Implicit Partitioning Methods for Unknown Parameter Domains

Bernhard Wieland

Preprint Series: 2013 - 09

Fakultät für Mathematik und Wirtschaftswissenschaften
UNIVERSITÄT ULM

Noname manuscript No.
(will be inserted by the editor)

Implicit Partitioning Methods for Unknown Parameter Domains

in the Context of the Reduced Basis Method

Bernhard Wieland

Received: date / Accepted: date

Abstract The key condition for the application of the Reduced Basis Method (RBM) to Parametrized Partial Differential Equations (PPDEs) is the availability of affine decompositions of the systems in parameter and space. The efficiency of the RBM depends on both the number of reduced basis functions and the number of affine terms. A possible way to reduce the costs is a partitioning of the parameter domain. One creates separate RB spaces [6] and affine decompositions [4] on each subdomain. Since the solutions are supposed to be smooth in parameter, the variation of the solutions on a subdomain becomes small and only few basis functions and affine terms are needed.

Based upon the Empirical Interpolation Method (EIM), we generalize the existing partitioning concepts to arbitrary input functions with possibly unknown, high-dimensional, or even without direct parameter dependencies. No a-priori information about the input is necessary. We create affine decomposition and partitions without the knowledge of either an explicit description of the parameter domain or of the form of the partitions. An application includes PPDEs with stochastic influences [7, 12]. For a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, the parameter domain is now associated with Ω. Each element $\omega \in \Omega$ represents a stochastic event. Hence, ω is not a parameter in classical sense and there usually is no explicit description of Ω.

Keywords model reduction · reduced basis method · empirical interpolation method · parametrized partial differential equations · adaptive partitioning

Mathematics Subject Classification (2000) 05C70 · 78M34 · 35R60

This paper was partly written while B.W. was funded by the state of Baden-Württemberg according to the State Postgraduate Scholarships Act (Landesgraduiertenförderungsgesetz).

B. Wieland
Institute of Numerical Mathematics, Ulm University, Helmholtzstraße 20, 89069 Ulm, Germany
Tel.: +49-731-5015391
Fax: +49-731-5023548
E-mail: bernhard.wieland@uni-ulm.de

1 Introduction

In the context of Reduced Basis Methods (RBMs) for Partial Differential Equations (PDEs) with deterministic parameter dependencies [5,8,9,11,13], it is common to split the parameter domain into several parts and construct separate reduced bases for each parameter subdomain [2,3,4,6]. It is assumed that the variation of the parametric influences of the PDEs and therefore the variation of the corresponding solutions become small on each subdomain. Then, only small numbers of basis functions are needed and the online cost of the RBM decreases.

In [3] and [2], the so-called *hp*-Partitioning has been introduced for stationary and instationary problems, respectively, for already affine problems. The methods differ only slightly from the so-called *p*-Partitioning [6]. The main distinction are two different procedures for the splitting into subdomains, leading to theoretical convergence results for some special cases. In [4], the *hp*-Partitioning is introduced for non-affine problems. It is based upon the EIM and generates affine decompositions and partitions simultaneously.

In this work, we generalize the partitioning concepts developed for deterministic and compact parameter domains to arbitrary, possibly unknown parameter domains. No explicit description of the parameter domain — if existent at all — will be required, and no particular information about the problem is needed. Furthermore, we will show that our new implicit partitioning methods also outperform the existing methods for wide classes of problems even in the setting of known parameter domains.

In Section 2, we briefly introduce the basic idea of partitioning and the *hp*-Partitioning according to [4] for known, explicitly given parameter domains. In Section 3, we introduce the general concept of unknown parameter domains and of affine decompositions with respect to unknown parameters. Furthermore, we introduce some necessary assumptions and requirements for our Implicit Partitioning Method (IPM). As the *hp*-Partitioning, it generates affine decompositions and partitions in parallel. We will develop two different concepts of the IPM. In Section 4, we introduce an IPM where the form of the subdomains is not fixed but depends on the used collateral basis size. The method is therefore called Moving Shapes (MS) IPM. Next, in Section 5, we develop IPMs where the forms of the subdomains are supposed to be stationary. This method is called Fixed Shapes (FS) IPM. Finally, in Section 6, we provide several numerical examples and compare the different methods.

2 Preliminaries

We start introducing the basic ingredients and tasks of parameter domain partitioning and review some partitioning concepts for known, deterministic, and compact parameter domains.

Let $D \subset \mathbb{R}^d$ denote a bounded spatial domain and let $\mathcal{P} \subset \mathbb{R}^p$ be an arbitrary parameter domain. Furthermore, let $c : D \times \mathcal{P} \to \mathbb{R}$ denote a parametrized coefficient of a given PDE. For each $\mu \in \mathcal{P}$, we assume to obtain a trajectory $c(\mu) \in X \subset L_\infty(D) \cap C^0(D)$ for some appropriate discrete Hilbert space X on D of dimension $\mathcal{N}$.

In the context of Reduced Basis Methods (RBM), it is essential that $c(x;\mu)$ is affine with respect to the parameter μ. This allows for an efficient decomposition of expensive offline computations and highly efficient online simulations. In general, however, this requirement is not fulfilled. Hence, affine approximation of the form

$$c(x;\mu) \approx c_M(x;\mu) = \sum\nolimits_{j=1}^{M} \theta_j(\mu) q_j(x), \quad x \in D,\ \mu \in \mathcal{P}, \tag{1}$$

are desired, with so-called *collateral basis* functions $q_j \in X$ and efficiently evaluable parametric coefficients $\theta_j(\mu)$, $j = 1, \ldots, M$. Such approximations can be obtained using the Empirical Interpolation Method (EIM) which is described below.

For the partitioning, we then define $M_{\max}$ as the largest allowed collateral basis size. At the same time, an approximation error tolerance ε_{tol} is desired. Hence, the objective is to divide $\mathcal{P}$ into multiple subdomains and generate individual collateral bases such that

(i) the dimension of all affine approximations is smaller than $M_{\max}$,
(ii) the maximal approximation error on each subdomain does not exceed ε_{tol},
(iii) each parameter $\mu \in \mathcal{P}$ can be assigned efficiently to the right subdomain,
(iv) whereas the number of subdomains should be as small as possible.

2.1 Empirical Interpolation Method (EIM)

We briefly review the EIM as introduced for example in [1] and [10]. The main idea of the EIM is to generate a collateral basis such that the evaluation of the coefficients θ_j in (1) requires only the values of c at a small set of interpolation points.

2.1.1 EIM Offline-phase

A general form of the EIM offline procedure is described in Algorithm 1. It generates the so-called collateral basis $Q_M = \{q_1, \ldots, q_M\}$ and the corresponding set of interpolation points $T_M = \{t_1, \ldots, t_M\}$, $M \leq M_{\max}$. We describe the main steps below. The ingredient of the algorithm is a training set $\Xi_{\text{tr}} \subset \mathcal{P}$ such that the space $\operatorname{span}\{c(\mu) \,|\, \mu \in \Xi_{\text{tr}}\}$ sufficiently covers the family of functions $\{c(\mu) \,|\, \mu \in \mathcal{P}\}$. Furthermore, we start with an empty set of basis functions $Q_0 = \{\}$ and an empty set of interpolation points $T_0 = \{\}$.

We start with the procedure that computes the affine approximation in line 3 of Algorithm 1. For an empty basis Q_0, the procedure $\mathsf{getApproximation}(Q_0, T_0, c)$ returns $c_0^{\text{EIM}} = 0$ for all functions $c \in X$. Otherwise, $\mathsf{getApproximation}(Q_M, T_M, c)$ computes the coefficients $\boldsymbol{\theta}_M(c) = (\theta_j(c))_{j=1}^{M}$ by solving the linear system

$$\sum\nolimits_{j=1}^{M} \theta_j(c) q_j(t_i) = c(t_i),\ i = 1, \ldots, M, \tag{2}$$

and returns the approximation $c_M^{\text{EIM}} = \sum_{j=1}^{M} \theta_j(c) q_j$.

The procedure $\mathsf{getNextBasisFunction}(Q_{M-1}, T_{M-1}, \Xi_{\text{tr}})$ in line 2 evaluates approximations $c_{M-1}^{\text{EIM}}(\mu)$ of all functions $c(\mu), \mu \in \Xi_{\text{tr}}$, and returns the trajectory that is so far worst approximated in the L_∞-sense. In line 4, the residual is evaluated. The next knot t_M is defined in line 5 in order to supremize the residual,

Algorithm 1 EIM-Offline($\Xi_{\mathrm{tr}}, M_{\max}$)

```
1  for M = 1 to M_max do
2      c = getNextBasisFunction(Q_{M-1}, T_{M-1}, Ξ_tr)
3      c^EIM_{M-1} = getApproximation(Q_{M-1}, T_{M-1}, c)
4      r_M = c − c^EIM_{M-1}
5      t_M = arg ess sup_{x∈D} |r_M(x)|,   T_M = {T_{M-1}, t_M}
6      q_M = r_M / r_M(t_M),   Q_M = {Q_{M-1}, q_M}
7  end for
```

i.e., as that point where c is so far worst approximated. The next collateral basis function q_M is added in line 6, defined as the L_∞-normalized residual.

By construction, the approximation is exact at the knots $t_i, i = 1, \ldots, M$. This implies that the linear system (2) is lower triangular. The computational complexity of the evaluation of the EIM coefficients $\boldsymbol{\theta}_M$ is thus $\mathcal{O}(M^2)$.

2.1.2 EIM Online-phase

In the online phase, for a new parameter μ, we choose an $M < M_{\max}$ sufficiently large for a good approximation quality. Additionally, we define M^+ with $M < M^+ \leq M_{\max}$ that is used for the error estimation.

We first call getCoefficients$(M^+, c(\mu))$ that evaluates the trajectory at the knots $(t_i)_{i=1}^{M^+}$ and returns the solution $\boldsymbol{\theta}_{M^+}(\mu)$ of the linear system (2). Due to its lower triangular form, the solution shows a hierarchical structure, i.e., $\boldsymbol{\theta}_{M^+} = (\boldsymbol{\theta}_M, \theta_{M+1}, \ldots, \theta_{M^+})$. We use $\boldsymbol{\theta}_M$ to evaluate the approximation $c_M^{\mathrm{EIM}}(\mu)$ and the remaining coefficients for the error estimator

$$\Delta_{M,M^+}^{\mathrm{EIM}}(\mu) = \sum\nolimits_{j=M+1}^{M^+} |\theta_j(\mu)|. \tag{3}$$

The error estimator for the L_2-error would be given by $\sum_{j=M+1}^{M^+} \|q_j\|_2 |\theta_j(\mu)|$. For more details on EIM error estimators and more accurate bounds, see [10].

