3GPP TR 26.843 V16.0.0 (2018-09)

Technical Specification

3rd Generation Partnership Project;

Technical Specification Group Services and System Aspects;

Codec for Enhanced Voice Services (EVS);

Study on non-bit-exact conformance criteria and tools for floating-point EVS codec

(Release 16)

** 

The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP..  
The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented.  
This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification.  
Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices.

Keywords

Non Bit-Exact Conformance, Floating-Point EVS Codec

***3GPP***

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis

Valbonne - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

***Copyright Notification***

No part may be reproduced except as authorized by written permission.  
The copyright and the foregoing restriction extend to reproduction in all media.

© 2018, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).

All rights reserved.

UMTS™ is a Trade Mark of ETSI registered for the benefit of its members

3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  
LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners

GSM® and the GSM logo are registered and owned by the GSM Association

Contents

Foreword [5](#__RefHeading___Toc524274757)

Introduction [5](#__RefHeading___Toc524274758)

1 Scope [6](#__RefHeading___Toc524274759)

2 References [6](#__RefHeading___Toc524274760)

3 Definitions and abbreviations [7](#__RefHeading___Toc524274761)

3.1 Definitions [7](#__RefHeading___Toc524274762)

3.2 Abbreviations [7](#__RefHeading___Toc524274763)

4 Overview [7](#__RefHeading___Toc524274764)

4.1 Introduction [7](#__RefHeading___Toc524274765)

5 Methods Description [8](#__RefHeading___Toc524274766)

5.1 Description [8](#__RefHeading___Toc524274767)

5.2 Signal Based Methods [8](#__RefHeading___Toc524274768)

5.2.1 General considerations [8](#__RefHeading___Toc524274769)

5.2.2 SNR [9](#__RefHeading___Toc524274770)

5.2.2.1 Methodology [9](#__RefHeading___Toc524274771)

5.2.2.2 Thresholds and Criteria [9](#__RefHeading___Toc524274772)

5.2.3 RMS error threshold [10](#__RefHeading___Toc524274773)

5.2.3.1 Methodology [10](#__RefHeading___Toc524274774)

5.2.3.2 Thresholds and Criteria [10](#__RefHeading___Toc524274775)

5.2.4 Spectral Distortion [10](#__RefHeading___Toc524274776)

5.2.4.1 Methodology [10](#__RefHeading___Toc524274777)

5.2.4.2 Thresholds and Criteria [11](#__RefHeading___Toc524274778)

5.2.5 Analysis Flow and Reporting [11](#__RefHeading___Toc524274779)

5.2.6 List of Thresholds [12](#__RefHeading___Toc524274780)

5.3 Perceptually Based methods [13](#__RefHeading___Toc524274781)

5.3.1 General Consideration [13](#__RefHeading___Toc524274782)

5.3.2 MOS-LQO Validation [13](#__RefHeading___Toc524274783)

5.3.2.1 General Methodology [13](#__RefHeading___Toc524274784)

5.3.2.2 Test Cases [14](#__RefHeading___Toc524274785)

5.3.2.3 Scores Reporting and Analysis [14](#__RefHeading___Toc524274786)

5.3.2.4 Thresholds and Criteria [15](#__RefHeading___Toc524274787)

5.3.3 Maximum Loudness Difference [18](#__RefHeading___Toc524274788)

5.3.3.1 General Methodology [18](#__RefHeading___Toc524274789)

5.3.3.2 Proposed Encoder Conformance Test [18](#__RefHeading___Toc524274790)

6 Results [19](#__RefHeading___Toc524274791)

6.1 Experiment A [19](#__RefHeading___Toc524274792)

6.1.1 Compiler Options [19](#__RefHeading___Toc524274793)

6.1.2 MOS-LQO Results [19](#__RefHeading___Toc524274794)

6.2 Experiment B [23](#__RefHeading___Toc524274795)

6.2.1 Compiler Options [23](#__RefHeading___Toc524274796)

6.2.2 Results [23](#__RefHeading___Toc524274797)

6.2.3 Restricted Results [26](#__RefHeading___Toc524274798)

6.3 Experiment C [27](#__RefHeading___Toc524274799)

6.3.1 Compiler Options [27](#__RefHeading___Toc524274800)

6.3.1.1 Introduction [27](#__RefHeading___Toc524274801)

6.3.1.2 Icc Compiler on Atom Platform [28](#__RefHeading___Toc524274802)

6.3.1.3 Gcc compiler on Xeon platform [28](#__RefHeading___Toc524274803)

6.3.1.4 Gcc compiler on ARM platform [28](#__RefHeading___Toc524274804)

6.3.2 Decoder Test Results [28](#__RefHeading___Toc524274805)

6.3.3 Robustness Decoder Test Results [30](#__RefHeading___Toc524274806)

6.4 Experiment D [30](#__RefHeading___Toc524274807)

6.4.1 Delta-MOS-LQO Behaviour for Mixed-Music Signals [30](#__RefHeading___Toc524274808)

6.4.2 Clean Speech Input Example [32](#__RefHeading___Toc524274809)

6.4.3 MOS LQO evaluation [33](#__RefHeading___Toc524274810)

6.4.4 Decoder test results [35](#__RefHeading___Toc524274811)

6.5 Experiment E [36](#__RefHeading___Toc524274812)

6.5.1 Delta-POLQA Limitations with Noisy Speech and Frame Erasures [36](#__RefHeading___Toc524274813)

6.5.2 MOS-LQO Verification test [37](#__RefHeading___Toc524274814)

6.5.3 Decoder Test [39](#__RefHeading___Toc524274815)

6.6 Experiment F [40](#__RefHeading___Toc524274816)

7 Conformance Process [42](#__RefHeading___Toc524274817)

7.1 Description [42](#__RefHeading___Toc524274818)

8 Interoperability [43](#__RefHeading___Toc524274819)

8.1 Description [43](#__RefHeading___Toc524274820)

8.2 Interoperability Testing [43](#__RefHeading___Toc524274821)

8.3 Conclusion [45](#__RefHeading___Toc524274822)

9 Coverage Assessment [46](#__RefHeading___Toc524274823)

10 Conclusions [46](#__RefHeading___Toc524274824)

Annex A: POLQA Considerations for Conformance Test [47](#__RefHeading___Toc524274825)

A.1 Specific Recommendation from ITU-T P.863.1 [9] [47](#__RefHeading___Toc524274826)

A.2 Averaging of MOS-LQO Scores Over Long Inputs [47](#__RefHeading___Toc524274827)

A.3 Concerns over the Suitability of Perceptually Based Methods [48](#__RefHeading___Toc524274828)

Annex B: Change history [49](#__RefHeading___Toc524274829)

# Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

# Introduction

The EVS coder (TS 26.441) provides enhanced quality for speech and audio communications compared to AMR-WB and 3GPP has standardized both a fixed-point version (TS 26.442) and a floating-point version (TS 26.443). Currently in TS 26.444 (Codec for Enhanced Voice Services (EVS); Test sequences) the conformance of the EVS coder implementation is achieved by checking the bit-exactness of output test vectors with the reference test vectors for both encoder and decoder, for both the fixed-point and floating-point implementations.

However, the bit-exact criteria defined in TS 26.444 is of very limited use for the floating-point implementation in TS 26.443, as the output values will change slightly without affecting the speech/audio quality - depending on the compiler, compile options, OS and platform - and therefore failing the bit-exactness test. This has the effect that the EVS floating-point code cannot generally be used for 3GPP voice services as the test vectors have been generated using Microsoft Visual Studio version 10 which is unlikely to match the target platform.

The product and application space using voice services is changing, resulting in different architectures using a variety of different types of core processing units. Being able to use either fixed-point or floating-point embedded implementation based on architectural capabilities would allow a wider and faster proliferation of EVS, thereby benefiting end user experience. In addition, it would provide more flexibility in architectural implementations regarding factors such as power and cost.

The present document investigates possible tools and criteria to develop non bit-exact conformance for the floating-point code in TS 26.443, ensuring that high quality floating-point implementations preserve the quality of EVS.

# 1 Scope

The present document provides a study on the Conformance of Non Bit Exact implementation for EVS floating point standard in TS. 26.443. The study focuses on:

- To investigate the behaviour of different implementations of the floating-point reference code (TS 26.443), for example, those built with different versions / settings of various compilers and running on various floating-point architectures.

- To do the investigation using different test material, including clean speech, noisy speech, mixed/music content and taking into account interoperability aspects including floating-point - fixed-point and among various floating-point implementations.

- To identify and propose reliable conformance criteria and methodologies that would be able to reject any undesirable deviation, i.e. bad implementation.

- To develop one or more tools, in the form of scripts or executables, that could be used for determining acceptance/rejection based on the provided conformance criteria.

- To develop any potential additional test vectors that would be needed.

- To propose recommendation(s) on the suitability of the potential new non bit-exact conformance process for 3GPP services (e.g., gives carrier-grade quality).

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 26.442: "Codec for Enhanced Voice Services (EVS); ANSI C code (fixed point)".

[3] 3GPP TS 26.443: "Codec for Enhanced Voice Services (EVS); ANSI C code (floating point)".

[4] 3GPP TS 26.444: "Codec for Enhanced Voice Services (EVS); Test Sequences".

[5] Tdoc S4-141287: "Verification of the EVS Floating Point Code".

[6] Tdoc S4-141392: "EVS Permanent Document EVS-7c: Processing functions for characterization phase".

[7] 3GPP TR 26.952: "Codec for Enhanced Voice Services (EVS); Performance Characterization".

[8] Proc. IEEE Digital Signal Processing Workshop. "Comparison of distance measures in discrete spectral modelling", B. Wei and J. D. Gibson, Oct. 2000.

[9] ITU-T Recommendation P.863.1 (09/2014): "Application guide for recommendation ITU-T P.863".

[10] ISO/IEC 14496-26:2010: "Information technology -- Coding of audio-visual objects -- Part 26: Audio conformance".

[11] ITU-R Recommendation BS.1387-1 (11/2001): "Method for objective measurements of perceived audio quality".

[12] ITU-T Recommendation P.863 (03/2018): "Perceptual objective listening quality assessment".

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

UL Up-link

# 4 Overview

## 4.1 Introduction

The EVS coder (TS 26.441) provides enhanced quality for speech and audio communications relative to AMR-WB and 3GPP has standardized both a fixed-point version (TS 26.442) and a floating-point version (TS 26.443). Currently in TS 26.444 (Codec for Enhanced Voice Services (EVS); Test sequences) the conformance of the EVS coder implementation is achieved by checking the bit-exactness of output test vectors with the reference test vectors for both encoder and decoder, for both the fixed-point and floating-point implementations.

The bit-exact criteria defined in TS 26.444 are of very limited use for the floating-point implementation in TS 26.443 as the output values will be similar but not bit-exact for different compilers, compile options, OS and platform used and therefore failing the bit-exactness test.

This technical report aims at documenting possible methods for non bit-exact conformance process for floating-point implementation that would allow conforming implementations to be used in all scenarios acceptable for bit-exact implementations of fixed-point version (TS 26.442) and floating-point version (TS 26.443).

The scope of the study item is to assess the use of various floating-point processing cores and compilers with various levels of optimization, and establish potential conformance criteria and a tool(s) that could be used for confirming conformance under those variations. The scope of the intended conformance tool(s) is to assess the conformance of implementations to the developed criteria. The conformance criteria to be developed will aim at accepting proper floating-point processing compute core and compiler optimizations, while rejecting all bad implementations, for example coming from functional or code changes, too aggressive optimization of the compiler or insufficient arithmetic precision.

In Clause 5 various methods are described.

In Clause 6 results obtained with the various methods described in Clause 5 are presented..

In Clause 7 possible conformance process and criteria are discussed.

Interoperability is an important feature of the coder, and implication for non bit-exact conformance will be investigated in Clause 8.

Clause 9 will look at coverage of the proposed method.

Clause 10 will address any other topics relevant to the context of this study.

Clause 11 will conclude on the feasibility of using non bit-exact tools and criteria for EVS floating point conformance.

# 5 Methods Description

## 5.1 Description

The EVS codec uses multiple coding schemes to get the best coding efficiency. For the decoder, these different modes are defined by the parameters in the bit-stream. Methods based on comparison of decoded PCM signal that are close to bit exactness, could be used to assess the quality of EVS decoder implementation.

EVS encoder is using many different modes for encoding efficiency that are based on threshold decision. A non bit-exact computation of the threshold based decision may result in selecting a different mode, which may impact strongly the signal characteristic, without necessarily affecting the perceived quality. In this case analysis methods based on perceptual consideration could be more adequate to assess the encoder implementation.

## 5.2 Signal Based Methods

### 5.2.1 General considerations

The reference PCM signals are taken from the decoded floating point test vector library of TS 26.444. The PCM signal under test are obtained by running the floating point bit-stream included in TS 26.444 through the Decoder under Test (Figure 1). The reference decoder is the floating-point code of TS 26.443.



Figure 1: Flow diagram for the decoder test using signal based metrics

All metrics are calculated on the reference PCM signal  and the PCM signal under test  based on 20ms frames. The frames of the two signals will be time aligned, this means the delay compensation in EVS encoder and decoder remains ON (the default configuration). Furthermore the frame processing is aligned with the encoded frame by adding the decoder delay. Table 1 shows the delay values used for the different sampling frequencies.

Table 1: Delay used for alignment of processing frames with encoded frames

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sampling frequency | 8000 Hz | 16000 Hz | 32000 Hz | 48000 Hz |
| Delay (samples) | 10 | 37 | 74 | 111 |

The number of samples  for a 20ms frame size is defined by , where  represents the sampling rate.

The PCM signals  and  should be scaled between -1 and 1.

### 5.2.2 SNR

#### 5.2.2.1 Methodology

The segmental SNR method is derived from the decoder conformance used in ISO/IEC 14496-26 [10]. For each 20 ms segment, the following values need to be calculated:

Energy of reference signal:

Noise energy:

Signal to noise ratio  with 

As EVS is a switched codec containing a LPC based speech coder and a MDCT based transform coder, the SNR values vary significantly depending on the used coding mode. Therefore, a constant threshold for the SNR is not suitable but instead, a reference value per frame and test vector should be specified. The SNR should be compared against the thresholds by

where  is a 20 ms frame index and  is the test vector index

This means, a potential conformance package needs to provide the  values for all test vectors and frames.

#### 5.2.2.2 Thresholds and Criteria

The SNR reference values are created per test vector  and frame   Three example platforms are compared to the test vectors created using the reference platform (Windows). The final SNR references are the minimum values out of the three example platforms.

The three example platforms are listed in the following:

- Linux, GCC, OPTIM=3, TARGET\_PLATFORM=x86\_64

- macOS, CLANG, OPTIM=3, TARGET\_PLATFORM=x86\_64

- arm-linux-gnueabihf\_armv7, OPTIM=3

For all platform, the default test vectors are processed by

- ./Readme\_AMRWB\_IO\_dec\_multi.txt;

- ./Readme\_EVS\_dec\_multi.txt;

- ./Readme\_JBM\_dec\_multi.txt

For each platform  the SNR values are determined for each frame and vector by:



The combined SNR reference value is then given by:



### 5.2.3 RMS error threshold

#### 5.2.3.1 Methodology

The RMS method is derived from the decoder conformance used in ISO/IEC 14496-26 [10]. The RMS error is calculated for each 20ms frame and compared to a threshold according to:

#### 5.2.3.2 Thresholds and Criteria

Ideally the difference between fixed-point and floating-point implementation will be due to rounding in mathematical operation. One obvious value to choose for an RMS error threshold is to assume change on the last bit of the audio signal:

 with 

### 5.2.4 Spectral Distortion

#### 5.2.4.1 Methodology

The spectral distortion method can be conducted on a 20 ms frame base by the following steps:

Calculate the absolute FFT spectrum of  and  using a Hanning window





with  

The 32768 is due to MATLAB scaling and to align to 16bit PCM C-code. This scaling is dependent on the input value range.

For all spectral bins the distortion d is calculated according to the following pseudo code:

cnt=0

d=0

for k=1..N/2-1

    if (==0 && ==0)

        X\_Y = 1;

        Y\_X = 1;

    else

        if (==0)

            X\_Y = 0;

            Y\_X = 2;

        else if (==0)

            X\_Y = 2;

            Y\_X = 0;

        else

            X\_Y = ( \* ) / ( \* );

            Y\_X = ( \* ) / ( \* );

        end

  end

COSH = (X\_Y + Y\_X - 2)/2;

    d = d + COSH;

    cnt = cnt+1;

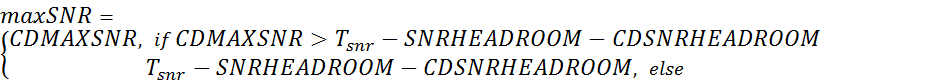
end

d = d/cnt;

The distortion value  is to be compared against a threshold.

#### 5.2.4.2 Thresholds and Criteria

The frame will be considered as pass if 

with 

### 5.2.5 Analysis Flow and Reporting

The three metrics are computed in a specific order, as shown in Figure 2. Once a frame passes a metrics, the process is stopped and the next frame is analysed. The SNR metrics is computed on the frames failing the RMS error criteria. Similarly the Spectral Distortion metrics is computed on the frames failing the SNR criteria.



Figure 2: Flow chart for decoder tool

In a file one or two frames could slightly be above the threshold. To avoid to relax the threshold, a criteria could be to add a constraint on the number of frames failing per file.

if number\_of\_frames\_failing =< THRESH\_GOOD\_FRAMES\_TO\_PASS \* number\_of\_frame\_in file, the test signal will be considered equivalent to the reference signal.

All the test vectors need to pass for the implementation to be conformant.

In addition to the number of fail/pass test vectors, the statistics from the three methods should be displayed. Table x2 shows an example of reporting.

