Numerical solution of the Schrödinger equation for radiation-matter interaction models using the Runge-Kutta method implemented in Python

L. Hernández-Sánchez[a,c], A. Flores-Rosas[b], I. Ramos-Prieto[a], F. Soto-Eguibar[a], and H. M. Moya-Cessa[a]

[a]*Instituto Nacional de Astrofísica, Óptica y Electrónica,*
Calle Luis Enrique Erro No. 1 Santa María Tonantzintla, Puebla, 72840, México.
[b]*Facultad de Ciencias en Física y Matemáticas, Universidad Autónoma de Chiapas,*
Carretera Emiliano Zapata, Km. 8, Rancho San Francisco, 29050 Tuxtla Gutiérrez, Chiapas, México.
[c]*Centro de Estudios Tecnológicos Industrial y de Servicios No. 136,*
Carretera Comalapa-Cd. Cuauhtemoc, Km. 3.5 El Anonal, 30146 Frontera Comalapa, Chiapas, México.

Received 24 June 2025; accepted 25 August 2025

Obtaining exact solutions to the time-dependent Schrödinger equation in complex quantum systems presents significant challenges. In this context, numerical methods offer powerful alternatives for exploring their dynamics. In this pedagogical article, we present a numerical approach based on the fourth-order Runge-Kutta method, implemented in Python, to simulate radiation-matter interaction models. The methodology is illustrated using the well-known Jaynes-Cummings model, and the numerical results are compared with its exact analytical solution for illustrative purposes. Although only this model is explicitly solved, the numerical framework is general and can be readily extended to more complex Hamiltonians, including time-dependent cases. This makes the approach a practical and accessible tool for exploring a wide range of quantum systems.

Keywords: Schrödinger equation, radiation-matter interaction, Runge-Kutta method, Jaynes-Cummings model, Python.

DOI: https://doi.org/10.31349/RevMexFisE.23.020202

1. Introduction

The study of radiation-matter interactions is a fundamental aspect of quantum physics, with key applications in quantum optics, spectroscopy, and other areas [1-3]. A prominent example of such interactions is the Jaynes-Cummings model, which describes the coupling between a two-level atom and a quantized electromagnetic field inside a single-mode cavity. This model is particularly valuable, as it provides exact analytical solutions under specific approximations, such as the dipole approximation and the rotating wave approximation, thus facilitating the analysis of specific interactions under well-defined conditions [4-6].

In quantum optics, the study of radiation-matter interaction has led to more elaborate generalizations of the Jaynes-Cummings model, involving multilevel atoms [7,8] multiple electromagnetic field modes [9,10], and interactions with external driving fields [11-13], among others [14,15]. However, the complexity of these models makes it increasingly difficult to obtain exact solutions to the Schrödinger equation. Although certain approximations or transformations can simplify the Hamiltonians that describe these systems, they do not always provide a complete description [1,16]. Therefore, numerical methods become essential tools for analyzing such advanced quantum phenomena [17-20].

The fourth-order Runge-Kutta method (RK4) is a widely used numerical technique to solve differential equations, including the time-dependent Schrödinger equation. Its accuracy and versatility make it ideal for simulating the time evolution of quantum systems with nontrivial interactions [17,18]. In addition, it can be conveniently implemented in programming environments such as Python, offering a robust and efficient computational framework to simulate the dynamics of quantum systems. It is important to note that these properties are intrinsic to the RK4 method itself, while Python simply provides an accessible and flexible platform for its application [19].

This article presents a methodology based on RK4, implemented in Python, to solve the time-dependent Schrödinger equation in the context of radiation-matter interaction systems. The goal is to provide a numerical solution that overcomes the limitations of traditional analytical approaches, enabling the study of complex quantum phenomena. The structure of the article is as follows: in Sec. 2, we introduce and discuss in detail the radiation-matter interaction model, focusing on the Jaynes-Cummings Hamiltonian. Section 3 presents the step-by-step methodology used to solve the corresponding Schrödinger equation. In Sec. 4, we compare and analyze both the analytical and numerical results obtained using this approach. Finally, Sec. 5 summarizes the main findings of this work.

2. Description of a quantum system of radiation and matter in interaction

The Jaynes-Cummings model is fundamental for the analysis of quantum systems of radiation and matter in interaction [4]. This model describes the interaction between a two-level atom and a quantized electromagnetic field in a single-mode cavity, as illustrated in Fig. 1. Its main advantage is that

it allows for exact analytical solutions under certain approximations, making it a solid basis for studying more complex interactions [5,6].

