Journal of Scientific Computing (2019) 80:1369–1394
https://doi.org/10.1007/s10915-019-00978-3

Solving Elliptic PDE's on Domains with Curved Boundaries with an Immersed Penalized Boundary Method

Larry L. Schumaker[1]

Received: 21 December 2018 / Revised: 12 May 2019 / Accepted: 16 May 2019 / Published online: 24 May 2019

Abstract
The purpose of this paper is to describe a method (which we call IPBM) for solving boundary value problems on domains with curved boundaries. The method combines two ideas from the PDE literature: (a) the idea of immersing the problem in a larger and simpler domain, and (b) the idea of enforcing boundary conditions by using a penalty term. The method has a number of advantages as compared to existing methods in the literature and can be considered as a viable alternative to the very popular isogeometric analysis methods. It can be used with a wide variety of spline spaces including tensor-product splines and splines on triangulations. It also works with splines on H-triangulations and splines on T-meshes, which opens the door to adaptive methods. The paper contains a series of examples both in 2D and 3D to illustrate the capability of the method to produce high order approximations.

Keywords Boundary-value problem · Curved domains · Immersed domain · Penalized boundary condtions · IPBM · Splines

1 Introduction

Suppose L is a linear elliptic partial differential operator defined on a domain Ω with boundary $\partial\Omega$. Given f defined on Ω, suppose there exists a unique u satisfying the partial differential equation $Lu = f$ on Ω along with certain boundary conditions defined at points on $\partial\Omega$. The problem of numerically computing approximate solutions to such problems has been heavily studied for many years. Recently a lot of attention has been devoted to an approach called isogeometric analysis (IGA) which is based on using the same functions used in CAD systems to define the domain Ω, see e.g. [10] or one of the more recent papers such as [30]. In IGA practice these approximating functions are usually Nurbs defined on rectangular domains, so to apply the method the first step is to find a parametric mapping to Ω from the simple domain on which the Nurbs are defined. Since in currently available CAD systems designers usually produce a representation of the boundary $\partial\Omega$ only, finding this parametric mapping is a significant first step to using IGA.

✉ Larry L. Schumaker
larry.schumaker@vanderbilt.edu

[1] Department of Mathematics, Vanderbilt University, Nashville, USA

The purpose of this paper is to describe a method (see Sect. 2), which we call the *immersed penalized boundary method (IPBM)*, that may be a viable alternative to IGA. The method combines two ideas from the PDE literature (see Remarks 9.1–9.3): (a) the idea of immersing the problem in a larger and simpler domain D, and (b) the idea of enforcing boundary conditions by using a penalty term. The method has the following features:

- Unlike the IGA approach, the IPBM approach does not require a map from some computational domain (such as a rectangle) to the desired domain Ω.
- The method makes use of boundary conditions only at points on the boundary $\partial\Omega$, not at some nearby points.
- The method works with a variety of standard boundary conditions, including Dirichlet, Neumann, and mixed boundary conditions.
- The method works with standard spline spaces defined on triangulations or tetrahedral partitions, and also with tensor-product splines defined on rectangles or boxes. It also works with splines defined on partitions with hanging vertices such as H-triangulations and T-meshes. These are particularly convenient for use with adaptive methods.
- The method works with any given representation of the boundary that permits us to find a set of points on the boundary. In particular, we can use boundary representations defined by NURBs curves or surfaces as produced by standard CAD software. It is also possible to work with other simpler direct parametric representations of $\partial\Omega$ (such as lines, circular arcs, and conic sections). In fact, to apply the methods here we require only a cloud of (reasonably spaced) points on the boundary.
- The method can be used with a Galerkin type approach, or with least-squares collocation. In either case, we do not have to construct approximating functions that vanish on the boundary. For curved boundaries constructing such functions can be complicated.
- Unlike many immersion methods in the literature, there is no need to modify or stabilize our approximating functions in a neighborhood of $\partial\Omega$ or to refine the mesh near the boundary $\partial\Omega$, and no need to compute integrals along the boundary.

Our goal in this paper is restricted to illustrating the usefulness of immersed penalized boundary methods for solving elliptic boundary problems in two and three variables. A mathematical analysis of the methods is in progress.

