UCL project Documentation

Release 1

Miriam Gerharz

Apr 01, 2020

CONTENTS:

CHAPTER

ONE

TOOLS

1.1 Calculating spectrum

`tools.chi` (*_omega*, *_omega_j*, *_Gamma*)
Calculates what is defined as chi in the paper

Parameters

- **_omega** (*1D numpy array*) – The frequency range of which chi shall be calculated
- **_omega_j** (*float*) – respective mechanical frequency
- **_Gamma** (*float*) – Damping (either $Gamma$ or $kappa$)

Returns chi(omega)

Return type np.array

`tools.eta` (*_omega*, *_detuning*, *_phi*, *_kappa*)
Calculates optical susceptibility

Parameters

- **_omega** (*numpy array*) – The frequency range of which chi shall be calculated
- **_detuning** (*float*) – The detuning
- **_phi** (*float*) – Phase ???
- **_kappa** (*float*) – cavity linewidth

Returns eta(omega)

Return type np.array

`tools.mu` (*_omega*, *_omega_j*, *_Gamma*)
Calculates the mechanical susceptibilities

Parameters

- **_omega** (*1D numpy array*) – The frequency range of which chi shall be calculated
- **_omega_j** (*numpy array of length 3*) – mechanical frequencies
- **_Gamma** (*float*) – Damping (either $Gamma$ or $kappa$)

Returns mu(omega)

Return type np.array

`tools.M` (*_omega*, *_omega_j*, *_detuning*, *_phi*, *_Gamma*, *_kappa*, *_g*)
Calculates the normalization factor

Parameters

- **_omega** (*1D numpy array*) – The frequency range of which chi shall be calculated
- **_omega_j** (*numpy array of length 3*) – mechanical frequencies
- **_detuning** (*float*) – Detuning
- **_phi** (*np.array*) – [0,0,pi/2]
- **_Gamma** (*float*) – Damping (either $Gamma$ or $kappa$)
- **_kappa** (*float*) – linewidth of cavity
- **_g** (*np.array*) – Couplings (g_x, g_y, g_z, g_xy, g_yz, g_zx)

Returns M(omega, mode)

Return type 2D np.array

tools.**Q_opt**(*_omega*, *_detuning*, *_kappa*, *_phi*)

Calculates optical noise

Parameters

- **_omega** (*1D numpy array*) – The frequency range of which chi shall be calculated
- **_detuning** (*float*) – The detuning
- **_kappa** (*float*) – cavity linewidth
- **_phi** (*float*) – Phase ???

Returns Q_opt(omega, mode)

Return type 2D np.array

tools.**Q_mech**(*_omega*, *_omega_j*, *_Gamma*)

Calculates the mechanical noises

Parameters

- **_omega** (*1D numpy array*) – The frequency range of which chi shall be calculated
- **_omega_j** (*numpy array of length 3*) – mechanical frequencies
- **_Gamma** (*float*) – Damping (either $Gamma$ or $kappa$)

Returns Q_mech(omega, mode)

Return type 2D np.array

tools.**q_1D**(*_omega*, *_omega_j*, *_detuning*, *_g*, *_Gamma*, *_kappa*, *_phi*)

Calculates the operator q_j $propto$(b_j+b_j^dagger) without taking into account the 3D contributions

Parameters

- **_omega** (*1D numpy array*) – The frequency range of which chi shall be calculated
- **_omega_j** (*numpy array of length 3*) – mechanical frequencies
- **_detuning** (*float*) – Detuning
- **_g** (*np.array*) – Couplings (g_x, g_y, g_z, g_xy, g_yz, g_zx)
- **_Gamma** (*float*) – Damping (either $Gamma$ or $kappa$)
- **_kappa** (*float*) – linewidth of cavity
- **_phi** (*np.array*) – [0,0,pi/2]

Returns q(omega, mode) 1D

Return type 2D np.array

`tools.`**`q_3D`**(*_omega*, *_omega_j*, *_detuning*, *_g*, *_Gamma*, *_kappa*, *_phi*)

Calculates the operator q_j $propto$(b_j+b_j^dagger) with taking into account the 3D contributions

Parameters

- **`_omega`** (*`1D numpy array`*) – The frequency range of which chi shall be calculated
- **`_omega_j`** (*`numpy array of length 3`*) – mechanical frequencies
- **`_detuning`** (*`float`*) – Detuning
- **`_g`** (*`np.array`*) – Couplings (g_x, g_y, g_z, g_xy, g_yz, g_zx)
- **`_Gamma`** (*`float`*) – Damping (either $Gamma$ or $kappa$)
- **`_kappa`** (*`float`*) – linewidth of cavity
- **`_phi`** (*`np.array`*) – [0,0,pi/2]

Returns q(omega, mode) 3D

Return type 2D np.array

`tools.`**`expectation_value`**(*_operator*, *_n*, *_pair*)