Based on the dipole approximation and the rotating wave approximation, the Hamiltonian describing this system is expressed as [1,4]:

$$\hat{H} = \hbar\omega_c \hat{a}^\dagger \hat{a} + \frac{1}{2}\hbar\omega_{eg}\hat{\sigma}_z + \hbar g(\hat{\sigma}_+ \hat{a} + \hat{\sigma}_- \hat{a}^\dagger), \qquad (1)$$

where $\hbar$ is the reduced Planck constant. The terms that comprise this Hamiltonian are physically interpreted as follows:

- **Energy of the electromagnetic field**: The first term, $\hbar\omega_c \hat{a}^\dagger \hat{a}$, represents the energy of the electromagnetic field within the cavity. In this expression, $\hat{a}$ and $\hat{a}^\dagger$ are the annihilation and creation operators, respectively, and ω_c is the frequency of the field.
- **Energy of the two-level atom**: The second term, $\frac{1}{2}\hbar\omega_{eg}\hat{\sigma}_z$, corresponds to the energy of the atom, where $\hat{\sigma}_z$ is the Pauli operator and ω_{eg} is the transition frequency between the two energy levels.
- **Atom-field interaction**: The third term, $\hbar g(\hat{\sigma}_+ \hat{a} + \hat{\sigma}_- \hat{a}^\dagger)$, describes the interaction between the electromagnetic field and the atom. In this expression, g is the coupling constant that quantifies the strength of the interaction. This term has two components:
  - **Photon absorption**: The part $\hat{\sigma}_+ \hat{a}$ describes the process in which the atom absorbs a photon and transitions from its ground state $|g\rangle$ to the excited state $|e\rangle$.
  - **Photon emission**: The part $\hat{\sigma}_- \hat{a}^\dagger$ corresponds to the emission process, where the atom transitions from the excited state $|e\rangle$ to the ground state $|g\rangle$, emitting a photon during the process.

The temporal evolution of the system is described by the Schrödinger equation

$$\mathrm{i}\,\hbar \frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle, \qquad (2)$$

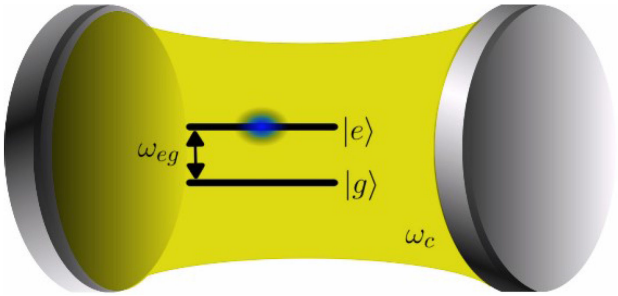

FIGURE 1. Representation of the Jaynes-Cummings model: a system composed of a two-level atom, where the excited state is denoted as $|e\rangle$ and the ground state as $|g\rangle$. This atom is confined within an optical cavity, which is made of perfectly reflective mirrors.

where $|\psi(t)\rangle$ represents the quantum state of the system at time t. The solution to this equation provides a complete description of the evolution of the system.

For Hamiltonians who model more complex interactions, finding exact solutions can be extremely challenging and often impossible. Although approximations or transformations can be employed, they do not always yield a complete analysis. In such circumstances, numerical methods become essential to overcome these limitations and to obtain a more accurate understanding of the system.

3. Methodology

This section presents the fourth-order Runge-Kutta method (RK4) as an effective tool to solve the time-dependent Schrödinger equation (2) associated with the Jaynes-Cummings Hamiltonian (1). This approach is not limited to this particular model and can be extended to more complex Hamiltonians for which analytical solutions are not available.

3.1. Problem statement

As discussed previously, the time evolution of the quantum state $|\psi(t)\rangle$ is governed by the Schrödinger equation (2) associated with the Hamiltonian system (1). The primary objective of this work is to develop a methodology based on the RK4 algorithm, implemented in Python, to numerically solve this equation and obtain the wave function $|\psi(t)\rangle$ as a function of time. This methodology will subsequently be generalized for application to other Hamiltonians that describe more complex systems.

Moreover, the proposed approach enables the simulation of the dynamics of the system under various initial conditions, including the initial states of both the atom and the electromagnetic field, given by

$$|\psi(0)\rangle = |\psi(0)\rangle_{\text{atom}} \otimes |\psi(0)\rangle_{\text{field}}. \qquad (3)$$

3.2. Fourth-order Runge-Kutta method

The RK4 method was originally developed in 1895 by the physicist and mathematician Carl David Runge for the numerical solution of differential equations and was later extended in 1901 by Wilhelm Kutta to differential equation systems. This method is an iterative procedure that approximates the solution of a differential equation through a sequence of discrete steps. Unlike simpler techniques, such as the method proposed by Leonhard Euler, RK4 provides a fourth-order approximation, which means that the local truncation error per step is proportional to h^5, where h is the size of the time step [17,18].

