Journal of Applied Research and Technology

www.jart.icat.unam.mx

Journal of Applied Research and Technology 24 (2026) 42-52

Original

A Python-based web tool to simulate atomic optical emission spectra

O. Rosales-Martínez[1] • Allan A. Flores-Fuentes[* 2]
R. Peña-Eguiluz[2, 3] • A. Mercado-Cabrera[3]
E. E. Granda-Gutiérrez[2] • J. F. García-Mejía[2]

[1]Universidad Autónoma del Estado de México, Centro Universitario Tianguistenco, Santiago Tianguistenco, México, MX 52640.
[2]Universidad Autónoma del Estado de México, Centro Universitario Atlacomulco Km. 60 Carretera Toluca-Atlacomulco, 50450, Atlacomulco, México, MX 50450.
[3]Instituto Nacional de Investigaciones Nucleares, Plasma Physics Laboratory, Carretera México-Toluca s/n, La Marquesa, Ocoyoacac, México, MX 52750.

Received: 11 22 2024; Accepted: 02 27 2025
Available: 02 28 2026

Abstract: Optical emission spectroscopy (OES) is an experimental technique for analyzing, diagnosing, and characterizing atomic and molecular species in physics and chemistry. In this work, an interactive web tool is designed to simulate atomic OES from which users can generate synthetic data from optical emission spectra at a wavelength range from 390 to 700 nm with any reported energy level. The reference data for eighty-three elements was retrieved from the National Institute of Standards and Technology (NIST) using web scraping techniques. The web tool incorporates key parameters such as wavelength step size (λ), full width at half maximum (FWHM), and emission source temperature (T). The proposed method for generating spectra consists of preprocessing, calculation, and plotting of the data using Python, Pandas, NumPy, and Matplotlib, respectively. Finally, the interactive user interface, built with the open-source Python framework Streamlit, displays changes in spectral plots generated from input data and updates live, supporting zoom, serving as a valuable resource for researchers and students in the field of OES.

*Corresponding author.
E-mail address: aafloresf@uaemex.mx (Allan A. Flores-Fuentes).
Peer Review under the responsibility of Universidad Nacional Autónoma de México.

Keywords: **Optical emission spectroscopy, Python, interactive web tool, NIST data, synthetic data.**

1. Introduction

Interactive simulations that include physics concepts are viable approaches to improve understanding of abstract subjects, particularly in quantum mechanics, such as the atomic spectra of the elements.

As the time of confinement due to the COVID-19 pandemic increased, some universities worldwide implemented virtual education systems for teaching-learning in various areas of knowledge (Huang et al., 2021; Ochkov et al., 2022; Ojeda Misses & Jimenez, 2021). Some studies emphasize implementing software tools for several spectroscopy topics, for instance, web interfaces, to understand the basic principles of chromatography (Jordheim et al., 2021). Other works focus on using standard spreadsheet software for several tasks, including basic calculations for chemistry majors (Cooper, 2018), modeling peak shapes for simulation of predicted chromatograms (Fasoula et al., 2017), and estimating peak area and height from spectroscopic or chromatographic data (Vieira et al., 2019). Synthetically generated data have extensive applications (Blackmore et al., 2023; Oinonen et al., 2024; Shah et al., 2023), such as creating simulated environments like urban scenes for autonomous driving in deep learning (Nikolenko, 2021) or synthetic data to train and choose machine learning models (Fintz et al., 2021). Another interesting area of exploration involves the simulation process. There are several works in the literature that present Python-based tools for simulating physical and chemical processes, which allow users to perform better in calculating and evaluating results (Huang et al., 2021; Ochkov et al., 2022; Ojeda Misses & Jimenez, 2021). Later in post-pandemic era there are remote options allow students access to a physical laboratory to conduct experiments (Pretz, 2021); virtual reality (VR) headsets that facilitates students to learn math and science academic and research communities through conference hybrid events, while the availability of universal access to the knowledge even that was successfully implemented during pandemic time, nowadays is declined in some spaces (Brown, 2023).

In this proposal the generation of optical emission spectra from atomic chemical elements is proposed through the option of parameters such as the wavelength step size (λ), full width at half maximum (FWHM), emission source temperature (*T*), and energy level. It helps to easily understand the creation of synthetic spectra and the effect of the variations of their parameters. The suggested method presents an interactive web application in Python, with capabilities to: a) generate an atomic spectrum for a user-defined temperature and wavelength range from one of the eighty-three elements based on NIST reported database, b) create a spectrum via an interactive user interface made with *Streamlit*, c) integrate an overlapping spectrum from two or more elements, d) generate a synthetic data spectra in .png format, and e) use updated data reported in NIST, through web scraping technique. Those characteristics provide a comprehensive and dynamic free software tool to support understanding applied spectroscopy concepts (Kolpaková et al., 2011). It is important to mention that a class called *Generator* can be modified to add requirements and extend its applications.

In addition, the presented methodology uses free software and represents an advantage concerning the conventional use of licensed software to simulate mathematical models (Kolpaková et al., 2011; Martinez-Urreaga et al., 2003). While fees and restrictions are associated with utilizing the app-licensed software, this web-based interface offers a permitted option for real-time simulation parameter adjustment. This tool promotes learning among graduate and undergraduate students; moreover, it is useful for researchers. Furthermore, it also facilitates the identification of real spectra for different investigations that involve this technique.

