Lab 1: Modeling Packet Losses

Kyeong Soo (Joseph) Kim
Department of Communications and Networking
School of Advanced Technology
Xi'an Jiaotong-Liverpool University
14 March 2023

I. Introduction

You are to carry out the following tasks in this Lab:

- Use the Python scripts provided in the Appendix (also downloadable from the Learning Mall) to generate sample binary sequences, analyze their run length statistics, and plot histograms for run length distributions. *Note that this is for your practice, and you don't have to submit anything.*
- Model packet losses using a simple Gilbert model (SGM) and a Gilbert model (GM) based on the sample sequences of packet losses available on the ICE and carry out a simple statistical analysis of the constructed models. Details are given in Sec. III.

You need to submit the Lab report and program source code through the Learning Mall by the end of Sunday, 9 April 2023.

II. Gilbert-Elliott Model

Here we introduce the Gilbert-Elliott model (GEM) [1], i.e., a generalised version of the SGM that we studied during the lectures. The GEM is a two-state Markov chain with state-dependent loss probabilities—i.e., $(1-k)$ at **GOOD** and $(1-h)$ at **BAD** state—as shown in Fig. 1. The GEM is quite straightforward

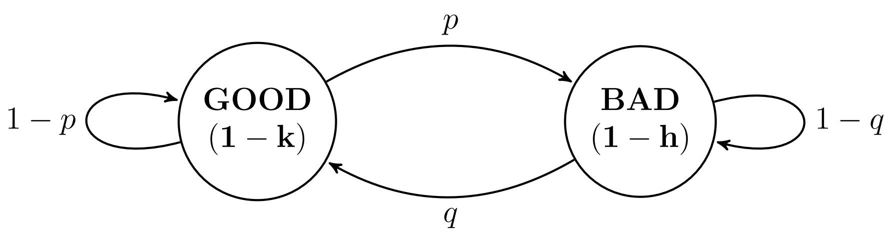

Fig. 1: Gilbert-Elliott model (GEM).

to understand and still useful in modeling burst packet loss with correlation. Transitions between the states are per-packet basis.

The transition matrix is given by

$$\boldsymbol{P} = \begin{bmatrix} (1-p) & q \\ p & (1-q) \end{bmatrix} \tag{1}$$

where p and q are transition probabilities from **GOOD** to **BAD** and from **BAD** to **GOOD** state, respectively. The steady state probability vector $\boldsymbol{\pi}$ satisfies the following conditions:

$$\boldsymbol{\pi} = \boldsymbol{P}\boldsymbol{\pi}, \ \ \mathbf{1}^t\boldsymbol{\pi} = 1, \tag{2}$$

where

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_G \\ \pi_B \end{bmatrix}. \tag{3}$$

The steady state probabilities exist for $0<p, q<1$ and are given by

$$\pi_G = \frac{q}{p+q}, \ \pi_B = \frac{p}{p+q}. \tag{4}$$

From (4), we obtain the packet loss probability p_L as follows:

$$p_L = (1-k)\pi_G + (1-h)\pi_B. \tag{5}$$

The special cases of $k=1$ and $k=1, h=0$ are called a Gilbert Model (GM) and a simple Gilbert Model (SGM), respectively.

In case of the SGM, the probability distribution of loss run length *conditioned on the observation of a transition from 0 to 1* (see Fig. 2) has a geometric distribution: For $k=1, 2, \ldots, \infty$,

$$\begin{aligned} p_k &\triangleq Prob\,\{\text{Loss run length} = k | \text{Transition from 0 to 1 observed}\} \\ &= (1-q)^{k-1}\, q. \end{aligned} \tag{6}$$

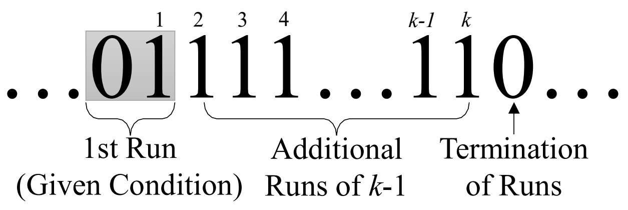

Fig. 2: Illustration of packet loss run length.

III. Task: Packet Loss Modeling

To model packet losses with an SGM, we need to decide its transition probabilities p and q. There have been proposed many techniques for the SGM parameter estimation based on the measured loss trace, e.g., see [1] and [2]. Here we use a simple method proposed by Yajnik et al. [3], where p and q are estimated as follows:

$$p = \frac{n_{01}}{n_0}, \ q = \frac{n_{10}}{n_1} \tag{7}$$

where n_{01} is the number of times in the observed time series that 1 follows 0 and n_{10} is the number of times 0 follows 1. n_0 is the number of 0s and n_1 is the number of 1s in the trace.

