Contents lists available at ScienceDirect

Materials and Design

journal homepage: www.elsevier.com/locate/matdes

pyDEM: A generalized implementation of the inductive design exploration method

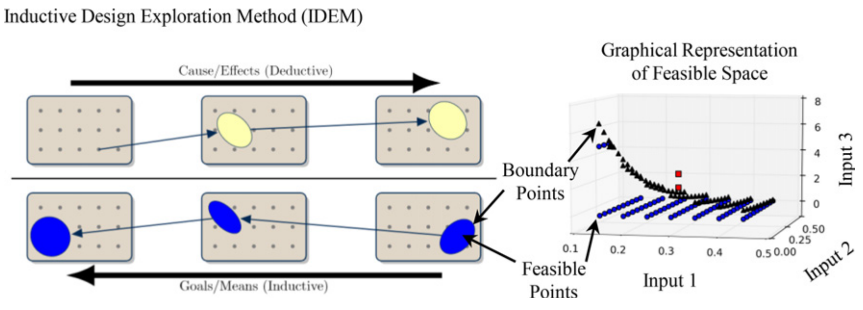
Paul C. Kern[a], Matthew W. Priddy[b], Brett D. Ellis[c], David L. McDowell[a,*]

[a] George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0405, United States
[b] Department of Mechanical Engineering, Mississippi State University, Mississippi State, MS 39762, United States
[c] Mechanical Engineering Technology, The University of Maine, 5711 Boardman Hall, Orono, ME 04469-5711, United States

HIGHLIGHTS

- An extended implementation of the Inductive Design Exploration Method for multilevel robust design of hierarchical materials.
- The IDEM framework is scripted via Python (pyDEM) to allow for widespread adaption and further alterations/modifications.
- Extended for disjoint and nonconvex feasible design spaces that satisfy ranged sets of performance requirements.
- Two test cases demonstrate the capability of pyDEM: (i) four-level UHPC panel, and (ii) WEDM process of titanium.

GRAPHICAL ABSTRACT


Inductive Design Exploration Method (IDEM)
Cause/Effects (Deductive)
Goals/Means (Inductive)
Graphical Representation of Feasible Space
Boundary Points
Feasible Points
Input 1
Input 2
Input 3

ARTICLE INFO

Article history:
Received 8 March 2017
Received in revised form 17 August 2017
Accepted 19 August 2017
Available online 24 August 2017

Keywords:
Robust design
Computational model
Modeling
Multiscale simulations

ABSTRACT

The emergence of multiscale design of materials and products has necessitated development of inductive robust design methods to rapidly develop and deploy new material systems. In addition, practical applications require robust designs which ensure performance goals are satisfied while accounting for model, noise, and control factor uncertainties. Recognizing the utility of a robust platform for design exploration, the Python Design Exploration Module (pyDEM) has been developed. The purpose of this work is to present this improved, generalized implementation of the Inductive Design Exploration Method (IDEM) to support integrated multiscale materials, process, and product design. The capabilities of pyDEM are highlighted and demonstrated via two test cases: (i) four-level Ultra High Performance Concrete (UHPC) panel and (ii) wire electric discharge machining (WEDM) process of titanium.

1. Introduction

As outlined by the Materials Genome Initiative (MGI) [1], new and improved material deployment can take 20 years or more from discovery to commercial production. Materials development typically employs trial-and-error strategies, along with empirical relationships, and a sequential design and deployment protocol with limited iteration due to a reliance on experiments. Olson [2] distinguished inductive and deductive decision paths commonly employed in materials development. The deductive path is referred to as a cause-and-effect path (bottom-up experiments and/or models) and the inductive path is the goals-means (top-down) path. Whereas scientific experiments and modeling protocols seek to determine accurate

* Corresponding author.
E-mail addresses: pkern3@gatech.edu (P.C. Kern), mwpriddy@me.msstate.edu (M.W. Priddy), brett.ellis@maine.edu (B.D. Ellis), david.mcdowell@me.gatech.edu (D.L. McDowell).

http://dx.doi.org/10.1016/j.matdes.2017.08.042
0264-1275/© 2017 Elsevier Ltd. All rights reserved.

deductive paths, these approaches are too "hit and miss" to efficiently search inductive decision paths for process-structure and structure-property relationships that satisfy multiple performance requirements. A significant thrust of MGI seeks to utilize approximate inverse methods with computational tools to produce concurrent and iterative design processes.

One such computational tool is Dream.3D, which provides a full featured environment for synthetic microstructure instantiation based on information obtained from materials characterization. This open-source tool provides a more comprehensive suite of tools than most small research labs could maintain while allowing for custom functionality via a plugin architecture [3]. Free, open source tools such as Dream.3D are invaluable to increasing the rate of scientific and engineering advances by reducing redundant software development and allowing for more rapid community adoption of new problem-solving techniques. This paper introduces a new, open source software package to improve a previously developed inductive robust design approach [4] for materials that satisfy specified ranged sets of performance requirements.

Robust designs should be considered more desirable than single point, optimal solutions due to the uncertainties associated with design variables, models, and propagation across multiple levels of design. Three robust solution categories describe performance obtained by minimizing variation of different elements of the solution space: (i) Type I robust solutions [5] seek minimization of variation due to noise factors, (ii) Type II robust solutions [6] seek minimization of variation due to control factors, and (iii) Type III robust solutions [4] seek minimization of variation due to the uncertainty inherent to the model. Each of these types of uncertainties can occur in robust design of materials and processes, and must be considered in providing decision support.