In general, the numerical solution of the initial value problem $y' = f(x, y)$, with $y(x_0) = y_0$ and step size h, is

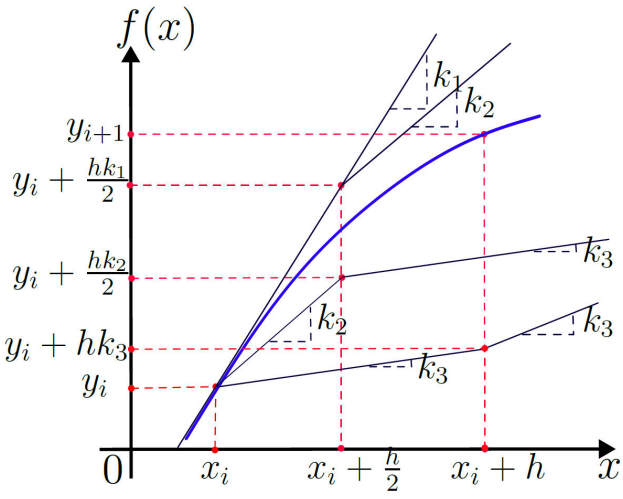

FIGURE 2. Geometric interpretation of the fourth-order Runge-Kutta method.

obtained through the points (x_{i+1}, y_{i+1}), calculated using the following recurrence formulas:

$$
\begin{aligned}
x_{i+1} &= x_i + h, \\
k_1 &= f(x_i, y_i), \\
k_2 &= f\left(x_i + \frac{h}{2}, y_i + h\frac{k_1}{2}\right), \\
k_3 &= f\left(x_i + \frac{h}{2}, y_i + h\frac{k_2}{2}\right), \\
k_4 &= f(x_i + h, y_i + hk_3), \\
y_{i+1} &= y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),
\end{aligned}
$$

where $i = 0, 1, 2, \ldots$. The sum $(k_1 + 2k_2 + 2k_3 + k_4)/6$ can be interpreted as an average slope. Note that k_1 is the slope at the beginning of the interval, k_2 is the slope at the midpoint approaching $x_i + h/2$, k_3 is a second estimate of the slope at the same midpoint, and k_4 is the slope at $x_i + h$. The geometric interpretation of this RK4 scheme can be seen in Fig. 2. In the case of the Schrödinger Eq. (2), it is clear that $x_i = t_i$, *i.e.*, time and $y_i = |\psi_i\rangle$, which correspond to the quantum state of the system.

The main advantage of RK4 is that it is easy to program, making it an attractive option for the numerical solution of differential equations. However, a significant disadvantage is the need to evaluate the function $f(x, y)$ at multiple different points in x and y during each step. This can increase computational time, especially in problems where evaluating $f(x, y)$ is costly or for systems with high dimensionality. Despite this drawback, RK4 remains widely used because of its balance between accuracy and simplicity of implementation.

3.3. Implementation in Python

Next, the code that implements RK4 to solve the Schrödinger equation (2), associated with the Hamiltonian that describes the system (1), is presented. Note that the Python interface is set to English, which restricts the use of accented characters and special symbols. For this reason, any word appearing without accents within the code is not a spelling mistake, but rather a limitation imposed by the programming environment.

1. **Importing libraries**
   In this first step, the necessary libraries for performing numerical operations, matrix manipulation, and graph generation are imported. These libraries are fundamental for defining quantum operators, solving the equations, and visualizing the simulation results.

```
import numpy as np
from scipy.special import factorial
import matplotlib.pyplot as plt
```

   - `numpy`: A widely used library in Python for creating and manipulating arrays and matrices. It is essential for defining the quantum operators of the system.
   - `scipy.special.factorial`: This function from `scipy` calculates the factorial of a number, which is key for expressing Fock states (discrete states of the quantum field) in the simulation.
   - `matplotlib.pyplot`: A library for generating plots. It will allow us to visualize the time evolution of quantum observables, such as the atomic inversion or the average number of photons in the field.

2. **Defining system constants**
   In this step, the fundamental parameters of the model are specified, representing the physical properties of the quantum system.

```
N         = 50
omega_c   = 0.4
omega_eg  = 0.9
g         = 1.0
hbar      = 1.0
t_f       = 30
dt        = 0.05
```

   - `N`: Represents the maximum number of Fock states considered in the system.
   - `omega_c`: The frequency of the electromagnetic field.
   - `omega_eg`: The transition frequency between the energy levels of the atom.
   - `g`: The coupling constant, which determines the strength of the interaction between the atom and the field.
   - `hbar`: The reduced Planck constant ($\hbar$), here set to 1 to simplify units.

- `t_f`: The final time of the simulation, that is, until when the evolution of the system will be calculated.
- `dt`: The time step used in the numerical evolution (previously denoted as h). The smaller this value is, the higher the accuracy of the simulation will be, although at the expense of increased computational cost.

