | 7,9 |
| 1 000,0 | 9 | 9,2 |
| NOTE Approximated critical bands 1–20 contain their individual core loudness. Approximated critical band 21 is empty, only slope loudness is taken into account. | | |

**Table A.8** *(continued)*

| $f_T$<br>Hz | Approximated critical band number<br>dB | Critical band number<br>Bark |
|---|---|---|
| 1 250,0 | 10 | 10,6 |
| 1 600,0 | 11 | 12,3 |
| 2 000,0 | 12 | 13,8 |
| 2 500,0 | 13 | 15,2 |
| 3 150,0 | 14 | 16,7 |
| 4 000,0 | 15 | 18,1 |
| 5 000,0 | 16 | 19,3 |
| 6 300,0 | 17 | 20,6 |
| 8 000,0 | 18 | 21,8 |
| 10 000,0 | 19 | 22,7 |
| 12 500,0 | 20 | 23,6 |
| — | 21 | 24,0 |

NOTE Approximated critical bands 1–20 contain their individual core loudness. Approximated critical band 21 is empty, only slope loudness is taken into account.

**Table A.9 — Steepness of upper slopes for the specific loudness in dependence of the value range of the specific loudness and critical band number**

| Index | Specific loudness | Critical band | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | > 7 |
| **1** | $0{,}000 \leq N' < 0{,}035$ | 0,06 | 0,05 | 0,03 | 0,02 | 0,02 | 0,02 | 0,02 | 0,02 |
| **2** | $0{,}035 \leq N' < 0{,}100$ | 0,09 | 0,08 | 0,07 | 0,06 | 0,06 | 0,06 | 0,06 | 0,05 |
| **3** | $0{,}100 \leq N' < 0{,}150$ | 0,12 | 0,11 | 0,10 | 0,08 | 0,08 | 0,08 | 0,08 | 0,08 |
| **4** | $0{,}150 \leq N' < 0{,}220$ | 0,16 | 0,15 | 0,14 | 0,12 | 0,11 | 0,11 | 0,11 | 0,11 |
| **5** | $0{,}220 \leq N' < 0{,}300$ | 0,27 | 0,21 | 0,20 | 0,18 | 0,17 | 0,17 | 0,17 | 0,17 |
| **6** | $0{,}300 \leq N' < 0{,}420$ | 0,40 | 0,33 | 0,26 | 0,24 | 0,22 | 0,22 | 0,22 | 0,22 |
| **7** | $0{,}420 \leq N' < 0{,}820$ | 0,59 | 0,53 | 0,51 | 0,50 | 0,42 | 0,42 | 0,42 | 0,42 |
| **8** | $0{,}820 \leq N' < 1{,}360$ | 0,72 | 0,67 | 0,64 | 0,63 | 0,62 | 0,62 | 0,62 | 0,62 |
| **9** | $1{,}360 \leq N' < 2{,}130$ | 1,50 | 1,20 | 0,94 | 0,86 | 0,82 | 0,82 | 0,82 | 0,82 |
| **10** | $2{,}130 \leq N' < 3{,}100$ | 1,95 | 1,45 | 1,30 | 1,15 | 1,10 | 1,10 | 1,10 | 1,10 |
| **11** | $3{,}100 \leq N' < 4{,}400$ | 2,40 | 1,70 | 1,50 | 1,35 | 1,30 | 1,30 | 1,30 | 1,30 |
| **12** | $4{,}400 \leq N' < 6{,}100$ | 2,90 | 2,30 | 2,10 | 1,90 | 1,80 | 1,70 | 1,70 | 1,70 |
| **13** | $6{,}100 \leq N' < 9{,}000$ | 3,70 | 3,00 | 2,80 | 2,35 | 2,20 | 2,20 | 2,20 | 2,20 |
| **14** | $9{,}000 \leq N' < 11{,}50$ | 4,50 | 3,80 | 3,60 | 3,20 | 2,90 | 2,70 | 2,70 | 2,70 |
| **15** | $11{,}50 \leq N' < 15{,}10$ | 6,20 | 5,40 | 4,60 | 4,00 | 3,50 | 3,20 | 3,20 | 3,20 |
| **16** | $15{,}10 \leq N' < 18{,}00$ | 7,80 | 6,70 | 5,60 | 4,90 | 4,40 | 3,90 | 3,90 | 3,90 |
| **17** | $18{,}00 \leq N' < 21{,}50$ | 9,00 | 7,50 | 6,00 | 5,10 | 4,50 | 4,50 | 4,50 | 4,50 |
| **18** | $21{,}50 \leq N'$ | 13,0 | 8,20 | 6,30 | 5,50 | 5,50 | 5,50 | 5,50 | 5,50 |



## A.4 Program for the calculation of loudness according to the method for stationary sounds and the method for time-varying sounds

