Documentation for using the VAE-FIR

Author: louis tomczyk
Date: May 30, 2024
Version: v1.0.0

TELECOM Paris
IP PARIS
Institut Mines-Télécom

1 Context

This documentation aims to ease the handling of the code of VAE-inspired equalisers for optical telecommunications. This code is greatly inspired of the one [1] that was shared under the GNU General Public License v2.0. Its objective is to simulate an end-to-end optical telecommunication chain using a linear (optical) channel and to explore new equalisation algorithms.

2 Architecture of the project

The tree of the project is given below and explained in the following subsections.

VAE-FIR
- **python/**
  - main.py
  - processing.py
  - lib_*.py ..... custom libraries
  - **data-YY-MM-DD/** ..... permanent data folder
  - **data-YY-MM-DD-<x>/** ..... temporary data folder if no error in execution, $< x >\in \mathbb{N}$
    - *YYYY-MM-DD hh:mm:ss <param name 1> <param value 1> - *.xml*
    - *YY-MM-DD hh:mm:ss <param name 1> <param value 1> - *.csv*
    - *YY-MM-DD hh:mm:ss <param name 1> <param value 1> - *.mat*
    - *YY-MM-DD hh:mm:ss <param name 1> <param value 1> - *.<img ext>* ..... <img ext>$\in$ {png,svg,pdf,...}
    - *matlabYY-MM-DD hh:mm:ss <param name 1> <param value 1> - * - poincare*.png
    - *Err Theta-matlabYY-MM-DD hh:mm:ss <param name 1> <param value 1> - *.csv*
    - *<Err Theta>-matlabYY-MM-DD hh:mm:ss <param name 1> <param value 1> - *.csv*
- **matlab/**
  - main_0_python2matlab.py
  - lib
    - *.m ..... custom functions

2.1 Versioning and licenses

For this code version v.1.0.0, here the versions of the different files.

file name	version	date (YYY-MM-DD)
python files (*.py), GPL2.0		
main	1.3.0	2024-05-30
processing	1.2.2	2024-05-28
general	1.4.1	2024-05-24
kit	1.1.0	2024-05-22
maths	1.2.1	2024-05-27
matlab	1.1.0	2024-04-24
misc	1.2.1	2024-05-28
prop	1.0.3	2024-05-27
rxdsp	1.3.0	2024-05-27
rxhw	1.0.2	2024-04-01
txdsp	1.0.4	2024-05-27
txhw	1.2.0	2024-05-29
matlab files (*.m), CC-BY-NC-SA		
main_0_python2matlab	1.1.4	2024-04-19
FIR2Stockes	1.1.0	2024-03-08
get_value_from_filename	1.1.0	2024-03-05
Jones2Stockes	1.1.0	2024-03-07
moving_average	1.0.0	2024-05-23
moving_std	1.0.0	2024-03-14
sort_strings	?	?

We have chosen to license the Python and Matlab files separately. They work independently of each other. The Python functions are used in a project licensed under GPL2.0, so we have used the same licence to make it easier to license the Python project.

However, the Matlab functions are licensed under the Creative Commons :

can: modify - distribute

cannot: use for commercial purposes

must: share alike - include license

2.2 General description

2.2.1 Home folder

The home folder contains two sub-folders:

- matlab: permanent folder with the Matlab scripts and functions to process the data generated by the Python codes.
- python: permanent folder with the Python scripts and functions for the optical telecommunication simulations.

Once the Python scripts have been launched and data sorted in the appropriate folders within the Python's home, the data post-processing is performed with Matlab code. The post-processing data generated by Matlab are still stored in the sub-folders in the Python' home.

2.2.2 "Python" folder

Python home

This repository contains:

- `main.py`: the script to execute with the minimal parameters
- `processing.py`: the core function
- `lib_*.py`: the libraries, $* \in$ {`general, matlab, misc, prop, txdsp, txhw, rxdsp, rxhw`}

The dependencies (libraries) are within the same folder in order to ease the access to the different files through keyboard shortcuts (`__init__.py` not set yet)

The execution is going to automatically generate folders with the following scheme:

- **data-<year>-<month>-<day>**: permanent folder with all the generated result files
- **data-<year>-<month>-<day>-<X>**: <*X*>th temporary folder created at each execution which will be merged with the **data-<year>-<month>-<day>** if no error occur. Otherwise it will not be merged.

Data folders After the execution of the python files, the data folders will contain the following files:

- *<date> <hour> - <parameter name 1> <parameter value 1> - ... .csv*: contains the value of the metrics, the monitored values at each iteration of the algorithm
- *<date> <hour> - <parameter name 1> <parameter value 1> - ... .mat*: contains the value of some parameters and quantities that will need to be processed using Matlab scripts
- *<date> <hour> - <parameter name 1> <parameter value 1> - ... .<image extension>*: *<image extension>* can be {png, pdf, jpg, svg,...} and is the plot of the evolution of the values in the *.csv file
- *<date> <hour> - <parameter name 1> <parameter value 1> - ... .xml*: contains the minimal parameters of the simulation.