Table 2: Template for result presentation

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMS | WSNR | Spectral Distortion |
| Number of frames tested |  |  |  |
| Number of frames passing |  |  |  |
| Number of frames failing |  |  |  |
| Ratio of frames passing |  |  |  |
| Ratio of frames failing |  |  |  |

As part of conformance criteria, thresholds could be set for the ratio of frames passing with RMS and WNR tests (Ratio\_RMSframespassing\_and RatioWSNRframespassing respectively).

To illustrate the need for the RMS criterion the following histogram (Figure 3) shows SNR values for frames passing the RMS criterion for a platform considered conformant (GCC, O3). The total ratio of RMS frames passing is around 48%. As can be seen, the majority of those frames show a very low SNR value (e.g. <5 dB), not useful for a secure classification of the frame.

The second plot shows the correlation of the SNR values to the actual signal power. As one can see, a low signal power corresponds to a low SNR value, indicating that in order to use the SNR criterion in a reliable way, a certain amount of signal power is required. It would therefore be good if the RMS criterion be able to capture at least all low-power frames. To ensure that these low-power frames would not be handled by the by SNR criterion in an uncertain way, a dedicated number of frames is expected to pass the RMS criterion based on all the files tested.



Figure 3: Plot of SNR passing the RMS criterion

### 5.2.6 List of Thresholds

The list of the thresholds used in decoder test are summarized in table 3 with example values.

Table 3: List of thresholds

|  |  |  |
| --- | --- | --- |
| Thresholds | Description | Example value |
| SNRHEADROOM | Headroom compare to the Tsnr threshold | 3 dB |
| CDSNRMAX | Limit of SNR for the spectral distortion test | 0 dB |
| CDSNRHEADROOM | Headroom compare to Tsnr threshold for the spectral distortion test | 10 dB |
| Tsd | Threshold for the spectral distance | 6.6 |
| THRESH\_GOOD\_FRAMES\_TO\_PASS | Factor for number of failing frame per file | 0.005 |
| Ratio\_RMSframespassing | Minimal percentage for frames passing RMS error test | 47% |
| RatioWSNRframespassing | Minimal percentage for frames passing WSNR test | 95% |

## 5.3 Perceptually Based methods

### 5.3.1 General Consideration

For perceptual metrics, the fixed-point code should be the target scores to achieve, as the fixed-point code is considered as the reference in TS 26.444.

### 5.3.2 MOS-LQO Validation

#### 5.3.2.1 General Methodology

EVS floating point standard has been validated using comparison of MOS-LQO scores between the fixed point implementation and the floating point implementation for various combinations of encoder / decoder [4]. The same methodology could be used to assess EVS floating-point implementations. For this validation, four combinations of encoder/decoder are used (3GPP EVS encoder/decoder executables are taken from TS 26.442):

a) 3GPP fixed-point encoder and 3GPP fixed-point decoder (FX/FX),

b) floating-point Encoder under Test and floating-point Decoder under Test (FL/FL),

c) 3GPP fixed-point encoder and floating-point Decoder under Test (FX/FL),

d) floating-point Encoder under Test and 3GPP fixed-point decoder (FL/FX).

The MOS-LQO scores are computed for each of the four cases using the decoded files and the original test files.

The test files are based on P.501 Annex D to be compliant with POLQA tool. 30 files representing various talkers and languages are used for each test conditions, and the average MOS-LQO scores are reported.

When using POLQA, one need to be aware of limitation of the current tool. Annex A highlights some issues that could be relevant to the conformance process.

As the EVS extensive subjective test reported in TR 26.952 has been carried out using the fixed point implementation, the average MOS-LQO score obtained for scenario a) is considered the reference score. For the three other scenarios (b, c and d), the difference in MOS-LQO of a) are then computed:

- a) - b)

- a) - c)

- a) - d)

The difference a) - b) assesses the encoder + decoder floating-point implementation, the difference a) - c) assesses the decoder implementation and a) - d) assesses the encoder implementation.

Figure 4 represents the flow diagram to obtain the MOS-LQO in the various scenario.







Figure 4: Flow diagram to obtain the MOS-LQO in the three scenario

#### 5.3.2.2 Test Cases

The differences are computed for various test conditions:

- All the codec modes of EVS

- All the bandwidths of EVS

- All the bit-rates of EVS, including bit-rate switching

- DTX ON and OFF

- Various levels: -26 dB, -36 dB, -16 dB

- Various noise conditions

- Various impairment conditions

The files have been processed according to EVS-7c (EVS processing plan) for the various test conditions [6].

In all, 941 test conditions are assessed, representing 225,600 second of speech, or a little bit more than 62 hours.

#### 5.3.2.3 Scores Reporting and Analysis

The score difference for all the test conditions could be reported using template shown in Table 4

Table 4: Template for result presentation

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input signal | Bandwidth | Bit rate | DTX | Level | FER/Profile | a) - b) | a) - c) | a) - d) |
| clean speech, noisy speech, mixed/music | NB, WB, SWB, or FB | e.g. 7,2 | off or on | -26, -16, or -36 dBov | No errors, 3%, 6%, or JBM profiles | MOS-LQO(FX/FX) - MOS-LQO(FL/FL) | MOS-LQO(FX/FX) - MOS-LQO(FX/FL) | MOS-LQO(FX/FX) - MOS-LQO(FL/FX) |

The distribution of the difference for the decoder under test should be similar to the distribution of the floating point standard decoder.

The histogram of the MOS-LQO difference can be plotted for the three scenarios (a)-b), a)-c), a)-d)).

One way of assessing the distribution is to look at the mean, standard deviation, 95% percentile and maximum value.

Another option is to plot the Cumulative Distribution Frequency (CDF) of the MOS-LQO difference. In this case the absolute difference is used.

#### 5.3.2.4 Thresholds and Criteria

From the MOS-LQO differences of the test condition reported in clause 5.3.2.3, the average, 95%, 99% and Maximum are computed for all bandwidths combined, as well as for each set of bandwidth condition. The number of test condition for each bandwidth and the total are summarized in Table 5.

Table 5: Number of test conditions per bandwidth

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bandwidth | NB | WB | WBIO | SWB | FB | All |
| Number | 136 | 236 | 216 | 192 | 161 | 941 |

For a possible conformance criteria, it is proposed to have thresholds on the Average, 95%, 99% and Maximum of the MOS-LQO differences for the three scenarios (A-B, A-C and A-D). Thresholds are defined for all conditions combined, as well as for each set of bandwidth conditions. Table 6 summarizes the various thresholds.

An implementation will be considered passing the MOS-LQO verification if all the average, 95 percentile, 99 percentile and maximum MOS-LQO differences are below the thresholds proposed in Table 6 for all conditions.

Table 6: Possible thresholds for MOS\_LQO difference

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| All | Average | 95% | 99% | Max |
| A-B | 0.002 | 0.04 | 0.07 | 0.12 |
| A-C | 0.002 | 0.02 | 0.04 | 0.06 |
| A-D | 0.002 | 0.04 | 0.08 | 0.17 |
| NB | Average | 95% | 99% | Max |
| A-B | 0.009 | 0.07 | 0.08 | 0.11 |
| A-C | 0.002 | 0.02 | 0.04 | 0.06 |
| A-D | 0.011 | 0.07 | 0.09 | 0.12 |
| WB | Average | 95% | 99% | Max |
| A-B | 0.002 | 0.04 | 0.07 | 0.08 |
| A-C | 0.002 | 0.02 | 0.04 | 0.06 |
| A-D | 0.002 | 0.04 | 0.07 | 0.17 |
| WBIO | Average | 95% | 99% | Max |
| A-B | 0.002 | 0.02 | 0.06 | 0.08 |
| A-C | 0.002 | 0.02 | 0.03 | 0.07 |
| A-D | 0.002 | 0.02 | 0.03 | 0.08 |
| SWB | Average | 95% | 99% | Max |
| A-B | 0.002 | 0.04 | 0.06 | 0.12 |
| A-C | 0.003 | 0.03 | 0.04 | 0.04 |
| A-D | 0.002 | 0.04 | 0.07 | 0.08 |
| FB | Average | 95% | 99% | Max |
| A-B | 0.006 | 0.04 | 0.07 | 0.08 |
| A-C | 0.005 | 0.04 | 0.05 | 0.06 |
| A-D | 0.005 | 0.04 | 0.06 | 0.08 |

These proposed thresholds have been obtained by using the good implementations tested in Clause 6:

- EVS C80 Reference code

- EVS C90 Reference code

- EVS CA0 Reference code

- EVS C80 code compiled for Atom 32 bits platform using icc without optimization

- EVS C80 code compiled for Atom 32 bits platform using icc with normal optimization level

- EVS C90 code compiled for Mac\_OS 64bits using clang with -o2 optimization

- EVS C90 code compiled for Xeon 64 bits platform using gcc with -o2 optimization

For each case the maximum score difference among the six implementations, rounded to higher digit, was used. For average the threshold is rounded to the 3rd next digit (e.g. 0.0082 -> 0.009), and for the other metrics the threshold is rounded to the 2nd next digit (e.g. 0.036 -> 0.04). The statistics for each of the 7 implementations are presented in Table 7.

Table 7: Statistics of MOS-LQO difference

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| All | A-B | | | | A-C | | | | A-D | | | |
| A-B | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max |
| 3GPP C80 | 0.001 | 0.034 | 0.061 | 0.108 | 0.001 | 0.020 | 0.035 | 0.061 | 0.001 | 0.036 | 0.059 | 0.114 |
| 3GPP C90 | 0.001 | 0.036 | 0.061 | 0.082 | 0.001 | 0.020 | 0.036 | 0.063 | 0.001 | 0.037 | 0.064 | 0.083 |
| 3GPP CA0 | 0.001 | 0.038 | 0.061 | 0.082 | 0.001 | 0.019 | 0.036 | 0.059 | 0.001 | 0.037 | 0.064 | 0.083 |
| Opt\_None | 0.001 | 0.034 | 0.059 | 0.120 | 0.001 | 0.020 | 0.035 | 0.061 | 0.000 | 0.034 | 0.057 | 0.081 |
| Opt\_Quality | 0.001 | 0.036 | 0.064 | 0.109 | 0.001 | 0.018 | 0.035 | 0.056 | 0.001 | 0.034 | 0.075 | 0.162 |
| Xeon gcc\_o2 | 0.000 | 0.036 | 0.062 | 0.081 | 0.001 | 0.019 | 0.036 | 0.063 | 0.001 | 0.037 | 0.064 | 0.083 |
| Mac\_OS\_o2 | 0.001 | 0.036 | 0.062 | 0.082 | 0.001 | 0.019 | 0.036 | 0.063 | 0.001 | 0.038 | 0.065 | 0.083 |
| NB | A-B | | | | A-C | | | | A-D | | | |
| A-B | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max |
| 3GPP C80 | 0.007 | 0.051 | 0.072 | 0.108 | 0.002 | 0.020 | 0.033 | 0.046 | 0.010 | 0.057 | 0.075 | 0.114 |
| 3GPP C90 | 0.008 | 0.054 | 0.074 | 0.082 | 0.001 | 0.014 | 0.033 | 0.055 | 0.011 | 0.061 | 0.074 | 0.080 |
| 3GPP CA0 | 0.008 | 0.055 | 0.074 | 0.082 | 0.001 | 0.014 | 0.033 | 0.055 | 0.011 | 0.061 | 0.074 | 0.080 |
| Opt\_None | 0.006 | 0.050 | 0.060 | 0.075 | 0.002 | 0.020 | 0.034 | 0.038 | 0.007 | 0.053 | 0.068 | 0.081 |
| Opt\_Quality | 0.008 | 0.062 | 0.079 | 0.109 | 0.001 | 0.013 | 0.033 | 0.048 | 0.011 | 0.065 | 0.081 | 0.103 |
| Xeon gcc\_o2 | 0.008 | 0.054 | 0.073 | 0.081 | 0.001 | 0.013 | 0.033 | 0.055 | 0.010 | 0.061 | 0.075 | 0.079 |
| Mac\_OS\_o2 | 0.008 | 0.054 | 0.078 | 0.082 | 0.001 | 0.013 | 0.038 | 0.055 | 0.011 | 0.061 | 0.079 | 0.080 |
| WB | A-B | | | | A-C | | | | A-D | | | |
| A-B | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max |
| 3GPP C80 | -0.001 | 0.038 | 0.062 | 0.071 | -0.001 | 0.014 | 0.031 | 0.055 | 0.001 | 0.039 | 0.054 | 0.059 |
| 3GPP C90 | -0.003 | 0.030 | 0.053 | 0.061 | 0.000 | 0.017 | 0.033 | 0.056 | -0.001 | 0.035 | 0.059 | 0.083 |
| 3GPP CA0 | -0.002 | 0.032 | 0.053 | 0.061 | -0.001 | 0.017 | 0.032 | 0.058 | -0.001 | 0.035 | 0.059 | 0.083 |
| Opt\_None | -0.001 | 0.031 | 0.053 | 0.064 | -0.001 | 0.014 | 0.031 | 0.059 | 0.000 | 0.037 | 0.057 | 0.064 |
| Opt\_Quality | 0.000 | 0.032 | 0.047 | 0.071 | -0.001 | 0.016 | 0.033 | 0.056 | -0.002 | 0.028 | 0.057 | 0.162 |
| Xeon gcc\_o2 | -0.003 | 0.031 | 0.054 | 0.065 | 0.000 | 0.018 | 0.034 | 0.055 | -0.001 | 0.033 | 0.059 | 0.083 |
| Mac\_OS\_o2 | -0.003 | 0.029 | 0.055 | 0.059 | 0.000 | 0.016 | 0.036 | 0.056 | -0.001 | 0.035 | 0.062 | 0.083 |
| WBIO | A-B | | | | A-C | | | | A-D | | | |
| A-B | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max |
| 3GPP C80 | -0.004 | 0.014 | 0.043 | 0.061 | 0.000 | 0.009 | 0.020 | 0.061 | -0.004 | 0.006 | 0.023 | 0.044 |
| 3GPP C90 | -0.002 | 0.016 | 0.038 | 0.063 | 0.000 | 0.011 | 0.024 | 0.063 | -0.002 | 0.010 | 0.023 | 0.038 |
| 3GPP CA0 | -0.002 | 0.015 | 0.038 | 0.055 | 0.000 | 0.009 | 0.021 | 0.055 | -0.002 | 0.01 | 0.023 | 0.038 |
| Opt\_None | -0.003 | 0.017 | 0.048 | 0.063 | 0.000 | 0.011 | 0.022 | 0.061 | -0.003 | 0.008 | 0.023 | 0.063 |
| Opt\_Quality | -0.003 | 0.013 | 0.052 | 0.075 | 0.000 | 0.010 | 0.017 | 0.052 | -0.002 | 0.010 | 0.021 | 0.075 |
| Xeon gcc\_o2 | -0.002 | 0.015 | 0.038 | 0.063 | 0.000 | 0.011 | 0.025 | 0.063 | -0.002 | 0.010 | 0.023 | 0.038 |
| Mac\_OS\_o2 | -0.002 | 0.015 | 0.038 | 0.063 | 0.000 | 0.010 | 0.026 | 0.063 | -0.002 | 0.010 | 0.023 | 0.038 |
| SWB | A-B | | | | A-C | | | | A-D | | | |
| A-B | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max |
| 3GPP C80 | 0.000 | 0.027 | 0.050 | 0.080 | 0.002 | 0.023 | 0.034 | 0.036 | -0.002 | 0.022 | 0.046 | 0.076 |
| 3GPP C90 | -0.001 | 0.035 | 0.048 | 0.071 | 0.002 | 0.022 | 0.031 | 0.037 | -0.003 | 0.034 | 0.060 | 0.076 |
| 3GPP CA0 | -0.002 | 0.039 | 0.047 | 0.072 | 0.002 | 0.022 | 0.031 | 0.037 | -0.003 | 0.034 | 0.060 | 0.076 |
| Opt\_None | 0.000 | 0.029 | 0.059 | 0.120 | 0.002 | 0.023 | 0.034 | 0.037 | -0.002 | 0.032 | 0.062 | 0.062 |
| Opt\_Quality | 0.000 | 0.032 | 0.047 | 0.084 | 0.002 | 0.021 | 0.032 | 0.037 | 0.000 | 0.025 | 0.047 | 0.075 |
| Xeon gcc\_o2 | -0.002 | 0.036 | 0.048 | 0.071 | 0.002 | 0.022 | 0.030 | 0.036 | -0.003 | 0.034 | 0.060 | 0.076 |
| Mac\_OS\_o2 | -0.001 | 0.034 | 0.046 | 0.071 | 0.002 | 0.022 | 0.030 | 0.037 | -0.003 | 0.033 | 0.059 | 0.076 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FB | A-B | | | | A-C | | | | A-D | | | |
| A-B | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max | Avg | 95% | 99% | Max |
| 3GPP C80 | 0.005 | 0.034 | 0.057 | 0.060 | 0.003 | 0.030 | 0.038 | 0.041 | 0.004 | 0.029 | 0.053 | 0.066 |
| 3GPP C90 | 0.005 | 0.036 | 0.060 | 0.072 | 0.004 | 0.029 | 0.038 | 0.040 | 0.004 | 0.035 | 0.060 | 0.062 |
| 3GPP CA0 | 0.006 | 0.04 | 0.064 | 0.07 | 0.004 | 0.032 | 0.046 | 0.052 | 0.004 | 0.032 | 0.059 | 0.061 |
| Opt\_None | 0.003 | 0.032 | 0.053 | 0.061 | 0.004 | 0.031 | 0.038 | 0.041 | 0.002 | 0.022 | 0.045 | 0.070 |
| Opt\_Quality | 0.004 | 0.036 | 0.065 | 0.077 | 0.004 | 0.029 | 0.043 | 0.056 | 0.004 | 0.030 | 0.052 | 0.075 |
| Xeon gcc\_o2 | 0.005 | 0.035 | 0.060 | 0.072 | 0.004 | 0.029 | 0.038 | 0.042 | 0.004 | 0.035 | 0.060 | 0.062 |
| Mac\_OS\_o2 | 0.005 | 0.036 | 0.057 | 0.072 | 0.004 | 0.029 | 0.036 | 0.040 | 0.004 | 0.035 | 0.059 | 0.062 |

### 5.3.3 Maximum Loudness Difference

#### 5.3.3.1 General Methodology

This clause describes the calculation of the maximum loudness difference (MLD) per item. The procedure is adopted from the loudness calculation of PEAQ [11] using the Filter bank-based ear model. The following steps need to be processed:

- Filterbank (Annex 2 section 2.2.5 of [11]):

- subsample factor F changed to 16 for higher time resolution.