3. **Definition of the operators and the Hamiltonian modeling the system**

In this step, we define the quantum operators that model the system, which consists of a single-mode electromagnetic field interacting with a two-level atom. The field is described using the annihilation and creation operators $\hat{a}$ and $\hat{a}^\dagger$, while the atom is represented by the Pauli operators $\hat{\sigma}_z$, $\hat{\sigma}_+$, and $\hat{\sigma}_-$.

The operators $\hat{a}$ and $\hat{a}^\dagger$ act on the Fock space of the electromagnetic field and satisfy the commutation relation $[\hat{a}, \hat{a}^\dagger] = 1$. Their action on the Fock states $\{|n\rangle\}$ is defined by

$$\hat{a}|n\rangle = \sqrt{n}|n-1\rangle, \quad \hat{a}^\dagger|n\rangle = \sqrt{n+1}|n+1\rangle.$$

For numerical implementation, these infinite-dimensional operators are truncated to a finite dimension N and represented as matrices. The general matrix form of $\hat{a}$ in the Fock basis is:

$$\hat{a} = \begin{pmatrix} 0 & \sqrt{1} & 0 & 0 & \cdots \\ 0 & 0 & \sqrt{2} & 0 & \cdots \\ 0 & 0 & 0 & \sqrt{3} & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

The atomic operators act on a two-dimensional Hilbert space spanned by the excited state $|e\rangle$ and ground state $|g\rangle$. Their matrix representations in this basis are:

$$\hat{\sigma}_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \hat{\sigma}_+ = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \hat{\sigma}_- = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

These operators are used in the next step to construct the Jaynes-Cummings Hamiltonian, which governs the unitary dynamics of the system.

```
def destroy(N):
    return np.diag(np.sqrt(np.arange(1, N))
        , 1)

a = destroy(N)
ad = a.T.conj()
sigma_z = np.array([[1, 0], [0, -1]])
sigma_p = np.array([[0, 1], [0, 0]])
sigma_m = np.array([[0, 0], [1, 0]])

def tensor(A, B):
    return np.kron(A, B)

def H_JCM(omega_c, omega_eg, g, hbar):
    return (hbar * omega_c * tensor(np.dot(ad
        , a), np.eye(2)) +
            hbar * (omega_eg / 2) * tensor(np
                .eye(N), sigma_z) +
            hbar * g * (tensor(a, sigma_p) +
                tensor(ad, sigma_m)))
```

The meaning of each component is as follows:

- `destroy(N)`: Creates the matrix representation of the annihilation operator $\hat{a}$ of dimension N using `np.diag`, which places square roots of integers above the main diagonal.
- `a` and `ad`: Represent the annihilation operator $\hat{a}$ and the creation operator $\hat{a}^\dagger$, respectively, where `ad = a.T.conj()`.
- `sigma_z`, `sigma_p`, `sigma_m`: Define the Pauli operators acting on the atomic two-level system.
- `tensor(A, B)`: Computes the tensor product of matrices A and B using `np.kron`, allowing the construction of operators acting on the combined Hilbert space of the atom-field system.
- `np.eye(N)`: Returns the $N \times N$ identity matrix. This is used to build operators that act only on one part (field or atom) of the composite system.
- `H_JCM(omega_c, omega_eg, g, hbar)`: Constructs the Jaynes-Cummings Hamiltonian (1), incorporating the free evolution of the field and the atom, as well as their mutual interaction.

4. **Definition of the Schrödinger equation**

In this step, the time-dependent Schrödinger equation is formulated, which describes the evolution of the quantum state of the system under the influence of the previously defined Hamiltonian. Although the current Hamiltonian is time-independent, the function `Schrodinger(t, psi, H)` includes the time parameter t explicitly to maintain consistency with the Runge-Kutta implementation described in subsec. 3.2., and to allow for a straightforward extension to time-dependent Hamiltonians.

```
def Schrodinger(t, psi, H):
    return -1j / hbar * np.dot(H, psi)
```

- `Schrodinger(t, psi, H)`: This function implements the Schrödinger equation, which relates the temporal evolution of the quantum state $|\psi\rangle$ to the Hamiltonian of the system.
- `t`: Represents the time at which the evolution of the quantum state is evaluated.
- `psi`: Denotes the state vector of the system at a given instant, which changes as the simulation progresses.

- `H`: Refers to the Hamiltonian of the system, which acts on the state $|\psi\rangle$ to determine its evolution.
- The equation is expressed in matrix form, where the Hamiltonian acts on the quantum state vector $|\psi\rangle$. The result is multiplied by $-\mathrm{i}/\hbar$ (where `1j` in Python denotes the imaginary unit i), and $\hbar$ is the reduced Planck constant. This expression yields the rate of change of the quantum state with respect to time.

