ColdTOFU

Release 0.0.3

Chetan Sriram Madasu

Oct 03, 2022

CONTENTS:

CHAPTER

ONE

COLDTOFU

1.1 ColdTOFU package

1.1.1 Subpackages

ColdTOFU.Physics package

Submodules

ColdTOFU.Physics.allenDeviation module

class ColdTOFU.Physics.allenDeviation.**TSeries**(*dataPath*, *channel*, *f0*, *Amplitude=None*, *offset=None*)

Bases: object

allenDev(*m*)

allenVar(*m*)

fracFreq()

oAllenDev(*m*)

oAllenVar(*m*)

timeDiff()

ColdTOFU.Physics.basisMatrices module

class ColdTOFU.Physics.basisMatrices.**GellMannBasis**

Bases: object

compose(*decomposition*)

decompose(*A*)

matrices()

structureConstant(*i*, *j*, *k*)

structureConstants()

class ColdTOFU.Physics.basisMatrices.**PauliBasis**

Bases: object

A class to represent Pauli's spin basis.

decompose(*A*)
Decomposes given non-singular 2×2 matrix in the Pauli basis.

Parameters **A** – 2×2 matrix to be decomposed.

Returns list of 4 components.

matrices()

Returns a list of 4 2×2 matrices corresponding to $I_2, \sigma_x, \sigma_y, \sigma_z$

ColdTOFU.Physics.createAndDestroy module

ColdTOFU.Physics.createAndDestroy.**a**(*N=5*)
Matrix representation of bosonic destruction operator.

Parameters

- **n** – destroy boson at n in an N state Fock space
- **N** – dimensions of the Fock space, default 5

Returns a sparse matrix representation of a of shape (N, N) in Fock basis, $\{|n_1, n_2, n_3, ..., N\rangle\}$.

ColdTOFU.Physics.createAndDestroy.**a_dag**(*N=5*)
Matrix representation of bosonic creation operator.

Parameters

- **n** – create a boson at n in an N state Fock space
- **N** – dimensions of the Fock space, default 5

Returns a sparse matrix representation of a of shape (N, N) in Fock basis, $\{|n_1, n_2, n_3, ..., N\rangle\}$.

ColdTOFU.Physics.spinInDickeBasis module

ColdTOFU.Physics.spinInDickeBasis.**S_m**(*s*)
Matrix representation of ladder lowering operator for spin, in general, angular momentum.

Parameters **s** – spin quantum number.

Returns a sparse matrix representation of S^- of shape (2s+1, 2s+1) in Dicke basis, $\{|s, m_s\rangle\}$.

ColdTOFU.Physics.spinInDickeBasis.**S_p**(*s*)
Matrix representation of ladder raising operator for spin, in general, angular momentum.

Parameters **s** – spin quantum number.

Returns a sparse matrix representation of S^+ of shape (2s+1, 2s+1) in Dicke basis, $\{|s, m_s\rangle\}$.

ColdTOFU.Physics.spinInDickeBasis.**S_x**(*s*)
Matrix representation of x component of the operator for spin, in general, angular momentum.

Parameters **s** – spin quantum number.

Returns a sparse matrix representation of S_x of shape (2s+1, 2s+1) in Dicke basis, $\{|s, m_s\rangle\}$.

ColdTOFU.Physics.spinInDickeBasis.**S_y**(*s*)
Matrix representation of y component of the operator for spin, in general, angular momentum.

Parameters **s** – spin quantum number.

Returns a sparse matrix representation of S_y of shape (2s+1, 2s+1) in Dicke basis, $\{|s, m_s\rangle\}$.

ColdTOFU.Physics.spinInDickeBasis.**S_z**(*s*)
Matrix representation of z component of the operator for spin, in general, angular momentum.

Parameters **s** – spin quantum number.

Returns a sparse matrix representation of S_z of shape (2s+1, 2s+1) in Dicke basis, $\{|s, m_s\rangle\}$.