In the RBM context, the evaluation of $c_M^{\mathrm{EIM}}(\mu)$ is not necessary and only the coefficients $\boldsymbol{\theta}_M(\mu)$ are returned. Then, the online complexity reads $\mathcal{O}((M^+)^2)$.

2.2 *hp*-Partitioning

Let us now briefly describe the *hp*-Partitioning as it has been introduced in [4] for compact parameter domains. The term "*hp*" is adopted from the finite element (FE) theory. In the context of parameter domain partitioning, the "*h*" represents the refinement of the partition and the "*p*" stands for the improvement of the basis on a subdomain. Accordingly, the *hp*-Partitioning is divided into two separate parts, the *h*-part with the refinement of the partition and the *p*-part with the basis construction. We introduce two separate error tolerances $\varepsilon_{\mathrm{tol}}^h$ and $\varepsilon_{\mathrm{tol}}^p$ and two maximal numbers of collateral EIM basis functions $M_{\max}^h$ and $M_{\max}^p$. The *h*-indexed quantities are only employed to make the subdividing scheme cheaper whereas the *p*-indexed quantities refer to the actual desired values.

Algorithm 2 describes the general *h*-part. Given an initial partition, we call the procedure for each initial subdomain. The refinement and basis construction

Algorithm 2 *hp*-Partitioning($\mathcal{P}^j$, $M^h_{\max}$, $\varepsilon^h_{\text{tol}}$, J)

```
1  create Ξ^j_tr from P^j
2  for M = 1 to M^h_max do
3      {S^j_EIM,M, μ^j_M} = addBasisFunction(S^j_EIM,M−1, Ξ^j_tr)
4      ε_M,max = getMaxError(S^j_EIM,M, Ξ^j_tr)
5      if ε_M,max < ε^h_tol then return S^j_EIM,M, P^j end if
6  end for
7  {P^{J+i} | i = 1, ..., J_add} = refinePartition(P^j, μ^j_1, ..., μ^j_{M_max})
8  J_new = J + J_add
9  for i = 1 to J_add do
10     hp-Partitioning(P^{J+i}, M^h_max, ε^h_tol, J_new)
11 end for
```

works recursively. A rather large error tolerance $\varepsilon^h_{\text{tol}}$ is used and only a small number $M^h_{\max}$ of maximal basis functions per subdomain is allowed. In that way, the construction of superfluous bases functions for subdomains that are discarded anyway is avoided. The total number of current subdomains is denoted by J. The algorithm generates the structs $\mathcal{S}^j_{\text{EIM},M}$, $j = 1, \ldots, J$, that contain the complete EIM data for the respective subdomain, denoted by $\mathcal{P}^j$. The term "struct" is adopted from programming languages like C where it denotes a structured data type that unites a set of components of different data types.

The procedure addBasisFunction$(\cdot)$ in line 3 performs one iteration of the offline EIM construction as described in Section 2.1.1. Additionally, it now returns the parameter that corresponds to the just selected basis function. These parameters are used for the new refinement procedures in line 7. Since no error estimators for the EIM can be evaluated during the construction of the collateral basis, the exact L_∞-error is evaluated in line 4 and used as termination condition in line 5.

Before we introduce two different refinement procedures that can be used in line 7, we briefly provide the second step of the *hp*-Partitioning, the *p*-part. For each final subdomain, we call separately start the EIM procedure and iterate until the small error tolerance $\varepsilon^p_{\text{tol}}$ or the maximal number $M^p_{\max}$ is reached.

2.2.1 Gravity Center Splitting Scheme

It is assumed that that the parameter domain $\mathcal{P} \subset \mathbb{R}^p$ and each subdomain are given by p-dimensional hypercubes. In the refinement step, we cut the current subdomain $\mathcal{P}^j$ into $J_{\text{add}} = 2^p$ subhypercubes. The splitting is based on a so-called "gravity center" $\bar{\mu}^j$ which is evaluated using the parameters that correspond to the selected basis functions of the EIM in the subdomain $\mathcal{P}^j$,

$$\bar{\mu}^j := \frac{1}{M^h_{\max}} \sum\nolimits_{M=1}^{M^h_{\max}} \mu^j_M.$$

The gravity center denotes the point of $\mathcal{P}^j$ that all 2^p new subdomains share, i.e., the coordinates of $\bar{\mu}^j$ define the splitting positions of $\mathcal{P}^j$. Figure 1 exemplarily shows two refinement steps for the square $\mathcal{P} = [0, 1]^2$. First, the square is split based upon the gravity center $\bar{\mu}^1 = [0.35, 0.40]$. The subdomain in the upper right corner is then divided based upon the gravity center $\bar{\mu}^2 = [0.75, 0.60]$.

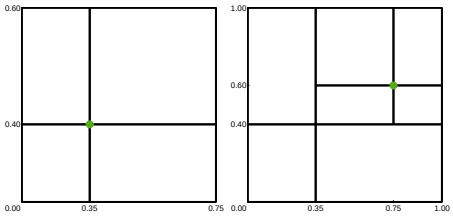


Fig. 1 Two refinement steps using the gravity center splitting scheme for $\mathcal{P} = [0,1]^2$. Gravity centers $\bar{\mu}^1 = [0.35, 0.4]$ for the first step (left) and $\bar{\mu}^2 = [0.75, 0.6]$ for the second step (right).

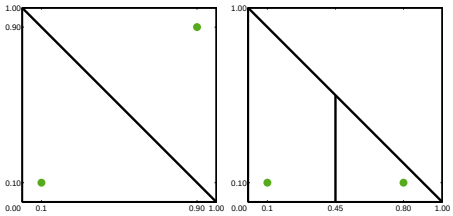


Fig. 2 Two refinement steps using the anchor point splitting scheme for $\mathcal{P} = [0,1]^2$. Anchor points for the first (left) and second refinement step (right).

The online assignment of a new parameter $\mu \in \mathcal{P} \subset \mathbb{R}^p$ to the appropriate subdomain is done using a tree search. Only the gravity centers have to be stored to completely define the final partition as well as the partition tree. In each step, the identification of the next subdomain is of complexity $\mathcal{O}(p)$. Thus, for a well balanced tree of depth $\mathcal{O}(\log J)$, the assignment complexity reads $\mathcal{O}(\log J \cdot p)$.

2.2.2 Anchor Point Splitting Scheme

The anchor point splitting scheme divides the current parameter domain $\mathcal{P}^j$ into $J_{\text{add}} = 2$ subdomains, independently of its shape and dimension. For the splitting, it is assumed that one can define a distance measure $d : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ on the parameter domain. The two subdomains are then specified by the proximity to the parameters μ_1^j and μ_2^j — the so-called anchor points — that have been returned by the procedure addBasisFunction$(\cdot)$ in line 3 of Algorithm 2 and correspond to the two first selected EIM basis functions in the subdomain $\mathcal{P}^j$. Then, the new subdomains in line 7 of Algorithm 2 are defined in the following way,

$$\begin{aligned} \mathcal{P}^{J+1} &:= \{\mu \in \mathcal{P}^j \,|\, d(\mu, \mu_1^j) < d(\mu, \mu_2^j)\}, \\ \mathcal{P}^{J+2} &:= \{\mu \in \mathcal{P}^j \,|\, d(\mu, \mu_2^j) \leq d(\mu, \mu_1^j)\}. \end{aligned} \tag{4}$$

Figure 2 exemplarily shows two refinement steps using the anchor point splitting scheme for $\mathcal{P} = [0,1]^2$. In the first step (left), the anchor points $\mu_1^1 = [0.1, 0.1]$ and $\mu_2^1 = [0.9, 0.9]$ have been used. In the second step, the anchor points $\mu_1^2 = [0.1, 0.1]$ and $\mu_2^2 = [0.8, 0.1]$ lead to the separation parallel to the y-coordinate at $x = 0.45$.

Since only two anchor points are needed for the next refinement step, it is enough to set $M_{\max}^h = 2$. Furthermore, the two subdomains can inherit the basis function of the "parent" domain that corresponds to their respective anchor point. In other words, for the domain $\mathcal{P}^j$ with the two "child" subdomains $\mathcal{P}^{J+1}$ and $\mathcal{P}^{J+2}$ as defined in (4), we have $\mu_1^{J+1} := \mu_1^j$ and $\mu_1^{J+2} := \mu_2^j$. Thus, only one more iteration has to be performed for each new subdomain.

We use a tree search in the online stage to find the subdomain of a new parameter $\mu \in \mathcal{P}$. We iteratively select the nearest anchor point and "move" to the corresponding subdomain. The evaluation of the distances to the anchor points is of complexity $\mathcal{O}(p)$. Assuming a balanced tree of depth $\mathcal{O}(\log J)$, the total tree search is again of complexity $\mathcal{O}(\log J \cdot p)$.

3 Partitioning of Unknown Parameter Domains

3.1 Unknown Parameter Domains

Let us start with the illustration of the concept of unknown parameter domains using some practical examples. First, one may consider coefficient functions of PDEs that are based upon measurements. On the one hand, underlying parameters can be hidden since the information of the system that produces the measured outcome is not completely accessible. On the other hand, the measured input functions could be non-parametric and merely belong to a common class of functions in terms of boundedness, regularity, or similar shape. Another application of unknown parameters are stochastic inputs, where the "parameter domain" can be seen as a set Ω of stochastic events that does not imply a feasible metric. Hence, the theory of compact parameter domains does not apply [12]. As an example of such events, one may consider the porosity structure of any physical medium such as sandstone or Li-ion batteries.

In the following, it is assumed that the input coefficient functions can be obtained without the detailed knowledge of any underlying parameter or stochastic event. Hence, no information about the parameter domain is required, and therefore, no distance measures on the parameter domain can be assumed to exist. We now define the family of possible input functions by

$$\mathcal{M} := \{c(\mu) : D \to \mathbb{R} \,|\, \mu \in \mathcal{P}\}, \tag{5}$$

where $D \subset \mathbb{R}^d$ denotes a bounded spatial domain. The parameter $\mu \in \mathcal{P}$ can also be interpreted as a reference to an arbitrary real life event that underlies the function $c(\mu)$, or just as an index to the associated $c(\mu) \in \mathcal{M}$. In any case, μ is not a parameter in the classical sense and the p- or hp-Partitioning are not applicable.