5. **Implementation of RK4**
In this step, RK4 is used to numerically solve the Schrödinger equation. This method allows the calculation of the quantum state $|\psi(t)\rangle$ of the system over time by dividing the total duration into small discrete steps and approximating the solution step by step.

```
def RK4(psi_0, H, t_list):
    psi_t = np.zeros((len(t_list), len(psi_0)
        ), dtype=complex)
    psi_t[0] = psi_0

    for i, t in enumerate(t_list[:-1]):
        k1 = Schrodinger(t, psi_t[i], H)
        k2 = Schrodinger(t + dt / 2, psi_t[i
            ] + k1 * dt / 2, H)
        k3 = Schrodinger(t + dt / 2, psi_t[i
            ] + k2 * dt / 2, H)
        k4 = Schrodinger(t + dt, psi_t[i] +
            k3 * dt, H)

        psi_t[i + 1] = psi_t[i] + (dt / 6)
             * (k1 + 2 * k2 + 2 * k3 + k4)

    return psi_t
```

Next, each step of the code is explained:

- `psi_t`: This creates an array filled with zeros to store the quantum states at each time point. The array size corresponds to the number of time steps specified, and each element has the same dimension as the initial state `psi_0`. This array is of complex data type because quantum states involve complex numbers.
- `psi_t[0]`: This sets the initial quantum state `psi_0` at time zero.
- `for i, t in enumerate(t_list[:-1])`: This loop iterates over all time points in `t_list` except the last one. In each iteration, it computes the quantum state at the next time step.
- `k1`: The initial estimate at the start of the step. It evaluates the Schrödinger equation at current time `t` with the current state `psi_t[i]`.
- `k2`: The estimate at the midpoint, calculated using `k1` to update the state, advancing by half a step. It evaluates the Schrödinger equation at time `t + dt/2`.
- `k3`: Similar to `k2`, but uses `k2` for the midpoint estimate; again evaluated at time `t + dt/2`.
- `k4`: The estimate at the full step, using the `k3` value to update the state, evaluated at time `t + dt`.
- `psi_t[i + 1]`: The next quantum state is obtained by combining these four estimates according to the RK4 formula, which provides a weighted average to improve accuracy.

6. **Definition of the initial state of the atom-field system**
In this section, the initial quantum state of the system, composed of the electromagnetic field and the atom, is defined. The field state can be represented by various types of quantum states, such as Fock states, coherent states, or squeezed states, among others. For the atom, initial conditions can include the ground state, the excited state, or a superposition of both.

For this implementation, a coherent state $|\alpha\rangle$ for the field and the excited state $|e\rangle$ for the atom have been chosen as initial conditions. However, this formalism is sufficiently general to be applied to other types of states for both subsystems.

The mathematical expressions corresponding to each component of the initial state, as presented in Eq. (3), are the following

$$|\psi(0)\rangle_{\text{field}} = |\alpha\rangle = e^{-|\alpha|^2/2} \sum_{n=0}^{N-1} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \,,$$

$$|\psi(0)\rangle_{\text{atom}} = |e\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} .$$

The combined initial state of the atom-field system is then defined in the code as follows:

```
alpha = 3.0
psi_field = np.exp(-np.abs(alpha)**2 / 2) *
    np.array([alpha**n / np.sqrt(factorial(n)
    ) for n in range(N)])
psi_atom = np.array([1, 0])
psi_0 = np.kron(psi_field, psi_atom)
```

- `alpha`: Defines the parameter `alpha`, which characterizes the coherent state of the field. This parameter is related to the mean amplitude of the electromagnetic field.
- `psi_field`: Defines the coherent state of the field as a truncated superposition of Fock states, consistent with the analytical expression of $|\alpha\rangle$.
- `psi_atom`: Represents the excited atomic state $|e\rangle$.
- `psi_0`: The initial state of the full atom-field system is constructed via the tensor product (`np.kron`) between the field and atomic states.

7. **Calculation of the time evolution of the quantum state of the system and the extraction of any observable**

In this step, RK4 is used to solve the time-dependent Schrödinger equation. Using the previously defined system Hamiltonian, the time evolution of the atom-field quantum state is computed by evaluating the Schrödinger equation at a list of discrete times.

Once the state vector $|\psi(t)\rangle$ is obtained, it is possible to calculate any quantum observable of the system. In this case, the atomic inversion $W(t)$ has been chosen, which corresponds to the expectation value of the operator $\hat{\sigma}_z$ in the atomic subspace. This observable quantifies the population difference between the excited and ground states of the atom, thus providing crucial information about its dynamics during the interaction with the electromagnetic field.