ColdTOFU.Physics.spinInDickeBasis.**Spin**(*s*)
Matrix representation of quantum mechanical spin, in general, angular momentum, s.

Parameters **s** – int or half int, the spin quantum number.

Returns a tuple of sparse matrices corresponding to S_x, S_y, S_z

ColdTOFU.Physics.spinInDickeBasis.**SpinAngularMomenta**(*I*, *L*, *S*)
Returns angular momenta operators of a state given I, L, S in the tensor product basis.

Parameters

- **I** – nuclear spin quantum number of the atomic state
- **L** – orbital angular momentum quantum number
- **S** – spin quantum number of the state

Returns a tuple of angular momenta, $((I_x, I_y, I_z), (L_x, L_y, L_z), (S_x, S_y, S_z))$ (each a tuple of components of $\mathbf{I} \otimes \mathbf{L} \otimes \mathbf{S}$ as sparse matrices) in tensor product space.

Module contents

1.1.2 Submodules

1.1.3 ColdTOFU.AndorSifReader module

class ColdTOFU.AndorSifReader.**AndorSifFile**(*filename*)
Bases: object

This is main class to handle the reading of sif files. All the work is done in the constructor, the sif file is opened, data is read and the file is closed. This is adopted from https://github.com/ardiloot/AndorSifReader with modifications.

Parameters **filename** – str, Path to .sif file to read

signal
_SifFrame Instance of the _SifFrame helper class to store signal information.

data
ndarray, Convenience property linking to signal.data.

props
dict, Convenience property linking to signal.props.

ANTSIF_PROPS = ['Type', 'Active', 'Version', 'Time', 'FormattedTime', 'FileName', 'Tem

ATSIF_AT_32 = 1073741826

ATSIF_AT_64 = 1073741828

ATSIF_AT_8 = 1073741824

ATSIF_AT_U32 = 1073741827

ATSIF_AT_U64 = 1073741829

ATSIF_AT_U8 = 1

ATSIF_Background = 1073741826

ATSIF_CONV = {1: <class 'numpy.uint8'>, 1073741824: <class 'numpy.int8'>, 1073741826

ATSIF_CalibX = 1073741824

ATSIF_CalibY = 1073741825

ATSIF_CalibZ = 1073741826

ATSIF_Double = 1073741831

ATSIF_Float = 1073741830

ATSIF_Live = 1073741827

ATSIF_ReadAll = 1073741824

ATSIF_ReadHeaderOnly = 1073741825

ATSIF_Reference = 1073741825

ATSIF_Signal = 1073741824

ATSIF_Source = 1073741828

ATSIF_String = 1073741832

property data

property props

class ColdTOFU.AndorSifReader.**SifError**

Bases: object

This is a helper class to handle error codes produced by SIFReaderSDK.

ATSIF_DATA_NOT_PRESENT = 'ATSIF_DATA_NOT_PRESENT'

ATSIF_FILE_ACCESS_ERROR = 'ATSIF_FILE_ACCESS_ERROR'

ATSIF_FILE_NOT_FOUND = 'ATSIF_FILE_NOT_FOUND'

ATSIF_NO_SIF_LOADED = 'ATSIF_NO_SIF_LOADED'

ATSIF_P1INVALID = 'ATSIF_P1INVALID'

ATSIF_P2INVALID = 'ATSIF_P2INVALID'

ATSIF_P3INVALID = 'ATSIF_P3INVALID'

ATSIF_P4INVALID = 'ATSIF_P4INVALID'

ATSIF_P5INVALID = 'ATSIF_P5INVALID'

ATSIF_P6INVALID = 'ATSIF_P6INVALID'

ATSIF_P7INVALID = 'ATSIF_P7INVALID'

ATSIF_P8INVALID = 'ATSIF_P8INVALID'

ATSIF_SIF_FORMAT_ERROR = 'ATSIF_SIF_FORMAT_ERROR'

ATSIF_SUCCESS = 'ATSIF_SUCCESS'

ERROR_STR = {22002: 'ATSIF_SUCCESS', 22003: 'ATSIF_SIF_FORMAT_ERROR', 22004: 'ATSIF

static FromCode(*errorCode*)

Converts SIFReaderSDK error code to string.

