TED Documentation

Release 0.7.0

Lev Givon, Yiyin Zhou

October 24, 2016

Part I

Introduction

The Time Encoding and Decoding Toolkit (TED) is a library of MATLAB functions and demos that implement time encoding and decoding algorithms. A Python/PyCUDA implementation is also available from the Bionet Group's code repository page.

To get started using the toolkit, go to the installation page and try running the demos. Check out the machines page for a high-level description of the time encoding and decoding routines included in the toolbox.

Part II

Contents

CHAPTER 1

Installation Instructions

1.1 Prerequisites

The Time Encoding and Decoding Toolkit requires a relatively recent version of MATLAB (e.g., R14 or later) to run.

To rebuild the documentation, the following Python packages are also required:

- Docutils 0.5 or later.
- Jinja2 2.2 or later
- Pygments 0.8 or later
- Sphinx 1.0.1 or later.
- Sphinx ReadTheDocs Theme 0.1.6 or later.

1.2 Building and Installation

Unpack the archive (or check out the project GitHub repository) into an installation directory of your choice. Assuming that Matlab is installed in the directory whose name is stored in the environmental variable `$MATLAB`, one may wish to install the software in `$MATLAB/toolbox/ted` if one has administrative access to the machine.

Add the subdirectories of the installation directory to Matlab's search path so that the functions comprised by the toolbox may be easily accessed. For example, if the software is installed in `$MATLAB/toolbox`, one may add the following lines to the list of paths in `$MATLAB/toolbox/local/pathdef.m` and restart Matlab to make the toolbox contents available:

```
matlabroot,'/toolbox/ted/ted:', ...
matlabroot,'/toolbox/ted/teddemos:', ...
matlabroot,'/toolbox/utils/utils:', ...
matlabroot,'/toolbox/utilsdemos:', ...
```

Alternatively, one may also add the path to the toolbox for the duration of one session by running the commands

```
addpath(strcat(getenv('MATLAB'),'/toolbox/ted/ted'));
addpath(strcat(getenv('MATLAB'),'/toolbox/ted/teddemos'));
addpath(strcat(getenv('MATLAB'),'/toolbox/utils/utils'));
addpath(strcat(getenv('MATLAB'),'/toolbox/utils/utilsdemos'));
```

To rebuild the documentation, run:

```
make
```

from within the `docs/` subdirectory and follow the directions.

CHAPTER 2

Time Encoding and Decoding Machines

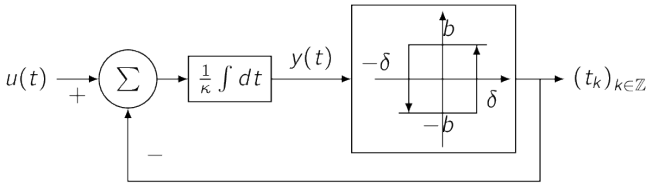
The TED Toolkit provides implementations of a range of time encoding and decoding machines. Brief descriptions of these machines are provided in the following pages, along with links to the relevant articles that describe them in more detail and documentation for the Python functions and classes that implement them.

2.1 Asynchronous Sigma-Delta Modulators

2.1.1 Encoding with Asynchronous Sigma-Delta Modulator (ASDM)

- **ASDM Encoding** (`asdm_encode.m`) [2]:

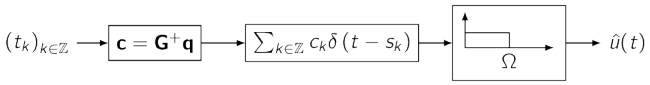
  Encodes a bandlimited signal using an ASDM encoder. A signal can be encoded by a single ASDM encoder (Single-Input Single-Output Encoding), as shown below, or it can be encoded by multiple of such ASDM encoders (Single-Input Multiple-Output Encoding).



2.1.2 Decoding for Signal Encoded with Single-Input Single-Output ASDM Encoder

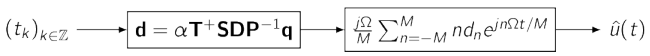
- **Decoding** (`asdm_decode.m`) [2]:

  Reconstructs a bandlimited signal encoded by an Single-Input Single-Output ASDM encoder using sinc kernels.



- **Decoding using Fast Approximation Method** (`asdm_decode_fast.m`) [4]:

  Reconstructs a bandlimited signal encoded by a Single-Input Single-Output ASDM encoder using a fast approximation method.