In the code, this observable is calculated and stored in the array `W_t`, which numerically evaluates the expression

$$W(t) = \langle\psi(t)|\hat{\sigma}_z|\psi(t)\rangle,$$

where $\hat{\sigma}_z$ is extended to the total Hilbert space using the tensor product `tensor(np.eye(N), sigma_z)`. For each time step, this is computed as the real part of the inner product between the state vector and the operator acting on it.

```
t_list = np.arange(0, t_f, dt)
H = H_JCM(omega_c, omega_eg, g, hbar)
psi_t = RK4(psi_0, H, t_list)
W_t = np.real(np.array([np.vdot(psi, np.dot(
    tensor(np.eye(N), sigma_z), psi)) for psi
     in psi_t]))
```

- `t_list`: Defines the time array from `0` to `t_f`, with a step size `dt`. These values determine the points at which the quantum state is evaluated during the time evolution.
- `H`: Constructs the Hamiltonian of the Jaynes-Cummings model, based on the atomic and field frequencies (`omega_c`, `omega_eg`), the reduced Planck constant `hbar`, and the coupling constant `g`.
- `psi_t`: Computes the time evolution of the quantum state using the RK4 method, starting from the initial condition `psi_0` and the Hamiltonian `H`.
- `W_t`: Calculates the atomic inversion at each time step as the expectation value of the σ_z operator acting on the atomic part of the system. Since the state includes both field and atom, the operator is extended with a tensor product. For each $|\psi(t)\rangle$, the observable is applied and the inner product is computed. Only the real part is kept, as the result must be real.

3.4. Method validation

Once the state vector $|\psi(t)\rangle$ is obtained, the atomic inversion $W(t)$ has been calculated as an example of an observable. To validate that the code implemented in the methodology correctly analyzes the time evolution of the system, the atomic inversion obtained numerically via RK4 has been plotted and compared with the exact solution for the atomic inversion in the Jaynes-Cummings model under the same initial conditions. The exact solution is given by [21]:

$$W(t) = \sum_{n=0}^{\infty} \frac{P_n}{\Omega_{n+1}^2}\left[\left(\frac{\omega_c - \omega_{eg}}{2}\right)^2 + g^2(n+1)\cos\left(2\Omega_{n+1}t\right)\right], \tag{4}$$

where

$$\Omega_{n+1}^2 = \sqrt{\left(\frac{\omega_c - \omega_{eg}}{2}\right)^2 + g^2(n+1)},$$

is the Rabi frequency and $P_n = e^{-|\alpha|^2}(n|\alpha|^{2n}/n!)$ is the probability distribution associated with a coherent state. The exact expression in Python can be written as follows:

```
def DPE_EC(n, alpha):
    return np.exp(-abs(alpha)**2) * abs(alpha)
        **(2*n) / factorial(n)
def Omega_n1(n, omega_c, omega_eg, g):
    return np.sqrt(((omega_c - omega_eg) / 2)
        **2 + g**2 * (n+1))
def Parentesis_n1(t, n, omega_c, omega_eg, g):
    return ((omega_c - omega_eg) / 2)**2 + g
        **2 * (n+1) * np.cos(2 * Omega_n1(n,
        omega_c, omega_eg, g) * t)
def W_JCM(t, alpha, omega_c, omega_eg, g, N):
    return sum(DPE_EC(n, alpha) *
    Parentesis_n1(t, n, omega_c, omega_eg, g) / (
        Omega_n1(n, omega_c, omega_eg, g)**2) for
        n in range(0, N))
```

- `DPE_EC(n, alpha)`: This function calculates the probability distribution P_n.
- `Omega_n1(n, omega_c, omega_eg, g)`: Computes the Rabi frequency Ω_{n+1}, which depends on the field frequency `omega_c`, the atom frequency `omega_eg`, and the coupling constant `g`.
- `Parentesis_n1(t, n, omega_c, omega_eg, g)`: Evaluate the term in parentheses in the formula for $W(t)$.
- `W_JCM(t, alpha, omega_c, omega_eg, g, N)`: Finally, this function performs the summation to calculate the atomic inversion $W(t)$.

Next, we illustrate how to plot both the atomic inversion computed numerically using the Runge-Kutta method (solid line) and the exact solution (stars):

```
step = 10
t_exact_list = t_list[::step]
W_exact = np.array([W_JCM(t, alpha, omega_c,
    omega_eg, g, N) for t in t_exact_list])


plt.plot(t_list, W_t, label="RK4", color="blue",
    linestyle="-")
plt.plot(t_exact_list, W_exact, label="Exact",
    color="red", marker="*", linestyle="")

plt.xlabel(r'Time ($gt$)', fontsize=14)
plt.ylabel(r'$W(t)$', fontsize=14)
plt.title('RK4 vs Exact', fontsize=16)
plt.grid(True)
plt.legend()

plt.ylim(-1, 1)
plt.yticks(np.arange(-1, 1.5, 0.5))
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.savefig('atomic_inversion.pdf', format='pdf')
plt.show()
```

In the previous code:

