mpes Documentation

Release 0.1

R. Patrick Xian

Aug 15, 2018

Contents

the data processing package for (tr)ARPES data

CHAPTER 1

(tr)ARPES concepts:

In a photoemission process, an extreme UV or X-ray photon liberates an electron from the confines of the electronic potential within the material. ARPES directly measures the electronic energy and momentum parallel to the surface of the sample under study to infer the electronic states of the material. For a tutorial review on ARPES and its applications in physics and material science, see here. The data structure of ARPES is a stack of 2D images measured at different sample geometries, which are used to reconstruct the full static 3D band structure of the material.

TrARPES is an emerging technique that combines state-of-the-art ultrafast laser systems (~ fs resolution) with an existing ARPES experimental setup. TrARPES studies light-induced electronic dynamics such as phase transition, exciton dynamics, reaction kinetics, etc. It adds a time dimension, usually on the order of femtoseconds to nanoseconds, to the scope of ARPES measurements. Due to complex electronic dynamics, various coupling effects between the energy and momentum dimensions come into play in time. A complete understanding of the multidimensional time series from trARPES measurements can reveal dynamic constants crucial to the understanding of material properties and aid in simulation, design and further device applications.

CHAPTER 2

Contents:

2.1 Installation

Install from git repository

```
pip install git+https://github.com/RealPolitiX/mpes.git
```

Install from PyPI

2.2 File I/O & Processing

Custom methods to handle ARPES data I/O and standard data processing methods (filtering, dewarping, etc.)

2.3 Analysis

Data analysis pipeline including background removal, segmentation and fitting

2.4 Visualization

Custom methods for visualizing 2D-4D datasets for ARPES and beyond

2.5 Utility functions

Utility functions for the mpes package

@author: R. Patrick Xian

mpes.utils.**numFormatConversion**(*seq*, *form='int'*, ***kwds*)

When length keyword is not specified as an argument, the function returns a format-converted sequence of numbers

The function returns nothing when the conversion fails due to errors

Parameters

seq [1D numeric array] the numeric array to be converted

form [str | 'int'] the converted format

Return

numseq [converted numeric type] the format-converted array

mpes.utils.**replist**(*entry*, *row*, *column*)

Generator of nested lists with identical entries. Generated values are independent of one another.

Parameters

entry [numeric/str] repeated item in nested list

row [int] number of rows in nested list

column [int] number of columns in nested list

Return

nested list

mpes.utils.**revaxis**(*arr*, *axis=-1*)

Reverse an ndarray along certain axis

Parameters arr : nD numeric array

array to invert

axis [int | -1] the axis along which to invert

Return revarr : nD numeric array

axis-inverted nD array

mpes.utils.**shuffleaxis**(*arr*, *axes*, *direction='front'*)

Move multiple axes of a multidimensional array simultaneously to the front or end of its axis order

Parameters

arr [ndarray] array to be shuffled

axes [tuple of int] dimensions to be shuffled

direction [str | 'front'] direction of shuffling ('front' or 'end')

Return

sharr [ndarray] dimension-shuffled array

mpes.utils.**to_odd**(*num*)

Convert a single number to its nearest odd number

Parameters

num : float/int

Return

oddnum [int] the nearest odd number

CHAPTER 3

General data processing pipeline:

1. Load file (Igor binary or txt files)
2. Use image processing to find initial guess for fitting
3. Define the fitting model (model function and regularizer)
4. Carry out fitting

Python Module Index

m

Index