- Outer and Middle Ear Filtering (Annex 2 section 2.2.6 of [11])

- Frequency Domain Smearing (Annex 2 section 2.2.7 of [11])

- Rectification (Annex 2 section 2.2.8 of [11])

- Time Domain Smearing 1 - Backward Masking (Annex 2 section 2.2.9 of [11])

- Adding of Internal Noise (Annex 2 section 2.2.10 of [11])

- Time Domain Smearing 2 - Forward Masking (Annex 2 section 2.2.11 of [11])

- Loudness (Annex 2 section 3.3 of [11]):

- This section defines the specific loudness patterns  for  subbands and  time samples

- The specific loudness patterns are calculated for:

- reference signal 

- signal under test 

- Maximum Loudness Difference (MLD):

- The loudness difference  is calculated as follows:

- 

- The maximum loudness difference (MLD) for this item is then the maximum over all  time samples. Note that  has a granularity of 2 ms:

- 

Note: In the context of this study, the "loudness difference" is calculated as difference in some values (subtraction, not division).

#### 5.3.3.2 Proposed Encoder Conformance Test

The MLD metrics could be used to test the EVS floating-point codec [3] encoder implementation. Figure 5 shows the flow diagram of the proposed encoder conformance test:



Figure 5: Flow diagram for the encoder test using MLD Loudness Difference metric

All encoder test vectors from TS 26.444 will be encoded using the Float implementation under test. The bit-stream obtained will be then decoded using the 3GPP reference Float decoder from TS 26.443 to obtain the test signals. The test signals will then be compared with the decoded outputs from TS 26.444 according to method described in Clause 5.3.3.1. The reference signals are already available as part of TS 26.444 and therefore do not need to be generated. Since the loudness tool in the presented form operates on 48kHz sample rate only, additional resampling is applied before processing.

For a conformance test a global limit for MLD could be defined, which is expected not be exceeded by any time sample in each test vector. Setting such threshold is for further study.

# 6 Results

## 6.1 Experiment A

### 6.1.1 Compiler Options

In this experiment the code from TS 24.443 was compiled with various optimization levels to evaluate the sensitivity of the conformance tools. Intel compiler is used with three levels of optimization:

- Opt\_None: the code was compiled without any optimization.

- Opt\_Quality: the code was compiled with various optimization level depending on the file and functions to provide best computational performance while insuring quality.

- Opt\_Agg: the code was compiled with a very aggressive setting for computation performance, without checking on the possible consequences on quality.

The tests were done using a 32 bits version of the Atom platform.

The 3GPP floating-point C80 reference code was also used as a reference.

The methodology described in clause 5.2.3 was used to compute the difference in MOS-LQO scores. POLQA version 2.4 was used to compute the MOS-LQO scores. The POLQAswb mode was used with the level adjustment turn off. Results are reported in clause 6.1.3.

### 6.1.2 MOS-LQO Results

Table 8 summarizes the results obtained for the 3 compiler version as well as the result obtained for the 3GPP C80 code (executable from TS 24.443). The average, Min and Max values, Standard deviation as well as the 95% percentile are displayed.

Table 8: Summary of MOS-LQO differences

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Metric | Opt\_None | Opt\_Quality | Opt\_Agg | 3GPP C80 |
| a) - b) | AVG | 0.001 | 0.001 | 0.035 | 0.001 |
| MIN | -0.07 | -0.1 | -0.058 | -0.071 |
| MAX | 0.12 | 0.109 | 0.529 | 0.108 |
| STD | 0.019 | 0.02 | 0.098 | 0.019 |
| 95% | 0.034 | 0.036 | 0.281 | 0.034 |
| a) - c) | AVG | 0.001 | 0.001 | 0.022 | 0.001 |
| MIN | -0.065 | -0.039 | -0.073 | -0.068 |
| MAX | 0.061 | 0.056 | 0.383 | 0.061 |
| STD | 0.01 | 0.01 | 0.059 | 0.01 |
| 95% | 0.02 | 0.018 | 0.143 | 0.019 |
| a) - d) | AVG | 0.000 | 0.001 | 0.044 | 0.001 |
| MIN | -0.064 | -0.090 | -0.078 | -0.07 |
| MAX | 0.081 | 0.162 | 0.522 | 0.114 |
| STD | 0.018 | 0.02 | 0.099 | 0.019 |
| 95% | 0.034 | 0.034 | 0.286 | 0.036 |

Figures 6 to 8 show the histograms of the MOS-LQO difference for the three cases, a) - b), a) - c) and a) - d). The last point on the graph (difference above 0.18) represents accumulation between 0.18 and the maximum value.

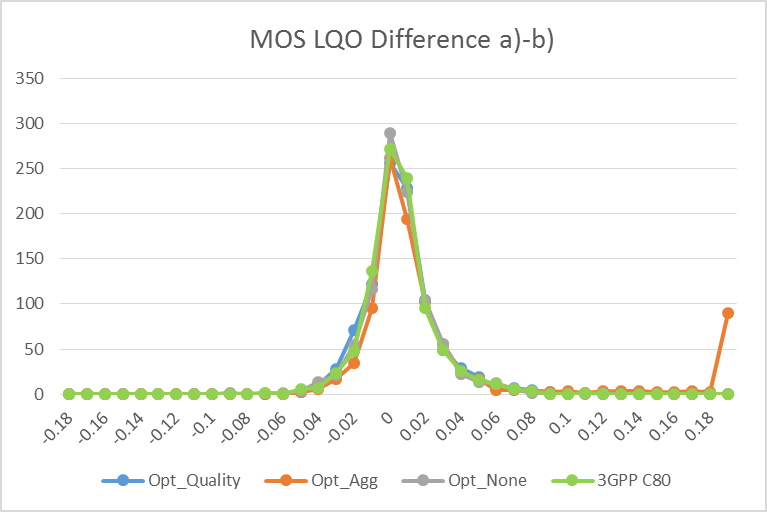


Figure 6: Histogram for a)-b) test case

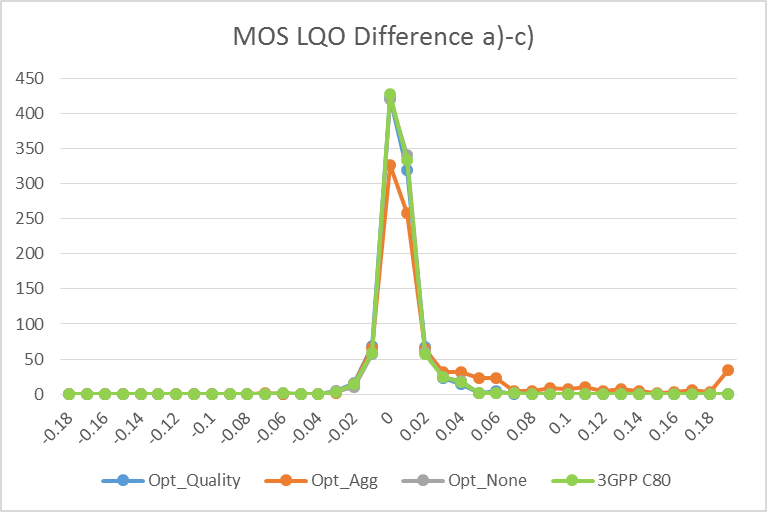


Figure 7: Histogram a) - c) test case

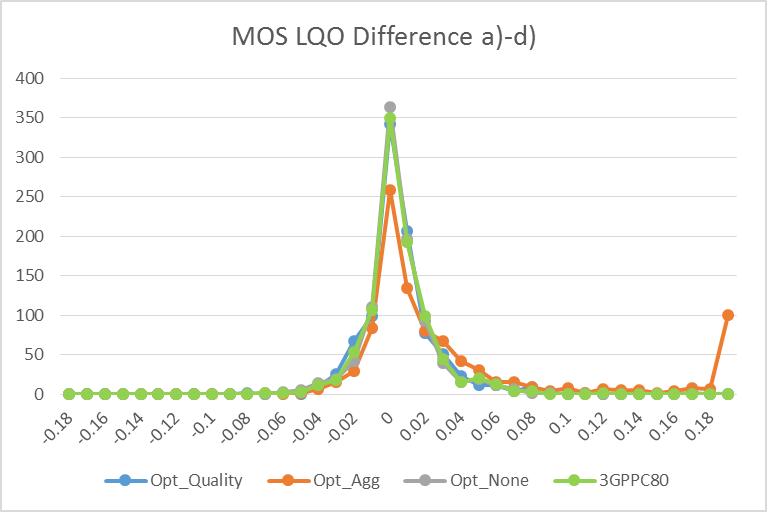


Figure 8: Histogram a) - d) test case

Figures 9 to 11 show the Cumulative Distribution Frequency (CDF) of the absolute MOS-LQO difference for the three cases, a) - b), a) - c) and a) - d).

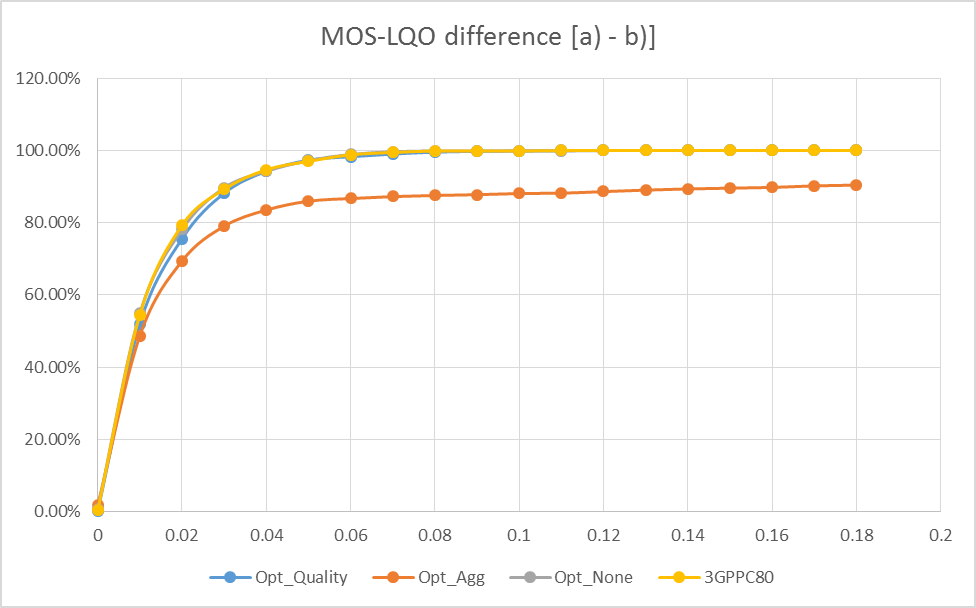


Figure 9: CDF plot of MOS-LQO difference for a)-b)

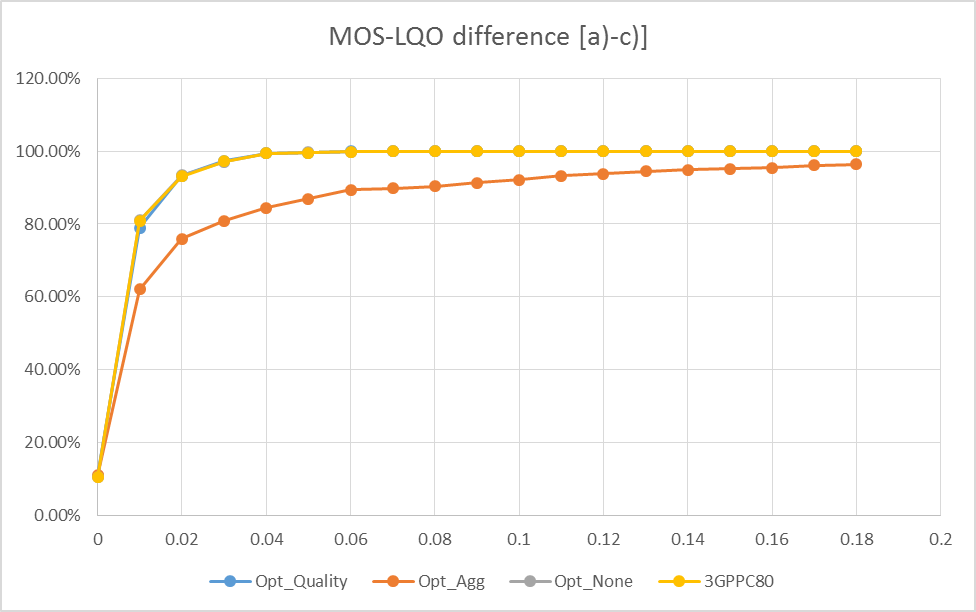


Figure 10: CDF plot of MOS-LQO difference for a)-c)

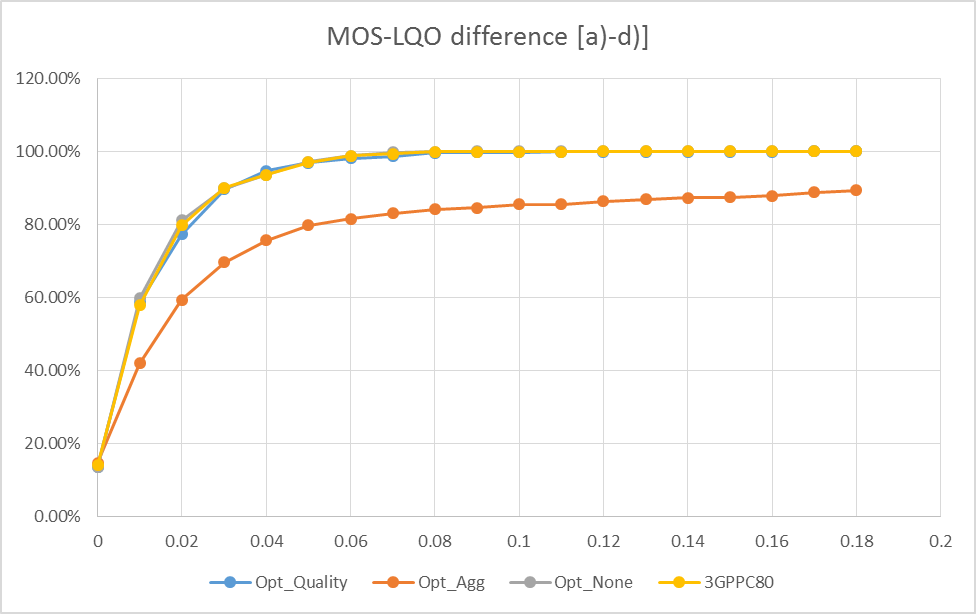


Figure 11: CDF plot of MOS-LQO difference for a)-d)

As it can be seen the results for Opt\_None and Opt\_Quality are very close to the 3GPP float and could be considered similar to 3GPP fixed point version.

However, Opt\_Agg shows some clear outliers in the results. The outliers are not constrained to a particular mode or bandwidth but are present in all the experiments.

## 6.2 Experiment B

### 6.2.1 Compiler Options

The code has been compiled for x86\_64 macOS 10.12.4 with various optimization levels to evaluate the sensitivity of the conformance tools. Note that gcc calls on macOS are mapped to clang, in this case clang 4.2.1.

Two levels of optimization were used:

- O2: the code was compiled with the gcc O2 option, which should improve performance without affecting output.

- Ofast: the code was compiled with gcc Ofast setting for computation performance, without checking on the possible consequences on quality.

First, Version C80 of the code was compiled with two options above, producing two code called O2 and Ofast\_v1 in this experiment.

Then the EVS FL reference code was changed to include the code improvement of denormal operation in generate\_masking\_noise(). This code was compiled with Ofast option to obtain Ofast\_v2 code.

The EVS FL reference code was further change to patch inaccurate behaviour in re8\_k2y() and SWB\_BWE\_decoding(). This code was also compiled with Ofast option to obtain Ofast\_v3 code.

The methodology described in 5.2.3 was used to compute the difference in MOS-LQO scores for the four code version (O2, Ofast\_v1, Ofats\_v2, Ofast\_v3). POLQA version 2.4 was used to compute the MOS-LQO scores. The POLQAswb mode was used with the level adjustment turn off.

### 6.2.2 Results

Table 9 summarizes the results obtained for the four code versions.

Table 9: Summary of differences

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | O2 | Ofast\_v3 | Ofast\_v2 | Ofast\_v1 |
| a) - b) | AVG | 0.0005 | 0.0022 | 0.0022 | 0.1396 |
| MIN | -0.1136 | -0.0781 | -0.0760 | -0.0760 |
| MAX | 0.0819 | 0.1595 | 0.1595 | 0.9455 |
| STD | 0.0194 | 0.0224 | 0.0224 | 0.2242 |
| 95% | 0.0425 | 0.0482 | 0.0490 | 0.5734 |
| a) - c) | AVG | 0.0011 | 0.0026 | 0.0331 | 0.1869 |
| MIN | -0.0341 | -0.0411 | -0.0411 | -0.0304 |
| MAX | 0.0629 | 0.0607 | 0.5291 | 0.9913 |
| STD | 0.0102 | 0.0114 | 0.0850 | 0.2218 |
| 95% | 0.0240 | 0.0269 | 0.1994 | 0.5685 |
| a) - d) | AVG | 0.0009 | 0.0015 | 0.0191 | 0.0221 |
| MIN | -0.0928 | -0.0959 | -0.0959 | -0.0959 |
| MAX | 0.0829 | 0.1677 | 0.3737 | 0.3737 |
| STD | 0.0195 | 0.0209 | 0.0577 | 0.0616 |
| 95% | 0.0444 | 0.0453 | 0.1098 | 0.1205 |

Unilateral CDFs of the difference MOS-LQO scores are plotted in Figures 12, 13 and 14, for the cases a)-b), a)-c) and a)-d), respectively. Regions were the CDF reaches 100% are indicated with oversized markers. When the oversized marker is at only 0.1, this indicates overload.

