J. Adv. Educ. Sci. Humanit. (July - December 2026) 4(2): 15-22
https://doi.org/10.5281/zenodo.21854571
ISSN 3091-1583

JAESH
JOURNAL OF ADVANCES IN EDUCATION, SCIENCES AND HUMANITIES

ORIGINAL ARTICLE

Automated Geometric Modeling and Parametric Analysis in COMSOL Multiphysics® Using MATLAB® LiveLink™

Automatización del modelado geométrico y análisis paramétrico en COMSOL Multiphysics® utilizando MATLAB® LiveLink™

Jaime E. Pérez • **Duliet Hong**

Received: 09 March 2026 / Accepted: 04 July 2026 / Published online: 31 July 2026

Abstract The manual construction of multiphysics models in COMSOL Multiphysics® with variable geometries is an iterative task, prone to human errors and hardly reproducible when large design spaces need to be explored. This paper presents a systematic methodology for the full automation of the COMSOL Multiphysics® modeling cycle using the MATLAB® LiveLink™ interface: from parametric CAD geometry construction to multidisciplinary physics configuration, adaptive meshing and solver execution. The architecture of a four- module automation environment is described —parametric input, geometry construction, physics and multiphysics assignment, and resolution— and each module is formalized with its methodological rationale, reusable code patterns and numerical quality considerations. The methodology is illustrated with an engineering case study: the simulation of twelve geometric variants of molds for a pulsed electric field sintering process, in which manual exploration would have required between 9 and 18 hours of work and was completed in under 2 hours with the automated environment. Results validate the effectiveness of the approach and confirm its transferability to any engineering domain requiring parametric studies over variable geometries in COMSOL Multiphysics®.

Keywords COMSOL Multiphysics®, MATLAB® LiveLink™, parametric modeling, CAD automation, multiphysics FEM, simulation-driven design, design space exploration.

Resumen La construcción manual de modelos multifísicos en COMSOL Multiphysics® con geometrías variables es una tarea iterativa, propensa a errores humanos y difícilmente reproducible cuando se necesita explorar grandes espacios de diseño. Este artículo presenta una metodología sistemática para la automatización completa del ciclo de modelado en COMSOL Multiphysics® mediante la interfaz MATLAB® LiveLink™: desde la parametrización de la geometría CAD hasta la configuración de la física multidisciplinar, el mallado adaptativo y la ejecución del solucionador. Se describe la arquitectura de un entorno de automatización de cuatro módulos—entrada paramétrica, construcción geométrica, asignación de física y multifísica, y resolución— y se formaliza cada módulo con su fundamento metodológico, los patrones de código reutilizables y las consideraciones de calidad numérica. La metodología se ilustra con un caso de estudio de ingeniería: la simulación de doce variantes geométricas de moldes para un proceso de sinterización por campos eléctricos pulsados, en el que la exploración manual habría requerido entre 9 y 18 horas de trabajo y se completó en menos de 2 horas con el entorno automatizado. Los resultados validan la efectividad del enfoque y confirman su transferibilidad a cualquier dominio de ingeniería que requiera estudios paramétricos sobre geometrías variables en COMSOL Multiphysics®.

Palabras clave COMSOL Multiphysics®, MATLAB® LiveLink™, modelado paramétrico, automatización CAD, MEF multifísico, diseño basado en simulación, exploración de espacio de diseño.

How to cite

Pérez, J. E., & Hong, D. (2026). Automated Geometric Modeling and Parametric Analysis in COMSOL Multiphysics® Using MATLAB® LiveLink™. *Journal of Advances in Education, Sciences and Humanities*, *4*(2), 15-22. https://doi.org/10.5281/zenodo.21854571

✉ Jaime E. Pérez
jaime.fernandez@unic.co.ao

Faculdade de Engenharia, Universidade Internacional do Cuanza, Cuito, Angola.

PUBLIS EDITORIAL

Introduction

Multiphysics simulation using the Finite Element Method (FEM) has become a fundamental tool in modern engineering. Platforms such as COMSOL Multiphysics® make it possible to model the simultaneous interaction of thermal, mechanical, electrical, electromagnetic, and fluid phenomena within a unified environment (COMSOL, 2022). However, when the objective is to explore system behavior as a function of geometric or parametric variations—as occurs in design studies, optimization, or sensitivity analysis—the manual construction of each variant through COMSOL's graphical user interface (GUI) becomes a significant bottleneck.