In case of GM-based packet loss modeling, we need to decide the values of three parameters—i.e., p, q, and h. Gilbert suggested to estimate those model parameters from another set of three parameters that can be estimated from the measured loss trace [4]:

$$a = P(1), \ b = P(1|1), \ c = \frac{P(111)}{P(101) + P(111)}, \tag{8}$$

where $P(\cdot)$ is the probability of a given loss pattern and $P(1|1)$ is a conditional probability.[1] From the values of a, b, and c, we can obtain the three model parameters as follows:

$$1-q = \frac{ac - b^2}{2ac - b(a+c)}, \ h = 1 - \frac{b}{1-q}, \ p = \frac{aq}{1-h-a}. \tag{9}$$

For this task, you need to submit a Lab report and program source code summarizing the following activities:

[1]Refer to the pages 1260–1261 of [4] for more details on this.

#1 Create a Python script to build an SGM for a given trace based on the method described above and submit the source code with detailed comments.

Solution

Sample scripts:

- `binary_runlengths.py`
- `sgm_generate.py`
- `sgm_modeling.py`
- `model_analysis.py`

The scripts need to be on the same directory. Then, you can run it by typing (e.g., "`python sgm_modeling.py`") on the command line.

- Run the script over the two sample binary sequences available on the Learning Mall (i.e., "dataset-A-*-*-*-*.bitmap") and provide the following for *each of them*:
  - [**10 points**] Estimated model parameters.

    Solution

    `dataset-A-adsl5-cbr1.0-20091011-035000.bitmap` (**short trace**):

    * p: 5.0713E-02
    * q: 9.0000E-01

    `dataset-A-adsl1-cbr6.0-20090628-223500.bitmap` (**long trace**):

    * p: 2.3886E-02
    * q: 4.8417E-01
  - [**10 points**] Histograms of run lengths for zero and one for both the sample binary sequence and the sequence generated by the constructed model.

    Solution

    See Figs. 3 and 4.
  - [**10 points**] Power spectral densities (PSDs) of sample binary sequence and the sequence generated by the constructed model.

    Solution

    See Fig. 5 and 6.
  - [**10 points**] Comparison (i.e., discussion) of the two sequences based on their histograms and PSDs.

    Solution

    For both sequences, Figs. 3 and 4 demonstrate that the synthetic traces generated by the SGM show *one runlength* distributions similar to those of the real traces, while the *zero runlength* distributions of the synthetic and real traces show significant differences; the synthetic traces cannot generate well the extremely long zero runlengths found in the real traces.

    The differences in the zero runlength distributions of the real and the synthetic traces could be a reason for the differences in their PSDs shown in Figs. 5 and 6, again for