After the execution of the matlab files, the data folders will contain the following files:

- *matlab<date> <hour> - <parameter name 1> <parameter value 1> - ... -poincare.png*: SoP representation in the poincare sphere
- *matlab<date> <hour> - <parameter name 1> <parameter value 1> - ... .mat*: plot of SoP evolution and FIR filters at the last iteration

2.2.3 "Matlab" folder

Matlab home

This repository contains:

- `main_0_python2matlab`: extraction of channel parameters like the SoP angles, FIR filters, ...

The dependencies (libraries) are in the folder **`lib`**.

2.3 Further description

2.3.1 Python Libraries

The libraries are the following:

(1) libraries not specifically linked to the project:

maths: advanced mathematical operations.

matlab: Matlab-fashioned functions.

misc: Miscellaneous functions.

(2) libraries specifically linked to the project:

general: miscellaneous function related to the project

kit: **K**arlshrue **I**nstitute of **T**echnology library (named "shared_func" originally)

prop: functions related to the **prop**agation medium (here, optical fibre)

rxdsp: **r**eceiver side functions related to **d**igital **s**ignal **p**rocessing

rxhw: **r**eceiver side functions related to **h**ard**w**are

txdsp: **t**ransmitter side functions related to **d**igital **s**ignal **p**rocessing txhw] **t**ransmitter side functions related to **h**ard**w**are

Each library should contain an **updated** preamble divided into:

Information: such as authors, versioning etc.

Bibliography: divided into publications and source codes

Contents: alphabetically sorted list of functions

2.3.2 Functions

Here are some personal recommendations:

- Each function should be put be clearly identifiable (put in a cell or whatever)
- Each function should, as much as possible, use input and output **dictionaries**.
- We strongly recommend the developers to distinguish the parameters given using the International Unit System of units (referred as SI) or not, and to mention them in comments. In this documentation, the parameters are only given in this SI system without distinction, but are (mostly) clearly made in the code.

3 Description of chosen parameters

The channel parameters that can be set are:

residual chromatic dispersion:

we assume that the CD compensation is already performed at the receiver but not perfectly, leaving a residual chromatic dispersion. We chose to model it in terms of number of mixed symbols referred as `nSymbResCD`, equivalent to a residual propagation distance `DistRes`.

The chromatic dispersion has a closed form expression:

$$\hat{H}_{CD} \approx \exp\left(-\frac{1}{2}\beta_2 \cdot \texttt{DistRes} \cdot \Delta\omega^2\right)$$

Here the spectral span $\Delta\omega \approx 2\pi \cdot R_s$, and the equivalent time delay induced by CD is:

$$\Delta\tau_{CD,res} = \beta_2 \cdot \texttt{DistRes} \cdot \Delta\omega = \texttt{nSymbResCD}/R_s \implies \boxed{\texttt{DistRes} \approx \frac{\texttt{nSymbResCD}}{2\pi \cdot |\beta_2| \cdot R_s^2}}$$

The Chromatic dispersion parameter is then:

$$\boxed{\texttt{fibre['tauCD']} = \cdot\sqrt{2\pi \cdot \texttt{DistRes} \cdot |\beta_2|}} \tag{1}$$

polarisation mode dispersion:

The PMD and State of Polarisation changes are modelled as:

$$\hat{H}_{PMD} = R_0(\theta_k) \cdot \begin{pmatrix} \exp(+\texttt{fibre['tauPMD']}/2 \cdot \Delta\omega) & 0 \\ 0 & \exp(-\texttt{fibre['tauPMD']}/2 \cdot \Delta\omega) \end{pmatrix} R_1(\theta_k) \tag{2}$$

and the rotation matrices:

$$R_0(\theta) = \begin{pmatrix} \cos(\texttt{theta[0]}) & \sin(\texttt{theta[0]}) \\ -\sin(\texttt{theta[0]}) & \cos(\texttt{theta[0]}) \end{pmatrix} \quad \& \quad R_1(\theta) = \begin{pmatrix} \cos(\texttt{theta[1]}) & \sin(\texttt{theta[1]}) \\ -\sin(\texttt{theta[1]}) & \cos(\texttt{theta[1]}) \end{pmatrix} \tag{3}$$

with the angles `theta` in `fibre["thetas"][rx["Frame"]]` defined in the function `prop.set_thetas`. The value of `fibre["tauPMD]` is given by:

$$\boxed{\texttt{fibre['tauPMD']} = \texttt{fibre['PMD']} \cdot \sqrt{\texttt{fibre['DistProp']}}} \tag{4}$$

4 Perspectives

Here is a list without importance hierarchy of things that could be interesting to implement:

polarisation angle rotations: the current (optical) channel matrices only enable **one** rotation around the equator of the Poincare sphere

Phase Recovery: for Probabilistic Constellation Shaping after the CMA.

Bibliography

[1] Vincent Lauinger, Fred Buchali, and Laurent Schmalen. "VAE-Equalizer". In: *Source code* (2022). URL: https://github.com/kit-cel/vae-equalizer.