A software package including the listing of the program code can be freely downloaded from the following URL: http://standards.iso.org/iso/532/-1/ed-1/en

```
/*************************************************************************/
/* Loudness calculation according to ISO 532-1, */
/*  methods for stationary and time-varying sounds                       */
/*  This implementation requires signals sampled at 48 kHz!              */
/*************************************************************************/

#include <math.h>     /* for sqrt, exp, pow, floor, log10               */
#include <stdlib.h>   /* for memory allocation                           */

/*  Loudness calculation methods                                         */
enum _LoudnessMethod
{
    LoudnessMethodStationary  = 0,
    LoudnessMethodTimeVarying = 1,
};

/* Sound field types                                                     */
enum _SoundField
{
    SoundFieldFree    = 0,
    SoundFieldDiffuse = 1
};

/* Error codes                                                           */
enum _LoudnessErrorCodes
{
LoudnessErrorOutputVectorTooSmall = -1,
LoudnessErrorMemoryAllocFailed = -2,
LoudnessErrorUnsupportedMethod = -3,
LoudnessErrorSignalTooShort = -4
};

/*  Inputfile data structure                                             */
struct InputData
{
    int     NumSamples;
    double  SampleRate;
    double  *pData;
};

/*  Reference value for intensity calculation                            */
#define I_REF           4e-10

/*  Sampling rate to which third-octave-levels are downsampled           */
#define SR_LEVEL        2000

/*  Number of third octave bands                                         */
#define N_LEVEL_BANDS   28

/*  Number of Bark bands (resolution 0.1 Bark)                           */
#define N_BARK_BANDS    240

/*  Number of lower critical bands consisting of more than one           */
/*  third octave band                                                    */
#define N_LCBS          3

/*  Number of third octave bands for N_LCBS lower bands                  */
#define N_LCB_BANDS     11
/*  Number of core loudness values                                       */
#define N_CORE_LOUDN    21

/*  Number of level ranges for consideration of equal loudness contours */
#define N_RAP_RANGES    8

/*  Number of loudness and critical band ranges for calculating          */
/*  steepness of upper slopes in the specific loudness -                 */
/*  critical-band-rate pattern                                           */
#define N_RNS_RANGES    18
#define N_CB_RANGES     8

/*  Constants for third octave filters                                   */
```

```
#define N_FILTER_STAGES 3
#define N_FILTER_COEFS  6

/*  Factors for virtual upsampling/inner iterations                    */
#define NL_ITER         24
#define LP_ITER         24

/*  Time constants for non-linear temporal decay                       */
#define TSHORT          0.005
#define TLONG           0.015
#define TVAR            0.075

////////////////////////////////////////////////////////////////////////
//  BLOCK   Memory management
////////////////////////////////////////////////////////////////////////

static void freeRaggedArray(double** raggedArray, int numArrays)
{
    int i;
    for (i = 0; i < numArrays; i++)
    {
        if (raggedArray[i] != NULL)
        {
            free(raggedArray[i]);
        }
    }
}

static int callocRaggedArray(double** raggedArray, int numArrays, int lengthArray)
{
    int i;
    for (i = 0; i < numArrays; i++)
    {
        raggedArray[i] = (double*)calloc(lengthArray, sizeof(double));
        if (raggedArray[i] == NULL)
        {
            freeRaggedArray(raggedArray, i - 1);
            return LoudnessErrorMemoryAllocFailed;
        }
    }
    return 0;
}
////////////////////////////////////////////////////////////////////////
//  BLOCK   NL
////////////////////////////////////////////////////////////////////////

/* Inputfile structure                                                 */
struct NlLpData
{
    double B[6]          /*  coefficients    */
       , Uolast        /*  Uo, last sample */
       , U2last;       /*  U2, last sample */
};

/*  Initializes constants B and states of capacitors C1 and C2,
    1/delta_t = sampling rate                                          */
struct NlLpData f_init_nl_lp(double SampleRate)
{
    double Lambda1, Lambda2, P, Q, Den, E1, E2, DeltaT;
    double Tvar     = TVAR;
    double Tshort   = TSHORT;
    double Tlong    = TLONG;

    struct NlLpData NlLp;

    DeltaT      = 1 / SampleRate;
    P           = (Tvar + Tlong) / (Tvar*Tshort);
    Q           = 1/(Tshort*Tvar);
    Lambda1     =-P/2 + sqrt(P*P/4 - Q);
    Lambda2     =-P/2 - sqrt(P*P/4 - Q);
    Den         = Tvar * (Lambda1 - Lambda2);
```

```
    E1          = exp(Lambda1 * DeltaT);
    E2          = exp(Lambda2 * DeltaT);

    NlLp.B[0]   = (E1 - E2) / Den;
    NlLp.B[1]   =((Tvar * Lambda2 + 1) * E1 - (Tvar * Lambda1 + 1) * E2) / Den;
    NlLp.B[2]   =((Tvar * Lambda1 + 1) * E1 - (Tvar * Lambda2 +1) * E2) / Den;
    NlLp.B[3]   = (Tvar * Lambda1+1) * (Tvar * Lambda2 + 1) * (E1-E2) / Den;
    NlLp.B[4]   = exp(-DeltaT / Tlong);
    NlLp.B[5]   = exp(-DeltaT / Tvar);

    /*  At beginning capacitors C1 and C2 are discharged    */
    NlLp.UoLast = 0;
    NlLp.U2Last = 0;

    return NlLp;
}

/*  NL: Calculates Uo(t) from Ui(t) using UoLast and U2Last             */
double f_nl_lp(double Ui, struct NlLpData *pNlLp)
{
    double Uo, U2;

    if (Ui < pNlLp->UoLast)                 /*  case 1          */
        if (pNlLp->UoLast > pNlLp->U2Last)  /*  case 1.1        */
        {
            U2 = pNlLp->UoLast*pNlLp->B[0] - pNlLp->U2Last*pNlLp->B[1];
            Uo = pNlLp->UoLast*pNlLp->B[2] - pNlLp->U2Last*pNlLp->B[3];
            if (Uo < Ui)                    /*  Uo can't become */
                Uo = Ui;                    /*  lower than Ui   */
            if (U2 > Uo)                    /*  U2 can't become */
                U2 = Uo;                    /*  higher than Uo  */
        }
        else                                /*  Case 1.2        */
        {
            Uo = pNlLp->UoLast*pNlLp->B[4];
            if (Uo < Ui)                    /*  Uo can't become */
                Uo = Ui;                    /*  lower than Ui   */
            U2 = Uo;
        }
    else
    {
        if (fabs(Ui - pNlLp->UoLast)<1e-5)  /*  Case 2          */
        {
            Uo = Ui;
            if (Uo > pNlLp->U2Last)         /*  Case 2.1        */
                U2 = (pNlLp->U2Last - Ui)*pNlLp->B[5] +  Ui;