A multi-level, robust design method that considers uncertainty propagation, such as that arising in material design problems incorporating process-structure-property relationships, is the Inductive Design Exploration Method (IDEM) [4]. IDEM is a recursive process that links multiple design levels to determine the combinations of input parameters that provide robust solutions to the ranged set of requirements specified for top-level performance parameters. As illustrated in Fig. 1, the experimental or model-based mappings in steps (i) and (ii) are performed in a bottom-up manner, while the top-down, inductive compromise selection of solutions is undertaken in step (iii). Accordingly, IDEM combines, in a practical way, bottom-up projections with imposition of top-down design search for feasible regions that satisfy ranged sets of performance requirements at the top level. This approach is compelling since process-structure and structure-property relations are typically highly complex, nonlinear, non-equilibrium, and high dimensional in nature and therefore are not amenable to direct inversion. IDEM emphasizes bottom-up mappings of process-structure and structure-property relations, along with projection of uncertainty of these mappings. Then, the top-down design search in step (iii) can be undertaken in the context of robust multi-objective approach, presenting the designer with a Pareto front of solutions. The concept is straightforward, and examples will be used as a means to clearly convey the methodology later in this paper as well as the in-depth example given by Ellis and McDowell [7].

For m objective functions, performance parameters may be considered as the feasible space in m-dimensions for the top design level $p = l$. At each level p, a three-step process is performed to determine the feasible region at the $p-1$ level: (i) the $p-1$ level is sampled at multiple nominal input configurations $\bar{x}$, (ii) each $\bar{x}$ is projected onto the p level as a range of outputs determined by the input and mapping function uncertainties, and (iii) this output range (and associated $\bar{x}$) is either accepted or rejected based on an error margin defined with regard to the bounds of the feasible region at level p [4,8,9]. The accepted nominal inputs are considered to be feasible points, and the rejected inputs are infeasible with respect to robust satisfaction of the performance requirements. The feasible space is some minimum hypervolume of unknown shape (or multiple, disjoint shapes) which contains all feasible points and does not contain any infeasible points. The newly determined feasible region of inputs in level $p-1$ may be used to repeat the aforementioned process to determine the feasible region of the $p-2$ level of design and so on, until feasible regions are found for input variables associated with mappings at all levels $p \in [1, l]$. It is entirely possible that no feasible points exist to meet a given set of performance requirements (in this case expressed as property targets). In such cases, we consider design exploration as a feasibility study that facilitates either reframing of the performance requirements or the need to assess and compare competing design system alternatives.

The goal of this work is to contribute a Python-scripted framework for IDEM and to document its underlying functionality. This software package, pyDEM (Python Design Exploration Module) [10], implements a fully general IDEM instantiation that allows for the linkages of n input variables, m objective functions, and l levels or mappings, where n, m, and l are positive integers. The specific contributions from this work in developing pyDEM are threefold: (i) development of an efficient, open-source implementation of IDEM, (ii) improved feasible space representation relative to the initial conception of IDEM [4], and (iii) generalized input definitions. Each of these improvements will be discussed in detail and the improved functionality of this implementation demonstrated through the use of two test cases: (i) a four-level ultra-high performance concrete design scenario [7] and (ii) optimization of wire electric discharging machining (WEDM) of titanium [11]. The resulting software produced from this work is provided with an MIT License on GitHub as part of the MATIN platform [10].

2. Methods

In IDEM, a mapping or projection step must be performed to relate every nominal input in the $p-1$ level to a region in the p output level. The projection to the output space from a nominal input $\bar{x}$ to nominal output $\bar{y}$ is represented by $\bar{y} = \langle f_1(\bar{x}), f_2(\bar{x}), \ldots, f_m(\bar{x})\rangle$, where m is the number of output functions, i.e., the dimension of the output space. For example, these mappings can consist of experiments, data correlations, analytic theory forms, or surrogate models. Each mapping function f_i is bounded by a set of functions such that $f_{i,lower}(x) \le f_i(x) \le f_{i,upper}(x)$, thus incorporating uncertainties from the regression process. These bounds can be established, for example, using Gaussian pseudolikelihood statistics or other means. For m output functions, an $m \times 3$ matrix $\mathbf{Z}$ can be constructed to describe the nominal and bounding functions with components z_{ij} being the jth bounding function of the ith output dimension. A hyper-rectangle approximation of the boundary of the mapped output region can thus be constructed using Eq. (1). Observed noise in data used to produce bounding functions implicitly incorporates Type I robustness. Type II robustness is incorporated into the variation of the input parameters Δx_k along each kth input dimension. Type III robustness is incorporated via the inclusion of z_{ij} bounding functions. Together these values describe the total assumed variability in the ith dimension Δy_i i.e.,

$$\Delta y_i = \begin{cases} \left|\max\limits_j \left(z_{ij}(\bar{x}) + \left|\frac{\partial z_{ij}}{\partial x_k}\Delta x_k\right|\right) - \bar{y}\right|, & b_i > \bar{y}_i \\ \left|\min\limits_j \left(z_{ij}(\bar{x}) - \left|\frac{\partial z_{ij}}{\partial x_k}\Delta x_k\right|\right) - \bar{y}\right|, & b_i \le \bar{y}_i \end{cases} \tag{1}$$

The boundary point $b_i \in S$ where S is the $(m-1)$-dimensional boundary of the feasible space Ω, is selected to minimize the distance required to reach the boundary along the ith dimension (Fig. 2).

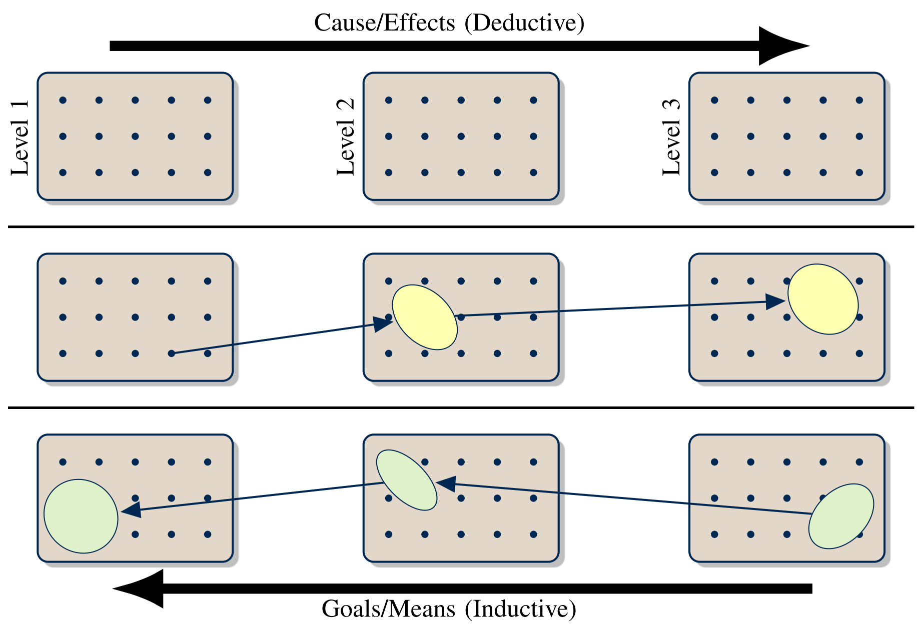


Fig. 1. Comparison of deductive and inductive design methods as applied multiple levels of design with associated uncertainties.