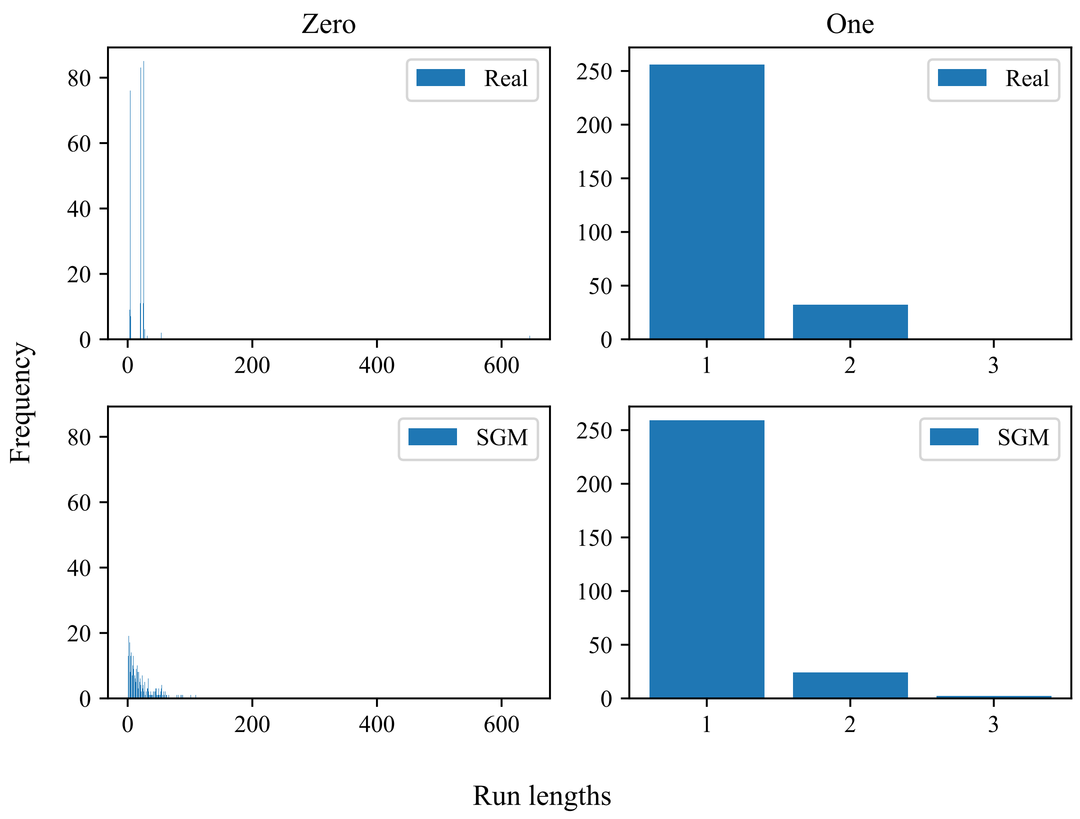

Fig. 3: Runlength histograms for the real (i.e., short trace) and the synthetic (i.e., SGM) loss traces.

both sequences. In addition, the PSD for the short trace shown in Fig. 5 reveals *periodic components*, which, however, are not observed in the PSD for the synthetic trace by the SGM.

In summary, the comparative analyses based on histograms and PSDs show the weakness of the SGM in generating extremely long runlengths and periodic behaviours.

- [**10 points**] Discuss the impact of the lengths of the sample binary sequences on loss modeling.

Solution

There are no major issues related with the lengths of sample binary sequences in loss modeling based on the SGM:

Though there is a huge difference between the two binary sequences (i.e., 5,999 vs 5,999,999), there are no problems in estimating the SGM parameters for both sequences. Also, the comparison based on runlength distributions and PSDs do not show any visible impact of sequence lengths.

#2 Repeat the steps 1) and 2) for a GM this time.

Solution

Sample scripts:

- `binary_runlengths.py`
- `gm_generate.py`
- `gm_modeling.py`

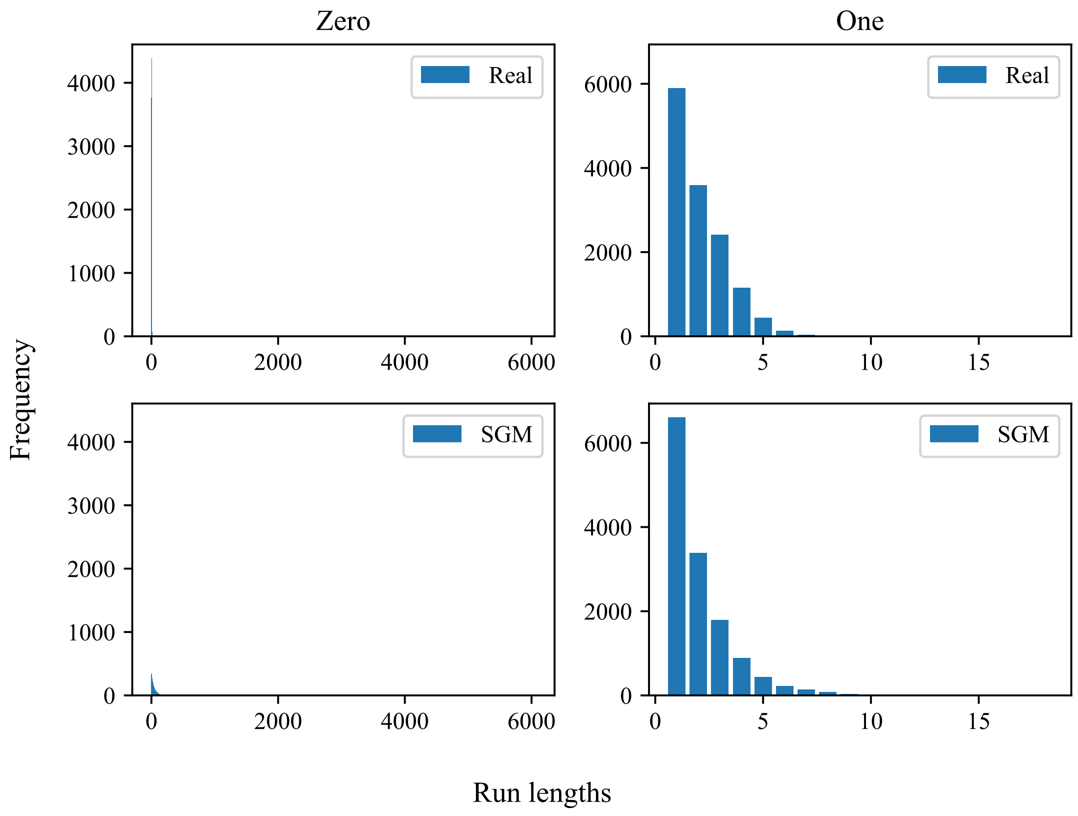

Fig. 4: Runlength histograms for the real (i.e., long trace) and the synthetic (i.e., SGM) loss traces.

