Eur. Phys. J. Plus (2024) 139:986
https://doi.org/10.1140/epjp/s13360-024-05747-2

THE EUROPEAN PHYSICAL JOURNAL PLUS

Regular Article

Computation of effective collision strengths for plasma applications using JAC

Aloka Kumar Sahoo[1,2,a], **Stephan Fritzsche**[2,3,4,b], **Lalita Sharma**[1,c]

[1] Department of Physics, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand 247667, India
[2] Helmholtz-Institut Jena, Fröbelstieg 3, 07743 Jena, Germany
[3] GSI Helmholtzzentrum für Schwerionenforschung, 64291 Darmstadt, Germany
[4] Theoretisch-Physikalisches Institut, Friedrich-Schiller-Universität Jena, 07743 Jena, Germany

Received: 21 August 2024 / Accepted: 14 October 2024

Abstract The electron impact excitation rate coefficients or effective collision strengths are one of the ingredients for plasma modeling or plasma diagnosis. The present work provides a computational implementation for calculating excitation rate coefficients using the Jena Atomic Calculator toolbox within the framework of the relativistic distorted wave approximation. To address the need for large-scale calculations involving complex shells and heavy atoms and ions, parallel programming is implemented to expedite the computation of cross sections, collision strengths, effective collision strengths and rate coefficients. As a case study, we have calculated collision strengths, effective collision strengths and rate coefficients for Fe XXII.

1 Introduction

Electron impact processes are among the fundamental processes of nature, holding importance in several branches of physics [1] like plasma physics, astrophysics, atmospheric physics, etc. Electron impact excitation (EIE) is of paramount interest in plasma physics for plasma modeling and diagnosis, where it plays a significant role in the level populating mechanism. The excitation rate coefficients are the key input parameters to several plasma models, making their reliable computation essential for proper plasma diagnosis. Plasma modeling requires a vast amount of excitation data, especially for atoms and ions with heavy and complex electron shells. In such applications, the number of possible excitations can easily range from hundreds to thousands, necessitating large-scale calculations. These extensive computational tasks demand specialized approaches, such as parallel computing techniques.

In the present work, we have implemented the relativistic distorted wave (RDW) approximation to calculate the collision strengths, effective collision strengths and excitation rate coefficients within the framework of the Jena Atomic Calculator (JAC) toolbox [2]. The RDW approximation has proven to be reliable for the study of electron impact excitation studies [3–6]. While this approximation does typically not account for resonance contributions due to the dielectronic capture and re-autoionization of electrons, which are significant primarily in the low-energy regions, several studies in the literature have also demonstrated the application of the RDW theory for a reliable diagnosis of plasma temperature and density [7, 8]. In addition, the JAC toolbox is developed in the present work for EIE studies such that it includes parallel computing strategies to manage extensive computations efficiently. For this purpose, we leverage the capabilities of the modern programming language Julia [9], known for its strong support for parallel computing and used in developing the JAC toolbox. Our implementation includes both multithreading and multiprocessing approaches, allowing EIE calculations to be executed efficiently across a range of computational systems, from personal computers to multi-node cluster computers. This versatility ensures that the JAC toolbox can now be utilized efficiently, providing a powerful tool for the accurate and efficient study of electron impact excitations.

A robust atomic structure calculation is fundamental for studying electron impact excitation in atoms and ions. JAC is a versatile toolbox designed to perform complex atomic structure calculations using the multiconfiguration Dirac–Hartree–Fock (MCDHF) approach with configuration interaction methods [2, 10]. Beyond these, the JAC toolbox also supports a variety of other atomic process calculations that are crucial for plasma modeling. These include electron impact ionization [11], dielectronic recombination rate coefficients [12], radiative recombination rate coefficients [13], as well as photoexcitation and photoionization processes [14, 15]. These processes are among the primary contributors to plasma behavior, and together they provide a comprehensive model for understanding plasma dynamics. In addition, the capability of the JAC toolbox to convert the level representation from the jj-coupling basis to LSJ-coupling basis can provide a unique advantage for plasma researchers working with OPEN-ADAS [16]

[a] e-mail: aloka_s@ph.iitr.ac.in (corresponding author)
[b] e-mail: s.fritzsche@gsi.de
[c] e-mail: lalita.sharma@ph.iitr.ac.in (corresponding author)

Published online: 11 November 2024

specific file formats such as ADF04 format, which makes use of the LSJ-coupling scheme. Another collisional-radiative modeling package, ColRadPy [17], also accepts ADF04-specific file formats as input. This feature allows users to easily prepare OPEN-ADAS specific file formats for further plasma modeling, streamlining the workflow, or to enhance the integration of JAC with other plasma modeling/diagnostic tools.

The paper is organized in the following sections. In Sect. 2, we briefly described the basic underlying relativistic distorted wave theory and calculation of collision strengths, effective collision strengths and rate coefficients. Section 3 details the computational implementation of the EIE module in the JAC toolbox, focusing on the calculation of effective collision strengths and rate coefficients. This section also covers our parallel computing strategies, such as multithreading and multiprocessing, to handle large-scale calculations efficiently. For the sake of completeness, in Sect. 3.3, we demonstrate a practical application of the EIE module, illustrating the calculation of collision strengths, excitation rate coefficients and effective collision strengths for Fe XXII using the JAC toolbox. Finally, a few conclusions from our work are given in Sect. 4.