pyDEM selects b_i using a binary search along the path $\langle\bar{y}, \max(S_i)\rangle$. This binary search is implemented as an improvement (and an addition) to the initial proposal of the IDEM by Choi et al. [4], which utilized a gridded space with no explicit representation of the associated hypervolume. In that work, computation of the distance to the boundary was performed along the nearest neighbor points within this discretized space. While this is sufficient in the case of a fine discretization of Ω, expensive objective functions may limit the density of points relative to the output uncertainty, Δy_i. Note also how b_i^* may be selected instead of the exact b_i in the event that the boundary is represented by a linear approximation, and how the nearest boundary point projected along $i=1$ would indicate the distance to the boundary is less than the output deviation Δy_1.

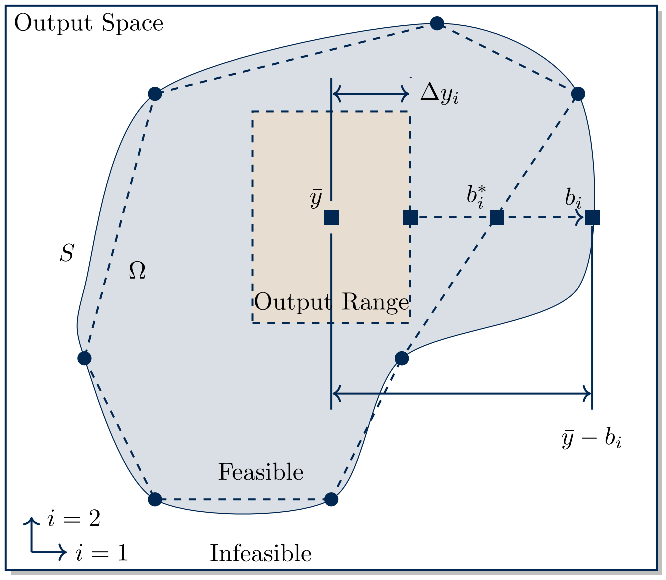


Fig. 2. Schematic of the Hyper-Dimensional Error Margin Index (HD_{EMI}) showing the range of outputs for a nominal input value and distance computations for the nominal boundary S (solid) and discretized boundary (dashed).
Source: Adapted from [4].

The next step in IDEM is the computation of an error margin to determine if the output range for $\bar{x}$ satisfies the performance requirements, and thus may be considered a robust solution. The error margin must have a consistent threshold for accepting or rejecting a potential solution and be defined for all i dimensions (where i is a positive integer) in the input and output spaces as explained by Choi et al. [4]. Several error margins are currently implemented in pyDEM, the first of which is the Hyper-Dimensional Error Margin Index (HD_{EMI}) which is defined by

$$HD_{EMI} = \begin{cases} \min\limits_{i} \left[\frac{\|\bar{y}-b_i\|}{\Delta y_i} \right], & \text{for } \bar{y} \in \Omega \\ -1, & \text{for } \bar{y} \notin \Omega \end{cases} \quad (2)$$

$HD_{EMI} < 1$ indicates that the distance to the boundary is less than the output range in one or more dimensions, i.e., a portion of the output range lies outside the feasible region. In the current implementation, the distance computation is only performed from the centroid of the predicted output space for each dimension in the m-dimensional output space. This can lead to erroneous results for feasible regions that contain sharp features near the nominal output, e.g., Fig. 2. Eliminating this scenario requires parameterization of the search for the closest b_i over the range of outputs $y_{j\neq i}$, a potentially costly procedure which must be repeated for all $i \le m$ dimensions.

While HD_{EMI} is useful in that it provides both a threshold for accepting a robust solution, as well as a relative measure of robustness, it also tends to be computationally expensive in implementation due to the necessity of computing the distance to the nearest boundary within the feasible space. In addition, as previously mentioned, less complete implementations of HD_{EMI} may mistakenly quantify some infeasible regions as feasible. We propose the use of two new, simpler metrics to determine the robustness of candidate solutions which address the aforementioned shortcomings of HD_{EMI}. Both metrics are Boolean in nature and evaluate to True only if all provided points in a list are within the feasible space. The first of these metrics, the Valid Output Region (*VOR*), examines the vertices of the output region to see if all vertices are within the feasible space. If yes, the solution may be considered robust; if not, the solution is not considered robust. The second metric, the Maximum Independent Variation (*MIV*), is similar to the HD_{EMI} in that only the perturbation of a single output parameter is considered at a time, i.e., only the midpoint of each face of the output region is examined. Both the *MIV* and *VOR* methods are significantly more computationally efficient than the HD_{EMI}. Boundary intersection for distance computation will require ray tracing over the discretized feasible space, thus runtime will depend on the number of samples taken as discussed in the following section. By contrast, *MIV* is only a single evaluation per face or $2m$, while *VOR* will scale poorly at 2^m but still tend to result in fewer evaluations than HD_{EMI} while $m < 7$. For example, in the UHPC Level 1 test case, HD_{EMI} takes 97 s, *VOR* 17 s and *MIV* 16 s for a gridded exploration space of 13,600 points. As in HD_{EMI} computation, there is the potential for the finite number of samples in both *MIV* and *VOR* to erroneously conclude that an output region is robust. It should be verified that the output regions produced by potential uncertainties will be smaller than any "holes" or interior infeasible regions (for cases where Ω is not simply connected) within the feasible space prior to utilizing the design exploration methods in pyDEM.