Parameters **errorCode** (*int*) – Error code number

Returns Corresponding error string.

Return type str

static ProcessErrorCode(*errorCode*)
Processes error codes, raises RuntimeError if the error code indicate problems. In case of success, does nothing.

Parameters **errorCode** (*int*) – Error code number

1.1.4 ColdTOFU.Images module

class ColdTOFU.Images.**FluorescenceImage**(*filePath*)
Bases: object

The main class to extract relevant information from the fluorescence image sequences that are obtained in the experiment. The multiple image sequence consists of fluorescence signal image and a background image acquired immediately after the signal from the atomic cloud for as many runs of the experiment.

Parameters **filePath** – str, Path to the multiple-image file

filePath
str, Path to the multiple-image file

im
PIL.TiffImagePlugin.TiffImageFile or AndorSifFile._SifFrames

n
int, total number of data = number of images/3

frames
ndarray, all the frames in the image file

tags
dict, the named tag dictionary of the TiffImage or properties of sif file

averagedSignal(*nAveraging*, *truncate=()*)
Calculates the average signal with nAveraging being the Superloop in the experiment and finds the fluorescence after the averaging.

Parameters

- **nAveraging** –

  type int, the repetitions of the experiment

- **truncate** –

  type tuple or list, the super loops to be ignored.

Returns an ndarray of length equal to Serie in the experiment.

fluorescence(*xSpan*, *ySpan*)
Returns the fluorence calculated from the FluorencenceImage as an ndarray.

Parameters

- **xSpan** – a list containing the range of pixels of the image in x direction.
- **ySpan** – a list containing the range of pixels of the image in y direction.

images()
Returns the images present in the multiple image file as an ndarray.

plotAveragedSignal (*nAveraging*, *ROI*, *truncate=()*)
Calculates and plots the average signal with nAveraging being the Superloop in the experiment.

Returns None.

class ColdTOFU.Images.**ShadowImage** (*filePath*)
Bases: object

The main class to extract relevant information from the shadow imaging sequences that are obtained in the experiment. The multiple image sequence consists of the shadow or the absorption image, image of the incident probe and the image of the background for as many runs of the experiment.

Parameters **filePath** – str, Path to the multiple-image file

filePath
str, Path to the multiple-image file

ext
str, extension of the image file passed.

im
PIL.TiffImagePlugin.TiffImageFile or AndorSifFile._SifFrames

n
int, total number of data = number of images/3

frames
ndarray, all the frames in the image file

tags
dict, the named tag dictionary of the TiffImage or properties of sif file

transmission
ndarray, All the transmission images after subtracting the background

incidence
ndarray, All the images of incident probe after subtracting the background

OD
ndarray, optical depth all the runs of the experiment calculated from the transmission and incidence

ODaveraged
ndarray, OD averaged with averaging like a superloop

averagedOD
ndarray, OD of the averaged signal with averaging on superloop

averagedOD2
ndarray, OD of the averaged signal with averaging on loop

averagedSignalOD (*nAveraging*, *truncate=()*)
Calculates the average signal with nAveraging being the Superloop in the experiment and finds the optical depth after the averaging.

Parameters

- **nAveraging** – int, number of averaging to be used.
- **truncate** – tuple or list, the superloops to be ignored while averaging. Default is ().

Returns an ndarray of length equal to Serie in the experiment.

averagedSignalOD2 (*nAveraging*, *truncate=()*)
Calculates the average signal with nAveraging being the loops in the experiment and finds the optical depth after the averaging.

Parameters

- **nAveraging** – int, number of averaging to be used.
- **truncate** – tuple or list, the superloops to be ignored while averaging. Default is ().