            else                            /*  Case 2.2        */
                U2 = Ui;
        }
        else                                /*  Case 3          */
        {
            Uo = Ui;
            U2 = (pNlLp->U2Last - Ui)*pNlLp->B[5] + Ui;
        }
    }

    pNlLp->UoLast = Uo;                     /*  Preparation for */
    pNlLp->U2Last = U2;                     /*  next step       */

    return(Uo);
}

/*  Non-linear temporal decay, block NL, uses inner iterations/linear
    interpolation for increased precision                               */
void f_nl(double **CoreLoudness, double SampleRate, int NumSamples)
{
    int     IdxCL, IdxTime, IdxI;
    double  *pCoreL, NextInput, Delta, Ui, Uo;
    struct  NlLpData    NlLp = f_init_nl_lp(SampleRate * NL_ITER);

    for (IdxCL = 0; IdxCL < N_CORE_LOUDN; IdxCL++)
```

```
    {
        pCoreL = CoreLoudness[IdxCL];

        /*  instead of calling f_init_nl_lp */
        NlLp.UoLast = 0;
        NlLp.U2Last = 0;

        for (IdxTime = 0; IdxTime < NumSamples-1; IdxTime++)
        {
            /*  next sample                                                 */
            NextInput = *(pCoreL + 1);
            /*  interpolation steps between current and next sample     */
            Delta = (NextInput - *pCoreL) / (double)NL_ITER;

            Ui = *pCoreL;
            *pCoreL = f_nl_lp(Ui, &NlLp);
            Ui += Delta;




            /*  inner iterations                                            */
            for (IdxI = 1; IdxI < NL_ITER; IdxI++)
            {
                Uo = f_nl_lp(Ui, &NlLp);
                Ui += Delta;
            }
            pCoreL++;
        }
        *pCoreL = f_nl_lp(*pCoreL, &NlLp);
    }
}

////////////////////////////////////////////////////////////////////////
//  BLOCK   Third-octave filtering
////////////////////////////////////////////////////////////////////////

/*  Reference filter coefficient table  */
/*  b0r, b1r, b2r, a0r, a1r, a2r        */
static double ThirdOctaveFilterRef[N_FILTER_STAGES][N_FILTER_COEFS]=
{ {1, 2, 1, 1, -2, 1},
{1, 0,-1, 1, -2, 1},
{1,-2, 1, 1, -2, 1}
};

/*  Filter tables of difference from reference table for third octave
    filter for sampling rate = 48kHz                                    */
/*  b0d, b1d, b2d, a0d, a1d, a2d                                        */
static double ThirdOctaveFilters[N_LEVEL_BANDS][N_FILTER_STAGES][N_FILTER_COEFS]=
{   { {0,0,0,0,-6.70260e-004,6.59453e-004},
      {0,0,0,0,-3.75071e-004,3.61926e-004},
      {0,0,0,0,-3.06523e-004,2.97634e-004} },

    { {0,0,0,0,-8.47258e-004,8.30131e-004},
      {0,0,0,0,-4.76448e-004,4.55616e-004},
      {0,0,0,0,-3.88773e-004,3.74685e-004} },

    { {0,0,0,0,-1.07210e-003,1.04496e-003},
      {0,0,0,0,-6.06567e-004,5.73553e-004},
      {0,0,0,0,-4.94004e-004,4.71677e-004} },

    { {0,0,0,0,-1.35836e-003,1.31535e-003},
      {0,0,0,0,-7.74327e-004,7.22007e-004},
      {0,0,0,0,-6.29154e-004,5.93771e-004} },

    { {0,0,0,0,-1.72380e-003,1.65564e-003},
      {0,0,0,0,-9.91780e-004,9.08866e-004},
      {0,0,0,0,-8.03529e-004,7.47455e-004} },

    { {0,0,0,0,-2.19188e-003,2.08388e-003},
```

```
    {0,0,0,0,-1.27545e-003,1.14406e-003},
    {0,0,0,0,-1.02976e-003,9.40900e-004} },

  { {0,0,0,0,-2.79386e-003,2.62274e-003},
    {0,0,0,0,-1.64828e-003,1.44006e-003},
    {0,0,0,0,-1.32520e-003,1.18438e-003} },

  { {0,0,0,0,-3.57182e-003,3.30071e-003},
    {0,0,0,0,-2.14252e-003,1.81258e-003},
    {0,0,0,0,-1.71397e-003,1.49082e-003} },

  { {0,0,0,0,-4.58305e-003,4.15355e-003},
    {0,0,0,0,-2.80413e-003,2.28135e-003},
    {0,0,0,0,-2.23006e-003,1.87646e-003} },

  { {0,0,0,0,-5.90655e-003,5.22622e-003},
    {0,0,0,0,-3.69947e-003,2.87118e-003},
    {0,0,0,0,-2.92205e-003,2.36178e-003} },

  { {0,0,0,0,-7.65243e-003,6.57493e-003},
    {0,0,0,0,-4.92540e-003,3.61318e-003},
    {0,0,0,0,-3.86007e-003,2.97240e-003} },

  { {0,0,0,0,-1.00023e-002,8.29610e-003},
    {0,0,0,0,-6.63788e-003,4.55999e-003},
    {0,0,0,0,-5.15982e-003,3.75306e-003} },

  { {0,0,0,0,-1.31230e-002,1.04220e-002},
    {0,0,0,0,-9.02274e-003,5.73132e-003},
    {0,0,0,0,-6.94543e-003,4.71734e-003} },

  { {0,0,0,0,-1.73693e-002,1.30947e-002},
    {0,0,0,0,-1.24176e-002,7.20526e-003},
    {0,0,0,0,-9.46002e-003,5.93145e-003} },

  { {0,0,0,0,-2.31934e-002,1.64308e-002},
    {0,0,0,0,-1.73009e-002,9.04761e-003},
    {0,0,0,0,-1.30358e-002,7.44926e-003} },

  { {0,0,0,0,-3.13292e-002,2.06370e-002},
    {0,0,0,0,-2.44342e-002,1.13731e-002},
    {0,0,0,0,-1.82108e-002,9.36778e-003} },

  { {0,0,0,0,-4.28261e-002,2.59325e-002},
    {0,0,0,0,-3.49619e-002,1.43046e-002},
    {0,0,0,0,-2.57855e-002,1.17912e-002} },

  { {0,0,0,0,-5.91733e-002,3.25054e-002},
    {0,0,0,0,-5.06072e-002,1.79513e-002},
    {0,0,0,0,-3.69401e-002,1.48094e-002} },

  { {0,0,0,0,-8.26348e-002,4.05894e-002},
    {0,0,0,0,-7.40348e-002,2.24476e-002},
    {0,0,0,0,-5.34977e-002,1.85371e-002} },

  { {0,0,0,0,-1.17018e-001,5.08116e-002},
    {0,0,0,0,-1.09516e-001,2.81387e-002},
    {0,0,0,0,-7.85097e-002,2.32872e-002} },

  { {0,0,0,0,-1.67714e-001,6.37872e-002},
    {0,0,0,0,-1.63378e-001,3.53729e-002},
    {0,0,0,0,-1.16419e-001,2.93723e-002} },

  { {0,0,0,0,-2.42528e-001,7.98576e-002},
    {0,0,0,0,-2.45161e-001,4.43370e-002},
    {0,0,0,0,-1.73972e-001,3.70015e-002} },

  { {0,0,0,0,-3.53142e-001,9.96330e-002},
    {0,0,0,0,-3.69163e-001,5.53535e-002},
    {0,0,0,0,-2.61399e-001,4.65428e-002} },
```

```
    { {0,0,0,0,-5.16316e-001,1.24177e-001},
      {0,0,0,0,-5.55473e-001,6.89403e-002},
      {0,0,0,0,-3.93998e-001,5.86715e-002} },

    { {0,0,0,0,-7.56635e-001,1.55023e-001},
      {0,0,0,0,-8.34281e-001,8.58123e-002},
      {0,0,0,0,-5.94547e-001,7.43960e-002} },

    { {0,0,0,0,-1.10165e+000,1.91713e-001},
      {0,0,0,0,-1.23939e+000,1.05243e-001},
      {0,0,0,0,-8.91666e-001,9.40354e-002} },

    { {0,0,0,0,-1.58477e+000,2.39049e-001},
      {0,0,0,0,-1.80505e+000,1.28794e-001},
      {0,0,0,0,-1.32500e+000,1.21333e-001} },

    { {0,0,0,0,-2.50630e+000,1.42308e-001},
      {0,0,0,0,-2.19464e+000,2.76470e-001},
      {0,0,0,0,-1.90231e+000,1.47304e-001} },
};

/*  Filter gains/scale values                                          */
static double FilterGain[N_LEVEL_BANDS][N_FILTER_STAGES]=
{   {4.30764e-011,1,1},
    {8.59340e-011,1,1},
    {1.71424e-010,1,1},
    {3.41944e-010,1,1},
    {6.82035e-010,1,1},
    {1.36026e-009,1,1},
    {2.71261e-009,1,1},
    {5.40870e-009,1,1},
    {1.07826e-008,1,1},
    {2.14910e-008,1,1},
    {4.28228e-008,1,1},
    {8.54316e-008,1,1},
    {1.70009e-007,1,1},
    {3.38215e-007,1,1},
    {6.71990e-007,1,1},
    {1.33531e-006,1,1},
    {2.65172e-006,1,1},
    {5.25477e-006,1,1},