- `step`: A step size of 10 is defined to space out the stars in the exact solution plot more widely. In this way, fewer points are selected from the list of times `t_list`, making the stars appear more separated visually.
- `t_exact_list`: A new list of times `t_exact_list` is generated, containing only every tenth element from the original `t_list`, to spread the markers in the exact solution plot.
- `W_exact`: Calculate the exact atomic inversion using the previously defined function `W_JCM`, for each time in the list `t_exact_list`.
- `plt.plot()`: Both results are plotted. The numerical calculation (`W_numeric`) is shown as a continuous blue line, while the exact solution (`W_exact`) is represented with red stars to mark the discrete values.
- `plt.xlabel`, `plt.ylabel`, `plt.title`: Set the labels and title of the plot, with font size adjustments.
- `plt.grid(True)`: Activates a grid on the plot to make it easier to visualize the points and lines.
- `plt.legend()`: Displays the legend to distinguish between the numerical results (RK4) and the exact solution.
- `plt.ylim(-1, 1)`: Sets the y-axis limits between -1 and 1 so that the values of the atomic inversion fit within this range.
- `plt.yticks(np.arange(-1, 1.5, 0.5))`: Defines the tick labels of the y-axis with intervals of 0.5 from -1 to 1, for improved readability.
- `plt.xticks`, `plt.yticks`: Adjust the font sizes of the x and y axis labels for better visibility.
- `plt.savefig()`: Saves the plot in any format, in this case as a PDF file, allowing for high-quality inclusion in documents or presentations.

The resulting plot looks like this (see Fig. 3).

4. Results and discussion

The results obtained through the numerical implementation of RK4 to solve the time-dependent Schrödinger equation exhibit close agreement with the exact solution of the Jaynes-Cummings model. The comparative plot of the atomic inversion (Fig. 3) shows that the numerical approach reproduces the characteristic oscillatory behavior of the atom-field system under the chosen initial conditions, as expected from the theoretical model.

While the visual comparison confirms that the method captures the correct qualitative dynamics, we emphasize that this comparison is illustrative and does not constitute a rigorous convergence test. A detailed quantitative analysis of the numerical error, including the convergence rate and its dependence on the model parameters and initial data, lies beyond the scope of this pedagogical work.

Nevertheless, the fourth-order Runge-Kutta method remains a powerful and accessible tool for simulating quantum systems, especially when exact analytical solutions are unavailable. The methodology, as presented, is not restricted to the Jaynes-Cummings model. Since the procedure is general, it can be extended to a variety of radiation-matter interaction models simply by modifying the Hamiltonian that defines the system, as discussed in step 3 of the methodology (Sec. 3).

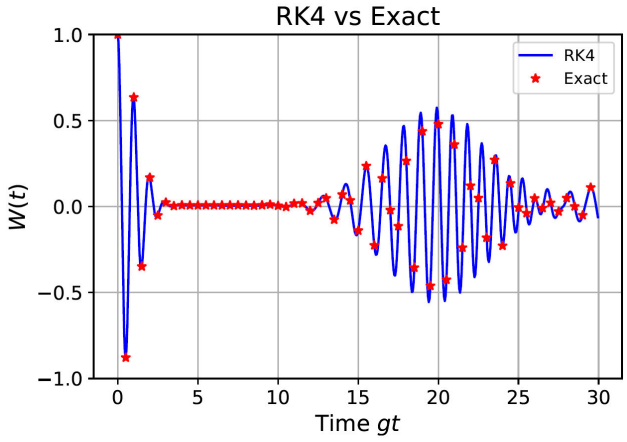

FIGURE 3. Graph of atomic inversion in the Jaynes-Cummings model. The solid blue line represents the numerical solution obtained using the Runge-Kutta method, while the red stars show the exact solution. Time is expressed in units of gt, assuming natural units with $\hbar = 1$.

5. Conclusions

This article has presented a numerical methodology based on the fourth-order Runge-Kutta method to solve the time-dependent Schrödinger equation in radiation-matter interaction systems. The approach was implemented in Python and illustrated through the Jaynes-Cummings model, a paradigmatic example in quantum optics.

The results qualitatively reproduce the characteristic dynamics of the model, suggesting that the method is suitable for pedagogical purposes and initial explorations of more complex systems. Although a rigorous convergence analysis lies beyond the scope of this work, the general framework provided here can be extended to Hamiltonians without known analytical solutions.

Due to its flexibility and implementation in an open-source language, this numerical scheme constitutes a valuable tool for students and researchers interested in simulating the dynamics of quantum systems beyond exactly solvable models.

Acknowledgements

L. Hernández-Sánchez thanks the Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) for the postdoctoral fellowship granted (CVU No. 736710).

