Project 2 Documentation

Emiliano Staffoli, Alexander Ferraro, Matteo Zortea

Jun 01, 2021

CONTENTS:

class do.**GHF**(*l=10*, *grid_length=10*, *num_grid_points=201*, *alpha=1.0*, *a=0.25*, *Omega=0.25*, *omega=2*, *epsilon0=1*, *nparticles=2*, *antisymmetrize=True*)

Initializes a system in the general spin representation. The default values in the creator are referred to the main analyzed case. A more detailed description is provided in the paper folder at https://github.com/Matteo294/FYS4411. The quantum-system documentation appears in https://schoyen.github.io/quantum-systems/

Args:

l (int) : number of harmonic oscillator eigenstates used for the expansion of the radial part of the single-particle wavefunctions

grid_length (int) : extension of the mesh goes from -grid_length to grid_length

num_grid_points (int) : number of points in the mesh

alpha (float) : strength parameter in the shielded Coulomb potential

a (float) : shielding parameter in the shielded Coulomb potential

Omega (float) : frequency of the harmonic oscillator potential in which the electrons are trapped

omega (float) : frequency of the laser source

epsilon0 (float) : amplitude of the sinusoidal potential associated to the laser

nparticles (int) : number of electrons in the system

antisymmetrize (bool) : if True, antisymmetrizes the system.u matrix

Attributes:

potential (func) : ODQD.HOPotential(Omega)

system : GeneralOrbitalSystem object (see https://schoyen.github.io/quantum-systems/)

Omega (float) : frequency of the harmonic oscillator potential in which the electrons are trapped

omega (float) : frequency of the laser source

epsilon0 (float) : amplitude of the sinusoidal potential associated to the laser

nparticles (int) : number of electrons in the system

anima(*i*, *text*, *line1*, *line2*, *integrator*, *dt*, *t_max*, *save_every_n*)

Iterative function needed for self.gif_generator

eval_dipole(*C*)

Returns the expected value of the position operator for the system, given by

$\overline{x}(t) = \sum_i^{occ} \sum_{\alpha\beta} C^*_{\alpha,i}(t) C_{\beta,i}(t) \langle \chi_\alpha(x) | \hat{x} | \chi_\beta(x) \rangle$

Args: C (np.ndarray) : coefficient matrix

Returns: dipole (complex) : expected value for position operator

eval_one_body_density(*C*)

Returns the one-body density of the system, according to

$\rho(x) = \sum_i^{occ} \sum_{\alpha\beta} C^*_{\alpha,i} C_{\beta,i} \chi^*_\alpha(x) \chi_\beta(x)$

Args: C (np.ndarray) : coefficient matrix

Returns: one_body_density (np.array)

eval_total_energy(*C*)

Returns the total energy of the system, given by

$E[H] = \sum_{i=1}^{2} \sum_{\alpha,\beta} C^*_{\alpha,i} C_{\beta,i} h^{ho}_{\alpha\beta} + \frac{1}{2} \sum_{i,j=1}^{2} \sum_{\alpha\beta\gamma\delta} C^*_{\alpha,i} C^*_{\gamma,j} C_{\beta,i} C_{\delta,j} u^{\alpha\gamma}_{\beta\delta,AS}$

Args: C (np.ndarray) : coefficient matrix

Returns: energy (complex)

fill_density_matrix(*C*)

Returns the density matrix evaluated using the coefficient matrix C, according to

$M_{\alpha\beta} = \sum_i^{occ} C^*_{\alpha,i} C_{\beta,i}$

Args: C (np.ndarray) : coefficient matrix

Returns: density_matrix (np.ndarray)

fill_fock_matrix(*C*, *t*)

Returns the Fock matrix evaluated using the coefficient matrix C evaluated at time t, according to

$f_{\mu\nu}(t) = h^{ho}_{\mu\nu} + x_{\mu\nu}\varepsilon_0 \sin(\omega t) + \sum_j^{occ} \sum_{\gamma\delta} C^*_{\gamma,j} C_{\delta,j} u^{\mu\gamma}_{\nu\delta,AS}$

Args: C (np.ndarray) : coefficient matrix

t (float) : time instant

Returns: fock_matrix (np.ndarray)

fourier_analysis(*tolerance*, *max_iter*, *t_laser_ON*, *t_max*, *dt*, *eval_energy=False*)

Solves the time-independent Ruthaan-Hall equations and then performs a time-evolution of the system switching off the laser source at a certain time. Performs the Fourier analysis on the curves for overlap and dipole moment obtained for t>t_laser_ON.