2 The IPBM Approach in Two Variables

We first explain the IPBM approach in the following simple setting involving a second order PDE in two variables with Dirichlet boundary conditions. Let Ω be a bounded domain in $\mathbb{R}^2$ with boundary $\partial\Omega$. We allow Ω to have one or more holes, and assume that the boundary is piecewise smooth. Let a_1, a_2, a_3, f be functions defined on Ω, and let

$$Lu := a_1(x, y)u_{xx} + a_2(x, y)u_{xy} + a_3(x, y)u_{yy}. \tag{2.1}$$

Given a function g defined on $\partial\Omega$, suppose a_1, a_2, a_3, f are such that the following problem has a unique solution

$$Lu = f, \quad \text{on } \Omega, \tag{2.2}$$

$$u = g, \quad \text{on } \partial\Omega. \tag{2.3}$$

Suppose D is a polygonal domain D containing Ω, and that L is defined not only on Ω, but also on D. In the examples below we will take D to be the smallest rectangle containing

Ω. Let $\mathcal{S}$ be a space of approximating functions defined on D. For example $\mathcal{S}$ might be a space of polynomial splines on a triangulation of D, or a space of tensor-product splines. Suppose $\{\phi_j\}_{j=1}^N$ is a basis for $\mathcal{S}$. We are looking for a numerical solution of the form

$$s = \sum_{j=1}^{N} c_j \phi_j. \tag{2.4}$$

We now describe two approaches to finding the coefficient vector c, both of which reduce to solving a linear system of equations.

(A) **The IPBM/F method:** Following the standard Galerkin (FEM) approach, we require

$$\int_D Ls\,\phi_i = \int_D f\phi_i, \qquad i = 1, \ldots, N. \tag{2.5}$$

Note that we are integrating over D, not Ω, and that we will need to use approximating spaces for which Ls is in $L_2(D)$. Next, let $\{(\xi_i, \eta_i)\}_{i=1}^{n_b}$ be a set of points lying on the boundary $\partial\Omega$. Then given $\lambda > 0$, we consider the following *penalized least-squares problem*: Find c to minimize

$$\Phi(c) := \sum_{i=1}^{N} \left[\int_D (Ls - f)\phi_i \right]^2 + \lambda \sum_{i=1}^{n_b} [s(\xi_i, \eta_i) - g(\xi_i, \eta_i)]^2. \tag{2.6}$$

This problem can be reduced to a linear system of equations of the form $Gc = r$ with G a symmetric and nonnegative definite matrix. In general, G will actually be positive define, which ensures there is a unique solution vector c. We call the corresponding approximating function (2.4) an *IPBM/F solution* of the boundary-value problem.

(B) **The IPBM/C method:** Suppose that Γ is a finite set of points in D, and that $(\xi_i, \eta_i)\}_{i=1}^{n_b}$ are points on the boundary $\partial\Omega$. Then given a positive function w defined on D and $\lambda > 0$, we consider the following *penalized least-squares problem*: Find c to minimize

$$\Psi(c) := \sum_{\zeta \in \Gamma} w(\zeta)\,[Ls(\zeta) - f(\zeta)]^2 + \lambda \sum_{i=1}^{n_b} [s(\xi_i, \eta_i) - g(\xi_i, \eta_i)]^2. \tag{2.7}$$

As before this leads to a system of equations of the form $Gc = r$, where the matrix G is symmetric and nonnegative definite. In general, G will actually be positive definite, which ensures there is a unique solution vector c. We call the corresponding approximating function (2.4) an *IPBM/C solution* of the boundary-value problem. For a discussion of why we included a weight function w in this formulation and how to choose it in practice, see Remarks 9.16–9.17.

The rest of the paper is organized as follows. In Sect. 3 we illustrate the performance of IPBM/F. We give examples using a macro-element space, C^0 splines, and tensor-product splines on a variety of domains, some with holes. In the course of this discussion we address the issues of how to choose the points on the boundary and discuss the role of λ. The examples in Sect. 4 make use of the IPBM/C method, while problems with mixed boundary conditions are discussed in Sect. 5. The use of IPBM methods with splines on H-triangulations and T-meshes are treated in Sects. 6 and 7. The focus is on adaptive methods involving partitions with hanging vertices. Finally, in Sect. 8 we give two examples in 3D. All of the computations were done in Matlab using the spline package associated with my SIAM book [31].