Another interpretation could be to consider the whole function $c(\mu)$ as a parameter, i.e., to consider a parameter function $\mu(x)$ in a certain function space $\mathcal{M}$. The subsequent theory and methods remain valid for such cases.

3.2 Affine Decomposition for Unknown Parameters

For the application of the EIM even for unknown parameter domains or arbitrary sets of functions, and for the applicability of partitioning methods, we postulate:

Assumption 1 *A mechanism is available that delivers arbitrarily many functions $c(\mu) \in \mathcal{M}$ as defined in (5). For any given $\varepsilon > 0$, it is possible to create a finite training set of functions $\mathcal{M}_{\mathrm{tr}} \subset \mathcal{M}$ of cardinality $n_{\mathrm{tr}} \in \mathbb{N}$ that sufficiently covers the variety of $\mathcal{M}$ up to the maximal error tolerance ε, i.e.,*

$$\sup_{c(\mu)\in\mathcal{M}} \inf_{v\in span(\mathcal{M}_{\mathrm{tr}})} \|c(\mu) - v\|_X \leq \varepsilon \tag{6}$$

for a given norm $\|\cdot\|_X$. Furthermore, let $\mathcal{M}$ be replaced by any subset $\mathcal{M}^0 \subset\subset \mathcal{M}$ with significantly less variation, i.e., of less complexity. Then, $\mathcal{M}_{\mathrm{tr}}$ can be replaced by a subset $\mathcal{M}^0_{\mathrm{tr}} \subset\subset \mathcal{M}_{\mathrm{tr}}$ of significantly less cardinality $n^0_{\mathrm{tr}} \ll n_{\mathrm{tr}}$ such that (6) still holds.

Now, the offline and online EIM Algorithms from Section 2.1 can directly be adopted. Instead of a training parameter set Ξ_{tr} for the Greedy step (in line 2 of Algorithm 1), we can directly use the training functions $\mathcal{M}_{\mathrm{tr}}$. For any function $c \in \mathcal{M}$, we can evaluate the vector $\boldsymbol{c}_M := (c(t_i))_{i=1}^M$ and compute the coefficients $\boldsymbol{\theta}_M(c) = (\theta_i(c))_{i=1}^M$ using the linear system (2) without the knowledge of a possibly underlying parameter.

3.3 Implicit Partitioning Problem Formulation

We now formulate the tasks and the main idea of the IPM. Input functions that are based upon unknown parameters naturally do not directly admit for an affine decomposition. Hence, the partitioning is connected to the EIM as we have already seen for the *hp*-Partitioning. We define the implicit partitioning problem.

Problem 1 (Implicit Partitioning Problem) *For a family of input functions $\mathcal{M}$ that suffices Assumption 1, create a partition of the parameter domain,*

(a) without the use of an explicit description of either $\mathcal{P}$ or $\mathcal{M}$,
(b) without an explicit description of the partitions and subdomains,
(c) with efficient and suitable assignments of new input functions $c(\mu)$.

For each subdomain, create separate affine decompositions with respect to the unknown parameter. The partition is supposed to be fine enough such that

(d) the affine approximations are precise up to a tolerance $\varepsilon_{\mathrm{tol}}$,
(e) the number of collateral basis functions per subdomain does not exceed $M_{\max}$.

The basic idea of the following implicit partitioning methods is the construction of several EIM bases that cover different parts of the family of input functions $\mathcal{M}$. As opposed to the *hp*-Partitioning, the splitting of the parameter domain is based upon the proximity of functions in $\mathcal{M}$ to the spaces spanned by the different collateral EIM bases and not on geometrical aspects of the parameter domain.

Under Assumption 1, it is possible to generate a training set of functions $\mathcal{M}_{\mathrm{tr}} \subset \mathcal{M}$ of cardinality $n_{\mathrm{tr}} \in \mathbb{N}$ that sufficiently covers the complexity of $\mathcal{M}$. Furthermore, the second part of Assumption 1 assures that a partitioning based upon a training set $\mathcal{M}_{\mathrm{tr}}$ is possible under the condition that $\mathcal{M}$ itself can be split into several parts of less complexity.

The presented implicit partitioning methods could also be seen as a partitioning of the family $\mathcal{M}$ or of the space spanned by $\mathcal{M}$. Thus, functions $c(\mu)$ are assigned to an appropriate subspace of $\operatorname{span}(\mathcal{M})$ rather than μ is assigned to a subdomain of $\mathcal{P}$. However, for an easier understanding, we stay in the parameter setting and refer to parameters and subdomains. We construct the structs $\mathcal{S}^j_{\mathrm{EIM},M}$, $j = 1, \ldots, J$, that contain the complete EIM data for all subdomains. These structs also define subspaces $\mathcal{M}^j$ of dimension M which correspond to the parameter subdomains $\mathcal{P}^j$, $j = 1, \ldots, J$. In the following, we just refer to subdomain j and mean the subdomain defining components $\mathcal{P}^j$, $\mathcal{M}^j$, or $\mathcal{S}^j_{\mathrm{EIM},M}$.

4 Moving Shapes IPM

We introduce different implicit partitioning procedures. As mentioned before, their common approach is the construction of several EIM bases that are supposed to

Algorithm 3 MovingShapesIPM($\mathcal{M}_{\mathrm{tr}}$, $\varepsilon_{\mathrm{tol}}$, J)

1 **set** $M = 0$
2 **repeat**
3 $M = M + 1$
4 **if** M==1 **then**
5 $\mathcal{S}^0_{\mathrm{EIM},J+1}$ = doInitialEIM($\mathcal{M}_{\mathrm{tr}}$, $J+1$)
6 $\{\mathcal{S}^1_{\mathrm{EIM},1}, \ldots, \mathcal{S}^J_{\mathrm{EIM},1}\}$ = initialFirstBasisFunction($\mathcal{S}^0_{\mathrm{EIM},J+1}$, J)
7 **else**
8 **for** $j = 1$ **to** J **do**
9 $\mathcal{S}^j_{\mathrm{EIM},M}$ = addBasisFunction($\mathcal{S}^j_{\mathrm{EIM},M-1}$, $\mathcal{M}^j_{\mathrm{tr}}$)
10 **end for**
11 **end if**
12 $\{\mathcal{I}^1_M, \ldots, \mathcal{I}^J_M\}$ = getOfflineAssignment($\mathcal{S}^1_{\mathrm{EIM},M}, \ldots, \mathcal{S}^J_{\mathrm{EIM},M}, \mathcal{M}_{\mathrm{tr}}$)
13 **for** $j = 1$ **to** J **do**
14 $\mathcal{M}^j_{\mathrm{tr}} = \{c(\mu) \in \mathcal{M}_{\mathrm{tr}} \,|\, \mu \in \mathcal{I}^j_M\}$
15 $\varepsilon^j_{M,\max}$ = getMaxError($\mathcal{S}^j_{\mathrm{EIM},M}$, $\mathcal{M}^j_{\mathrm{tr}}$)
16 **end for**
17 **until** $\max\limits_{j\in\{1,\ldots,J\}} \left\{\varepsilon^j_{M,\max}\right\} < \varepsilon_{\mathrm{tol}}$
18 **return** $\left\{\mathcal{S}^1_{\mathrm{EIM},M}, \ldots, \mathcal{S}^J_{\mathrm{EIM},M}\right\}$

cover different parts of the family of input functions $\mathcal{M}$. The first procedure, the Moving Shapes (MS) Implicit Partitioning Method (IPM), simultaneously generates the number of J EIM bases for a previously fixed number J of subdomains. It is desired that the partition is formed such that the complexity of $\mathcal{M}$ is equally distributed on the J different subdomains and the least possible number of basis functions is obtained. This is achieved by letting the subdomains reshape in each iteration instead of using a fixed partition. Thus, the actual partition depends on the used number M of basis functions.

4.1 Outline of the Method

The MS IPM is described in Algorithms 3 and 4. Let J denote the desired number of subdomains and let $\varepsilon_{\mathrm{tol}} > 0$ be the desired approximation error tolerance. Furthermore, let the set of training parameters be given by $\{\mu_1, \ldots, \mu_{n_{\mathrm{tr}}}\}$ such that the set of training functions reads $\mathcal{M}_{\mathrm{tr}} = \{c(\mu_n) \,|\, n = 1, \ldots, n_{\mathrm{tr}}\}$. Algorithm 3 generates EIM data structs $\mathcal{S}^j_{\mathrm{EIM},M}$, $j = 1, \ldots, J$, $M \in \mathbb{N}$. Since the number of subdomains and the error tolerance $\varepsilon_{\mathrm{tol}}$ are (at least for now) fixed, we do not set a maximal number of basis functions per subdomain, differently to the *hp*-Partitioning where $M_{\max}$ and $\varepsilon_{\mathrm{tol}}$ were fixed and J was flexible.

We start the description of the MS IPM with the initialization of the EIM structs $\mathcal{S}^j_{\mathrm{EIM},1}$, $j = 1, \ldots, J$, in the first iteration of the loop in Algorithm 3, for $M = 1$. In line 5, we perform $J + 1$ steps of the normal EIM, based upon the training set $\mathcal{M}_{\mathrm{tr}}$ and without any partitioning. We refer to this step as initial EIM and denote the resulting EIM struct by $\mathcal{S}^0_{\mathrm{EIM},J+1}$. Then, in line 6, we discard the first basis function of $\mathcal{S}^0_{\mathrm{EIM},J+1}$ and distribute the remaining J functions that have been selected by the initial EIM to the EIM structs $\mathcal{S}^j_{\mathrm{EIM},1}$, $j = 1, \ldots, J$, as initial basis functions, respectively. Neglecting the first basis function of the initial EIM

Algorithm 4 getOfflineAssignment($\mathcal{S}^1_{\mathrm{EIM},M}, \ldots, \mathcal{S}^J_{\mathrm{EIM},M}$, $\mathcal{M}_{\mathrm{tr}}$)

1 $\mathcal{I}^1 = \ldots = \mathcal{I}^J = \{\}$
2 **for** $n = 1$ **to** n_{tr} **do**
3 **for** $j = 1$ **to** J **do**
4 $\varepsilon^j_M(\mu_n) = \mathsf{getError}(\mathcal{S}^j_{\mathrm{EIM},M}, c(\mu_n))$
5 **end for**
6 $i = \arg\inf_j \{\varepsilon^j_M(\mu_n) \,|\, j = 1, \ldots, J\}$
7 $\mathcal{I}^i_M = \mathcal{I}^i_M \cup \{\mu_n\}$
8 **end for**

is not crucial but, in our experiments, it led to a more balanced initial distribution of the complexity to the subdomains.

