Direct Simulation for CAD Models undergoing Parametric Modifications

Liangchao Zhu[a], Ming Li[a,*], Ralph R. Martin[b]

[a]*State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou, China*
[b]*School of Computer Science & Informatics, Cardiff University, Cardiff CF24 3AA, UK*

Abstract

We propose a novel approach—*direct simulation*—for interactive simulation with accuracy control, for CAD models undergoing parametric modifications which leave Dirichlet boundary conditions unchanged. This is achieved by computing *offline* a generic solution as a function of the *design modification parameters*. Using this parametric expression, each time the model parameters are edited, the associated simulation solution for this model instance can be cheaply and quickly computed *online* by evaluating the derived parametric solution for these parameter values. The proposed approach furthermore works for models undergoing topological changes, and does not need any mesh regeneration or mesh mapping. These results are achieved by use of the *proper generalized decomposition* model reduction technique, in combination with R-functions. We believe this is the first approach that can interactively simulate the physical properties of a CAD model, even undergoing topological change, without expensive re-computation. The approach is demonstrated for linear elasticity analysis; numerical results demonstrate its simulation accuracy and efficiency in comparison with the classic FE method.

Keywords: direct simulation, interactive simulation, proper generalized decomposition, model reduction, parametric CAD models, feature intersections

1. Introduction

A CAD model typically goes through many design modifications, each time requiring simulation of its physical properties and functionality, before finally being manufactured into an engineered product. The model modifications are in many cases defined or controlled via various shape or feature parameters, describing their locations, sizes etc. Each time the model is modified, its physical behaviour are typically recomputed by performing classical FE analysis (or some variant) on the new model. This cycle of design and simulation is computationally expensive, involving volume mesh generation anew from the CAD model, physical solution computation, geometric mapping between the CAD model and the FE volume mesh and so on [1, 2]. To make analysis and meshing tractable, the model may also need to be simplified and features removed [3, 4]. This overall process of CAD-CAE integration occupies a significant proportion of conventional engineering design process time.

It would greatly facilitate the product design process if the model's physical properties could be rapidly and interactively (within 10 ms, for example) predicted as soon as the design model is modified—we refer to this goal as *direct simulation* here. Such direct simulation is however very challenging to achieve via the traditional complex CAD-CAE integration process, especially as the model's

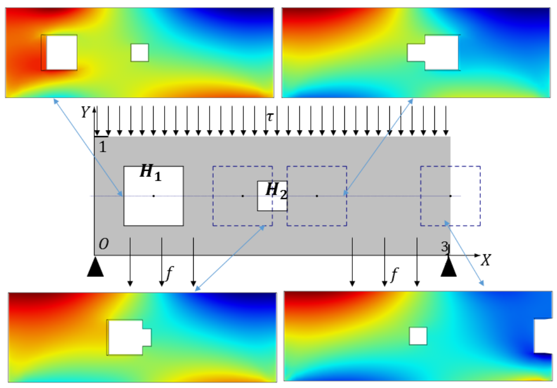

Figure 1: Direct simulation for a CAD model undergoing parametric modification. The simulation results are given rapidly and directly as hole H_1 moves from left to right.

topology may change when feature interactions occur as a result of parameter modification. Consider for example the model in Fig. 1. The original model has two square holes H_1, H_2 with side lengths 0.4, 0.2 within its interior, respectively centered at $(p, 0.5)$ and $(1.5, 0.5)$, where p is a parameter. As p changes from -0.2 to 3.2, moving the center of hole H_1 the x-direction, the resulting model undergoes topological change. Direct simulation aims to interactively predict the modified model's physical properties, for example point-wise displacement in an elasticity analysis, as the designer moves feature H_1. No existing simulation approach can directly handle such complex

*Corresponding author: liming@cad.zju.edu.cn cases involving topological change.

This challenging task of direct simulation for CAD models undergoing parametric modifications cannot readily be solved by classical model reduction techniques, such as POD (proper orthogonal decomposition) [5], subspace methods [6], or PCA (principle component analysis) [7]. Such techniques essentially approximate the target physical property space by a linear space using a basis set, using e.g. the eigenvectors of some sampled design space. However, such technologies have various intrinsic limitations. The first lies in the well-known *curse of dimensionality*. Consider for example, a design space with ten design variables, each sampled 10 times. The total number of samples in the space is 10^{10}, whose associated engineering analysis solutions are far too many to compute. For example, Kim et al [8], used several thousand CPU-hours to perform a limited exploration of the space of detailed clothing effects on a character. Moreover, the size of the simulated results was very large: tens of gigabytes of raw data. Secondly, as the physical space is discretely sampled, it is quite possible that some key physical phenomenon may not be captured by the sampling process. Thirdly, the original physical space may be very complex and intrinsically nonlinear, so approximating it using a linear space will either lose accuracy or require a large basis set. Fourthly, even after deriving the bases, computing approximations to the original problem still requires the solution of a large system of equations, which although smaller or simpler than the original problem, still takes time. Finally, we note that traditional model reduction approaches generally only work for shapes without topological changes, whereas our approach can handle such changes.

Instead, here we give a novel approach for direct simulation for CAD models undergoing parametric modifications. This is achieved by computing *offline* a generic solution as a function of the *design modification parameters*. For example, the physical solution to the model in Fig. 1 is computed as a function in terms of the translation parameter p (as well as x, y, z spatial coordinates). As hole H_1 moves in the x-direction to some new parameter value $p^1, p^2, \dots$, the physical solution for the modified model can be easily and cheaply derived *online* by evaluating the generic parametric function for the parameter value $p^1, p^2, \dots$. Note particularly that this solution expression is very different from the conventional FE analysis process which only gives a simulation solution in terms of the spatial coordinates, without the additional parametric dimension. Thus, conventionally, each time the design's geometry is modified, the overall CAD-CAE integration process has to be reperformed, which is very computationally expensive.

This approach is enabled by use of a newly introduced model reduction technique, *proper generalized decomposition* or PGD [9], allowing offline parametric solution computation. Unlike the conventional model reduction techniques as mentioned above, PGD is based on the assumption of a separated form for the unknown physical solutions (in terms of both spatial coordinates and the design parameters). It has demonstrated its abilities to deal with high-dimensional problems, and that it can overcome the limitations of classical approaches [10]. Since its first introduction by Ammar and Chinesta [11, 12], the PGD method has been applied to various linear and nonlinear engineering problems involving computational rheology [13], the chemical master equation [14], geometrically parameterized heat problems [15], etc. Here, we extend it to direct simulation for parametrically varying CAD models, which may involve topological change. PGD has not previously been applied to this problem.

Changes in the domain during parametric modification pose a big challenge, particularly topological changes. We resolve this issue by using R-functions, implicit functions that can easily represent a solid's interior, boundary and exterior. R-functions were first suggested in Russian by Rvachev in 1963 [16], and popularised by Shapiro [17]. Unlike other functions with this property, for example RBFs (radial basis functions), R-functions have the useful property that they can readily represent Boolean operations between different geometries, and can also incorporate geometric design parameters. They can thus easily describe models undergoing topological changes. Further discussion of R-functions is deferred until Section 4. By using R-functions and characteristic functions, a physical simulation problem originally defined over a set of CAD models generated by parametric variations is now redefined as a high-dimensional problem on a fixed domain. This allows the PGD computation to be readily performed generically.

Chen, Shapiro and Suresh have also considered using R-functions for design optimization with topological changes [18, 19]. Our work differs in its use of the PGD approach to permit fast simulation. We also note that [15, 20] have also proposed using PGD for fast simulation involving deformed shapes. However, their work assumes that the FE meshes used have the same topology before and after deformation, which is too restrictive to be of use in general design problems.

In summary, this paper proposes a novel approach for direct simulation for CAD models undergoing parametric deformation; we illustrate it in the context of linear elasticity. It is assumed in this paper that the model's Dirichlet boundary conditions are maintained unchanged, or, the fixed boundary is kept unchanged, during the model modification process. It can interactively predict a simulation's physical solution almost immediately after the designer changes the model's design parameters, even for models undergoing topological changes, unlike previous work. This is achieved by computing *offline* a generic solution as a function of the *design modification parameters* based on the *PGD* model reduction technique in combination with *R-functions*. It overcomes the limitations of previous model reduction approaches, avoiding large and insufficient sampling spaces, inaccurate approximations, additional online equation solutions, and so on. The proposed approach works for varying models whose Dirichelt boundary conditions (or fixed boundaries) do not change.

We demonstrate the computational accuracy and efficiency of the proposed approach, and compare it to conventional FE methods using various numerical examples.

The remainder of the paper is arranged as follows. The problem and overall approach to solve it are described in Section 2. The approach to computing the PGD solution for a CAD model undergoing parametric changes is presented in Section 3. The strategy for unifying the computational domain by using a parametric R-function is explained in Section 4. Numeric examples are presented to illustrate the validity and efficiency of our proposed method in Section 5. The paper is finally concluded in Section 6.