Several stages were carried out to achieve the proposed goal: 1) retrieve atomic emission lines from the National Institute of Standards and Technology (NIST) cloud database, 2) preprocess the acquired data, 3) perform *Python* class definition for creating synthetic spectra, 4) filter preprocessed data, 5) data integration into *Python Pandas DataFrames*, 6) class functionality addition to fit and create synthetic spectra, and 7) use *Streamlit* to parse *Python* script into an interactive website (Kumar, 2020; Schares, 2022; Yu et al., 2017). Following the outlined steps, we created a user-friendly web page interface to generate synthetic spectra. Moreover, interested readers can use the ideas presented here to build

their own synthetic spectrum generator in their preferred programming language.

2. Methodology

We used *Miniconda* to build a virtual environment with *Python 3.7.1*. After the virtual environment is activated, *JupyterLab*, *Pandas*, *Seaborn*, and *Streamlit* are installed as follows:

```
(base) C:\Users\Propietario>conda create -n synthetic
python=3.7.1
(base) C:\Users\Propietario>conda activate synthetic
(synthetic) C:\Users\Propeitario>pip install jupyterlab
pandas seaborn streamlit
```

After this procedure, the software packages are ready to use. Data acquisition was performed by modifying the NIST download URL (see the Associated Content section) and replacing the string representing each chemical element with the corresponding element from a custom comma-separated values (CSV) file, one element at a time. Subsequently, the acquired data underwent preprocessing to remove any unwanted characters. A *type* column was then added to distinguish between data captured in a vacuum or an air environment. The columns were reordered, as depicted in the first two stages within the dashed box in Figure 1. This procedure is executed only once, and the resulting data is stored in a *pickle* object for later use.

Later, we programmed a class named *Generator,* as depicted in stage 3 (see Figure. 2). This class is a fundamental component for creating instances with customizable arguments, and its *DataFrames* and methods will be expanded upon in the forthcoming stages (see Figure. 1).

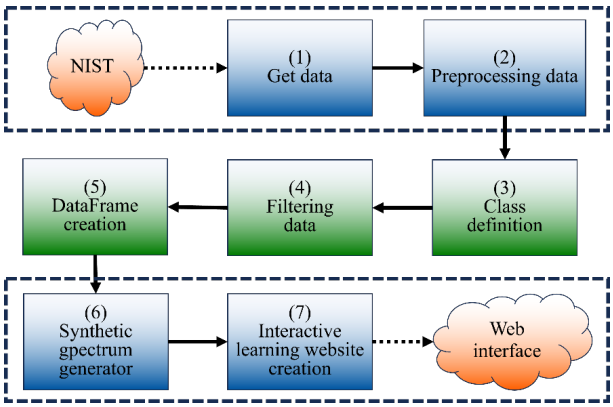

Figure 1. General scheme for the generation of synthetic spectra.

Then, the *__init__* method is called and executed when a class instance is created; also, a pickle file containing the NIST data is loaded and initializes the attributes specified in Table 1. For stage 4 (Figure. 1), the *fit* method yields the *DataFrame df_base* containing the NIST-filtered data based on the *sp_num*, *stype*, element attributes, and an index between *lower_limit* and *upper_limit*. The *__init__* method initializes the object attributes and assigns them to their corresponding class attributes, denoted by a preceding '_' (underscore symbol).

The latter task is achieved by calling the private method *_ filtering*, and it is updated every time any of the parameters is modified in the columns shown in Table 2, section A. Moving forward to stage 5 (Figure. 1), a *DataFrame* called *df_calc* is created to perform calculations and generate the synthetic spectrum with values (λ) from the Table 2 section B. The index is created at this stage based on the following priorities: *index*, *num*, and *step*. This process is executed by calling the private method *_create_df_calc*.

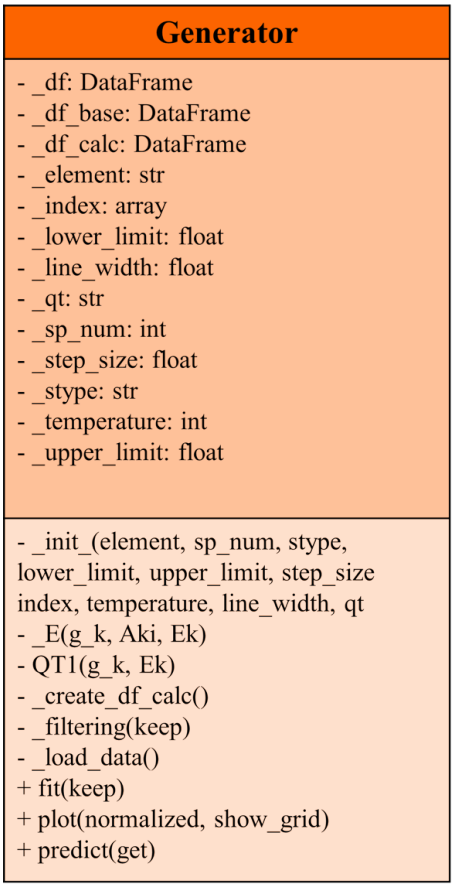

Figure 2. Class diagram for the *Generator class*.

