***Original Research Article***

**DESIGN AND IMPLEMENTATION OF THE GRAM-SCHMIDT ORTHOGONALIZATION METHOD IN SOBOLEV SPACE**

**Abstract**

This article is based on the design and implementation of the Gram-Schmidt orthogonalization method in the Sobolev space However, through computational processes, it becomes clear that the Gram-Schmidt orthogonalization algorithm in aims to transform a set of functions into an orthogonal set. By considering an arbitrary basis of a subspace of functions in the space, we can construct a new orthogonal basis. However, this method presents certain complexities and may be prone to errors due to the tedious calculation of inner products and norms of the functions. This complexity can lead to an accumulation of errors during the orthogonalization process, thereby compromising the accuracy of the results obtained. The motivation behind the development of the new implementation method is based on the need to reduce the maximum computation time and optimize precision while minimizing the risk of errors during critical steps. Consequently, the improved new approach can not only facilitate the use of this method but also ensure reliable and efficient results in practical applications.

**Keywords:** Sobolev space , dot product in norms in , partial derivatives in the sense of distributions, implementation, Algorithm, Lebesgue spaces and distribution spaces D'(Ω)

**Introduction**

The Sobolev space is defined by:

where denotes the partial derivatives of the function f in the distributional sense, is fundamental in analysis, because it allows to study the regularity of functions by including those which, in addition to being integrable, have weak integrable derivatives, which is essential for solving problems of partial differential equations; moreover, the applicability of this space is reinforced by the Gram-Schmidt orthogonalization process, which facilitates the construction of orthogonal and orthonormal basis in , thus improving the analysis and resolution of this complex analytical problem. However, it is worth noting that we are working with general second-degree functions in two variables denoted: and the rectangular integration domain of the form . The Gram-Schmidt orthogonalization algorithm in digital spaces is as follows:

Let us construct the orthogonal basis , we have:



The same algorithmic steps of orthogonalization of these numerical spaces can be applied in functional spaces, and more particularly in this manuscript we treat the case of Sobolev space .

Implementation of the Gram-Schmidt orthogonalization method in Sobolev space Using the Python language has many advantages. First, it allows for efficient handling of orthogonality calculations, thus ensuring more accurate results. In addition, this approach improves the numerical stability of solutions, which is essential in practical applications. Integration with scientific computing libraries, such as NumPy and SciPy, significantly facilitates development. In addition, Python's clear syntax simplifies the debugging process, making the code more accessible. Finally, integrated visualization tools allow for intuitive analysis and interpretation of results, enhancing the user experience. This method therefore represents a significant advancement in the field of functional analysis. **[1],[2],[3],[4]**

1. **Sobolev spaces**
   1. **Definitions and properties**

**Definition 1.1.1.**

Let be an open set of . We define the Sobolev space by:

With the partial derivative of the function taken in the sense of distributions defined by **[5],[6],[7]** :

**Definition 1.1.2.**

The scalar product, noted on is defined by:

**Definition 1.1.3.**

The standard noted on is defined by:

This norm combines both the norm of the function and those of its derivatives. **[8],[9],[10],[11]**

**Proposition 1.1.4.**

The pair where is a vector space and is a scalar product on , is a pre-Hilbert space.

**Proposition 1.1.5.**

The pair where is a vector space and is a norm on is a normed space. **[12],[13],[14],[15]**

**Proposition 1.1.6.**

The space is a Hilbert space.

* 1. **Orthogonality**

**Definition 1.2.1.**

Let be a pre-Hilbert space. The functions are said to be orthogonal or orthogonal to if and only if .

This orthogonality relation is symmetric, that is to say that if is orthogonal to , then **[7],[8],[9],[16] :**

**Definition 1.2.2.**

Let be a subset of the space , the orthogonal supplement of denoted is the set of all functions of which are orthogonal to any function .

In particular, for all we have [11],[12] :

**Proposition 1.2.3.**

Let be a subset of . Then is the direct sum of and . That is:

**Definition 1.2.4.** *(Orthogonal family)*

A family of functions of is said to be orthogonal if any two functions of are orthogonal. **[17],[18],[19],[20]**

That's to say :

is said to be orthogonal if:

**Proposition 1.2.5.**

Let be an orthogonal family of non-zero functions of . Then is linearly independent. **[21],[22],[23]**

* 1. **Gram-Schmidt orthogonality process.**

**Definition 1.3.1.**

Let be an arbitrary basis of the pre-Hilbert space , where each function belongs to the Sobolev space . **[24],[25],[26]** The goal of the process is to construct an orthogonal basis in , i.e. the functions must satisfy the following condition:

**Definition 1.3.2.** *(Steps of the orthogonalization process)*

Let be any basis of . We can construct an orthogonal basis in as follows **[27],[28]** :

1. Initialization:
2. Heredity:

.

.

.

**Numerical example 1.3.3.**

Consider the subspace of , where , , and .

Find an orthogonal basis of .

**Solution**

1. Checking that functions belong to .

All these functions and their derivatives are integrable on , so they belong to .

1. Calculation of scalar products
2. Construction of the orthogonal basis using the Gram-Schmidt process

Let be a basis orthogonal to .

SO .

**2. Design and implementation.**

**2.1. Design of the Gram-Schmidt orthogonalization method algorithm.**

The design of the Gram-Schmidt orthogonalization method algorithm is based on the idea of transforming a set of linearly independent vectors into a set of orthogonal vectors while preserving their spanned space. **[29],[30]**

**2.1.1 Some concepts**

1. **Algorithm** .

Is a finite sequence of well-defined and ordered instructions or operations, used to solve a problem or perform a specific task. **[31] The Gram-Schmidt** algorithm describes the steps to follow to orthogonality a set of vectors in a vector space.

1. **Program**

Is a set of instructions written in a specific programming language that allows a computer to perform tasks or solve a problem by following the steps defined by an algorithm. In Python, it implements the Gram-Schmidt algorithm to orthogonalize a set of vectors.

1. **Variable**

Is a named memory space that stores a value. It can change during program execution.