2.1. Hull of feasible points

When searching for a robust solution, it is often necessary to identify the boundary of the feasible space. In the case of linked scales, the level of refinement for the feasible space in the m-dimensional output space will impact the value of the HD_{EMI} since computation depends upon the boundary point b_i located in each dimension. This discretization error is inherent in multiscale design and is distinct from the aforementioned Type I, II, and III robust design strategies since it cannot be considered an input, noise, or modeling uncertainty.

In a previous version of this algorithm [7], a Delaunay Triangulation (convex hull) has been used to describe this feasible space for further exploration at subsequent design levels. This approach neglects potential concavities and multiple, disjoint domains which may compose the multidimensional feasible space. A method of determining a concave and/or disjoint volume from a supplied point cloud is desired. Of particular note are α-shapes which may be used to remove a simplex (generalization of triangle or tetrahedron to n-dimensions) from the set of simplices defining a convex hull to better represent the underlying concavity of a point-cloud [12]. This method is predicated upon the ability to discern a threshold for the α parameter (maximum edgelength of a simplex) where simplices with $\alpha > \alpha_{thresh}$ are not included in the α-shape. This threshold is usually set via the assertion of human intuition, and thus, not applicable to automated design algorithms.

Instead, the structured nature by which the feasible space is explored in pyDEM may be used to develop an accurate definition of the boundary surface. A potential feasible point is explored at each unique combination of input variables. These points form the vertices of hyperrectangular regions bounded in each dimension by consecutive inputs for the corresponding input variable. For each hyper-rectangular region, or "cell", two possibilities exist: (i) all vertices were found to be feasible points, in which case any point within the cell is also within the feasible space; or (ii) at least one vertex was found to be infeasible, in which case a convex hull adequately describes the feasible and infeasible regions within the cell. It is trivial to determine if a query point lies within the cell for the first case since the cell boundaries are hyperplanes with normal vectors along each orthogonal basis vector with known intervals. The second case may be further subdivided into cases where a boundary point was discovered between the infeasible and feasible vertices, or where such exploration was not performed. In either case, a concave hypervolume cannot arise since all points are located along the cell edges. Ultimately, this approach reduces, but does not eliminate, the discretization errors associated with the representation of the surface S that encloses a feasible region Ω. Fundamentally, this error is still limited by the initial discretization of the feasible space and thus represents a classic computational efficiency trade-off of accuracy and expense.

Runtime is reduced by use of a hashmap to look up the constituent feasible and boundary points for a given cell index in $O(1)$ time. This requires $O(k2^d)$ storage for k points and d dimensions and the local convex hulls may be dynamically computed in $O(d2^d)$ time for $d \leq 3$ and $O([d2^d]^{\lfloor d/2 \rfloor})$ for $d > 3$ [13]. Interior, infeasible regions are easily detected using ray-tracing within the discretized space. Again, ordered structure allows for trivial traversal of the feasible space. A feasible cell may be traversed with a single $O(1)$ lookup and the localized convex hull utilized to detect feasible boundary intersection for a cell with some non-feasible vertices as demonstrated in Fig. 3. Finding the correct intersection is also $O([d2^d]^{\lfloor d/2 \rfloor})$ [13]. Ray tracing continues until the ray is stopped in the interior of a cell or the edge of the explored space is reached. This ray-tracing allows for easy computation of HD_{EMI} and queries to determine if a region lies within the feasible space. In addition, since only local information is used to traverse the space, disjoint regions and holes are represented within a single *ConcaveBoundary* object while providing the greatest possible accuracy for a given discretization of the feasible space exploration.

2.2. Generalized framework

To improve the overall utility of pyDEM, flexible inputs were developed using the Object Oriented nature of Python. A summary of the currently supported input features appears in Fig. 4. In this work, *functions* and *classnames* from pyDEM will be referenced with italicized text. Both discrete and continuous function inputs are supported via the *AnonymousFunction* class. *AnonymousFunction* objects must be able to evaluate the nominal output $\bar{y}$ and the output ranges based on a nominal input $\bar{x}$ and its uncertainty. Interpolation and regression models can be directly applied by the use of a *NumericFunction* object. Partial derivatives for these numeric functions are computed by the package numdifftools [14]. In addition, symbolic equations are supported by a *SymbolicFunction* object which utilizes the Sympy package [15]. Evaluation of the partial derivatives of a symbolic function in Python is generally quicker than evaluating numeric derivatives, and thus *SymbolicFunction* objects are generally preferred over the *NumericFunction* objects. Discrete functions may be defined using a custom *AnonymousFunction* which, e.g. use input

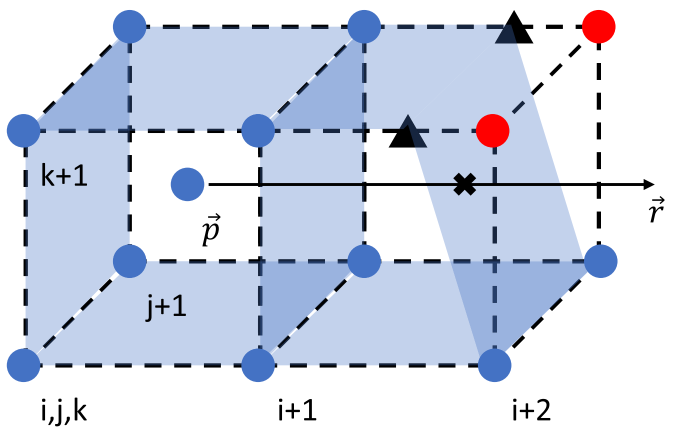


Fig. 3. Illustration of ray tracing to the nearest external boundary through two adjacent cells in the discretized feasible space with feasible query point $\vec{p}$ and ray $\vec{r}$. Blue circle points are feasible while black triangles represent the discovered boundary between feasible and infeasible vertices whose x, y, and z points index the initially provided vectors with i, j, and k respectively.