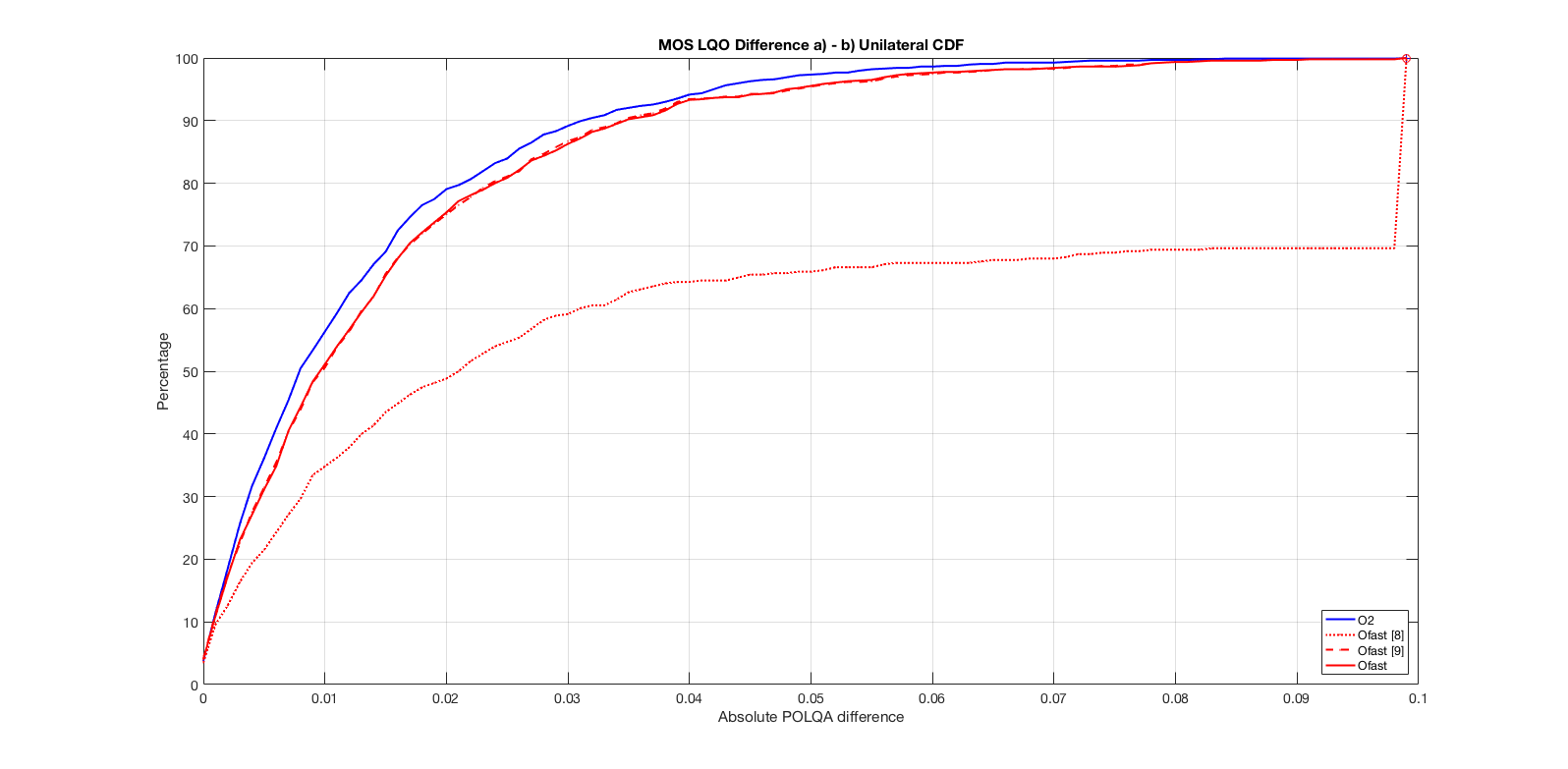


Figure 12: Unilateral CDF for a) - b) test case for O2 (blue), Ofast\_v1 (light red), Ofast\_v2 (dotted red) and Ofast\_v3 (dark red)

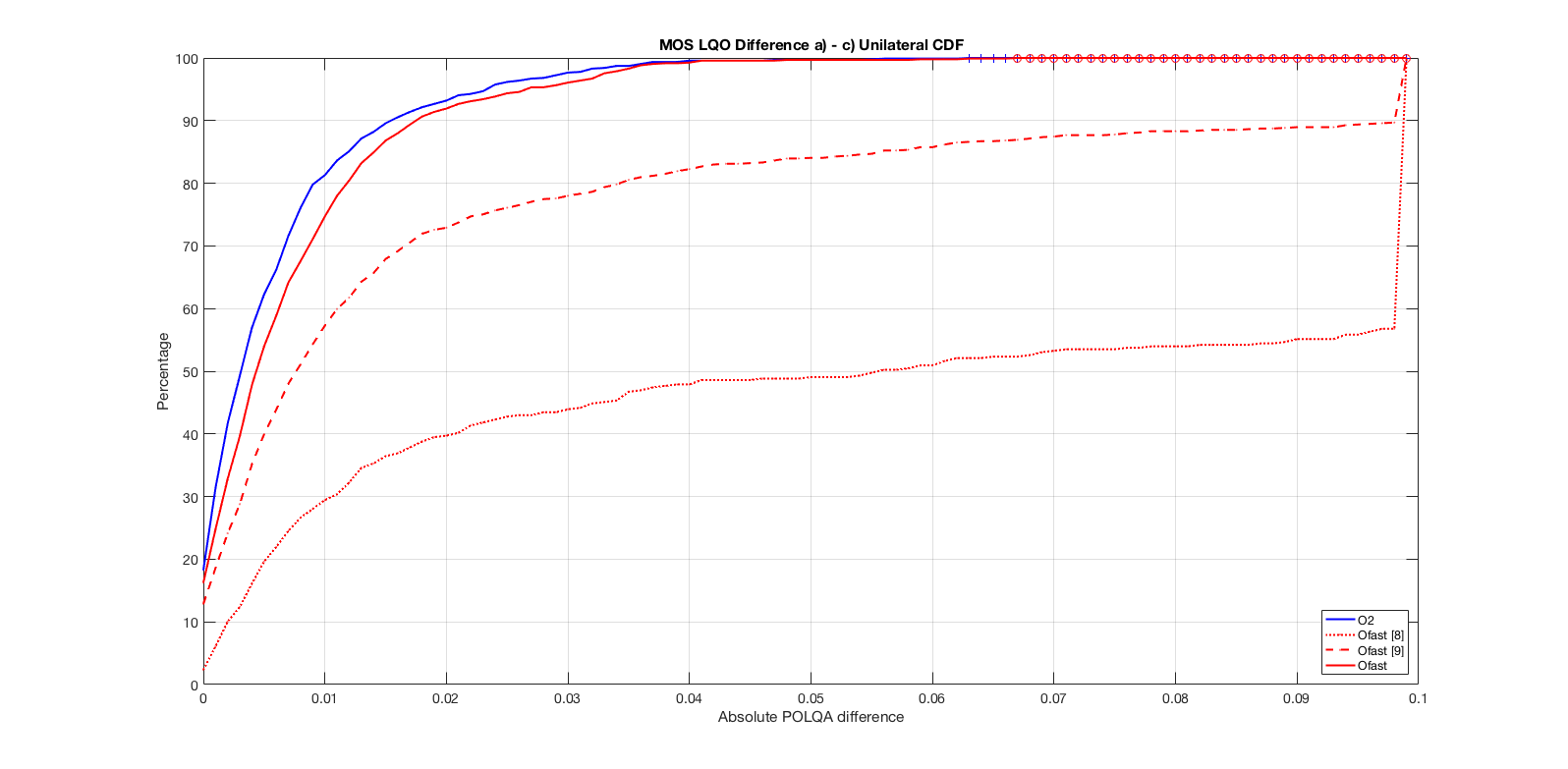


Figure 13: Unilateral CDF for a) - c) test case for O2 (blue), Ofast\_v1 (light red), Ofast\_v2 (dotted red) and Ofast\_v3 (dark red)

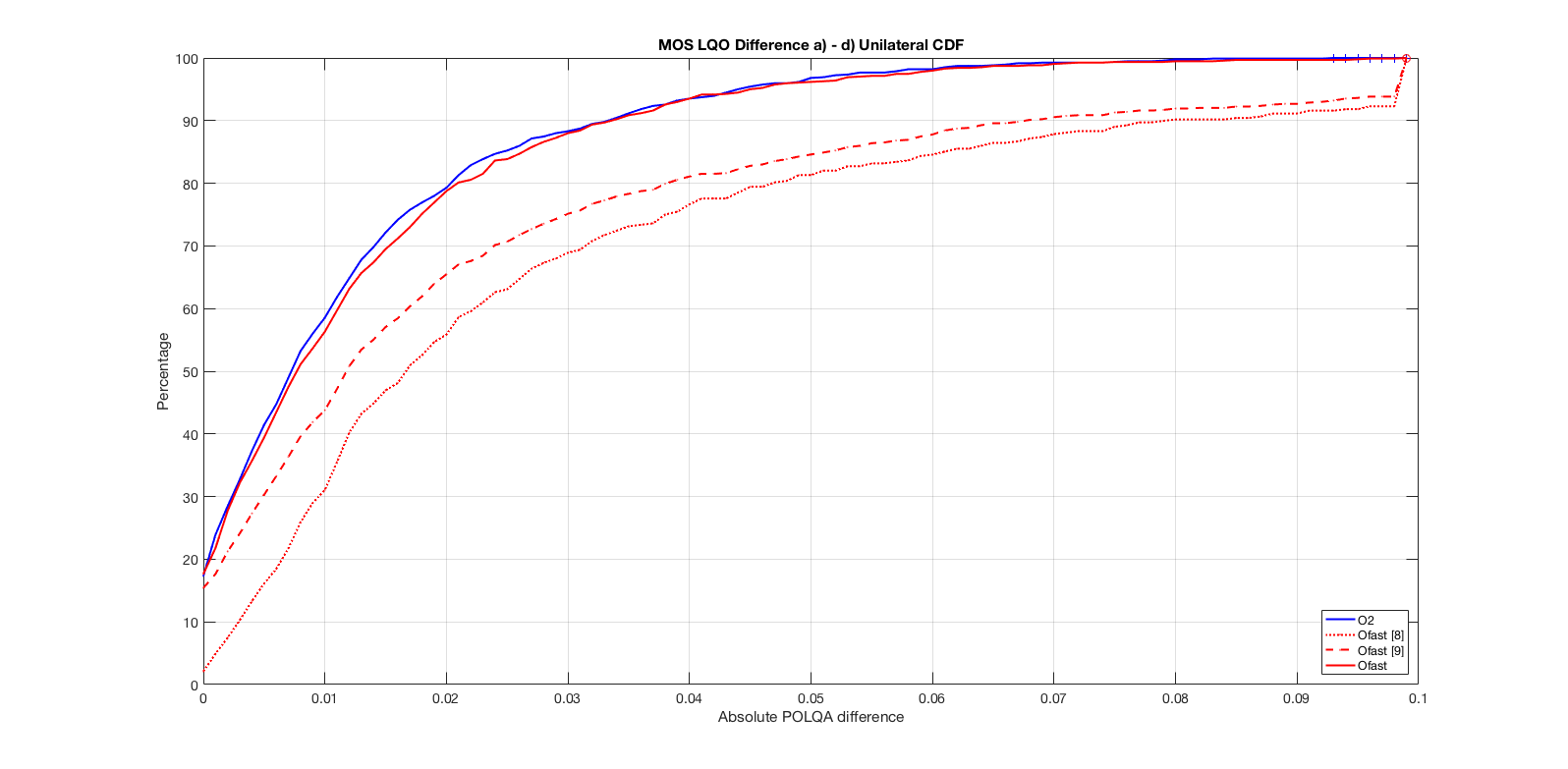


Figure 14: Unilateral CDF for a) - d) test case for O2 (blue), Ofast\_v1 (light red), Ofast\_v2 (dotted red) and Ofast\_v3 (dark red)

With Ofast\_v1, obvious outliers are observed in all three scenarios.

With Ofast\_v2 better result are observed than with Ofast\_v1. The results for O2 and Ofast\_v2 are similar in the a) - b) test case, but that Ofast\_v2 drops off for the cross-connect cases. This indicates that the deviations produced by Ofast\_v2 compilation are "balanced" between the encoder and decoder - but manifest as unacceptable degradations when connecting to the fixed point encoder/decoder.

The results for Ofast\_v3 are much improved compare to Ofast\_v2. This indicates that the patches (re8\_k2y() and SWB\_BWE\_decoding()) addressed the bulk of the FX/FL interoperability issues observed in Ofast\_v2 . Examining the results in detail, however, indicates that there are still a small number of suspicious results, and so this patched Ofast\_v3 implementation should be considered to fail conformance. Thus, this configuration represents a valuable reference for evaluating conformance criteria.

### 6.2.3 Restricted Results

It can be observed that, for the O2 case, channel error and DJB test conditions display greater variance and outliers in MOS-LQO scores relative to other conditions. A restricted test that excludes them and focuses on clean channel conditions has been conducted. This allows the consideration of strict criteria that give very high confidence that the core coding modes are implemented correctly. For completeness, this should be combined with a second set of criteria covering all conditions, to ensure that mandatory PLC functionality is correctly implemented. The results for this restricted test are summarized in Table 10. Note that the maximum, standard deviation, and 95% intervals for the O2 cases all shrink significantly relative to Table 9.

Table 10: Summary of differences - excluding PLC and DJB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | O2 | Ofast\_v3 | Ofast\_v2 | Ofast\_v1 |
| a) - b) | AVG | -0.0014 | -0.0011 | -0.0011 | 0.1223 |
| MIN | -0.0492 | -0.0507 | -0.0507 | -0.0449 |
| MAX | 0.0508 | 0.0877 | 0.0877 | 0.6659 |
| STD | 0.0132 | 0.0156 | 0.0156 | 0.2186 |
| 95% | 0.0284 | 0.0326 | 0.0326 | 0.5734 |
| a) - c) | AVG | 0.0017 | 0.0034 | 0.0498 | 0.2074 |
| MIN | -0.0292 | -0.0215 | -0.0215 | -0.0186 |
| MAX | 0.0557 | 0.0589 | 0.5291 | 0.6391 |
| STD | 0.0092 | 0.0093 | 0.1102 | 0.2228 |
| 95% | 0.0230 | 0.0227 | 0.3472 | 0.5720 |
| a) - d) | AVG | -0.0018 | -0.0019 | 0.0259 | 0.0354 |
| MIN | -0.0492 | -0.0526 | -0.0526 | -0.0526 |
| MAX | 0.0649 | 0.0590 | 0.3737 | 0.3737 |
| STD | 0.0119 | 0.0133 | 0.0733 | 0.0792 |
| 95% | 0.0241 | 0.0279 | 0.2141 | 0.2236 |

Unilateral CDFs of the difference MOS-LQO scores are plotted in Figures 15, 16 and 17, for the cases a) - b), a) - c) and a) - d), respectively. Regions were the CDF reaches 100% are indicated with oversized markers. Oversized markers at 0.1 only indicate overload.