2. Problem statement and approach overview

The purpose of our work is to predict the simulation solution of a CAD model in boundary representation (B-rep) as users perform some parametric editing on it, e.g. moving or reshaping some geometric entities or features; see Fig. 1. We use the popular problem of linear elasticity analysis as a concrete example of our approach.

2.1. Linear elasticity problem with parameters

Suppose we have a 3D CAD model $\Omega^{\mathbf{p}} \in \mathbb{R}^3$, in the 3D Euclidean space, where $p_1, \dots, p_m$ are parameters used to modify the model geometry. The model is described by a parameter vector,

$$\mathbf{p} = (p_1, \dots, p_m) \quad \in \quad \mathbf{I_p} = I_1 \times \dots \times I_m \subset \mathbb{R}^m, \quad (1)$$

where I_j is the range of variation of parameter p_j.

Given an arbitrary design configuration $\mathbf{p}^0$, the model $\Omega^{\mathbf{p}^0} \in \mathbb{R}^3$ is deformed following the principle of (stationary) linear elasticity: we wish to find the displacement solution $\mathbf{u}^0(\mathbf{x})$ satisfying

$$P_O: \quad \begin{cases} -\mathrm{div}\boldsymbol{\sigma}^0(\mathbf{u}^0(\mathbf{x})) = f, & \text{in } \Omega^{\mathbf{p}^0}, \\ \boldsymbol{\sigma}^0(\mathbf{u}^0(\mathbf{x})) \cdot \mathbf{n} = \tau, & \text{on } \Gamma_N(\mathbf{p}^0), \\ \mathbf{u}^0(\mathbf{x}) = \mathbf{u}_D, & \text{on } \Gamma_D, \end{cases} \quad (2)$$

where the stress $\boldsymbol{\sigma}^0$ is defined via the fourth-order stiffness tensor (in matrix form) $\mathbf{C}$ as follows

$$\boldsymbol{\sigma}^0(\mathbf{u}^0) = \mathbf{C}\varepsilon(\mathbf{u}^0), \quad \varepsilon(\mathbf{u}^0) = \nabla\mathbf{u}^0(\mathbf{x}), \quad (3)$$

f is the body force per unit volume, $\mathbf{u}_D$ is a prescribed displacement on part of the boundary Γ_D,and τ is an exerted external force applied on part of the boundary $\Gamma_N(\mathbf{p}^0)$, with outer normal direction $\mathbf{n}$.

The above problem in (2) is a classical boundary value problem in $\mathbb{R}^3$. It can be solved using the conventional FE method as follows. For a specific configuration with parameter vector $\mathbf{p}^0$, we construct the CAD model $\Omega^{\mathbf{p}^0}$ and generate a corresponding volume mesh model. We then compute the displacement field $\mathbf{u}^0(\mathbf{x})$ for parameter $\mathbf{p}^0$ by the FE method. Later, suppose the parameter is changed to $\mathbf{p}^1$, so a new CAD model $\Omega^{\mathbf{p}^1}$ is generated, its associated mesh model determined, and the field solution $\mathbf{u}^1(\mathbf{x})$ found for parameter $\mathbf{p}^1$. As the user may change the parameter very many times, the above repeated procedure can be very laborious and computationally expensive, preventing interactive exploration of the design space.

Instead, direct simulation permits offline computation of a generic solution for all possible parameter vectors $\mathbf{p} \in \mathbf{I_p}$. Details are explained next.

2.2. Approach overview

In order to permit direct simulation for models undergoing parametric modifications, we reformulate the problem in (2), originally defined in $\mathbb{R}^3$, as a new one in a higher-dimensional space which includes the design parameters, $\mathbb{R}^3 \times \mathbf{I_p}$, a space of dimension $3 + m$. The problem is now defined as below: find the solution $\mathbf{u}(\mathbf{x}, \mathbf{p})$ such that

$$P_p: \quad \begin{cases} -\mathrm{div}\boldsymbol{\sigma}(\mathbf{u}(\mathbf{x}, \mathbf{p})) = f, & \text{in } \Omega^{\mathbf{p}} \times \mathbf{I_p}, \\ \boldsymbol{\sigma}(\mathbf{u}(\mathbf{x}, \mathbf{p})) \cdot \mathbf{n} = \tau, & \text{on } \Gamma_N(\mathbf{p}), \\ \mathbf{u}(\mathbf{x}, \mathbf{p}) = \mathbf{u}_D, & \text{on } \Gamma_D. \end{cases} \quad (4)$$

Note it is assumed here that Γ_N may change during the model modification process while Γ_D remains unchanged. The situation is very general as long as the fixed boundary (i.e. Γ_D) of the model $\Omega^{\mathbf{p}}$ does not change during the model modification process; other cases cannot be handled by the proposed approach.

Suppose the solution $\mathbf{u}(\mathbf{x}, \mathbf{p})$ to the above high-dimensional simulation problem is obtained, via FE for example, as a piecewise function in terms of the domain variables $\mathbf{x}, \mathbf{p}$. Direct simulation for the CAD model then becomes a simple matter of evaluating the result for a parameter vector $\mathbf{p}$ of interest. However, the high-dimensionality of (4) makes it almost impossible to solve directly using a classical FE approach, due to the well-known curse of dimensionality: increasing the number of parameters leads to an explosion in computational complexity due to the additional degrees of freedom, as will further be explained at the end of Section 3.

Resolving this challenging issue first requires us to reformulate problem (4), originally defined over a parametrically varying domain $\Omega^{\mathbf{p}}$ into a new simulation problem on a fixed domain. This is achieved by use of R-functions and a characteristic function. Specifically, let Ω^U be a domain that contains all points of $\Omega^{\mathbf{p}}$, for all $\mathbf{p} \in \mathbf{I_p}$. We define a characteristic function representing the domain for a particular choice of parameter $\mathbf{p}$:

$$H(\mathbf{x}, \mathbf{p}) := \begin{cases} 1 & \mathbf{x} \in \Omega^{\mathbf{p}}, \\ 0 & \mathbf{x} \notin \Omega^{\mathbf{p}}, \end{cases} \quad \text{for} \quad \mathbf{x} \in \Omega^U, \quad (5)$$

The concrete expression for $H(\mathbf{x}, \mathbf{p})$ is built via use of R-functions, as detailed in Section 4.

Replacing $\mathbf{u}(\mathbf{x}, \mathbf{p})$ with $\mathbf{u}(\mathbf{x}, \mathbf{p})H(\mathbf{x}, \mathbf{p})$ for $\mathbf{x} \in \Omega^U$ in (4), we now have a simulation problem defined on the fixed domain Ω^U: find the solution $\mathbf{u}(\mathbf{x}, \mathbf{p})$ such that

$$P_H: \quad \begin{cases} -\mathrm{div}\boldsymbol{\sigma}(\mathbf{u}(\mathbf{x}, \mathbf{p})H(\mathbf{x}, \mathbf{p})) = f, & \text{in } \Omega^U \times \mathbf{I_p}, \\ \boldsymbol{\sigma}(\mathbf{u}(\mathbf{x}, \mathbf{p})H(\mathbf{x}, \mathbf{p})) \cdot \mathbf{n} = \tau, & \text{on } \Gamma_N^U, \\ \mathbf{u}(\mathbf{x}, \mathbf{p})H(\mathbf{x}, \mathbf{p}) = \mathbf{u}_D, & \text{on } \Gamma_D, \end{cases} \quad (6)$$

where Γ_N^U is the union of the varying Neumann boundaries of $\Gamma_N(\mathbf{p})$.

However, the simulation problem 6, although now defined over a fixed domain Ω^U, is still of a dimension $m+3$, and intractable. This is overcome by further use of the PGD model reduction technique [11, 12].

A PGD solution is usually defined in the following form as sum of a series of separated variables:

$$\mathbf{u}(\mathbf{x},\mathbf{p}) \approx \mathbf{u}^N(\mathbf{x},\mathbf{p}) = \sum_{i=1}^{N} \mathbf{T}_i(\mathbf{x},\mathbf{p}), \tag{7}$$

with

$$\mathbf{T}_i(\mathbf{x},\mathbf{p}) := \mathbf{d}^i(\mathbf{x}) \prod_{j=1}^{m} g_j^i(p_j), \tag{8}$$

where each $\mathbf{d}^i(\mathbf{x})$ is a piecewise polynomial function of spatial coordinates $\mathbf{x} = (x, y, z)$, and each $g_j^i(p_j)$ is a function of parameter p_j, whose coefficients are to be computed. In other words, we aim to approximate the solution to Eqn. (6) using functions in the form in Eqn. (7), whose design variables and domain coordinates are independent of each other.

