Development and implementation of a Python functions for automated chemical reaction balancing

Pankaj Dumka[1], Rishika Chauhan[2], Dhananjay R. Mishra[1], Feroz Shaik[3], Pavithra Govindaraj[4], Abhinav Kumar[5], Chandrakant Sonawane[6], Vladimir Ivanovich Velkin[5]
[1]Department of Mechanical Engineering, Jaypee University of Engineering and Technology, Madhya Pradesh, India
[2]Department of Electronics and Communication Engineering, Jaypee University of Engineering and Technology, Madhya Pradesh, India
[3]Department of Mechanical Engineering, Prince Mohammad Bin Fahd University, Dhahran, Saudi Arabia
[4]Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bangalore, India
[5]Department of Nuclear and Renewable Energy, Ural Federal University Named After the First President of Russia Boris Yeltsin, Ekaterinburg, Russia
[6]Department of Mechanical Engineering, Symbiosis International University, Pune, India

Article Info

Article history:

Received Dec 25, 2023
Revised Feb 24, 2024
Accepted Mar 16, 2024

Keywords:

Chemical engineering
Coding
Linear algebra
Pandas
Python
Reaction balancing
SumPy

ABSTRACT

Chemical reaction balancing is a fundamental aspect of chemistry, ensuring the conservation of mass and atoms in reactions. This article introduces a specialized Python functions designed for automating the balancing of chemical reactions. Leveraging the versatility and simplicity of Python, the module employs advanced algorithms to provide an efficient and user-friendly solution for scientists, educators, and industry professionals. This article delves into the design, implementation, features, applications, and future developments of the Python functions for automated chemical reaction balancing. The functions thus developed were tested on some typical chemical reactions and the results are the same as that in the literature.

Corresponding Author:

Pankaj Dumka
Department of mechanical Engineering, Jaypee University of Engineering and Technology
Madhya Pradesh, India
Email: p.dumka.ipec@gmail.com

1. INTRODUCTION

Chemical reactions serve as the bedrock of chemical understanding, offering insights into the intricate transformations of matter. At the heart of this understanding lies the essential practice of balancing chemical reactions, ensuring fidelity to the principles of mass conservation and atomic integrity [1]. However, manual balancing can prove challenging, particularly with the complexity of advanced chemistry reactions [2]–[4]. Recognizing this challenge, there is a pressing need for specialized computer programs designed to automate and optimize the balancing process [5], [6].

Motivated by this necessity, our research endeavours to present a novel approach to chemical reaction balancing, offering a solution that is both efficient and precise [7]. We introduce a dedicated Python module engineered to automate the balancing process while prioritizing versatility, ease of integration, and robust algorithmic foundations [8]–[10]. This module fills a crucial gap left by previous efforts, by not only providing automation but also delving into the intricate details of analysis and programming.

As the landscape of chemical research continues to evolve, the demand for computational tools that streamline processes becomes increasingly pronounced. The developed Python module aims to meet this demand, serving as a sophisticated yet accessible resource for researchers, educators, and industry professionals alike. With its broad scope encompassing a diverse range of chemical reactions, the module promises applicability across various domains within the realm of chemistry.

2. PYTHON AND SOME OF ITS STANDARD MODULES

Python has emerged as one of the most popular programming languages, celebrated for it is simplicity, readability, and versatility. It is open-source nature and a vast ecosystem of libraries make it a preferred choice for diverse applications, from web development to data science [11]–[17]. Python's appeal lies in its syntax, which is designed to be clear and readable, fostering a codebase that is easy to understand and maintain. It is versatility spans across domains, making it an ideal language for beginners and seasoned developers alike [9], [18]. Python supports object-oriented, imperative, and functional programming paradigms, providing developers with flexibility in their coding approaches. In this note, a brief skim over into the significance of Python programming has been done to explore four essential modules-SymPy, re, and Pandas-that contribute to it is widespread adoption.