Table 1. Default values for an instance of the Generator class.

Attribute	Default value	Description
element	'H'	Choice of element for synthetic spectrum generation (H=Hydrogen)
sp_num	1	Energy level
stype	'Air'	'Air' or 'Vacuum' for the spectrum's environment
lower_limit	390.0	Wavelength's lower limit in nm
upper_limit	700.0	Wavelength's upper limit in nm
index	[ ]	Array of points between lower_limit and upper_limit
step_size	0.5	Step size for points generation between lower_limit and upper_limit
temperature	5000	Synthetic spectrum temperature in K
line_width	0.3	FWHM of each peak of the synthetic spectrum
qt	'auto'	In 'auto' an automatic calculation is performed for the temperature-dependent partition function; otherwise, a value must be provided

In stage 6 (Figure 1), the predict method calculates the temperature-dependent partition function ($Q(T)$), defined by Eq. (1) when *qt='auto'*; otherwise, the provided value is used (Fowler & Guggenheim, 1939).

$$\mathrm{Q}\left(\mathrm{T}\right) = \sum_{\mathrm{k}=0}^{\mathrm{n}} \mathrm{g_k e}^{\left(-\mathrm{E_k}/\mathrm{k_b T}\right)} \quad (1)$$

Where g_k is the statistical weight, E_k is the energy of the electronic level k in electron volts (eV), k_b is the Boltzmann constant (eV/K), T is the temperature in Kelvin (K), and n is the electronic level before ionization. For certain species, such as inert gases: He, Ne, Ar, etc., the value of the partition function is unity, while for H, its value is two (Hill, 2012)This characteristic is valid for a temperature range from 1500 K to 15000 K. For other species, the partition function value is obtained by applying Eq. (1).

Eq. (2) is used to calculate the intensity of the emission spectrum (I) for each considered line. The numerator and denominator are calculated separately to avoid issues with *NumPy* broadcasting.

$$\mathrm{I} = \mathrm{A_{ki}} \frac{\mathrm{g_k}}{\lambda_\mathrm{C} \mathrm{Q}} \left[\frac{2}{\pi} \bullet \frac{\mathrm{w}}{4\left(\lambda - \lambda_\mathrm{C}\right)^2 + \mathrm{w}^2} \right] \mathrm{e}^{-\mathrm{E_k}/\mathrm{k_b T}} \quad (2)$$

Where w represents the full width at half maximum (nm), A_{ki} is the transition probability (s^{-1}), Q is the temperature-dependent partition function, λ_C is the NIST wavelength within the indicated range (nm), and λ is the wavelength with a defined step size (nm) between each pair of adjacent points. The term in square brackets denotes a Lorentzian peak profile. This type of profile was chosen because in general, in all processes that involve interaction of the emitting species with charged or neutral species (Stark effect and pressure effect) they generate a widening of the emitted line, hence the first approximation is the Lorentzian profile, which is also considered the natural profile of a line. The Stark effect predominates in the broadening of spectral lines at atmospheric pressure and higher pressures (Stark broadening) due to a high collision rate, which occurs when the ionization rate exceeds 1%. While the Doppler effect leads to a Gaussian profile, which is generally present at low pressures (Cabannes, 1974). Afterward, each calculation is integrated as a new column of a single row corresponding to a wavelength, aggregating the resulting rows. The outcomes are then placed in the *final_spectrum* column, which, after normalization, is stored in the *normalized_spectrum* column.

3. Results

The *DataFrame df_calc*, as visualized in Table 2 section B, transforms into the *DataFrame* shown in Table 3. Hence, it becomes feasible to plot the *normalized spectrum for Hydrogen I (H I), for example, as depicted in Figure* 3A and Neon I (Ne) in Figure 3B. Both H I and Neon I spectra in Figure 3 were simulated under the same conditions: a temperature of 5000 K, a wavelength step size of 0.5 nm, and a FWHM (denoted as "line width" in the UI) of 0.3 nm. In both figures, peaks with intensities in hundredths and thousandths of a unit are observed through zooming in. This is because the data used to build the spectrum from the NIST URL is constantly downloaded and updated. A comparison between the spectra obtained in an Excel spreadsheet by (Flannigan, 2014) and those generated by the proposed Python interface were included in Figure 3. As a result, some differences are observed between the spectra because our results use the most recent data reported by NIST. Magnified sections of each spectrum highlight our software's sensitivity in plotting each element's lines.

Table 2. Extracted data from *JupyterLab* sections: (A) the generated DataFrame with default arguments for H I, and (B) the *DataFrame* for performing calculations with an index (λ) at a step of 0.5.