Note that $\mathbf{u}(\mathbf{x},\mathbf{p})$ is a now a function of the parameter variable $\mathbf{p}$, besides the space coordinate $\mathbf{x}$; furthermore, the variables are separated from each other. Such PGD solution representations are very different from conventional FE solution expressions, which are only function of spatial coordinates, generally represented in a form whose variables are not separated. The reduced representation permits an efficient numerical approach to computing such solutions, under accuracy control, as shown in [21, 22].

To numerically compute the PGD solution (7), we substitute its expression into an integral form of Eqn. (6), and build a nonlinear equation. The problem is then solved by an enrichment process, followed by a fixed-point iteration step. Each enrichment step adds a further summation term $\mathbf{T}_i(\mathbf{x},\mathbf{p})$, until convergence. In each step, a fixed-point numerical strategy is applied to compute the concrete expressions of functions $\mathbf{d}^i(\mathbf{x})$ and $g_j^i(p_j)$ to the current $\mathbf{T}_i(\mathbf{x},\mathbf{p})$ until convergence; further details are given in Section 3. The above PGD solution to problem (6) is computed offline. Once determined, online solution computation just requires function evaluation for each new choice of parameters.

The technical details are now expanded in the following sections. Section 3 first describes the numerical approach to computing a PGD-based solution in the form in (7). Approaches to constructing the characteristic function $H(\mathbf{x},\mathbf{p})$ and its use in solving problem (6) are explained in Section (4).

3. Parametric solution to parametric linear elasticity problem

We start by giving a general numerical procedure to compute the parametric displacement field $\mathbf{u}(\mathbf{x},\mathbf{p})$ in the form given in (7), for the high-dimensional problem defined in (6). For this purpose, we first assume that $H(\mathbf{x},\mathbf{p}) = 1$ for ease of explanations; its extensions to general $H(\mathbf{x},\mathbf{p})$ is straightforward and will be further explained in Section 4. The numerical procedure used follows [9, 23, 24].

The basic theory and main computational procedure in PGD is similar to that of the classical FE method [25, 26] except for the form of the solution expression (7). As in FE, we first convert the boundary value problem (6) into an equivalent integration form, leading to the following weak formulations:

$$A(\mathbf{u},\delta\mathbf{u}) = L(\delta\mathbf{u}), \tag{9}$$

where

$$A(\mathbf{u},\delta\mathbf{u}) = \int_{I_1} \cdots \int_{I_m} \int_{\Omega^{\mathbf{P}}} (\nabla\mathbf{u})^T \mathbf{C} \nabla\delta\mathbf{u} \; d\Omega dp_1 \cdots dp_m,$$

$$L(\delta\mathbf{u}) = \int_{I_1} \cdots \int_{I_m} \Big(\int_{\Omega^{\mathbf{P}}} f^T \delta\mathbf{u} \; d\Omega + \int_{\Gamma_N(\mathbf{p})} \tau^T \delta\mathbf{u} dS \Big) dp_1 \cdots dp_m,$$

$\mathbf{d}^i \in V = \{\mathbf{d} | \mathbf{d} \in \mathcal{H}^1(\Omega^{\mathbf{P}}),\ \mathbf{d} = u_D \text{ on } \Gamma_D\}$, $g_j^i \in \mathcal{L}^2(I_j)$ in (7), and $\delta\mathbf{u}$ is the test function of $\mathbf{u}$ in an appropriate functional space.

In order to compute the solution $\mathbf{u}(\mathbf{x},\mathbf{p})$ to (4) in PGD form (7), it is usually assumed that the functions $\mathbf{d}^i(\mathbf{x})$ or $g_j^i(p_j)$ are 3D or 1D FE descriptions defined over Ω and I_j in the following form:

$$\mathbf{d}^i(\mathbf{x}) = \sum_{k=1}^{N_d} {}_3\phi_k^i \mathbf{B}_k^3(\mathbf{x}),\ \ g_j^i(p_j) = \sum_{k=1}^{N_{p_j}} {}_1\phi_{j,k}^i \mathbf{B}_{j,k}^1(p_j), \tag{10}$$

where $\mathbf{B}_k^3(\mathbf{x})$ and $\mathbf{B}_{j,k}^1(p_j)$ are 3D and 1D FE bases for $\mathbf{d}^i$ and g_j^i respectively, N_d, N_{p_j} are the numbers of degrees of freedom for the corresponding FE mesh, and ${}_3\phi_k^i, {}_1\phi_{j,k}^i$ are coefficients to be determined ($j = 1, \ldots, m$). For simplicity, we now write

$$\boldsymbol{\phi}_d^i = ({}_3\phi_1^i, \ldots, {}_3\phi_{N_d}^i), \quad \boldsymbol{\phi}_{g_j}^i = ({}_1\phi_{j,1}^i, \ldots, {}_1\phi_{j,N_{p_j}}^i). \tag{11}$$

Determining the coefficients $\boldsymbol{\phi}_d^i$ and $\boldsymbol{\phi}_{g_j}^i$ is based on the fact that the PGD solution (7) has to satisfy the weak formulation (9). This is a highly nonlinear equation system. The numerical algorithm used to solve it consists of a greedy enrichment procedure with an iterative fixed-point procedure as already noted, following [9, 23]. Specifically, each enrichment step adds a further term $\mathbf{T}_i(\mathbf{x},\mathbf{p})$ (defined in (8)) until a convergence criterion is satisfied; the iterative fixed-point procedure is used to compute a concrete expression for each $\mathbf{T}_i$ by iteratively computing the FE coefficients (11) for one specific function by fixing the coefficients of all the other functions, until convergence. This is done by following the steps below.

Suppose that, after the $(i-1)$-th enrichment step, we have a solution $\mathbf{u}^{i-1}(\mathbf{x},\mathbf{p})$, which is not accurate enough. We thus enrich it by adding an extra term

$$\mathbf{T} := \mathbf{d}(\mathbf{x}) \prod_{j=1}^{m} g_j(p_j),$$

in the form given in Eqn. (10), to improve the approximation. We thus want to find a solution $\mathbf{u}^i(\mathbf{x},\mathbf{p})$ in the following form

$$\mathbf{u}^i(\mathbf{x},\mathbf{p}) = \mathbf{u}^{i-1}(\mathbf{x},\mathbf{p}) + \mathbf{d}(\mathbf{x}) \prod_{j=1}^{m} g_j(p_j), \qquad (12)$$

where $\mathbf{d}(\mathbf{x})$ and $g_j(p_j)$ $(j = 1,\dots,m)$, or their coefficients $\boldsymbol{\phi}_d$, $\boldsymbol{\phi}_{g_j}$ defined in (11), are to be determined in the i-th enrichment procedure.

Substituting (12) into (9), the weak problem becomes

$$A(\mathbf{d}\prod_{j=1}^{m} g_j, \delta\mathbf{u}) = L(\delta\mathbf{u}) - A(\mathbf{u}^{i-1}, \delta\mathbf{u}), \qquad (13)$$

where the test function $\delta\mathbf{u}$ may also be separated as

$$\delta\mathbf{u} = \delta\mathbf{d}\prod_{j=1}^{m} g_j + \sum_{k=1}^{m} \mathbf{d}\delta g_k \prod_{j=1, j\neq k}^{m} g_j, \qquad (14)$$

$\delta\mathbf{d} \in V_0 = \{\mathbf{d} | \mathbf{d} \in \mathcal{H}^1(\Omega^{\mathbf{p}}),\ \mathbf{d} = 0 \text{ on } \Gamma_D\}$ and $\delta g_k \in \mathcal{L}^2(I_k)$ are test functions for $\mathbf{d}$ and g_k, $k = 1,\dots,m$ respectively.

Following [9, 23], an iterative fixed-point procedure is applied to iteratively compute $\mathbf{d}$ and g_k, $k = 1,\dots,m$, one by one, by fixing values of the other functions. Specifically, the procedure is:

1. Compute $\mathbf{d}$ with $g_1,\dots,g_m$ known;

2. Compute g_1 with $\mathbf{d}, g_2,\dots,g_m$ known;

3. Compute g_2 with $\mathbf{d}, g_1, g_3,\dots,g_m$ known;
......

m+1. Compute g_m with $\mathbf{d}, g_1,\dots,g_{m-1}$ known.

The above procedure is iterated until convergence, which is guaranteed [27].

Following the above, the overall numerical algorithm is summarized in Algorithm 1 for the case $m = 2$. There are two nested loops. The outer loop (lines 03–15) are the greedy enrichment strategy that adds terms $\mathbf{T} = \mathbf{d}(\mathbf{x})g_1(p_1)g_2(p_2)$ one by one until a convergence criterion is satisfied. The inner loop (lines 06–12) is a fixed-point algorithm that iteratively computes the FE coefficients for one specific function $\mathbf{d}(\mathbf{x})$, $g_1(p_1)$ or $g_2(p_2)$ by fixing coefficients of all the other functions, until convergence. The exact rank N of the approximation needed to accurately approximate the solution depends on solution separability and regularity. The overall enrichment procedure stops as soon as the equation residual or other error criterion is satisfied [21, 28, 22, 23].