Args: tolerance (float) : stopping condition for the time-independent solver.

max_iter (int) : maximum number of iterations before the time-independent solver stops.

t_laser_ON (float) : instant at which the laser is switched off

t_max (float) : final instant for time evolution

dt (float) : time step

eval_energy (bool) : if True, evaluates energy at every time step (for faster computations, set to False)

Returns: C2 (np.ndarray) : coefficient matrix for t=t_max

time (np.array) : time instants from tstart to t_max spaced by dt

dipole (np.array) : dipole evaluated at each time instant

overlap (np.array) : overlap evaluated at each time instant

dipoleFFT (np.array) : fft of dipole values for t>t_laser_ON

dipolefreqFFT (np.array) : array of frequencies corresponding to dipoleFFT

overlapFFT (np.array) : fft of overlap values for t>t_laser_ON

overlapfreqFFT (np.array) : array of frequencies corresponding to overlapFFT

energy (np.array) : total energy at each time instant

gif_generator(*dt*, *t_max*, *C0*, *save_every_n=500*)

Generates an animated image (.gif) with the time evolution of the one-body density and the time evolution of the one body potential.

Args: dt (float) : time step

t_max (float) : total duration of the time evolution

C0 (np.ndarray) : coefficient matrix at time t=0

save_every_n (int) : number of time steps between two successive frames in the final .gif file

laser_potential(*t*)

Returns the value of the laser potential at time t.

Args: t (float) : time instant

Returns: epsilon0 * sin(omega * t) (float)

plot_AO()

Plots the square modulus of the Atomic Orbitals (harmonic oscillator eigenstates). The zero-level for each single particle state is its corresponding eigenvalue.

plot_MO(*epsilon*, *C*)

Plots the square modulus of the Molecular Orbitals (after the basis change). The zero-level for each single particle state is its corresponding eigenvalue.

plot_fourier_analysis(*time*, *dipole*, *overlap*, *xFFT*, *xfreqFFT*, *overlapFFT*, *overlapfreqFFT*, *energy=None*)

Plots the results of the Fourier analysis.

Args: time (np.array) : array of time instants

dipole (np.array) : dipole for each time instant

overlap (np.array) : overlap for each time instant

xFFT (np.array) : fast Fourier transform of the dipole signal

xfreqFFT (np.array) : frequency spectrum of the dipole signal (useful for plotting)

overlapFFT (np.array) : fast Fourier transform of the overlap signal

overlapfreqFFT (np.array) : frequency spectrum of the overlap signal (useful for plotting)

energy (np.array) : energy for each time instant, may not be included

plot_one_body_density(*one_body_density*)

Plots the comparison between the one-body density and the result by Zanghellini et al..

Args: one_body_density (np.array) : result of self.eval_one_body_density()

plot_overlap(*time*, *overlap*)

Plots the comparison between the overlaps evaluated in the time-dependent solver and by Zanghellini et al..

Args: time (np.array) : array of time instants

overlap (np.array) : overlap for each time instant

rhsf(*t*, *C*)

Returns the right-hand side of the Ruthaan-Hall equations with the laser source on, corresponding to

$\dot{C}(t) = -if(t)C(t)$

The output is reshaped into an array in order to be used into self.solve_TDHF()

Args: t (float) : time

C (np.array) : reshaped coefficient matrix into array

Returns: rhs (np.array)

rhsf_OFF(*t*, *C*)

Returns the right-hand side of the Ruthaan-Hall equations with the laser source off, given by

$\dot{C}(t) = -if(t)C(t)$

The output is reshaped into an array in order to be used into self.solve_TDHF()

Args: t (float) : time

C (np.array) : coefficient matrix reshaped into array

Returns: rhs (np.array)

solve_TDHF(*tstart*, *dt*, *t_max*, *C0*, *eval_overlap=False*, *eval_dipole=False*, *eval_energy=False*, *laser_ON=True*)

Solves iteratively the time-dependent Ruthaan-Hall equations for the system.

Args: t_start (float) : time at which the evolution begins

dt (float) : time step

t_max (float) : time at which the evolution ends

C0 (np.ndarray) : coefficient matrix at time t_start

eval_overlap (bool) : if True, the overlap with the wavefunction at t=t_start is evaluated at every instant

eval_dipole (bool) : if True, the expected value for the dipole operator is evaluated at every instant

eval_energy (bool) : if True, the total energy is evaluated at every instant

laser_ON (bool) : if True, the laser source is on

Return: C (np.ndarray) : coefficient matrix at t=t_max

time (np.array) : time instants from tstart to t_max spaced by dt

overlap (np.array) : None if eval_overlap==False

dipole (np.array) : None if eval_dipole==False

energy (np.array) : None if eval_energy==False

solve_TIHF(*tolerance*, *max_iter*, *print_ON=False*, *eval_energy_per_step=False*, *eval_delta_per_step=False*)

Solves the Ruthaan-Hall equations for the system.