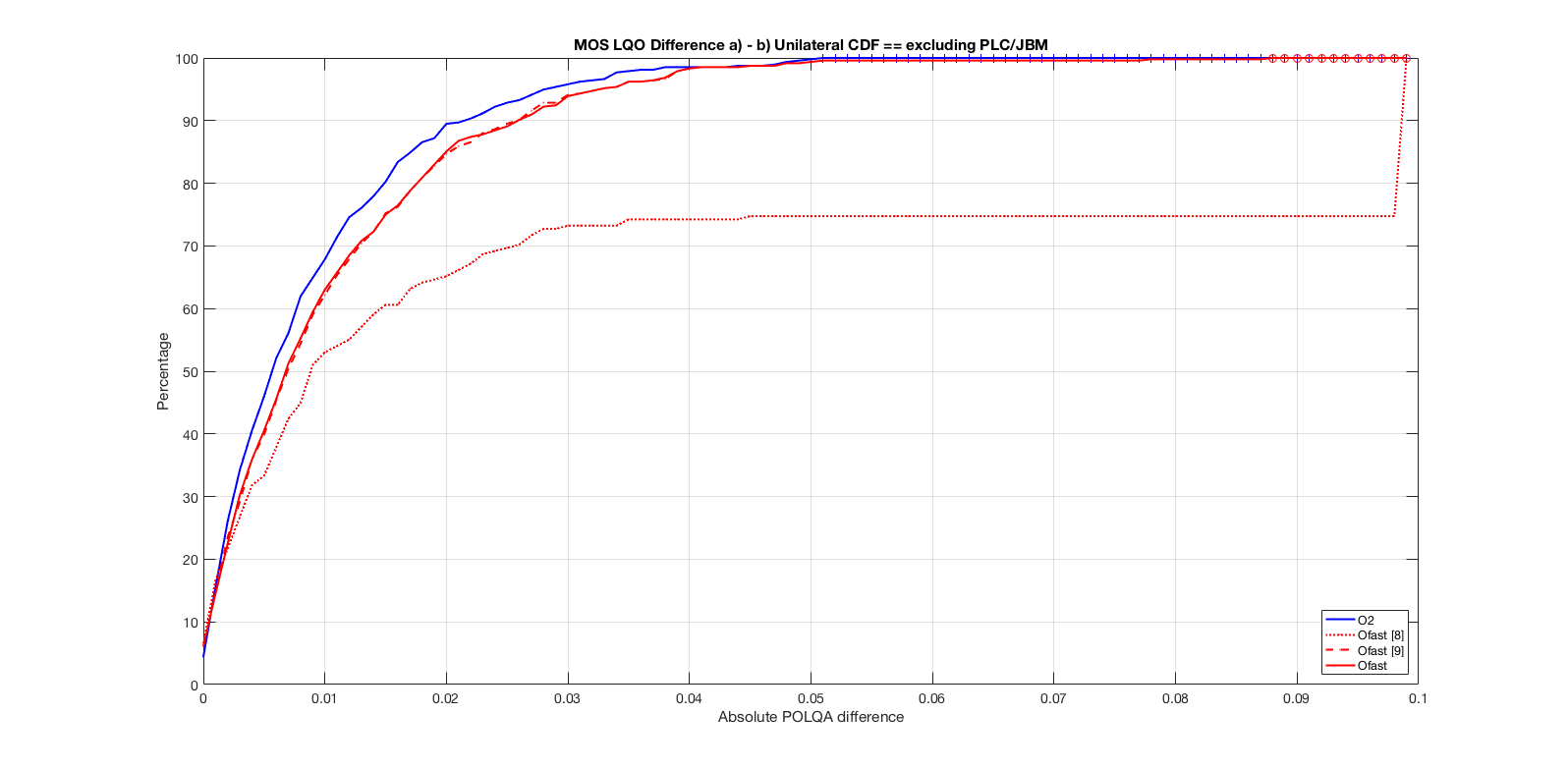


Figure 15: CDF for a) - b) test case for O2 (blue), Ofast\_v1 (light red), Ofast\_v2 (dotted red) and Ofast\_v3 (dark red)

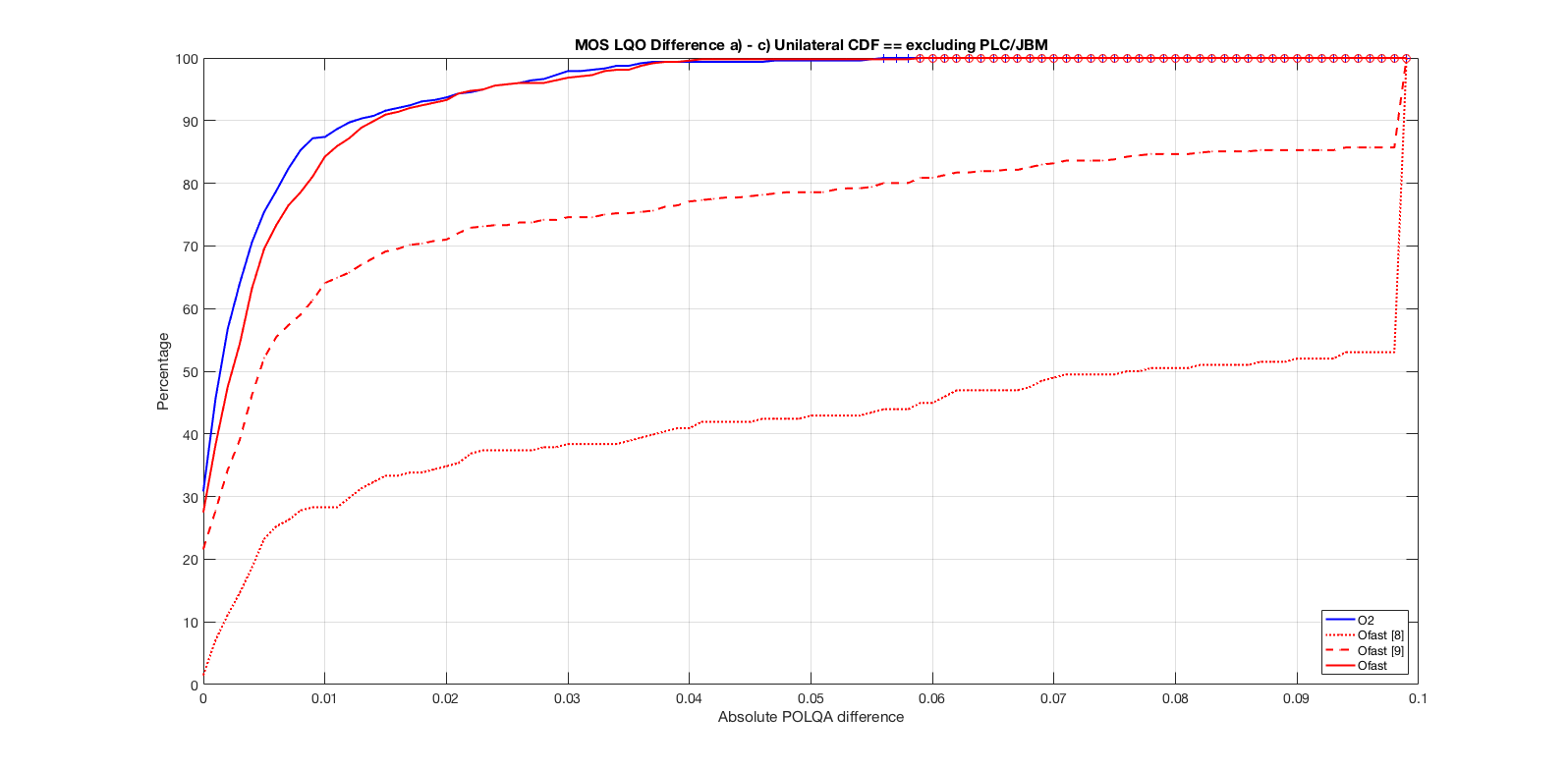


Figure 16: CDF for a) - c) test case for O2 (blue), Ofast\_v1 (light red), Ofast\_v2 (dotted red) and Ofast\_v3 (dark red)

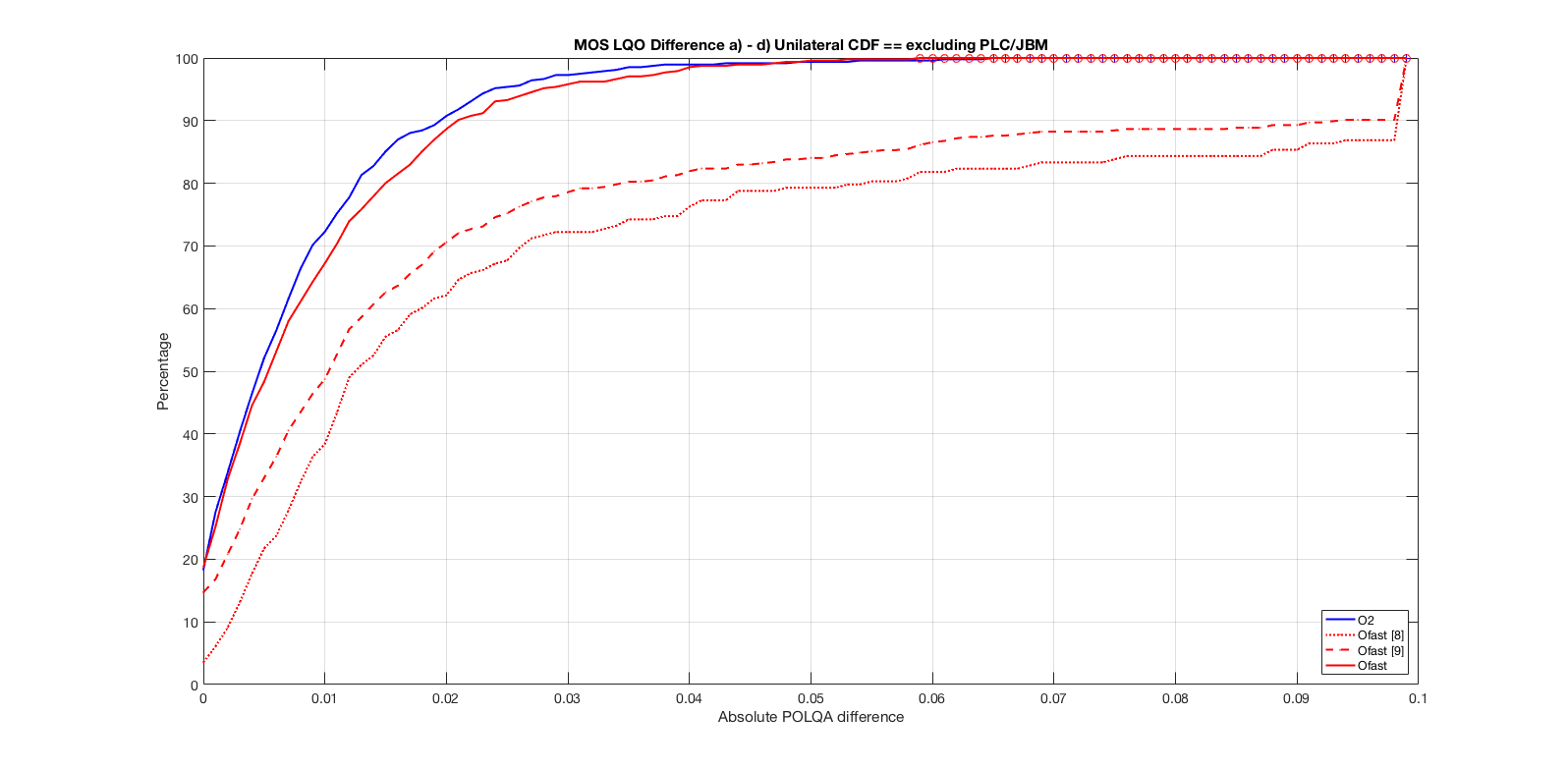


Figure 17: CDF for a) - d) test case for O2 (blue), Ofast\_v1 (light red), Ofast\_v2 (dotted red) and Ofast\_v3 (dark red)

Similarly to the results of clause 6.3.2, the Ofast\_v3 results are greatly improved compare to Ofast\_v2 and Ofast\_v1 but that the a) - b) case still exhibits inferior outlier performance relative to O2.

## 6.3 Experiment C

### 6.3.1 Compiler Options

#### 6.3.1.1 Introduction

Three compilers/platforms have been used for this study. In both cases the code from TS 26.443 (Version C80) has been compiled with various optimization levels to evaluate the sensitivity of the conformance tools.

#### 6.3.1.2 Icc Compiler on Atom Platform

This configuration is the same that was used to report result with MOS-LQO presented in clause 6.1. Three levels of optimization were used:

- Opt\_None: the code was compiled without any optimization.

- Opt\_Quality: the code was compiled with various optimization level depending on the file and functions to provide best computational performance while insuring quality.

- Opt\_Agg: the code was compiled with a very aggressive setting (-o3, -fast2) for computation performance, without checking on the possible consequences on quality.

The Atom platform used was 32 bits.

#### 6.3.1.3 Gcc compiler on Xeon platform

In this configuration, three level of optimization were used.

- O0: the code was compiled without any optimization.

- O2: the code was compiled with normal optimization level for speed and memory.

- O2+avx2: the code was compiled to take advantage of vector extensions math routine and can lead to variation in the arithmetic results. The avx2 option in gcc is -march=avx2.

The Xeon platform is a 64bits platform.

#### 6.3.1.4 Gcc compiler on ARM platform

In this configuration, two level of optimization were used with GCC compiler (version 6.3.0).

- O3: the code was compiled with -o3 option only.

- O3-fast-math: the code was compiled with -o3 and -fast-math option.

In this experiment a raspberry Pi board (model B generation 1) has been used to test a floating-point implementation using ARMv6.

### 6.3.2 Decoder Test Results

For this test, the EVS and AMR\_WBIO test vectors from TS 26.444 are used, representing 2675 test vectors.

The decoder test described in clause 5.2 was used to assess the different platform/compiler options. The various thresholds have been set to the examples values presented in clause 5.2.6, table 6.

Tables 11, 12 and 13 show the number of failed files in each cases for the two systems under test:

Table 11: Result for icc and Atom system

|  |  | Opt\_None | Opt\_Quality | Opt\_Agg |
| --- | --- | --- | --- | --- |
|  | Frames tested | 2349831 | 2349831 | 2349831 |
| RMS | Frames passing | 2227191 | 1118136 | 1072142 |
| Frames failing | 122640 | 1231695 | 1277689 |
| % passing | 94.8 | 47.6 | 45.6 |
| % failing | 5.2 | 52.4 | 54.4 |
| SNR | Frames passing | 121642 | 1230563 | 1160530 |
| Frames failing | 998 | 1132 | 117159 |
| % passing | 99.2 | 99.9 | 90.8 |
| % failing | 0.8 | 0.1 | 9.1 |
| Spectral Distortion | Frames passing | 923 | 864 | 25075 |
| Frames failing | 75 | 268 | 92084 |
| % passing | 92.5 | 76.3 | 21.5 |
| % failing | 7.5 | 23.7 | 78.5 |
| Overall % frames passing | | 99.997 | 99.989 | 96.081 |
| Overall % frames failing | | 0.003 | 0.011 | 3.92 |
| Number of files failing | | 2 | 1 | 650 |
| Number of files passing | | 2673 | 2674 | 2025 |

The 2 files failing the opt\_none (T16\_6600\_16kHz.b10.OUT, T16\_dtx\_6600\_16kHz.b10.OUT) are the same condition with error impairment. These 2 files, as well as the reference test vectors from TS 26.444 are attached to this contribution.

For the Opt\_quality, the file failing is due to time shifting of the signal and contributes 186 of the total failing frames(T06\_dtx\_12650\_16kHz.dly\_error\_profile\_5.dat.netsimoutput.OUT). This file is a JBM test case in the AMRWB\_IO set of test vector.

The results from Table 8 are in correlation with the results reported in clause 6.2. Both approaches flag the Opt\_Agg as a non-conformant floating-point implementation.

Table 12: Result for gcc and Xeon system

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | -o0 | -o2 | -o2-avx2 |
|  | Frames tested | 2349831 | 2349831 | 2349831 |
| RMS | Frames passing | 1131386 | 1131464 | 1157459 |
| Frames failing | 1218445 | 1218367 | 1192372 |
| % passing | 48.1 | 48.1 | 49.2 |
| % failing | 51.9 | 51.9 | 50.8 |
| SNR | Frames passing | 1218443 | 1218298 | 1152506 |
| Frames failing | 2 | 69 | 39866 |
| % passing | 100 | 100 | 96.7 |
| % failing | 0 | 0 | 3.3 |
| Spectral Distortion | Frames passing | 1 | 64 | 37678 |
| Frames failing | 1 | 5 | 2188 |
| % passing | 50 | 92.7 | 94.5 |
| % failing | 50 | 7.3 | 5.5 |
| Overall % frames passing | | 100.000 | 100.000 | 99.907 |
| Overall % frames failing | | 0.00 | 0.00 | 0.09 |
| Number of files failing | | 0 | 0 | 83 |
| Number of files passing | | 2675 | 2675 | 2592 |

The number of passing files is 100%, even if not all the frames are passing for the -o0 and -o2 option as the thresholds used (Table 3) allows 0.5% failing frames per file.

The results of Table 9 show similar result as Table 8 in the sense that change in the arithmetic precision or execution will be flagged. A detailed analysis of the 83 failed vectors, shows that the majority of the failed vectors are 32 kHz and 48 kHz noisy speech files test vectors.

Table 13: Results for gcc on ARM platform

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | -o3 | -o3-fast-math |
|  | Frames tested | 2349830 | 2349829 |
| RMS | Frames passing | 1131118 | 87625 |
| Frames failing | 1218712 | 2262204 |
| % passing | 48.1 | 3.73 |
| % failing | 51.9 | 96.27 |
| SNR | Frames passing | 1218712 | 301747 |
| Frames failing | 0 | 1960457 |
| % passing | 100 | 13.34 |
| % failing | 0 | 86.66 |
| Spectral Distortion | Frames passing | 0 | 261677 |
| Frames failing | 0 | 1698780 |
| % passing | -- | 13.35 |
| % failing | -- | 86.65 |
| Overall % frames passing | | 100 | 27.7 |
| Overall % frames failing | | 0 | 72.3 |
| Number of files failing | | 0 | 2547 |
| Number of files passing | | 2675 | 128 |

It can be seen that with the more aggressive compiler settings (-fast-math), the number of frames and files failing increases significantly compared to the more conservative compiler setting (-o3). Based on the proposed method and example thresholds, this implementation would not be conformant with TS 26.443 [2].

It should be noted that the higher number of frames failing, compare to other results reported in Clause 6.3, seems to be due to unexpected sample delay introduced in several of the decoded files for the -o3-fast-math compiler setting. This sample delay could be the root cause of the difference in the number of frame tested.

### 6.3.3 Robustness Decoder Test Results

In the context of EVS floating point code fix, a temporary code was evaluated where the seed of the random generator in the decoder was changed. This code was tested only using the gcc+xeon system. The results are reported in Table 14.

Table 14: Result for gcc and Xeon system with code change

|  |  |  |  |
| --- | --- | --- | --- |
| Compiler option | -o0 | -o2 | -o2-avx2 |
| Number failed vectors | 235 | 235 | 274 |

It can be seen that independently of the compiler options, the decoder test flagged the code as non-conformant. It shows that the decoder test can detect some code changes.