Algorithm 1 Numerical approach to computing PGD solution when $m = 2$

```
01 Set maximum iterations max_iter, error tolerance
   tol, and initial solution $\mathbf{U} = \mathbf{0}$
02 initialize iteration counter num_iter = 0 and
   approximation error error_iter = 1
03 while error_iter > tol and num_iter < max do
04   num_iter = num_iter + 1
05   initialize $\boldsymbol{\phi}_\mathbf{d} = 0$, $\boldsymbol{\phi}_{g_1} = \boldsymbol{\phi}_{g_2} = 1$, error = 1
06   while error > tol do
07     $\boldsymbol{r}_\mathbf{d} = \boldsymbol{\phi}_\mathbf{d}$, $\boldsymbol{r}_1 = \boldsymbol{\phi}_{g_1}$, $\boldsymbol{r}_2 = \boldsymbol{\phi}_{g_2}$
       //record the original values for fixed point check
08     Find $\mathbf{d} \in V$ or $\boldsymbol{\phi}_d$ for all $\delta\mathbf{d} \in V_0$, such that
       $A(\mathbf{d}g_1g_2, \delta\mathbf{d}g_1g_2) = L(\delta\mathbf{d}g_1g_2) - A(\mathbf{u}^{n-1}, \delta\mathbf{d}g_1g_2)$
09     Find $g_1 \in \mathcal{L}^2(I_1)$ or $\boldsymbol{\phi}_{g_1}$ for all $\delta g_1 \in \mathcal{L}^2(I_1)$,
        such that
       $A(\mathbf{d}g_1g_2, \mathbf{d}\delta g_1g_2) = L(\mathbf{d}\delta g_1g_2) - A(\mathbf{u}^{n-1}, \mathbf{d}\delta g_1g_2)$
10     Find $g_2 \in \mathcal{L}^2(I_2)$ or $\boldsymbol{\phi}_{g_2}$ for all $\delta g_2 \in \mathcal{L}^2(I_2)$,
        such that
       $A(\mathbf{d}g_1g_2, \mathbf{d}g_1\delta g_2) = L(\mathbf{d}g_1\delta g_2) - A(\mathbf{u}^{n-1}, \mathbf{d}g_1\delta g_2)$
11     error = $||\boldsymbol{r}_\mathbf{d} - \boldsymbol{\phi}_\mathbf{d}|| + ||\boldsymbol{r}_1 - \boldsymbol{\phi}_{g_1}|| + ||\boldsymbol{r}_2 - \boldsymbol{\phi}_{g_2}||$
       //check if at fixed point
12   end while
13   $\boldsymbol{U} = \boldsymbol{U} + \mathbf{d}(\mathbf{x})g_1(p_1)g_2(p_2)$
14   error_iter = $||\boldsymbol{\phi}_\mathbf{d}||\ ||\boldsymbol{\phi}_{g_1}||\ ||\boldsymbol{\phi}_{g_2}||$ //or other error
     //estimators to check if $\mathbf{U}$ accurate enough
15 end while
16 return U
```

We further comment on the complexity of the PGD approach. Suppose the functions $\mathbf{d}(\mathbf{x})$ and $g_j(p_j)$ are discretized with N_d and N_{p_j} nodes (or degrees of freedom) respectively. The numerical complexity of the original multidimensional problem is $N_d \prod_{j=1}^{m} N_{p_j}$. But with PGD, it becomes $N_d + \sum_{j=1}^{m} N_{p_j}$. To take a concrete example, if we assume $N_d = 1000$ and $N_{p_j} = 10$ (a very coarse description in practice), and $m = 10$ (a very simple model), the numeric complexity is $1000 \times 10^{10} = 10^{13}$ for an FE approach *vs.* $1000 + 10 \times 10 = 1100$ for a PGD approach. The curse of dimensionality is resolved!

4. Using R-functions to describe domain changes

To successfully implement the above described PGD procedure to compute a parametric solution to (6) still requires constructing an R-function and a characteristic function $H(\mathbf{x},\mathbf{p})$, as described in Section 2. Details are explained in this section.

4.1. R-function with parameters

It is well known that every solid can be represented by a real-valued function f, such that $f > 0$ for all interior points, $f = 0$ for all boundary points, and $f < 0$ for all exterior points. The theory of R-functions [17, 29, 30] gives an algorithmic method for constructing such functions for general shapes in engineering.

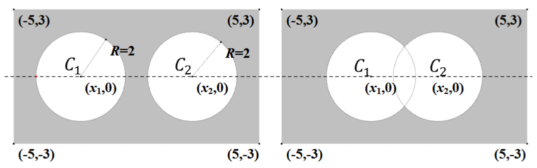

Figure 2: R-functions can uniformly express a model undergoing parameter modification even where this results in intersecting.

An R-function is a real-valued function whose sign is completely defined by the sign of its arguments. Such functions can encode Boolean operations that help to construct complex combinations of simpler basis functions. For instance, given two models Ω_1, Ω_2 represented by two R-functions f_1, f_2, the following functions behave like the logical operators **and** and **or**:

$$f_1 \wedge f_2 \equiv f_1 + f_2 - \sqrt{f_1^2 + f_2^2}$$

$$f_1 \vee f_2 \equiv f_1 + f_2 + \sqrt{f_1^2 + f_2^2}$$

Using R-functions, any set theoretic expression can be translated into a real-valued function by syntactically replacing Boolean operations by corresponding R-functions.

An R-function can be easily built for parametrical models, and the above Boolean operations always work even when topological changes are involved. For examplein Fig. 2, let x_1, x_2 be the respective centers of circles C_1, C_2. The left and right domains in Fig. 2 can be represented by the same R-function (with parameters x_1, x_2):

$$\omega(x_1, x_2) = f_1 \wedge f_2 \wedge f_3 \wedge f_4 \wedge f_5 \wedge f_6,$$

where

$$f_1 = x + 5; \quad f_2 = 5 - x;$$
$$f_3 = y + 3; \quad f_4 = 3 - y;$$
$$f_5(x_1) = x^2 - 2x_1 x + x_1^2 + y^2 - 4;$$
$$f_6(x_2) = x^2 - 2x_2 x + x_2^2 + y^2 - 4.$$

4.2. Representing the domain with R-functions

We can now replace the parametric domain $\Omega^{\mathbf{p}}$ with a fixed domain Ω^U by use of the characteristic function defined in (15) in combination with R-functions.

For the changing domain $\Omega^{\mathbf{p}}$, $\mathbf{p} \in \mathbf{I}$, we find a regular, fixed domain Ω^U such that $\Omega^{\mathbf{p}} \subset \Omega^U$, $\forall \mathbf{p} \in I_{\mathbf{p}}$. For example, we can choose an axis-aligned bounding box of $\bigcup\limits_{\mathbf{p} \in \mathbf{p}} \Omega^{\mathbf{p}}$ as Ω^U. Using R-functions, we can then construct a function $\Phi : \Omega^U \times I_{\mathbf{p}} \to \mathbb{R}$ such that

$$\begin{cases} \Phi(\mathbf{x}, \mathbf{p}) > 0, & \mathbf{x} \text{ inside } \Omega^{\mathbf{p}}, \\ \Phi(\mathbf{x}, \mathbf{p}) = 0, & \mathbf{x} \text{ on the boundary of } \Omega^{\mathbf{p}}, \\ \Phi(\mathbf{x}, \mathbf{p}) < 0, & \mathbf{x} \text{ outside } \Omega^{\mathbf{p}}. \end{cases}$$

Such Φ is constructed as the product of the implicit functions determined by the outer boundary of model $\Omega^{\mathbf{p}}$.

We also define the following characteristic function [18]:

$$H(\Phi(\mathbf{x}, \mathbf{p})) = \begin{cases} 1 \ , & \text{if } \Phi(\mathbf{x}, \mathbf{p}) \geq 0, \\ 0 \ , & \text{if } \Phi(\mathbf{x}, \mathbf{p}) < 0, \end{cases} \tag{15}$$

which indicates whether a given point belongs to $\Omega^{\mathbf{p}}$ or not.