2.1. SymPy: Symbolic mathematics

SymPy facilitates symbolic computation by representing mathematical objects as symbolic expressions. Variables, equations, and mathematical operations are manipulated symbolically rather than numerically, allowing for exact and precise results. This is particularly useful in scenarios where maintaining the symbolic representation of mathematical expressions is essential, such as in algebraic simplifications, calculus, and solving equations symbolically [19]–[21].

2.2. re: Regular expressions in Python

The 're' module in Python stands for regular expressions, a powerful tool for pattern matching and string manipulation. Regular expressions allow developers to search, match, and manipulate strings based on specified patterns. This module is invaluable for tasks such as data cleaning, text parsing, and pattern recognition. Python's 're' module facilitates the use of regular expressions, enabling developers to handle complex string operations with ease [22].

2.3. Pandas: Data manipulation made easy

Pandas is a high-level data manipulation library that simplifies working with structured data. It provides data structures like DataFrames and Series, which are intuitive and powerful for handling and analyzing tabular data. Pandas seamlessly integrates with NumPy, allowing for efficient data manipulation and analysis [12]. Whether it is cleaning messy data, aggregating information, or performing complex operations on datasets, Pandas is an indispensable tool in the data scientist's arsenal.

3. METHODOLOGY OF BALANCING ALGORITHM

The core algorithm of the Python module is based on linear algebraic principles [23]. It transforms an unbalanced chemical reaction into a system of linear equations [24]–[26], solving for the coefficients that achieve mass and atom balance. This algorithm is optimized for both efficiency and accuracy, making it suitable for reactions of varying complexities. The algorithm is explained with the help of a simple example as mentioned below. Consider a simple oxidation reaction as (1).

$$\alpha\, C_2H_4 + \beta\, O_2 \rightarrow \gamma\, CO_2 + \delta\, H_2O \tag{1}$$

Now the task is to evaluate the unknowns α, β, γ, and δ. Here first the different elements involved in the chemical reaction are identified viz. C, H, and O. Now as there are 3 elements, so the number of each element in different compounds are written in the vector form as follows:

$$C_2H_4 \rightarrow \begin{bmatrix} C \\ H \\ O \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix}$$

$$O_2 \rightarrow \begin{bmatrix} C \\ H \\ O \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

$$CO_2 \rightarrow \begin{bmatrix} C \\ H \\ O \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

$$H_2O \rightarrow \begin{bmatrix} C \\ H \\ O \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

now the (1) can be written in the vector form as:

$$\alpha \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \rightarrow \gamma \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} + \delta \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \quad (2)$$

by bringing the terms from right side to the left side, the rearranged from of the (2) becomes:

$$\boldsymbol{\alpha} \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix} + \boldsymbol{\beta} \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} - \boldsymbol{\gamma} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - \boldsymbol{\delta} \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3)$$

finally in the matrix form the (3) takes the shape of (4).

$$\begin{bmatrix} 2 & 0 & 1 & 0 \\ 4 & 0 & 0 & 2 \\ 0 & 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ -\gamma \\ -\delta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (4)$$

As shown in (4) is a system of homogeneous linear equation of the form: $Ax = 0$. The task is to find the null space of matrix [23], [27]. So, the matrix has to be reduced into row reduced echelon form (RREF). After reduction to RREF the (4) becomes:

$$\begin{bmatrix} 1 & 0 & 0 & 1/2 \\ 0 & 1 & 0 & 3/2 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ -\gamma \\ -\delta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (5)$$

as shown in (5) has three pivot columns (marked red in colour). To start the solution an initial guess for δ is required which has been taken as 1. Thereafter, on solving the system backward the values of the unknowns will come out to be: $\gamma = 1, \beta = 3/2$, and $\alpha = 1/2$. Hence, the balanced chemical equation will be as (6).

$$(1/2)\ C_2H_4 + (3/2)\ O_2 \rightarrow (1)\ CO_2 + (1)\ H_2O \quad (6)$$

This algebraic method is very powerful but the complexity increases to a great extent for the equations which are having large number of terms. Therefore, the use of programming for the to automate the task of chemical reaction balancing becomes so much important.