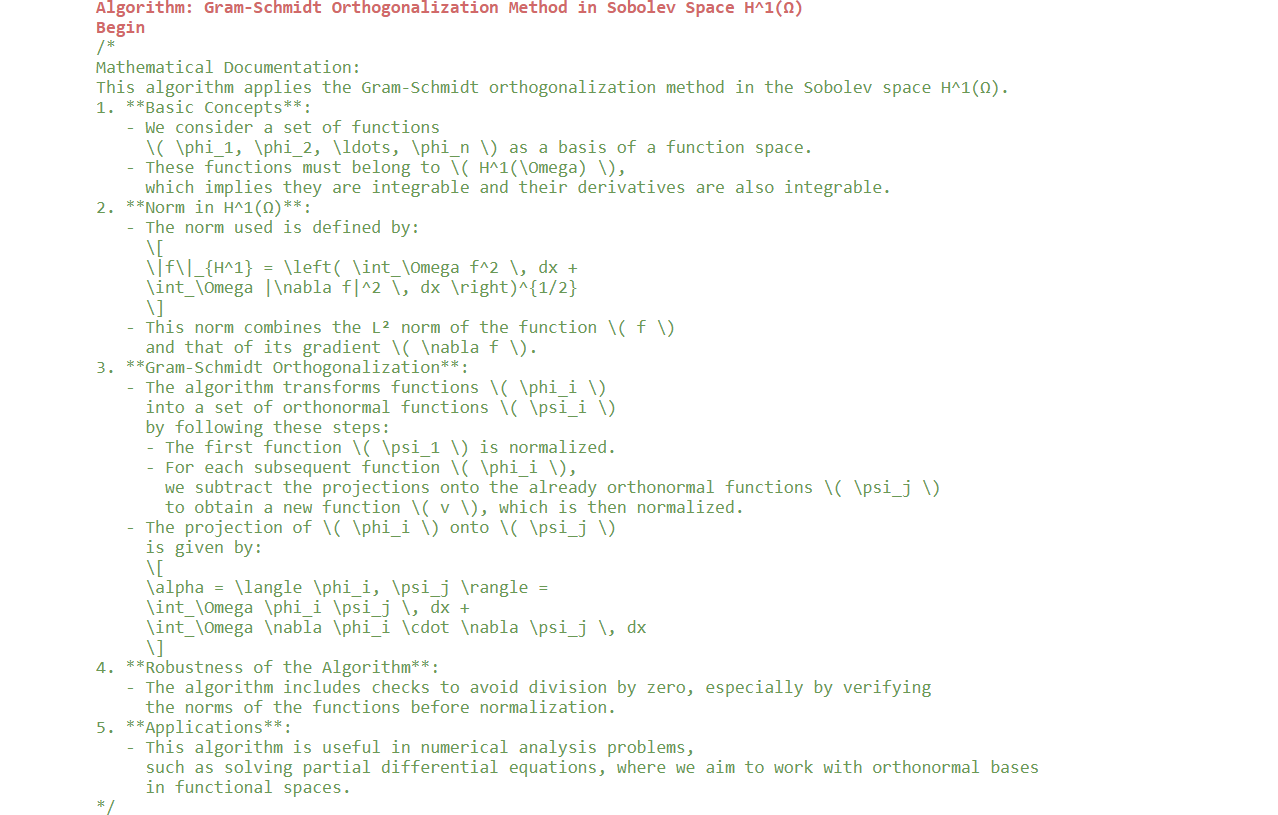
* **Presentation of design variables**

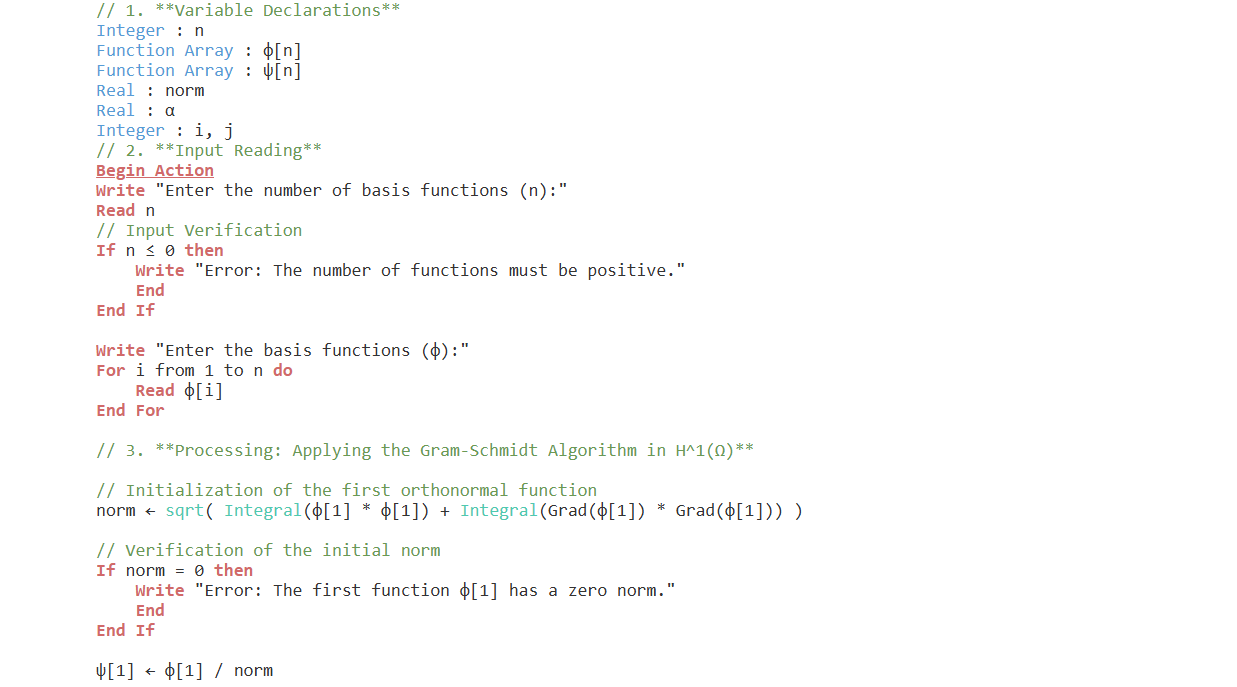
The table below contains all the variables used in our algorithm. Each variable was selected based on the needs of our algorithm and is adaptable to the Gram-Schmidt orthogonalization method in Sobolev space.