Then the weak form of (6) can be formulated as follows:

$$\begin{aligned} &\int_{I_1} \cdots \int_{I_m} \int_{\Omega^U} (\nabla \mathbf{u})^T \mathbf{C} \nabla \delta \mathbf{u} H(\Phi) \; d\Omega dp_1 \cdots dp_m = \\ &\int_{I_1} \cdots \int_{I_m} \int_{\Omega^U} [f^T \delta \mathbf{u} + \operatorname{div}(\tau^T \delta \mathbf{u n})] H(\Phi) \\ &\qquad\qquad d\Omega dp_1 \cdots dp_m. \end{aligned} \tag{16}$$

Note that we have an integral of $\operatorname{div}(\tau^T \delta \mathbf{un})$ over the entire domain Ω^U, which is converted from the boundary integral on $\Gamma_N(\mathbf{p})$ in (9) by the divergence theorem; the same strategy was also applied in [18]. Thus the boundary traction τ, originally defined on Γ_N, must be extended from the boundary to the entire domain Ω^U. This can be accomplished, for example, using transfinite interpolation with approximate distance as described in [31]. Note also that the weak formulation in (16) is different from the one in (9) in that the former has a characteristic function $H(\Phi)$ in its integral terms.

4.3. Integration using the characteristic function

Numerically computing (16) involves evaluating a high-dimensional integral with a characteristic function. This is achieved using the Gaussian integration method in high dimensions, as explained below.

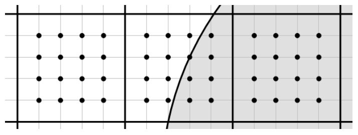

Figure 3: Gaussian integration methods with the characteristic function

The integral in (16) needs to be calculated for each FE element in turn. Let Ω^e be a 3D FE element under consideration: see Fig. 3. We wish to evaluate the integral for a general function g,

$$\begin{aligned} &\int_{I_1} \cdots \int_{I_m} \int_{\Omega^e} g(x, y, z, p_1, \ldots, p_m) \\ &\qquad H(\Phi(x, y, z, p_1, \ldots, p_m)) \; d\Omega dp_1 \cdots dp_m. \end{aligned}$$

The integral is computed via a Gaussian integration approach via picking up a set of sampling points within the integration domain, and computing the integral as weighted sum of the function values at these points. Specifically, let k be the number of sampling points in $m+3$ dimensions. We pick k points $(x^i, y^i, z^i, p_1^i, \ldots, p_m^i)$,

$1 \le i \le k$, for spatial coordinates (x, y, z). We then calculate values of

$$g_i = g(x^i, y^i, z^i, p_1^i, \dots, p_m^i),\ \Phi_i = \Phi(x^i, y^i, z^i, p_1^i, \dots, p_m^i),$$

and evaluate

$$H_i = \begin{cases} 1\ , & \text{if } \Phi_i \ge 0, \\ 0\ , & \text{if } \Phi_i < 0, \end{cases}$$

for $i = 1, \dots, k$. We then have the approximation

$$\int_{I_1} \cdots \int_{I_m} \int_{\Omega^e} gH\ d\Omega dp_1 \cdots dp_m \approx v \sum_{i=1}^{k} g_i H_i w_i, \quad (17)$$

where w_i is the weight of the i-th Gaussian point such that $\sum_{1 \le i \le k} w_i = 1$, and $v = V_{\Omega^e} \prod_{j=1}^{m} l_{I_j}$ is the 'volume' of the integration domain (V_{Ω^e} is the volume of Ω^e and l_{I_j} is the length of the interval I_j). The integral

$$\int_{I_1} \cdots \int_{I_j^e} \cdots \int_{I_m} \int_{\Omega^U} g(x, y, z, p_1, \dots, p_m) \cdot H(\Phi(x, y, z, p_1, \dots, p_m))\ d\Omega dp_1 \cdots dp_m.$$

is computed in a similar way, where I_j^e is a 1D FE element under consideration. Due to the variable separation, the numerical integral can be evaluated without difficulty.

5. Numerical results

The proposed approach has been implemented in Matlab on a computer with an Intel Core i5-3470 3.20GHz CPU and 8GB RAM. Various numerical examples are shown in this section to validate the idea, and to test accuracy and efficiency of the proposed method. We include cases involving changes in position and size of positive and negative features, and feature intersections. All our experiments concerned 3D linear elasticity with a Young's modulus $E = 2 \times 10^{11}$ Pa and Poisson's ratio $\nu = 0.33$. The body force per unit volume for all models was $f = -1 \times 10^6$ N/m^3 in the z-direction.

The results computed by our method were compared with those obtained by direct FE using *relative error*:

$$re = \frac{||\mathbf{u} - \mathbf{u}_{FEA}||}{||\mathbf{u}_{FEA}||}, \quad (18)$$

where $\mathbf{u}$ and $\mathbf{u}_{FEA}$ are the computed simulation solutions using the proposed approach and the FE method respectively, and $||\cdot||$ represents the Euclidean length of a vector.

For each test example, we also list the number of mesh elements, the PGD offline computation time, the online solution evaluation time, the direct FE computation time, and the speedup. The results are summarized in Table 1. Details for each example are explained below.

Table 1: Execution time for examples

Example	Mesh	Offline	Online	FEA	Speedup
#1	750 (3D) ×10 (1D)	32 min (10 terms)	1.8 ms	0.20s	110×
#2	1000 (3D) ×4 (1D) ×4 (1D)	135 min (7 terms)	1.6 ms	0.43s	270×
#3	3000 (3D) ×34 (1D)	955 min (16 terms)	10.7 ms	7.94s	740×
#4	6000 (3D) ×10 (1D)	355 min (13 terms)	11.6 ms	16.84s	1456×
#5	1840 (3D) ×27 (1D)	210 min (18 terms)	6.7 ms	1.92s	287×

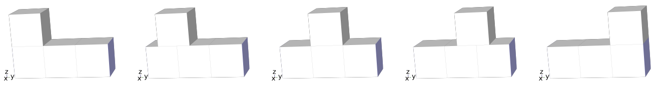

Figure 4: Example #1: Model with a moving positive feature

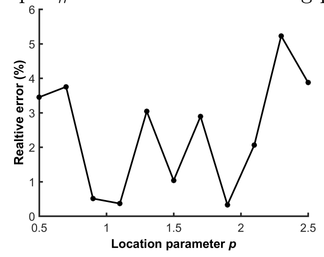

Figure 5: Average relative error for Example #1 when the design parameter changes from 0.5 to 2.5

5.1. Example #1: a moving positive feature

We first tested the approach's ability to handling a moving positive feature. The example in Fig. 4 was built from a combination of four $1 \times 1 \times 1$ unit cubes. The top one is centered at point $(0.5, p, 1.5)$, with a translation parameter $p \in [0.5, 2.5]$. The left and right side faces of the model are fixed along its sides as boundary conditions. In order to perform a PGD simulation of the model, we created a uniform mesh of $5 \times 15 \times 10$ hexahedral elements for the domain $\Omega^U = [0, 1] \times [0, 3] \times [0, 2]$, and a 10-cell mesh for the 1-D parameter interval $[0.5, 2.5]$. It took about 32 minutes to compute offline a PGD representation of ten summation terms, 0.0018 s to evaluate the PGD solution for a specific parameter value p, In contrast it took 0.20 s to perform FE analysis for a specific parameter value p. A speedup of 110 times is achieved, demonstrating the approach's efficiency in direct simulation.

The computed results are compared with the benchmark FE results in Fig. 5 for 10 uniformly sampled values of p. The relative errors are all below 6% and are considered both acceptable and reliable, considering the fact that various numerical errors may be introduced both by the PGD computations and the FE computations. Numerical errors in our offline computation and the FE analysis lead to the asymmetry seen the results from the FE analysis are not completely symmetric even though the model is symmetric. We also plot and compare point-wise the PGD and FEA solutions in Fig. 6 when $p = 1.5$, showing the x-, y-,

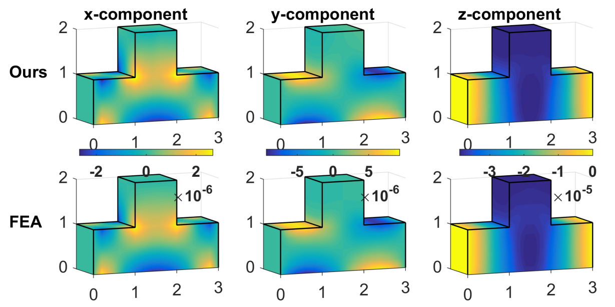

Figure 6: Computed displacement fields for proposed approach and the FE method for Example #1, for $p = 1.5$.

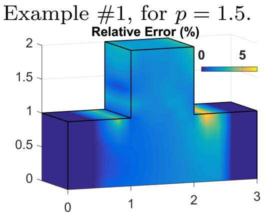

Figure 7: Relative error of Example #1 when $p = 1.5$.

and z- components of the displacement field. Their relative point-wise error is also shown in Fig. 7). The two results have very close point-wise approximation. These results demonstrate the approach's simulation accuracy.