The GUI-based workflow presents three structural limitations for parametric exploration: (1) the configuration time required for each model increases in proportion to the number of variants, making the analysis of dozens or hundreds of cases impractical; (2) manual data entry introduces consistency errors among variants; and (3) the steps performed through the GUI are not documented in a reproducible manner, hindering traceability and peer review. These limitations are particularly critical in fields where device geometry is the primary design variable, such as manufacturing molds, fluid channels, bone structures in bioengineering, antenna geometries, or components with graded properties.

COMSOL Multiphysics® provides a direct solution through the MATLAB® LiveLink™ interface, which exposes the core modeling engine as a Java object library accessible from the MATLAB® programming environment. Through this interface, the entire modeling cycle—geometry, physics, meshing, and solution—can be encoded as an executable, parameterized, and reusable script (COMSOL, 2022; Mayboudi, 2020).

Despite the evident advantages of this approach, the literature lacks a systematic and transferable description of the automation methodology. Studies that use LiveLink™ for MATLAB® often present the code as a secondary artifact of the article, without documenting the design decisions underlying the modeling environment or the reusable patterns that would allow its adaptation to new problems (Pepper & Heinrich, 2017; Griebel et al., 2017). This article addresses that gap.

The objective of this study was to describe a formal and transferable methodology for automating modeling in COMSOL Multiphysics® through MATLAB® LiveLink™, applicable to any engineering field requiring the systematic evaluation of models with variable geometry. The methodology is structured around four functional modules and is illustrated through a materials engineering case study. The contributions of the article are: (i) the formalization of the automation architecture, (ii) the documentation of fundamental coding patterns, (iii) a mixed-meshing strategy for models with kinematically heterogeneous domains, and (iv) the quantification of the computational benefits compared with the manual workflow.

Methodology

COMSOL Multiphysics® exposes its modeling engine through a high-level Java API. The LiveLink™ for MATLAB® interface acts as a bridge between this API and the MATLAB® environment, allowing COMSOL models to be instantiated, manipulated, and solved directly from MATLAB® code. From a software engineering perspective, LiveLink™ implements the Facade design pattern: it abstracts the internal complexity of the COMSOL engine and exposes it through a coherent and hierarchical object interface (COMSOL, 2022).

A COMSOL model is represented in MATLAB® as a `model` object created using the `ModelUtil.create('Model') command. This object provides access to all model subsystems through chained methods that follow the hierarchy: `model → component → {geom, physics, mesh, study}. Table 1 outlines this hierarchy and its correspondence with the modules of the proposed automation framework.

The LiveLink™ interface allows global model parameters to be defined using `model.param.set (name, value, description). These parameters are evaluated by the COMSOL engine throughout all stages of the model—including geometry construction, boundary conditions, and material properties—and can be modified programmatically before each solution run. This constitutes the fundamental mechanism for automated parametric exploration (Figure 1).

Table 1. Object hierarchy of the COMSOL Multiphysics® API accessible through MATLAB® LiveLink™

Level	COMSOL Object	Responsibility
1	model	Root object: global parameters, label, and file path
2	model.component('comp1')	Model component comprising geometry, physics, and mesh
3a	.geom('geom1')	1D, 2D, 3D, or axisymmetric CAD geometry
3b	.physics('id')	Physics interface, such as FEM or heat transfer
3c	.mesh('mesh1')	Mesh configuration
4	model.study('std1')	Study type: stationary, time-dependent, among others

```
% Definición de parámetros globales
model.param.set('h_muestra', '0.01[m]',   'Espesor de la muestra');
model.param.set('r_muestra', '0.005[m]', 'Radio de la muestra');
model.param.set('T_sinter',    '1173[K]',   'Temperatura de sinterización');
model.param.set('P_comp',      '50[MPa]',   'Presión de compactación');
```

Figure 1. Definition of the global parameters of the sample and the sintering process in COMSOL Multiphysics.

The use of global parameters instead of literal values in the code provides two advantages: (1) it centralizes variation among models at a single control point, and (2) it allows COMSOL to evaluate parametric expressions in the boundary conditions, thereby eliminating inconsistencies between modules.