In line 12 of Algorithm 3, we call the procedure $\mathsf{getOfflineAssignment}(\cdot)$ that is further described in Algorithm 4. For each subdomain j, the procedure returns a set of assigned parameters $\mathcal{I}^j_M$ that refer to the corresponding functions in $\mathcal{M}_{\mathrm{tr}}$, where the assignment is based upon the EIM approximation error. In detail, for a given parameter μ_n, $n \in \{1, \ldots, n_{\mathrm{tr}}\}$, we evaluate the EIM approximation error of the corresponding function $c(\mu_n)$ in all subdomains. This is performed in Algorithm 4, line 3 to 5. Then, the parameter is assigned to the subdomain that best approximates $c(\mu_n)$ in line 6 and 7. Note that we distinguish $\mathcal{P}^j$ used for the *hp*-Partitioning from $\mathcal{I}^j$. While $\mathcal{I}^j$ denotes a discrete set of parameters, $\mathcal{P}^j$ provides the explicit description of the complete subdomain j.

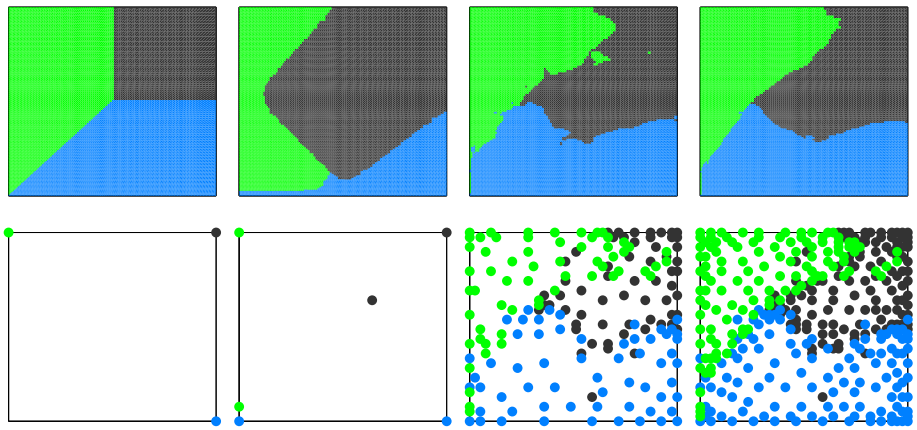
The further steps work similar to Algorithm 2, but simultaneously for all subdomains. In line 15 of Algorithm 3, we evaluate the maximal error on each subdomain, or more precisely, the maximal error out of the set of currently assigned functions $\mathcal{M}^j_{\mathrm{tr}} := \{c(\mu) \in \mathcal{M}_{\mathrm{tr}} \,|\, \mu \in \mathcal{I}^j_M\}$. In line 17, we check if all maximal errors fall below the tolerance $\varepsilon_{\mathrm{tol}}$. We do not stop the basis extensions until convergence on all subdomains is obtained, i.e., even if the tolerance is reached on a certain subdomain, we add more basis functions if the error on other subdomains still exceeds the desired value. For $M > 1$, the basis extension is done in line 9. We select the so far worst approximated function of the subdomain j, i.e., out of $\mathcal{M}^j_{\mathrm{tr}}$.

A new effect in comparison to the *hp*-Partitioning is that the basis extension also changes the shape of the partitions. The selection of a new basis function $c(\mu^j_M)$ for some μ^j_M located close to the boundary of the subdomain j yields a movement of the respective shape towards the just selected parameter since the assignment of parameters is based upon the EIM approximation error. Functions $c(\mu)$ close to $c(\mu^j_M)$ will be assigned to subdomain j in the next iteration.

This effect is illustrated in Figure 3. It provides the result of the MS IPM for an explicitly given parametric function $c : \mathcal{D} \times \mathcal{P} \to \mathbb{R}$ on the spatial domain $\mathcal{D} = [0, 1]^2$ and with parameters $\mu = (\mu_1, \mu_2) \in \mathcal{P} = [0.3, 0.7]^2$, given by

$$c(x; \mu) = e^{-50\left((x_1-\mu_1)^2+(x_2-\mu_2)^2\right)}. \tag{7}$$

In detail, Figure 3 shows the partitions of the parameter domain after $M = 1$, 2, 60, and 120 iterations in the top row with a resolution of $40 \cdot 40$ pixels. The respective parameters that have been selected for the bases extensions are provided in the bottom row. The shapes of the subdomains change especially during the first iterations. E.g., for $M = 2$, the black part selected a basis function that



(a) $M = 1$ (b) $M = 2$ (c) $M = 60$ (d) $M = 120$

Fig. 3 MS IPM subdomains (top row) and selected parameters for basis extension (bottom row) for four different basis sizes M.

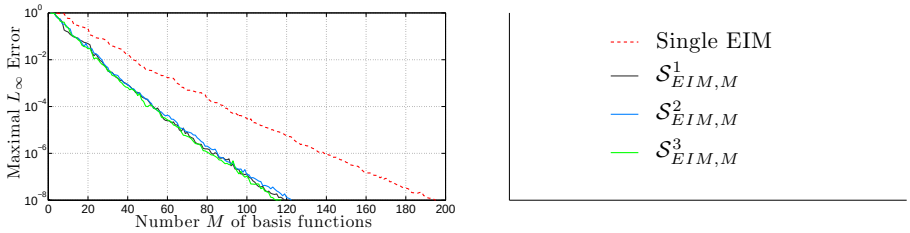


Fig. 4 Convergence of the MS IPM for $J = 3$ compared to a single EIM.

corresponds to a parameter from the lower left corner of the black subdomain for $M = 1$. Therefore, it "takes over" huge parts of the other subdomains. In the iterations $M = 60$ to $M = 120$, the basis extensions are mostly based upon parameters inside the subdomains and the boundaries do not change significantly.

The objective of the subdomain reshaping is a more effective use of the basis functions. For fixed shapes, the first basis functions are usually selected at the border of subdomains. Consequently, adjacent subdomains would select basis functions that cover the same area. Furthermore, the reshaping results in a good distribution of the complexity of $\mathcal{M}$ on the different subdomains. The subdomains are likely to be formed such that the respective numbers of basis functions necessary for a given approximation tolerance differs only very slightly. In Figure 4, we confirm this assumption for the given example. The figure shows the error convergence of a single EIM without partitioning and the convergence result using the MS IPM and $J = 3$ subdomains. More examples are provided in Section 6.

It can be observed in Figure 3(b) that two subdomains can select basis functions close to each other in the same step which is not optimal but very difficult to avoid. Successive instead of simultaneous basis extensions lead to extremely unbalanced convergence. No general heuristic that works as a black box for all kind of input functions is known but would be necessary in the case of unknown parameters.

Algorithm 5 getOnlineAssignment($\mathcal{S}^1_{\mathrm{EIM},M^+}, \dots, \mathcal{S}^J_{\mathrm{EIM},M^+}$, $c(\mu)$, M, M^+)

```
for j = 1 to J do
    θ^j_{M+}(μ) = getCoefficients(S^j_{EIM,M+}, c(μ))
    Δ^j_{M,M+}(μ) = Σ_{m=M+1}^{M+} |θ^j_m(μ)|
end for
i = arg inf_j {Δ^j_{M,M+}(μ) | j = 1, ..., J}
return {i, θ^i_{M+}(μ)}
```

Even though we obtain very balanced convergence rates, we can not completely prevent that two subdomains partially cover the same part of the parameter domain. It can be seen in Figure 3(c) and 3(d) that some of the selected basis functions are separated and enclosed by a different subdomain. However, it is not possible to discard such functions from the basis even for values of M where they are separated from their subdomain. In other words, they still play an important role for the approximation quality.

Let $\mathcal{N}$ be the number of degrees of freedom of the discretized functions in $\mathcal{M}$. Then, the complexity of an iteration in the offline stage of the MS IPM consists of $\mathcal{O}(JM^2 \cdot n_{\mathrm{tr}})$ for the computation of the approximations of the training samples in $\mathcal{M}$, $\mathcal{O}(JM^2 \cdot n_{\mathrm{tr}} \cdot \mathcal{N})$ for the evaluation of the EIM errors, and $\mathcal{O}(J \cdot n_{\mathrm{tr}})$ to assign the training snapshots to the subdomains. Thus, the total complexity is given by $\mathcal{O}(JM^2 n_{\mathrm{tr}} \mathcal{N})$.

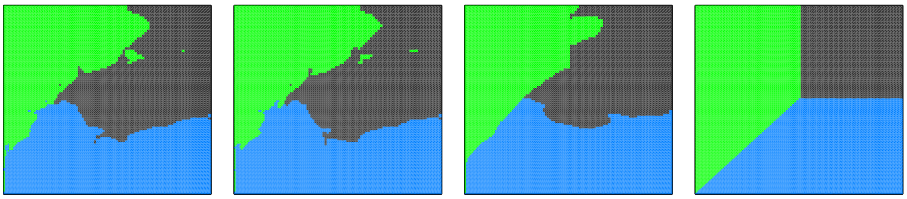
4.2 Online Assignment

In the online stage, it is not possible to evaluate the exact EIM approximation errors independently of the dimension $\mathcal{N}$. Hence, the assignment is now based upon the EIM error estimator. The straightforward procedure is given in Algorithm 5. For a new parameter μ and an input function $c(\mu)$, we evaluate the coefficients $\boldsymbol{\theta}^j_{M^+}(\mu)$ and error bounds $\Delta^j_{M,M^+}(\mu)$ for all $j = 1, \dots, J$. Then, we select the subdomain with the smallest error estimator. The algorithm returns the selected subdomain i and the corresponding coefficients $\boldsymbol{\theta}^i_{M^+}(\mu)$ that can be used for the further processing of the input function $c(\mu)$.