2 Electron impact excitation

The JAC toolbox [2] incorporates the RDW theory to investigate electron impact excitation process. A detailed discussion on the implementation of RDW in JAC can be found in the previous work [18]. In this study, we provide a concise overview of the foundational RDW theory and the associated formulae used to compute collision strengths, effective collision strengths, and excitation rate coefficients. The equations presented in this section are in atomic units unless stated otherwise. The collision strength for the initial state $|\gamma_i J_i\rangle$ to the final state $|\gamma_f J_f\rangle$ (or equivalently $i \rightarrow f$) is given as

$$\Omega_{i\rightarrow f} = 2 \sum_{\kappa_i,\kappa_f} \sum_{J_t} [J_t] \left| \left\langle (\gamma_f J_f, \varepsilon_f \kappa_f) J_t \right| \left| V^{(e-e)} \right| \left| (\gamma_i J_i, \varepsilon_i \kappa_i) J_t \right\rangle \right|^2 \tag{1}$$

Here, $V^{(e-e)}$ represents the electron–electron Coulomb interaction operator. The states $|\gamma_i J_i; \varepsilon_i \kappa_i\rangle$ and $|\gamma_f J_f; \varepsilon_f \kappa_f\rangle$ denote the total initial and final states wavefunctions, respectively, that are taken as anti-symmetrized product states of the bound and continuum electrons wavefunctions. The notation $[J_t]$ stands for $(2J_t + 1)$, where J_t is the total angular momentum quantum number resulting from the coupling of the atomic and projectile electron angular momenta. ε and κ represent the energy and relativistic quantum number corresponding to the projectile electron. The bound state wavefunction of the target atom or ion is obtained using the MCDHF approach with configuration interaction within the JAC toolbox [2, 10]. The continuum electron wavefunctions are obtained with the relativistic partial wave expansion method [3]. The large ($P_{\varepsilon\kappa}$) and small ($Q_{\varepsilon\kappa}$) radial components of each partial wave are obtained by solving the following coupled Dirac equations using the Dirac–Fock–Slater-type distortion potential (U),

$$\left(\frac{\mathrm{d}}{\mathrm{d}r} + \frac{\kappa}{r}\right) P_{\varepsilon\kappa}(r) - \frac{1}{c}(\varepsilon - U + 2c^2) Q_{\varepsilon\kappa}(r) = 0, \tag{2}$$

$$\left(\frac{\mathrm{d}}{\mathrm{d}r} - \frac{\kappa}{r}\right) Q_{\varepsilon\kappa}(r) + \frac{1}{c}(\varepsilon - U) P_{\varepsilon\kappa}(r) = 0. \tag{3}$$

The continuum radial wavefunctions are energy normalized and the phase shift due to the distortion potential is obtained by imposing the asymptotic boundary conditions [19] as

$$P_{\varepsilon\kappa}(r) \xrightarrow{r\rightarrow\infty} \sqrt{\frac{k}{\pi\varepsilon}} \sin\left(kr - l\frac{\pi}{2} - \eta \ln(2kr) + \Delta\right), \tag{4}$$

$$Q_{\varepsilon\kappa}(r) \xrightarrow{r\rightarrow\infty} \sqrt{\frac{k}{\pi(\varepsilon + 2c^2)}} \cos\left(kr - l\frac{\pi}{2} - \eta \ln(2kr) + \Delta\right). \tag{5}$$

Here, c represents the speed of light in free space, k denotes the wavenumber, and η represents Sommerfeld's parameter. l is the orbital angular quantum number of the partial wave, and Δ represents the corresponding total phase shift, which includes both the Coulomb phase shift and short-range inner phase shift.

Once the collision strengths are obtained, these are used to calculate the effective collision strengths (Ω^{eff}) and excitation rate coefficients. At a given electron temperature (T_e), the effective (and still dimensionless) collision strength is obtained by integrating the collision strengths over a Maxwellian electron energy distribution function as

$$\Omega^{\mathrm{eff}}_{i\rightarrow f}(T_e) = \frac{1}{k_B T_e} \int_0^\infty \mathrm{d}\varepsilon_f \, \Omega_{i\rightarrow f}(\varepsilon_f) \exp\left(-\frac{\varepsilon_f}{k_B T_e}\right). \tag{6}$$

Here, k_B denotes the Boltzmann constant. Similarly, the excitation rate coefficient (α^{EIE}) is obtained by integrating the excitation cross sections (σ) over (Maxwellian) electron energy distribution function as

$$\alpha^{\mathrm{EIE}}_{i\rightarrow f}(T_e) = \frac{4}{(k_B T_e)^{3/2}\sqrt{2\pi m_e}} \int_{\Delta E}^\infty \mathrm{d}\varepsilon_i \, \varepsilon_i \, \sigma^{\mathrm{EIE}}_{i\rightarrow f}(\varepsilon_i) \exp\left(-\frac{\varepsilon_i}{k_B T_e}\right) \qquad \left[\frac{cm^3}{s}\right]. \tag{7}$$

Here, m_e represents the mass of electron. The electron temperature (T_e) is considered in Kelvin. The computational implementation of effective collision strength and excitation rate coefficient calculation within the framework of JAC toolbox are discussed in Sect. 3.