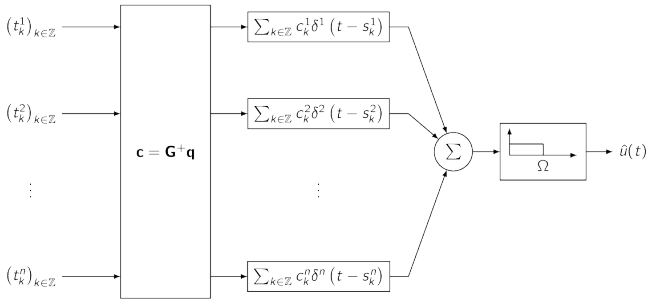
- **Decoding using Threshold-Insensitive Method** (`asdm_decode_ins.m`) [2]:

  Reconstructs a bandlimited signal encoded by a Single-Input Single-Output ASDM encoder. This reconstruction method does not require the specification of an integrator threshold.

2.1.3 Decoding for Signal Encoded with Single-Input Multiple-Output ASDM Encoder

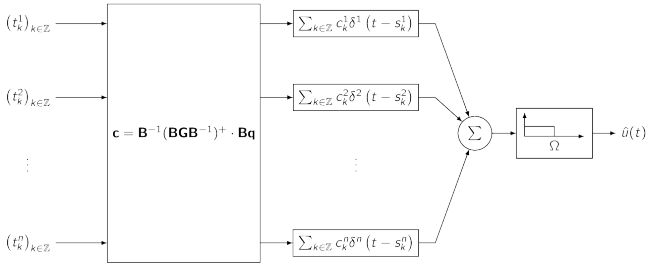
- **Population Decoding** (`asdm_decode_pop.m`) [7]:

  Reconstructs a bandlimited signal encoded by multiple ASDM encoders using sinc kernels.



- **Population Decoding using Threshold-Insensitive Method** (`asdm_decode_pop_ins.m`) [2] [7]:

  Reconstructs a bandlimited signal encoded by multiple Asynchronous Sigma-Delta Modulators using sinc kernels. This reconstruction method does not require the specification of an integrator thresholds.

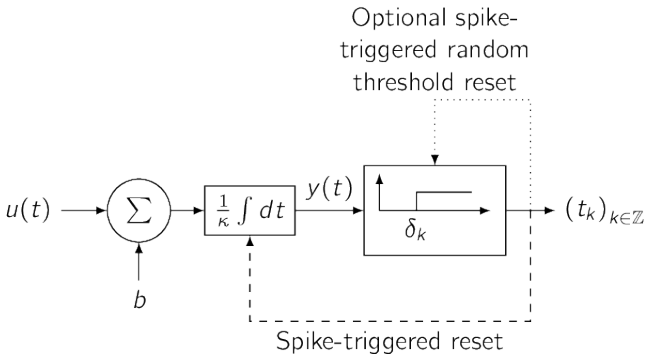


2.2 Integrate-and-Fire Neurons

2.2.1 Encoding with Integrate-and-Fire (IAF) Neurons

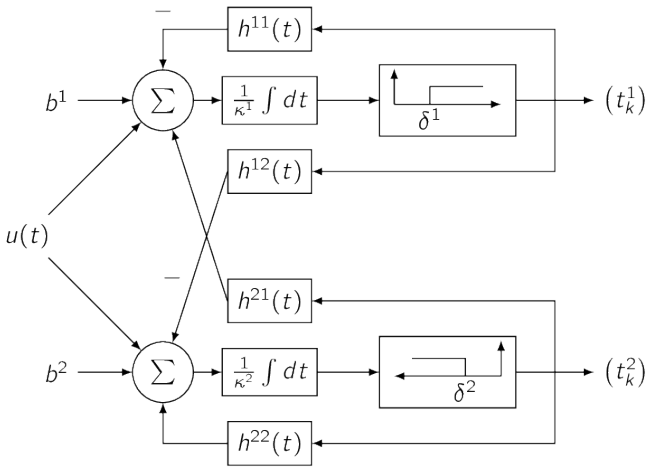
- **IAF Encoding** (`iaf_encode.m`) [2] [11]:

  Encodes a time-varying signal using an IAF neuron. Leaky and ideal neuron models are supported. In addition, IAF neuron with random (Gaussian) thresholds is also supported. A signal can be encoded by a single IAF encoder (Single-Input Single-Output Encoding), as shown in the figure below, or it can be encoded by a population of IAF neurons (Single-Input Multi-Output Encoding).