4. METHOD TO DEVELOP PYTHON MODULE

On the basis of the computation procedure explained in the previous section the algorithm to balance the chemical reaction can be layout as follows: i) first the elements are identified; ii) then the counting must be done that how many elements are there in each compound of the equation; iii) then they are arranged in a matrix form; iv) convert the matrix into row reduced echelon form; and v) find the null space.

To perform these tasks two algoritms are developed, the Algoritms 1 and Algoritms 2 are as follows:

Algoritm 1

```
def count_elements(elements,chemical_formula):
    # Regular expression to match element symbols and their counts
    pattern=compile(r'([A-Z][a-z]*)(\d*)')

    # Dictionary to store element counts
    element_counts={}

    # Find all matches in the chemical formula
```

```
    matches=pattern.findall(chemical_formula)

    # Loop through matches and update element counts
    for match in matches:
        element_symbol, count_str=match
        count=int(count_str) if count_str else 1
        element_counts[element_symbol]=element_counts.get(element_symbol, 0) +
count

    # Making list of numbers according to the element list
    lst=[]
    for i in elements:
        if i in element_counts.keys():
            lst.append(element_counts[i])
        else:
            lst.append(0)
    return element_counts,lst
```

Algoritm 2

```
def Reaction_coefficients(elements, list_of_compounds):
    Mat=[]
    for i in list_of_compounds:
        Mat.append(count_elements(elements,i)[1])

    M=Matrix(Mat)
    M=M.transpose()
    M_rref=M.rref()

    # No. of pivots
    n_p=len(M_rref[1])

    # Null space
    x_n=M_rref[0][:n_p,-1]

    # appending 1 as last element
    a=list(x_n)
    a.append(-1)
    a=Matrix(a)

    ch_cm=list_of_compounds
    data_main={"Ch. composition":ch_cm,"coefficient":list(a)}
    df=DataFrame(data_main)
    return df
```

Explanation of Algoritm 1:

- Inputs:
  - (a) ‘elements’: A list of element symbols.
  - (b) ‘chemical_formula’: The chemical formula for which element counts need to be determined.
- Processing steps:
  - (a) Utilizes a regular expression (‘compile(r'([A-Z][a-z]*)(\d*)')’) to match element symbols and their counts in the chemical formula.
  - (b) Initializes an empty dictionary (‘element_counts’) to store the counts of each element.
  - (c) Finds all matches in the chemical formula using the defined pattern, resulting in a list of tuples (‘matches’).
  - (d) Iterates through the matches, extracting element symbols and counts, converting count strings to integers (or defaulting to 1 if no count is provided), and updating the ‘element_counts’ dictionary accordingly.
- List generation:
  - (a) Initializes an empty list (‘lst’) to store counts of elements in the order specified by the input list of elements.
  - (b) Iterates through the input list of elements, appending the corresponding counts from the ‘element_counts’ dictionary to the list (‘lst’). If an element is not present in the chemical formula, appends 0 for that element.
- Outputs:

  Returns a tuple containing:
  - (a) ‘element_counts’: A dictionary with element symbols as keys and their counts in the chemical formula as values.

(b) 'lst': A list representing the counts of elements in the order specified by the input list of elements.