- `model_analysis.py`

  The scripts need to be on the same directory. Then, you can run it by typing (e.g., "`python gm_modeling.py`") on the command line.

- Run the script over the two sample binary sequences available on the Learning Mall (i.e., "dataset-A-*-*-*-*.bitmap") and provide the following for *each of them*:
  - [**10 points**] Estimated model parameters.

    Solution

    `dataset-A-adsl5-cbr1.0-20091011-035000.bitmap` (**short trace**)[2]:

    ∗ p: 9.5540E-02

    ∗ q: 8.0000E-01

    ∗ h: 5.0000E-01

    `dataset-A-adsl1-cbr6.0-20090628-223500.bitmap` (**long trace**):

    ∗ p: 1.5841E-02

    ∗ q: 2.0194E-01

    ∗ h: 3.5365E-01

[2]These results are based on the alternative estimation method with h=0.5 as suggested in [4].

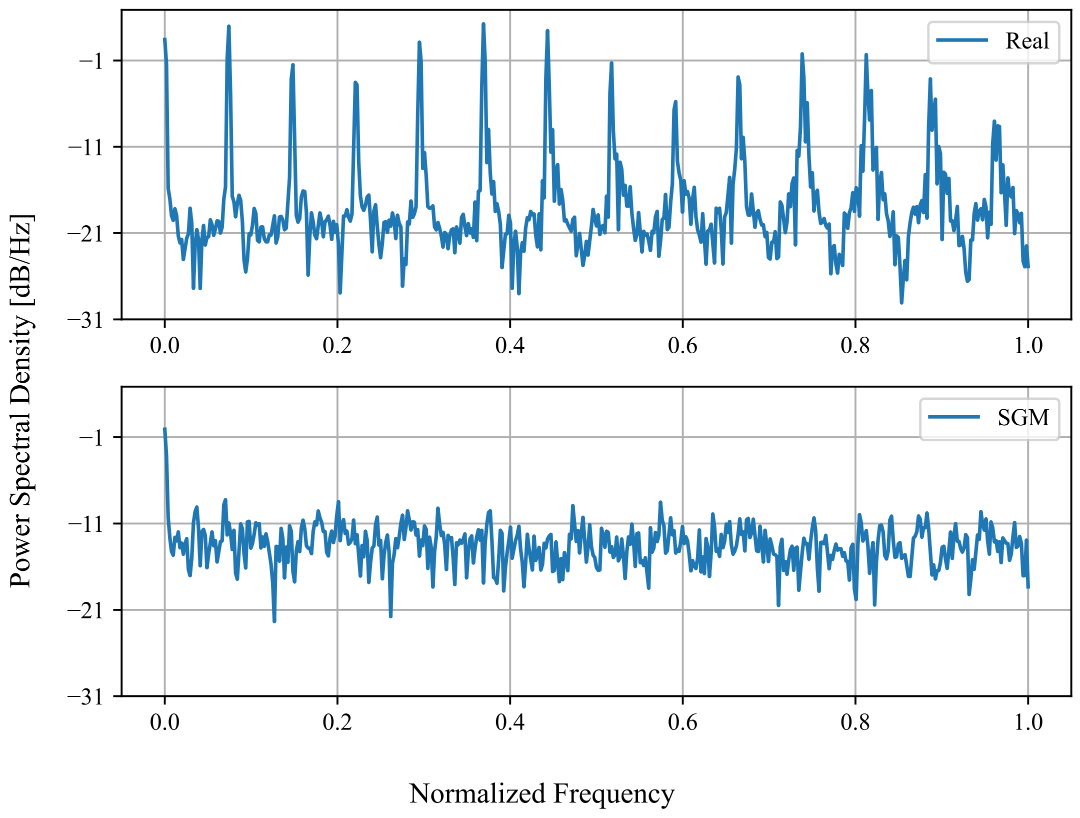

Fig. 5: PSDs for the real (i.e., short trace) and the synthetic (i.e., SGM) loss traces.

– [**10 points**] Histograms of run lengths for zero and one for both the sample binary sequence and the sequence generated by the constructed model.

Solution

See Figs. 7 and 8.

– [**10 points**] Power spectral densities (PSDs) of sample binary sequence and the sequence generated by the constructed model.

Solution

See Fig. 9 and 10.

– [**10 points**] Comparison (i.e., discussion) of the two sequences based on their histograms and PSDs.