3 Computational implementation

3.1 Effective collision strength and plasma rate coefficient

The JAC toolbox has been developed as an open-source package to provide the physics community with an accessible and adaptable platform for computing atomic properties and various atomic processes. The toolbox is implemented in Julia [9], a programming language specifically designed for high-performance numerical computing. Key features of Julia that make it particularly suitable for scientific computing include multiple dispatch, which facilitates the writing of efficient and flexible code, and robust out-of-the-box support for parallel computing. These capabilities enable the efficient execution of large-scale computations, which are often required in atomic physics and plasma modeling. The source code for the JAC toolbox, including the latest extensions, is available on the GitHub repository [20]. This open-source availability encourages collaboration and further development, making it a valuable resource for researchers in the field.

JAC toolbox is developed with well-designed data structures that facilitate seamless data movement and sharing between different subprograms by using `abstract type` [21] of Julia programming language [9]. This design choice not only simplifies the program's architecture, but also enhances its readability and ease of understanding, reducing the complexity typically associated with scientific computing software. This extension to JAC also follows the same legacy of the JAC toolbox ensuring consistency and ease of use. For a detailed explanation of data structures used in the EIE study in JAC, the readers are encouraged to refer to Sections 2.4 and 2.5 of the previous report [18].

To compute excitation rate coefficients and effective collision strengths, `ImpactExcitation.Settings` is used through the keywords `calcRateCoefficient` with Boolean input `true` or `false`. When the `calcRateCoefficient` is `true`, the excitation rate coefficients and effective collision strengths are calculated for the selected lines given by the `LineSelection()` singleton data type at input. The program first computes collision strengths for the selected excitations at a given number of incident electron energies defined with `numElectronEnergies` with maximum incident electron energy equal to the excitation threshold times `maxEnergyMultiplier`. By default, the program considers six incident electron energies, with the default maximum energy set to thirty times the excitation threshold for a particular transition. After calculating the collision strengths, the program proceeds to compute the effective collision strengths and rate coefficients at the electron temperatures specified by `temperatures` keyword. These temperatures are provided as a one-dimensional `Array{Float64,1}` within the `ImpactExcitation.Settings`, as illustrated in Fig. 1.

Upon completion, the program returns a tuple if the optional `output` keyword is set to `true`. The first element of the tuple is a one-dimensional `Array` of `DataType` `ImpactExcitation.Line`. The `ImpactExcitation.Line` contains details about a particular excitation, partial waves, collision strengths, etc., as shown in the upper panel of Fig. 2. The second element of the tuple is a one-dimensional `Array` with element `DataType` of `ImpactExcitation.RateCoefficients`, which includes information on the transition, effective collision strengths, and rate coefficients corresponding to the specified temperatures. The specifics of these `DataTypes` are shown in the lower panel of Fig. 2.

3.2 Parallel computing implementation

Over the past decades, the computational technology has seen remarkable advancement from a single-core processor to tens of cores in a single processor. Modern motherboards often feature multiple processors with a shared RAM architecture, and individual nodes can be connected via network protocols to form computing clusters. This evolution allows access to computational power ranging from a single core to thousands of cores, enabling efficient computation of complex and large-scale calculations.

Julia programming language inherently supports parallel computing [21], which includes both multithreading and multiprocessing. With shared-memory parallelism through multithreading, the calculations can be distributed over all the available threads within a single node. On the other hand, multiprocessing enables parallel execution across all cores within a node and is particularly effective in multi-node configurations, such as high-performance computing clusters with distributed computing. The present implementation of electron impact excitation calculation supports both multithreading and multiprocessing with distributed computing techniques. This dual approach enables the capability to handle the requirement of large-scale calculations for accurate modeling of laboratory and astrophysical plasmas.
The multithreading or multiprocessing execution in Julia can be started as follows:

Multithreading: For performing a multithreading execution, Julia could be started with multiple threads with the command `julia -t N`, where `N` represents the number of threads. For better performance, the number of threads should be less than or equal to the total number of threads available in a computer. The total number of threads available in a system could be identifiable in a workstation (for most Linux-based systems) with the `lscpu` command.

```
"""
`struct  ImpactExcitation.Settings  <:  AbstractProcessSettings`
    ... defines a type for the details and parameters of computing electron-impact excitation lines.
  + lineSelection            ::LineSelection      ... Specifies the selected levels, if any.
  + electronEnergies         ::Array{Float64,1}   ... List of impact-energies of the incoming elecgtrons (in user-defined units).
  + energyShift              ::Float64            ... An overall energy shift for all transitions |i> --> |f>.
  + maxKappa                 ::Int64              ... Maximum kappa value of partial waves to be included.
  + calcRateCoefficient      ::Bool               ... True, if the plasma rate coefficients to be calculated, false otherwise.
  + maxEnergyMultiplier      ::Float64            ... Maximum initial electron energy for eff. collision strength integration.
                                (maxEnergyMultiplier * Excitation threshold energy), after this assymptotic limit is applied.
  + numElectronEnergies      ::Int64              ... No. of different electron energy points at which collision strengths
                                 to compute in electron energy range [0, (maxEnergyMultiplier * Excitation threshold energy)]
  + temperatures             ::Array{Float64, 1}  ... Electron temperatures [K] for eff. collision strengths and
                                                      rate coefficients calculations.
  + printBefore              ::Bool               ... True, if all energies and lines are printed before their evaluation.
  + operator                 ::AbstractEeInteraction
                               ... Interaction operator that is to be used for evaluating the e-e interaction amplitudes;
           allowed values are: CoulombInteraction(), BreitInteraction(), ...
"""
struct Settings  <:  AbstractProcessSettings
    lineSelection             ::LineSelection
    electronEnergies          ::Array{Float64,1}
    energyShift               ::Float64
    maxKappa                  ::Int64
    calcRateCoefficient       ::Bool
    maxEnergyMultiplier       ::Int64
    NoFreeElectronEnergy      ::Int64
    temperatures              ::Array{Float64, 1}
    printBefore               ::Bool
    operator                  ::AbstractEeInteraction
end
```