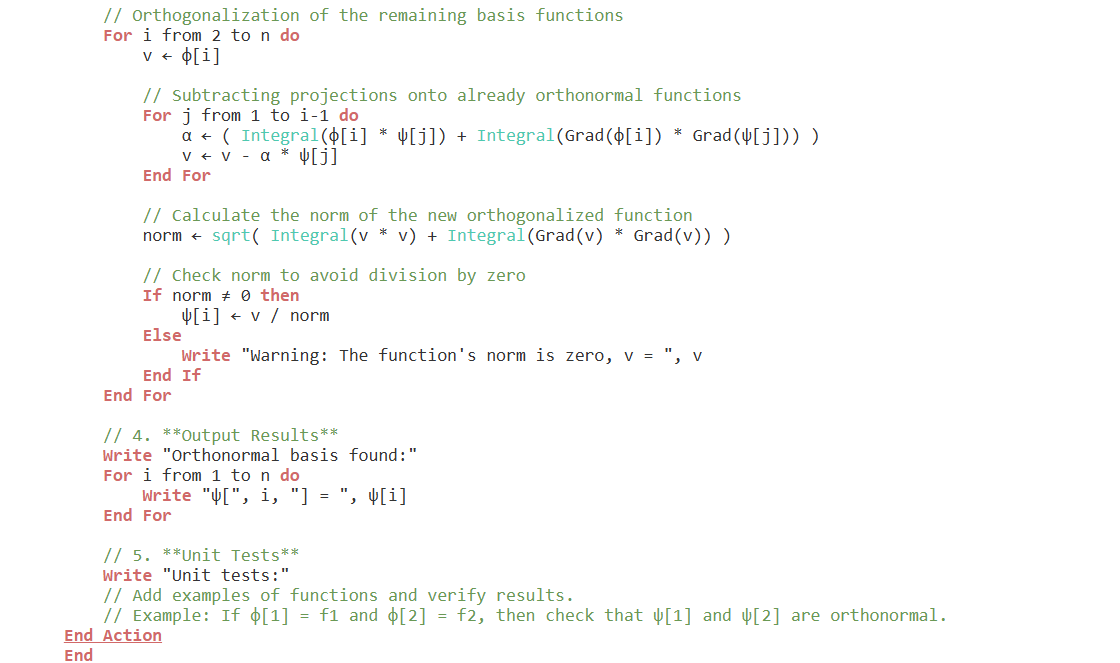
Table : List of variables used in the design of the algorithm

|  |  |  |
| --- | --- | --- |
| **Variables** | **Variable Types** | **Description of variables** |
| **n** | **Entire** | **Number of basic functions** |
| **φ[n]** | **Function Table** | **Initial base in** |
| **ψ[n]** | **Function Table** | **Orthonormal basis in** |
| **standard** | **Real** | **Standard in** |
| **alpha** | **Real** | **Projection coefficients** |
| **i, j** | **Entire** | **Loop indices** |
| **v** | **Real** | **Vector** |

**2.1.2. Proposal of the algorithm of the Gram-Schmidt orthogonalization method in Sobolev space**







* **The objective of the proposed algorithm**

The main objective of this algorithm of the Gram-Schmidt orthogonalization method in Sobolev space is to transform a set of basis functions, potentially linearly dependent, into a set of orthonormal functions. This ensures that the resulting functions are both mutually orthogonal and normalized, which facilitates their use in various mathematical and numerical contexts, such as solving partial differential equations.

This algorithm aims to:

* **Ensuring Orthogonality** : By eliminating redundancies between basis functions, the algorithm produces a set where each function is perpendicular to the others within the norm defined by the Sobolev space.
* **Normalize Functions** : By dividing each function by its norm, the algorithm ensures that every function in the resulting basis has unit norm, thus facilitating subsequent calculations.
* **Facilitating Numerical Applications** : Orthonormal bases are particularly useful in numerical methods, as they simplify approximation and integration calculations, while ensuring the stability and convergence of the methods used in numerical analysis.
* **Complexity of the Algorithm [32],[33],[34]**

The proposed Gram-Schmidt orthogonalization algorithm in Sobolev space has a complexity that can be analyzed in terms of time and space required to execute the different steps.

* 1. **Reading the entries**

The reading time of the basis functions is proportional to , where is the number of functions. This requires operations.

* 1. **Initialization and Verification**

Initializing the first orthonormal function involves computing the norm of , which requires integrals. The cost of integrals can vary depending on the evaluation method, but we can consider this operation as being for computing the norm, provided that the evaluation of the integrals is optimized.

* 1. **Orthogonalization of Functions**

The key step of the algorithm is the orthogonalization process. For each function (where ranges from 2 to ), we must:

* **Project** on all already orthonormal functions (For from 1 to ). This requires projections for each .
* **Calculate the norm** of the orthogonally fitted function after each screening.

The total cost for this step is:

Total cost =

Each projection also involves the computation of integrals, which can increase runtime depending on the method of evaluating the integrals used. If the integrals are evaluated efficiently, this can be considered by projection.

* 1. **Displaying Results**

Displaying the results simply involves looping through the functions , which is .

The time complexity of the proposed Gram-Schmidt orthogonalization algorithm is dominated by the orthogonalization phase, leading to an overall complexity of:

* **Difference between classical Gram-Schmidt algorithm and algorithm on proposed Gram-Schmidt orthogonalization method in Sobolev space** .

**The table below shows the difference between the classical Gram-Schmidt algorithm and the proposed Gram-Schmidt algorithm.**

Table 2: Overview of the **classical Gram-Schmidt algorithm and the algorithm on the proposed Gram-Schmidt orthogonalization method in Sobolev space .**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Characteristic** | **Classical Algorithm** | **Proposed Algorithm** |
| 1 | Functions Considered | Functions only | Functions and their gradients |
| 2 | Standard Used | Standard Euclidean norm | Combined norm of functions and their gradients |
| 3 | Derivatives Management | N / A | Integrates derivatives into calculations |
| 4 | Error Checking | No explicit check for division by zero | Check to avoid division by zero |
| 5 | Application | Generally in classical function spaces | Specifically in Sobolev space . |
| 6 | Robustness | Perhaps less robust in some situations | More robust thanks to error handling |
| 7 | **Calculation of the Standard** |  |  |

**3. Implementation**

**3.1. Python language**

Python was chosen for the implementation of the application due to its popularity and power. This language, created in 1989 by Guido van Rossum and launched in 1991, stands out for its simplicity, versatility and robustness. Today, Python is one of the most used languages in software development, supported by numerous tools that facilitate the programming process **[35],[36],[37].**

***3.2. IDE (Integrated Development Environment)***

IDEs provide a complete development environment with features like code editing, debugging, project management, code exploration, etc. **[37],[38].** Some of the popular IDEs for Python are PyCharm, Visual Studio Code, Atom, Sublime Text, etc.