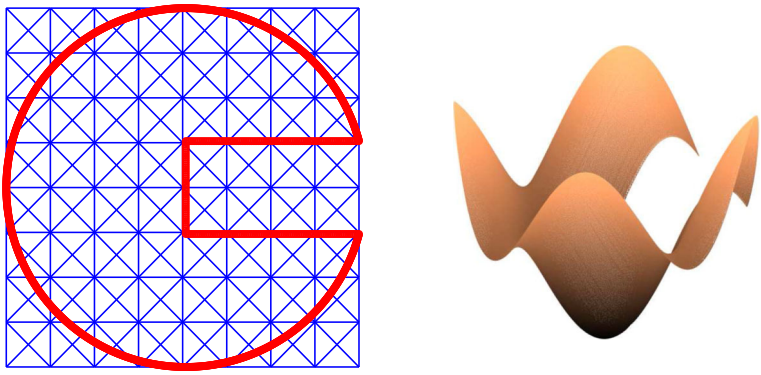

Fig. 1 A slotted disk immersed in a 9×9 type-2 triangulation of the unit square and the associated IPBM/F spline for Example 3.1

3 Examples of IPBM/F to Solve Poisson's Equation in 2D

In this section we solve the Poisson equation

$$\Delta u = u_{xx} + u_{yy} = f \tag{3.1}$$

on a curved domain Ω in $\mathbb{R}^2$ with Dirichlet boundary conditions. For each example we begin with a known true solution u, set $f = \Delta u$ and $g = u|_{\partial\Omega}$. Since we have the true solution, we can compute error estimates for our spline approximation s by comparing s with u on some large well-distributed set of points in Ω. We will report both max and RMS errors.

3.1 IPBM/F with $\mathcal{S}_5^{1,2}(\triangle)$

In this section we illustrate the use of IPBM methods using the Argyris macro-element space $\mathcal{S}_5^{1,2}(\triangle)$. It consists of piecewise polynomials of degree five that are C^1 globally, and that have C^2 supersmoothness at the vertices of $\triangle$, see Sect. 6.1 of [21] or Sect. 5.5 of [31]. Our first two examples explore the interplay between the size of λ and the number n_b of boundary points used for the Dirichlet boundary condition.

Example 3.1 Solve the Poisson problem with true solution $u(x, y) = \sin(10x) + \sin(10y)$ on the slotted disk Ω shown in Fig. 1 using IPBM/F with the Argyris space $\mathcal{S}_5^{1,2}(\triangle)$ on the type-2 triangulation of the unit square with 9 grid lines in each direction.

Discussion: The boundary of the domain Ω is defined by four simple parametric curves. Here is a table of the results based on 322 points on the boundary. The column labeled *size* gives the number of degrees of freedom (size of the linear system being solved), and the one labeled *cond* gives the estimated condition number of this system. To estimate the error on Ω, we pick a large number of nearly equally-spaced points in Ω. For this domain we used 33466 points for the error calculations. The associated max and RMS errors are given in the columns labelled emax and rms. The columns labelled emaxB and rmsB report the max and RMS errors measured at 3278 points on the boundary. The reported time is for assembling and solving the system of equations for the spline coefficients, measured in seconds. The table shows that we can get very good results with a wide range of lambdas, although using

λ	Time	Size	Cond	emaxB	rmsB	emax	rms
0.001	0.15	1270	1.26e+14	4.68e−05	9.65e−06	4.68e−05	4.04e−06
0.01	0.16	1270	1.11e+13	3.11e−05	5.87e−06	3.11e−05	2.39e−06
0.10	0.16	1270	5.35e+11	1.02e−05	2.16e−06	1.02e−05	1.18e−06
1	0.16	1270	9.96e+10	2.27e−06	7.36e−07	3.49e−06	9.80e−07
10	0.16	1270	5.11e+10	1.26e−06	3.86e−07	1.45e−05	5.05e−06
100	0.16	1270	1.10e+12	2.13e−06	2.02e−07	4.25e−05	1.22e−05
1000	0.16	1270	7.99e+13	3.80e−06	2.37e−07	1.75e−04	2.06e−05
10000	0.16	1270	7.03e+15	2.74e−05	1.12e−06	3.70e−04	5.31e−05

too large a value for λ leads to a large condition numbers and some loss of accuracy. The errors are minimal for the choice $\lambda = 1$. □

Example 3.2 Repeat Example 3.1 keeping $\lambda = 1$ but varying the number of equally spaced sample points used for the Dirichlet boundary conditions.