5.2. Example #2: a variable-sized negative feature

We next tested the performance of the proposed approach for a variable-sized negative feature using the example in Fig. 8. The example model is a $[0, 1]^3$ cube with a $1\times p_1\times p_2$ through-hole feature centered at $(0.5, 0.5, 0.5)$. There are two size parameters p_1, $p_2 \in [0.2, 0.4]$. The left and right side faces of the model are fixed as boundary conditions. To compute the PGD-based solution, we created a uniform $10 \times 10 \times 10$ mesh of the domain $\Omega^U = [0, 1]^3$, and a uniform 4-cell mesh for each size parameter p_1, p_2. It took about 135 minutes to compute offline a PGD solution of 7 summation terms, and 0.0016 seconds to evaluate the PGD solution at a specific parameter. It took 0.43 seconds to perform FE analysis for a specific configuration. A speedup of 270 times is achieved, again demonstrating the approach's efficiency.

The computed results are compared with the benchmark FE results in Fig. 9 for 25 parameter combinations. The relative errors are all below 5% and are acceptable. We also plot and compare the point-wise PGD and FEA solutions in Fig. 10 when $(p_1, p_2) = (0.2, 0.4)$, again showing the x-, y-, and z- components of the displacement field. Their relative point-wise error is also shown in Fig. 11). The two results demonstrate very close point-wise approximation. All these results demonstrate the approach's simulation accuracy.

5.3. Example #3: Intersecting features

We also tested performance of the proposed approach for a case involving intersecting features using the example in

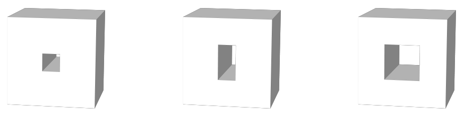

Figure 8: Example #2: Model with a variable-sized negative feature

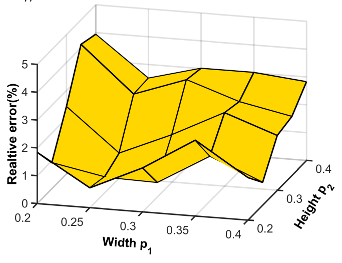

Figure 9: Average relative error for Example #2 for parameters values in $[0.2, 0.4]^2$

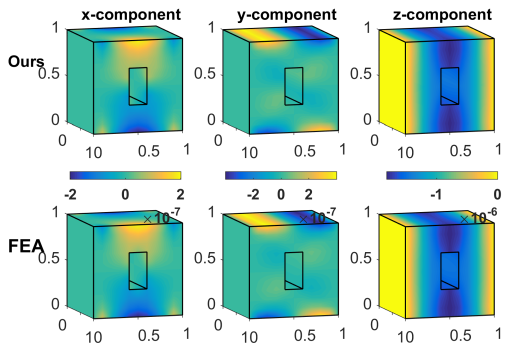

Figure 10: Comparison of the computed displacement fields for the proposed approach and the FE method for Example #2, for $p_1 = 0.2$, $p_2 = 0.4$

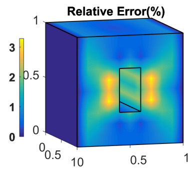

Figure 11: Relative error for Example #2 when $p_1 = 0.2$, $p_2 = 0.4$

Fig. 12. The model is built as the union of three $1 \times 1 \times 1$ cubes with a fixed $1\times0.2\times0.2$ through hole H_1, centered at $(0.5, 1.5, 0.5)$, and a moving $1 \times 0.4 \times 0.4$ through hole H_2, centered at $(0.5, p, 1.5)$, with parameter $p \in [-0.2, 3.2]$. The hole H_2 moved along the y-axis. There were various intersection configurations with the hole H_1 and the solid model Ω^U, for example partial intersection with hole H_1 or model Ω^U, H_1 completely within hole H_2 etc, resulting in models of varying topologies, as can be seen in Fig. 12. The model was subject to an extra boundary load $\tau = -10^6$ N/m^2 on the upper face, as well as the body force. The edges parallel to the x-axis on the bottom face were fixed to give the Dirichlet boundary conditions.

The PGD simulation was performed on a mesh of $10 \times 30 \times 10$ hexahedral elements for domain $\Omega^U = [0, 1] \times [0, 3] \times [0, 1]$, and a 34-cell mesh for each 1D parameter interval in the range $[-0.2, 3.2]$. It took about 16 hours to compute offline a PGD representation of 16 summa-

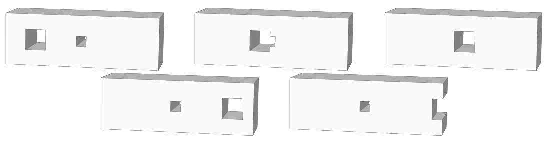
Figure 12: Example #3: Model with feature intersections.

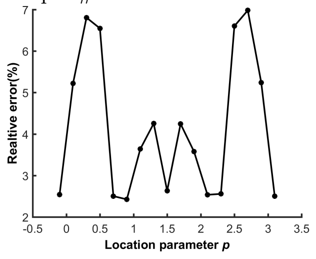

Figure 13: Average relative error for Example #3 as the parameter changes in $[-0.1, 3.1]$.

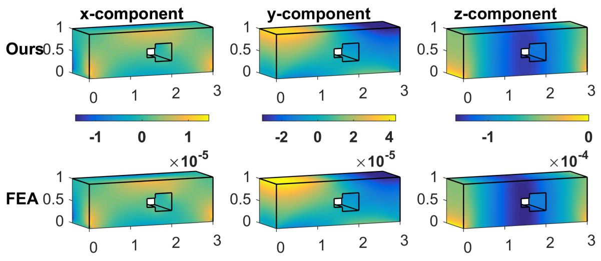

Figure 14: Displacement field for Example #3 when $p = 1.8$.

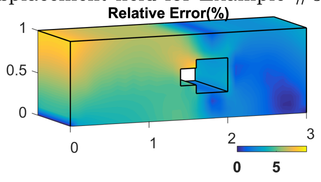

Figure 15: Relative error for Example #3 when $p = 1.8$.

tion terms, and 0.01 s to evaluate the PGD solution at a specific parameter. It took 7.94 s to perform FE analysis for a specific configuration. A speedup of 740 times was achieved, demonstrating the approach's efficiency in direct simulation.

The relative errors between the PGD and FEA results are plotted in Fig. 13 for 17 uniformly sampled parameter values. As can be seen from the results, the relative errors are all below 7% and are acceptable. The point-wise PGD and FEA solutions are also plotted and compared in Fig. 14 when $p = 1.8$, where the x-, y-, and z- components of the displacement field are shown. The relative point-wise error is also shown in Fig. 15; it is small. All these results demonstrate the approach's simulation accuracy in cases of intersecting features.

5.4. *Example #4: Model with variable-sized negative feature intersecting another feature*

We test a case of a variable-sized negative feature intersecting another negative feature, as shown in Fig. 16. In this example, the model is built from a union of four

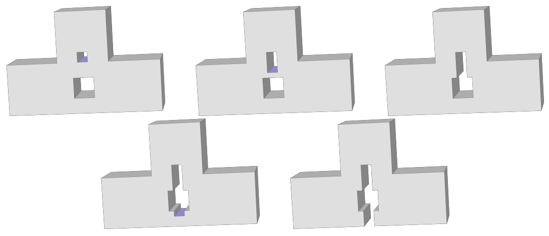
Figure 16: Example #4: Model with deforming negative feature intersecting another feature

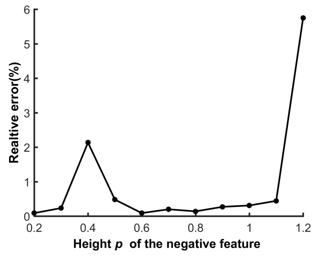

Figure 17: Average relative error for Example #4 as the parameter changes in $[0.2, 1.2]$

$1 \times 1 \times 1$ cubes with a fixed $1 \times 0.4 \times 0.4$ through-hole feature, and a variable-sized $1 \times 0.2 \times p$ through-hole feature, where $p \in [0.2, 1.2]$ controls size. The left and right sides of the model are fixed. As can be seen from Figure 16, as the through hole's length increases, the topology changes.

In order to compute the PGD-based solution, we created a mesh of $10 \times 30 \times 20$ hexahedral elements for the domain $\Omega^U = [0, 1] \times [0, 3] \times [0, 2]$, and a 10-cell mesh for the 1-D parameter interval $[0.2, 1.2]$. It took about 355 minutes to compute offline a PGD representation with 13 summation terms, and 0.01 s to evaluate the PGD solution at a specific parameter. It took 16.84 s to perform FE analysis for a specific configuration. A speedup of 1456 times was achieved, demonstrating the approach's efficiency in direct simulation.

Relative errors between the PGD and FEA results are plotted in Fig. 17 for 11 uniformly sampled parameter values. As can be seen from the results, the relative errors are all below 10% and are acceptable. The point-wise PGD and FEA solutions are also plotted and compared in Fig. 18 for $p = 0.5$, where the x-, y-, and z- components of the displacement field are shown. The relative point-wise error is also shown in Fig. 15, and is small. All these results demonstrate the approach's simulation accuracy even in cases involving intersecting features.