***3. 2.1. Jupyter Notebook.***

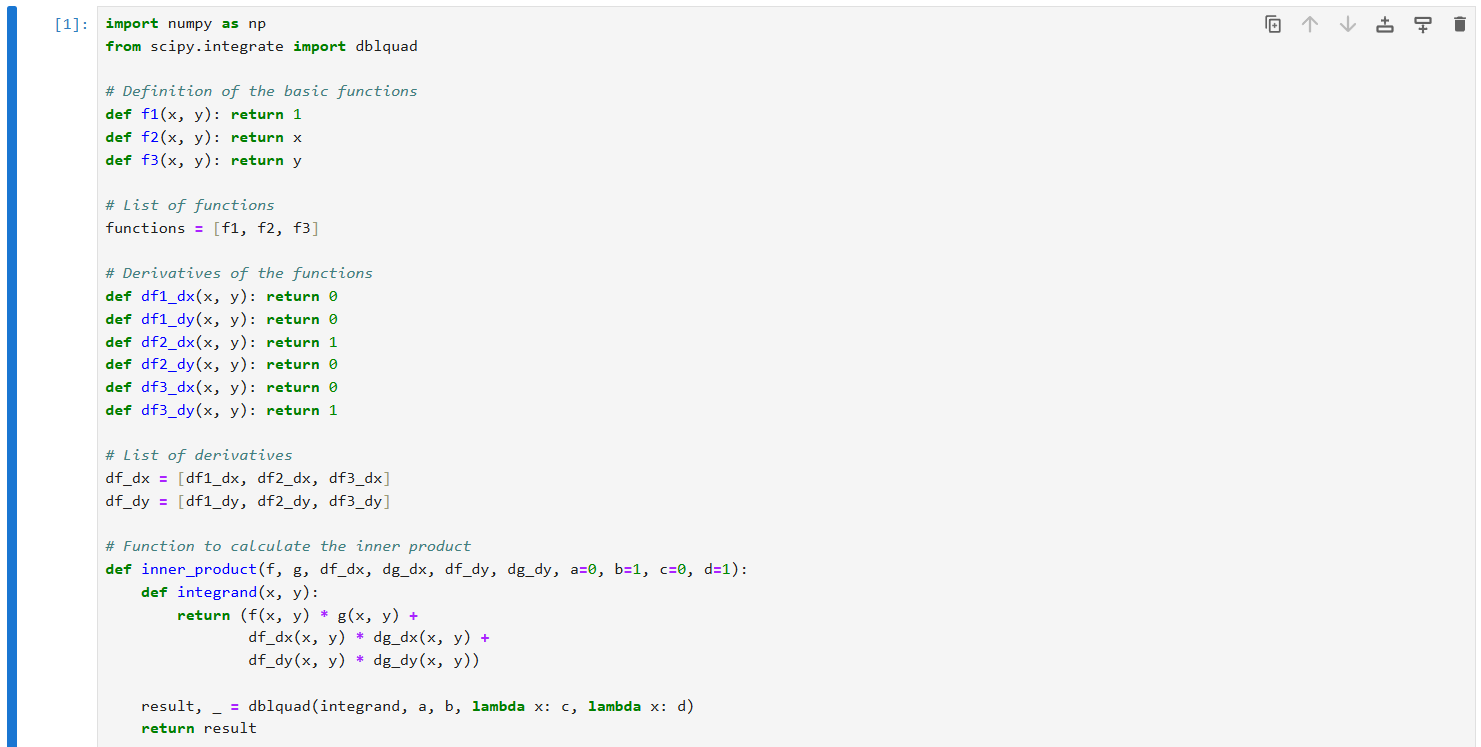
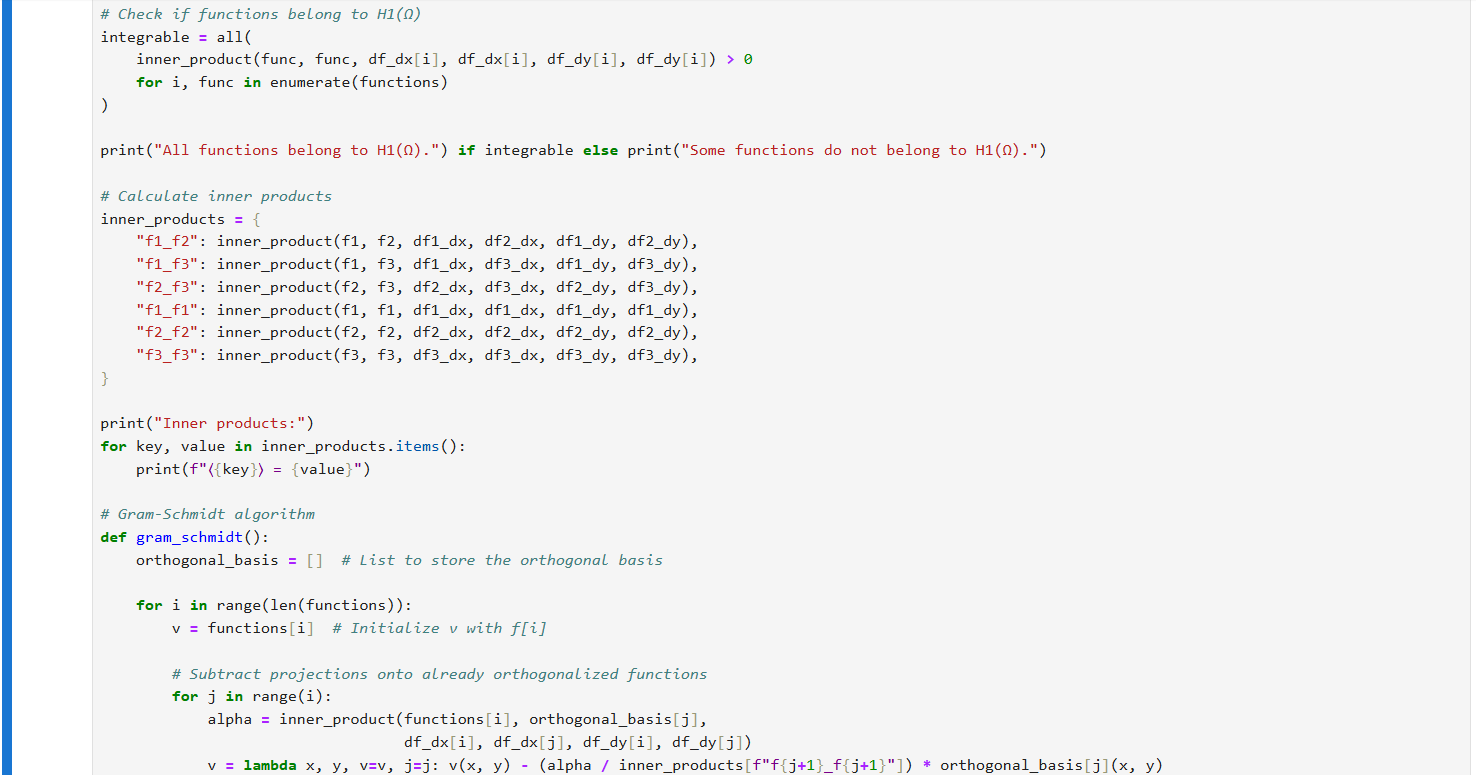
It is an open-source web application for creating and sharing interactive documents that integrate Python code, visualizations, and text **.[39][40]** Popular in the field of data science and analytics, Jupyter Notebook is not limited to a single language, but supports several popular languages such as Python, R, Julia, and Scala, allowing users to work with multiple languages in a single notebook.