Returns an ndarray of length equal to Serie in the experiment.

blueProbeIntensity(*ROI*)

Estimates blue probe intensity and power from the reference image using the losses in imaging system and quantum effeiciency of the camera for red light. rcParams used in calculating the intensity are `quantum efficiency`, `pixelSize`, `binning` and `magnification`.

Parameters **ROI** – list, ROI in which to estimate the intensity.

Returns intensity ($\mu W/cm^2$) and power (μW)

comment(*comment*)

Adds comment to the tif image under the ImageDescription tag and replaces it silently with the new image, if the user has the permissions.

Parameters **comment** – str, comment to be added.

description()

Returns the ImageDescription tag of the tiff image file.

images()

Returns the images present in the ShadowImage object as an ndarray.

opticalDepth(*xSpan*, *ySpan*)

Returns the optical depth calculated from the ShadowImage as an ndarray.

Parameters

- **xSpan** – list, containing the range of pixels of the image in x direction.
- **ySpan** – list, containing the range of pixels of the image in y direction.

opticalDepthAveraged(*nAveraging*)

Calculates the optical depth from every triad of the images and the take average with nAveraging as the Superloop in the experiment.

Parameters **nAveraging** – int, number of averaging to be used.

Returns an ndarray of length equal to Serie in the experiment.

plotAveragedSignalOD(*nAveraging*, *ROI*, *truncate=()*)

Calculates and plots the average signal with nAveraging being the Superloop in the experiment and finds the optical depth after the averaging.

Parameters

- **nAveraging** – int, number of averaging to be used.
- **ROI** – list, ROI in the images to be plotted.
- **truncate** – tuple or list, the superloops to be ignored while averaging. Default is ().

Returns None.

redProbeIntensity(*ROI*)

Estimates red probe intensity and power from the reference image using the losses in imaging system and quantum effeiciency of the camera for red light. rcParams used in calculating the intensity are `quantum efficiency`, `pixelSize`, `binning` and `magnification`.

Parameters **ROI** – list, ROI in which to estimate the intensity.

Returns intensity ($\mu W/cm^2$) and power (μW)

class ColdTOFU.Images.**rcParams**
Bases: object

A class that is designed to read and update resource parameters of the package. These parameters are usually the hardware parameters that are used while converting certain digital values to real units. Currently these are imaging parameters like magnification, pixel size etc. The idea behind this is that once these parameters are set, these are used by many function and classes throughout the package without having to explicitly pass them. Open the file rcParams.json using the notepad to see the current parameters used throughout the library.

params
dict, current parameters used by the package

update(*key*, *value*)
A method to update the rcParams.

Parameters

- **key** – string, key of the parameter. Careful about the spelling of the key. If its wrong, then you will add another parameter with the wrong spelling instead of updating the desired parameter.
- **value** – value of the parameter to update.

1.1.5 ColdTOFU.fits module

ColdTOFU.fits.**gaussian**(*x*, *amplitude*, *xo*, *sigma_x*, *offset*)

ColdTOFU.fits.**gaussian2D**(*X*, *amplitude*, *xo*, *yo*, *sigma_x*, *sigma_y*, *theta*, *offset*)

ColdTOFU.fits.**gaussian2DFit**(*image*, *p0=None*, *bounds=[(), ()]*, *plot=True*, *title=''*)
Fits an image with a 2D gaussian.

Parameters

- **image** – numpy ndarray
- **p0** – ndarray, initial guess for the fit params in the form of [amplitude, xo, yo, sigma_x, sigma_y, theta, offset]. Default None (fits for OD images).
- **bounds** – tuple of lower bound and upper bound for the fit. Default None (fits for OD images)
- **plot** – bool to show the plot of the fit. Default True.

Returns optimized parameters in the same order as p0 pCov: covarience parameters of the fit. Read scipy.optimize.curve_fit for details.

Return type pOpt

ColdTOFU.fits.**gaussianFit**(*x*, *array*, *p0=[]*, *bounds=[(), ()]*, *plot=True*)
Fits the given array to an 1D-gaussian.