Discussion: The following table displays the same information as the table in the previous example, except that the first column now gives the number of boundary points n_b while the third column gives the spacing h_b of the points along the boundary.

n_b	h_b	Time	Cond	emaxB	rmsB	emax	rms
76	0.0400	0.16	1.71e+11	8.50e−06	1.44e−06	8.50e−06	1.39e−06
159	0.0200	0.16	1.20e+11	2.51e−06	8.00e−07	3.50e−06	9.98e−07
322	0.0100	0.22	9.96e+10	2.27e−06	7.36e−07	3.49e−06	9.80e−07
651	0.0050	0.23	8.47e+10	2.15e−06	6.88e−07	3.61e−06	1.01e−06
1307	0.0025	0.17	7.35e+10	2.03e−06	6.46e−07	3.78e−06	1.11e−06
2622	0.0013	0.22	6.54e+10	1.88e−06	6.01e−07	4.58e−06	1.34e−06
5249	0.0006	0.40	5.97e+10	1.70e−06	5.49e−07	5.94e−06	1.81e−06
10502	0.0003	0.52	5.57e+10	1.46e−06	4.87e−07	8.04e−06	2.60e−06
21009	0.0002	0.79	5.27e+10	1.33e−06	4.20e−07	1.11e−05	3.77e−06

The table shows that for this problem the choice of $h_b = .01$ which leads to 322 points on the boundary gives the smallest error, although there is only a small difference in accuracy over the entire range. Use of more points requires extra computational time and actually makes the errors slightly worse. □

These experiments shed some light on the interplay between the number of boundary points and the parameter λ. However, they do not suggest a simple prescription for how to select n_b and λ in general. Clearly, optimal choices of n_b and λ will depend on a variety of factors: (1) the mesh size of the partitions, (2) size and support of the basis functions, (3) complexity of the solution both inside and on the boundary Ω, and (4) the nature of the boundary. For the remaining examples in this section we have generally taken the spacing of the boundary points to be $h_b = .01$, which gave values of n_b between 300 and 800. We also do all of the examples using $\lambda = 1$.

We now give an example to explore the rate of convergence using the Argyris space on a nested sequence of type-2 triangulations of the unit square.

Example 3.3 Solve the Poisson problem on the domain shown in Fig. 1 (left) with true solution $u(x, y) = \sin(10x) + \sin(10y)$ using the Argyris space $\mathcal{S}_5^{1,2}(\triangle)$ on type-2 triangulations of the unit square.

Discussion: We work with a nested sequence of type-2 partitions with m grid lines in each direction. Since the mesh sizes decrease by a factor of two for each refinement, we can compute the log base 2 of the ratio of two successive errors to estimate the rate of convergence. The results are shown in the supplementary table. We do not attempt to find the optimal values of n_b and λ for each mesh, but instead use $n_b = 322$ and $\lambda = 1$ for all meshes.

m	Time	Size	Cond	emax	rms	Rates	
3	0.00	106	8.73e+06	7.91e−03	1.80e−03		
5	0.05	350	4.05e+08	2.99e−04	8.29e−05	4.72	4.44
9	0.12	1270	9.49e+10	3.43e−06	9.73e−07	6.45	6.41
17	1.85	4838	7.69e+13	6.73e−08	1.23e−08	5.67	6.31

The table shows that the method is exhibiting order 6 convergence, which is optimal order convergence for quintic splines. Figure 1 (right) shows a plot of the corresponding spline solution for $m = 9$. It is visually indistinguishable from the true solution. □

Example 3.4 Solve the Poisson problem on the domain Ω shown in Fig. 2 (left) with true solution $u(x, y) = \sin(10x) + \sin(10y)$ using the Argyris space $\mathcal{S}_5^{1,2}(\triangle)$ on type-2 triangulations.