Explanation of Algoritm 2:

- Inputs:
  (a) 'elements': A list of element symbols.
  (b) 'list_of_compounds': A list of chemical compounds for which reaction coefficients need to be determined.
- Processing steps:
  (a) Initializes an empty matrix ('Mat') to store the counts of elements in each compound using the 'count_elements' function.
  (b) Iterates through the 'list_of_compounds', using the 'count_elements' function to obtain the list of element counts for each compound and appends it to the matrix 'Mat'.
- Matrix operations:
  (a) Creates a matrix ('M') from the obtained 'Mat'.
  (b) Transposes the matrix ('M') to facilitate further operations.
  (c) Computes the row reduced echelon form of the transposed matrix ('M') using the 'rref' method.
  (d) Determines the number of pivots in the reduced row-echelon form, denoted as 'n_p'.
- Null space calculation:
  (a) Extracts the null space of the matrix, specifically the last column, representing the coefficients of the compounds in the balanced chemical reaction.
  (b) Appends '-1' to the null space vector, creating a list ('a').
  (c) Converts the list to a matrix ('a') for further processing.
- Data frame creation:
  (a) Constructs a DataFrame ('df') containing two columns:
  (b) "Ch. composition": List of chemical compounds ('list_of_compounds').
  (c) "Coefficient": Reaction coefficients corresponding to each compound.
- Output:
  (a) Returns the DataFrame ('df') containing the chemical compositions of the compounds and their corresponding reaction coefficients in a balanced chemical reaction.

These functions provide a straightforward and intuitive interface for users to integrate into their Python scripts or applications. Users can input chemical reactions in a human-readable format, and the module automatically balances them, returning the balancing coefficients as results. The point to be noted here is that the coefficients on the left side of the equation will come positive whereas the one on the right side of the equation will come as negative.

5. RESULTS AND DISCUSSION ON THE ASSESSMENT OF DEVELOPED PYTHON FUNCTIONS

The Python functions for automated chemical reaction balancing has broad applications across academic, research, and industrial domains. Its seamless integration into computational workflows makes it a valuable resource for researchers seeking to automate repetitive tasks. Educators can incorporate the module into their teaching materials to enhance students' understanding of reaction balancing principles. Additionally, industry professionals can leverage the module for process optimization, ensuring the efficient use of resources and maintaining the quality of chemical processes. The module undergoes a rigorous performance evaluation to assess its capabilities across various scenarios. Benchmarking is conducted against known chemical reactions, ranging from simple to complex cases. The results not only demonstrate the efficiency and accuracy of the module but also provide insights into its limitations and potential areas for improvement. The steps to be adopted for the effective utilization of functions are as follows:

- First, the SymPy, Pandas, and Re modules are imported.
- Second, the elements are identified and placed in a list of characters (called as elements).
- Third the chemical formulas list is created as list of strings (called as list_of_compounds). Important thing to note here is that the compounds are written from left to right as they appear in the chemical reaction.

Below are some of the examples which shows the use of functions developed in the previous sections.

Example 1: Balance the following chemical reaction.

$$H_3PO_4 + KOH \rightarrow K_3PO_4 + H_2O$$

Solution: The program to balance the above equation along with its solution is shown in Table 1.

Therefore, the balanced reaction will be:

$$(1/3)H_3PO_4 + (1)KOH \rightarrow (1/3)K_3PO_4 + (1)H_2O$$

Example 2: Balance the following chemical reaction.

$$K + B_2O_3 \rightarrow K_2O + B$$

Solution: Table 2 displays the program used to balance the above equation, along with its solution. Therefore, the balanced reaction will be:

$$(3)K + (1/2)B_2O_3 \rightarrow (3/2)K_2O + (1)B$$

Example 3: Balance the following chemical reaction.

$$H_3PO_4 + Mg(OH)_2 \rightarrow Mg_3(PO_4)_2 + H_2O$$

Solution: Here the point to be noted is that as brackets are not permitted in the list of compounds so we have to write the expanded forms i.e. $Mg(OH)_2$ is written as MgO2H2 and $Mg_3(PO_4)_2$ is written as Mg3P2O8. The same philosophy will be followed in the subsequent example as well. Table 3 exhibits the program used to balance the mentioned equation, along with its corresponding solution.