Fig. 1 Inputs for EIE calculations with `ImpactExcitation` module of JAC

Multiprocessing: Multiprocessing jobs offer greater scalability than multithreaded jobs, as a multiprocess job can be scaled across all nodes of a cluster computer arrangement also known as distributed computing, unlike a multithreaded job that is limited to a single computer or node. Similar to multithreaded jobs, a multiprocessing job in Julia can be initiated with the command `julia -p N`, where `N` refers to the number of processes to be started. In a cluster computing environment, the execution of multiprocessing jobs depends on the workload manager employed in the cluster. In the Julia programming language, multiprocessing job execution on clusters can be efficiently managed using the `ClusterManagers.jl` package [22], which supports a variety of workload managers. Specifically, for clusters using the SLURM workload manager, multiprocessing jobs can be handled with the `SlurmClusterManager.jl` package [23].

Figure 3 illustrates our approach to parallel computing in the `ImpactExcitation` module of the JAC toolbox. We utilize `Distributed.jl` for multiprocessing execution, and `Threads.jl`, which is included in the Julia `Base`, for multithreading execution. While several other packages and approaches are available for parallel computing in Julia, our implementation is designed to be generic and user-friendly. This simplicity ensures that the code is easy to understand and modify, making it accessible for the scientific community to adapt it to their specific needs.

As shown in Fig. 3, the program first determines the number of `ImpactExcitation.Line` to be calculated as per the input given by the user, as discussed in Sect. 3.1. After this, we decide whether to start a multiprocessing, multithreading or serial execution. The number of processes available to Julia is accessible by `Distributed.nprocs()`. If this value is greater than one, we start a multiprocessing execution. Otherwise, we start a multithreading execution, which also works for the serial execution, considering the number of threads as one. In this way, the present implementation in the JAC toolbox automatically detects whether the execution is serial, multithreading, or multiprocessing, and distributes the workload accordingly among the user-defined number of threads or processes.

The total computational time depends on factors like the total number of `ImpactExcitation.Line` to be calculated, the number of threads (for multithreading) or processes (for multiprocessing) allocated. When the number of threads/processes assigned is equal to the number of `ImpactExcitation.Line` to be calculated, the overall computation time is determined by the longest-running thread or process, which, in turn, depends on the total number of partial waves required to achieve convergence and configuration state functions used to generate the bound states of the target atom.

```
"""
`struct  ImpactExcitation.Line`
    ...defines a type for an electron-impact excitation line that may include the definition of channels and their corresponding
       amplitudes.

    + initialLevel           ::Level         ... initial- (bound-state) level
    + finalLevel             ::Level         ... final- (bound-state) level
    + initialElectronEnergy  ::Float64       ... energy of the incoming (initial-state) free-electron
    + finalElectronEnergy    ::Float64       ... energy of the outgoing (final-state) free-electron
    + crossSection           ::Float64       ... total cross section of this line
    + collisionStrength      ::Float64       ... total collision strength of this line
    + channels               ::Array{ImpactExcitation.Channel,1}  ... List of ImpactExcitation channels of this line.
    + convergence            ::Float64       ... convergence of calculation
"""
struct  Line
    initialLevel             ::Level
    finalLevel               ::Level
    initialElectronEnergy    ::Float64
    finalElectronEnergy      ::Float64
    crossSection             ::Float64
    collisionStrength        ::Float64
    channels                 ::Array{ImpactExcitation.Channel,1}
    convergence              ::Float64
end
```

```
"""
`struct  ImpactExcitation.RateCoefficients`     ... Defines a type for the output results from excitation rate or
                                                    effective collision strengths calculations
    + initialLevel        ::Level               ... initial- (bound-state) level
    + finalLevel          ::Level               ... final- (bound-state) level
    + temperatures        ::Array{Float64,1}    ... Temperatures in [K] to calculate excitation rates and
                                                    effective collision strengths
    + alphas              ::Array{Float64,1}    ... Excitation rate coefficients in [cm^3/s] for the input temperatures
    + effOmegas           ::Array{Float64,1}    ... Effective collision strengths for the input temperatures
"""
struct  RateCoefficients
    initialLevel          ::Level
    finalLevel            ::Level
    temperatures          ::Array{Float64,1}
    alphas                ::Array{Float64,1}
    effOmegas             ::Array{Float64,1}
end
```

Fig. 2 Important data structures used in the `ImpactExcitation` module of JAC

If the number of threads/processes assigned is lesser than the number of `ImpactExcitation.Line` to be calculated, then the execution time may vary significantly. Also, in this case, the multiprocessing jobs perform better than multithreading jobs as the `Distributed.pmap` (for multiprocessing) dynamically manages the workload. This means, as soon as one process completes a task, it is assigned a new task ensuring all the processes are optimally utilized. On the other hand, `Threads.@threads` (for multithreading) divides the workload statically at the beginning of the execution. As a result, if a thread finishes earlier than others, it remains idle while other threads continue working.