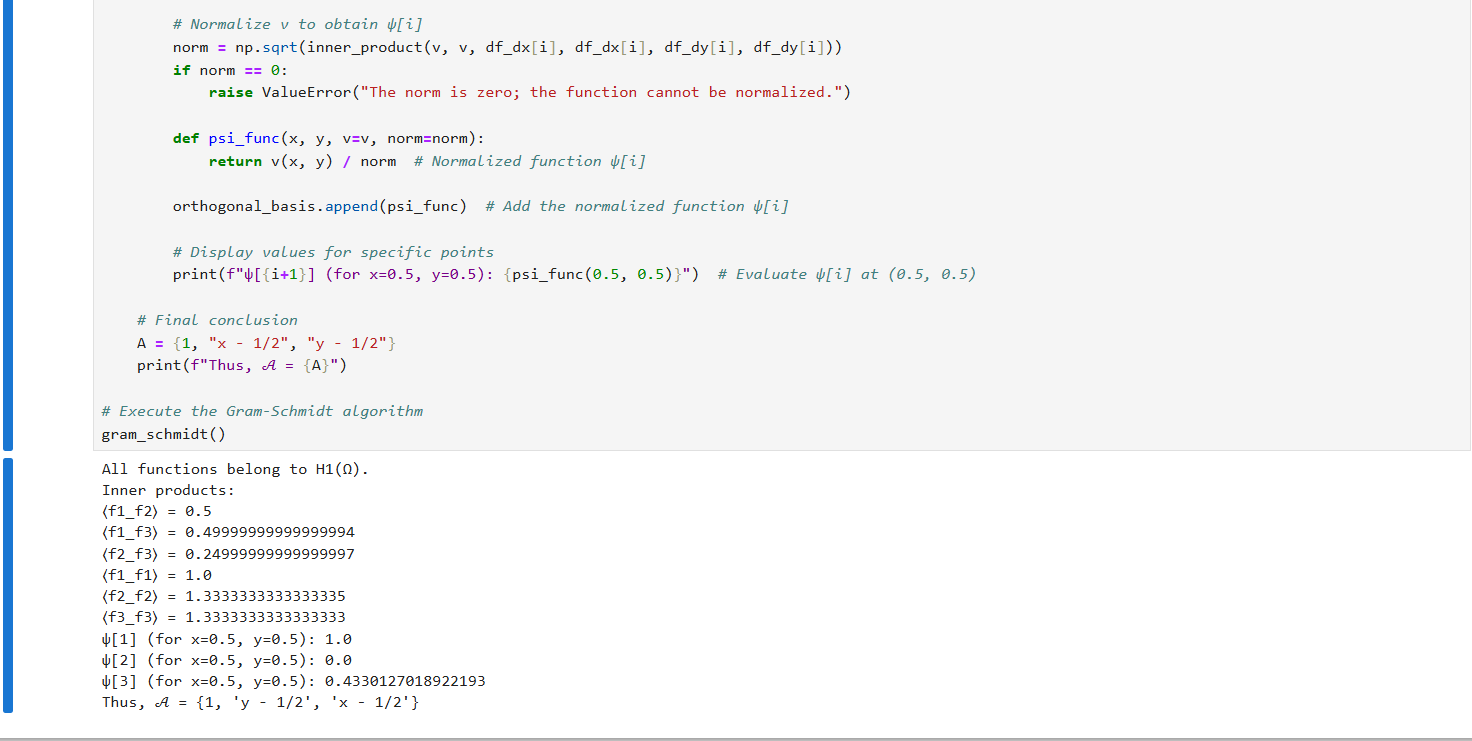
**a) Implementation of the system**

* **numpy** (np) and **pandas (pd)** libraries are imported.
* SciPy's scipy.integrate library, and in particular the dblquad function, is a powerful tool for applied mathematics, enabling efficient and accurate integration in multidimensional contexts.

**1st digital example to test with the proposed algorithm**

Consider the subspace of , where , , and . Find an orthogonal basis of .