    {1.03780e-005,1,1},
    {2.04870e-005,1,1},
    {4.05198e-005,1,1},
    {7.97914e-005,1,1},
    {1.56511e-004,1,1},
    {3.04954e-004,1,1},
    {5.99157e-004,1,1},
    {1.16544e-003,1,1},
    {2.27488e-003,1,1},
    {3.91006e-003,1,1}
};

/*  1st order low-pass                                                 */
void f_lowpass(double *pInput, double *pOutput, double Tau,
               double SampleRate, int NumSamples)
{
    int     IdxTime;
    double  A1, B0, Y1 = 0;
    double  *pX = pInput
       ,  *pY = pOutput;
A1 = exp(-1 / (SampleRate * Tau));
    B0 = 1 - A1;

    for (IdxTime = 0; IdxTime < NumSamples; IdxTime++)
    {
        *pY = B0 * *pX + A1 * Y1;
        Y1  = *pY;
        pX++;
        pY++;
    }
```

```
}

/*  2nd order filtering                                                 */
void f_filter_2ndOrder(double *pInput, double *pOutput, double *Coeffs,
                       int NumSamples, double Gain)
{
    int     IdxTime;
    double  *pX = pInput;
    double  *pY = pOutput;
    double  Wn0 = 0
       ,  Wn1 = 0
       ,  Wn2 = 0;

    for (IdxTime = 0; IdxTime < NumSamples; IdxTime++)
    {
        Wn0 = *pX*Gain - Coeffs[4]*Wn1 - Coeffs[5]*Wn2; /* coeffs[3]=1 */
        *pY = Coeffs[0]*Wn0 + Coeffs[1]*Wn1 + Coeffs[2]*Wn2;
        Wn2 = Wn1;
        Wn1 = Wn0;
        pX++;
        pY++;
    }
}

/*  Squaring and smoothing by three 1st order lowpass filters           */
int f_square_and_smooth(  double *pInput, double CenterFrequency,
                          double SampleRate, int NumSamples,
                          int Method, double TimeSkip)
{
    int     IdxTime, IdxFS, NumSkip;
    double  *pX = pInput, out;
    double  Tau;

    if (Method == LoudnessMethodTimeVarying)
    {
        /*  Frequency dependant time-constant                           */
        if (CenterFrequency <= 1000)
        {
            Tau = 2/(3*CenterFrequency);
        }
        else
        {
            Tau = 2/(3*1000.);
        }

        /*  Squaring                                                    */
        for (IdxTime = 0; IdxTime < NumSamples; IdxTime++)
        {
             *pX = pow(*pX,2);
             pX++;
        }
        /*  Three smoothing low-pass filters                            */
        for (IdxFS = 0; IdxFS < 3; IdxFS++)
        {
            f_lowpass (pInput,pInput,Tau,SampleRate,NumSamples);
        }
    }
    else
    {
        out = 0;
        NumSkip = (int)floor(TimeSkip * SampleRate);
        if (NumSkip >= NumSamples)
            return LoudnessErrorSignalTooShort;

        pX += NumSkip;
        for (IdxTime = NumSkip; IdxTime < NumSamples; IdxTime++)
        {
            out += pow(*pX,2);
            pX++;
        }
        *pInput = out / (NumSamples-NumSkip);
```

```
    }
    return 0;
}

/*  3rd octave filtering, squaring, smoothing, level calculation and
    downsampling by DecFactor to SR_LEVEL                                */

#define TINY_VALUE  1e-12

int f_calc_third_octave_levels(struct InputData *pSignal,
                               double **ThirdOctaveLevel,
                               int DecFactor, int Method,
                               double TimeSkip)
{
    short   IdxFB, IdxFS, IdxC;
    int     NumSamples, NumDecSamples;
    int     IdxTime, retval;
    double  Coeffs[N_FILTER_COEFS], Gain, CenterFrequency;
    double  *pInput, *pThirdOctaveLevel, *pOutput, *pSPL;

    NumSamples      = pSignal->NumSamples;
    NumDecSamples   = NumSamples/DecFactor;

    pOutput     = (double*)calloc(NumSamples,sizeof(double));
    if (pOutput == NULL)
    {
        return LoudnessErrorMemoryAllocFailed;
    }

    for (IdxFB = 0; IdxFB < N_LEVEL_BANDS; IdxFB++)     /*  All filters */
    {
        pInput  = pSignal->pData;

        /*  Filter stages                                               */
        for (IdxFS = 0; IdxFS < N_FILTER_STAGES; IdxFS++)
        {
            for (IdxC = 0; IdxC < N_FILTER_COEFS; IdxC++)
            {
                Coeffs[IdxC] =  ThirdOctaveFilterRef[IdxFS][IdxC] -
                                ThirdOctaveFilters[IdxFB][IdxFS][IdxC];
            }
            Gain    = FilterGain[IdxFB][IdxFS];

            /*  2nd order filtering                                     */
            f_filter_2ndOrder(pInput,pOutput,Coeffs,NumSamples,Gain);
            pInput  = pOutput;
        }
        /*  Calculate center frequency of filter                        */
        CenterFrequency = pow(10, ((double)IdxFB-16)/10.) * 1000;

        /*  Squaring and smoothing of filtered signal                   */
        retval = f_square_and_smooth(pInput,CenterFrequency,
            pSignal->SampleRate,NumSamples,Method,TimeSkip);
        if (retval < 0)
        {
            if (pOutput != NULL)
                free (pOutput);
            return retval;
        }

        pThirdOctaveLevel   = ThirdOctaveLevel[IdxFB];

        /*  SPL calculation                                              */
        pSPL    = pInput;

        for (IdxTime = 0; IdxTime < NumDecSamples; IdxTime++)
        {
            *pThirdOctaveLevel = 10*log10((*pSPL + TINY_VALUE) / I_REF);
            pSPL    += DecFactor;
            pThirdOctaveLevel++;
        }
```

```
    }
    if (pOutput != NULL)
        free (pOutput);
    return 0;
}

////////////////////////////////////////////////////////////////////////
//  BLOCK   Temporal weighting of loudness
////////////////////////////////////////////////////////////////////////

/*  1st order low-pass with linear interpolation of signal for
    increased precision                                                 */
void f_lowpass_intp(double *pInput, double *pOutput, double Tau,
                    double SampleRate, int NumSamples)
{
    int     IdxTime, IdxI;
    double  A1, B0, X0, Xd, Y1  = 0;
    double  *pX = pInput
       ,  *pY = pOutput;

A1 = exp(-1 / (SampleRate * LP_ITER * Tau));
    B0 = 1 - A1;

    for (IdxTime = 0; IdxTime < NumSamples; IdxTime++)
    {
        X0  = *pX;
        pX++;

        Y1  = B0 * X0 + A1 * Y1;
        *pY = Y1;
        pY++;
        /*  Linear interpolation steps between current and next sample */
        if (IdxTime < NumSamples - 1)
        {
            Xd  = (*pX - X0) / (double)LP_ITER;

            /*  Inner iterations/interpolation                          */
            for (IdxI = 1; IdxI < LP_ITER; IdxI++)
            {
                X0  += Xd;
                Y1  = B0 * X0 + A1 * Y1;
            }
        }
    }
}

/*  Two 1st order lowpasses (with interpolation) and weighted summation
    for duration dependent behavior of loudness perception              */