5.5. *Example #5: Curved restrictions*

We finally tested performance of the proposed approach for a case involving curved restrictions using the example in Fig. 20. The model is built as the intersection of a $1 \times 2.3 \times 0.8$ cuboid and the union of 2 radius 1 and height 0.8 cylinders, Z-axis as the axial direction, centered at $(0.5, \sqrt{3}/2 + 0.15, 0.4)$ and $(0.5, -\sqrt{3}/2 + 2.15, 0.4)$, with a moving radius 0.2 and height 1 cylinder through hole H_1,

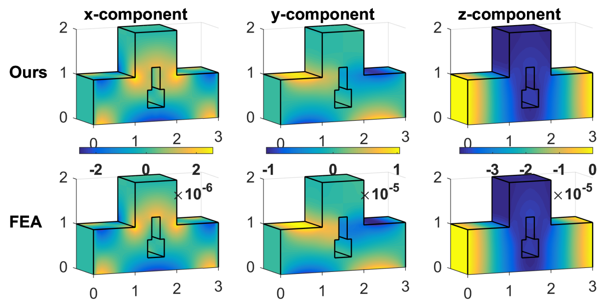

Figure 18: Displacement field for Example #4 when $p = 0.5$

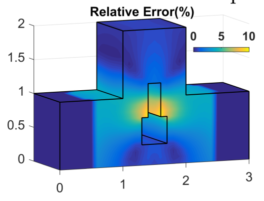

Figure 19: Relative error for Example #4 when $p = 0.5$

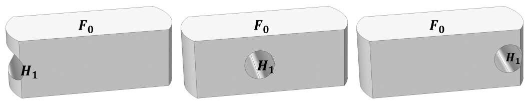

Figure 20: Example #5: Model with curved restrictions.

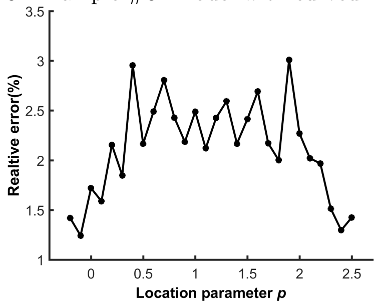

Figure 21: Average relative error for Example #5 as the parameter changes in $[-0.2, 2.5]$.

X-axis as the axial direction, centered at $(0.5, p, 0.4)$, with parameter $p \in [-0.2, 2.5]$. The hole H_1 moved along the y-axis. The upper face F_0 is fixed giving the Dirichlet boundary condition.

The PGD solution was performed on a mesh of $10\times23\times8$ hexahedral elements for domain $\Omega^U = [0, 1] \times [0, 2.3] \times [0, 0.8]$, and a 27-cell mesh for each 1D parameter interval in the range $[-0.2, 2.5]$. It took about 210 minutes to compute offline a PGD representation of 18 summation terms, and 0.0067 s to evaluate the PGD solution at a specific parameter. It took 1.92 s to perform FE analysis for a specific configuration. A speedup of 287 times was achieved, demonstrating the approach's efficiency in direct simulation.

The relative errors between the PGD and FEA results are plotted in Fig. 21 for 28 uniformly sampled parameter values. As can be seen from the results, the relative errors are all below 3.5% and are acceptable.

5.6. Discussion

As can be observed from the above examples, the proposed approach of direct simulation can very rapidly give simulation results after computing a single offline parametric solution. It can also handle cases involving topological changes caused by parameter modifications. The computed results for a given parameter are within 10% of the FE analysis results, and are acceptable considering the numerical errors induced by the PGD and FE computations. The simulation speed is increased by at least 2 orders compared to those obtained by FE analysis. The real speedup is expected to be much higher for complex models as FE analysis takes time $O(n^2)$ with respect to the number of mesh elements, while solution evaluation from the offline parametric PGD solution only takes time $O(n)$, and furthermore the time-consuming meshing or remeshing process is not included in the FE computation time.

We further discuss below the limitations of the proposed approach.

5.6.1. Complex examples and R-function

The geometric complexity that the proposed approach can handle depends on the successful construction of the associated R-function for the features undergoing parametric deformations. For a complex CAD model in a general NURBS representation, the proposed approach can be applied directly without any difficulty as long as the shapes undergoing deformation can be represented using a parametric R-function. The property can be observed from (16), where the modification parameters are used in the integration function $H(\Phi(\mathbf{x}, \mathbf{p}))$ while the integration domain Ω_U remains unchanged.

On the other hand, classical level-set approaches have been widely applied to approximate freeform complex surfaces without parametric deformation [32]. However, constructing a parametric level-set function to approximate a set of shapes undergoing parametric modifications is rarely studied in previous researches. The issue may be resolved via constructing a level-set function in a higher dimension including the shape modification parameters, and is to be addressed in our future work.

5.6.2. Accuracy control

The physical simulation accuracy of the proposed approach comes from several aspects: the PGD approximation error and the usage of R-function. The PGD approach has demonstrated its approximation under any accuracy control for simulation problems in a wide range of industrial applications [27, 33], and are not further discussed here. The error induced by the R-function is mainly due to the inaccurate numerical computation of the Gaussion integration in (17) due to insufficient point samplings. For example, it is noted from the numerical results that, for parameters where the topology of the edited model changes during the parameter variation process, the relative approximation error is usually higher than for other

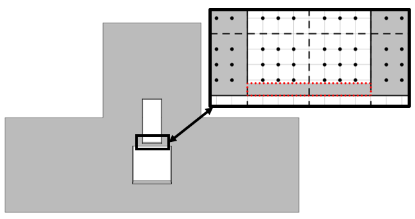

Figure 22: Local mesh and the Gaussian points for one instance of the topology changing points of Example #4. The area in the red box is not properly represented by the Gaussian integration points.

parameter values, although still acceptable: see for example the curves of relative errors in Fig. 13 and 17. This can perhaps be explained from the fact the uniform coarse samplings of the Gaussian points in the integration process in (17) is not dense enough to account for such complex geometric or topological changes in the case. See also Fig. 22 for an explanation. A more accurate numerical approach needs to be developed to further improve the simulation accuracy. Low discrepancy methods could provide one potential solution to this problem [34, 35].

We have to mention here that the usage of the R-function here also introduces great advantages for fast predicting physical properties of models undergoing parametric modifications in that no additional efforts of remeshing is required for the modified model and the mesh quality in ensured, which in turn improves the numerical approximation accuracy. Actually, although various efficient approaches for shapes undergoing deformations have been proposed, they usually suffer the issues of bad mesh quality (see for example the produced meshes in [36]), and correspondingly low FE computational accuracy. The proposed approach combining PGD and R-function performs the integration on a fixed mesh, and does not have this issue and can be applied to shape undergoing large deformations.

5.6.3. Offline computation cost

It is also noticed here that the offline computation in Table 1 takes much more time than performing one single FE analysis. This is predictable as in the whole PGD solution computation process in Algorithm 1, solutions $\mathbf{d}(\mathbf{x})$ needs to be iteratively computed until convergence, one single step of which is similar to computing an FE solution. The process can be further accelerated via additional separations of the spatial coordinates [21, 37, 38], i.e., approximating $\mathbf{d}(\mathbf{x})$ as $\mathbf{d}^1(x) * \mathbf{d}^2(y) * \mathbf{d}^3(z)$, where $\mathbf{d}^1(x), \mathbf{d}^2(y), \mathbf{d}^3(z)$ are piecewise function built from one dimension FE basis, and '$*$' represents the *element-wise multiplication* (if $A = [a_1, a_2, a_3]$ and $B = [b_1, b_2, b_3]$, then $A * B = [a_1b_1, a_2b_2, a_3b_3]$). For example, suppose that $\mathbf{d}$ has $10 \times 10 \times 10(Nodes) \times 3(DOFs/Node) = 3000$ DOFs, then each of $\mathbf{d}^1(x), \mathbf{d}^2(y), \mathbf{d}^3(z)$ has only $10(Nodes) \times 3(DOFs/Node) = 30$ DOFs, and can be much more efficiently computed in the 1D FE computations.

On the other hand, it is also noticed that the main computational costs during the PGD computations are due to the involved high-dimensional integrals. The low discrepancy sequences which has been used successfully to evaluate integrals in 365 dimensions in finance, is a nice candidate to further improve the computational efficiency, besides via additional separations of the spatial coordinates.

6. Conclusions

This paper has proposed a novel approach for direct simulation of parametric CAD models in real time. This is achieved via computing offline a generic solution that includes all solutions for every possible choice of design parameters, in a high-dimension space that includes the space coordinates and design variables. It is based on the PGD model reduction technique in combination with R-functions. The approach works even for cases involving topological changes, and does not need remeshing or mesh mapping. The computational accuracy and efficiency of the proposed approach were demonstrated via various numerical examples.