Solution

Nearly identical observations can be made for the GM, too:

For both sequences, Figs. 7 and 8 demonstrate that the synthetic traces generated by the GM show *one runlength* distributions similar to those of the real traces, while the *zero runlength* distributions of the synthetic and real traces show significant differences; the synthetic traces cannot generate well the extremely long zero runlengths found in the real traces.

The differences in the zero runlength distributions of the real and the synthetic traces could be a reason for the differences in their PSDs shown in Figs. 9 and 10, again for

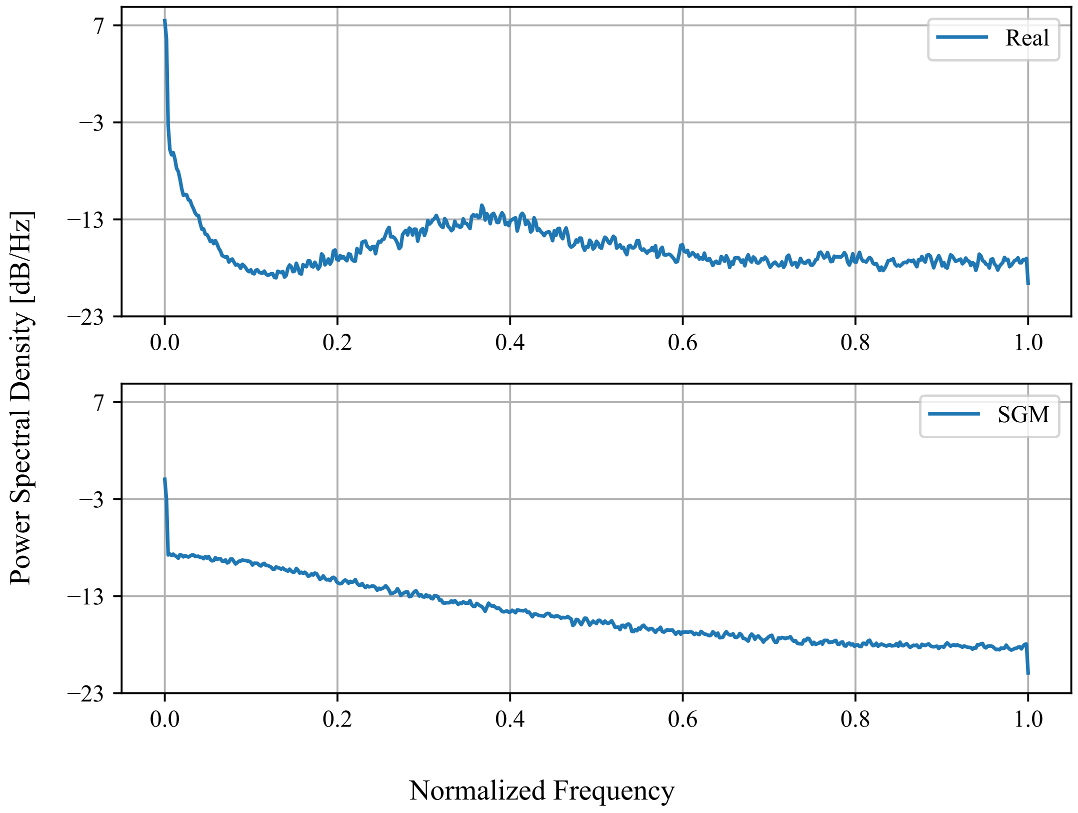

Fig. 6: PSDs for the real (i.e., long trace) and the synthetic (i.e., SGM) loss traces.

both sequences. In addition, the PSD for the short trace shown in Fig. 9 reveals *periodic components*, which, however, are not observed in the PSD for the synthetic trace by the SGM.

In summary, the comparative analyses based on histograms and PSDs show the weakness of the GM in generating extremely long runlengths and periodic behaviours.

- [**10 points**] Discuss the impact of the lengths of the sample binary sequences on loss modeling.

Solution

Unlike the SGM, there is a major issue in loss modeling based on the GM: In case of the short trace with only 5,999 sample, the estimation of the parameter c is not possible because the denominator of $\frac{P(111)}{P(101)+P(111)}$ is zero; this is not the case for the long trace, which has 5,999,999 samples.

Other than the parameter estimation, the comparison based on runlength distributions and PSDs do not show any visible impact of sequence lengths.