Discussion: This domain is defined by a NURBS curve of degree 2 with 12 control points in $\mathbb{R}^2$. We imbed the domain in type-2 triangulations with m grid lines in both the x and y variable. Figure 2 (left) shows the case $m = 5$. Here is the corresponding table of errors for a nested sequence of triangulations, calculated on 33261 nearly equally spaced points in Ω. Here $n_b = 394$ and $\lambda = 1$.

m	Time	Size	Cond	emax	rms	Rates	
3	0.00	106	1.31e+07	4.08e−02	9.65e−03		
5	0.01	350	9.07e+08	1.10e−03	2.72e−04	5.21	5.15
9	0.13	1270	2.00e+11	1.70e−05	3.69e−06	6.02	6.21
17	1.81	4838	1.95e+14	2.75e−07	5.35e−08	5.95	6.11

This table shows that the method is providing order six convergence, which is optimal for splines of degree five. Figure 2 (right) shows a plot of the Argyris spline surface for this triangulation. □

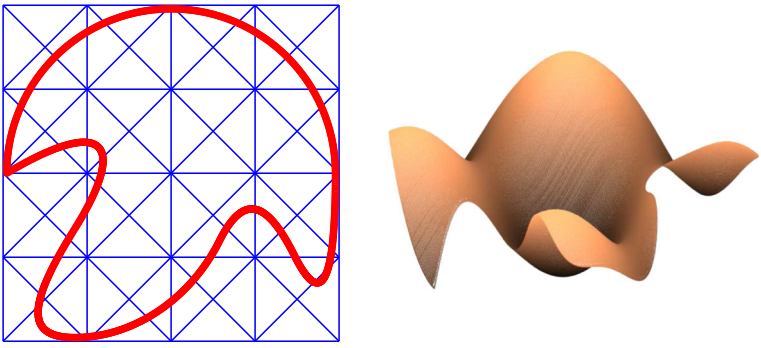

Fig. 2 A domain defined by a Nurbs curve immersed in a 5×5 type-2 triangulation and the associated IPBM/F quintic spline for Example 3.4

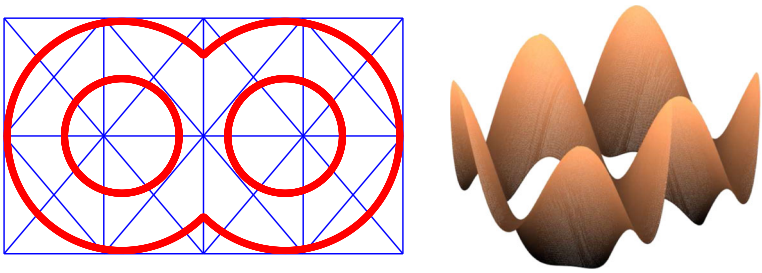

Fig. 3 A domain with two holes immersed in a 5×3 type-2 triangulation and the associated IPBM/F spline for Example 3.5

We now give an example of a domain with two holes.

Example 3.5 Solve the Poisson problem on the domain Ω shown in Fig. 3 (left) with true solution $u(x, y) = \sin(10x) + \sin(10y)$ using the Argyris space $\mathcal{S}_5^{1,2}(\triangle)$.

Discussion: This domain is defined directly via four simple parametric curves in $\mathbb{R}^2$. Figure 3 (left) shows the domain imbedded in a type-2 triangulation with $m = 5$ and $n = 3$ grid lines in the x and y variables, respectively. The Argyris spline surface corresponding to this triangulation is shown in Fig. 3 (right). For this domain we are using 782 points on the boundary with $\lambda = 1$. The errors are calculated on a set of 42930 points. Here is the corresponding table of errors for a nested sequence of type-2 triangulations.

m	n	Time	Size	Cond	emax	rms	Rates	
5	3	0.02	192	1.54e+07	1.05e−02	1.72e−03		
9	5	0.05	666	6.03e+08	2.89e−04	6.15e−05	5.18	4.80
17	9	0.53	2478	2.99e+11	3.83e−06	7.73e−07	6.24	6.31
33	17	8.00	9558	3.80e+14	5.07e−08	1.08e−08	6.24	6.16

This table shows that the method is providing order six convergence, which is optimal for splines of degree five. □

To conclude this subsection we give one example with a more complicated true solution and a more complicated domain.

Example 3.6 Solve the Poisson problem on the domain shown in Fig. 4 (left) with true solution $u(x, y) = \sin(x^2+y^2)+0.1\sin(25(x^2+y^2)$ using the Argyris space on type-2 triangulations.