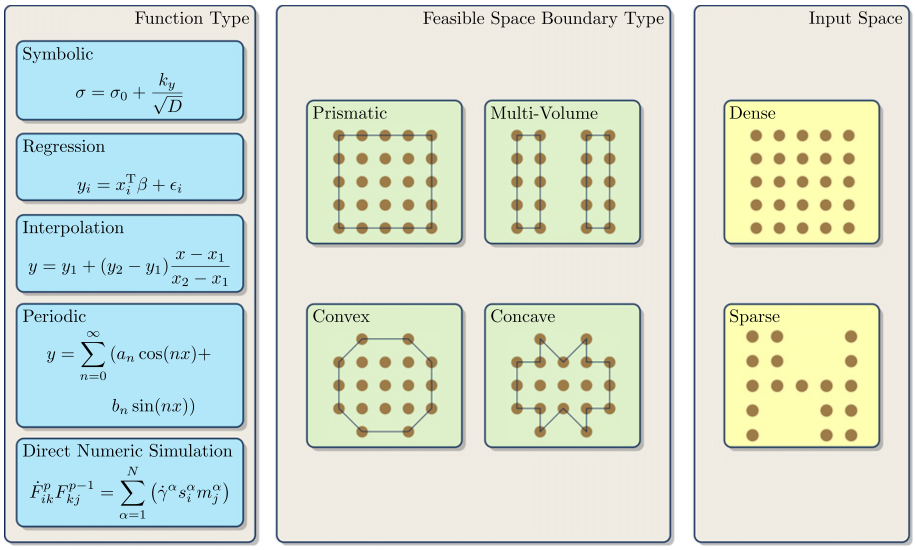

Function Type
Symbolic
$\sigma = \sigma_0 + \frac{k_y}{\sqrt{D}}$
Regression
$y_i = x_i^T\beta + \epsilon_i$
Interpolation
$y = y_1 + (y_2 - y_1)\frac{x - x_1}{x_2 - x_1}$
Periodic
$y = \sum_{n=0}^{\infty}(a_n \cos(nx) + b_n \sin(nx))$
Direct Numeric Simulation
$\dot{F}^p_{ik}F^{p-1}_{kj} = \sum_{\alpha=1}^{N}(\dot{\gamma}^\alpha s^\alpha_i m^\alpha_j)$
Feasible Space Boundary Type
Prismatic
Multi-Volume
Convex
Concave
Input Space
Dense
Sparse

Fig. 4. The various function types, feasible space boundary types, and input spaces available in pyDEM.

uncertainties and probability distributions to return an expected outcome of the mapping function as Δy_i.

Boundaries S of the feasible space Ω and objective function constraints are represented by *Boundary* objects. *ConcaveBoundary* objects may represent simply connected, multiply connected, and disconnected convex and concave hypervolumes as introduced in the previous section. *PrismaticBoundary* objects represent a simple hyperrectangle and are used in this text to provide the threshold and ranged objective values for the examined test cases. Finally, *MultipleBoundary* objects represent combinations of boundary regions via set operations: union, intersection, and difference. Recursive use of these operations may define arbitrary feasible regions. *MultipleBoundary* objects are particularly useful in material design problems because they can handle disjoint spaces. *ConcaveBoundary* objects are created as the feasible space representation at each design level in pyDEM and have functionality to identify and split discontinuous spaces. All *Boundary* classes implement the following functions: (i) *is_inner* to determine if a queried point (or list of points) lies within the represented hypervolume, (ii) *bound_dist* to return the distance to the nearest boundary for both increasing and decreasing directions along each dimension, as well as (iii) *find_boundary* to the return the nearest boundary location between two locations in the m-dimensional space.

In an effort to further reduce the total computation time, pyDEM allows the user to define a sparse, n-dimensional input space based on regions which are known a-priori to be infeasible. These points should reflect realistic constraints on the range of input variables (e.g., a machine will not run for specific combinations of input parameters). A supplied *Boundary* object may be used to short circuit the error margin calculation by applying the *is_inner* function of this input *Boundary* and not evaluating $\bar{x}$ for which the result is True. By not evaluating the robustness of $\bar{x}$, computation of $\bar{y}$ as well as the creation of additional simplices (and potential evaluation of additional simplices for concavity) are also not performed, thus improving performance. Following boundary determination with the reduced set of inputs, *is_inner* is computed for all simplex centroids, as in the concavity check, and simplices intersecting the constraint region are similarly excluded from the feasible region.

With the appropriate definition of the inputs, error margins, boundary exploration, and volumetric representations, pyDEM logic may be summarized in the following manner. The discretized input space is explored for robust solutions based on the input feasible region and error margin. Following the evaluation of this input space, a binary search is used to determine the location of the feasible region boundary located between pairs of satisfactory and unsatisfactory (as determined by the robustness criteria) nominal inputs. Finally, the volumetric representation of this feasible space is constructed as a *ConcaveBoundary* and returned for use in dependent design levels.

Overall, codifying IDEM in Python resulted in an efficient, compact code (approximately 700 lines excluding the substantial plotting functionality) which should encourage rapid adoption of the software and principles outlined in this work. The use of multi-platform, free, open source software (FOSS) components has enabled pyDEM to be a FOSS suitable for development and distribution on multiple operating systems.

3. Test cases

Application of pyDEM is first demonstrated on the design of ultra-high performance concrete (UHPC) material for a blast resistant panel application. Design of specific attributes of material structure at various length scales is pursued. These attributes include volume fraction of porosity at the μm length scale, crack-bridging fibers at the mm length scale, and panel geometry at the m length scale. Variables that relate to material composition as inputs to the bottom level process-structure mappings include volume fractions of silica fume, cement, water to cementitious material ratio, and fibers. In addition, the fiber pitch (twisted fibers) is a design variable. These design variables at various scales constitute a well-defined, 4-level test case. Top level ranged sets of requirements are specified for quasi-static tensile strength, energy dissipation density, and panel thickness. For additional information on the material, design problem background, and response surfaces used in this manuscript please refer to [7].