- **ON-OFF IAF Encoding** (`iaf_encode_ideal_on_off.m`) [12]:

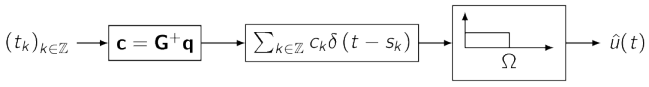
  Encodes a time-varying signal using an ON-OFF IAF neuron pair. Only ideal IAF neuron models are supported.



2.2.2 Decoding for Signal Encoded with Single-Input Single-Output IAF neuron

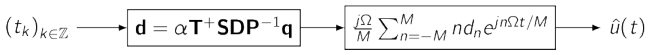
- **Decoding** (`iaf_decode.m`) [1]:

  Reconstructs a bandlimited signal encoded by an IAF neuron using sinc kernels.



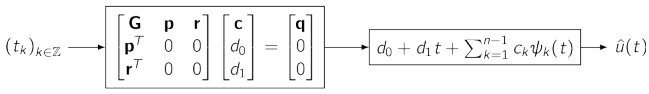
- **Decoding using Fast Approximation Method** (`iaf_decode_fast.m`) [4]:

  Reconstructs a bandlimited signal encoded by an IAF neuron using a fast approximation method.



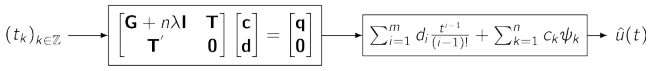
- **Decoding using Spline Interpolation** (`consistent_decoding_LIF.m`) [12]:

  Reconstructs a finite-energy signal encoded by an Leaky-Integrate-and-Fire (LIF) neuron. It uses spline interpolation algorithm.



- **Decoding using Smoothing Spline** (`LIF_decode_S1.m`, `LIF_decode_S2.m`) [11]:

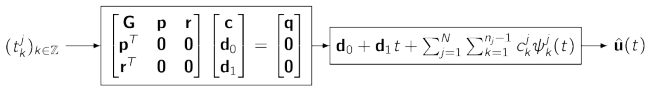
  Reconstructs a signal in Sobolev space S_1 or S_2 encoded by a LIF neuron using smoothing splines. Signals encoded by a LIF with random threshold should be decoded using this function.



2.2.3 Decoding for Signal Encoded with Single-Input Multiple-Output IAF neurons

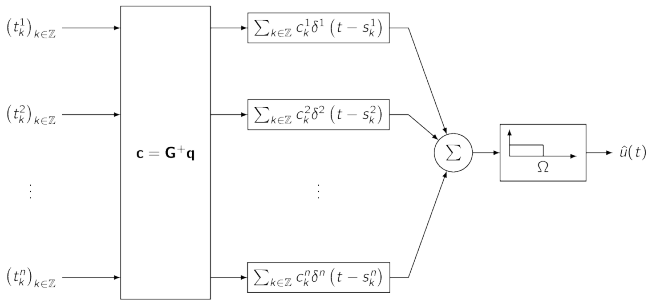
- **Decoding for ON-OFF IAF** (`consistent_decoding_IF_ONOFF.m`) [12]:

  Reconstructs a finite energy signal encoded by ON-OFF IAF neuron pair. The reconstruction is performed using spline interpolation method.



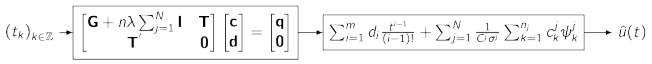
- **Population Decoding** (`iaf_decode_pop.m`) [7]:

  Reconstructs a bandlimited signal encoded by an ensemble of IAF neurons using sinc kernels.



- **Population Decoding using Smoothing Splines** (`LIF_pop_decode_S1.m`, `LIF_pop_decode_S2.m`) [11]:

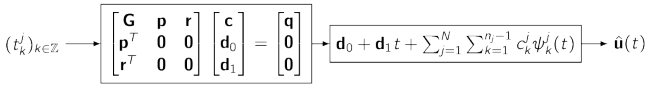
  Reconstructs a signal in Sobolev Space S_1 or S_2 encoded by a population of LIF neurons. The reconstruction uses smoothing spline method in the RKHS. Signals encoded by population of LIF with random threshold should be decoded using this function.



