Introductions: < 1 min (instructors) + ~5 mins (attendees)

Daniel:

**Quick talk about Xray emission spectra**

radiative, auger and satellite transitions. Here we will explain the base blocks of what we need to calculate to obtain the spectra, i.e. one-hole configurations, two-hole configurations and the transitions between them. For radiative we need one-hole to one-hole combinations, for auger one-hole to two-hole and for satellite two-hole to two-hole.

Also talk about the emission spectra applied to optically thin plasma emission. The main difference will be the emission from different charge states and the ionization/excitation mechanism also being driven by thermal energy, not just electron impact ionization.

**Timetable: 5-10 mins depending on how in depth we explain + 5 min for questions. Total 5-15 mins**

**Make a case for why we, as the scientific community, need synthetic spectra to improve the results of our experiments and further probe the laws of physics.**

Here we can show examples of fitted synthetic spectra to experimental spectra and explain that this will be the main objective of the session: calculate, visualize and fit synthetic spectra to an experimental one.

**Timetable: 5 mins this should be mainly showing a few examples + 5 min for questions. Total 5-10 mins**

**Present the tools we will be using**

Text

Description automatically generatedTo perform our calculations, we will use MCDFGME, with a custom parallelization bash script. We will then visualize the results with a custom python script. First we will showcase the parallelization script and calculate a spectrum, afterwards we will show you the visualization program and explore parts of its code.

MCDFGME (Multi-Configuration Dirac-Fock General Matrix Element) is a state-of-the-art code that implements the Multi-Configuration Dirac-Fock method to calculate most properties of an atomic system. This code also implements other QED effects such as vacuum polarization and self-energy and other interactions, either during the self-consistent field calculation or as a perturbation in the end of the calculation. We will be using it to calculate the energies of each one and two-hole configurations and the transition rate and width between them.

Below is an example of a decomposition for the fundamental LS configuration of Ni. If there are no external fields, we don’t need to calculate for each MJ as these configurations will be degenerate.

A picture containing diagram

Description automatically generated

To fully calculate the spectrum of an atomic system we need to often calculate hundreds of one-hole and two-hole configurations, which will generate 100000+ transitions. For example, the neutral Fe atom has 26 electrons which makes it possible to have 7 one-hole LS configurations and 28 two-hole configurations. After converting to jj coupling, this will generate 210276 radiative transitions and 951621 auger transitions, and even more satellite transitions.

Text

Description automatically generatedThis large number of calculations creates a need to automate and parallelize the task, which was also performed by prof. Jorge Machado from the department. This was done by developing a bash script to take advantage of the gnu parallel tool and execute in the cluster computing server (orange) of the department, where we have access to a total of 64 cores to parallelize this task.

This script first calculates the energies and wavefunctions of each jj configuration and attempts to obtain a good convergence on the energy of each configuration. To assess this, we use a value that MCDF calculates for us that corresponds to the difference between the energy calculated by summing the energies of each orbital and the energy calculated from integrating the total wavefunction calculated. A good convergence on the calculation is signified by a difference in the order of 0.01eV, and by using this value the script performs 4 cycles where it selects the configurations where the convergence was poor and recalculates them using different parameters on the routine that obtains the convergence. In the end if there are still configurations that have not achieved a good convergence, the script prints their quantum numbers so that they can be identified and rerun by hand. Another value that could be important in the case of having to choose the parameters by hand are the overlap, or internal product, between the orbitals. If this value is larger than ~10^-7 the orbital wavefunctions might not be sufficiently orthogonal.

**We will be calculating the spectrum for 10 charge state of Fe and simulate the spectrum of an optically thin plasma, according to a charge state distribution. (Fe +13 through Fe +22)**

**Text

Description automatically generatedExample for Mg:**

**This atom has 12 electrons and a fundamental LS configuration of 1s2 2s2 2p6 3s2**

**A picture containing arrow

Description automatically generatedThis is a quick atom to calculate because all the orbitals are filled resulting in much less combinations of J when we generate the one and two-hole configurations.**