Blast panel resistance is computed as function of T°, E_{dis}, and the design parameter t_{panel} (Table 1). Constraints on the performance requirements specified at Level 4 are passed in as a *PrismaticBoundary* object with a minimum bound of 1.5 MPa-ms for the estimated blast impulse loading survivable by the panel. The bounding matrix is constructed from the estimated survivable loading $z_{11} = -0.857 + 0.0262t_{panel} + 6.51 \times 10^{-4} t_{panel}T^{\circ} + 4.22 \times 10^{-4} t_{panel}E_{dis}$ and the upper and lower bounding functions from the fitting procedure $z_{12} = 1.1z_{11}$ and $z_{13} = 0.9z_{11}$. *SymbolicFunction* objects are used to define these polynomial mappings. The resultant feasible space is convex; thus, no differences in the boundary representation will occur due to the concavity checks.

Table 1
Input and output variables with limits for the UHPC case study.

Parameters	Lower limit	Upper limit	Interval	Units
Quasi-static tensile strength (T°)	10	22	2	MPa
Energy dissipation density (E_{dis})	20	105	5	MPa-mm
Panel thickness (t_{panel})	39	63	6	mm
Matrix tensile strength (f_t)	5	12	0.5	MPa
Axial length per revolution ($pitch$)	6	36	3	mm
Fiber volume fraction (V_f)	0.5	2	n/a	%
Water to cementitious material ratio (w/cm)	0.15	0.45	0.05	1
Volume fraction of pores (V_{pore})	0.01	0.51	0.05	1
Mean pore radii (r_{pore})	0.1	30.1	5	nm
Curing temperature (T_{cure})	20	90	n/a	$^{\circ}$C
Volume fraction cement (V_{cem})	0.1	0.3	0.02	1
Volume fraction silica fume (V_{SF})	0.03	0.08	0.01	1

Panel design at Level 3 as function of the f_t, t_{panel}, V_f, and $pitch$ is thus constrained by the selection of the Level 4 feasible space, with V_f held at 0.019% to produce 3D (instead of 4D) plots and for direct comparison with previous work [7]. The mapping functions are $z_{11} = E_{dis} = 0.166 + 4320V_f - 62.4 pitch V_f$, $z_{21} = T^{\circ} = 3.47 + 0.569(1 - V_f)f_t + 4830\,\langle V_f - 8.66 \times 10^{-3}\rangle\, pitch^{-0.937}$, and $z_{31} = t_{panel}$, where $\langle\ \rangle$ are Macaulay brackets signifying $\langle x \rangle = 0.5\,(x + |x|)$. Fitted functions were assumed to have a constant percentage of error $z_{12} = 0.9z_{11}$, $z_{13} = 1.1z_{11}$, $z_{22} = 0.75z_{21}$, and $z_{23} = 1.25z_{21}$. Of particular interest, however, is the feasible space of Level 2. Since the inputs to the Level 3 mostly represent the physical design parameters, it is assumed that V_f and t_{panel} are selected such that the matrix tensile strength $f_t \geq 8$MPa. This becomes the lone constraint on the level 2 design space of r_{pore}, V_{pore}, and w/cm with the bounding functions $z_{11} = f_t = 0.177(99.3\,(1 - V_{pore})\,/\sqrt{r_{pore}}\,(w/cm))^{0.74}$, $z_{12} = 1.22z_{11}$, and $z_{13} = 0.814z_{11}$. This design space is found to have significant concavities, thus deviating from the convex representations employed in previous implementations of IDEM. The convex representation will greatly overestimate the feasible space, as shown in Fig. 5, which would lead to erroneous conclusions of robustness for non-robust solutions in the dependent design level (Level 1).

The Level 1 design spac;e of V_{SF}, V_{cem}, and w/cm is subject to the objective functions $z_{11} = w/cm$, $z_{21} = V_p = -3.98 \times 10^{-3} - 1.67 \times 10^{-4}T_{cure} + 0.201V_{cem} + 0.193V_{SF} + 0.298w/cm - 7.61 \times 10^{-4}T_{cure}V_{cem} - 7.35 \times 10^{-4}T_{cure}V_{SF} - 1.98 \times 10^{-3}T_{cure}w/cm - 0.315V_{cem}^2 - 0.558V_{cem}V_{SF} + 1.37V_{cem}w/cm + 1.08V_{SF}w/cm - 0.165(w/cm)^2$, and $z_{31} = r_{pore} = 70.9 - 0.76T_{cure} - 71.5V_{cem} - 91.1V_{SF} - 16.8w/cm + 1.1T_{cure}V_{cem} + 1.33T_{cure}V_{SF} + 0.307T_{cure}w/cm - 31.9V_{cem}^2 - 309V_{cem}V_{SF} - 65.2V_{cem}w/cm - 64.5V_{SF}^2 - 78.3V_{SF}w/cm$. Functional uncertainties of 10% and 15% were assumed, respectively. The robust outputs satisfying the constraint of the Level 2 feasible for a matrix tensile strength of $f_t = 8$MPa and curing temperature (T_{cure}) = 90°C are depicted for two cases in Fig. 6. It is apparent that the Level 1 feasible space as a function of these mappings is greatly restricted as the Level 2 bounds change from a convex to concave representation. Note that both spaces share a common vertex in the far corner of Fig. 6, however the translucent red surface (constrained by convex Level 2)