The COMSOL geometry API supports 2D primitives (Rectangle, Circle, Ellipse, Polygon, and BezierPolygon) and 3D primitives (Block, Cylinder, Sphere, and Cone), as well as Boolean operations (Union, Difference, and Intersection) and geometric transformations (Move, Rotate, Scale, and Mirror). For problems with axial symmetry, the spatial dimension can be configured as 2D axisymmetric, thereby reducing the computational cost of the corresponding three-dimensional problem without compromising solution accuracy.

The proposed methodology organizes the automation framework into four sequential functional modules, as shown in Table 2. This modular separation follows the single-responsibility principle: each module encapsulates a clearly defined function within the modeling cycle, thereby facilitating the replacement or extension of any module without affecting the others.

Table 2. Modular architecture of the proposed automation framework

Module	Function	Output
M1: Parametric input	Receive and validate the geometric and physical parameters of the problem	Parameter vectors/structures
M2: CAD geometry	Programmatically construct the system geometry in COMSOL	geom object with *n* subdomains
M3: Physics and multiphysics	Assign physics interfaces, material properties, and multiphysics couplings	Multiphysics model ready to be solved
M4: Meshing and solution	Configure the adaptive mesh and run the solver	Results dataset for post-processing

The input module defines the interface between the analyst and the automation framework. Its function is to receive the parameters that characterize each model variant and transfer them to the COMSOL engine through `model.param.set()`. Three implementation modes are available, each offering a different level of automation:

Interactive input: uses MATLAB®'s `input()` function. It is suitable for exploratory analyses or models with a limited number of parameters.

Input from a structured file (JSON, CSV, or Excel): enables batch execution without user intervention. It is recommended for design studies involving more than ten variants.

Input from an external calling script: the automation framework is invoked as a function within a MATLAB® `for` loop, with the parameters passed as arguments. This is the most appropriate pattern for integration with optimization algorithms.

Regardless of the selected mode, Module M1 should include range validation and geometric constraints—for example, ensuring that the inner radius of the mold is greater than the specimen radius—to prevent invalid geometries from reaching the COMSOL engine and causing late-stage errors that are difficult to diagnose.

This module (Module M2: CAD Geometry Construction) implements the programmatic construction of the system geometry. Its fundamental design principle is exclusive parameter dependence: no coordinate or dimension should appear as a literal value in the code; instead, all values must be derived from the parameters defined in M1. This ensures that any modification made in M1 is automatically propagated throughout the entire geometry.

The general pattern for creating a two-dimensional geometric component is presented in Figure 2.

```
% Patrón general: crear primitiva y asignar posición/dimensiones paramétricas
g = model.component('comp1').geom('geom1');
g.create('id_elemento', 'TipoPrimitiva');
g.feature('id_elemento').label('Nombre descriptivo');
g.feature('id_elemento').set('pos',    [r0, z0]);   % posición (r, z)
g.feature('id_elemento').set('size', [dr, dz]);   % dimensiones
g.run('id_elemento');
```

Figure 2. General pattern for creating geometric primitives and assigning parametric positions and dimensions in COMSOL Multiphysics.

For systems with repeated components—for example, multiple spacers or material layers—the preceding pattern is encapsulated within a parameterized loop. Figure 3 illustrates the relative-positioning logic: each component is positioned according to the boundary position of the preceding component, accumulated in a displacement variable (offset_z). This approach eliminates the need to recalculate absolute positions when the dimensions of any component change.

```
offset_z = h_muestra / 2;  % borde superior de la muestra

for k = 1:n separadores
    id = sprintf('sep_%d', k);
    g.create(id, 'Rectangle');
    g.feature(id).set('pos',  [0, offset z]);
    g.feature(id).set('size', [r_sep(k), h_sep(k)]);
    g.run(id);
    offset z = offset z + h sep(k);  % actualizar offset
end
```

Figure 3. Parametric creation and vertical positioning of separator elements in the COMSOL geometry.

For molds with complex geometric profiles—such as conical, curved, or control-point-defined profiles—the Bezier-Polygon primitive makes it possible to construct arbitrary contours using segments of configurable degree. When the mold profile is the variable design parameter, the set of control points is recalculated during each iteration based on the M1 parameters, thereby automatically generating the desired geometry.