int f_temporal_weight_loudness(double *Loudness, double SampleRate,
                               int NumSamples)
{
    int     IdxTime;
    double  Tau;
    double  *pLoudness = Loudness
        , *pLoudness_1, *pLoudness_2, *pLoudness_t1, *pLoudness_t2;

    /*  Memory allocation                                               */
    pLoudness_t1 = (double*)calloc(NumSamples, sizeof(double));
    if (pLoudness_t1 == NULL)
    {
        return LoudnessErrorMemoryAllocFailed;
    }
    pLoudness_t2 = (double*)calloc(NumSamples, sizeof(double));
    if (pLoudness_t2 == NULL)
    {
        if (pLoudness_t1 != NULL)
            free (pLoudness_t1);

        return LoudnessErrorMemoryAllocFailed;
    }
```

```
    pLoudness_1 = pLoudness_t1;
    pLoudness_2 = pLoudness_t2;

    Tau = 3.5e-3;
    f_lowpass_intp(pLoudness,pLoudness_1,Tau,SampleRate,NumSamples);

    Tau = 70e-3;
    f_lowpass_intp(pLoudness,pLoudness_2,Tau,SampleRate,NumSamples);

    for (IdxTime = 0; IdxTime < NumSamples; IdxTime++)
    {
        *pLoudness = 0.47 * (*pLoudness_1) + 0.53 * (*pLoudness_2);
        pLoudness++;
        pLoudness_1++;
        pLoudness_2++;
    }

    /*  Memory deallocation                                                 */
    if (pLoudness_t1 != NULL)
        free(pLoudness_t1);

    if (pLoudness_t2 != NULL)
        free(pLoudness_t2);
    return 0;
}

////////////////////////////////////////////////////////////////////////////
//  BLOCK   Loudness calculation
////////////////////////////////////////////////////////////////////////////

/*  Third octave band level range for correction of low frequencies
    according to the equal loudness contours (RAP, LT + ΔL)             */
static double RAP[] = {45.f, 55.f, 65.f,71.f, 80.f,90.f,100.f,120.f};

/*  Third octave band level reduction at low frequencies according
    to the equal loudness contours within the eight sectors
    defined by RAP(DLL)                                                  */
static double DLL[N_RAP_RANGES][N_LCB_BANDS] =
{   {-32.f,-24.f,-16.f,-10.f,-5.f,0.f,-7.f,-3.f,0.f,-2.f,0.f},
    {-29.f,-22.f,-15.f,-10.f,-4.f,0.f,-7.f,-2.f,0.f,-2.f,0.f},
{-27.f,-19.f,-14.f, -9.f,-4.f,0.f,-6.f,-2.f,0.f,-2.f,0.f},
{-25.f,-17.f,-12.f, -9.f,-3.f,0.f,-5.f,-2.f,0.f,-2.f,0.f},
{-23.f,-16.f,-11.f, -7.f,-3.f,0.f,-4.f,-1.f,0.f,-1.f,0.f},
{-20.f,-14.f,-10.f, -6.f,-3.f,0.f,-4.f,-1.f,0.f,-1.f,0.f},
{-18.f,-12.f, -9.f, -6.f,-2.f,0.f,-3.f,-1.f,0.f,-1.f,0.f},
{-15.f,-10.f, -8.f, -4.f,-2.f,0.f,-3.f,-1.f,0.f,-1.f,0.f}
};

/*  Critical band level at the threshold in quiet without consideration
    of the ear's transmission characteristics (LTQ)                      */
static double LTQ[] =
{   30.f,18.f,12.f,8.f,7.f,6.f,5.f,4.f,3.f,
    3.f,3.f,3.f,3.f,3.f,3.f,3.f,3.f,3.f,3.f,3.f };

/*  Level correction in accordance with the ear's transmission
    characteristics (A0)                                                 */
static double A0[] =
{   0.0f,0.0f,0.0f,0.0f,0.0f,0.0f,0.0f,0.0f,0.0f,0.0f,
-0.5f,-1.6f,-3.2f,-5.4f,-5.6f,-4.0f,-1.5f,2.0f,5.0f,12.0f
};

/*  Level difference between free and diffuse sound
    field (DDF)                                                          */
static double DDF[] =
{   0.0f, 0.0f, 0.5f, 0.9f, 1.2f, 1.6f, 2.3f, 2.8f,
    3.0f, 2.0f, 0.0f,-1.4f,-2.0f,-1.9f,-1.0f, 0.5f,
    3.0f, 4.0f, 4.3f, 4.0f
};

/*  Adaptation of the third octave band levels to the corresponding
    critical band levels due to different bandwidth (DCB)                */
```

```
static double DCB[] =
{   -.25f,-0.6f,-0.8f,-0.8f,-0.5f,0.0f,0.5f,1.1f,1.5f,1.7f,
    1.8f,1.8f,1.7f,1.6f,1.4f,1.2f,0.8f,0.5f,0.0f,-0.5f
};

/*  Upper limits of the approximated critical bands
    in critical band rate scale (ZUP)                                    */
static double ZUP[] =
{   0.9f,1.8f,2.8f,3.5f,4.4f,5.4f,6.6f,7.9f,9.2f,10.6f,12.3f,
    13.8f,15.2f,16.7f,18.1f,19.3f,20.6f,21.8f,22.7f,23.6f,24.0f
};
/*  Value-range of specific loudness, which defines the
    steepness of upper slopes in the specific
    loudness - critical-band-rate pattern (RNS)                          */
static double RNS[] =
{   21.5f,18.0f,15.1f,11.5f,9.0f,6.1f,4.4f,3.1f,2.13f,1.36f,
    0.82f,0.42f,0.30f,0.22f,0.15f,0.10f,0.035f,0.0f
};

/*  Steepness of upper slopes in the specific loudness
    - critical-band-rate pattern for the value range
    RNS as function of the number of the critical band (USL)             */
static double USL[N_RNS_RANGES][N_CB_RANGES] =
{   { 13.00f,8.20f,6.30f,5.50f,5.50f,5.50f,5.50f,5.50f},
    { 9.00f,7.50f,6.00f,5.10f,4.50f,4.50f,4.50f,4.50f},
    { 7.80f,6.70f,5.60f,4.90f,4.40f,3.90f,3.90f,3.90f},
    { 6.20f,5.40f,4.60f,4.00f,3.50f,3.20f,3.20f,3.20f},
    { 4.50f,3.80f,3.60f,3.20f,2.90f,2.70f,2.70f,2.70f},
    { 3.70f,3.00f,2.80f,2.35f,2.20f,2.20f,2.20f,2.20f},
    { 2.90f,2.30f,2.10f,1.90f,1.80f,1.70f,1.70f,1.70f},
    { 2.40f,1.70f,1.50f,1.35f,1.30f,1.30f,1.30f,1.30f},
    { 1.95f,1.45f,1.30f,1.15f,1.10f,1.10f,1.10f,1.10f},
    { 1.50f,1.20f,0.94f,0.86f,0.82f,0.82f,0.82f,0.82f},
    { 0.72f,0.67f,0.64f,0.63f,0.62f,0.62f,0.62f,0.62f},
    { 0.59f,0.53f,0.51f,0.50f,0.42f,0.42f,0.42f,0.42f},
    { 0.40f,0.33f,0.26f,0.24f,0.22f,0.22f,0.22f,0.22f},
    { 0.27f,0.21f,0.20f,0.18f,0.17f,0.17f,0.17f,0.17f},
    { 0.16f,0.15f,0.14f,0.12f,0.11f,0.11f,0.11f,0.11f},
    { 0.12f,0.11f,0.10f,0.08f,0.08f,0.08f,0.08f,0.08f},
    { 0.09f,0.08f,0.07f,0.06f,0.06f,0.06f,0.06f,0.05f},
    { 0.06f,0.05f,0.03f,0.02f,0.02f,0.02f,0.02f,0.02f}
};

/*  Correction of third octave band levels according to the equal
    loudness contours and calculation of the intensities for the
    third octave bands up to 20 Hz                                       */
void f_corr_third_octave_intensities( double **ThirdOctaveLevel,
                                      double **ThirdOctaveIntens,
                                      int    IdxTime)
{
    short   IdxIntens, IdxLevelRange;
    double  CorrLevel;
    double  *pLevel, *pIntens;

    for (IdxIntens = 0; IdxIntens < N_LCB_BANDS; IdxIntens++)
    {
        pIntens = ThirdOctaveIntens[IdxIntens]+IdxTime;
        pLevel  = ThirdOctaveLevel[IdxIntens]+IdxTime;

        IdxLevelRange = 0;

        while ((*pLevel >   RAP[IdxLevelRange] - DLL[IdxLevelRange][IdxIntens]) &&
                            (IdxLevelRange < N_RAP_RANGES-1))
           IdxLevelRange++;

        CorrLevel = *pLevel + DLL[IdxLevelRange][IdxIntens];
        *pIntens = pow(10., CorrLevel / 10.);
    }
}

/*  Determination of the levels LCB(1), LCB(2) und LCB(3)
```

```
    within the first three critical bands                                   */
void f_calc_lcbs(double **ThirdOctaveIntens, double **Lcb, int IdxTime)