It is clear that the assigned subdomain is not necessarily optimal in the sense of the real error. However, it is not essential that we hit the best subdomain but to select a sufficiently precise approximation. Figure 5(b) shows the online assignment based upon the smallest error estimator for the example provided in (7) for $M = 60$ and $M^+ = 66$. In comparison to the partition based upon the exact EIM approximation error in Figure 5(a), only minor deviations can be observed. The use of more than 6 coefficients for the error estimates would furthermore lead to results closer to the "true" partition.

4.2.1 Online Complexity

The online complexity for the assignment of a new parameter $\mu \in \mathcal{P}$ to the appropriate subdomain according to Algorithm 5 consists of $\mathcal{O}(JM)$ for the evaluation of $c(\mu) \in \mathcal{M}$ at the interpolation points, $\mathcal{O}(JM^2)$ for the computation of the



(a) "True" partition (b) Online assignment using Alg. 5 (c) Online assignment using Alg. 6 and $\varepsilon_{\text{tol}} = 10^{-4}$ (d) First assignment trial in Alg. 6 using $\arg\max |\theta_1^j|$

Fig. 5 MS IPM online assignments for $M = 60$ and $M^+ = 66$.

coefficients and the error bounds, and $\mathcal{O}(J)$ for the actual assignment to the subdomain. Thus, the total complexity reads $\mathcal{O}(JM^2)$, where it has been assumed that $M^+ = \mathcal{O}(M)$.

The assignment according to Algorithm 5 is independent of the dimension of the parameter domain. Yet, compared to the *hp*-Partitioning, where the online complexity for the assignment and the evaluation of the coefficients sums up to $\mathcal{O}(p \log J + M^2)$, the costs seem to increase significantly. Nevertheless, the online complexity of the MS IPM is acceptable. On the one hand, the number M of basis functions decreases with increasing number J of subdomains. In the current example, the run-time is approximately constant in J. On the other hand, the main complexity in the context of RB methods commonly amounts to $\mathcal{O}(M^2 \cdot N^2 + N^3)$, where N denotes the number of basis functions for the reduced basis. Since separate reduced bases are constructed for each subdomain, N is decreasing in J, too. Hence, the most expensive computations in the RB context decrease significantly.

4.2.2 Improved Online Complexity

The key requirement of the assignment is to obtain adequate approximations but not to find the *best* subdomain. As a consequence, it is possible to break the loop over j in Algorithm 5 as soon as $\Delta^j_{M,M^+}(\mu)$ falls below ε_{tol}. Then, the average online complexity is already reduced to half. In Algorithm 6, we present a heuristic that provides a more suitable search order of the subdomains than just checking the error estimators step by step.

The EIM is generated in a form such that the importance of the basis functions decreases in M. In other words, the coefficients of the first basis functions are usually larger than the following ones. At the same time, using a collateral basis that does not fit to the input data, the coefficients are rather equally distributed over all basis functions and the first coefficients are therefore comparatively small. We use this effect for the search order heuristic. In line 2 of Algorithm 6, we evaluate only the first coefficients θ_1^j of the affine approximations of a given input function $c(\mu)$ for all subdomains $j = 1, \ldots, J$. In line 4, we sort these coefficients in descending order with respect to their absolute values and return an ordered list of subdomains. Then, we iteratively check if the error estimator of the subdomains fall below the tolerance ε_{tol}, starting with the subdomain with the largest coefficent

Algorithm 6 getFastOnlineAssignment($\mathcal{S}^1_{\mathrm{EIM},M^+}, \dots, \mathcal{S}^J_{\mathrm{EIM},M^+}$, $c(\mu)$, M, M^+)

```
for j = 1 to J do
    θ_1^j(μ) = getCoefficients(S^j_{EIM,1}, c(μ), 1)
end for
{j_1, ..., j_J} = sortCoefficientsDescending(|θ_1^1(μ)|, ..., |θ_1^J(μ)|)
for k = 1 to J do
    θ^{j_k}_{M+}(μ) = getCoefficients(S^{j_k}_{EIM,M+}, c(μ), M+)
    Δ^{j_k}_{M,M+}(μ) = Σ_{i=M+1}^{M+} |θ_i^{j_k}(μ)|
    if Δ^{j_k}_{M,M+}(μ) < ε_tol then return {j_k, θ^{j_k}_{M+}(μ)} end if
end for
```

$|\theta_1^j|$. Once we find the first subdomain that approximates $c(\mu)$ sufficiently precise, we return the subdomain and the corresponding coefficients.

Figure 5(c) shows the online assignment based upon Algorithm 6 for $\varepsilon_{\mathrm{tol}} = 10^{-4}$, $M = 60$, and $M^+ = 66$. The partition reveals some larger deviations compared to the "true" partition 5(a) and the direct assignment 5(b) based upon Algorithm 5. Still, both the error estimator and the true error fall below $\varepsilon_{\mathrm{tol}}$ for all parameters.

In Figure 5(d), the result of the heuristic of Algorithm 6 is provided. It shows the first assignment attempt, i.e., the assignment based upon the largest first coefficient. We can see that large parts coincide with the assignment in Figure 5(c). In fact, 80.8% of the parameters in Figure 5(d) are associated to the same subdomain as in Figure 5(c) and are therefore directly assigned after just one iteration. Hence, in most cases, the online complexity reduces to $\mathcal{O}(J + M^2)$. For another 17.6%, we need two attempts until a subdomain is found that approximates the corresponding function sufficiently well. For only 1.6% of the parameters, we have to evaluate the coefficients for all subdomains. More examples are provided in Section 6.

4.3 Refinement Procedure

So far, we fixed the number of subdomains in advance, whereas for many applications, a certain maximal number $M_{\max}$ of basis functions is desired and the necessary number of subdomains is unknown. Thus, we start the MS IPM with an initial guess J_0 of needed subdomains. Once we detect that $M_{\max}$ will be reached but the error still exceeds the tolerance $\varepsilon_{\mathrm{tol}}$, we need a refinement of the partition. In contrast to the *hp*-Partitioning, it is not possible for the MS IPM to directly divide a subdomain into several parts. Hence, a refinement now yields to a complete restart of the procedure with an increased number of subdomains.

It is too expensive to perform the MS IPM until $M_{\max}$ is reached before a refinement is performed. We could directly adopt the idea of the *hp*-Partitioning and define additional quantities $M^h_{\max} \ll M_{\max}$ and $\varepsilon^h_{\mathrm{tol}} \gg \varepsilon_{\mathrm{tol}}$ to separate h- and p-part. However, the procedure may still be expensive and it is very difficult to define $\varepsilon^h_{\mathrm{tol}}$ and $M^h_{\max}$ such that the final number of subdomains is indeed sufficient. Hence, it may happen that additional expensive refinements are needed.

Alternatively, we can adopt the prediction methodology from a different partitioning methodology presented in [6]. We start the MS IPM as described in Algorithm 3 with an initial number J of subdomains. Let $M_J(\varepsilon_{\mathrm{tol}})$ denote the

number of basis functions that are necessary to reach the tolerance ε_{tol} using J subdomains. After each basis extension, before line 17 of Algorithm 3, we predict $M_J(\varepsilon_{\text{tol}})$ by extrapolating the maximal errors of the previous steps. We denote the prediction of $M_J(\varepsilon_{\text{tol}})$ by $M_J^{\text{pred}}(\varepsilon_{\text{tol}})$. If $M_J^{\text{pred}}(\varepsilon_{\text{tol}}) \leq M_{\max}$, we proceed the basis extension. Otherwise, we increase the number of subdomains to some $J_{\text{new}} > J$ and restart the MS IPM.

For the efficiency of this refinement procedure, it is crucial to appropriately select the new number of subdomains. In the ideal case, a perfectly separable family of functions $\mathcal{M}$, the relation $J_0 M_{J_0}(\varepsilon_{\text{tol}}) = J_1 M_{J_1}(\varepsilon_{\text{tol}})$ would hold. Hence, the new number of subdomains should be determined by $J_{\text{new}} = J M_J^{\text{pred}}(\varepsilon_{\text{tol}})/M_{\max}$. However, this ideal case is very unrealistic and provides only a lower bound for the actually needed number of subdomains. Instead, we assume a nonlinear dependence and use

$$J_{\text{new}} = J \cdot \left(\frac{M_J^{\text{pred}}(\varepsilon_{\text{tol}})}{M_{\max}} \right)^{\alpha} \tag{8}$$

for some $\alpha > 1$. The exponent α depends on the separability of $\mathcal{M}$. We therefore start with a rather small α, e.g., $\alpha = 2$. If further refinement steps are necessary, α can be increased step by step.

5 Fixed Shapes IPM

In the previous section, we developed a partitioning method for unknown parameter domains that is very flexible and automatically adapts the shapes to the given problem. The convergence in the different subdomains is well-balanced, the online assignments are adequate, and, for the majority of parameters, fast.

However, the refinement procedure can be relatively expensive since the bases on all subdomains are discarded and a complete restart is necessary. Furthermore, it is common in the EIM context to adaptively determine the number M of basis functions. Coefficients are added until the error estimator is precise enough. This can usually be done without an increased complexity. Since the use of more basis functions may now yield a shift to a different subdomain, this adaptive selection of M is difficult in the context of moving shapes.

The objective of the Fixed Shapes (FS) IPM in this section is the development of an adaptive implicit partitioning method that fulfills the requirements of the Implicit Partitioning Problem 1 but allows a fast refinement procedure and an adaptive use of the number of basis functions M. Furthermore, the assignment of parameters are supposed to be based upon a tree based structure. Altogether, we could decrease offline and online complexity.