A							B
λ_C	Aki(S^{-1})	Ek(eV)	g_k	type	Sp_num	element	λ
397.007500	439000.0	13.320918	98	1	1	H	390.0
410.173400	973000.0	13.220704	72	1	1	H	390.5
434.047200	2530000.0	13.054502	50	1	1	H	391.0
486.128695	9670000.0	12.748538	4	1	1	H	391.5
486.129776	9670000.0	12.748532	2	1	1	H	392.0
486.13500	8420000.0	12.748539	32	1	1	H	392.5
..	..	..	..	..	..	..	..
656.270970	53900000.0	12.087507	4	1	1	H	698.0
656.273483	22400000.0	12.087507	4	1	1	H	698.5
656.277153	22400000.0	12.087494	2	1	1	H	699.0
656.279000	44100000.0	12.087505	18	1	1	H	699.5
656.285175	64700000.0	12.087511	6	1	1	H	700.0

Table 3. *DataFrame* extracted from the *JupyterLab* after performing the calculations. From left to right, λ is the index with a step size of 0.5 nm, λ_C are the lines reported in the NIST for H I, and the last two columns are the results of applying Eq. (2).

	λc			…	λc	Results	
λ	397.0075	410.1734	434.0472	…	656.285175	final_ spectrum	normalized_ spectrum
390.0	2.1491426e-07	4.7283196e-0	2.1537410e-08	…	4.8611487e-09	0.0000003	0.0000167
390.5	2.4919042e-07	4.9717003e-08	2.2034820e-08	…	4.8794556e-09	0.0000004	0.0000187
291.0	2.9236956e-07	5.2343672e-08	2.2549662e-08	…	4.8978662e-09	0.0000004	0.0000211
391.5	3.4782384e-07	5.518413le-08	2.3082761e-08	…	4.9163812e-09	0.0000005	0.0000242
392.0	4.2068686e-07	5.8262226e-08	2.363499le-08	…	4.9350031e-09	0.0000005	0.0000283
…	…	…	…	…	…	…	…
698.0	1.1654105e-10	2.3228815e-10	5.997663e-10	…	1.9808270e-07	0.0000008	0.0000412
698.5	1.1615482e-10	2.314832le-10	5.9750102e-10	…	1.9341829e-07	0.0000008	0.0000402
699.0	1.1577052e-10	2.3068244e-10	5.9524803e-10	…	1.8891672e-07	0.0000007	0.0000393
699.5	1.153881le-10	2.2088582e-10	5.9300775e-10	…	1.8457049e-07	0.0000007	0.0000384
700.0	1.1500760e-10	2.2009332e-10	5.9078010e-10	…	1.8037254e-07	0.0000007	0.0000375

In stage 7 (Figure 1), *Streamlit* is used to parse the *Python Script* to the web interface, automating the generation of synthetic spectra from any computer with a browser and Internet access, with the design depicted in Figure 4A. The parameters, namely *element*, *lower_limit*, *upper_limit*, *sp_num*, and *type* for filtering NIST data, are previously saved in a *pickle* object. It should be noted that each time an element is chosen from the list box, the energy levels are updated Figure 4B depicts an additional section for configuring the synthetic spectrum with the arguments *temperature* (*T*), *QT* (manual or automatic), *step size* (nm), *line width* (nm), and the option to select the kind of plot with the *Normalized* (*True* or *False*) radio button.

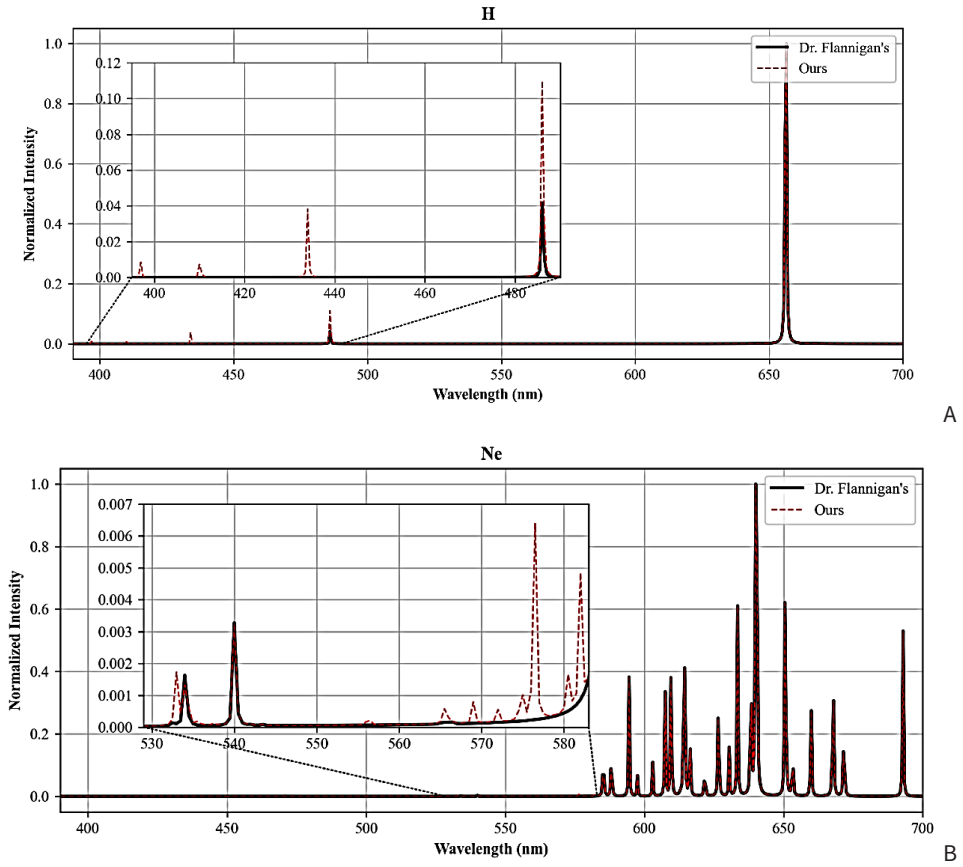

Figure 3. Normalized synthetic spectrum in the wavelength range from 390 to 700 nm using default arguments, A) Hydrogen I, B) Neon I.