{
    short   IdxIntens, Idx3CB;
    double  *pcbi, *pIntens;

    pcbi    = Lcb[0]+IdxTime;
    pIntens = ThirdOctaveIntens[0]+IdxTime;
    *pcbi   = *pIntens;
    for (IdxIntens = 1; IdxIntens < 6; IdxIntens++)
    {
        pIntens = ThirdOctaveIntens[IdxIntens]+IdxTime;
        *pcbi   += *pIntens;
    }
    pcbi    = Lcb[1]+IdxTime;
    pIntens = ThirdOctaveIntens[6]+IdxTime;
    *pcbi   = *pIntens;
    for (IdxIntens = 7; IdxIntens < 9; IdxIntens++)
    {
        pIntens = ThirdOctaveIntens[IdxIntens]+IdxTime;
        *pcbi   += *pIntens;
    }
    pcbi    = Lcb[2]+IdxTime;
    pIntens = ThirdOctaveIntens[9]+IdxTime;
    *pcbi   = *pIntens;
    for (IdxIntens = 10; IdxIntens < N_LCB_BANDS; IdxIntens++)
    {
        pIntens = ThirdOctaveIntens[IdxIntens]+IdxTime;
        *pcbi   += *pIntens;
    }
    for (Idx3CB = 0; Idx3CB < N_LCBS; Idx3CB++)
    {
        pcbi    = Lcb[Idx3CB]+IdxTime;
        if (*pcbi >0.) *pcbi=(double)(10. * log10(*pcbi));
    }
}

/*  Calculation of loudness related to critical band level              */
void f_calc_core_loudness(double **ThirdOctaveLevel, double **Lcb,
                          double **CoreLoudness, int SoundField, int IdxTime)
{
    short   IdxCL;
    double  S;
    double  MP1, MP2;
    double  *pLe, *pLtq, *pCoreL;

    pLtq = LTQ;

    for (IdxCL =0; IdxCL < N_CORE_LOUDN-1; IdxCL++)
    {
        if (IdxCL < N_LCBS)
            pLe = Lcb[IdxCL]+IdxTime;
        else
            pLe = ThirdOctaveLevel[IdxCL+8]+IdxTime;

        pCoreL = CoreLoudness[IdxCL]+IdxTime;

        *pLe -= A0[IdxCL];
        *pCoreL = 0.;

        if (SoundField == SoundFieldDiffuse)
            *pLe += DDF[IdxCL];
        if (*pLe > *pLtq)
        {
            *pLe += -DCB[IdxCL];
            S = .25;
            MP1 = .0635f * (double)pow(10., 0.025 * (*pLtq));
            MP2 = (double)pow((1. - S + S * pow(10., 0.1 * (*pLe - *pLtq))), .25) - 1.0f;
            *pCoreL = MP1 * MP2;
            if (*pCoreL <= 0.)
                *pCoreL = 0.;
```

```
        }
        pLtq++;
    }

    /*  Set last critical band to zero                                  */
    pCoreL  = CoreLoudness[IdxCL]+IdxTime;
    *pCoreL = 0.;
}

/*  Correction of the specific loudness within the lowest critical band
    for the consideration of the run of threshold in quiet within this
    critical band                                                       */
void f_corr_loudness(double **CoreLoudness, int IdxTime)
{
    double  CorrCL, *pCoreLoudness;

    pCoreLoudness = CoreLoudness[0]+IdxTime;
    CorrCL = 0.4f + 0.32f * (double)pow(*pCoreLoudness, .2);
    if (CorrCL < 1.) *pCoreLoudness *= CorrCL;
}

/*  Calculation of specific loudness pattern and integration of overall
    loudness by attaching slopes towards higher frequencies             */
void f_calc_slopes(double **CoreLoudness, double *Loudness,
                   double *SpecLoudness[N_BARK_BANDS], int IdxTime)
{
    short   IdxCL, IdxNS, IdxCBN, IdxRNS;
    int     NextCriticalBand;
    double  N1, N2, Z, Z1, Z2, ZK, DZ;
    double  _USL, _ZUP, CoreL;
    double  *pLoudness = Loudness+IdxTime, *pCoreL;
    double  NS[N_BARK_BANDS];

    N1          = 0.;
    Z           = 0.1f;
    Z1          = 0.;
    IdxRNS      = 0;
    IdxNS       = 0;
    *pLoudness  = 0;

    for (IdxCL = 0; IdxCL < N_CORE_LOUDN; IdxCL++)  /* Do for all core
                                                       loudness values */
    {
        pCoreL  = CoreLoudness[IdxCL]+IdxTime;
        CoreL   = *pCoreL;
        _ZUP    = ZUP[IdxCL];
        _ZUP    += .0001f;
        IdxCBN  = IdxCL - 1;
        if (IdxCBN > N_CB_RANGES-1)
            IdxCBN = N_CB_RANGES-1;
        NextCriticalBand    = 0;
        do
        {   if (N1 > CoreL) /*  Slope loudness > core loudness?         */
            {   _USL    = USL[IdxRNS][IdxCBN];
                /*  Contribution of the value N2 of the specific
                    loudness at the cut-off frequency of the
                    corresponding critical band                         */
                N2      = RNS[IdxRNS];
                if (N2 < CoreL)
                    N2  = CoreL;
                DZ  = (N1 - N2) / _USL;
                Z2  = Z1 + DZ;
                if (Z2 > _ZUP)
                {
                    NextCriticalBand=1;
                    Z2  = _ZUP;
                    DZ  = Z2 - Z1;
                    N2  = N1 - DZ * _USL;
                }
                /*  Contribution of the loudness of slope excitation
                    to the overall loudness and calculation of the
```

```
                    associated interpolated values NS(IdxNS) in the
                    interval Z=IdxNS*0.1 BARK                          */
                *pLoudness += DZ * (N1 + N2) / 2.;
                for (ZK = Z; ZK <= Z2; ZK = ZK + 0.1f)
                {
                    NS[IdxNS] = N1 - (ZK - Z1) * _USL;
                    SpecLoudness[IdxNS][IdxTime] = NS[IdxNS];
                    IdxNS++;
                }
                Z    = ZK;
            }
            else    /*  Slope loudness <= core loudness               */
            {   /*  Determination of the number J of the sector of
                    specific loudness                                  */
                if (N1 < CoreL)
                {
                    IdxRNS  = 0;
                    while ((IdxRNS < N_RNS_RANGES) && RNS[IdxRNS] >=CoreL)
                        IdxRNS++;
                }
                /*  Contribution of the non-masked loudness related to
                    critical band level to the total loudness and
                    calculation of the interpolated values NS(IdxCL)
                    in the interval Z=IdxNS*0.1 BARK                   */
                NextCriticalBand = 1;
                Z2   = _ZUP;
                N2   = CoreL;
                *pLoudness += N2 * (Z2 - Z1);
                ZK   = Z;
                while (ZK <= Z2)
                {
                    NS[IdxNS] = N2;
                    SpecLoudness[IdxNS][IdxTime] = NS[IdxNS];
                    IdxNS++;
                    ZK   = ZK + 0.1f;
                }
                Z    = ZK;
            }

        /*  Step to next segment  */
        while  ((N2 <= RNS[IdxRNS]) && (IdxRNS < N_RNS_RANGES-1))
            IdxRNS++;
        if (IdxRNS > N_RNS_RANGES-1)
            IdxRNS  = N_RNS_RANGES-1;
        Z1   = Z2;
        N1   = N2;
        } while (!NextCriticalBand);

        if (*pLoudness < 0.)
            *pLoudness  = 0;
    }
}

/*  Loudness calculation from provided third octave levels
    Input parameters:
    ThirdOctaveLevel:   Array of 28 double arrays with one array per