2.2.4 Decoding for Signals Encoded with Multiple-Input Multiple-Output IAF neurons

- **Population Decoding using Spline Interpolation** (`consistent_decoding_IF_MIMO.m`) [12]:

  Reconstructs multiple finite energy signals encoded by a population of ideal IAF neurons in Multiple-Input Multiple Output setting. The reconstruction uses spline interpolation method.



CHAPTER 3

Publications

The Time Encoding and Decoding Toolkit implements algorithms from the following publications:

[1] A.A. Lazar, **Time Encoding with an Integrate-and-Fire Neuron with a Refractory Period**, *Neurocomputing*, Vol. 58-60, pp. 53-58, Jun. 2004

[2] A.A. Lazar and L.T. Toth, **Perfect Recovery and Sensitivity Analysis of Time Encoded Bandlimited Signals**, *IEEE Transactions on Circuits and Systems-I: Regular Papers*, Vol. 51, No. 10, pp. 2060-2073, Oct. 2004

[3] A.A. Lazar, **Multichannel Time Encoding with Integrate-and-Fire Neurons**, *Neurocomputing*, Vol. 65-66, pp. 401-407, Jun. 2005

[4] A.A. Lazar, E.K. Simonyi, and L.T. Toth, **Fast Recovery Algorithms of Time Encoded Bandlimited Signals**, *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 4, pp. 237-240, Philadelphia, PA, Mar. 19-23, 2005

[7] A.A. Lazar, **Information Representation with an Ensemble of Hodgkin-Huxley Neurons**, *Neurocomputing*, Vol. 70, pp. 1764-1771, Jun. 2007

[11] A.A. Lazar and E.A. Pnevmatikakis, **Reconstruction of Sensory Stimuli Encoded with Integrate-and-Fire Neurons with Random Thresholds**, *EURASIP Journal on Advances in Signal Processing, Special Issue on Statistical Signal Processing in Neuroscience*, Vol. 2009, 2009

[12] A.A. Lazar and E.A. Pnevmatikakis, **Consistent Recovery of Sensory Stimuli Encoded with MIMO Neural Circuits**, *Computational Intelligence and Neuroscience, Special Issue on Signal Processing for Neural Spike Trains*, Feb. 2010

CHAPTER 4

Authors & Acknowledgments

The implementations in this toolbox were written by the following current and past members of the Bionet Group [1] at Columbia University's Department of Electrical Engineering under the supervision of Prof. Aurel A. Lazar [2]:

- Lev Givon
- Eftychios A. Pnevmatikakis
- Robert J. Turetsky

Some portions of the code are partially based upon older implementations written by

- Robert J. Turetsky
- Yevgeniy B. Slutskiy

Documentation for this toolbox was prepared by

- Lev Givon
- Yiyin Zhou

The software in this toolbox was integrated and packaged by Lev Givon.

[1] http://www.bionet.ee.columbia.edu/
[2] http://www.ee.columbia.edu/~aurel/

CHAPTER 5

License

CHAPTER 6

Change Log

6.1 Release 0.7 - (March 21, 2015)

- Change package name to Time Encoding and Decoding Toolkit.
- Change to PEP 440 version numbering.
- Use ReadTheDocs theme in HTML docs.

6.2 Release 0.061 - (August 6, 2012)

- Documentation build bugfix.

6.3 Release 0.06 - (August 6, 2012)

- Add documentation.
- Merge with utils toolbox.

6.4 Release 0.05 - (April 28, 2011)

- Improve performance of IAF encoder.
- Fix incorrect signs in delayed IAF algorithms.
- Add gammatone trigonometric polynomail demos.
- Update license.

6.5 Release 0.04 - (April 16, 2010)

- Add gammatone filter algorithms.
- Add delayed IAF algorithms.
- Add spline interpolation algorithms.

6.6 Release 0.03 - (January 07, 2010)

- Add IAF population decoding algorithms.
- Speed up decoding algorithms with faster sine integral approximation.

6.7 Release 0.02 - (October 26, 2009)

- Add ASDM population decoding algorithms.
- Make time array construction consistently use a fixed time step.

6.8 Release 0.011 - (May 08, 2009)

- Change package name to Time Encoding and Decoding Toolbox.

6.9 Release 0.01 - (May 08, 2009)

- First public release.

Part III

Index

- genindex
- search

A

C

I

L