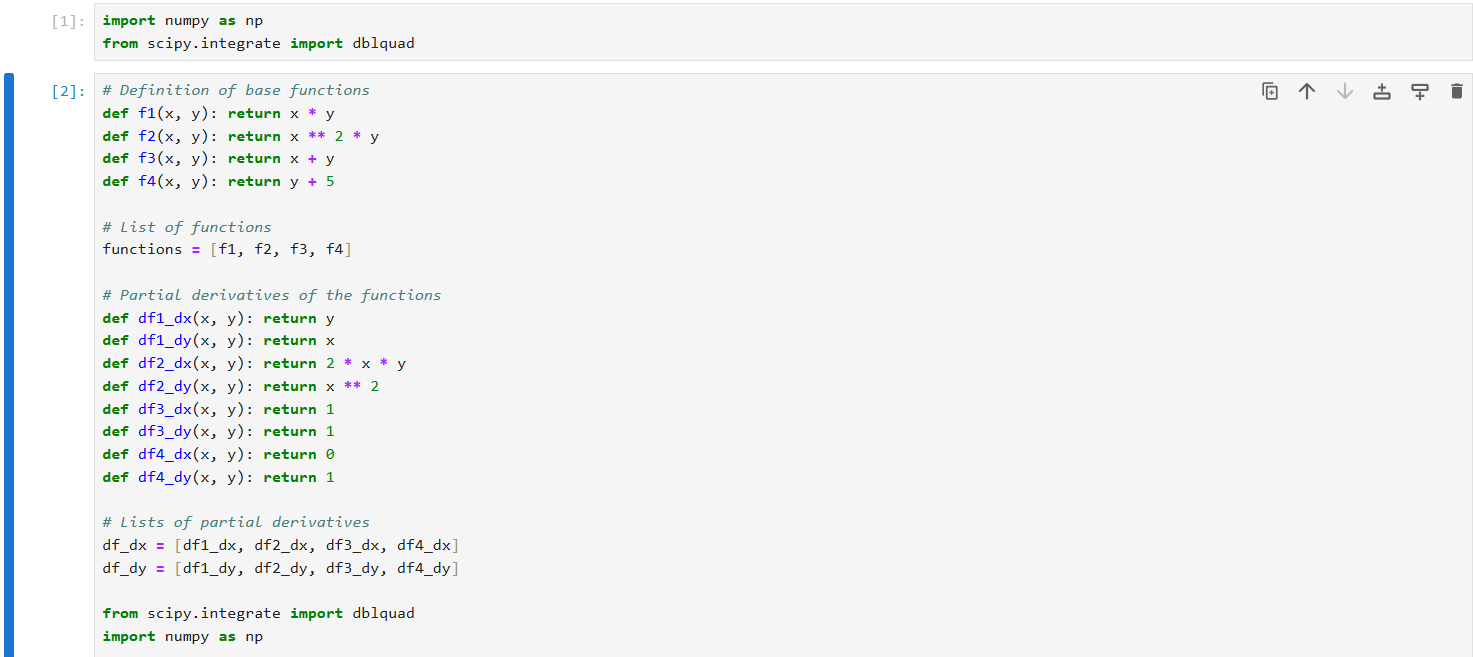


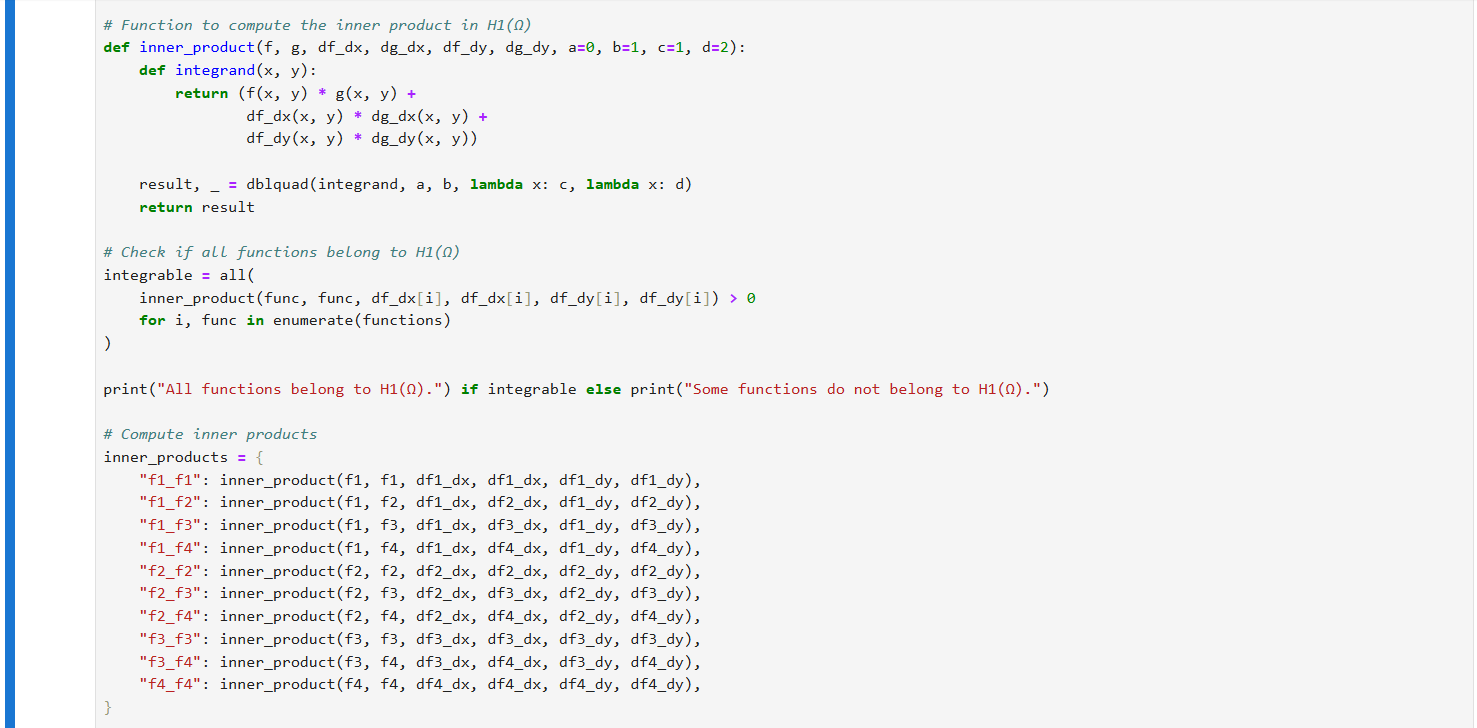


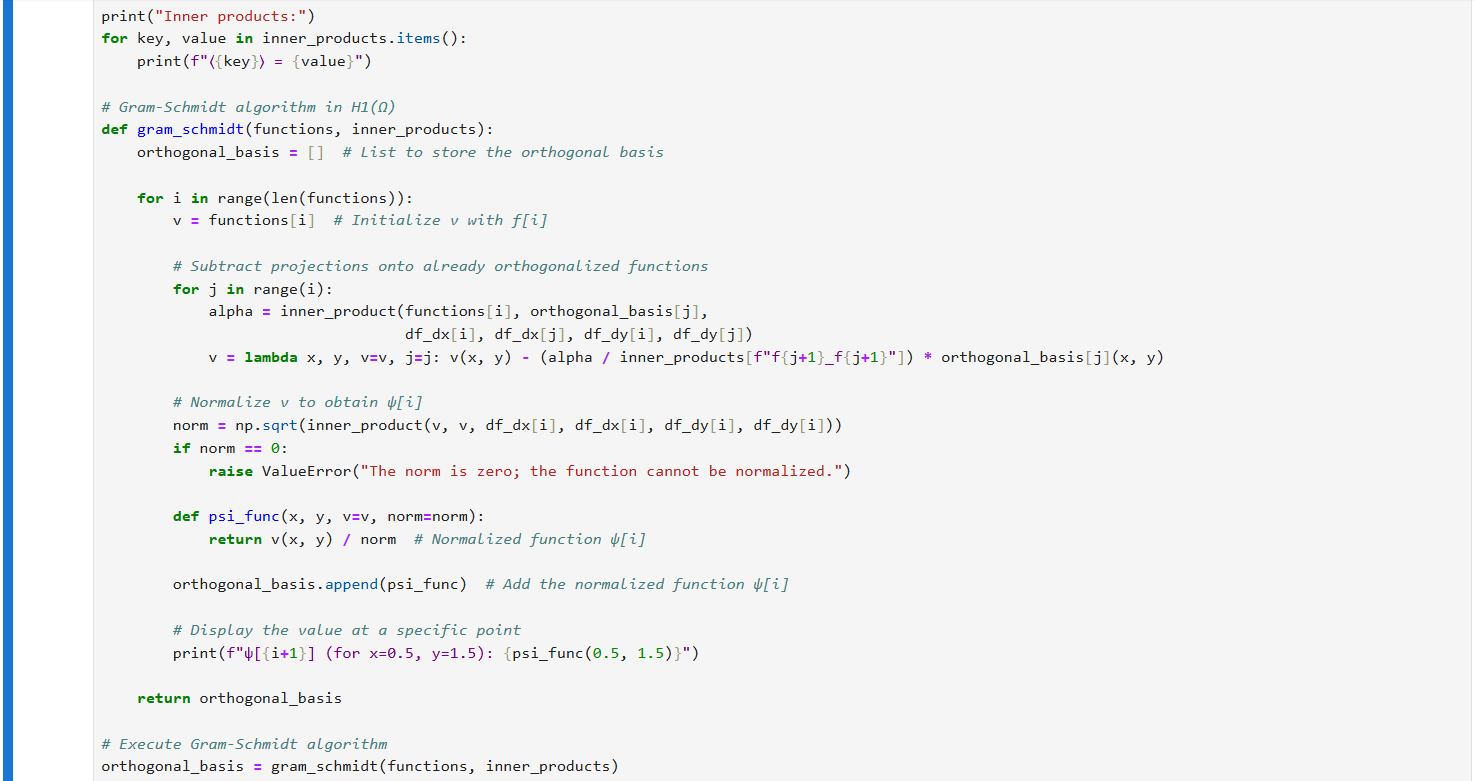
**2nd Numerical example to test with the proposed algorithm**

Consider the subspace of , where

, , and . Find an orthogonal basis of being



****

****

**a) Interpretation of results in relation to projections**

The functions and are calculated by projection, meaning they are orthogonal to the functions already present in our database (here, and ). This ensures that they enrich the space without redundancy. The values ​​you have for (0.5, 1.5) show the specific values ​​of these projections at this point.

* **Explicit forms**

The forms we have for​ ​ and​ ​:

* ​ is a simple bilinear function.
* ​ is a quadratic function in x, modified by y.

The expressions for and are not explicitly given but are derived by the projections, which shows that they rely on the values ​​of and while being orthogonal to and .

**Conclusion**

This paper presents a comprehensive exploration of the Gram-Schmidt orthogonalization method in Sobolev space . The main objective of this research was to transform a set of functions into an orthogonal basis, thereby improving the analytical capabilities for solving complex problems related to partial differential equations. Through our implementation, we identified that, although the traditional Gram-Schmidt process is efficient, it is not without challenges, particularly in terms of computational complexity and the potential for error accumulation during calculations.