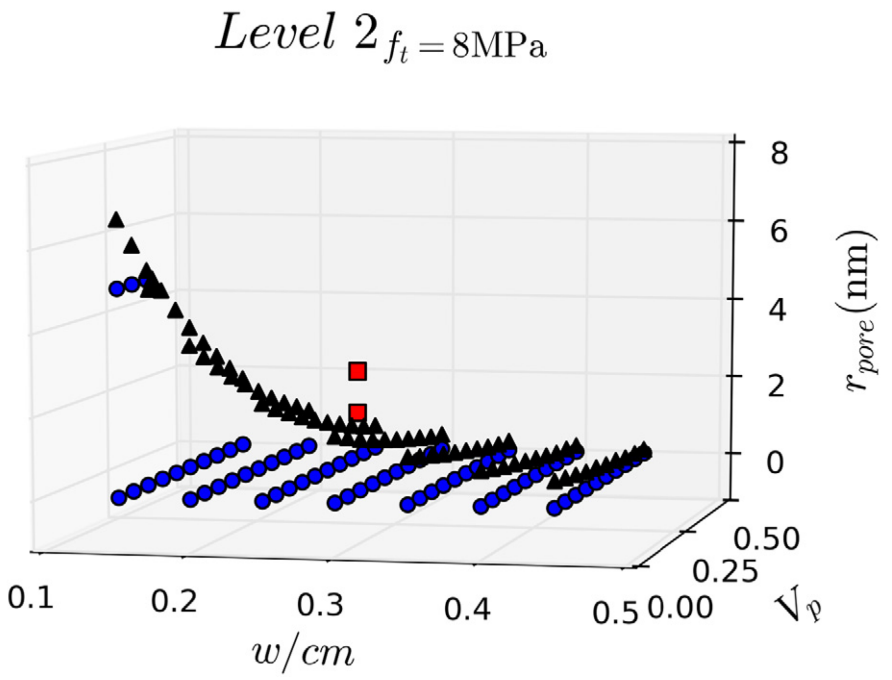


Fig. 5. Feasible space for Level 2 in terms of key material composition variables and pore size. Circles in blue are feasible, and black triangles lie on the boundary between the feasible and infeasible spaces. The red squares would be erroneously included in a convex representation of this feasible space.

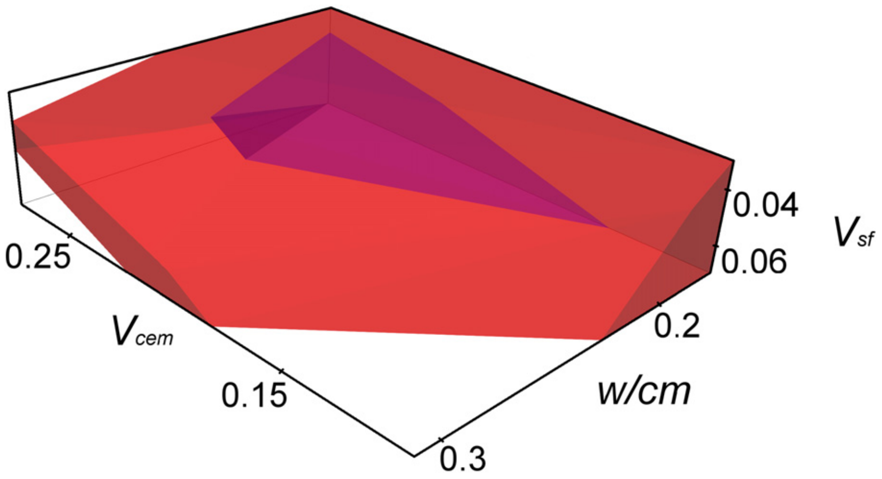


Fig. 6. Level 1 UHPC feasible spaces as a function of concave and convex representations of the feasible space of Level 2. The results from the concave Level 2 representation are a subset of the convex results, as indicated by the translucent blue surface being tinted magenta by the enclosing red translucent surface.

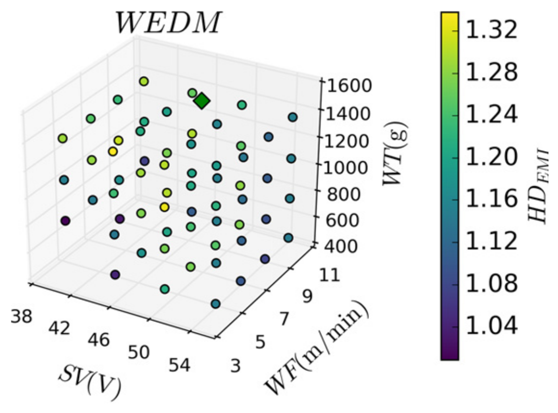
extends farther along all design variable axes, fully containing the feasible space constrained by the concave Level 2 feasible space. The magenta shading of this second surface is caused by the translucent blue surface being interior to the translucent red surface.

To demonstrate design exploration and visualization of more than three objective functions, a second test case of wire electronic discharge machining (WEDM) process design was selected. Six input variables and four response variables represent the design space of WEDM of pure titanium [11]; the minimum and maximum values of each variable used for design exploration are in Table 2. Functions relating the input variables to the response variables may be found in Kumar et al. [11]. Relatively conservative input and functional uncertainties of 2.5% were used to reflect the best case regression errors.

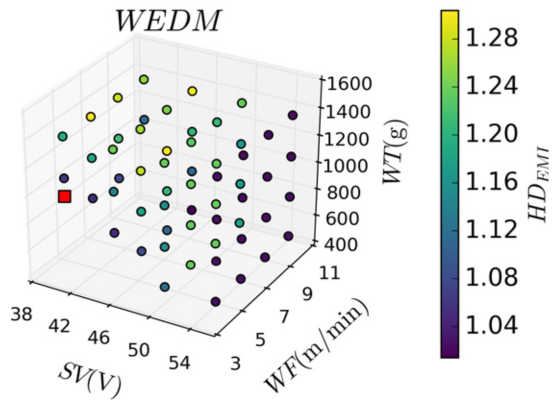
Following pyDEM execution to determine the feasible space, visualization is accomplished by displaying three-dimensional ‘slices’ holding the remaining parameters constant, as shown in Fig. 7. Selected optimized solutions (40 and 43) from [11] are shown as a green triangle and a red square along with the feasible input spaces in Fig. 7a and b. The first such optimal solution is also robust, whereas the second is not. Note how the visualization of the HD_{EMI} values yields insight to the 6D feasible space. For example, in Fig. 7a, the majority of points are comfortably above the $HD_{EMI} = 1$ boundary threshold and no obvious pattern is formed. Contrast that with the infeasible point in Fig. 7b which is clearly near two apparent boundary faces formed by several points near $HD_{EMI} = 1$. If such a non-robust solution was to be selected by a simple optimization the user may quickly find the machine to not satisfy one or more of the performance constraints due to uncertainty of inputs (machining parameters), inaccuracy of the fitted functions used for optimization, or noise factors inherent in the system. For both of these test cases with the listed intervals, it takes less than a minute for the completion of the pyDEM algorithm.