                        frequency band. All 28 double arrays shall
                        have the same length which is the number of
                        time samples.
    NumSamplesLevel:    Number of time samples in ThirdOctaveLevel
    SoundField:         One of SoundFieldDiffuse or SoundFieldFree
    Method:             One of LoudnessMethodStationary or
                        LoudnessMethodTimeVarying
    OutLoudness:        Pointer to a double array of length
                        NumSamplesLevel to which the loudness result
                        is written.
    OutSpecLoudness:    Pointer to a double array of length
                        NumSamplesLevel to which the specific loudness
                        is written.
    Output value:       Number of values actually written into
                        output parameters or negative error code.      */
```

```
int f_loudness_from_levels(
    double  **ThirdOctaveLevel,
    int     NumSamplesLevel,
    int     SoundField,
    int     Method,
    double  *OutLoudness,
    double  *OutSpecLoudness[N_BARK_BANDS]
  )
{
    double  *CoreLoudness[N_CORE_LOUDN];
    double  *ThirdOctaveIntens[N_LCB_BANDS], *Lcb[N_LCBS];

    int     IdxTime;
    int     SampleRateLevel;
    int     retval;

    SampleRateLevel = SR_LEVEL;

    /*  Memory allocation                                                   */
    retval = callocRaggedArray(CoreLoudness, N_CORE_LOUDN, NumSamplesLevel);
    if (retval < 0)
    {
        return retval;
    }

    retval = callocRaggedArray(Lcb, N_LCBS, NumSamplesLevel);
    if (retval < 0)
    {
        freeRaggedArray(CoreLoudness, N_CORE_LOUDN);
        return retval;
    }

    retval = callocRaggedArray(ThirdOctaveIntens, N_LCB_BANDS, NumSamplesLevel);
    if (retval < 0)
    {
        freeRaggedArray(CoreLoudness, N_CORE_LOUDN);
        freeRaggedArray(Lcb, N_LCBS);
        return retval;
    }

    /*  Calculate core loudness                                             */
    for (IdxTime = 0; IdxTime < NumSamplesLevel; IdxTime++)
    {
        f_corr_third_octave_intensities(ThirdOctaveLevel,
            ThirdOctaveIntens, IdxTime);
        f_calc_lcbs(ThirdOctaveIntens, Lcb, IdxTime);
        f_calc_core_loudness(ThirdOctaveLevel, Lcb, CoreLoudness,
            SoundField, IdxTime);
    }

    /*  Memory deallocation                                                 */
    freeRaggedArray(Lcb, N_LCBS);
    freeRaggedArray(ThirdOctaveIntens, N_LCB_BANDS);

    /*  Correction of specific loudness within lowest critical band     */
    for (IdxTime = 0; IdxTime < NumSamplesLevel; IdxTime++)
        f_corr_loudness(CoreLoudness, IdxTime);

    /*  Time-varying loudness: nonlinearity                             */
    if (Method == LoudnessMethodTimeVarying)
        f_nl(CoreLoudness, SampleRateLevel, NumSamplesLevel);

    /*  Calculation of specific loudness                                */
    for (IdxTime = 0; IdxTime < NumSamplesLevel; IdxTime++)
    {
        f_calc_slopes(CoreLoudness, OutLoudness, OutSpecLoudness,
            IdxTime);
    }

    /*  Memory deallocation                                                 */
    freeRaggedArray(CoreLoudness, N_CORE_LOUDN);
```

```
    /*  Time-varying loudness: temporal weighting                       */
    if (Method == LoudnessMethodTimeVarying)
    {
        retval = f_temporal_weight_loudness(OutLoudness, SampleRateLevel,
                    NumSamplesLevel);
        if (retval < 0)
        {
            return retval;
        }
    }

    return NumSamplesLevel;
}

/*  Loudness calculation from provided time signal
    Input parameters:
    pSignal:            Data structure with input signal at 48 kHz.
    SoundField:         One of SoundFieldDiffuse or SoundFieldFree
    Method:             One of LoudnessMethodStationary or
                        LoudnessMethodTimeVarying
    TimeSkip:           Value specifying number of seconds to
                        skip for level calculation. (only relevant
                        for Method = LoudnessMethodStationary)
    OutLoudness:        Pointer to a double array of length
                        NumSamplesLevel to which the loudness result
                        is written.
    OutSpecLoudness:    Pointer to a double array of length
                        NumSamplesLevel to which the specific loudness
                        is written.
    SizeOutput:         Size of output variables OutLoudness and
                        OutSpecLoudness
    Output value:       Number of values actually written into
                        output parameters or negative error code.       */
int f_loudness_from_signal(
    struct  InputData *pSignal,
    int     SoundField,
    int     Method,
    double  TimeSkip,
    double  *OutLoudness,
    double  *OutSpecLoudness[N_BARK_BANDS],
    int     SizeOutput
  )
{
    double  *ThirdOctaveLevel[N_LEVEL_BANDS];
    int     SampleRateLevel = 1;
    int     DecFactorLevel = 1;
    int     NumSamplesTime = 1;
    int     NumSamplesLevel = 1;

    int retval;

    if (Method == LoudnessMethodStationary)
    {
        DecFactorLevel = (int)(pSignal->NumSamples);
    }
    else if (Method == LoudnessMethodTimeVarying)
    {
        SampleRateLevel = SR_LEVEL;

        DecFactorLevel = (int)(pSignal->SampleRate / SampleRateLevel);

        NumSamplesTime = pSignal->NumSamples;
        NumSamplesLevel = NumSamplesTime / DecFactorLevel;
    }
    else
    {
        return LoudnessErrorUnsupportedMethod;
    }

    /*  Check size of output vectors                                    */
```

```
    if (SizeOutput < NumSamplesLevel)
    {
        return LoudnessErrorOutputVectorTooSmall;
    }

    /*  Memory allocation                                                 */
    retval = callocRaggedArray(ThirdOctaveLevel, N_LEVEL_BANDS, NumSamplesLevel);
    if (retval < 0)
    {
        return retval;
    }

    /*  Calculate third octave levels                                     */
    retval = f_calc_third_octave_levels(pSignal, ThirdOctaveLevel,
                DecFactorLevel, Method, TimeSkip);
    if (retval < 0)
    {
        freeRaggedArray(ThirdOctaveLevel, N_LEVEL_BANDS);
        return retval;
    }

    /*  Loudness calculation                                              */
    retval = f_loudness_from_levels(ThirdOctaveLevel, NumSamplesLevel,
                SoundField, Method, OutLoudness,
                OutSpecLoudness);

    /*  Memory deallocation                                               */
    freeRaggedArray(ThirdOctaveLevel, N_LEVEL_BANDS);

    return retval;
}
```