The proposed approach works well. In future, we will further examine its performance for realistic CAD models with fine meshes, which will further improve the numerical accuracy, but at much greater computational expense; a strategy of parallel computation may be required. The proposed approach also should work in principle for deforming freeform models [39, 40]. However, such deformations usually involve a large number of design variables, and a strategy to reduce this number has to be developed for practical usage.

PGD is a novel model reduction approach that has intrinsic advantages in resolving the curse of dimensionality and can be of potential use in various industrial applications. The PGD approach has already demonstrated its theoretical soundness in tackling linear or nonlinear physical phenomena, for example nonlinear elasticity [33]. More research efforts are needed to fully explore its potential uses in solid modeling.

Acknowledgements

The work described in this paper is partially supported by the NSF of China (No. 61472356, 61210007).

References

References

[1] V. Shapiro, I. Tsukanov, A. Grishin, Geometric issues in computer aided design/computer aided engineering integration, Journal of Computing and Information Science in Engineering 11 (2) (2011) 021005–1:13.

[2] K. Shimada, Current issues and trends in meshing and geometric processing for computational engineering analyses, Journal of Computing and Information Science in Engineering, ASME 11 (2011) 1–13.

[3] I. Turevsky, S. Gopalakrishnan, K. Suresh, Defeaturing: A posteriori error analysis via feature sensitivity, International Journal for Numerical Methods in Engineering 76 (9) (2008) 1379–1401.

[4] M. Li, S. Gao, R. Martin, Estimating effects of removing negative features on engineering analysis, Computer-Aided Design, Special Issue of ACM Solid and Physical Modeling 2011 43 (1) (2011) 1402–1412.

[5] R. Białecki, A. Kassab, A. Fic, Proper orthogonal decomposition and modal analysis for acceleration of transient fem thermal analysis, International journal for numerical methods in engineering 62 (6) (2005) 774–797.

[6] A. Ammar, D. Ryckelynck, F. Chinesta, R. Keunings, On the reduction of kinetic theory models related to finitely extensible dumbbells, Journal of Non-Newtonian Fluid Mechanics 134 (1) (2006) 136–147.

[7] B. Ganapathysubramanian, N. Zabaras, Modeling diffusion in random heterogeneous media: Data-driven models, stochastic collocation and the variational multiscale method, Journal of Computational Physics 226 (1) (2007) 326–353.

[8] D. Kim, W. Koh, R. Narain, K. Fatahalian, A. Treuille, J. F. O'Brien, Near-exhaustive precomputation of secondary cloth effects, ACM Transactions on Graphics 32 (4) (2013) 1.

[9] F. Chinesta, P. Ladeveze, E. Cueto, A short review on model order reduction based on proper generalized decomposition, Archives of Computational Methods in Engineering 18 (4) (2011) 395–404.

[10] F. Chinesta, a. Leygue, F. Bordeu, J. V. Aguado, E. Cueto, D. Gonzalez, I. Alfaro, a. Ammar, a. Huerta, PGD-Based Computational Vademecum for Efficient Design, Optimization and Control, Archives of Computational Methods in Engineering 20 (1) (2013) 31–59.

[11] A. Ammar, B. Mokdad, F. Chinesta, R. Keunings, A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids, Journal of Non-Newtonian Fluid Mechanics 139 (3) (2006) 153–176.

[12] A. Ammar, B. Mokdad, F. Chinesta, R. Keunings, A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids: Part ii: Transient simulation using space-time separated representations, Journal of Non-Newtonian Fluid Mechanics 144 (2) (2007) 98–121.

[13] F. Chinesta, A. Ammar, A. Leygue, R. Keunings, An overview of the proper generalized decomposition with applications in computational rheology, Journal of Non-Newtonian Fluid Mechanics 166 (11) (2011) 578–592.

[14] F. Chinesta, A. Ammar, E. Cueto, On the use of proper generalized decompositions for solving the multidimensional chemical master equation, European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique 19 (1-3) (2010) 53–64.

[15] S. Zlotnik, P. Díez, D. Modesto, A. Huerta, Proper generalized decomposition of a geometrically parametrized heat problem with geophysical applications, International Journal for Numerical Methods in Engineering.

[16] V. L. Rvachev, Analytical description of some geometric objects, Dokl AS USSR 153 (4) (1963) 765–768.

[17] V. Shapiro, Theory of r-functions and applications: A primer, Cornell University.

[18] J. Chen, V. Shapiro, K. Suresh, I. Tsukanov, Shape optimization with topological changes and parametric control, International journal for numerical methods in engineering 71 (3) (2007) 313–346.

[19] J. Chen, M. Freytag, V. Shapiro, Shape sensitivity of constructively represented geometric models, Computer Aided Geometric Design 25 (7) (2008) 470–488.

[20] A. Ammar, A. Huerta, F. Chinesta, E. Cueto, A. Leygue, Parametric solutions involving geometry: a step towards efficient shape optimization, Computer Methods in Applied Mechanics and Engineering 268 (2014) 178–193.

[21] A. Ammar, F. Chinesta, P. Diez, A. Huerta, An error estimator for separated representations of highly multidimensional models, Computer Methods in Applied Mechanics and Engineering 199 (25) (2010) 1872–1880.

[22] L. E. Figueroa, E. Süli, Greedy approximation of highdimensional ornstein–uhlenbeck operators, Foundations of Computational Mathematics 12 (5) (2012) 573–623.

[23] F. Chinesta, A. Ammar, E. Cueto, Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models, Archives of Computational methods in Engineering 17 (4) (2010) 327–350.

[24] F. Chinesta, A. Leygue, F. Bordeu, J. Aguado, E. Cueto, D. González, I. Alfaro, A. Ammar, A. Huerta, Pgd-based computational vademecum for efficient design, optimization and control, Archives of Computational Methods in Engineering 20 (1) (2013) 31–59.

[25] G. Nikishkov, Introduction to the finite element method, University of Aizu.

[26] M. S. Gockenbach, Understanding and implementing the finite element method, Siam, 2006.

[27] A. Falco, A. Nouy, A proper generalized decomposition for the solution of elliptic problems in abstract form by using a functional eckart–young approach, Journal of Mathematical Analysis and Applications 376 (2) (2011) 469–480.

[28] C. Le Bris, T. Lelievre, Y. Maday, Results and questions on a nonlinear approximation approach for solving high-dimensional partial differential equations, Constructive Approximation 30 (3) (2009) 621–651.

[29] V. L. Rvachev, T. I. Sheiko, R-functions in boundary value problems in mechanics, Applied Mechanics Reviews 48 (4) (1995) 151–188.

[30] V. Shapiro, I. Tsukanov, Implicit functions with guaranteed differential properties, in: Proceedings of the fifth ACM symposium on Solid modeling and applications, ACM, 1999, pp. 258–269.

[31] V. L. Rvachev, T. I. Sheiko, V. Shapiro, I. Tsukanov, Transfinite interpolation over implicitly defined sets, Computer aided geometric design 18 (3) (2001) 195–220.

[32] C. Bajaj, G. Xu, Q. Zhang, Smooth surface constructions via a higher-order level-set method. ices report 06-18, Institute for Computational and Engineering Sciences, The University of Texas at Austin.

[33] P. Ladeveze, J.-C. Passieux, D. Néron, The latin multiscale computational method and the proper generalized decomposition, Computer Methods in Applied Mechanics and Engineering 199 (21) (2010) 1287–1296.

[34] T. Davies, R. Martin, Low-discrepancy sequences for volume properties in solid modelling, in: Proceedings of CSG, Vol. 98, Citeseer, 1998, p. 13.

[35] X. Li, W. Wang, R. R. Martin, A. Bowyer, Using low-discrepancy sequences and the crofton formula to compute surface areas of geometric models, Computer-Aided Design 35 (9) (2003) 771–782.

[36] A. De Boer, M. Van der Schoot, H. Bijl, Mesh deformation based on radial basis function interpolation, Computers & structures 85 (11) (2007) 784–795.

[37] S. M. Nazeer, F. Bordeu, A. Leygue, F. Chinesta, Arlequin based pgd domain decomposition, Computational Mechanics 54 (5) (2014) 1175–1190.

[38] A. Ammar, F. Chinesta, E. Cueto, Coupling finite elements and proper generalized decompositions, International Journal for Multiscale Computational Engineering 9 (1).

[39] T. Ju, S. Schaefer, J. Warren, Mean value coordinates for closed triangular meshes, ACM Transactions on Graphics (TOG) 24 (2005) 561.

[40] S. Jinsp, Y. Zhang, C. L. Wang, Deformation with enforced metrics on length, area and volume, Computer Graphics Forum 33 (2) (2014) 429–438.