Discussion: This domain is defined by a quadratic NURBs curve with seven control points in $\mathbb{R}^2$. Here we use 616 points on the boundary with $\lambda = 1$. The number of points used to estimate the errors was 26,615. The $\mathcal{S}_5^{1,2}(\triangle)$ spline surface corresponding to the triangulation in Fig. 4 (left) is shown in Fig. 4 (right).

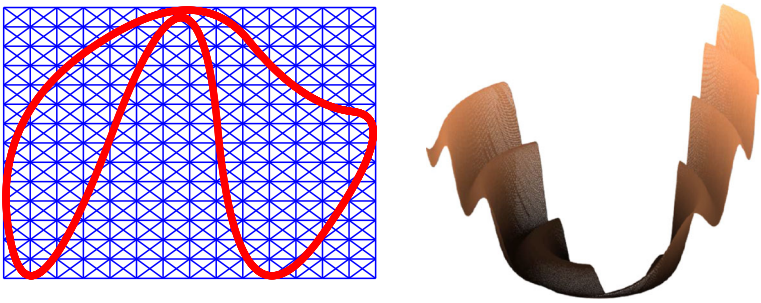

Fig. 4 A domain immersed in a 15×15 type-2 triangulation and the associated IPBM/F spline for Example 3.6

m	Time	Size	Cond	emax	rms	Rates	
5	0.03	350	4.42e+09	2.06e−01	4.16e−02		
9	0.16	1270	3.23e+12	7.08e−02	7.49e−03	1.54	2.47
17	2.12	4838	2.57e+15	3.10e−03	2.83e−04	4.51	4.73
33	27.68	18886	2.30e+18	6.91e−05	5.93e−06	5.49	5.58

The rate of convergence is somewhere between five and six. □

3.2 IPBM/F with the Space $\mathcal{S}_d^0(\triangle)$

As discussed in Sect. 2, to work with the space $\mathcal{S}_d^0(\triangle)$ we need to add a penalty term of the form $\lambda_s c^T E^T E c$ where E is a matrix such that $s \in C^1(D)$ if and only if $Ec = 0$. For the problems examined here, the choice of λ_s does not seem to be very critical—in our experiments we take $\lambda_s = 1$. Other values may give small improvements in accuracy.

Example 3.7 Solve the Poisson problem on the domain shown in Fig. 1 (left) with true solution $u(x, y) = \sin(10x) + \sin(10y)$, but with the space $\mathcal{S}_5^{1,2}(\triangle)$ replaced by $\mathcal{S}_5^0(\triangle)$.

Discussion: We present results for a nested sequence of type-2 triangulations of the unit square with m grid lines in both the vertical and horizontal directions, respectively. Figure 1 (left) shows the domain imbedded in a type-2 triangulation with $m = 5$. Here is the corresponding table of errors based on using 322 points on the boundary. The spline surfaces are visually indistinguishable from the one shown in Fig. 1 (right).

m	Time	Size	Cond	emax	rms	Rates	
3	0.02	221	7.03e+05	7.97e−03	1.89e−03		
5	0.08	841	3.21e+06	1.62e−04	4.17e−05	5.62	5.50
9	0.37	3281	8.57e+08	3.33e−06	9.10e−07	5.60	5.52
17	3.82	12961	1.08e+11	7.18e−08	2.10e−08	5.53	5.43

Comparing with the table shown in Example 3.3, we see that using the C^0 quintic spline space instead of the superspline space $\mathcal{S}_5^{1,2}(\triangle)$ leads to comparable errors, although the convergence rate is a bit lower. Condition numbers are somewhat lower, but the linear systems are larger and the computational times are somewhat larger. □

We now present an example to show the effect of increasing the degree of the splines while holding the triangulation fixed.

Example 3.8 Solve the Poisson problem with true solution $u(x, y) = \sin(10x) + \sin(10y)$ on the domain shown in Fig. 2 (left) using the space $\mathcal{S}_d^0(\triangle)$ for different values of d.