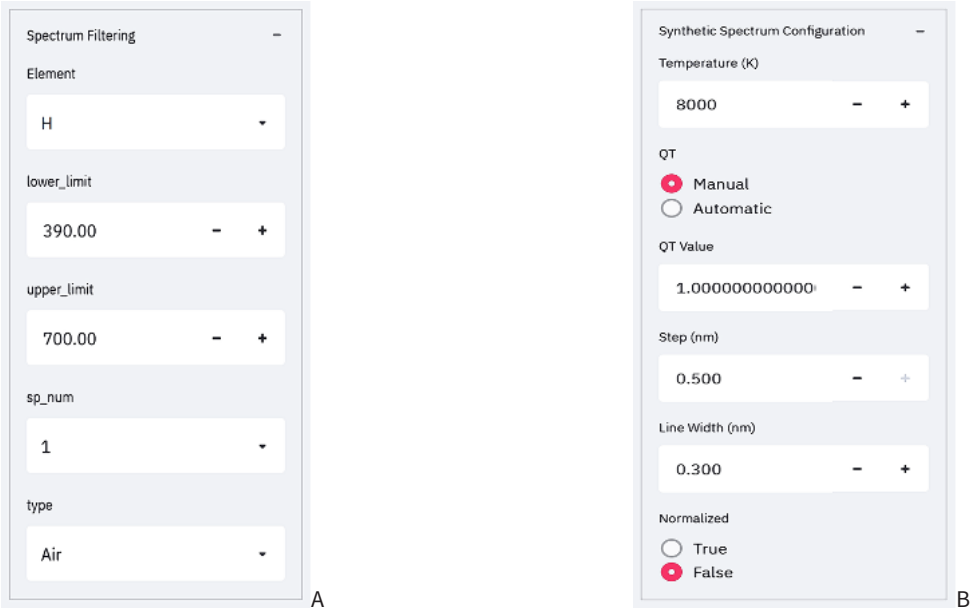

Figure. 4. Sections, A) to filter the data obtained from NIST, and B) to configure and generate the synthetic spectrum.

The methods for generating synthetic spectra are motivated by the object-handling capabilities of the free machine-learning *scikit-learn* package. The development and testing mode code is presented below:

```
from synthectic_spectrum import Generator
import matplotlib.pyplot as plt

ss1 = Generator('Ar')
(ss1
  .fit()
  .predict()['normalized_spectrum']
  .plot(figsize=(13, 5), label='Ar'))
ss2 = Generator('Hg')
(ss2
  .fit()
  .predict()['normalized_spectrum']
  .plot(figsize=(13, 5), label='Hg'))
ss3 = Generator('Ne')
(ss3
  .fit()
  .predict()['normalized_spectrum']
  .plot(figsize=(13, 5), label='Ne'))
plt.legend()
plt.ylabel('Normalized Intensity')
plt.xlabel('Wavelength (nm)')
    plt.show()
```

This code snippet executes the *Generator* class in *JupyterLab.* It demonstrates the generation and plotting of three synthetic spectra in a few lines, each in a color corresponding to argon (Ar), mercury (Hg), and neon (Ne). Moreover, it is possible to generate more element spectra and plot them on a single graph (refer to Figure 5). Notably, the plotted curves in the graph are independent, indicating no convolution operation between them.

This programming approach for generating synthetic spectra simplifies implementing an interactive website for stage 7 (Figure 1) by incorporating Streamlit controls and using their values as arguments to generator class objects. The *Python* script can be parsed into a web page, as illustrated in Figure 6. The interactive mode in the web tool allows the user to perform real-time tasks such as: a) display the options' menu, b) zoom in at any section of the spectrum, and c) download the plot as a file in PNG format, as shown in Figure 6.

4. Discussion

Figures 3 and 5 present more spectral lines than reported in the work by Flannigan (). This discrepancy arises from the increased number of transition probabilities available in the data extracted from the NIST database. Our methodology allows for the creation of flexible interfaces to generate synthetic spectra using free software like Python and the other mentioned packages. Furthermore, while Flannigan's work is limited to five elements (H, Li, Ne, Na, Hg) for a λ range from 390 to 700 nm at a fixed step size of 0.5 nm, our approach empowers the generation of spectra of 83 elements, with energy level reported in NIST. These comparative results are shown in the associated content through the link: https://github.com/orm030982/synthetic-spectra-files. As a result, users can experiment with all the parameters mentioned above and observe their effects in almost real-time due to the Streamlit library implemented in this work. Additionally, the simulator can generate synthetic spectra only if all variables in Eq. (2) are present in the NIST data.