## 6.4 Experiment D

### 6.4.1 Delta-MOS-LQO Behaviour for Mixed-Music Signals

A 10-second long mixed-music input was processed through the FL reference (REF) implementation and an FL test implementation. The FL test implementation TEST(Clip2.deg2.32k) includes e.g., certain optimizations related to parameter quantization, over the FL reference implementation REF (Clip2.deg1.32k), which introduced a clear, audible artifact as shown in Figure 18.

|  |  |
| --- | --- |
| Spectrogram of REF (Clip2.deg1.32k) | Spectrogram of TEST (Clip2.deg2.32k) |
|  |  |

Figure 18: Spectrogram of reference and degraded waveform

Table 15 shows the MOS-LQO scores for the two outputs using POLQA version 2.4.

Table 15: MOS-LQO scores for reference and degraded file

|  |  |  |  |
| --- | --- | --- | --- |
|  | REF (Clip2.deg1.32k) | TEST (Clip2.deg2.32k) | Delta-MOS-LQO |
| MOS-LQO Score | 4.2622 | 4.2662 | 0.004 |

Although this is a serious artifact and is clearly audible, Delta-MOS-LQO between these two samples is not noticeable at all, with a value of 0.004. Note that the nature of modification of source code in "Clip1.deg2.32k" is not relevant here. Rather the more important and relevant fact here is that POLQA tool only shows a negligible difference in the scores for two vastly different mixed-music signals.

### 6.4.2 Clean Speech Input Example

We present a clean speech input example below with relevant POLQA MOS LQO scores. 8 seconds long Super-Wideband clean speech input was bandpass filtered to match the required frequency range of POLQA (50 Hz to 14 kHz) and was processed through FL reference (REF) implementation and an FL test implementation. The FL test implementation TEST (Clip1.deg2.32k) includes certain optimizations related to parameter quantization, over the FL reference implementation REF (Clip1.deg1.32k).

|  |  |
| --- | --- |
| Spectrogram of REF Clip1.deg1.32k | Spectrogram of TEST Clip1.deg2.32k |
|  |  |

Figure 19: spectrogram of reference and degraded speech waveform

Table 16 shows the MOS-LQO scores for the two outputs using POLQA version 2.4.

Table 16: MOS-LQO scores for reference and degraded signal

|  |  |  |  |
| --- | --- | --- | --- |
|  | REF Clip1.deg1.32k | TEST Clip1.deg2.32k | Delta-MOS-LQO |
| MOS-LQO Score | 4.3888 | 4.3752 | 0.0136 |

Clip1.deg2.32k output has an annoying high-pitched chirp/whistle type artifact that is clearly audible, and it is shown in the spectrogram above. However, the Delta-MOS-LQO between the two cases is infinitesimal, at 0.0136.

Note that the nature of modification of source code in "Clip1.deg2.32k" here is not relevant. Rather the more important and relevant fact here is that POLQA tool only shows a negligible "Delta-MOS-LQO" score for two vastly different "clean speech" signals, which is the main category of signals intended to be used with POLQA.

Statistics from extending experiment D to a larger database of around 8.5 minutes of clean speech and the delta-POLQA are included in Table 17.

From the Table 17, it is clear that the delta-POLQA values are quite low while the subjective quality degradation is quite serious as shown in the spectrograms above.

Table 17: Experiment D: Delta-POLQA values between the Reference and Test signals

|  |  |  |
| --- | --- | --- |
|  |  | Clean speech (database including about 64 sentence pairs) |
| Delta-POLQA values | Average | 0.05425 |
| Std. dev | 0.04548 |
| Max. value | 0.0832 |
| 95 percentile | 0.04293 |

### 6.4.3 MOS LQO evaluation

The code change has been implemented using floating-point version C90 and tested using Linux.

Figure 20 shows the CDF of MOS-LQO difference for all conditions and use cases, and Table 18 reports the statistics of the MOS-LQO difference for the 2 codes (C90 and C90+AHEVS429\_D code change).

Table 18: Summary of MOS-LQO differences for all conditions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case | Min | Max | Mean | StdDev | Quantile\_95 | Quantile\_99 |
| A-B | -0.1138 | 0.0819 | 0.0006 | 0.0195 | 0.0359 | 0.0612 |
| A-C | -0.0538 | 0.0630 | 0.0011 | 0.0103 | 0.0198 | 0.0362 |
| A-D | -0.0928 | 0.0829 | 0.0009 | 0.0195 | 0.0373 | 0.0637 |
| A-B AHEVS-429\_D | -0.1138 | 0.1277 | 0.0106 | 0.0289 | 0.0668 | 0.0950 |
| A-C AHEVS-429\_D | -0.0538 | 0.1331 | 0.0109 | 0.0252 | 0.0631 | 0.0904 |
| A-D AHEVS-429\_D | -0.0928 | 0.0829 | 0.0009 | 0.0195 | 0.0373 | 0.0637 |

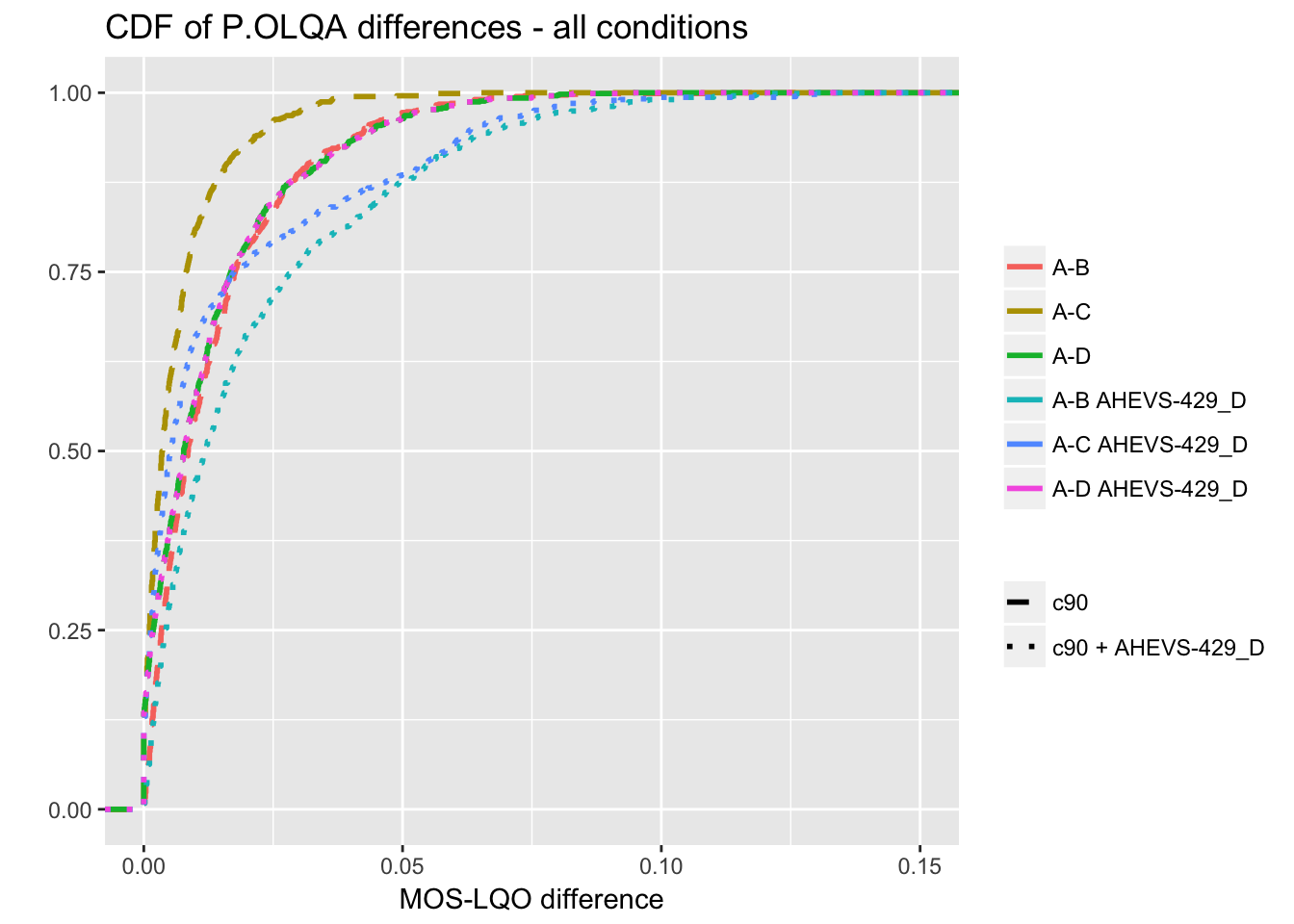


Figure 20: CDF of MOS-LQO differences for all conditions.

It can be seen that even if the code changes affects only SWB and FB the degradation in the CDF and statistic for the A-B and A-C case is noticeable. As the code change is only for decoder, the encoder case A-D is not affected. For example the Mean MOS-LQO difference is increased by a factor close to 20 for the A-B condition (testing both encoder and decoder float implementation).

As the code change only impacts higher bandwidth, the CDF and statistic for only the SWB conditions are reported in Figure 21 and Table 19.

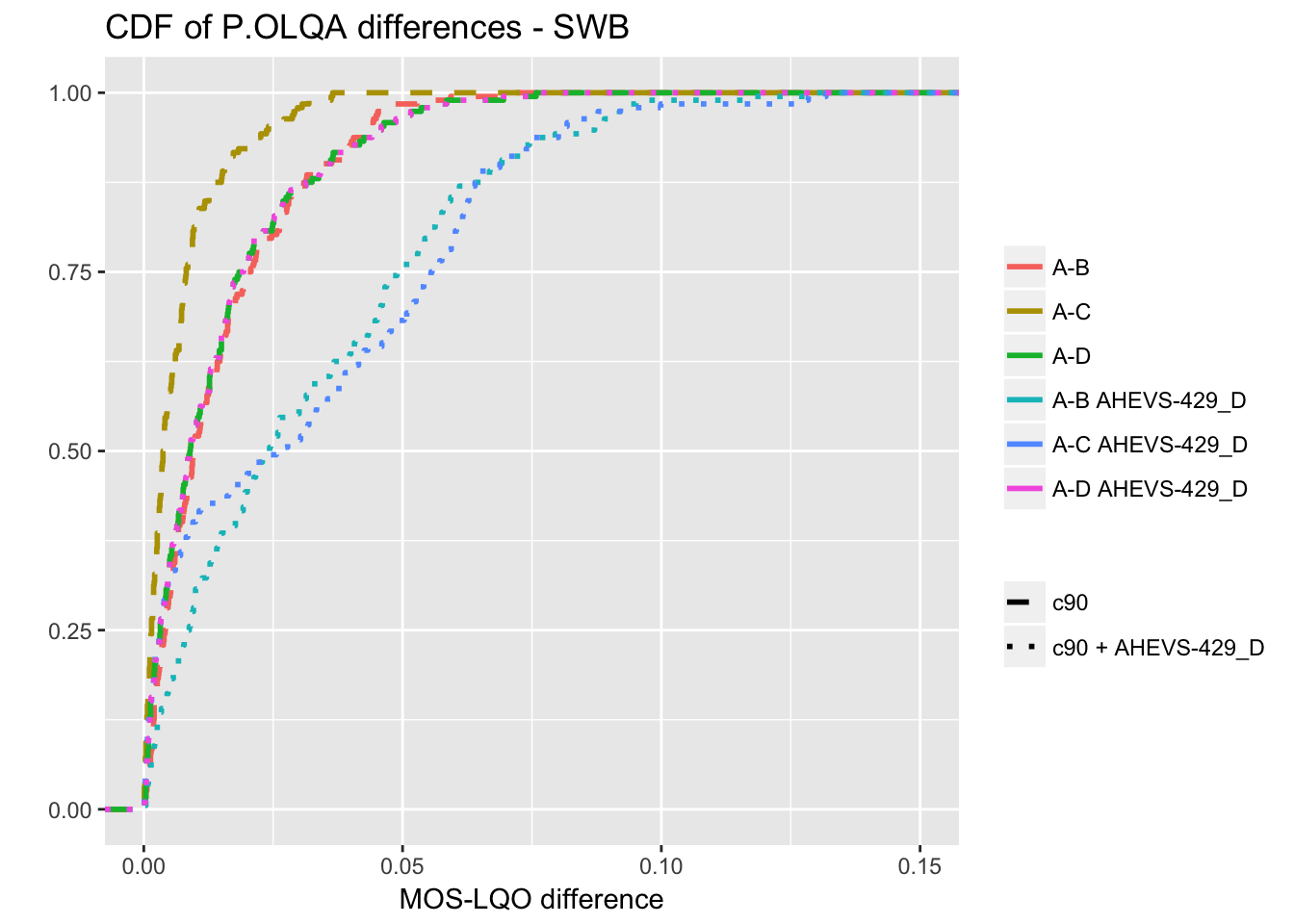


Figure 21: CDF plot of MOS-LQO difference for SWB condition

Table 19: Summary of MOS-LQO differences for SWB conditions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case | Min | Max | Mean | StdDev | Quantile\_95 | Quantile\_99 |
| A-B | -0.0528 | 0.0706 | -0.0013 | 0.0196 | 0.0351 | 0.0475 |
| A-C | -0.0319 | 0.0365 | 0.0019 | 0.0097 | 0.0224 | 0.0311 |
| A-D | -0.0568 | 0.0758 | -0.0026 | 0.0200 | 0.0340 | 0.0595 |
| A-B AHEVS-429\_D | -0.0444 | 0.1277 | 0.0264 | 0.0319 | 0.0865 | 0.0972 |
| A-C AHEVS-429\_D | -0.0127 | 0.1318 | 0.0298 | 0.0315 | 0.0812 | 0.1272 |
| A-D AHEVS-429\_D | -0.0568 | 0.0758 | -0.0026 | 0.0200 | 0.0340 | 0.0595 |

When the CDFs are computed for only the SWB the effect of the code change is even more noticeable. All the statistics for A-B and A-C used case show significant degradation.

Similar results are obtained in case of FB condition only.

### 6.4.4 Decoder test results

The code change has been implemented using floating-point version C90 and tested using Microsoft Visual Studio. In this test the decoder test described in Clause 5.2 was used. The various thresholds and criteria indicated in clause 5.2.6 were used.

The results indicate that 379 test vectors are failing. The detailed results are mentioned in Table 20. Note that the test is carried out on 2675 test vectors (The JBM test vectors were excluded).

Table 20: Statistics from the decoder test

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMS | SNR | Spectral Distortion |
| Number of frames tested | 2349830 | 211939 | 161210 |
| Number of frames passing | 2137891 | 50729 | 2818 |
| Number of frames failing | 211939 | 161210 | 158392 |
| Ratio of frames passing | 91 | 23.9 | 1.7 |
| Ratio of frames failing | 9 | 76.1 | 98.3 |

Overall 6.7% of the frames are failing.

An implementation with the proposed code change will not be conformant to TS 26.443 [2] according to the decoder conformance as currently described in Clause 5.2 of TR 26.843.

## 6.5 Experiment E

### 6.5.1 Delta-POLQA Limitations with Noisy Speech and Frame Erasures

In this clause, more examples are presented where source code modifications of the Reference EVS Floating Point implementation result in serious quality artifacts but show only negligible delta-POLQA values between the Reference and Test implementations. Figure 22 below shows example spectrograms that depict the signal artifacts and Table 17 provides the delta-POLQA analysis.

|  |  |
| --- | --- |
| Spectrogram of Reference signal | Spectrogram of Test signal |
|  |  |

Figure 22: Example spectrograms that depict the artifacts

From the Table 21, it is clear that the delta-POLQA values are quite low while the subjective quality degradation is quite serious.

Table 21: Experiment E: Delta POLQA values between the Reference and Test signals

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Noisy speech | FER 6% |
| **Delta-POLQA values** | Average | 0.00180 | 0.02734 |
| Std. dev | 0.01014 | 0.03610 |
| Max. Value | 0.04231 | 0.0713 |
| 95 percentile | 0.01589 | 0.04741 |

### 6.5.2 MOS-LQO Verification test

In this test MOS-LQO verification described in Clause 5.3.2 was carried out.

The code change has been implemented using floating-point version C90 and tested using Linux.

Figure 23 shows the CDF of MOS-LQO difference for all conditions and use cases, and Table 22 reports the statistics of the MOS-LQO difference for the 2 codes (C90 and C90+AHEVS429 code change).