Discussion: For this example we use the NURBS domain shown in Fig. 2 (left) imbedded in a type-2 triangulation with $m = n = 5$. Here is the corresponding table of errors for a sequence of values of d calculated on 33405 points in Ω. The table also includes the max and RMS errors computed on the 3942 points on the boundary. As we see from the table, the size of the system and the time for finding a solution increases somewhat as d increases, but the accuracy improves very significantly even though the condition numbers are also rising. □

d	Time	Size	Cond	emaxB	rmsB	emax	rms
3	0.01	313	4.21e+04	1.73e−02	5.73e−03	5.21e−02	1.65e−02
4	0.04	545	6.52e+05	2.28e−03	7.28e−04	1.16e−02	3.39e−03
5	0.03	841	1.09e+07	1.32e−04	3.87e−05	4.52e−04	1.22e−04
6	0.13	1201	3.70e+08	1.98e−05	3.58e−06	7.01e−05	1.61e−05
7	0.27	1625	5.90e+09	7.65e−07	1.40e−07	3.14e−06	6.19e−07
8	0.42	2113	1.78e+11	1.55e−07	1.22e−08	3.25e−07	7.03e−08
9	0.64	2665	4.12e+12	2.98e−09	3.28e−10	1.32e−08	2.56e−09

3.3 IPBM/F with Tensor-Product Splines

In this subsection we examine the performance of IPBM/F when used with tensor-product spline spaces defined on equally-spaced grids covering the enclosing rectangle.

Example 3.9 Solve the Poisson problem on the domain shown in Fig. 2 (left) with true solution $u(x, y) = \sin(10x) + \sin(10y)$ using tensor-product splines of bidegree 5×5 on an $m \times m$ grid covering the enclosing rectangle.

Discussion: Here is the corresponding table of errors for a nested sequence of tensor partitions, using 394 points on the boundary and $\lambda = 1$.

m	Time	Size	Cond	emax	rms	Rates	
5	0.06	81	3.26e+07	6.19e−02	1.74e−02		
9	0.12	169	2.79e+08	8.31e−04	2.68e−04	6.22	6.02
17	0.44	441	5.76e+10	5.75e−06	1.93e−06	7.17	7.12
33	2.75	1369	1.64e+13	7.84e−08	2.36e−08	6.20	6.35

This table shows that the method is providing order six convergence, which is optimal order for biquintic splines. As compared with polynomial splines on type-2 triangulations, we see that tensor-splines provide higher accuracy with far fewer degrees of freedom and in less computational time. □

We now explore how accuracy improves with tensor-product splines as we increase their degrees.

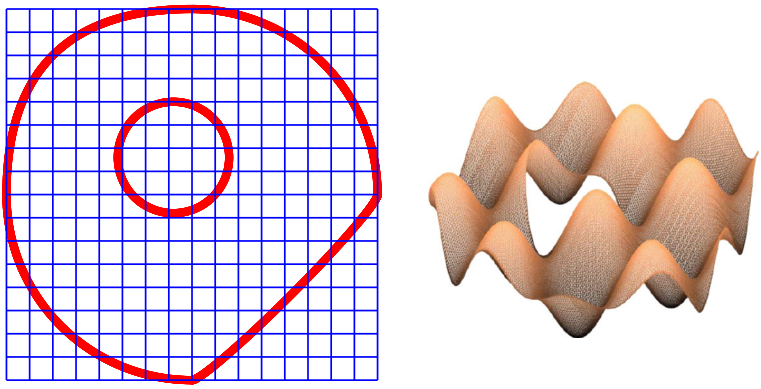

Fig. 5 A NURBS domain with a hole immersed in a 17×17 tensor grid and the corresponding biquintic tensor-product spline for Example 3.11

Example 3.10 Solve the Poisson problem with true solution $u(x, y) = \sin(10x) + \sin(10y)$ on the domain shown in Fig. 2 (left) using tensor-product splines of bidegree $d \times d$ on a 9×9 grid covering the enclosing rectangle.