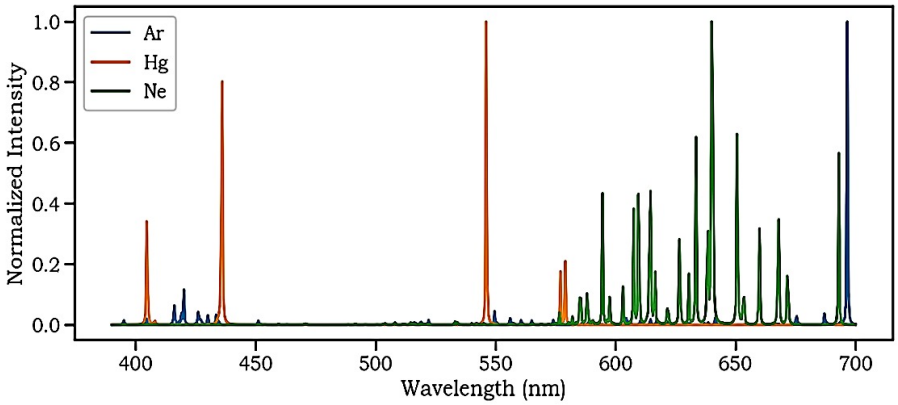

Figure. 5. Generated and plotted independent synthetic spectra for Ar, Hg, and Ne (this spectrum is not a mixture of gases) with default arguments in *JupyterLab.*

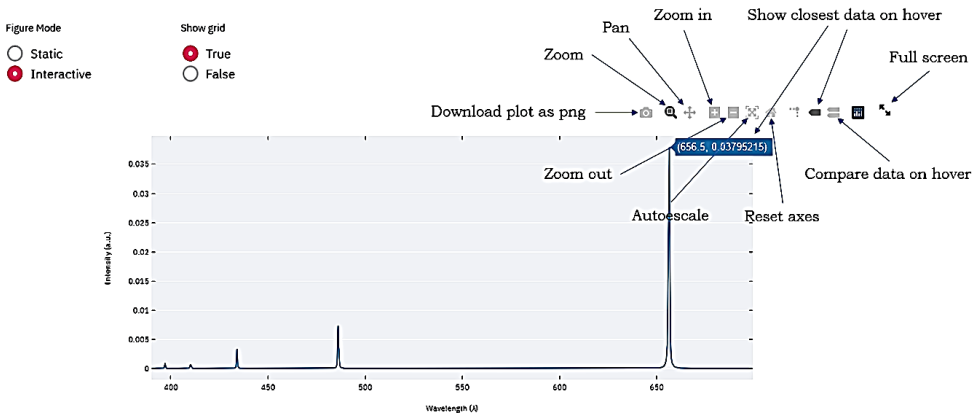

Figure 6. Web implementation of the spectra generator in interactive mode, with additional capabilities, including displaying line coordinates, enabling zoom functionality, and providing an option to save the plot.

Neon (Ne) is a useful example for showcasing various parameter variations, and the results are shown in Figure 7. These outcomes demonstrate some promising applications of the proposed web interface in creating synthetic spectra. In this regard, specifically varying the step-size allows for better resolution of the curve and localization of wavelength for the spectrum lines, as shown in Figures 7A and 7D. In addition, reducing the line_width parameter produces a more defined curve profile, preventing overlap of adjacent peaks, as depicted in Figures 7B and 7E, where this parameter takes on two different values. When defining a different absolute temperature, for instance, a magnitude of temperature difference modifies the intensity profile of the lines, resulting in the spectra displayed in Figures 7C and 7F.

Finally, this document outlines the process of generating computer-based to simulate atomic emission spectra using free software. Users are not required to possess a background in computer programming; however, with the assistance of an expert in the field, it is feasible to replicate, organize, and generate the code based on specific requirements and the ideas presented in this paper.

5. Conclusions

This study presents an approach to simulating OES using an interactive web tool developed with Python and *Streamlit*. Based on reference data retrieved from the National Institute of Standards and Technology (NIST), this tool allows users to generate synthetic spectra for up to eighty-three elements across a wavelength range of 390 to 700 nm, while key parameters manipulation such as wavelength step size, emission source temperature, and full width at half maximum (FWHM), enable to analyze the effects of these parameters on the simulated spectra. The methodology involved several stages, including data retrieval, preprocessing, and calculations using Python libraries such as Pandas, NumPy, and Matplotlib. Integrating a custom *Generator* class facilitated the generation of synthetic spectra based on user-defined parameters. Comparative analyses with existing methods highlighted the tool's enhanced capabilities, offering a broader spectrum of elements and finer control over simulation parameters. Finally, the presented web tool represents a significant advancement in the field of simulated spectral analysis, offering an intuitive interface for users to explore and understand the complexities of atomic emission spectra effectively. Future developments could extend its applicability to additional spectroscopic techniques and expand its user base within the scientific community.