Table 1. Program and solution for example 1

Code:

```
elements=['H','P','O','K']

list_of_compounds=['H3PO4','KOH', 'K3PO4','H2O']

Reaction_coefficients(elements, list_of_compounds)
```

Output:

	Ch. composition	coefficient
0	H3PO4	1/3
1	KOH	1
2	K3PO4	-1/3
3	H2O	-1

Table 2. Program and solution for balancing the equation (example 2)

Code:

```
elements=['K', 'B', 'O']

list_of_compounds=['K','B2O3','K2O','B']

Reaction_coefficients(elements, list_of_compounds)
```

Output:

	Ch. compostion	coefficient
0	K	3
1	B2O3	1/2
2	K2O	-3/2
3	B	-1

Table 3. Balancing program and solution overview for example 3

Code:

```
elements= ['H', 'P', 'O', 'Mg']

list_of_compounds=['H3PO4','MgO2H2','Mg3P2O8','H2O']

Reaction_coefficients(elements, list_of_compounds)
```

Output:

	Ch. composition	coefficient
0	H3PO4	1/3
1	MgO2H2	1/2
2	Mg3P2O8	-1/6
3	H2O	-1

Therefore, the balanced reaction will be:

$$(1/3)H_3PO_4 + (1/2)Mg(OH)_2 \rightarrow (1/6)Mg_3(PO_4)_2 + (1)H_2O$$

Example 4: Balance the following chemical reaction.

$$Ca_3(PO_4)_2 + SiO_2 + C \rightarrow CaSiO_3 + CO + P$$

Solution: Refer to Table 4 for the program and solution related to balancing the above equation.

Table 4. Program and solution reference for equation balancing (Example 4)

Code	Output

`elements= ['Ca', 'P', 'O', 'Si', 'C']`
`list_of_compounds=['Ca3P2O8','SiO2','C','CaSiO3','CO','P']`
`Reaction_coefficients(elements, list_of_compounds)`
	<table><tr><th></th><th>Ch. composition</th><th>coefficient</th></tr><tr><td>0</td><td>Ca3P2O8</td><td>1/2</td></tr><tr><td>1</td><td>SiO2</td><td>3/2</td></tr><tr><td>2</td><td>C</td><td>5/2</td></tr><tr><td>3</td><td>CaSiO3</td><td>-3/2</td></tr><tr><td>4</td><td>CO</td><td>-5/2</td></tr><tr><td>5</td><td>P</td><td>-1</td></tr></table>

Therefore, the balanced reaction will be:

$$(1/2)Ca_3(PO_4)_2 + (3/2)SiO_2 + (5/2)C \rightarrow (3/2)CaSiO_3 + (5/2)CO + (1)P$$

these examples demonstrate the effectiveness of the Python functions in balancing chemical reactions across a range of complexities. The balanced reactions, along with their corresponding coefficients, are provided as solutions, facilitating accurate and efficient reaction balancing.

6. CONCLUSION

In conclusion, chemical reaction balancing, a foundational aspect of chemistry ensuring mass and atom conservation, has been automated through dedicated Python functions. This article introduces these functions, leveraging Python's versatility and simplicity. Advanced algorithms provide an efficient and user-friendly solution for scientists, educators, and industry professionals. The article covers design, implementation, features, and applications of the Python functions. Tested on typical reactions, results align with literature, validating accuracy. Looking forward, continuous refinement, expanded capabilities, and integration with other Python libraries promise ongoing advancements. This automation represents a significant stride in modernizing and simplifying chemistry, catering to current needs, and laying the foundation for future innovations within the dynamic landscape of chemical sciences.

REFERENCES

[1] D. J. Higham, "Modeling and simulating chemical reactions," *SIAM Review*, vol. 50, no. 2, pp. 347–368, Jan. 2008, doi: 10.1137/060666457.