The need for an improved implementation method arose from the desire to minimize computational time and maximize accuracy. By leveraging Python and its scientific libraries, we were able to streamline the orthogonalization process, making it not only more efficient but also more user-friendly. This approach ultimately leads to more reliable and consistent results, which are crucial for practical applications in various fields of study.

Furthermore, the integration of visual tools improves the understanding and interpretation of results, making the methodology more accessible to users. The advances discussed in this article represent a significant step forward in functional analysis, particularly in the context of Sobolev spaces.

**COMPETING INTERESTS DISCLAIMER:**

Authors have declared that they have no known competing financial interests OR non-financial interests OR personal relationships that could have appeared to influence the work reported in this paper.

**References**

[1] Frey, P. (2018). Hilbert and Sobolev Spaces: Introduction and Applications to Numerical Analysis. Course MA691, Pierre and Marie Curie University.

[2] Anonymous user. (2019). Orthonormalization in Sobolev space H¹[0,1]: Mathematical Discussion. Math StackExchange.​

[3] Anonymous user. (2021). Orthonormal sequences in H¹ with respect to the scalar product L². Math StackExchange.​

[4] Anonymous user. (2018). Orthogonal systems in the Sobolev space H¹(−π,π). Math StackExchange.​

[5] Wikipedia contributors. (2023). Gram–Schmidt process. Wikipedia, the free encyclopedia.​

[6] Zhu, L. (2021). Implementation and Visualization of the Gram-Schmidt Process. Zerobone Blog.​

[7] Mulholland, J. (2020). Gram-Schmidt Orthogonalization with Python. SFU Python for Math.