Associated Content

The Python Streamlit web interface simulates atomic spectra, showcasing the immediate effect of varying their parameters for 83 elements withing a flexible wavelength range. Explore the tool at: https://orosalesm-synthetic-spectra-g-synthetic-spectra-2nvtv9.streamlit.app

NIST Atomic Spectra Database Lines can be accessed at: https://physics.nist.gov/PhysRefData/ASD/lines_form.html

Images comparing the spectra generated by Flannigan and those from this work can be found at https://github.com/orm030982/synthetic-spectra-files

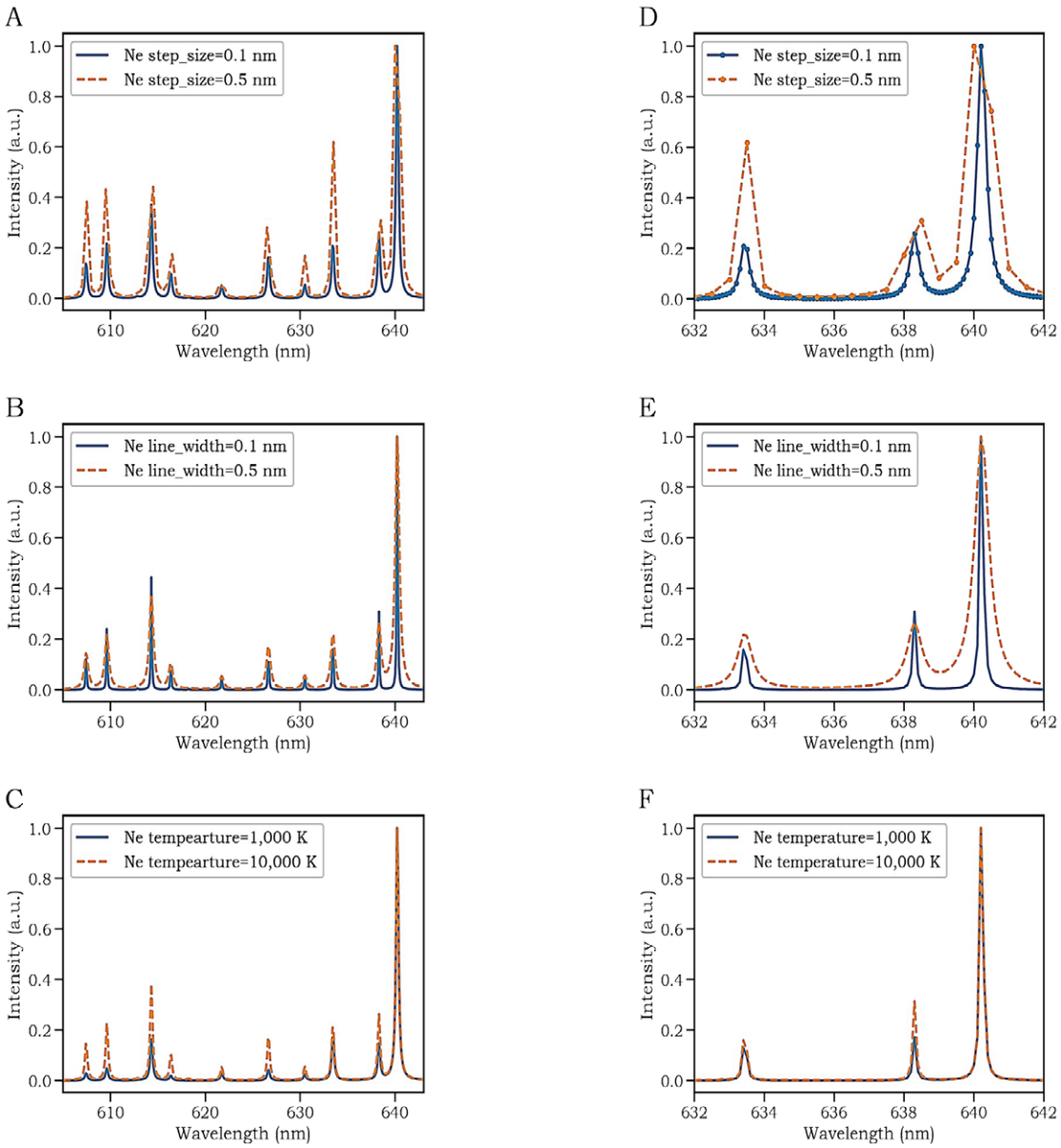

Figure 7. Synthetic spectrum tested with default attributes (Table 1). Panels A to C illustrate the variation of a single parameter, while panels D to F provide a zoom-in view to better distinguish the effect of the values assigned to the indicated argument.

Conflict of interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

O. Rosales-Martínez received support from a research grant (No. ESYCA2023-144285) awarded by the Consejo Mexiquense de Ciencia y Tecnología (COMECYT), México.

Acknowledgements

Thanks to the Universidad Autónoma del Estado de México/Centro Universitario UAEM Atlacomulco for the specialized research stay, and personnel of the Plasma Physics Laboratory of the Instituto Nacional de Investigaciones Nucleares (ININ), México, for their scientific contributions.