Calculates the expectation value of an operator by analyzing the noises

Parameters

- **`_operator`** (*`np.array`*) – operator as function of omega (containing all directions)
- **`_n`** (*`float`*) – Expectation value of the respective noise
- **`_pair`** (*`integer`*) – select pair (0=photon, 1=x, 2=y, 3=z)

Returns <operator>(omega)

Return type np.array

`tools.`**`spectrum`**(*_operator*, *_n_opt*, *_n_mech*)

Calculates the PSD

Parameters

- **`_operator`** (*`np.array`*) – operator as function of omega (containing all directions)
- **`_n_opt`** (*`float`*) – optical photon number (n_opt=0)
- **`_n_mech`** (*`np.array`*) – phonon numbers (n_x, n_y, n_z)

Returns <operator>_total(omega) (sum over all modes)

Return type np.array

`tools.`**`spectrum_output`**(*omega*, *_i*, *param*, *ThreeD*)

Calculates the PSD for a given omega regime and set of parameters

Parameters

- **`omega`** (*`np.array`*) – Frequency range in which the spectrum is to be computed
- **`_i`** (*`integer`*) – selection of operator (0=photon, 1=x, 2=y, 3=z)
- **`param`** (*`class param`*) – set of parameters
- **`ThreedD`** (*`boolean`*) – Consider 3D contribution (True) or not (False)

Returns PSD(omega)

Return type np.array

1.2 Phonon numbers

`tools.`**`n_from_area`**(*_S_plus*, *_S_minus*, *_Delta_omega*, *_N=0*, *_name=''*, *printing=True*)

Calculates phonon number from area and compares it to the one from the formula

Parameters

- **_S_plus** (*np.array*) – Spectrum for positive omega
- **_S_minus** (*np.array*) – Spectrum for negative omega
- **_Delta_omega** (*float*) – Spacing of omega
- **_N** (*float*) – Phonon number from formula
- **_name** (*str*) – Name of respective operator (x, y or z)
- **printing** (*boolean*) – Print the result (True), default is True

Returns Phonon numbers (N_plus, N_minus, N_total)

Return type list

`tools.`**`photon_number`**(*_n_j*, *_Gamma_opt*, *_Gamma*, *printing=True*)

Calculates the phonon number from the formula

Parameters

- **_n_j** (*np.array*) – phonon numbers at room temperature
- **_Gamma_opt** (*np.array*) – optical damping rate (x,y,z)
- **_Gamma** (*float*) – mechanical damping rate
- **printing** (*boolean*) – Print the result (True), default is True

Returns Phonon numbers (N_plus, N_minus, N_total)

Return type np.array

1.3 Parameters

class `tools.`**`parameters`**

This class contains all relevant parameters

DelFSR = 14000000000.0

Free spectral range [Hz], not used if couplings are given

EPSR = 2.1

relative permittivity [F m^-1]

Finesse = 73000.0

Finesse

Pin1 = 0.4

input power tweezer beam [W]

Press = 1e-06
air pressure [mbar]

R0 = 7.15e-08
sphere radius [m]

RHO = 2198
sphere density [kg/m^3]

WX = 6.7e-07
focus of tweezer in x-direction [m]

WY = 7.7e-07
focus of tweezer in y-direction [m]

X0 = 2.4472000000000004e-07
equilibrium position in x [m]

XL = 0.0107
cavity length [m]

Y0 = 0
y_0, equilibrium position in x-direction

Z0 = 0
z_0, equilibrium position in x-direction

detuning = -300000.0
detuning of trap beam (omega_cav - omega_tw) [2pi kHz]

lambda_tw = 1.064e-06
wavelength of tweezer [m]

n_opt = 0
Photon number at room temperature

opt_damp_rate(*printing=False*)
Calculates the optical damping rate

Parameters **printing** (*boolean*) – Result is printed (True) or not, default True

Returns Optical damping rate for (x,y and z)

Return type np.array

> **Warning:** Detuning has to be given in 2pi Hz

prepare_calc()
Calculates all the theoretical relevant parameters if only the experimental ones are given

> **Warning:** Detuning has to be given in 2pi Hz

print_param()
Prints all the parameters in a nice fashion

theta0 = 0.25
angle between tweezer polarization and cavity axis [pi]

waist = 4.11e-05
waist radius [m]

1.4 Helpers

`tools.`**`area`**(*_S*, *_Delta*)

Calculates area under curve by using the trapezoidal rule

Parameters

- **`_S`** (*`np.array`*) – spectrum
- **`_Delta`** (*`float`*) – spacing of omega

Returns Area under the spectrum

Return type float

`tools.`**`loop_progress`**(*L_inner*, *L_outer*, *inner*, *outer*, *start_time*)

Print nice progress control in terminal

Parameters

- **`L_inner`** (*`integer`*) – length of inner loop
- **`L_outer`** (*`integer`*) – length of outer loop
- **`inner`** (*`integer`*) – current value of loop parameter of inner loop
- **`outer`** (*`integer`*) – current value of loop parameter of outer loop
- **`start_time`** (*`float`*) – time when loops where started

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

t

INDEX