Therefore, the multiprocessing execution should be preferred for large-scale calculations even on a single computer, and only small-scale calculations should be performed with multithreading execution. Further, the computation time with respect to the number of required partial waves for convergence will depend on both the transition type and the projectile electron energy. More partial waves are required to obtain the converged collision strengths for dipole-allowed type of transitions and with the increasing projectile electron energy.

```
# Part of codes from the ImpactExcitation.computeLines() funtion from ImpactExcitation module

# Determines the number of ImpactExcitation.Line to be calculated
lines = ImpactExcitation.determineLines(finalMultiplet, initialMultiplet, settings)

# Display all selected lines before the computations start
if  settings.printBefore    ImpactExcitation.displayLines(lines)    end

# Calculate all amplitudes and requested properties

# for multiprocessing execution
if  Distributed.nprocs() > 1
    newLines = Distributed.pmap(line ->
            ImpactExcitation.computeAmplitudesProperties(line, nm, grid, settings), lines)

# for multithreading execution
else
    newLines = Vector{ImpactExcitation.Line}(undef, length(lines))
    Threads.@threads for l in eachindex(lines)
        newLines[l] = ImpactExcitation.computeAmplitudesProperties(lines[l], nm, grid,
                                                                   settings)
    end
end
```

Fig. 3 Parallel computing as implemented in ImpactExcitation module of JAC toolbox [2]

A sample example of the inputs for calculating collision strengths, effective collision strengths and rate coefficients with the JAC toolbox is given in Sect. 3.3 that works for serial, multithreading and multiprocessing executions.

3.3 Computation of effective collision strengths for Fe XXII

For illustration, we studied electron impact excitation of B-like iron (Fe XXII) in the present work. Fe is an abundant element in astrophysical objects detected in the Sun [24], planetary nebulae [25, 26] and interstellar medium [27]. Thus, highly charged Fe ions are important in high-temperature astrophysical plasmas. We calculated the EIE collision strengths, effective collision strengths and excitation rate coefficients for excitation from the ground state [He] $2s^2 2p\ ^2P_{1/2}$ to the excited states [He] $2s2p^2\ ^4P_{1/2}$, $^4P_{3/2}$ and $^4P_{5/2}$. The calculations could be performed in a similar way with `Atomic.Computation` as described in the previous study [18] with the updated `ImpactExcitation.Settings`. However, the current study presents more detailed step-wise calculation, shown in Fig. 4, for obtaining EIE rate coefficients and effective collision strengths.

The computation for rate coefficients starts with the atomic structure calculations. For Fe XXII, we first define a Fermi-distributed nucleus with the nuclear charge, $Z = 26$. The atomic states for which the excitations are to be studied are defined by their (non-relativistic) electronic configurations. Here, in Fig. 4, we have considered three configurations [He] $2s^2 2p$, [He] $2s2p^2$ and [He] $2p^3$. Although this study focuses on these three configurations, the JAC toolbox supports extensive calculations using a restricted active set approach, allowing for a more accurate representation of atomic wavefunctions. However, this comes with a computational cost. The atomic structure calculation helps us with the exact identification of the states both in jj-coupling and LSJ-coupling schemes. It also yields the level energies and assigns level indices in ascending order of energy. Next, for the calculation of effective collision strengths and rate coefficients, we give input of the temperatures as a one-dimensional `Array{Float64, 1}`s along with the selected transitions "(1,3), (1,4), (1,5)" that represents excitation to [He] $2s2p^2\ ^4P_{1/2}$, $^4P_{3/2}$ and $^4P_{5/2}$ states, respectively, from the ground state [He] $2s^2 2p\ ^2P_{1/2}$. The maximum number of partial waves could be provided with `maxKappa` keyword as defined in `ImpactExcitation.Settings` in Fig. 1. However, the program terminates the calculations for each `ImpactExcitation.Line` once convergence of 10^{-5} is achieved between the successive partial waves. The present calculation considers a maximum of 170 partial waves. The present calculation takes around 4.5 min with multiprocessing execution and around 6 min with multithreading execution on Intel Xeon 8268 processor with 24 cores. This includes the computational time for calculating atomic wavefunctions, as well as collision strengths and effective collision strengths. The total computational time will

```
# Calculation of effective collision strengths and excitation rate coefficients for Fe XXII
using JAC

# Atomic structure calculations
nucModel = Nuclear.Model(26., "Fermi")
grid = Radial.Grid(Radial.Grid(false), rnt = 4.0e-6, h = 2.5e-2, hp = 2.5e-2, rbox = 30.0)
asfSettings = AsfSettings(AsfSettings(); jjLS=LSjjSettings(true))
wa = Atomic.Computation(Atomic.Computation(),name="Atomic structure calculation - Fe XXII",
                        grid=grid, nuclearModel=nucModel,
                        configs=[Configuration("[He] 2s^2 2p"),Configuration("[He] 2s 2p^2"),
                        Configuration("[He] 2p^3")], asfSettings=asfSettings )
wb = perform(wa; output = true)     ;    multiplet = wb["multiplet:"]    ;

# Effective collision strength and excitation rate coefficient calculations
temps = [1e5, 2e5, 5e5, 7e5, 1e6, 2e6, 5e6, 7e6, 1e7, 2e7]      # temeratures are in Kelvin
settings     = ImpactExcitation.Settings(ImpactExcitation.Settings();
                lineSelection = LineSelection(true, indexPairs=[(1,3),(1,4),(1,5)]),
                calcRateCoefficient = true, maxEnergyMultiplier=40., temperatures = temps)

lines, rates = ImpactExcitation.computeLines(multiplet, multiplet, nucModel, grid, settings;
                                                                        output=true);
```