5.1 Outline of the Method

For the MS IPM, we accelerate the assignment using the heuristic introduced in Algorithm 6 which is based upon the first coefficients of the affine approximations. It seems to be an appropriate idea to use the heuristic not only in the online stage but for the complete partitioning procedure. In other words, we already base the splitting of a subdomain upon the largest coefficients. This assignment is

Algorithm 7 FixedShapesIPM($\mathcal{S}^j_{\mathrm{EIM},M_0}$, $\mathcal{M}^j_{\mathrm{tr}}$, $M_{\max}$, $\varepsilon_{\mathrm{tol}}$, J)

```
1  for M = M_0 + 1 to M_max do
2      S^j_{EIM,M} = addBasisFunction(S^j_{EIM,M-1}, M^j_tr)
3      ε^j_{M,max} = getMaxError(S^j_{EIM,M}, M^j_tr)
4      if ε^j_{M,max} < ε_tol then return S^j_{EIM,M} end if
5  end for
6  {S^{J+1}_{EIM,1}, ..., S^{J+J_add}_{EIM,1}} = initialFirstBasisFunction(S^j_{EIM,J_add+1}, J_add)
7  {I^{J+1}, ..., I^{J+J_add}} = getCoefficientBasedAssignment(S^{J+1}_{EIM,1}, ..., S^{J+J_add}_{EIM,1}, M^j_tr, M_0)
8  J_new = J + J_add
9  for i = 1 to J_add do
10     M^{J+i}_tr = {c(μ) ∈ M^j_tr | μ ∈ I^{J+i}}
11     FixedShapesIPM(S^{J+i}_{EIM,M_0}, M^{J+i}_tr, M_max, ε_tol, J_new)
12 end for
```

Algorithm 8 getCoefficientBasedAssignment($\mathcal{S}^1_{\mathrm{EIM},M_0}, \ldots, \mathcal{S}^J_{\mathrm{EIM},M_0}$, $\mathcal{M}_{\mathrm{tr}}$, M_0)

```
1  I^1 = ... = I^J = {}
2  for n = 1 to n_tr do
3      for j = 1 to J do
4          θ^j_{M_0}(μ) = getCoefficients(S^j_{EIM,M_0}, c(μ), M_0)
5      end for
6      i = arg max_j {‖θ^j_{M_0}(μ)‖_1 , j = 1, ..., J}
7      I^i = I^i ∪ {μ_n}
8  end for
```

independent of the actually used number of basis function. Thus, the subdomains will be fixed and online and offline shapes exactly coincide.

The detailed procedure of this FS IPM is described in Algorithm 7. It reveals strong similarities to the *hp*-Partitioning of Algorithm 2. We assume that the initialization of an arbitrary number J of subdomains has been performed analogously to the initial step of the MS IPM, producing disjoint sub-training sets $\mathcal{M}^j_{\mathrm{tr}} \subset \mathcal{M}_{\mathrm{tr}}$ and the initial EIM structs $\mathcal{S}^j_{\mathrm{EIM},1}$ with one basis function. In the algorithm, we use a more general notation with an arbitrary initial number of basis functions M_0 that will be used later. For now, we constantly set $M_0 \equiv 1$. Algorithm 7 is started independently for each subdomain.

From line 1 to 5 of Algorithm 7, the already known EIM basis extension on subdomain j is performed, using always the same set of training samples. Once the maximal error falls below the tolerance $\varepsilon_{\mathrm{tol}}$, the EIM struct $\mathcal{S}^j_{\mathrm{EIM},M}$ is returned in line 4.

When $M_{\max}$ is reached without convergence, a refinement procedure has do be performed. Let J_{add} denote the number of new subdomains. As in the initial step of the MS IPM, we use the first $J_{\mathrm{add}} + 1$ selected basis functions of the current subdomain to initialize the new EIM structs in line 6. Again, we omit the first basis for a better distribution of the subdomains. In line 7, we assign the functions in $\mathcal{M}^j_{\mathrm{tr}}$ to the appropriate subdomain. The procedure is described in Algorithm 8 and is based upon the introduced heuristic. In detail, we evaluate the first coefficient of all training functions and for all new subdomains in line 4. In line 6 and 7, each parameter is assigned to the subdomain where the ℓ_1-norm of the corresponding coefficients is maximal. We define the new training sets

Algorithm 9 refineCoefficientBased($\mathcal{S}^1_{\mathrm{EIM},1}, \dots, \mathcal{S}^J_{\mathrm{EIM},1}$, $\mathcal{M}_{\mathrm{tr}}$, $\mathcal{S}^0_{\mathrm{EIM},M_{\max}}$, J_{init})

```
1  define δ ∈ [0,1)        % subdomain i will be rejected if |I^i| < δ |M_tr|/J
2  for M_0 = 1 to M_0^max do
3      {I^1,...,I^J} = getCoefficientBasedAssignment(S^1_{EIM,M_0},...,S^J_{EIM,M_0}, M_tr, M_0)
4      if |I^1|,...,|I^J| > δ |M_tr|/J then return {I^j, S^j_{EIM,M_0} | j=1,...,J} end if
5      {S^j_{EIM,M_0+1} | j = 1,...,J} = do MSIPM Step(S^1_{EIM,M_0},...,S^J_{EIM,M_0}, M_tr)
6  end for
7  for j = 1 to J do
8      if |I^i| < δ |M_tr|/J then
9          J_init = J_init + 1
10         S^j_{EIM,1} = newInitialBasisFunction(S^0_{EIM,M_max}, J_init)
11     end if
12 end for
13 refineCoefficientBased(S^1_{EIM,1},...,S^J_{EIM,1}, M_tr, S^0_{EIM,M_max}, J_init)
```

and recursively start Algorithm 7 for each new subdomain from line 9 to 12. For additional accelerations of the offline stage, the method can be combined with the *hp* methodology and/or the prediction techniques.

However, it is not always possible to directly apply the heuristic. Consider the example problem used in [1] for the introduction of the EIM,

$$c(x;\mu) = (x_1 + \mu_1)^2 + (x_2 + \mu_2)^2, \tag{9}$$

$x \in \mathcal{D} = [0,1]^2$, $\mu \in \mathcal{P} = [0.01, 1]^2$. Independently of μ, the maximum of c is located at $x_{\max} = (1,1)$. Hence, each subdomains selects the same first EIM interpolation knot $t_1 = x_{\max} = \arg\max_{x\in\mathcal{D}} q_1^j(x)$, where q_1^j denotes the first basis function in subdomain j. Then, for the approximation of a function $c(\mu)$, $\mu \in \mathcal{P}$, the first coefficients in all subdomains are equal to $c(x_{\max}, \mu)$ due to the L_∞-normalization of the basis functions. Hence, we follow a slightly different approach and use a flexible and adaptive number of coefficients for the assignment.

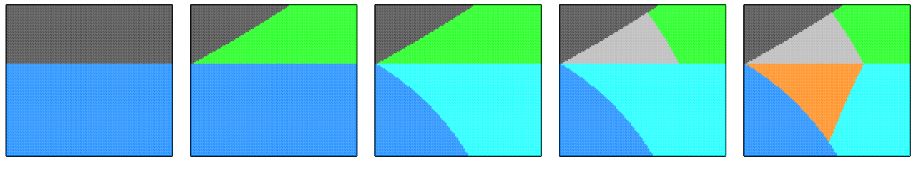
5.2 Adaptive Refinement

The only change in Algorithm 7 occurs in line 7. Instead of the direct use of the coefficient based assignment with $M_0 = 1$, we call

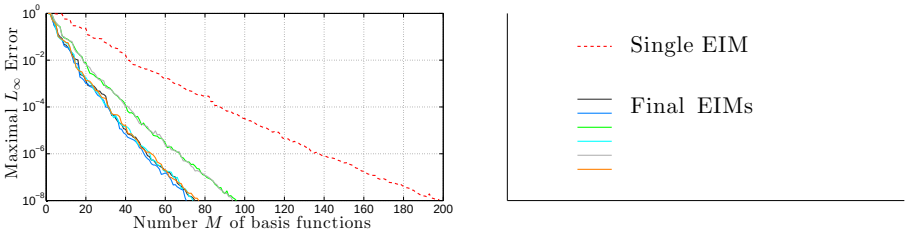
$$\begin{aligned}&\{\, \mathcal{I}^{J+i},\ \mathcal{S}^{J+i}_{\mathrm{EIM},M_0} \mid i=1,\dots,J_{\mathrm{add}} \,\}\\ &\quad = \mathsf{refineCoefficientBased}(\mathcal{S}^{J+1}_{\mathrm{EIM},1},\dots,\mathcal{S}^{J+J_{\mathrm{add}}}_{\mathrm{EIM},1},\ \mathcal{M}^j_{\mathrm{tr}},\ \mathcal{S}^j_{\mathrm{EIM},M_{\max}},\ J_{\mathrm{add}}+1)\end{aligned}$$

that is provided in Algorithm 9 and described in the following. It automatically detects the necessary number M_0 of used coefficients for an appropriate splitting of the domain.

Besides the initial EIM structs and the training functions, the input of the procedure also includes the EIM struct of the parent subdomain and the index of its last basis function that has been used as initial basis for a child subdomain. It returns not only the sets of assigned parameters $\mathcal{I}^{J+i}$ but also updated EIM structs $\mathcal{S}^{J+i}_{\mathrm{EIM},M_0}$, $i = 1,\dots,J_{\mathrm{add}}$, now with $M_0 \geq 1$ basis functions.



(a) refinement 1 (b) refinement 2 (c) refinement 3 (d) refinement 4 (e) refinement 5



(f) error decay in final subdomains

Fig. 6 Refinement steps using the FS IPM of Algorithm 7 and $J_{\text{add}} = 2$ ((a) – (e)). Convergence of the FS IPM for compared to a single EIM (f).

In line 1 of Algorithm 9, we define the acceptance condition for a subdomain. We require that the proportion of the number of assigned functions to a subdomain $|\mathcal{I}^i|$ and the average number of functions per subdomain $|\mathcal{M}_{\text{tr}}|/J$ is larger than δ. Otherwise, the subdomain is rejected. In our experiments, we use $\delta = 0.1$.

We start using a single coefficient for the assignment. Generally, for a given number M_0 of used coefficients, the coefficient based assignment procedure is already given in Algorithm 8. We evaluate the vector $\boldsymbol{\theta}^j_{M_0}(\mu)$ of the first M_0 approximation coefficients for all subdomains j and return the subdomain where the sum of the respective coefficients in absolute values is maximal.

If all subdomains fulfill the acceptance condition in line 4, we are done and return. For each subdomain, we furthermore store the number M_0 of coefficients used for the assignment. If at least one of the subdomains is rejected, we discard the assignment and try to use more coefficients. We perform one step of the MS IPM in line 5 to determine one additional basis function for each subdomain. Then, the assignment procedure is iterated.