Parameters

- **x** – 1darray, the argument values of the gaussian
- **array** – 1darray, the data to fit to the gaussian
- **p0** – ndarray, initial guess for the fit params in the form of [amplitude, xo, sigma, offset]. Default is None.
- **bounds** – tuple of lower bound and upper bound for the fit. Default is None.

Returns optimized parameters in the same order as p0 pCov: covarience parameters of the fit. Read scipy.optimize.curve_fit for details.

Return type pOpt

ColdTOFU.fits.**lorentzian**(*x*, *amplitude*, *xo*, *gamma*, *offset*)

ColdTOFU.fits.**lorentzianFit**(*x*, *array*, *p0=[]*, *bounds=[(), ()]*, *plot=False*)
Fits the given array to a Lorentzian.

Parameters

- **array** – 1darray, the data to fit to the lorentzian
- **p0** – ndarray, initial guess for the fit params in the form of [amplitude, xo, gamma, offset]. Default is None.
- **bounds** – tuple of lower bound and upper bound for the fit. Default is None.

Returns optimized parameters in the same order as p0 pCov: covarience parameters of the fit. Read scipy.optimize.curve_fit for details.

Return type pOpt

ColdTOFU.fits.**multipleGaussian2D**(*X*, **args*)

ColdTOFU.fits.**multipleGaussian2DFit**(*image*, *p0*, *bounds*, *TOF*, *plot=True*, *cropSize=6*, *tolerence=0.3*, *logNorm=False*)

ColdTOFU.fits.**threeGaussian2D**(*X*, **args*)

ColdTOFU.fits.**threeGaussian2DFit**(*image*, *p0*, *bounds*, *TOF*, *plot=True*, *cropSize=6*, *logNorm=False*)

1.1.6 ColdTOFU.jupyterTools module

ColdTOFU.jupyterTools.**disp**(*idx*, *symObj*)
Displays sympy symbolic objects in latex as an equation.

Parameters

- **idx** – string, latex code of varaible in RHS.
- **symObj** – sympy.symbol object, symbol object to be displayed in LHS.

Returns None.

ColdTOFU.jupyterTools.**update_progress**(*progress*, *clear=True*, *comment=''*)
Progress bar generated in a loop.

Parameters

- **progress** – int, current index
- **clear** – bool, to clear the last output of the cell. Default True
- **comment** – string, description of the progress bar.

Returns None

1.1.7 ColdTOFU.numberOfAtoms module

ColdTOFU.numberOfAtoms.**numAtomsBlue**(*image*, *delta*, *s=0*, *plot=True*, *p0=None*, *bounds=[(), ()]*)
Calculates number of atoms from blue shadow imaging.

Parameters

- **image** – a numpy.ndarray, OD from the experiment
- **delta** – a float, detuning of the probe, 2*(AOMFreq-69) MHz
- **s** (*optional*) – a float, saturation parameter of the probe. Default is 0.
- **plot** (*optional*) – a bool, flag to plot the gaussian fits if True. Default is True.
- **p0** (*optional*) – a list, initial guess parameters corresponding to gaussian2D function
- **bounds** (*optional*) – list of lists, bounds for parameters in the form *[[lower bounds], [upper bounds]]*.

Returns a tuple, *(number of atoms from 2D gaussian fit, number of atoms from pixel sum, number density from gaussian fit, sigma_x, sigma_y, amplitude, x0, y0)*

ColdTOFU.numberOfAtoms.**numAtomsGeneral**(*image*, *delta*, *wLen*, *Gamma*, *s=0*, *plot=True*, *p0=None*, *bounds=[(), ()]*)
Calculates number of atoms from blue shadow imaging.

Parameters

- **image** – a numpy.ndarray, OD from the experiment
- **delta** – a float, detuning of the probe in MHz
- **wLen** – float, wavelength of the probe in nm
- **Gamma** – flaot, linewidth of the excited state in MHz.
- **s** (*optional*) – a float, saturation parameter of the probe. Default is 0.
- **plot** (*optional*) – a bool, flag to plot the gaussian fits if True. Default is True.
- **p0** (*optional*) – a list, initial guess parameters corresponding to gaussian2D function
- **bounds** (*optional*) – list of lists, bounds for parameters in the form *[[lower bounds], [upper bounds]]*.