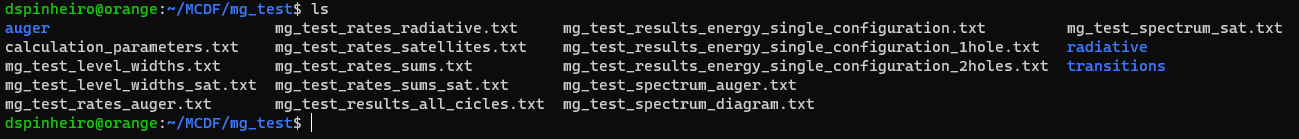
**In this case we make use of the function to automatically compute the possible configurations assuming that the number of electrons is the same as the atomic number. If we wish to calculate an ion or a specific set of transitions, we can read the configurations from a file such as:**

**A picture containing text, outdoor

Description automatically generated**

**This are the one and two-hole configurations used to calculate the spectrum of the Fe +8 ion.**

**After performing the calculations, the directory chosen will be populated with files containing all the results:**

****

**All the files are prefixed with the folder name and each one has a specific set of results:**

**auger**

folder containing the two-hole configuration results, i.e. energies and wavefunctions among other

this is structured as:

1s\_1s

└ 2jj

└ eig

└ input and outputs of the MCDFGME execution

**radiative**

folder containing the one-hole configuration results, i.e. energies and wavefunctions among other

1s

└ 2jj

└ eig

└ input and outputs of the MCDFGME execution

**transitions**

folder containing the transition results.

auger radiative satellites

└ transition number

└ input and output of the MCDFGME execution

**mg\_test\_results\_energy\_single\_configuration.txt**

combined results for all one and two-hole energy calculations. In here we can also see the accuracy of the convergence

**calculation\_parameters.txt**

file containing a more detailed log of the configurations calculated in each cycle

**mg\_test\_results\_energy\_single\_configuration\_1hole.txt**

results for all one-hole energy calculations. In here we can also see the accuracy of the convergence

**mg\_test\_level\_widths.txt**

file containing the radiative, auger and total widths of each orbital

**mg\_test\_rates\_sums.txt**

file containing the radiative and auger rates of each orbital as well as the fluorescence yield

**mg\_test\_results\_energy\_single\_configuration\_2holes.txt**

results for all two-hole energy calculations. In here we can also see the accuracy of the convergence

**mg\_test\_level\_widths\_sat.txt**

file containing the radiative, satellite and total widths of each orbital

**mg\_test\_rates\_sums\_sat.txt**

file containing the radiative rates of each two-hole combination

**mg\_test\_results\_all\_cicles.txt**

file containing the one-hole and two-hole convergence results for all the configurations that did not reach a good convergence in each of the 4 cycles

**mg\_test\_rates\_radiative.txt**

file containing the radiative transition rates for all calculated radiative transitions

**mg\_test\_spectrum\_diagram.txt**

file containing the radiative spectrum

**mg\_test\_spectrum\_sat.txt**

file containing the satellite spectrum

**mg\_test\_rates\_satellites.txt**

file containing the satellite transition rates for all calculated satellite transitions

**mg\_test\_spectrum\_auger.txt**

file containing the auger spectrum

**mg\_test\_rates\_auger.txt**

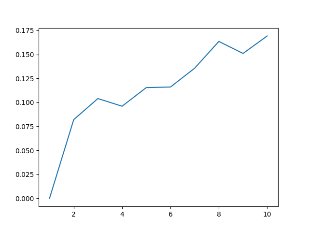
file containing the auger transition rates for all calculated auger transitions

**Timetable: 20-30 mins depending on how much we must explain about MCDF and the file structure + 5 min for questions. Total 25-35 mins.**

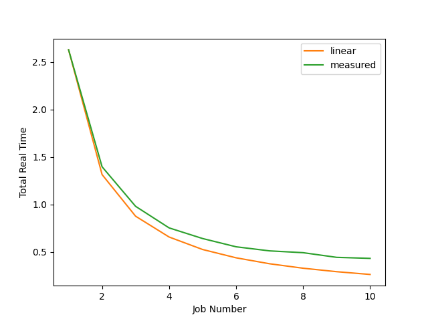
**Total time: 40-65 mins**

**From here we start the more hands-on section where they will be executing and exploring the results of a different atomic system.**

**Task 1 (15-30 mins) – Demonstration, the participants won’t need to code this**

**Write a function in bash to sum the first n integers and use GNU parallel to parallelize the execution of this function for various values of n.**

test\_func() { sum=0; for (( i = 0; i <= $1; i++ )); do (( sum += i )); done; echo $sum; }