Appendix
Generate a Loss Pattern based on SGM

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
##
# @file     sgm_generate.py
# @author   Kyeong Soo (Joseph) Kim <kyeongsoo.kim@gmail.com>
```

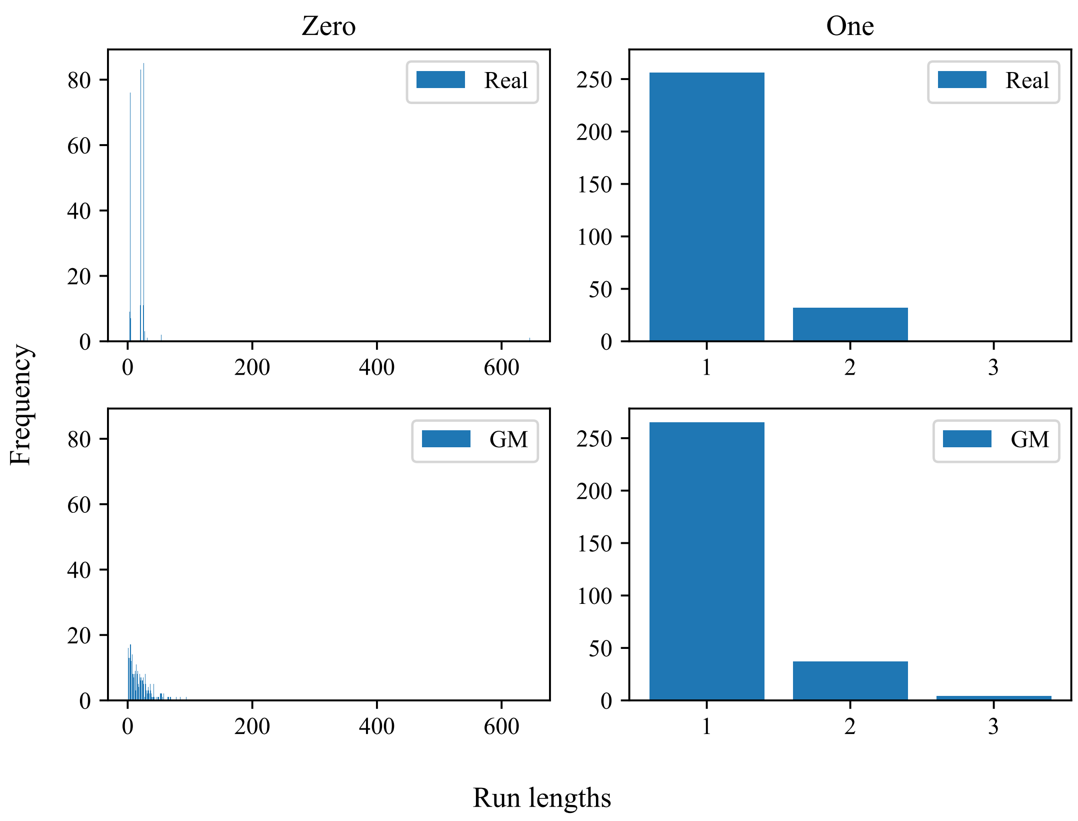

Fig. 7: Runlength histograms for the real (i.e., short trace) and the synthetic (i.e., GM) loss traces.

```python
# @date     2020-03-25
#           2023-04-10
#
# @brief    A function for generating loss pattern based on the simple
#           Guilbert model (SGM).
#

import numpy as np
import sys


def sgm_generate(len, p, q):
    """
    Generate a binary sequence of 0 (GOOD) and 1 (BAD) of length len
    from the SGM specified by transition probabilites 'p' (GOOD->BAD)
    and 'q' (BAD->GOOD).

    This function assumes that the SGM starts in GOOD (0) state.

    Examples:

    seq = sgm_generate(100, 0.95, 0.9)
    """

    seq = np.zeros(len)