Table 22: Summary of MOS-LQO differences for all conditions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case | Min | Max | Mean | StdDev | Quantile\_95 | Quantile\_99 |
| A-B | -0.1138 | 0.0819 | 0.0006 | 0.0195 | 0.0359 | 0.0612 |
| A-C | -0.0538 | 0.0630 | 0.0011 | 0.0103 | 0.0198 | 0.0362 |
| A-D | -0.0928 | 0.0829 | 0.0009 | 0.0195 | 0.0373 | 0.0637 |
| A-B AHEVS-429\_1 | -0.1138 | 0.0819 | 0.0021 | 0.0202 | 0.0395 | 0.0632 |
| A-C AHEVS-429\_1 | -0.0538 | 0.0630 | 0.0025 | 0.0123 | 0.0290 | 0.0429 |
| A-D AHEVS-429\_1 | -0.0928 | 0.0829 | 0.0009 | 0.0195 | 0.0373 | 0.0637 |

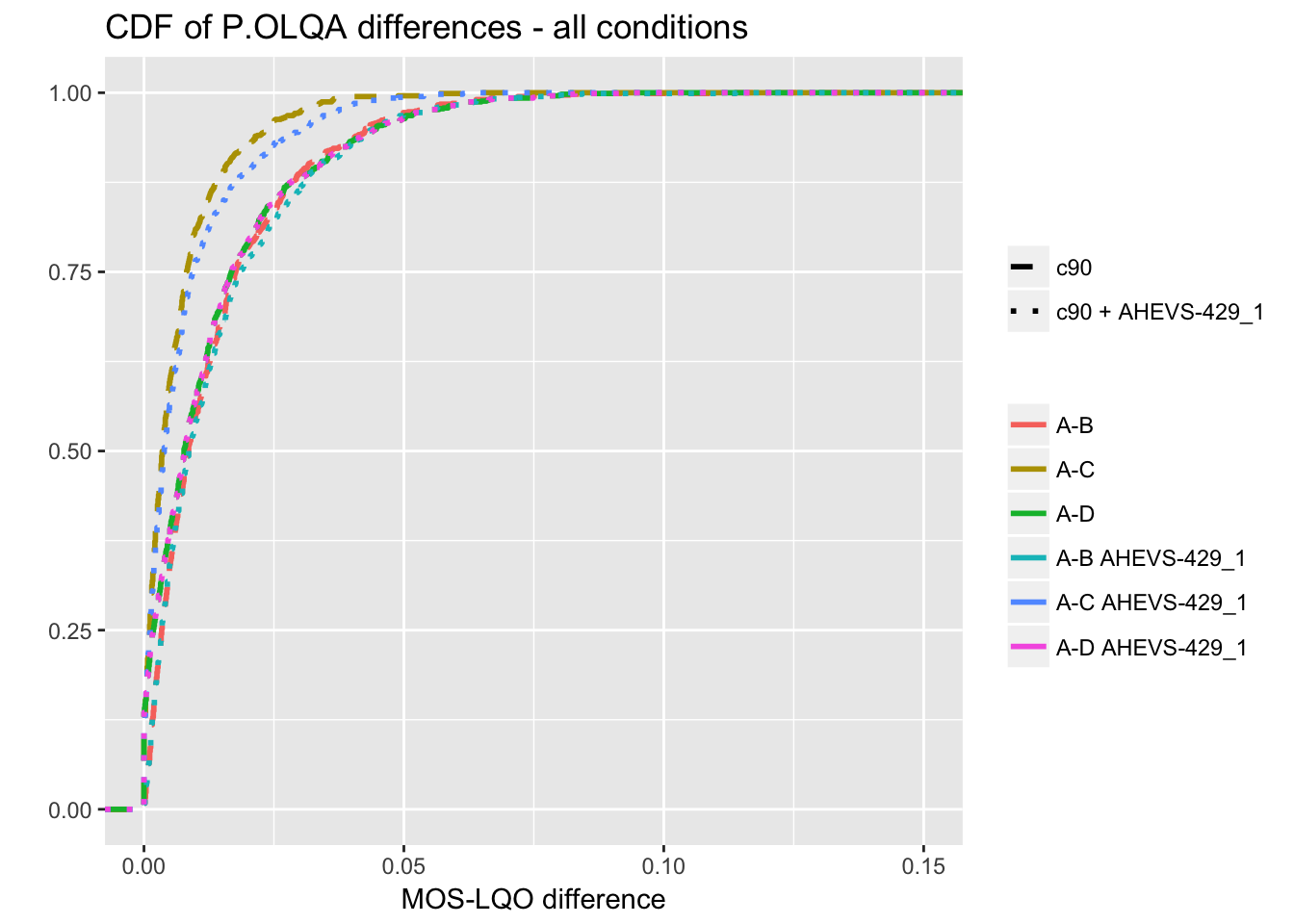


Figure 23: CDF of MOS-LQO differences for all conditions.

It can be see that A-C use case exhibits some difference in the CDF, but it is quite small. Similarly on the statistic the difference is small with and without the code change.

As the code change only impacts higher bandwidth, the CDF and statistic for only the SWB conditions are reported in Figure 24 and Table 23.

As can be seen in Figure 23, when plotted per condition the difference in CDF becomes more noticeable. The most significant degradation happened for the A-C case, but A-B test case also show some degradation. As the code change only impact the decoder the use case A-D is not affected.

This degradation in the POLQA scores is also visible in the statistics, for the 95% the POLQA difference increases from 0.022 to 0.039 for the A-C use case.

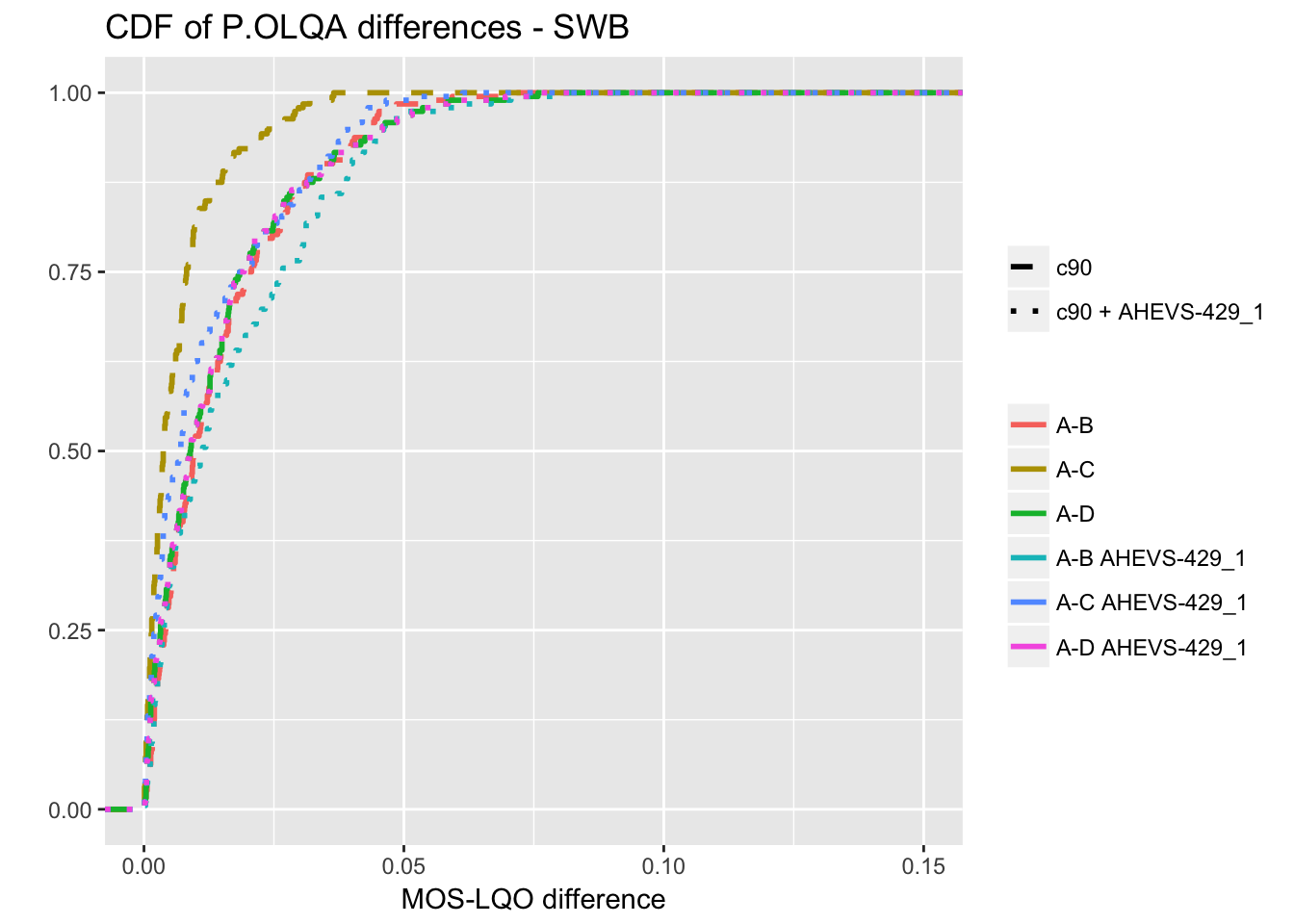


Figure 24: CDF plot of MOS-LQO difference for SWB condition

Table 23: Summary of MOS-LQO differences for SWB conditions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case | Min | Max | Mean | StdDev | Quantile\_95 | Quantile\_99 |
| A-B | -0.0528 | 0.0706 | -0.0013 | 0.0196 | 0.0351 | 0.0475 |
| A-C | -0.0319 | 0.0365 | 0.0019 | 0.0097 | 0.0224 | 0.0311 |
| A-D | -0.0568 | 0.0758 | -0.0026 | 0.0200 | 0.0340 | 0.0595 |
| A-B AHEVS-429\_1 | -0.0486 | 0.0787 | 0.0056 | 0.0225 | 0.0459 | 0.0671 |
| A-C AHEVS-429\_1 | -0.0319 | 0.0613 | 0.0086 | 0.0157 | 0.0389 | 0.0471 |
| A-D AHEVS-429\_1 | -0.0568 | 0.0758 | -0.0026 | 0.0200 | 0.0340 | 0.0595 |

By using a bandwidth approach, the POLQA scores can still discriminate a code change that impact a small numbers of the test vectors.

### 6.5.3 Decoder Test

In this test the decoder test described in Clause 5.2 was used. The test used the SNR criteria described in AHEVS-427 [4]. The various thresholds and criteria indicated in clause 5.2.6, Table 6 were used.

The code change has been implemented using floating-point version C90 and tested using Microsoft Visual Studio. This code change only affects the decoder output for the higher bandwidth (SWB & FB). Compare to the reference test vectors of TS 26.444, only 270 files, out of 2771 vectors, are none bit-exact.

The results indicates that 180 of test vectors are failing. The detailed results are mentioned in table 24.

Table 24: Statistics from the decoder test

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMS | SNR | Spectral Distortion |
| Number of frames tested | 2349830 | 57778 | 21170 |
| Number of frames passing | 2292952 | 36608 | 10987 |
| Number of frames failing | 57778 | 21170 | 10183 |
| Ratio of frames passing | 97.5 | 63.3 | 51.9 |
| Ratio of frames failing | 2.5 | 36.7 | 48.1 |

Overall 0.4% of the frames are failing.

An implementation with the proposed code change will not be conformant to TS 26.443 [2] according to the decoder conformance as currently described in Clause 5.2.

## 6.6 Experiment F

In this clause results with the loudness tool described in Clause 5.3.3 are presented. The encoder test vectors have been processed as described in Figure 5 of Clause 5.3.3.2, using the most recent test vectors, related to TS 26.443 in the currently available version v12.10.0.

The loudness tool has a scaling parameter Lmax, which sets the relation between the digital signal level and the assumed acoustic presentation level. The following procedure was used for determining Lmax

1) Select assumed acoustic presentation level 80.5 dBSPL for -26 dBov

2) Construct a mono calibration file with a 1 kHz pure sine tone representing 40 dBSPL (r.m.s. ‑66.5 dBov) and 16bit signed integer at 48kHz sampling frequency.

3) Repeat until the loudness model gives 1 sone for each channel:

a) Measure the loudness with the calibration file

b) Adjust Lmax

The application of this procedure gives a playback level Lmaxof 103.7535 dBSPL resulting in a loudness of 1.000066 sone. Lmax can be given as command line option (–l LEVEL) while the default value is 92 dBSPL, as described in ITU-R BS.1387-1 [2].

NOTE: The level 80.5 dBSPL was selected at 6 dB above the acoustic level expected for a UE with a receive loudness rating RLR of 8 dB (nominal for a binaural headset UE in TS 26.131), when receiving a signal a level of ‑16 dBm0.

The following 18 implementations (platforms & compiler optimization combination) have been tested:

- ARMv6 VFPv2 gcc-8 –Os

- ARMv7 NEON gcc-8 -O2

- ARMv7 VFPv3 gcc-8 –Os

- ARMv8 AArch32 gcc-8 –Os

- ARMv8 AArch64 gcc-8 –Os

- ARMv8 AArch64 gcc-8 -Os -ffp-contract=off

- i686 gcc-6 -O3

- i686 gcc-6 -O3 -mfpmath=sse -msse2

- i686 VS2017 Debug

- i686 VS2017 MinSizeRel

- i686 VS2017 Release

- x86-64 clang-4.0 -O3

- x86-64 clang-4.0 –Ofast

- x86-64 gcc-6 -O3

- x86-64 gcc-6 –Ofast

- x86-64 VS2017 Debug

- x86-64 VS2017 MinSizeRel

- x86-64 VS2017 Release

The boxplots indicate lower and upper quartile (left and right end of box) and median, the horizontal lines attached to the box indicate extreme values excluding outliers. Outliers are depicted as individual values.

In Figure 25 the box plot over all test vectors for a particular build is plotted for Lmax= 92dBSPL.

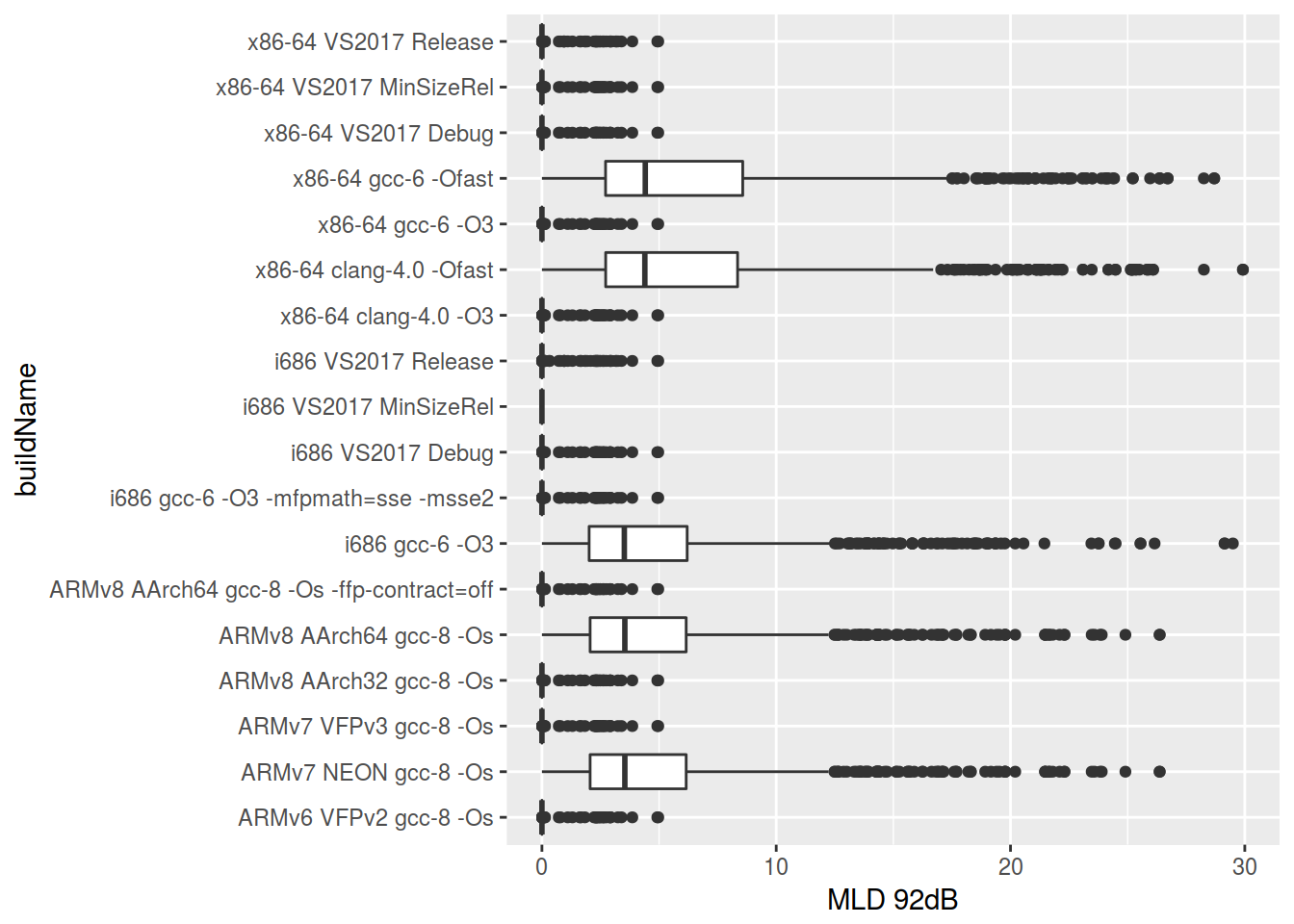


Figure 25: MLD Loudness Difference metric for various builds (Lmax= 92dBSPL)

In Figure 25 it can be observed that the builds "i686 VS2017 MinSizeRel" builds do not show any MLD. These builds are identical to the executables used for generating the test vectors. When deviating from the VS2017 Min Size Release builds to builds using an IEEE 754 compatible floating point unit, a maximum MLD of approx. 5 can be observed. Only for builds violating IEEE 754 ("ARMv7 NEON gcc-8 -O2", "ARMv8 AArch64 gcc-8 -Os", "i686 gcc-6 -O3", "x86-64 clang-4.0 -Ofast", "i686 gcc-6 -Ofast") significant higher MLD values can be observed.

In Figure 26 the box plot over all test vectors for a particular build is plotted for Lmax= 103.7535 dBSPL.

