Demo 8 cheat sheet: Infrared spectroscopy

qworld library

Function	Purpose	Inputs
qworld.probabilityDensity(psi_x)	Return the probability density associated to a wavefunction	**psi_x: array of wavefunction values**
qworld.my_plotting_function(x,functions_list,labels,title='Plot',xlab='x',ylab='f(x)',fts=12,lw=2,fs=(10,8))	Return a plot	- x: array with x values functions_list: list of arrays representing functions you want to plot labels: list of labels. It should have the same size as functions_list title: title of the plot (Default: 'Plot') other parameters defined by default
qworld normalize_wf(x, psi_x)	Return a normalized wavefunction	**psi_x: array of wavefunction values**

Miscellaneous (Libraries, arrays and integration)

Function	Purpose	Inputs
dir(library)	Display the names of all the function in a module/library	Name of the library (as importe
help(name_of_function)	Return the documentation of the function.	name of the function
np.arange(-1.0,1.0, 0.1)	Return an array evenly spaced values within a given interval according to a step size.	x_init, x_end, step_size
np.linspace(-1.0,1.0, 100)	Return an array of evenly spaced numbers over a specified interval.	x_init, x_end, number of points
np.exp(x)	Return the value of the Euler number exponentiated to x	x: exponent (number)
simps(f_x, x) (as imported from scipy.integrate)	Integrate a function f(x) represented as an array (f_x) defined on the grid x using Simpson's rule.	f_x = vector with function values , x= vector representing the grid of x values

Demo specific common variables and functions:

x == array with position values
t == time value
t_array == array with time values
omega_f=frequency of the field
omega_array == array with different values of \omega_f
c1_t == array with values of the c_1 for different times
prop_excited_state == array with the values of c1_t squared
IR_array == array with transition probabilities for the first excited state for different values of omega_f.

Function	Purpose	Inputs
time_dependent_psi(x, t, omega_f, omega_0 = 1, Lam = 1, E_0 = 1, m=1, hbar=1):	**Returns psi(x,t) for a harmonic oscillator under a time dependent field**	**x: array of position; t : time; omega_f : frequency of the field/laser that interacts with the particle in the Harmonic Oscillator**
overlap(t, omega_f, omega_0 = 1, Lam = 1, E_0 = 0.1, m=1, hbar=1):	**Returns the overlap of the solution to the harmonic oscillator interacting with a field and the first excited state of the standard harmonic oscillator.**	**t: array of time values; omega_f : requency of the field/laser that interacts with the particle in the Harmonic Oscillator**

Matplotlib : Plotting (plt)

Usage	Purpose	Inputs
plt.figure(figsize=(10,8))	Setup parameter for a graphic, in this case we will use it change size.	figsize= (inches width, inches height)
plt.plot(x,y)	Plot lines	**x, y = vectors**
plt.xlabel("Axis x name")	Set the *x* axis label of the current plot.	**Name = string**
plt.xlim([xmin,xmax])	Set the *x* limits of the current axes.	xmin,xmax = scalars
plt.title("Plot name")	Set a title of the current plot.	**Name = string**
plt.show()	Display a figure.	