Module M3 assigns the physics interfaces governing the problem to the geometry constructed in M2. The COMSOL API distinguishes between physics interfaces, which implement the governing equations over the subdomains, and multiphysics couplings, which link the dependent variables of different interfaces. Its programmatic declaration follows the pattern shown in Figure 4.

```
% Declarar interfaz física
model.component('comp1').physics.create('id', 'TipoFísica', 'geom1');

% Asignar propiedad a un subdominio específico
model.component('comp1').physics('id').feature('mat1') ...
    .set('NombrePropiedad', valor);

% Declarar acoplamiento multifísico
model.component('comp1').multiphysics.create('acoplo1', 'TipoAcoplamiento',
dim);
model.component('comp1').multiphysics('acoplo1').set('Source_physics',
'id_fuente');
model.component('comp1').multiphysics('acoplo1').set('Target_physics',
'id_destino');
```

Figure 4. Creation of the physics interface, assignment of subdomain properties, and definition of multiphysics coupling in COMSOL Multiphysics.

Material properties are assigned through subdomain groups so that all subdomains composed of the same material share identical properties without code duplication. When material properties depend on temperature or other model variables, they must be defined as COMSOL expressions—text strings evaluated by the engine during each solver iteration—rather than as scalar values.

A recommended practice is to separate material assignment from the definition of boundary conditions by placing them in independent subfunctions. This facilitates the reuse of material-related code when the geometry is modified, and vice versa.

Module M4 has the greatest impact on the accuracy and computational cost of the simulation. Mesh quality determines the discretization error of the numerical solution, whereas the number of elements determines the solution time. For problems involving multiple coupled physics and kinematically heterogeneous domains, the meshing strategy must be carefully designed.

A mixed-meshing strategy comprising three domain types is proposed for problems that combine deformation, displacement, and rigid domains:

Fixed mesh (Free triangular): used for rigid domains without displacement or deformation. It is generated automatically and conforms to the geometry. This is the default mesh type and the most computationally efficient.

Adaptive mesh with Laplacian smoothing (Deformed geometry): used for domains undergoing rigid-body displacement, meaning that they move as solids without deforming. Laplacian smoothing redistributes the mesh nodes according to the displacement field, thereby preventing element distortion.

Moving mesh (Moving mesh): used for domains undergoing actual material deformation, such as fluids or powder materials under compression. It requires the configuration of boundary conditions for mesh displacement.

Mesh quality (Figure 5) is evaluated using the index q, defined for triangular elements as $q = 4\sqrt{3}\cdot A / (h_1^2 + h_2^2 + h_3^2)$, where A is the area of the triangle and h_i represents the length of each side. A value of $q > 0.3$ ensures that mesh quality does not compromise the solution, whereas $q = 1$ corresponds to the optimal equilateral triangle.

```
mesh = model.component('comp1').mesh('mesh1');

% Malla fija para dominios rígidos
mesh.create('fmap', 'FreeTri');
mesh.feature('fmap').selection.geom('geom1', 2);
mesh.feature('fmap').selection.set(dominios fijos);
mesh.feature('fmap').set('method', 'del');   % Delaunay

% Refinamiento local en fronteras críticas
mesh.create('ref_local', 'Size');
mesh.feature('ref_local').selection.geom('geom1', 1);
mesh.feature('ref_local').selection.set(fronteras_criticas);
mesh.feature('ref_local').set('hauto', 3);   % tamaño fino

mesh.run();
```

Figure 5. Mesh generation with fixed-domain meshing and local refinement at critical boundaries in COMSOL Multiphysics.

For stationary problems, COMSOL offers three direct solvers based on LU decomposition: MUMPS, PARDISO, and SPOOLES. The optimal choice depends on the characteristics of the problem and the available hardware. Table 3 summarizes their comparative features.

For most parametric studies conducted on desktop hardware or conventional computing servers, PARDISO provides the best balance between computational speed and memory usage. Its configuration in LiveLink™ iis performed as in Figure 6.

To illustrate the methodology, a materials engineering problem was selected: the design of molds for the Spark Plasma Sintering (SPS) of functionally graded materials (FGMs). During the SPS process, a pulsed direct current flows through the punch–mold–specimen system, generating heat through the Joule effect.