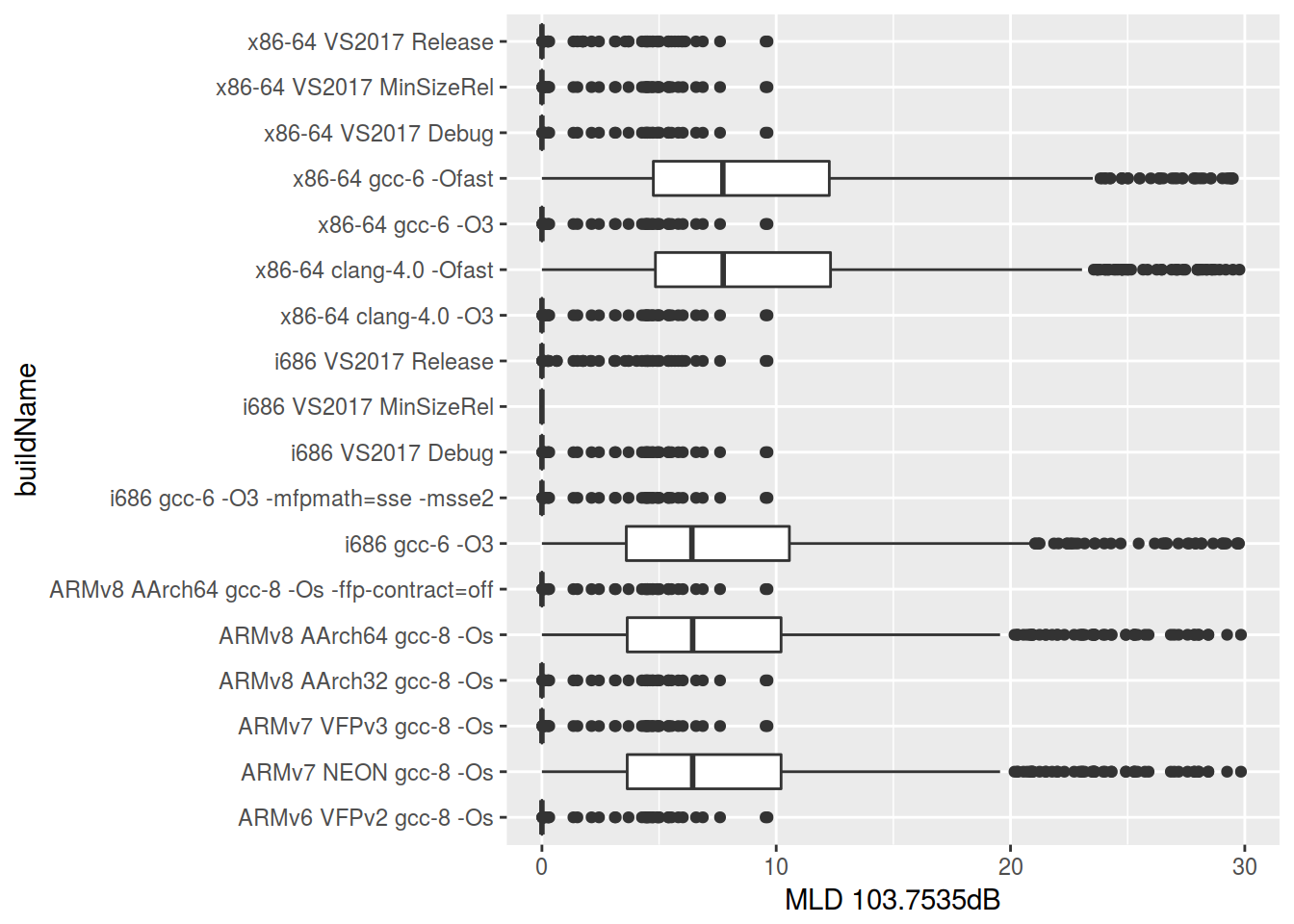


Figure 26: MLD Loudness Difference metric for various builds (Lmax= 103.7535 dBSPL)

In Figure 26 it can be observed that the builds "i686 VS2017 MinSizeRel" builds do not show any MLD. These builds are identical to the executables used for generating the test vectors. When deviating from the VS2017 Min Size Release builds to builds using an IEEE 754 compatible floating point unit, a maximum MLD of approx. 10 can be observed. Only for builds violating IEEE 754 ("ARMv7 NEON gcc-8 -O2", "ARMv8 AArch64 gcc-8 -Os", "i686 gcc-6 -O3", "x86-64 clang-4.0 -Ofast", "i686 gcc-6 -Ofast") significant higher MLD values can be observed.

Based on these results a single MLD threshold could be used as conformance criteria. For example a value of 6 could be selected when using Lmax= 92dBSPL or a value of 12 for Lmax= 103.7535 dBSPL.

# 7 Conformance Process

## 7.1 Description

For a floating-point implementation based on TS 26.443 to be conformant, it is proposed that the following three tests should be done and pass successfully:

- Decoder test based on Signal metrics described in Clause 5.2 comparing the CUT decoder implementation with TS 26.443 decoder.

- Encoder test (possibly based on Loudness metrics described in Clause 5.3.3) comparing CUT encoder implementation with TS 26.443 encoder.

- MOS-LQO verification based on POLQA described in Clause 5.3.2 comparing CUT implementation with TS 26.442 implementation.

Figure 27 presents an overview of the possible flow chart for EVS conformance process.



Figure 27: Flow chart for possible EVS float conformance process

# 8 Interoperability

## 8.1 Description

By using implementations being conformant to either TS 26.442 or TS 26. 443, there are two interoperability concerns that comes to mind.

The first concern is about interoperability between fixed-point and floating-point implementations. For a given release version of EVS, both implementations in TS 26.442 and TS 26.443 are tested together to minimize any interoperability issue that could be introduced by the code changes. Furthermore, interoperability relevant code parts including all bit-stream operations are included as fixed-point code into TS 26.443. Old versions of EVS implementations showing interoperability issues have been discarded. Based on these it could be assumed that any release of EVS fixed-point and floating-point standards are fully interoperable.

The second one arises from the belief that by not using bit-exact criteria two floating-point implementations could be non-interoperable. This is discussed in the next clause.

## 8.2 Interoperability Testing

Clauses 5.2 and 5.3 describe signal-based methods and perceptual-based methods for a conformance procedure for evaluating various EVS floating-point implementations.

In Clause 5.2, the decoder implementation on any given compiler is tested against the test vectors from 26.444 (corresponding to e.g., floating point 32-bit MSVC implementation). The decoder conformance procedure is depicted as in Figure 28.

In Clause 5.3, the encoder-decoder chain is proposed to be evaluated based on a delta MOS-LQO measure using the POLQA tool. The encoder-decoder conformance procedure is depicted in Figure 29.



Figure 28: Decoder conformance, where each of the decoder implementations on different compilers (e.g., N different compilers) verified based on the test vectors from TS 26.444

Figure 29: Encoder-decoder conformance, where each of the encoder/decoder implementations on different compilers (e.g., shown here for 2 compilers) verified against the Fixed (FX) implementation

The conformance procedure shown in Figure 28 evaluates only Float (FL) decoder implementations. The procedure is similar to what is typically followed in MPEG standards for evaluating decoder conformance, that serves streaming or playback type of applications (or decoder-only FL implementations in conversational applications).

For end-to-end conversational application, the conformance procedure shown in Figure 29 evaluates Encoder/Decoder chain for the three combinations, i.e., 1) FL\_Enc <-> FL\_Dec, 2) FX\_Enc <-> FL\_Dec, and 3) FL\_Enc <-> FX\_dec against FX\_Enc <-> FX\_Dec.

By combining both conformance procedures (figure 28 & 29), the CUT float implementation is evaluated against both the FX reference implementation and the FL reference implementation, minimizing potential interoperability issues. Furthermore, using both methods the coverage of test vectors is significantly increased, providing more confidence to the float conformance even if bit-exactness criteria are not used.

The conformance test of an implementation (fixed or float) should be done against the reference code and not against another custom implementation. If two implementations are conformant then they should not have interoperability issues

To show this point, the MOS-LQO verification describes in 5.3.2 of TR 26.843 have been run using two implementations considered conformant in Clause 6:

- The Atom 32 bits encoder with no compilation flag and based on C80 code

- The Xeon 64bits decoder using gcc -02 optimization and based on c90 code

The two codes represent some corner cases in the list of implementations currently tested.

The 3GPP C90 fixed point code was used for reference. Table 25 shows the results of the A-B use case, as well as the thresholds currently proposed in the TR.

Table25: Results for an interoperability test using MOS-LQO verification

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Enc\_Atom - Dec\_Xeon - RefCA0 | | | |  | Threshold | | | |
| Average | 95% | 0.99% | Max |  | Average | 95% | 99% | Max |
| 0.0010 | 0.0357 | 0.0569 | 0.1159 | All | 0.002 | 0.04 | 0.07 | 0.12 |
| 0.0058 | 0.0488 | 0.0611 | 0.0824 | NB | 0.009 | 0.07 | 0.08 | 0.11 |
| -0.0002 | 0.0301 | 0.0518 | 0.0587 | WB | 0.002 | 0.04 | 0.07 | 0.08 |
| -0.0030 | 0.0154 | 0.0488 | 0.0626 | WBIO | 0.002 | 0.02 | 0.06 | 0.08 |
| 0.0009 | 0.0325 | 0.0565 | 0.1159 | SWB | 0.002 | 0.04 | 0.06 | 0.12 |
| 0.0043 | 0.0340 | 0.0520 | 0.0598 | FB | 0.006 | 0.04 | 0.07 | 0.08 |

It can be see that all the MOS-LQO differences for all the bandwidths are below the proposed thresholds in Clause 5.3.2.4, Table 6, indicating that the combination of two potentially conformant implementations is still compliant.

An interoperability issue could arise when a bit-stream from FL compiler #1 implementation is decoded by compiler #2 implementation and there is a strong artefact observed at the UE #2, is it the issue with FL Encoder at compiler #1 or the issue with FL Decoder at compiler #2?

It should be noted that this scenario is not limited to FL compiler implementations but could also arise with fixed-point implementation, as such issue will happened with samples that are not part of the set of test vectors. In case of an artefact occurring, both implementations should be tested against the reference code (Fixed or Float) for the particular test sequence exhibiting the scenario. If the bit-stream from implementation #1 decoded using the 3GPP reference code yield no artefact, the problem is with the implementation #2. However if the decoded output of the 3GPP reference code has the same artefact, the problem is with implementation #1. Similarly if the decoded wave file obtained using 3GPP reference code and implementation #2 decoder yield artefact, the problem is with implementation #2.

The main difference from debugging perspective between floating-point and fixed-point implementations is that for the former bit-stream cannot be checked for bit exactness and thus require to assess the decoded output.

## 8.3 Conclusion

The assumption when performing this work was that if all the following conditions are met:

- passing conformance criteria for encoder only;

- passing conformance criteria for decoder only;

- passing conformance criteria for encoder-decoder pair;

there should not be interoperability issues.

No interoperability issue was found to contradict this approach, however if a counterexample was to be found, the gap would have to be addressed.

The way of debugging potential interoperability issue should be the same regardless if a floating-point or fixed-point implementation is used. If implementation #1 encoder with implementation #2 decoder yields artefact then each implementation should be checked against the reference code.

Adding the MOS-LQO verification to the conformance process allows to strengthen the interoperability testing as the implementation is evaluated not only with 3GPP floating-point reference code but also with 3GPP fixed-point reference code and the test coverage significantly increases using relevant material for voice communications.

# 9 Coverage Assessment

TBA

# 10 Conclusions

In Clause 5 various methods are presented for assessing floating-point implementations. For decoder verification objective waveform comparison methods (RMS error, SNR, SD) were studied (Clause 5.2). For encoder verification perceptually based objective methods (MOS-LQO, loudness metric) were studied (Clause 5.3).

In Clause 6 these methods have been tested with a total of more than 20 implementations using various compilers (gcc, clang, icc) and optimization levels, as well as code changes. In summary, the results show that these verification methods are able to discriminate between assumed correct implementations (no code changes, using only compiler optimization levels not impacting arithmetic behaviour) and assumed incorrect implementations (aggressive optimization levels, code changes).

Decoder code changes proposed that created audio artefacts (Clause 6.4.1 & 6.5.1) could be correctly flagged by the decoder test (Clause 6.4.4 & 6.5.3) and the MOS-LQO verification test (Clause 6.4.3 & 6.5.2). Another decoder code change that created a signal modification without audible distortion was correctly flagged by the decoder test (Clause 6.3.3).

There were no proposed encoder code changes to be verified.

Implementations using –fast optimization, which impact arithmetic behaviour, are flagged in the decoder test (6.3.2), MOS-LQO test (6.1.2 & 6.2.2) and encoder test (6.6).

Based on these results a possible conformance process was studied in Clause 7. It relies on passing the following three tests

- Decoder test based on signal metrics (RMS error, SNR, SD) described in Clause 5.2 to check closeness of the CUT decoder implementation with the TS 26.443 decoder.

- Encoder tests based on loudness metrics described in Clause 5.3.3 to check closeness of the CUT encoder implementation with the TS 26.443 encoder.

- MOS-LQO verification using POLQA described in Clause 5.3.2 to check closeness of the CUT implementation with the TS 26.442 implementation. This test also checks for potential interoperability issues as highlighted in Clause 8.

It is recommended to conduct normative work toward specifying a conformance process (including methods, thresholds and test vectors) in TS 26.444 based on the results presented in this technical report. Before a conformance process can be agreed, the following points will have to be finalized:

- Refinement of the conformance criteria in Clauses 5.2.6, 5.3.2.4 & 5.3.3.2 based on the latest reference code. The conformance process and criteria should be tight enough to avoid interoperability issues

- Tools and conformance test vector availability to perform conformance tests

- Further verification of the loudness tool

- Additional testing including validation that more general code changes are properly detected

Annex A:  
POLQA Considerations for Conformance Test

# A.1 Specific Recommendation from ITU-T P.863.1 [9]

A) **Section 8.9 "To use a reference speech sample longer than the recommended maximum 6 seconds of active speech, it is recommended that the signal is split into multiple 3 to 6 second active speech sections, and a 10 Rec. ITU-T P.863.1 (09/2014) separate score be computed for each section. Average the scores to determine a single score for the complete reference signal."**

a) This section explains how to use POLQA tool on a sample longer than the recommended maximum 6 seconds of active speech. The longer sample need to be split into multiple 6 seconds sections and a separate score to be computed for each section. The individual scores for each segment can be averaged later. However, such averaging can lead to the issues described in section 3 below.

B) **Section 8.1 "The list below describes a common set of required characteristics for both super-wideband and narrowband reference test signals: • at least three seconds of active speech; • at least 500 ms of silence between active speech periods; • no more than six seconds of active speech; • total length of test sample, including silence, should be no more than 12 seconds; • active speech level of -26 dBov; • 16-bit linear pulse code modulation (PCM) encoded; • noise floor < -75 dBov (A). Additional characteristics for [ITU-T P.863] super-wideband reference test signals are: • 48 k sample rate; • filtered 50 Hz to 14 kHz."**

b) This section explains that the reference signal should be a clean speech signal with a noise floor less than - 75 dBov. Hence, it is not appropriate to use POLQA with noisy speech vectors for conformance testing.

C) Section 14.2 "[ITU-T P.863] has not been validated against the variables given in Table 9 • Music as input to a codec"

c) This section mentions Music as input to a codec is not validated for POLQA. A relevant example will be discussed in Section 4 below.

D) **Section 8.4 "A reference signal should be filtered before presenting it to the [ITU-T P.863] model. A different filter is required for the super-wideband and narrowband modes. The filter definitions are provided in Tables 2 and 3."**

d) Table 2 referenced here mentions that a super wideband reference signal should be filtered to 50 Hz to 14 kHz. Any content above 14 kHz will be ignored by POLQA and hence any artifacts that can happen above 14 kHz during the conformance testing will remain undetectable.

# A.2 Averaging of MOS-LQO Scores Over Long Inputs

POLQA evaluation for long inputs (e.g., greater than 12 sec or more than 6 sec of active speech) should be done according the ITU P.863.1 specification text highlighted in Section 2.A. i.e. the signal is split into multiple 3 to 6 second active speech sections, and a separate MOS-LQO score be computed for each section, and averaged over all such segments. If an artifact occurs in one short speech segment (3-6 seconds as mentioned in Section 2.A above) out of a longer sample (e.g. 2 minutes), MOS-LQO score for this one small segment will show a higher deviation beyond the expected reference value (large segmental Delta-MOS-LQO value). However, after averaging over the long sample, overall Delta-MOS-LQO value will become insignificant and it will be impossible to detect the serious issue within one segment by only looking at the average Delta-MOS-LQO value and testing it against a threshold. This issue will persist even if the threshold applied to the average MOS-LQO score is very tight.

# A.3 Concerns over the Suitability of Perceptually Based Methods

The POLQA algorithm, which is standardized as Recommendation ITU-T P.863 [12], has been developed as an objective method to predict the scores of subjective ACR MOS tests for speech signals. The listening quality scores produced by POLQA, and similar objective algorithms, are denoted as MOS-LQO whereas those derived from human subjective assessment are denoted MOS-LQS.

ACR MOS-LQS scores are not precise single values but have an associated variance determined from the spread of individual votes cast by the subjects taking part in the ACR MOS test. The POLQA algorithm has been trained on many of the mean subjective test scores in order to derive its MOS-LQO scores but these are assumed to be point-values. From [12] it is claimed and can be seen that the MOS-LQO scores from Recommendation ITU-T P.863 provide good correlation but not perfect prediction of MOS-LQS scores from real tests involving clean speech signals.

Appendix I of [12] provides information about the prediction accuracy of POLQA for NB, WB and SWB when compared to actual scores from ACR tests after appropriate mapping of the results. Examining figures I.2, I.4 & I.5, it can be seen that, for a given MOS-LQO score from the POLQA algorithm after appropriate mapping and averaged over all appropriate samples in the test, the actual MOS-LQS score range in the very best cases would be in the region of between 0.5 MOS-LQS to 0.8 MOS-LQS. Examining figures 1.3, 1.5 & 1.7 from Appendix I of [12], it can be seen that in the worst case these errors increase to 1.6 MOS-LQS.

The points highlighted above raise concerns whether POLQA or delta-POLQA can be relied upon solely to detect different signal qualities and therefore non-conformant implementations of EVS.

Annex B:  
Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2018-06 | SA#80 | SP-180284 |  |  |  | Presented to TSG SA (for information) | 1.0.0 |
| 2018-09 | SA#81 | SP-180935 |  |  |  | Presented to TSG SA (for approval) | 2.0.0 |
| 2018-09 | SA#81 |  |  |  |  | Approved at TSG SA#81 | 16.0.0 |