Fig. 4 Inputs to calculate effective collision strengths and rate coefficients for Fe XXII with the JAC toolbox

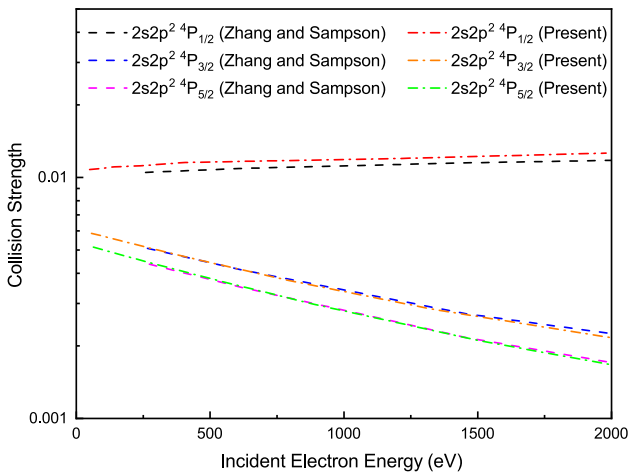

Fig. 5 Collision strength for electron impact excitation from the ground state to the excited states $2s2p^2\ ^4P_{1/2}$, $^4P_{3/2}$, $^4P_{5/2}$. Dash-dot line: present calculation, and dashed line: Zhang and Sampson [28]

remain nearly the same if additional excitations are included, provided that the number of allocated threads or processes is increased proportionally, as discussed in Sect. 3.2.

After the completion of the calculation, the calculated collision strengths, cross sections, effective collision strengths and rate coefficients are printed on the screen in a tabular format for the selected transitions. Here, our calculated collision strengths for Fe XXII are shown in Fig. 5 along with the comparison against the RDW calculations by Zhang and Sampson [28]. We found an excellent agreement between our calculated results and the previous RDW calculations. The marginal difference in collision strength for excitation to [He] $2s2p^2\ ^4P_{1/2}$ could be due to the different representations of atomic bound states and our implemented numerical procedure to obtain the collision strengths.

Additionally, we presented the effective collision strengths and rate coefficients for the aforementioned transitions in Figs. 6 and 7, respectively. The EIE collision strengths show a good agreement with the calculated results by Zhang and Sampson [28], reported

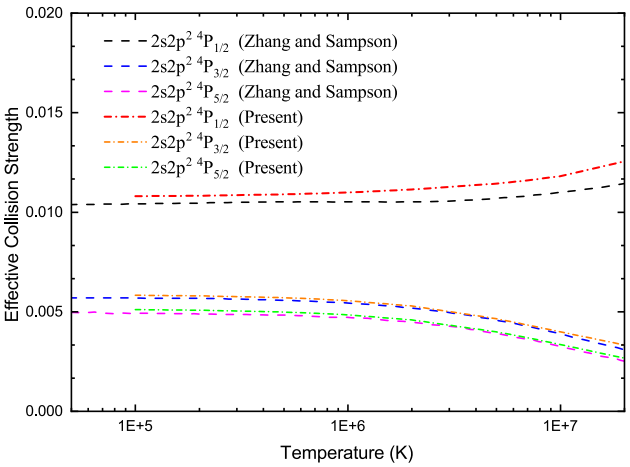

Fig. 6 Effective collision strength for electron impact excitation from the ground state to the excited states $2s2p^2\ ^4P_{1/2}$, $^4P_{3/2}$, $^4P_{5/2}$. Dash-dot line: present calculation, and dashed line: RDW calculation of Zhang and Sampson [28]

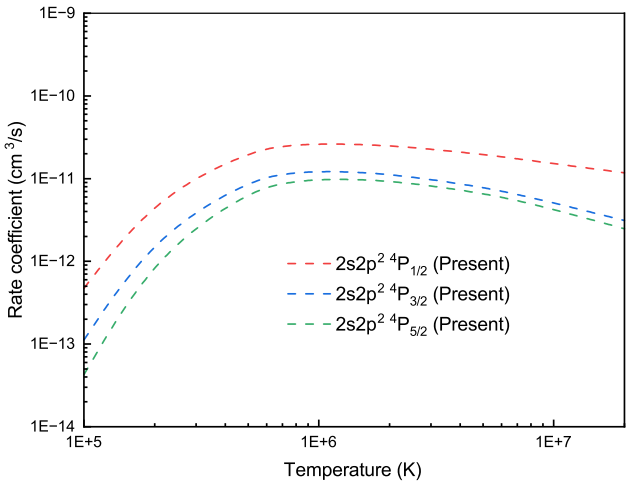

Fig. 7 Excitation rate coefficients for electron impact excitation from the ground state to the excited states $2s2p^2\ ^4P_{1/2}$, $^4P_{3/2}$, $^4P_{5/2}$

in the work of Zhang and Pradhan [29]. The small deviation observed for the excitation to [He] $2s2p^2\ ^4P_{1/2}$ state primarily arises from the difference in the collision strength as depicted in Fig. 5.