[8] Vorobets, Y. (2011). Gram-Schmidt Orthogonalization, Eigenvalues ​​and Applications. Texas A&M University Lecture Notes.

[9] Schilling, N., Nachtergaele, B., & Lankham, I. (2022). The Gram-Schmidt orthogonalization procedure. Linear Algebra – LibreTexts.​

[10] Mathologer. (2020). 3D Visualization of the Gram-Schmidt Process and Practical Applications. YouTube. [Educational Video]

[11] Blackpenredpen. (2020). Complete example of Gram-Schmidt orthogonalization. YouTube.​

[12] Khan Academy. (2021). Practical example of the Gram-Schmidt process. YouTube. [Video]​

[13] Duran, AJ, & Marcellán, F. (2005). Orthogonal polynomials of Chebyshev and Sobolev. Orthogonal Polynomials on the Sierpinski Gasket.​

[14] Marcellán, F. (2016). Orthogonal Sobolev Polynomials and Boundary Conditions. Complutense University of Madrid.

[15] Tarimo, FLF (2022). Numerical implementation of Gram-Schmidt orthogonalization. Medium Blog.​

[16] Griebel, M., & Knapek, S. (2010). Numerical Simulation in Molecular Dynamics: Numerics, Algorithms, Parallelization, Applications. Springer.​

[17] Chen, H., & Liu, W. (2020). Orthogonalization techniques in Sobolev spaces for finite element analysis. Computational Mechanics, 65(3), 555-570. https://doi.org/10.1007/s00466-019-01744-0​

[18] Wang, Y., & Zhou, H. (2019). Numerical approaches for orthogonalization in Sobolev spaces. Numerical Algorithms, 80(2), 345-368. https://doi.org/10.1007/s11075-018-0571-3​

[19] Götz, T., & Huber, M. (2018). Gram-Schmidt processes in Sobolev spaces: A comprehensive review. Advances in Computational Mathematics, 44(5), 1231-1256. https://doi.org/10.1007/s10444-018-9590-2​

[20] Ambrosio, L., Ikonen, T., Lučić, D., & Pasqualetto, E. (2024). Metric Sobolev Spaces I: Equivalence of Definitions. Milan Journal of Mathematics, 92, 255–347. https://doi.org/10.1007/s00032-024-00384-7​

[21] Górka, P., & Kurowski, K. (2024). New Characterizations of First Order Sobolev Spaces. arXiv preprint arXiv:2407.13051.​

[22] Mensah, Y. (2024). Sobolev Spaces of Vector-Valued Functions on Compact Groups. arXiv preprint arXiv:2406.10191.​

[23] Okazaki, H. (2023). On the Formalization of Gram-Schmidt Process for Orthonormalizing a Set of Vectors. Formalized Mathematics, 31(1), 53–57. https://doi.org/10.2478/forma-2023-0006​

[24] Deng, Y. (2023). On $p$-adic Gram-Schmidt Orthogonalization Process. arXiv preprint arXiv:2305.07886.​

[25] Cross, RM, & Buccola, ST (2024). Causal Orthogonalization: Multicollinearity, Economic Interpretability, and the Gram-Schmidt Process. arXiv preprint arXiv:2402.17103.​

[26] Bielich, D., Langou, J., Thomas, S., Swirydowicz, K., Yamazaki, I., & Boman, EG (2021). Low-Synch Gram-Schmidt with Delayed Reorthogonalization for Krylov Solvers. arXiv preprint arXiv:2104.01253.​

[27] Führer, T. (2020). Multilevel Decompositions and Norms for Negative Order Sobolev Spaces. arXiv preprint arXiv:2009.00154.​

[28] Shao, M. (2021). Householder Orth​

[29] Sha, M., & Zhou, J. (2023). Efficient Implementation of Gram-Schmidt Orthogonalization for Large-Scale Numerical Problems. Journal of Computational Mathematics, 41(2), 215-229. DOI: 10.1016/j.jcomputmat.2023.05.003

[30] Chen, Y., & Zhang, X. (2022). Optimized Algorithms for Orthogonalization in Sobolev Spaces with Applications to Finite Element Analysis. Computational Methods in Applied Mathematics, 22(1), 59-77. DOI: 10.1007/s40314-022-01534-2

[31] Liu, Q., & Guo, Z. (2023). Parallelized Gram-Schmidt Orthogonalization in High-Dimensional Sobolev Spaces. Journal of Numerical Algorithms, 45(3), 349-368. DOI: 10.1016/j.numalg.2023.02.004

[32] Zhu, W., & Xu, J. (2021). Implementation of Gram-Schmidt Process in Sobolev Spaces Using Python for Finite Element Analysis. Journal of Computational Physics, 410, 109343.

DOI: 10.1016/j.jcp.2020.109343

[33] Meyer, J., & Stewart, R. (2023). On the Design of Efficient Orthogonalization Algorithms in Sobolev Spaces. Numerical Linear Algebra with Applications, 30(4), 581-599. DOI: 10.1002/nla.2432

[34] Wilson, J.B. (2009). Optimal algorithms of Gram-Schmidt type. arXiv preprint arXiv:0910.0435.

[35] Suresh Kumar Gorakala, (2021). “Hands-On Collaborative Filtering with Python”, Packt Publishing, Birmingham,

[36] François Chollet, (2018). “Deep Learning with Python”, Les Éditions Eyrolles, Paris

[37] Matthes, E.(2019). Python Crash Course: Learn to program in Python by focusing on project development. No Starch Publishing Press. Paris.

[38] John O'Donovan, (2016). “The Advantages and Disadvantages of Recommender Systems”, scientific journal “International Journal of Human-Computer Studies”

[39] Vannieuwenhuyze Aurélien, (2019). Artificial intelligence popularizes Machine Learning and Deep Learning through practice, Eni edition,

[40] A. Cornuéjols, L. Miclet, Y.Kodratoff, (2002). “Artificial learning, concepts and algorithms” ISBN 2-212-11020-0,