Demo 10 cheat sheet: The Wigner distribution

qworld library

Function	Purpose	Inputs
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

Miscellaneous (Libraries, arrays, plotting 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.exp(x)	Return the value of the Euler number exponentiated to x	x: exponent (number)
quad(f, a,b,args) (as imported from scipy.integrate)	Integrate a function f(x) defined as such, from a to b.	f= function ; (a,b)=integration interval, args= additional argument for f.
np.meshgrid(x,y)	Return a coordinate matrix from coordinate vectors	x: array of x values; y: array of y values
plt.imshow(M)	Produces a 2D color plot from a matrix array	M: squared matrix
np.sqrt(x)	Return the squared root of a real number x	x: real number

Demo specific common variables and functions:

x == array with position values
p == array with momentum values
t == time value
t_array == array with time values
omega = frequency of the oscillator, set to 1 by default
sigma == standard deviation for a Gaussian
sigma_array == array with different sigma values
W= Wigner distribution
deltaX and deltaP == standard deviation for position and momentum.

Function	Purpose	Inputs
def gaussian_x(x, sigma):	**Returns a gaussian in the position representation**	**x: Scalar or array of positions sigma: Standard deviation of the gaussian**
gaussian_p(p, sigma)	**Returns a gaussian in the position representation**	**x: Scalar or array of positions sigma: Standard deviation of the gaussian**
def fw_HO(x, p, w, m = 1.0, hbar = 1.0)	**Returns the wigner representation of a gaussian in a harmonic oscillator potential**	**x: scalar or matrix of positions to compute the wigner density o** **p: scalar of matrix of momenta to compute the wigner density on** **w: scalar for the frequency**
fw_HO_01(x, p, t, w, m = 1.0, hbar = 1.0)	**Returns the wigner representation of a superposition of the states n=0 and n=1 for the harmonic oscillator**	**x: scalar or matrix of positions to compute the wigner density on** **p: scalar of matrix of momenta to compute the wigner density on** **w: scalar for the frequency** **t: scalar for the time**
def x_exp(x, sigma) **def p_exp(p, sigma)**	**Returns integrand for expectation value of x and p, respectively.**	**x (p): Scalar or array of positions (momentums) ; sigma: Standard deviation of the gaussian**
def x2_exp(x, sigma) **def p2_exp(p, sigma)**	**Returns integrand for expectation value of x squared and p squared, respectively**	**x (p): Scalar or array of positions (momentums) ; sigma: Standard deviation of the gaussian**