1. C. Gerry and P. Knight, Introductory Quantum Optics (Cambridge University Press, 2004), https://doi.org/10.1017/CBO9780511791239
2. A. B. Klimov and S. M. Chumakov, A Group-Theoretical Approach to Quantum Optics (John Wiley & Sons, Ltd, 2009), https://doi.org/10.1002/9783527624003.ch5
3. G. Agarwal, Quantum Optics, Quantum Optics (Cambridge University Press, 2013), https://books.google.com.mx/books?id=7KKw_XIYaioC
4. E. Jaynes and F. Cummings, Comparison of quantum and semiclassical radiation theories with application to the beam maser, *Proc. IEEE* **51** (1963) 89, https://doi.org/10.1109/PROC.1963.1664
5. B. W. Shore and P. L. Knight, The Jaynes-Cummings Model, *J. Mod. Opt.* **40** (1993) 1195, https://doi.org/10.1080/09500349314551321
6. J. Larson and T. K. Mavrogordatos, The Jaynes-Cummings Model and Its Descendants (Institute of Physics Publishing, 2022), https://api.semanticscholar.org/CorpusID:245226335
7. V. Buzek, N-level Atom Interacting with Single-mode Radiation Field: An Exactly Solvable Model with Multiphoton Transitions and Intensity-dependent Coupling, *J. Mod. Opt.* **37** (1990) 1033, https://doi.org/10.1080/09500349014551091
8. M. Nath, S. Sen, and G. Gangopadhyay, Dynamics of cascade three-level system interacting with the classical and quantized field, *Pramana-J. Phys.* **61** (2003) 1089, https://doi.org/10.1007/BF02704404
9. C. Sukumar and B. Buck, Multi-phonon generalisation of the Jaynes-Cummings model, *Phys. Lett. A* **83** (1981) 211, https://doi.org/10.1016/0375-9601(81)90825-2
10. M. S. Abdalla, M. M. A. Ahmed, and A.-S. F. Obada, Multimode and multiphoton processes in a non-linear Jaynes-Cummings model, *Phys. A: Stat. Mech. Appl.* **170** (1991) 393, https://doi.org/10.1016/0378-4371(91)90051-D
11. P. Alsing, D.-S. Guo, and H. J. Carmichael, Dynamic Stark effect for the Jaynes-Cummings system, *Phys. Rev. A* **45** (1992) 5135, https://doi.org/10.1103/PhysRevA.45.5135
12. I. A. Bocanegra-Garay *et al*., Invariant approach to the driven Jaynes-Cummings model, *SciPost Phys.* **16** (2024) 007, https://doi.org/10.21468/SciPostPhys.16.1.007
13. L. Hernández-Sánchez *et al*., Effects of Classical Drivings on the Power Broadening of Atomic Lineshapes, *J. Opt. Soc. Am. B* **41** (2024) C68,https://doi.org/10.1364/JOSAB.522587
14. L. Hernández-Sánchez *et al*., Discriminando Superposiciones de Estados Coherentes Mediante Formas de Línea, *Rev. Mex. Fis.* **70** (2024) 011302 1-8, https://doi.org/10.31349/RevMexFis.70.011302
15. L. Hernández-Sánchez *et al*., Effects of Squeezing on the Power Broadening and Shifts of Micromaser Lineshapes, *Photonics* **11** (2024), https://doi.org/10.3390/photonics11040371
16. R. Juarez Amaro, A. Zúñiga-Segundo, and H. Moya-Cessa, Several Ways to Solve the Jaynes-Cummings Model, *App. Math. Inf. Sci.* **9** (2015) 299, https://doi.org/10.12785/amis/090136
17. J. C. Butcher, Numerical Methods for Ordinary Differential Equations (John Wiley & Sons, Ltd, 2016), https://onlinelibrary.wiley.com/doi/book/10.1002/9781119121534
18. W. H. Press, B. P. Flannery, and S. A. Teukolsky, Numerical Recipes: The Art of Scientific Computing (Cambridge University Press, 1986), https://cfatab.harvard.edu/nr/
19. J. Johansson, P. Nation, and F. Nori, QuTiP 2: A Python framework for the dynamics of open quantum systems, *Computer Physics Communications* **184** (2013) 1234, https://doi.org/10.1016/j.cpc.2012.11.019
20. L. Hernandez-Sánchez, I. A. Bocanegra-Garay, A. Flores-Rosas, I. Ramos-Prieto, F. Soto-Eguibar, and H. Moya-Cessa, Numerical solution of the Lindblad master equation using the Runge-Kutta method implemented in Python, *Rev. Mex. Fis.*

E **23** (2026) 010214, https://doi.org/10.31349/RevMexFisE.23.010214.

21. L. Hernández-Sánchez *et al*., Formas de línea atómicas en modelos de tipo Jaynes-Cummings (Editorial Académica Española, 2023), https://www.eae-publishing.com/catalogue/details/es/978-620-2-15886-2/formas-de-l%C3%ADnea-at%C3%B3micas.