Table 3. Comparison of direct solvers for stationary studies in COMSOL Multiphysics®

Solver	Speed	RAM usage	Out-of-core	Recommended for
PARDISO	Very high	High	Yes	Medium and large models; Intel hardware
MUMPS	High	High	Yes	MPI cluster computing
SPOOLES	Low	Moderate	No	Small models; limited RAM

```
model.study('std1').feature('stat').set('solnum', 'auto');
model.sol('sol1').feature('s1').set('linsolver', 'pardiso');
model.sol('sol1').runAll();
```

Figure 6. Configuration and execution of the stationary study using the PARDISO solver in COMSOL Multiphysics.

The current distribution—and consequently the axial temperature profile—depends critically on the mold geometry. The objective of the parametric study was to identify the mold geometry that maximizes the axial thermal gradient, ΔT, within the specimen, which is a necessary condition for producing FGMs with a predetermined gradation.

The multiphysics model is a coupled electro-thermo-mechanical model that integrates the electric charge conservation equations, Fourier's heat conduction equation with a Joule-heating source term, and the mechanical deformation equations under uniaxial pressure. The system is axisymmetric and was therefore solved using a two-dimensional axisymmetric model. The sintering temperature was 1,173 K, and the compaction pressure was 50 MPa, corresponding to the processing conditions for barium hexaferrite (BaM).

Twelve geometric variants derived from two reference designs (Tokita et al., 2003, 2007) were evaluated and designated 1V1–1V6 and 2V1–2V6. The parameter varied among the alternatives within each family was the axial position of the transition point in the mold's conical profile. The specimen geometry remained constant across all variants, with a thickness of 0.01 m and a radius of 0.005 m.

Module M2 constructs the complete system geometry through a loop that iterates over the twelve variants. During each iteration, the mold profile is generated using a quadratic `BezierPolygon` primitive whose control points are recalculated according to the parameter P of the corresponding variant. The execution time of Module M2 for each variant, excluding the solution stage, is less than 1 second.

Module M3 defines three physics interfaces—solid mechanics, heat transfer, and electric currents—and three multiphysics couplings: thermal expansion, temperature coupling, and electromagnetic heating. The material properties—ED-3 graphite and Inconel 718 for the mold and electrodes, and BaM powder for the specimen—are assigned through subdomain groups.

Module M4 implements the mixed-meshing strategy described in Section 3.4.1: a fixed mesh for the Inconel components, an adaptive mesh with Laplacian smoothing for the punches, which move axially under the applied pressure without deforming, and a moving mesh for the compacted powder specimen. The resulting average mesh quality was q = 0.73, with 2,572 triangular elements.

Results and discussion

Table 4 presents the axial thermal gradients (ΔT) obtained for the twelve variants. Compliance with the minimum functional gradation criterion is also indicated (ΔT ≥ 25 K), a threshold established in the literature to ensure microstructural heterogeneity (El-Wazery & El-Desouky, 2015).

Table 4. Axial thermal gradient ΔT (K) for the Twelve geometric variants evaluated. Overall optimal variant.

Family	V1	V2	V3	V4	V5	V6	Optimal Variant	Maximum ΔT (K)
Family 1 (ΔT, K)	31.24	31.09	31.49	28.72	24.09	15.61	1V3	31.49
Family 2 (ΔT, K)	59.04	49.15	57.65	50.87	34.47	18.04	2V1	59.04
Minimum threshold	—	—	—	—	—	—	≥ 25 K	—

Nine of the twelve variants exceeded the functional gradation threshold. Variant 2V1 produced the highest gradient (ΔT = 59.04 K), approximately twice that of the best variant in Family 1 (1V3, ΔT = 31.49 K). The variants with the largest outer mold radius (1V6 and 2V6) produced the lowest gradients, indicating that excess conductive mass disperses the current flow excessively.

Table 5 quantifies the computational benefits of the automation framework compared with the manual workflow in the COMSOL graphical user interface for the present case study.