4 Conclusion

In this study, we have expanded the capabilities of the JAC toolbox to calculate effective collision strengths and EIE rate coefficients using the relativistic distorted wave theory. Additionally, the inclusion of multithreading and multiprocessing execution allows large-scale calculations of collision strengths, effective collision strengths and rate coefficients efficiently, which are crucial for modeling various laboratory and astrophysical plasmas. The enhancements made to the input parameters make the program more user-friendly, offering greater flexibility and control over the EIE calculations. As an example, the EIE of Fe XXII is considered to showcase the reliability of the current EIE developments within the JAC toolbox. Our calculated collision strengths and effective collision strengths exhibit a good agreement with the RDW results in literature, confirming the accuracy and robustness of our approach.

The current implementation considers a Maxwellian electron energy distribution for effective collision strength and rate coefficient calculations. However, we hope that the open-source nature of the JAC toolbox [20], combined with its development in the intuitive and high-level Julia programming language, ensures that researchers in the atomic and plasma physics communities can easily modify and extend it to accommodate other types of electron energy distributions.

Over the past years, the JAC toolbox has been developed to carry our cascade calculations [30, 31] with a particular interest in astro- and plasma physics. Looking ahead, there is a potential to expand cascade computations to include EIE along with dielectronic recombination and re-autoionizations. This expansion would allow for the inclusion of resonance effects, thereby further enhancing the accuracy of atomic data crucial for both laboratory and astrophysical plasma modeling.

Acknowledgements AKS acknowledges the Ministry of Education, India, for providing a PhD fellowship. AKS also like to acknowledge GSI GmbH, FAIR GmbH and Helmholtz Institute Jena for providing a scholarship under GET Involved 2023 (GI-231748 S-IN-SDN) program to support his research

visit to Jena, Germany. Additionally, AKS would like to thank Friedrich-Schiller-Universität Jena, Germany, for providing access to computational facilities during his stay in Jena. L S thanks IAEA, Vienna, for sanctioning the IAEA Research Contract No. 26504.

Author contributions AKS and LS conceived the present idea for this extension to the JAC toolbox. AKS designed the computational implementation of the present extension and carried out the relevant tests with support from SF. AKS and LS drafted the manuscript. SF reviewed the computational implementation and manuscript and provided critical feedback. All authors discussed the results and contributed to the final manuscript.

Funding Not applicable.

Data availability This manuscript has no associated data or the data will not be deposited. All data in figures can be reproduced using JAC toolbox.

Materials availability Not applicable.

Code availability The code is available on GitHub https://github.com/OpenJAC/JAC.jl along with the present extension.