Suppose a maximal number $M_0^{\max}$ of allowed coefficients is reached without an appropriate assignment, we reset the subdomains with new initial bases from line 7 to 12. For each subdomain j with $|\mathcal{I}^j| < \delta \frac{|\mathcal{M}_{\text{tr}}|}{J}$, we replace its initial basis by the next function of the parent subdomain that has not been used for any initial basis. we restart the whole procedure refineCoefficientBased$(\cdot)$ in line 13 with $M_0 = 1$. In other words, we discard all the selected basis functions except the first ones and use only the (partially new) initial EIM structs. Suppose $J_{\text{init}} > M_{\max}$, we can not assign a new initial basis and the algorithm has to be stopped without appropriate partitions.

In Figure 6, the refinements for the example provided in (7) with a desired accuracy of $\varepsilon_{\text{tol}} = 10^{-8}$ and a maximal number of basis function $M_{\max} = 100$ is shown. The initial partition has been given by the complete parameter domain and a constant number of $J_{\text{add}} = 2$ has been used. After five refinements, the

partition was fine enough. For the desired accuracy, the subdomains need between 75 and 96 basis functions. Hence, comparing with Figure 4, we can see that the convergence is now less balanced than for the MS IPM.

5.3 Online Assignment and Complexity

As mentioned before, we use the tree of subdomains to perform a very efficient tree search in the online stage. At the nodes in the tree, only M_0 basis functions have to be stored. Just for leaf subdomains, the complete EIM structs are stored. Let $c(\mu)$ be a new input function. At each node, we evaluate the respective number M_0 of coefficients $\boldsymbol{\theta}^j_{M_0}$ for all child subdomains j. The complexity reads $\mathcal{O}(J_{\text{add}} M_0^2) = \mathcal{O}(1)$ since both J_{add} and M_0 are very small compared to and independent of the actually used number of basis functions M. We move to the child subdomain where $\|\boldsymbol{\theta}^j_{M_0}\|_1$ is maximal until a leaf subdomain is reached. Hence, the assignment of a function to the right leaf subdomain can be achieved with complexity $\mathcal{O}(\log(J))$. In contrast to the *hp*-Partitioning, the complexity is independent of the dimension of the parameter domain.

It is possible to combine both implicit partitioning methods. To save offline run time but still generate flexible shapes and therefore a balanced convergence, it could be useful to first perform some steps of the FS IPM and generate a tree of subdomains. At some step, we switch to the MS IPM starting with the initial bases on the generated leaf subdomains. In the online stage, we first perform an efficient tree search to find the appropriate leaf subdomain. Then, the more expensive online assignment based upon Algorithm 5 on the final partition is only used for a smaller number of subdomains. Additionally, if in any case the FS IPM does not terminate and rejects all subdomains, it is still possible to proceed with the MS IPM to obtain the desired accuracy with less than $M_{\max}$ basis functions.

6 Numerical Examples and Comparisons

In this section, we consider three different examples to illustrate the different properties of the presented partitioning methods. For all examples, explicitly given parameter domains have been used. An additional example for stochastic input data can be found in [12]. We compare the results of the implicit partitioning with the *hp* methods and discuss advantages and disadvantages.

The desired L_∞ error tolerance in the construction of the partitions is given by $\varepsilon_{\text{tol}} = 10^{-8}$ for all examples. For the refinement steps of the FS IPM, J_{add} has been set to 2 for all cases to facilitate the comparison and to guarantee efficient tree structures. We rejected partitions if one of the two subdomain obtained less than 5% of the parameters of the parent subdomain. The maximal number of coefficients used for the assignment has been set to $M_0^{\max} = 6$.

No error prediction techniques to accelerate the offline process have been used in order not to generate more subdomains than necessary and to obtain "optimal" partitioning results. Consequently, we also used $\varepsilon^h_{\text{tol}} = \varepsilon^p_{\text{tol}}$ and $M^h_{\max} = M^p_{\max}$ for the *hp*-Partitioning.

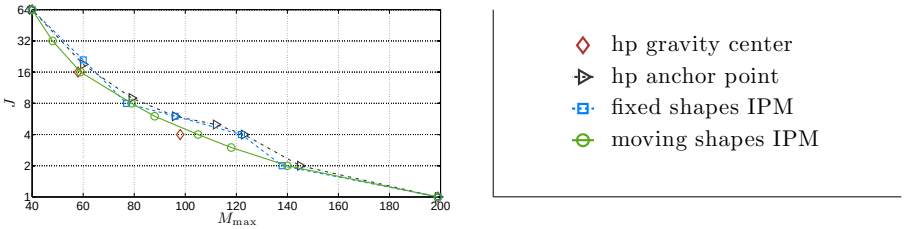
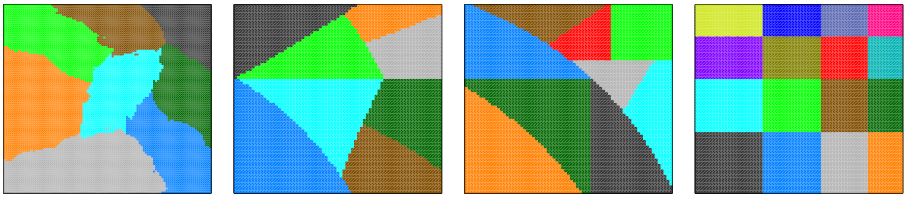


Fig. 7 Comparison: number of subdomains J necessary for a given maximal number of affine terms $M_{\max}$ for Example 1.



(a) MS IPM: $J = 8$ (b) FS IPM: $J = 8$ (c) hp anchor point: $J = 9$ (d) hp gravity center: $J = 16$

Fig. 8 Partitioning results for Example 1 with $M_{\max} = 80$ and $\varepsilon_{\text{tol}} = 10^{-8}$.

6.1 Example 1

We first consider the example adopted from [4] and already provided in (7). For the spatial domain $\mathcal{D} = [0,1]^2$ and the explicitly given parameter domain $\mathcal{P} = [0.3, 0.7]^2$, the input function $c : \mathcal{D} \times \mathcal{P} \to \mathbb{R}$ is given by

$$c(x;\mu) = e^{-50\left((x_1-\mu_1)^2+(x_2-\mu_2)^2\right)}.$$

For the discretization of the spatial domain, we used a uniform grid with edge length 0.02 such that the number of degrees of freedom is given by $\mathcal{N} = 2601$. The parameter samples for the offline stage are selected using a logarithmically distributed grid with 72 parameters in each direction and $n_{\text{tr}} = 5184$ samples.

In Figure 7, we compare the efficiency of the implicit methods with the *hp* results. For given maximal number of basis functions $M_{\max}$, the respective numbers J of generated subdomains for the error tolerance $\varepsilon_{\text{tol}} = 10^{-8}$ are displayed in a logarithmic scale. We can see that the differences of the numbers of needed subdomains for the shown methods are small. Especially the FS IPM generated very similar results to the *hp* anchor point method with about the same offline and online complexity but without any knowledge of the parameters. For all refinement steps of the FS IPM, it was sufficient to use only $M_0 = 1$ coefficient for the assignment.

Figure 8 compares the partitioning result of the implicit partitioning methods with the *hp* results for a given $M_{\max} = 80$ and $\varepsilon_{\text{tol}} = 10^{-8}$. The *hp* gravity center method is least flexible since only partitions with $J = 4, 16$ and 64 subdomains could be obtained. Hence, it generates more than necessary subdomains for most values $M_{\max}$. Furthermore, the determination of the gravity center is based upon

the total number of $M_{\max}$ parameters and may be less appropriate for $M^h_{\max} \ll M_{\max}$.

We tested the MS IPM online stage using a test sample set of 10,000 parameters and the fast assignment of Algorithm 6. For $J = 8$ subdomains and $\varepsilon_{\text{tol}} = 10^{-4}$, we used $M = 33$ and $M^+ = 36$ basis functions. Less than 1.39 assignment trials per sample have been needed on average. For a second example with a large number $J = 64$ of subdomains, $\varepsilon_{\text{tol}} = 10^{-8}$, $M = 40$, and $M^+ = 44$, the average number of assignment trials was still less than 2. Hence, the procedure is very efficient.

6.2 Example 2

We now consider the example that has been used in [1] to introduce the idea of the EIM and which has briefly been mentioned in Section **??**. For a spatial domain $\mathcal{D} = [0, 1]^2$ and the parameter domain $\mathcal{P} = [0.01, 1]^2$, the input functions $c : \mathcal{D} \times \mathcal{P} \to \mathbb{R}$ are defined by

$$c(x; \mu) = (x_1 + \mu_1)^2 + (x_2 + \mu_2)^2.$$

As for the Example 1, we use a uniform grid on $\mathcal{D}$ with edge length 0.02 and $\mathcal{N} = 2601$ degrees of freedom. The parameter domain $\mathcal{P}$ is sampled using a logarithmically distributed grid with 72 parameters in each direction leading to $n_{\text{tr}} = 5184$.

We already mentioned that the maximum of c is located at $x_{\max} = (1, 1)$ independently of μ. Since the first interpolation knot is located at the maximum of the first basis function, all subdomains select the same first knot and therefore, the first approximation coefficient $\theta^j_1(\mu)$ is identical for all subdomains j. Hence, the coefficient based FS IPM needs at least two coefficients for the assignments. On average over all performed runs and nodes, it selected about 2.8 coefficients which still leads to a very efficient online assignment. Especially in the lower parts of the subdomain trees, i.e., for very fine partitions, resets of the initial basis functions occurred in order to obtain appropriate partitions.

In Figure 9, we compare again the numbers of generated subdomains for different values of $M_{\max}$, where the numbers of subdomains are plotted logarithmically. For this example, the MS IPM clearly outperforms the other methods whereas on average, the FS IPM and the hp methods produced similar numbers of subdomains. For the hp gravity center method, only the very few numbers of $J = 4, 7, 16$ and 25 could be reached at all.

In Figure 10, we compare the partitions generated by the different methods for a desired number of $M_{\max} = 55$ basis functions. We observe that the shapes and numbers of subdomains differ significantly, where the MS IPM in Figure 10(a) seems to divide the parameter domain in the best way. It is also interesting to see that some of the subdomains of the FS IPM in Figure 10(b) are divided into several parts that are not connected. E.g., the "black subdomain" consists of parameters in the lower left and lower right part of the parameter domain. For the construction of this partition, two resets of the initial partition were necessary and the average number of coefficients for the assignment was 2.75.