**Time the execution of the parallel command for different number of parallel threads and plot the total execution time vs the number of threads used.**

**For a given $job and inputs: time parallel -j$job test\_func ::: {input values separated by spaces or a text file}**

**For 100 executions of n = 5000**

**Task 2 (20-30 mins, more likely it would be 20 mins but there might be problems or other delays)**

**First, we need to calculate our spectrum. Log into orange and, using the MCDF parallelizing script, calculate one of the suggested Fe ions. Afterwards process the spectrum with the provided conversion script to reformat it to be used in the visualization program.**

**André:**

**Now that we have our calculations, we will explore the visualization program.**

Graphical user interface, application

Description automatically generatedTable

Description automatically generatedThe custom python script was developed in the LIBPhys-UNL research group of the physics department where several standard and fully tested python modules were used to process (numpy), interpolate (scipy), fit (lmfit) and plot (matplotlib) the calculated data. Show general functionality of the program.

**Task 3 (20-30 mins, we can also give the formulas for each to make things faster)**

**In plotting synthetic spectra and comparing it to experimental data, depending on the experimental sample and setup, we might obtain a better fit if we use different line shapes. The line shape of only the emission can be considered a Lorentzian profile, and most broadening mechanisms can be considered to broaden the measured data according to a Gaussian profile. Sometimes a convolution of these two profiles is also used which is called a Voigt profile.**

* 1. **– Program a function that, given the centroid, intensity, total width of a transition and experimental resolution, returns the values of y for the given values of x, according to a Lorentzian profile:**

def L(T, energy, intens, res, width):

""" Return Lorentzian line shape of every x in T for the given transition with the given resolution """

* 1. **– Program a function that, given the centroid, intensity, total width of a transition and experimental resolution, returns the values of y for the given values of x, according to a Gaussian profile:**

def G(T, energy, intens, res, width):

""" Return Gaussian line shape of every x in T for the given transition with the given resolution """

* 1. **– Program a function that, given the centroid, intensity, total width of a transition and experimental resolution, returns the values of y for the given values of x, according to a Voigt profile:**

def V(T, energy, intens, res, width):

""" Return Voigt line shape of every x in T for the given transition with the given resolution """

**Text

Description automatically generated**

**Current Code:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Task 4 (30-60 mins)**

**Write several other functions used to manipulate data in the simulations program**

**4.1 – Read the spectrum files into arrays. This is done in the fileIO module:**

**def readRates(rates\_file):**

**"""**

**Function to read the rates file**

**Args:**

**rates\_file: file path of the rates file**

**Returns:**

**linerates: list with the data still in string format**

**"""**

**4.2 – Update the radiative and satellite transition rates depending on the selected transition. This is done in the functions module:**

**def updateRadTransitionVals(transition, num):**

**"""**

**Function to update the radiative and satellite rates for the selected transition**

**Args:**

**transition: which transition to fetch the rates of**

**num: total number of transitions processed**

**Returns:**

**num\_of\_transitions: total number of transitions processed**

**low\_level: low level of the selected transition**

**high\_level: high level of the selected transition**

**diag\_stick\_val: rates data for the selected transition**

**sat\_stick\_val: rates data for the possible satellite transitions for the selected transition**

**"""**

**4.3 – Using the transition rates, calculate the simulated y values according to the chosen profile. This is done in the functions module, you only need to write the diagram / auger calculations:**

**def y\_calculator(sim, transition\_type, fit\_type, xfinal, x, y, w, xs, ys, ws, res, energy\_values, efficiency\_values, enoffset):**

**"""**

**Function to calculate the simulated intensities for all the transitions requested, taking into account the simulated offsets.**

**This function is used only to apply the selected profile to the already filtered x, y and width values for the transitions.**

**Args:**

**sim: tkinter simulation window required to update the progress bar**

**transition\_type: type of transition to be simulated (diagram data comes in the x, y, w and satellite data in the xs, ys, ws)**