Declarations

Conflict of interest The authors declare no conflict of interest.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. A. Müller, Electron-ion collisions: fundamental processes in the focus of applied research. Adv. Atomic Mol. Opt. Phys. **55**, 293–417 (2008). https://doi.org/10.1016/S1049-250X(07)55006-8
2. S. Fritzsche, A fresh computational approach to atomic structures, processes and cascades. Comput. Phys. Commun. **240**, 1–14 (2019). https://doi.org/10.1016/j.cpc.2019.01.012
3. L. Sharma, A. Surzhykov, R. Srivastava, S. Fritzsche, Electron-impact excitation of singly charged metal ions. Phys. Rev. A **83**, 062701 (2011). https://doi.org/10.1103/PhysRevA.83.062701
4. V. Roman, A.I. Gomonai, L. Sharma, A.K. Sahoo, A.N. Gomonai, Electron impact excitation of the Tl^+ ion: resonance and cascade transitions. J. Phys. B: Atomic Mol. Opt. Phys. (2022). https://doi.org/10.1088/1361-6455/ac7924
5. A. Gomonai, V. Roman, A. Gomonai, A.K. Sahoo, L. Sharma, Electron-impact excitation of the λ190.8 nm and λ179.9 nm Intercombination Lines in the Tl^+ Ion. Atoms (2022). https://doi.org/10.3390/atoms10040136
6. A.K. Sahoo, L. Sharma, Electron impact excitation of bismuth. Phys. Scr. **99**(9), 095410 (2024). https://doi.org/10.1088/1402-4896/ad6f51
7. S.S. Baghel, S. Gupta, R.K. Gangwar, R. Srivastava, Diagnostics of low-temperature neon plasma through a fine-structure resolved collisional-radiative model. Plasma Sour. Sci. Technol. **28**(11), 115010 (2019). https://doi.org/10.1088/1361-6595/ab4684
8. A. Agrawal, S. Gupta, L. Sharma, R. Srivastava, Spectroscopic study of Kr^+ plasma through a detailed collisional radiative plasma model with extended ground, metastable and quasi-metastable electron impact excitation cross-section calculations. Spectrochim. Acta, Part B **217**, 106953 (2024). https://doi.org/10.1016/j.sab.2024.106953
9. J. Bezanson, A. Edelman, S. Karpinski, V.B. Shah, Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671
10. S. Fritzsche, Level structure and properties of open f-shell elements. Atoms (2022). https://doi.org/10.3390/atoms10010007
11. S. Fritzsche, L. Jiao, G. Visentin, Rapid access to empirical impact ionization cross sections for atoms and ions across the periodic table. Plasma **7**(1), 106–120 (2024). https://doi.org/10.3390/plasma7010008
12. S. Fritzsche, Dielectronic recombination strengths and plasma rate coefficients of multiply charged ions. Astron. Astrophys. **656**, 163 (2021). https://doi.org/10.1051/0004-6361/202141673
13. S. Fritzsche, A.V. Maiorova, Z. Wu, Radiative recombination plasma rate coefficients for multiply charged ions. Atoms (2023). https://doi.org/10.3390/atoms11030050
14. S. Schippers, M. Martins, R. Beerwerth, S. Bari, K. Holste, K. Schubert, J. Viefhaus, D.W. Savin, S. Fritzsche, A. Müller, Near L-edge single and multiple photoionization of singly charged iron ions. Astrophys J **849**(1), 5 (2017). https://doi.org/10.3847/1538-4357/aa8fcc
15. R. Beerwerth, T. Buhr, A. Perry-Sassmannshausen, S.O. Stock, S. Bari, K. Holste, A.L.D. Kilcoyne, S. Reinwardt, S. Ricz, D.W. Savin, K. Schubert, M. Martins, A. Müller, S. Fritzsche, S. Schippers, Near L-edge single and multiple photoionization of triply charged iron ions. Astrophys J **887**(2), 189 (2019). https://doi.org/10.3847/1538-4357/ab5118
16. OPEN-ADAS. Accessed on 09 July 2024. https://open.adas.ac.uk/
17. C.A. Johnson, S.D. Loch, D.A. Ennis, ColRadPy: a python collisional radiative solver. Nuclear Mater. Energy **20**, 100579 (2019). https://doi.org/10.1016/j.nme.2019.01.013
18. S. Fritzsche, L.-G. Jiao, Y.-C. Wang, J.E. Sienkiewicz, Collision strengths of astrophysical interest for multiply charged ions. Atoms (2023). https://doi.org/10.3390/atoms11050080
19. F. Salvat, J.M. Fernández-Varea, radial: a Fortran subroutine package for the solution of the radial Schrödinger and dirac wave equations. Comput. Phys. Commun. **240**, 165–177 (2019). https://doi.org/10.1016/j.cpc.2019.02.011
20. JAC toolbox source code. https://github.com/OpenJAC/JAC.jl
21. Julia programming language documentation. Accessed on 28 June 2024. https://docs.julialang.org/en/v1.10/
22. ClusterManagers.jl package. Accessed on 28 June 2024. https://github.com/JuliaParallel/ClusterManagers.jl
23. SlurmClusterManager.jl package. Accessed on 28 June 2024. https://github.com/kleinhenz/SlurmClusterManager.jl

24. M. Asplund, N. Grevesse, A.J. Sauval, P. Scott, The chemical composition of the sun. Annu. Rev. Astron. Astrophys. **47**, 481–522 (2009). https://doi.org/10.1146/annurev.astro.46.060407.145222
25. Y. Liu, X.-W. Liu, M.J. Barlow, S.-G. Luo, Chemical abundances of planetary nebulae from optical recombination lines - II. Abundances derived from collisionally excited lines and optical recombination lines. Mon. Not. Royal Astron. Soc. **353**(4), 1251–1285 (2004). https://doi.org/10.1111/j.1365-2966.2004.08156.x
26. S.R. Pottasch, R. Surendiranath, J. Bernard-Salas, T.L. Roellig, Abundances in planetary nebulae: NGC 2792*. Astron. Astrophys. **502**(1), 189–197 (2009). https://doi.org/10.1051/0004-6361/200912076
27. B.T. Draine, Interstellar dust grains. Annu. Rev. Astron. Astrophys. **41**, 241–289 (2003). https://doi.org/10.1146/annurev.astro.41.011802.094840
28. H.L. Zhang, D.H. Sampson, Relativistic distorted-wave collision strengths and oscillator strengths for the 105 $\Delta n = 0$ transitions with $n = 2$ in the 85 b-like ions with $8 \leq Z \leq 92$. At. Data Nucl. Data Tables **56**(1), 41–104 (1994). https://doi.org/10.1006/adnd.1994.1002
29. H.L. Zhang, A.K. Pradhan, Atomic data from the IRON Project* - XXIII. Relativistic excitation rate coefficients for Fe XXII with inclusion of radiation damping. Astron. Astrophys. Suppl. Ser. **123**(3), 575–580 (1997). https://doi.org/10.1051/aas:1997352
30. S. Fritzsche, P. Palmeri, S. Schippers, Atomic cascade computations. Symmetry (2021). https://doi.org/10.3390/sym13030520
31. S. Fritzsche, A.K. Sahoo, L. Sharma, Z.W. Wu, S. Schippers, Merits of atomic cascade computations. Eur. Phys. J. D **78**, 75 (2024). https://doi.org/10.1140/epjd/s10053-024-00865-z