    # check transition probabilites
```

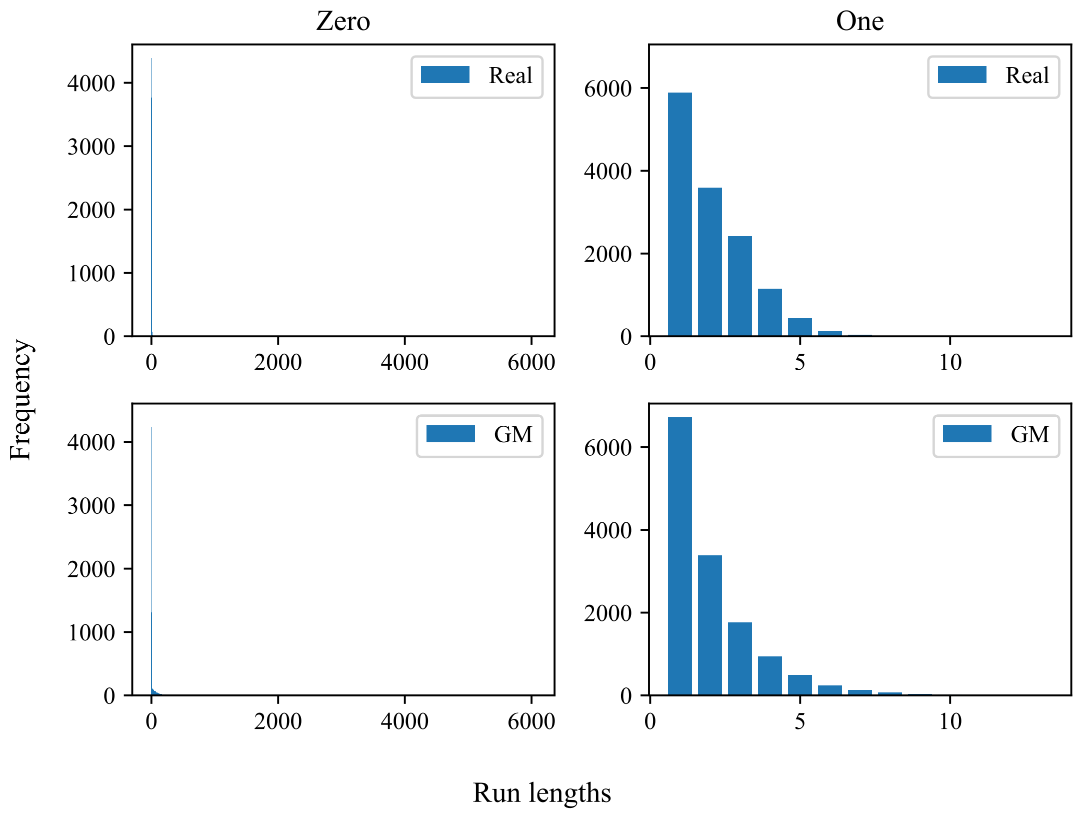

Fig. 8: Runlength histograms for the real (i.e., short trace) and the synthetic (i.e., GM) loss traces.

```
    if p < 0 or p > 1:
        sys.exit("The value of the transition probability p is not valid.")
    elif q < 0 or q > 1:
        sys.exit("The value of the transition probability q is not valid.")
    else:
        tr = [p, q]

    # create a random sequence for state changes
    statechange = np.random.rand(len)

    # Assume that we start in GOOD state (0).
    state = 0

    # main loop
    for i in range(len):
        if statechange[i] <= tr[state]:
            # transition into the other state
            state ^= 1
        # add a binary value to output
        seq[i] = state

    return seq


if __name__ == "__main__":
    import argparse

```

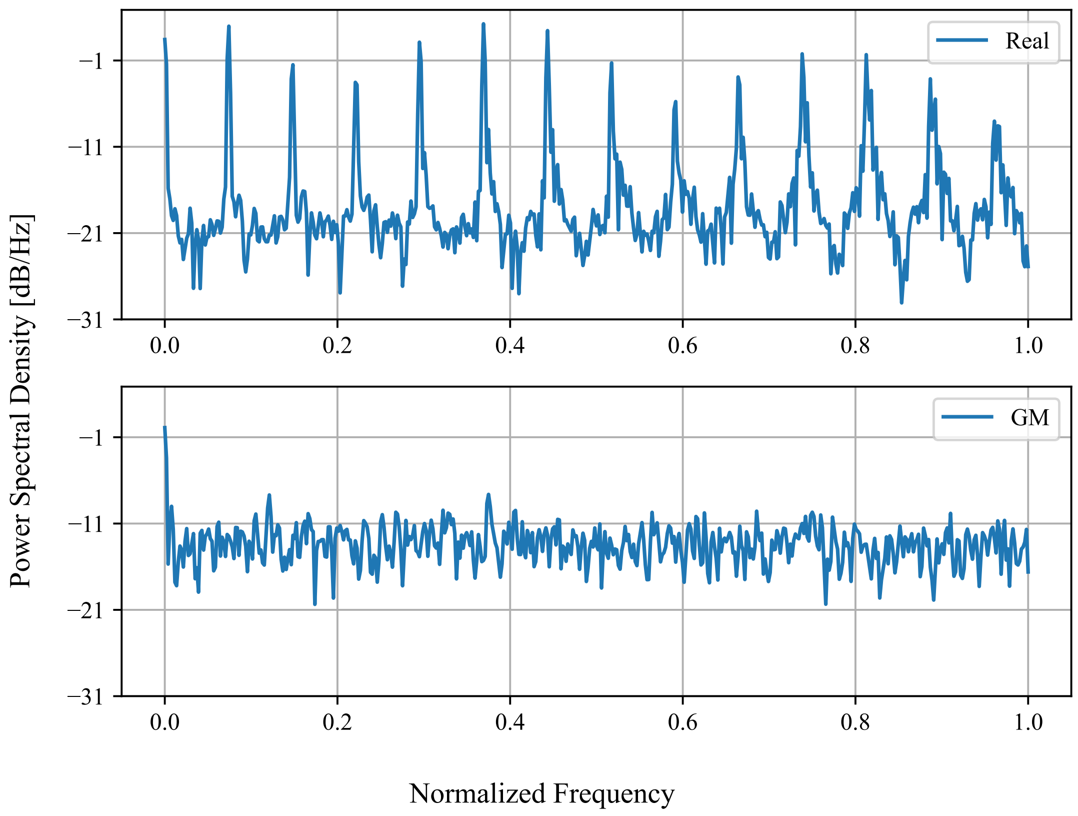

Fig. 9: PSDs for the real (i.e., long trace) and the synthetic (i.e., GM) loss traces.