Returns a tuple, *(number of atoms from 2D gaussian fit, number of atoms from pixel sum, number density from gaussian fit, sigma_x, sigma_y, amplitude, x0, y0)*

ColdTOFU.numberOfAtoms.**numAtomsRed**(*image*, *delta*, *s=0*, *plot=True*, *p0=None*, *bounds=[(), ()]*)
Calculates number of atoms from red shadow imaging.

Parameters

- **image** – a numpy.ndarray, OD from the experiment
- **delta** – float, detuning of the probe in kHz
- **s** (*optional*) – a float, saturation parameter of the probe. Default is 0.
- **plot** (*optional*) – a bool, a flag to plot the gaussian fits if True. Default is True.
- **p0** (*optional*) – a list, initial guess parameters corresponding to gaussian2D function
- **bounds** (*optional*) – list of lists, bounds for parameters in the form *[[lower bounds], [upper bounds]]*.

Returns a tuple, *(number of atoms from 2D gaussian fit, number of atoms from pixel sum, number density from gaussian fit, sigma_x, sigma_y, amplitude, x0, y0)*

ColdTOFU.numberOfAtoms.**temperature**(*sizes*, *timeStamps*, *plot=True*)

Calculates temperature of the cold gas given size of the cloud at various times of flight.

Parameters

- **sizes** – 2D array, sizes of the cloud extracted from gaussian2D fit in pixel units. Eg: ([[x1, y1], [x2, y2], [x3, y3]])
- **timeStamps** – array, corresponding times of flight in s.
- **plot** – bool, to plot v^2 vs TOF^2. Default:True.

Returns a tuple, (temperature in K along x, temperature in K along y)

1.1.8 ColdTOFU.picoMatTools module

ColdTOFU.picoMatTools.**PSD**(*path*, *avg=30*, *channels=['A']*)

Reads multiple waveforms .mat files folder exported using picoscope software and alculates power spectral density (PSD).

Parameters

- **path** – string, path of the folder to read.
- **avg** – int, number of waveforms in the folder to average.
- **channels** – list of strings, channels of the file to read. Ex: *['A', 'B']*

Returns a touple, (frequencies array, list of PSD arrays in the order of channels)

ColdTOFU.picoMatTools.**RIN**(*path*, *avg=30*, *channel='A'*)

Reads multiple waveforms .mat files folder exported using picoscope software and calculates relative intensity noise (RIN).

Parameters

- **path** – string, path of the folder to read.
- **avg** – int, number of waveforms in the folder to average.
- **channels** – list of strings, channels of the file to read. Ex: *['A', 'B']*

Returns a touple, (frequencies array, list of RIN arrays in the order of channels)

ColdTOFU.picoMatTools.**picoMatRead**(*filePath*, *channels=['A']*)

Reads .mat files exported using picoscope software.

Parameters

- **filePath** – string, path of the file to read.
- **channels** – list of strings, channels of the file to read. Ex: *['A', 'B']*

Returns a touple, (time array, list of data arrays)

1.1.9 ColdTOFU.sigma module

ColdTOFU.sigma.**sigmaBlue**(*delta*, *A*, *s=0*)

Calculates the scattering cross-section of Sr for $^1S_0 \rightarrow^1 P_1$ taking into account the isotope shift w.r.t. A=88 isotope and hyperfine levels present in the excited state.

Parameters

- **delta** – float, detuning of the probe w.r.t the reference in MHz.
- **s** – float, saturation parameter of the probe, I/I_s.
- **A** – int, mass number of the isotope.

Returns A float, Scattering cross-section.

Comment: All frequencies are in MHz.