Table 5. Efficiency comparison between the Manual GUI Workflow and the automated framework

Activity	Manual GUI Workflow	Automated workflow	Reduction
Configuration per variant (geometry + physics + mesh)	45–90 min	< 5 s	> 95%
Total configuration time for 12 variants	9–18 hours	< 1 min	> 99%
Consistency errors between variants	Frequent	None	100%
Exact reproducibility	Not guaranteed	Complete	—

The case study results demonstrate that the proposed methodology qualitatively transforms the types of parametric analyses that can be feasibly conducted in COMSOL. The reduction in configuration time of more than 95% is not merely a matter of convenience; it enables design studies involving a number of variants that would be impractical under a manual workflow. This creates opportunities to integrate the framework with gradient-based, evolutionary, or Bayesian optimization algorithms, which require the evaluation of hundreds or thousands of configurations.

From a software engineering perspective, the modular architecture of the framework complies with the SOLID principles. Each module has a single responsibility, in accordance with the Single Responsibility Principle (SRP); modules can be extended without modifying existing ones, as established by the Open–Closed Principle (OCP); and replacing one module—for example, changing the type of physics in M3 or the solver in M4—does not affect the others. This architecture facilitates the reuse of the framework in new engineering fields.

The mixed-meshing strategy presented in Section 3.4.1 is particularly relevant for models with kinematically heterogeneous domains, a common situation in biomechanics involving soft and rigid tissues, fluid–structure interaction, and manufacturing processes involving material deformation. The implementation of three mesh types—fixed, adaptive, and moving—within the same model, together with the appropriate Dirichlet boundary conditions for domains undergoing displacement, is a solution not documented in standard COMSOL tutorials and represents one of the technical contributions of this study.

One limitation of the current implementation is that interactive data entry using (input()) is unsuitable for integration with optimization algorithms. A natural extension would be to replace M1 with an interface capable of reading configuration files in formats such as JSON, YAML, or Excel, and to encapsulate the entire framework as a MATLAB® function with the signature `f(params) → results`. This structure would be compatible with the objective functions used by MATLAB® optimization toolboxes, including `fmincon`, `ga`, and `surrogateopt`.

Conclusions

The methodology presented and the results of the case study support several conclusions. First, a four-module methodology was formalized to fully automate the COMSOL Multiphysics® modeling cycle through MATLAB® LiveLink™. This framework documents reusable coding patterns and the numerical quality considerations associated with each module. The exclusive use of parameters in geometric construction, implemented in Module M2, ensures consistency among model variants and eliminates the human errors associated with manual workflows. In the case study, this approach reduced model configuration time by more than 95%. The proposed mixed-meshing strategy—combining fixed, adaptive with Laplacian smoothing, and moving meshes—successfully addresses the coexistence of domains with different kinematic regimes within a single multiphysics model. The strategy achieved an average mesh quality of q = 0.73 in the case study. The modular architecture of the automation framework is transferable to any engineering field requiring parametric analysis in COMSOL Multiphysics®, regardless of the type of physics involved, the dimensionality of the problem, or the solver employed. Finally, extending the framework toward automatic optimization using gradient-based, evolutionary, or Bayesian search algorithms would require only the replacement of Module M1 with a function interface compatible with MATLAB® optimization toolboxes. This development represents the main direction for future work.

References

Bódis, E., Tapasztó, O., Károly, Z., Balázsi, K. y Balázsi, C. (2022). Fabrication of graded alumina by spark plasma sintering. *The International Journal of Advanced Manufacturing Technology*, 118(9–10), 3519–3528. https://doi.org/10.1007/s00170-021-07855-0

Chen, H., Wei, X., Liang, Z., Zhou, J., Li, Y. y Xie, Z. (2024). Numerical simulation of heat transfer during spark plasma sintering of porous SiC. *Ceramics International*, 50(9), 15 680–15 691. https://doi.org/10.1016/j.ceramint.2024.02.097

COMSOL AB (2023). *COMSOL Multiphysics® Reference Manual, v6.2*. Stockholm: COMSOL AB. https://doc.comsol.com/6.2

COMSOL AB (2023). *LiveLink™ for MATLAB® User's Guide, v6.2*. Stockholm: COMSOL AB. https://doc.comsol.com/6.2/doc/com.comsol.help.llmatlab

El Fallaki Idrissi, M., Praud, F., Meraghni, F., Chinesta, F. y Chatzigeorgiou, G. (2025). Generative parametric design: a framework for real-time geometry generation and on-the-fly multiparametric approximation. *Engineering with Computers* (en prensa). https://doi.org/10.48550/arXiv.2512.11748

Govea Alcaide, E. et al. (2014). Simulación por MEF del proceso de sinterización por Spark Plasma de un óxido cerámico no conductor. *Revista Cubana de Física*, 31(1E), E33–E37.