References

Blackmore, J. A., Gregory, P. D., Hutson, J. M., & Cornish, S. L. (2023). Diatomic-py: A Python module for calculating the rotational and hyperfine structure of 1Σ molecules. *Computer Physics Communications*, 282, 108512. https://doi.org/10.1016/j.cpc.2022.108512

Brown, S. E. (2023). Missed opportunities? Accessibility in 'post-pandemic' academia. *Disability & Society*, *38*(7), 1271–1275. https://doi.org/10.1080/09687599.2023.2190478

Cabannes, F. (1974). Spectrometric plasma diagnostics. *Pure and Applied Chemistry*, *39*(3), 381–394. https://doi.org/10.1351/pac197439030381

Cooper, P. D. (2018). Employing Spreadsheets for Applying Calculus in Upper-Level Chemistry Courses. *Journal of Chemical Education*, *95*(10), 1890–1893. https://doi.org/10.1021/acs.jchemed.8b00193

Fasoula, S., Nikitas, P., & Pappa-Louisi, A. (2017). Teaching Simulation and Computer-Aided Separation Optimization in Liquid Chromatography by Means of Illustrative Microsoft Excel Spreadsheets. *Journal of Chemical Education*, *94*(8), 1167–1173. https://doi.org/10.1021/acs.jchemed.7b00108

Fintz, M., Shoshan, A., Bhonker, N., Kviatkovsky, I., & Medioni, G. (2021). Synthetic data for model selection. *arXiv preprint arXiv:2105.00717*, *1*. http://arxiv.org/abs/2105.00717

Flannigan, D. J. (2014). Spreadsheet-Based Program for Simulating Atomic Emission Spectra. *Journal of Chemical Education*, *91*(10), 1736–1738. https://doi.org/10.1021/ed500479u

Fowler, R. H., & Guggenheim, E. A. (1939). *Statistical thermodynamics: a version of statistical mechanics for students of physics and chemistry*. Cambridge University Press.

Hill, T. L. (2012). *An introduction to statistical thermodynamics*. Courier Corporation.

Huang, Y., Zhao, J., Qiang, Y., Hou, T., Ren, X., & Sun, H. (2021). The reform and exploration of intelligent python language teaching. In *2021 16th International Conference on Computer Science & Education (ICCSE)* (pp. 883-888). IEEE. https://doi.org/10.1109/ICCSE51940.2021.9569670

Jordheim, L. P., Denuzière, A., MacHon, C., & Zeinyeh, W. (2021). Interactive Web Application as a Teaching Tool to Introduce Basics of Chromatography and the Plate Theory. *Journal of Chemical Education*, *98*(7), 2440–2443. https://doi.org/10.1021/acs.jchemed.1c00314

Kolpaková, A., Kudrna, P., & Tichý, M. (2011). Study of plasma system by OES (optical emission spectroscopy). In *Proc. of 20th Annual Conference of Doctoral Students* (Vol. 2, pp. 180-185).

Kumar, R. (2020). Future for scientific computing using Python. *International Journal of Engineering Technologies and Management Research*, *2*(1), 30-41.

Martinez-Urreaga, J., Mira, J., & Gonzáles-Fernández, C. (2003). Introducing the stochastic simulation of chemical reactions: using the Gillespie algorithm and MATLAB. *Chemical Engineering Education*, *37*(1), 14-19.

Nikolenko, S. I. (2021). Synthetic Data for Deep Learning (Vol. 174). Springer International Publishing. https://doi.org/10.1007/978-3-030-75178-4

Ochkov, V. F., Stevens, A., & Tikhonov, A. I. (2022). Jupyter notebook, jupyterlab–integrated environment for stem education. In *2022 VI International Conference on Information Technologies in Engineering Education (Inforino)* (pp. 1-5). IEEE. https://doi.org/10.1109/Inforino53888.2022.9782924

Oinonen, N., Yakutovich, A. V., Gallardo, A., Ondráček, M., Hapala, P., & Krejčí, O. (2024). Advancing scanning probe microscopy simulations: A decade of development in probe-particle models. *Computer Physics Communications*, 305, 109341. https://doi.org/10.1016/j.cpc.2024.109341

Ojeda Misses, M. A., & Jiménez, N. J. (2021). Development of a platform with real-time performance for electrical circuits education. *IEEE Lat. Am. Trans.*, *19*(12), 2147-2155. https://doi.org/10.1109/TLA.2021.9480158

Pretz, K. (2021). Online labs give remote learners hands-on experience. *IEEE Spectrum*, *58*(6), 66–67.

Schares, E. (2022). Unsub Extender: A Python-based web application for visualizing Unsub data. *Quantitative Science Studies*, *3*(3), 600-623. https://doi.org/10.1162/qss_a_00200

Shah, S. A., Li, H., Bittner, E. R., Silva, C., & Piryatinski, A. (2023). QuDPy: A Python-based tool for computing ultrafast non-linear optical responses. *Computer Physics Communications*, 292, 108891. https://doi.org/10.1016/j.cpc.2023.108891

Vieira, A. L., Nespeca, M. G., Pavini, W. D., Ferreira, E. C., & Gomes Neto, J. A. (2019). A user-friendly excel spreadsheet for dealing with spectroscopic and chromatographic data. *Chemometrics and Intelligent Laboratory Systems*, 194. https://doi.org/10.1016/j.chemolab.2019.103816

Yu, W., Kind, M. C., & Brunner, R. J. (2017). Vizic: A Jupyter-based interactive visualization tool for astronomical catalogs. *Astronomy and computing*, *20*, 128-139. https://doi.org/10.1016/j.ascom.2017.06.004