[2] A. L. Chandrasegaran, D. F. Treagust, B. G. Waldrip, and A. Chandrasegaran, "Students' dilemmas in reaction stoichiometry problem solving: deducing the limiting reagent in chemical reactions," *Chem. Educ. Res. Pract.*, vol. 10, no. 1, pp. 14–23, 2009, doi: 10.1039/B901456J.

[3] S. Kar, N. Senapati, and B. K. Swain, "Effect of chemical reaction on MHD flow with heat and mass transfer past a vertical porousplate in the presence of viscous dissipation," *International Journal of Advances in Applied Sciences*, vol. 8, no. 1, pp. 83–94, Mar. 2019, doi: 10.11591/ijaas.v8.i1.pp83-94.

[4] A. L. M. Lewis and G. M. Bodner, "Chemical reactions: what understanding do students with blindness develop?," *Chem. Educ. Res. Pract.*, vol. 14, no. 4, pp. 625–636, 2013, doi: 10.1039/C3RP00109A.

[5] I. B. Risteski, "A new generalized algebra for the balancing of ℵ chemical reactions," *Materials and Technologies*, vol. 48, no. 2, pp. 215–219, 2014.

[6] A. Setiawan, S. Rostianingsih, and T. R. Widodo, "Augmented reality application for chemical bonding based on android," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 445–451, Feb. 2019, doi: 10.11591/ijece.v9i1.pp445-451.

[7] D. A. Johar, "Application of the concept of linear equation systems in balancing chemical reaction equations," *International Journal of Global Operations Research*, vol. 1, no. 4, pp. 130–135, Nov. 2020, doi: 10.47194/ijgor.v1i4.48.

[8] P. Dumka, K. Rana, S. P. S. Tomar, P. S. Pawar, and D. R. Mishra, "Modelling air standard thermodynamic cycles using python," *Advances in Engineering Software*, vol. 172, Oct. 2022, doi: 10.1016/j.advengsoft.2022.103186.

[9] P. Dumka, R. Chauhan, A. Singh, G. Singh, and D. Mishra, "Implementation of Buckingham's Pi theorem using Python," *Advances in Engineering Software*, vol. 173, Nov. 2022, doi: 10.1016/j.advengsoft.2022.103232.

[10] D. Nofriansyah and H. Freizello, "Python application: Visual approach of hopfield discrete method for hiragana images recognition," *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 4, pp. 609–614, Dec. 2018, doi: 10.11591/eei.v7i4.691.

[11] M. F. Sanner, "Python: A programming language for software integration and development," *Journal of Molecular Graphics and Modelling*, vol. 17, no. 1, pp. 57–61, 1999.

[12] C. Führer, J. E. Solem, and O. Verdier, *Scientific computing with Python - Second Edition: High-performance scientific computing with NumPy, SciPy, and pandas*. Packt Publishing; 2nd ed. edition, 2021.

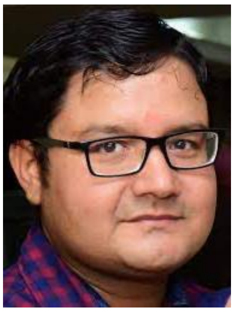
[13] A. Saha, *Doing Math with Python: Use programming to explore algebra, statistics, calculus, and more*. No Starch Press; 1st edition, 2015.

[14] E. Shein, "Python for beginners," *Communications of the ACM*, vol. 58, no. 3, pp. 19–21, Feb. 2015, doi: 10.1145/2716560.

[15] D. R. M. P. Dumka and R. Dumka, *Numerical methods using Python (for scientists and engineers)*. Bluerose Publishers Pvt. Ltd.; First Edition, 2022.