```
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-l",
        "--length",
        help="the length of the loss pattern to be generated; default is 10",
        default=10,
        type=int)
    parser.add_argument(
        "-p",
        help="GOOD to BAD transition probability; default is 0.95",
        default=0.95,
        type=float)
    parser.add_argument(
        "-q",
        help="BAD to GOOD transition probability; default is 0.9",
        default=0.9,
        type=float)
    args = parser.parse_args()
    print(sgm_generate(args.length, args.p, args.q))
```

References

[1] G. Haßlinger and O. Hohlfeld, “The Gilbert-Elliott model for packet loss in real time services on the Internet,” in Proc. 2008 MMB, Mar. 2008, pp. 1–15.

[2] M. Ellis, D. P. Pezaros, T. Kypraios, and C. Perkins, “A two-level Markov model for packet loss in UDP/IP-based real-time video applications targeting residential users,” Computer Networks, vol. 70, pp. 384–399, Sep. 2014.

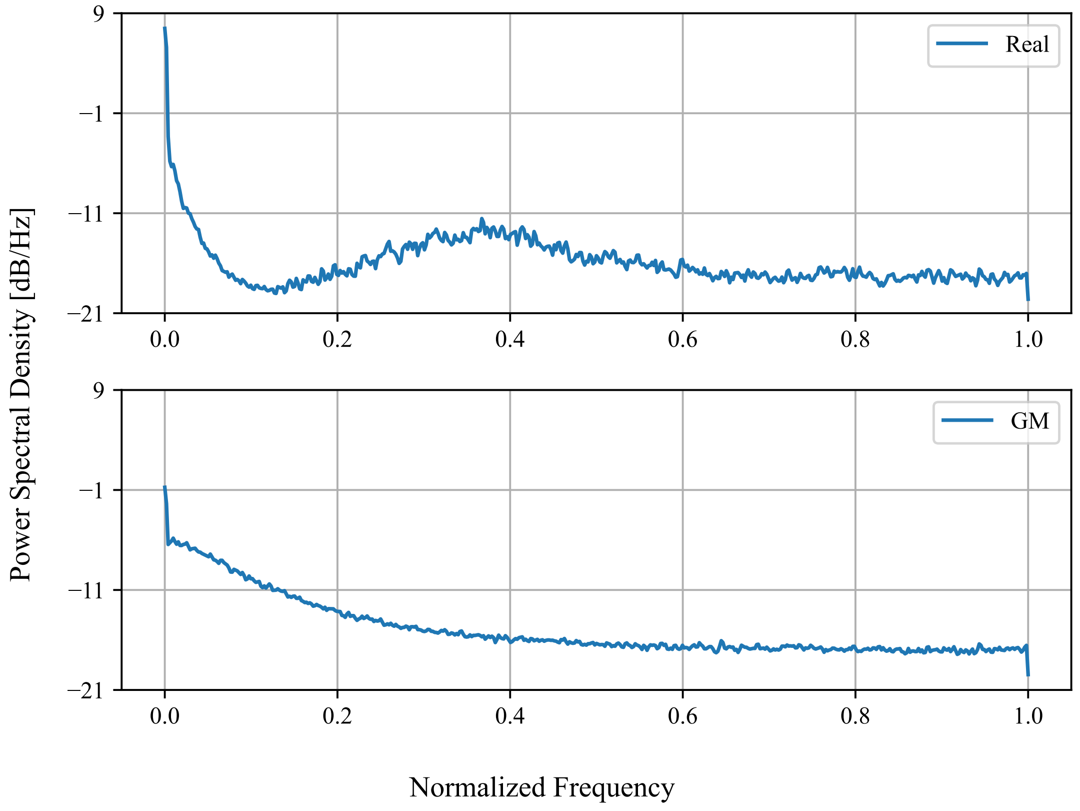

Fig. 10: PSDs for the real (i.e., long trace) and the synthetic (GM) loss traces.

[3] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in Proc. 1999 IEEE INFOCOM, vol. 1, Mar. 1999, pp. 345–352.

[4] E. N. Gilbert, "Capacity of a burst-noise channel," Bell System Technical Journal, vol. 39, no. 5, pp. 1253–1265, Sep. 1960.