ColdTOFU.sigma.**sigmaGeneral**(*delta*, *wLen*, *Gamma*, *s=0*)

Calculates the scattering cross-section of a two-level atom with a weak probe.

Parameters

- **delta** – float, detuning of the probe w.r.t the reference in MHz.
- **wLen** – float, wavelength of the probe in nm
- **Gamma** – flaot, linewidth of the excited state in MHz.
- **s** – float, saturation parameter of the probe, I/I_s.

Returns A float, Scattering cross-section.

Comment: All frequencies are in MHz.

ColdTOFU.sigma.**sigmaRed**(*delta*, *s=0*)

Calculates the scattering cross-section of Sr for $^1S_0 \rightarrow^3 P_1$ taking saturation into account.

Parameters

- **delta** – float, detuning of the probe w.r.t the reference in kHz.
- **s** – float, saturation parameter of the probe, I/I_s.

Returns A float, Scattering cross-section.

Comment: All frequencies are in kHz.

1.1.10 ColdTOFU.spectrum module

ColdTOFU.spectrum.**bv**(*f*, *f0*, *b0*, *T*, *s*)

Function representing convolution of the lorentzian line shape of the red transition and gaussian maxwell distribution. This is taken from 5.13 from Chang chi's thesis and added the effect of saturation parameter.

Parameters

- **f** – numpy.array, frequency vector
- **f0** – float, resonance frequency or centre of the spectrum
- **b0** – float, optical depth at resonance at zero temperature
- **T** – float, temperature in micro K

- **s** – float, saturation parameter of the probe, I/I_s.

Returns optical depth for frequencies f in the shape f.

ColdTOFU.spectrum.**bvFit**(*f*, *array*, *p0=None*, *bounds=None*)

Function to fit spectroscopy data to real line shape of the transition.

Parameters

- **f** – numpy.array, frequency vector
- **array** – float, optical depth at scan frequencies f
- **p0** – initial guess for the fit as [f_0, b_0, T (in μK), s]
- **bounds** – bounds for the fit as ([lower bounds], [upper bounds])

Returns (pOpt, pCov)

Return type a tuple with optimized parameters and covariance ex

ColdTOFU.spectrum.**spectroscopy**(*ODimages*, *f*, *d=4*, *loss=False*, *plot=True*, *fileNum=''*, *savefig=False*)

Adds the OD of the pixels around the centre and uses the sum to plot the spectrum of the scan corresponding to the given frequencies. This is then fit to a lorentzian to find the center and linewidth.

Parameters

- **ODimages** – ODimages extracted from ShadowImaging sequences
- **f** – array of frequencies for which the scan is done
- **d** – int, to specify size of the image area to consider around the centre of OD
- **plot** – bool, default is True to specify if the data has to be plotted
- **fileNum** – string, file number (plus any additional description) of the image file for which the analysis is done.
- **savefig** – bool, default is False. Change it to true if you want to save the spectrum as .png

Returns a tuple, (amp, centre, gamma, offset) of the lorentzian fit

ColdTOFU.spectrum.**spectroscopyFaddeva**(*ODimages*, *f*, *imaging_params*, *plot=True*, *atom_loss=False*, *fileNum=''*, *savefig=False*)

Fits od images to a gaussian and uses its amplitude to plot the spectrum of the scan corresponding to the given frequencies. This is fit to $b_v(\delta)$ from Chang Chi's thesis to extract temperature in addition to center.

Parameters

- **ODimages** – ODimages extracted from ShadowImaging sequences
- **f** – array of frequencies for which the scan is done
- **imaging_params** – a dictionary with keys as follows ex: {'binning':2, 'magnification': 2.2, 'pixelSize': 16*micro, 'saturation': 1 }
- **plot** – bool, default is True to specify if the data has to be plotted
- **fileNum** – string, file number (plus any additional description) of the image file for which the analysis is done.
- **savefig** – bool, default is False. Change it to true if you want to save the spectrum as .png

Returns a tuple, (centre, b_0, T, s) of fit to $b_v(\delta)$

1.1.11 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C