[16] G. Moruzzi, "Python basics and the interactive mode," in *Essential Python for the Physicist*, Cham: Springer International Publishing, 2020, pp. 1–39. doi: 10.1007/978-3-030-45027-4_1.
[17] R. Johansson, *Numerical Python: Scientific computing and data science applications with numpy, SciPy and matplotlib*. Berkeley, CA: Apress, 2019. doi: 10.1007/978-1-4842-4246-9.
[18] P. Dumka and D. R. Mishra, "Understanding the TDMA/thomas algorithm and its implementation in Python," *International Journal of All Research Education and Scientific Methods (IJARESM)*, vol. 10, no. 10, pp. 998–1002, 2022.
[19] M. Cywiak and D. Cywiak, "SymPy," in *Multi-Platform Graphics Programming with Kivy*, Berkeley, CA: Apress, 2021, pp. 173–190. doi: 10.1007/978-1-4842-7113-1_11.
[20] A. Meurer *et al.*, "SymPy: symbolic computing in Python," *PeerJ Computer Science*, vol. 3, Jan. 2017, doi: 10.7717/peerj-cs.103.
[21] M. Rocklin and A. R. Terrel, "Symbolic statistics with SymPy," *Computing in Science and Engineering*, vol. 14, no. 3, pp. 88–93, May 2012, doi: 10.1109/MCSE.2012.56.
[22] C. Chapman and K. T. Stolee, "Exploring regular expression usage and context in Python," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, Jul. 2016, pp. 282–293. doi: 10.1145/2931037.2931073.
[23] C. Börgers, *Introduction to numerical linear algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2022. doi: 10.1137/1.9781611976922.
[24] R. H. Abdul Rahim, A. Baharum, and H. Hijazi, "Evaluation on effectiveness of learning linear algebra using gamification," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 2, pp. 997–1004, Feb. 2020, doi: 10.11591/ijeecs.v17.i2.pp997-1004.
[25] S. Majumdar and S. Chakraverty, "Method for the solution of interval systems of linear equations," *International Journal of Advances in Applied Sciences (IJAAS)*, vol. 2, no. 2, pp. 73–76, 2013.
[26] J. Schrier, "Solution mixing calculations as a geometry, linear algebra, and convex analysis problem," *Journal of Chemical Education*, vol. 98, no. 5, pp. 1659–1666, May 2021, doi: 10.1021/acs.jchemed.0c01456.
[27] M. F. Sardella, E. Serrano, O. Camacho, and G. Scaglia, "Improvement of linear algebra controllers using sliding surface concepts: Applications to chemical processes," *IEEE Latin America Transactions*, vol. 19, no. 8, pp. 1299–1306, Aug. 2021, doi: 10.1109/TLA.2021.9475860.

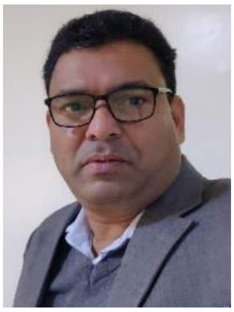
BIOGRAPHIES OF AUTHORS



Pankaj Dumka has completed his B. Tech. (Mechanical Engineering) from Inderprastha Engineering College, Ghaziabad in 2007, M. Tech. from Indian Institute of Technology, Kanpur in "Fluids and Thermal Sciences" in 2010, and Ph.D. from Jaypee University of Engineering and Technology, Guna in 2021. He has been working with the Jaypee University of Engineering and Technology as an assistant professor in the Department of Mechanical Engineering since 2011. He has published more than 40 research articles in reputed SCI and SCOPUS indexed Journals. His area of interest includes thermodynamics, fluid dynamics, computational fluid dynamics, numerical computations, Python programming, and solar water desalination. He can be contacted at email: p.dumka.ipec@gmail.com.

Rishika Chauhan has been working with the Jaypee University of Engineering and Technology as an assistant professor in the Department of Electronics and Communication Engineering since 2011. She teaches both undergraduate and postgraduate students of engineering at the University. Her area of interest includes wireless communication, numerical computations, artificial neural networks, and discrete mathematics. Mrs. Chauhan has published number of research articles in numerous journals of national and international repute. She can be contacted at email: rishikaster@gmail.com.