Args: tolerance (float) : stopping condition for the algorithm.

max_iter (int) : maximum number of iterations before the algorithm stops

print_ON (bool) : if True prints if the convergence or the max number of iterations has been reached.

eval_energy_per_step (bool) : if True, the energy of the system is evaluated at every step

eval_delta_per_step (bool) : if True, the Δ parameter of the system is evaluated at every step

Returns: epsilon (np.array) : final eigenvalues

C (np.ndarray) : final coefficient matrix

energy_per_step (np.array) : None if energy_per_step==False

delta_per_step (np.array) : None if delta_per_step==False

class do.**RHF**(*l=10*, *grid_length=10*, *num_grid_points=201*, *alpha=1.0*, *a=0.25*, *Omega=0.25*, *omega=2*, *epsilon0=1*, *nparticles=2*, *potential=None*, *antisymmetrize=False*)

Initializes a system in the restricted spin representation. The default values in the creator are referred to the analyzed case. A more detailed description is provided in the paper folder at https://github.com/Matteo294/FYS4411 . The quantum-system documentation appears in https://schoyen.github.io/quantum-systems/

Args: l (int) : number of harmonic oscillator eigenstates used for the expansion of the radial part of the single-particle wavefunctions

grid_length (int) : extension of the mesh goes from -grid_length to grid_length

num_grid_points (int) : number of points in the mesh

alpha (float) : strength parameter in the shielded Coulomb potential

a (float) : shielding parameter in the shielded Coulomb potential

Omega (float) : frequency of the harmonic oscillator potential in which the electrons are trapped

omega (float) : frequency of the laser source

epsilon0 (float) : amplitude of the sinusoidal potential associated to the laser

nparticles (int) : number of electrons in the system

antisimmetrize (bool) : MUST remain False

Attributes: potential (func) : ODQD.HOPotential(Omega)

system : ODQD object (see https://schoyen.github.io/quantum-systems/)

Omega (float) : frequency of the harmonic oscillator potential in which the electrons are trapped

omega (float) : frequency of the laser source

epsilon0 (float) : amplitude of the sinusoidal potential associated to the laser

nparticles (int) : number of electrons in the system

eval_one_body_density(*C*)

Returns the one-body density of the system, according to

$\rho(x) = 2\sum_{i}^{n/2}\sum_{\alpha\beta} C^*_{\alpha,i}C_{\beta,i}\chi^*_\alpha(x)\chi_\beta(x)$

Args: C (np.ndarray) : coefficient matrix

Returns: one_body_density (np.array)

eval_total_energy(*C*)

Returns the total energy of the system, given by

$E[H] = 2\sum_{i=1}^{n/2}\sum_{\alpha,\beta} C^*_{\alpha,i}C_{\beta,i}h^{ho}_{\alpha\beta} + 2\sum_{i,j=1}^{n/2}\sum_{\alpha\beta\gamma\delta} C^*_{\alpha,i}C^*_{\gamma,j}C_{\beta,i}C_{\delta,j}u^{\alpha\gamma}_{\beta\delta} - \sum_{i,j=1}^{n/2}\sum_{\alpha\beta\gamma\delta} C^*_{\alpha,i}C^*_{\gamma,j}C_{\beta,i}C_{\delta,j}u^{\alpha\gamma}_{\delta\beta}$

Args: C (np.ndarray) : coefficient matrix

Returns: energy (complex)

fill_density_matrix(*C*)

Returns the density matrix evaluated using the coefficient matrix C, according to

$M_{\alpha\beta} = \sum_{i}^{nparticles/2} C^*_{\alpha,i}C_{\beta,i}$

Args: C (np.ndarray) : coefficient matrix

Returns: density_matrix (np.ndarray)

fill_fock_matrix(*C*, *t*)

Returns the Fock matrix evaluated using the coefficient matrix C evaluated at time t, according to

$f_{\mu\nu} = h^{ho}_{\mu\nu} + 2\sum_{j}^{n/2}\sum_{\gamma\delta} C^*_{\gamma,j}C_{\delta,j}u^{\mu\gamma}_{\nu\delta} - \sum_{j}^{n/2}\sum_{\gamma\delta} C^*_{\gamma,j}C_{\delta,j}u^{\mu\gamma}_{\delta\nu}$

Args: C (np.ndarray) : coefficient matrix

t (float) : time instant

Returns: fock_matrix (np.ndarray)

CHAPTER

ONE

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

d

INDEX