Li, M., Lin, C., Chen, W., Liu, Y., Gao, S. y Zou, Q. (2023). XVoxel-based parametric design optimization of feature models. *Computer-Aided Design*, 161, 103546. https://doi.org/10.1016/j.cad.2023.103546

Li, Q., Wang, L., Mohebbi, M.S. y Evans, A.G. (2023). Ultra-large temperature gradient in field-assisted sintering for functionally graded materials. *Acta Materialia*, 254,

119032. https://doi.org/10.1016/j.actamat.2023.119032

Mahamood, R.M. y Akinlabi, E.T. (2017). *Functionally Graded Materials*. Topics in Mining, Metallurgy and Materials Engineering. Springer International Publishing.

Matejíček, J., Mušálek, R., Dlabáček, Z., Klevarová, V. y Kocmanová, L. (2022). Processing and properties of tungsten-steel composites and FGMs prepared by spark plasma sintering. *Materials*, 15(24), 9037. https://doi.org/10.3390/ma15249037

Mayboudi, L.S. (2020). *Heat Transfer Modeling with COMSOL®: Fundamentals and Applications*. Mercury Learning and Information.

Mihailova, O., Maslennikova, O. y Chuvil'deev, V. (2023). Functionally gradient material fabrication based on Cr-Ti-Fe-Ni-Co-Cu metal layers via spark plasma sintering. *Coatings*, 13(1), 138. https://doi.org/10.3390/coatings13010138

Miyamoto, Y., Kaysser, W.A., Rabin, B.H., Kawasaki, A. y Ford, R.G. (1999). *Functionally Graded Materials: Design, Processing and Applications*. Kluwer Academic Publishers.

Morin, C. et al. (2016). Influence of the die geometry on temperature distribution in SPS. *Journal of the European Ceramic Society*, 36(16), 4147–4157. https://doi.org/10.1016/j.jeurceramsoc.2016.05.040

Parekh, J., Bekemeyer, P. y Helm, S. (2024). Surrogate-based design space exploration and exploitation for efficient airfoil optimization under uncertainties. *Aerospace Science and Technology*, 154, 109532. https://doi.org/10.1016/j.ast.2024.109532

Pepper, D.W. y Heinrich, J.C. (2017). *The Finite Element Method: Basic Concepts and Applications with MATLAB, MAPLE, and COMSOL* (3.ª ed.). CRC Press.

Shimwell, J., Billingsley, J., Delaporte-Mathurin, R., Morbey, D., Bluteau, M., Shriwise, P. y Davis, A. (2021). The Paramak: automated parametric geometry construction for fusion reactor designs. *F1000Research*, 10, 27. https://doi.org/10.12688/f1000research.28224.1

Conflicts of interest

The author declare that he has no conflict of interest.

Author contributions

Conceptualization: Jaime E. Pérez, Duliet Hong. **Data curation:** Jaime E. Pérez, Duliet Hong. **Formal analysis:** Jaime E. Pérez, Duliet Hong. **Research:** Jaime E. Pérez, Duliet Hong. **Methodology:** Jaime E. Pérez, Duliet Hong. **Supervision:** Jaime E. Pérez, Duliet Hong. **Validation:** Jaime E. Pérez, Duliet Hong. **Visualization:** Jaime E. Pérez, Duliet Hong. **Writing the original draft:** Jaime E. Pérez, Duliet Hong. **Writing, review and editing:** Jaime E. Pérez, Duliet Hong.

Data availability statement

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Statement on the use of AI

The author acknowledges the use of generative AI and AI-assisted technologies to improve the readability and clarity of the article.

Disclaimer/Editor's note

The statements, opinions, and data contained in all publications are solely those of the individual authors and contributors and not of *Journal of Advances Education, Sciences and Humanities*.

Journal of Advances Education, Sciences and Humanities and/or the editors disclaim any responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products mentioned in the content.