Table 2
Input and output variables with limits for the WEDM case study.

Parameters	Lower limit	Upper limit	Interval	Units
Pulse on time (T_{on})	112	120	2	µs
Pulse off time (T_{off})	44	56	4	µs
Peak current (I_p)	120	200	20	A
Spark gap voltage (SV)	40	60	5	V
Wire feed (WF)	4	10	2	m/min
Wire tension (WT)	500	1400	300	g
Machining rate (MR)	0.395	1.28	n/a	mm/min
Surface roughness (SR)	2.15	3.28	n/a	µm
Dimensional deviation (DD)	140	165	n/a	µm
Wire wear ratio (WWR)	0.048	0.107	n/a	1



(a)



(b)

Fig. 7. Illustration of two optimal points from [11] relative to the robust feasible space for WEDM example problem. An optimal, robust solution (#40) is shown in a green diamond (a), while an optimal, but non-robust, solution (#43) is shown as red square (b). Design variables $T_{on} = 112$ µs, $T_{off} = 44$ µs, and $I_p = 120$ A for (a) and $I_p = 180$ A for (b) are held constant for 3D visualization.

4. Conclusion

This work introduced the Python Design Exploration Module (pyDEM) as a generalized framework for exploring robust solutions to problems involving materials and materials process design. The contributions of pyDEM to the community include (i) open-sourcing development of an efficient implementation of IDEM, (ii) improved feasible space representation, and (iii) generalized input definitions that allow for the use of a variety of datasets (Fig. 4). Documentation of the programmatic implementation was provided to guide potential users in determining if pyDEM is an appropriate tool for their design exploration problems. A three step process for inductive design was outlined in Section 2: (i) determine feasibility of gridded input space, (ii) use iterative search methods to determine boundary points between feasible and infeasible neighbors within the gridded space, and (iii) represent the feasible space as a series of convex hulls within the discretized space. *Boundary* objects (i.e., *ConcaveBoundary*, *PrismaticBoundary*, and *MultiBoundary*) were introduced in Section 2.2 to handle generalities in the representation of hypervolume feasible spaces, while *AnonymousFunction* classes provide generality to the objective functions for which a robust solution is desired.

Finally, the utility of pyDEM has been demonstrated in Section 3 via two different design scenarios that encompass various aspects of inductive design including the linkage of the multi-level UHPC material model and the exploration of the high-dimensional input space of WEDM.

Acknowledgments

The authors would like to thank Dr. Beverly P. DiPaolo and the Engineering Research Development Center (ERDC), U.S. Army Corps of Engineers (USACE), for empirical data used to validate the multiscale ultra-high performance concrete model. We also thank Dr. Min Zhou of the Georgia Institute of Technology for suggestions regarding the multiscale modeling of cementitous materials. DLM is grateful for the support of the Carter N. Paden, Jr. Distinguished Chair in Metals Processing. DLM and MWP acknowledge the support of the US National Science Foundation under GOALI Grant No. CMMI-1333083. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Holdren, et al. Materials Genome Initiative: Strategic Plan, 6, Office of Science and Technology Policy, Washington DC, 2014, https://mgi.nist.gov/sites/default/files/factsheet/mgi_strategic_plan_-_dec_2014.pdf.
[2] G.B. Olson, Computational design of hierarchically structured materials, Science 277 (5330) (1997) 1237–1242. http://dx.doi.org/10.1126/science.277.5330.1237.
[3] M.A. Groeber, M. Jackson, DREAM.3D: a digital representation environment for the analysis of microstructure in 3D, Integr. Mater. Manuf. Innov. 3 (2014) 1–17.
[4] H. Choi, D.L. McDowell, J.K. Allen, D. Rosen, F. Mistree, An inductive design exploration method for robust multiscale materials design, J. Mech. Des. 130 (3) (2008) 031402. http://dx.doi.org/10.1115/1.2829860.
[5] G. Taguchi, Taguchi on Robust Technology Development: Bringing Quality Engineering Upstream, ASME Press, New York, 1993.
[6] W. Chen, J.K. Allen, K.-L. Tsui, F. Mistree, A procedure for robust design: minimizing variations caused by noise factors and control factors, Design 118 (1996) 478–485.
[7] B.D. Ellis, D.L. McDowell, Application-specific computational materials design via multiscale modeling and the inductive design exploration method (IDEM), Integr. Mater. Manuf. Innov. 6 (1) (2017) 9–35. http://dx.doi.org/10.1007/s40192-017-0086-3.
[8] A. Sinha, N. Bera, J.K. Allen, J.H. Panchal, F. Mistree, Uncertainty management in the design of multiscale systems, J. Mech. Des. 135 (1) (2012) 011008. http://dx.doi.org/10.1115/1.4006186.
[9] S. Jang, Y. Park, H.-J. Choi, Integrated design of aluminum foam processing parameters and sandwich panels under uncertainty, Proc. Inst. Mech. Eng. C: J. Mech. Eng. Sci. 0 (0) (2014) 1–15. http://dx.doi.org/10.1177/0954406214558341.
[10] P.C. Kern, M.W. Priddy, B.D. Ellis, pydem 1.0.0, 2017, https://github.com/materialsinnovation/pydem.
[11] A. Kumar, V. Kumar, J. Kumar, Multi-response optimization of process parameters based on response surface methodology for pure titanium using WEDM process, Int. J. Adv. Manuf. Technol. 68 (9–12) (2013) 2645–2668. http://dx.doi.org/10.1007/s00170-013-4861-9.
[12] H. Edelsbrunner, E.P. Mücke, Three-dimensional alpha shapes, ACM Trans. Graph. (TOG) 13 (1) (1994) 43–72.
[13] B. Chazelle, An optimal convex hull algorithm in any fixed dimension, Discrete Comput. Geom. 10 (4) (1993) 377–409. http://dx.doi.org/10.1007/BF02573985.
[14] P.A. Brodtkorb, J. D'Errico, numdifftools 0.9.14, Nov. 2015, https://github.com/pbrod/numdifftools.
[15] SymPy Development Team, SymPy: Python library for symbolic mathematics, 2016, http://www.sympy.org.