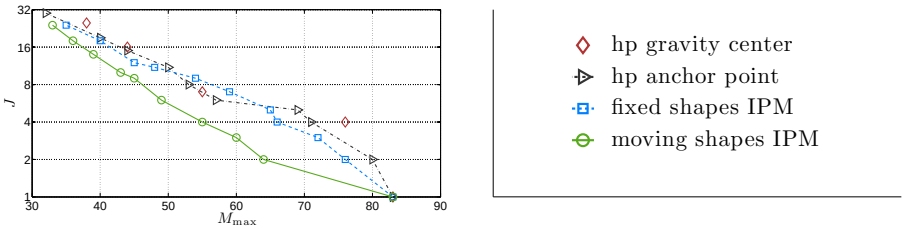
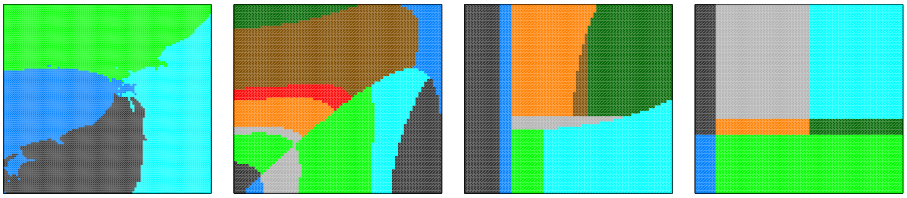


Fig. 9 Comparison: number of subdomains J necessary for a given maximal number of affine terms $M_{\max}$ for Example 2.



(a) MS IPM: $J = 4$ (b) FS IPM: $J = 9$ (c) hp anchor point: $J = 8$ (d) hp gravity center: $J = 7$

Fig. 10 Partitioning results for Ex. 2 and desired $M_{\max} = 55$ and $\varepsilon_{\text{tol}} = 10^{-8}$.

Example 3

In the last example of this section, we consider a special parameter dependency. For the spatial domain $\mathcal{D} = [0,1]^2$ and the explicitly given parameter domain $\mathcal{P} = [0,1]^2$, the input function $c : \mathcal{D} \times \mathcal{P} \to \mathbb{R}$ is given by

$$c(x;\mu) = e^{-50\left(\left(x_1 - 4\left(\mu_1 - \frac{1}{2}\right)^2 - \mu_2^2\right)^2 + x_2^2\right)}.$$

Now, for parameters $\mu \in \mathcal{P}$ on the elliptic curves

$$4\left(\mu_1 - \tfrac{1}{2}\right)^2 + \mu_2^2 \equiv const,$$

the input functions $c(\mu)$ are identical. Hence, it is desirable that the partitioning methods detect this dependency and adjust the splitting of the subdomains accordingly.

For the discretization of the spatial domain, we used again a uniform grid with edge length 0.02 and obtain $\mathcal{N} = 2601$ degrees of freedom. The parameter samples for the offline stage are now selected using a uniform grid on $\mathcal{P} = [0,1]^2$ with 72 parameters in each direction. Hence, we obtain $n_{\text{tr}} = 5184$ uniformly distributed samples.

A single EIM on the complete subdomain converged for $M = 27$ basis functions. For $M_{\max} = 18$, the partitioning results of the MS IPM and the two *hp* methods are provided in Figure 12. For the MS IPM, two subdomains are sufficient. We provide the initial partition for $M = 1$ and the final partition for $M = 18$ in Figures 12(a) and 12(b), respectively. In Figure 12(a), we also marked the two parameters that have been used for the initial basis. It can directly be seen that

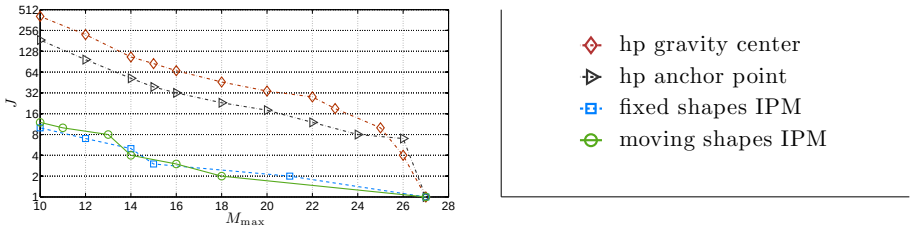
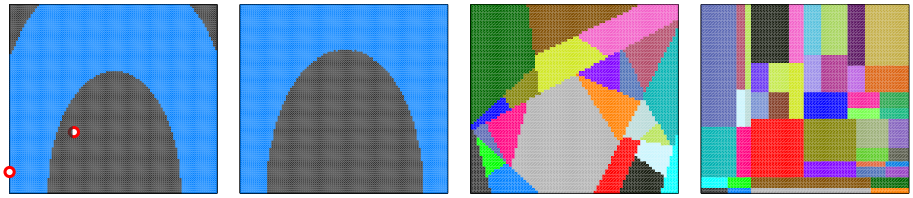
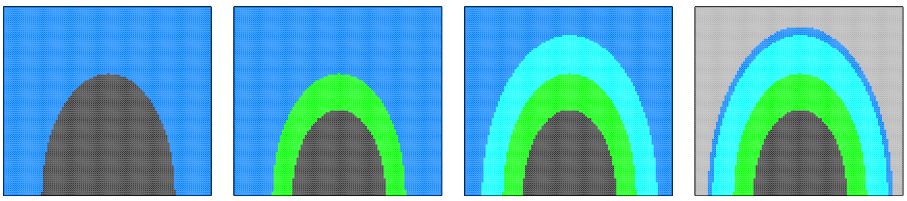


Fig. 11 Comparison: number of subdomains J necessary for a given maximal number of affine terms $M_{\max}$ for Example 3.



(a) MS IPM initial: $J = 2$ (b) MS IPM final: $J = 2$ (c) hp anchor point: $J = 23$ (d) hp gravity center: $J = 46$

Fig. 12 Partitioning results for Example 3 and desired $M_{\max} = 18$ and $\varepsilon_{\text{tol}} = 10^{-8}$.



(a) refinement 1 (b) refinement 2 (c) refinement 3 (d) refinement 4

Fig. 13 Tree structured refinement steps for Example 3 using the FS IPM.

the subdomains are defined by the band around the ellipse of parameters on which the initial parameters are located.

Contrarily, the *hp* results in Figures 12(c) and 12(d) do not properly detect the geometric parameter dependency of the input functions. Not even the symmetry along the axis $\mu_1 = \frac{1}{2}$ has been used. As a consequence, a huge number of subdomains is needed. At the same time, many subdomains cover the same part of the family of input functions. Especially for the *hp* anchor point method, one can see that many small subdomains have been created along one parameter ellipse.

Also in Figure 11, it can be observed that the *hp* methods do not detect the more special parameter dependency. On average, the anchor point method needs about 10 times more subdomains to appropriately cover the complexity of $\mathcal{M}$. The number of subdomains created by the gravity center splitting procedure is an additional factor of around two larger.

The number and shapes of the subdomains created for the MS IPM and the FS IPM differ only slightly. In Figure 13, the tree structure of the FS IPM is provided. In each step, one of the subdomains is divided into two parts. We see that shapes

of the subdomains keep their elliptic appearance. For the parameter assignments, one approximation coefficient was sufficient.

7 Conclusions

We developed implicit partitioning methods that assign arbitrary parametric input functions to an appropriate subdomain without the knowledge of the actual parameter or any other additional information. The subdomains are no longer defined using distance measures with respect to the parameter but can rather be seen as classes of similar functions. Hence, even non-parametric input functions can be processed. On each subdomain, an EIM is performed that creates affine approximations of the input functions with respect to the unknown parameter.

The methods automatically detect complex parametric structures such as symmetries or other patterns of the parametric dependency and the assignments are independent of the dimension of the parameter domain. Hence, for wide classes of problems, the implicit methods outperform other partitioning methods even for known explicitly given parameters.

References

1. Barrault, M., Maday, Y., Nguyen, N.C., Patera, A.T.: An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. C. R. Math. Acad. Sci. Paris **339**(9), 667–672 (2004)
2. Eftang, J.L., Knezevic, D.J., Patera, A.T.: An *hp* certified reduced basis method for parametrized parabolic partial differential equations. Math. Comput. Model. Dyn. Syst. **17**(4), 395–422 (2011)
3. Eftang, J.L., Patera, A.T., Rønquist, E.M.: An "*hp*" certified reduced basis method for parametrized elliptic partial differential equations. SIAM J. Sci. Comput. **32**(6), 3170–3200 (2010)
4. Eftang, J.L., Stamm, B.: Parameter multi-domain hp empirical interpolation. In Thesis: J. L. Eftang. Reduced Basis Methods for Parametrized Partial Differential Equations. (2011)
5. Grepl, M.A., Patera, A.T.: A posteriori error bounds for reduced-bias approximations of parametrized parabolic partial differential equations. M2AN Math. Model. Numer. Anal. **39**(1), 157–181 (2005)
6. Haasdonk, B., Dihlmann, M., Ohlberger, M.: A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space. Math. Comput. Model. Dyn. Syst. **17**(4), 423–442 (2011)
7. Haasdonk, B., Urban, K., Wieland, B.: Reduced basis methods for parametrized partial differential equations with stochastic influences using the Karhunen-Loève expansion. SIAM/ASA J. Uncertainty Quantification **1**, 79–105 (2013)
8. Patera, A.T., Rozza, G.: Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations Version 1.0, MIT, Cambridge, MA (2006)
9. Rozza, G., Huynh, D.B.P., Patera, A.T.: Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: Application to transport and continuum mechanics. Arch. Comput. Methods Eng. **15**(3), 229–275 (2008)
10. Tonn, T.: Reduced-basis method for non-affine elliptic parametrized PDEs (motivated by optimization in hydromechanics). Ph.D. thesis, Ulm University, Ulm, Germany (2012)
11. Veroy, K., Patera, A.T.: Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: rigorous reduced-basis a posteriori error bounds. Internat. J. Numer. Methods Fluids **47**(8-9), 773–788 (2005)
12. Wieland, B.: Reduced basis methods for partial differential equations with stochastic influences. Ph.D. thesis, Ulm University, Ulm, Germany (2013)
13. Yano, M., Patera, A.T., Urban, K.: A space-time certified reduced basis method for Burgers' equation. Preprint, Ulm University (2012).