# Annex B
(normative)

# Test signals for the validation of implementation

## B.1 Background

The implementation of this document shall be validated on a set of synthetic and technical signals covering representative applications. For compliance requirements see 5.1 or 6.1.

The test signals are sampled at 48 kHz and provided as 16-bit WAVE files. For the calibration of the WAVE files, 0 dB (relative to full scale) shall correspond to a sound pressure level of 100 dB. The calculation of the stationary method shall start from 0,2 s. For convenience the tables of the results from the test implementation (including tolerances) are supplied as EXCEL files. Modifying the test signals (e.g. appending zeros) is not allowed.

NOTE A software package including the test signals and the tables of the results from the test implementation (including tolerances) [WAVE and EXCEL files] can be freely downloaded from http://standards.iso.org/iso/532/-1/ed-1/en

## B.2 Third octave levels as input for stationary loudness

Test signal 1

A machinery noise presented binaurally in free field.

## B.3 Synthetic signals for stationary loudness

Test signal 2

A 250 Hz tone presented binaurally in free field with a level of 80 dB.

Test signal 3

A 1 kHz tone presented binaurally in free field with a level of 60 dB.

Test signal 4

A 4 kHz tone presented binaurally in free field with a level of 40 dB.

Test signal 5

A pink noise presented binaurally in free field with an overall level of 60 dB.

## B.4 Synthetic signals for time-varying loudness

Test signal 6

A 250 Hz tone presented binaurally in free field with a time-varying sound pressure level starting with 30 dB and increasing linearly to 80 dB.

Test signal 7

A 1 kHz tone presented binaurally in free field with a time-varying sound pressure level starting with 30 dB and increasing linearly to 80 dB.

Test signal 8

A 4 kHz tone presented binaurally in free field with a time-varying sound pressure level starting with 30 dB and increasing linearly to 80 dB.

Test signal 9

A pink noise presented binaurally in free field with a time-varying sound pressure level starting with 30 dB/third octave band and increasing linearly to 80 dB/third octave band.

Test signal 10

A 1 kHz tone pulse presented binaurally in free field with a peak rms level of 70 dB and a duration of 10 ms.

Test signal 11

A 1 kHz tone pulse presented binaurally in free field with a peak rms level of 70 dB and a duration of 50 ms.

Test signal 12

A 1 kHz tone pulse presented binaurally in free field with a peak rms level of 70 dB and a duration of 500 ms.

Test signal 13

A combination of two 1 kHz tone pulses presented binaurally in free field, the first pulse with a peak rms level of 60 dB and a duration of 100 ms and the second pulse with a peak rms level of 80 dB and a duration of 10 ms.

## B.5 Technical signals for time-varying loudness

Test signal 14

Propeller-driven airplane noise measured with a single microphone. Free field is assumed for the calculation of loudness.

Test signal 15

Vehicle interior noise measured at 40 km/h with a single microphone. Diffuse field is assumed for the calculation of loudness.

Test signal 16

Hairdryer noise measured with a single microphone. Free field is assumed for the calculation of loudness.

Test signal 17

Machine gun firing 6 bursts of automatic fire, of differing lengths per burst. Free field is assumed for the calculation of loudness.

Test signal 18

Hammering sound, banging a small picture-hanging type nail into a typical hollow wooden door. Free field is assumed for the calculation of loudness.

Test signal 19

Rattling slow creak from a door opening. Free field is assumed for the calculation of loudness.