**fit\_type: profile type selected in the interface**

**xfinal: simulate x values**

**x: energy values for each diagram transition to simulate**

**y: intensity values for each diagram transition to simulate**

**w: natural width values for each diagram transition to simulate**

**xs: energy values for each satellite transition in each radiative transition to simulate**

**ys: intensity values for each satellite transition in each radiative transition to simulate**

**ws: natural width values for each satellite transition in each radiative transition to simulate**

**res: experimental resolution to simulate**

**energy\_values: energy values read from the detector efficiency data**

**efficiency\_values: efficiency values read from the detector efficiency data**

**enoffset: energy offset to simulate**

**Returns:**

**yfinal: list of simulated y values for each diagrma transition we want to simulate for each of the x values in T**

**ytot: list of the simulated total y values for all transitions we want to simulate for each of the x values in T**

**yfinals: list of simulated y values for each satellite transition in each digram transition we want to simulate for each of the x values in T**

**"""**

**4.4 – Determine the red. chi^2 for the simulation and an experimental spectrum. For this the calculated spectra will need to be normalized to the experimental maximum or both to unity:**

**def calculateResidues(exp\_x, exp\_y, exp\_sigma, xfinal, enoffset, normalization\_var, normalize, y0, number\_of\_fit\_variables, residues\_graph):**

**"""**

**Function to calculate the residues, reduced chi^2 and update the respective graph**

**Args:**

**exp\_x: energy values from the experimental spectrum**

**exp\_y: intensity values from the experimental spectrum**

**exp\_sigma: error values from the experimental spectrum**

**xfinal: list of the simulated x values**

**enoffset: simulated energy offset**

**normalization\_var: normalization multiplyer**

**normalize: normalization type chosen**

**y0: simulated intensity offset**

**number\_of\_fit\_variables: total number of fitted variables**

**residues\_graph: matplotlib plot object where to plot the residue data**

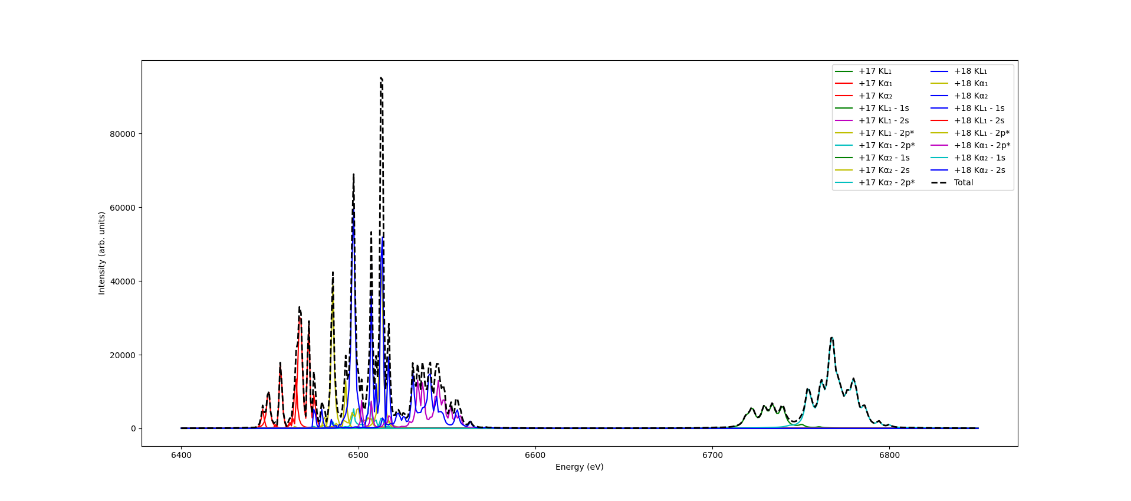
**Returns:**

**Nothing, the residues are plotted and the chi^2 value is updated in the variables module**

**"""**

**4.5 – After writing these functions, perform an autofit of a simulation to an experimental spectrum. Verify that the fitted parameters correspond to the lowest reduced chi^2.**

**Task 5 (5 mins, this should just be quickly using the program to visualize the spectrum they calculated in the orange)**

**A picture containing graphical user interface

Description automatically generatedShare the calculated spectra for different charge states between yourselves and use the provided plotting code to visualize the calculated spectrum using each of the programed functions.**

**Diagram, histogram

Description automatically generated**

**Total time: 130-220 mins -> 2h10min – 3h40min**