Dhananjay R. Mishra has obtained his bachelor's degree (Mechanical Engineering) from Amravati University (M.S.), Master's degree (Production Engineering) in 2007 from BIT, Durg of Pt. Ravi Shankar Shukla University, Raipur (C.G.), and Doctor of Philosophy in Mechanical Engineering from National Institute of Technology Raipur, Raipur, on August 2016. He is working as an Assistant Professor (SG) in the Mechanical Engineering Department of Jaypee University of Engineering and Technology, Guna (M.P.). He has published more than 125 research articles in peer-reviewed international, national journals, and conferences. His area of interest includes solar thermal applications, renewable energy, python programming, numerical computations, internal combustion engines, and solar water desalination. He can be contacted at email: dm30680@gmail.com.

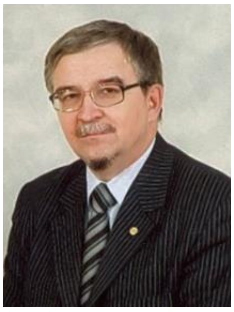


Feroz Shaik is currently working at Prince Mohammed bin Fahd University, Kingdom of Saudi Arabia. Dr. Feroz obtained his doctorate in the field of Chemical Engineering from Andhra University, India in 2004 and Post Doc Research Fellow from Leibniz University, Germany in 2015. M.Tech in Chemical Engineering from Osmania University, India in 1998 and B. Tech in Chemical Engineering from S. V. University, India in 1992. Post Graduate Diploma in Environmental Studies from Andhra University in 2003. Dr. Feroz Shaik is the visiting professor for School of Renewable Energy, Maejo University, Thailand. Dr. Feroz has more than 260 publications to his credit in journals and conferences of international repute. He can be contacted at email: fshaik@pmu.edu.sa.

Pavithra Govindaraj is having 18 years of experience in academics, currently working as Associate Professor in ECE Dept. of Dayananda Sagar College of Engg., Bangalore and has published more than 500+ research papers in various National, International Journals (Scopus Indexed, SCI, WoS, UGC, Impact Factors) and various Conferences in India and Abroad. She has also published 25 textbooks and has got 160+ Indian and Foreign patents (some filed, published, granted). She has got 5 National Awards for her research and academic and was a resource person for more than one dozen workshops, tech talks, conferences, seminars, keynotes, and FDPs. She has got 5 funded projects worth Rs. 40 Lakhs. She has also conducted more than 20 events for students and staffs. She can be contacted at email: dr.pavithrag.8984@gmail.com.

Abhinav Kumar is currently working as Scientific Officer in the Department of Nuclear and Renewable Energy at Ural Federal University, Russia. Dr. Abhinav Kumar is working in energy systems research from last 7 years where he has worked on solar PV energy, Parabolic Trough Solar collector, advanced solar cell materials, battery storage and superconducting magnetic energy systems. He has admirable skills on computational software including Comsol, and Ansys. Has shown his competence in publishing high quality articles and has a skill to present his research at international forums. Currently he is associated with international groups in performing research on solar PV assisted battery storage systems. He can be contacted at email: drabhinav@ieee.org.

Chandrakant Sonawane is currently working as Professor in Mechanical Engineering Department at Symbiosis Institute of Technology, (SIU), Pune, India. He has published more than 70+ research articles in reputed international Journals. His areas of interests include solar energy, IC engines, machine learning, and IoT. He can be contacted at email: crsonawane@gmail.com.



Vladimir Ivanovich Velkin is currently working as Professor in the Department of Nuclear and Renewable Energy at Ural Federal University, Russia. Dr. Velkin is having a teaching experience of 40 years. His area of expertise is Physics of nuclear reactors, Renewable energy and Disposal of radioactive waste. He has published more than 200 articles in SCI, WoS and other. Currently he is associated in solar PV assisted battery storage systems research. He can be contacted at email: v.i.velkin@urfu.ru.