Discussion: We use 394 points on the boundary and take $\lambda = 1$. Here is the corresponding table of errors for a sequence of increasing values of d, measured on 33405 points in Ω. The columns labelled emaxB and rmsB report the max and RMS errors measured at 3942 points on the boundary. The relative size of the errors on the boundary as compared to the size of the errors at points inside Ω can help in selecting λ to balance these errors.

d	Time	Size	Cond	emaxB	rmsB	emax	rms
3	0.03	121	1.38e+06	7.43e−03	2.87e−03	1.20e−02	4.11e−03
4	0.07	144	4.06e+07	1.82e−03	6.59e−04	3.18e−03	9.73e−04
5	0.14	169	2.79e+08	7.07e−04	1.74e−04	8.31e−04	2.68e−04
6	0.12	196	1.58e+09	1.70e−04	4.06e−05	2.53e−04	7.10e−05
7	0.15	225	1.44e+10	3.57e−05	1.02e−05	4.75e−05	1.53e−05
8	0.22	256	5.84e+11	7.31e−06	2.36e−06	2.17e−05	5.26e−06
9	0.27	289	2.05e+13	2.03e−06	6.03e−07	5.89e−06	1.31e−06

As we can see the accuracy improves as we increase d, although the condition numbers increase. Times are almost the same for all d since the numbers of degrees of freedom increases very slowly. □

3.4 Examples with Variable Coefficients

In this section we give three examples involving second order elliptic PDE's with nonconstant coefficients.

Example 3.11 Solve the boundary value problem (2.2)–(2.3) with $a_1(x, y) = 2 - 0.1y$, $a_2(x, y) = 0.1x$, and $a_3(x, y) = 1 + 0.1y$, with right-hand side f chosen to give a true solution of $u(x, y) = \sin(20x) + xy + \sin(20y)$. Use the domain with a hole shown in Fig. 5 (left), and use IPBM/F with biquintic tensor-product splines.

Discussion: Here is the corresponding table of errors for a nested sequence of $m \times m$ grids on the unit square. This table again shows order of approximation six which is optimal for tensor-product splines of bidegree 5×5. We show a plot of the tensor-product spline surface corresponding to $m = 17$ in Fig. 5 (right). □

m	Time	Size	Cond	emax	rms	Rates	
9	0.23	169	2.49e+10	3.08e−01	8.44e−02		
17	0.67	441	1.34e+13	8.26e−04	2.43e−04	8.54	8.44
33	1.77	1369	3.33e+15	6.13e−06	1.92e−06	7.07	6.98
65	20.87	4761	4.85e+17	6.48e−08	2.33e−08	6.56	6.37

Example 3.12 Solve the boundary value problem (2.2)–(2.3) with $a_1(x, y) = 0.01$, $a_2(x, y) = 0$, and $a_3(x, y) = 1$ on the domain shown in Fig. 2 (left). Choose f to give the true solution $u(x, y) = \sin(25x) + \sin(10y)$. Use IPBM/F with biquintic tensor-product splines on an $m \times m$ grid.

Discussion: We use tensor-product splines of bidegree 5×5 defined on the minimal enclosing rectangle for a nested sequence of tensor partitions. Here is the corresponding table of errors.

m	Time	Size	Cond	emax	rms	Rates	
9	0.11	169	2.38e+08	1.29e+00	4.51e−01		
17	0.47	441	1.77e+08	3.57e−03	1.00e−03	8.49	8.81
33	2.66	1369	1.38e+10	3.27e−05	6.45e−06	6.77	7.28
65	19.07	4761	1.82e+13	2.29e−07	6.68e−08	7.16	6.59

This table again shows order of approximation six which is optimal for tensor-product splines of bidegree 5×5. □

This example involves anisotropy—the coefficient a_1 is much smaller than the coefficient a_3 and the solution behaves much differently in the x variable than in the y variable. When we are aware of anisotropy, we can use different mesh sizes in the two directions, or different degrees in the two variables for the tensor-product splines (Fig. 6).

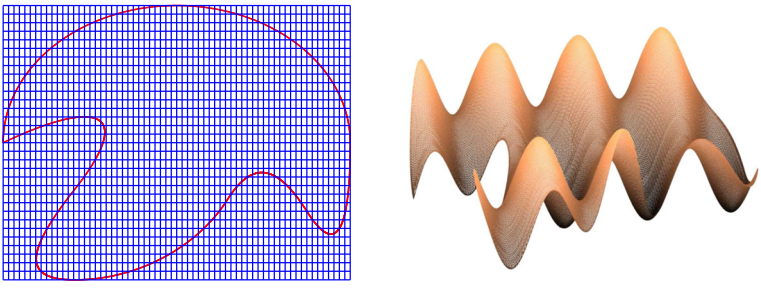

Fig. 6 A domain immersed in a 65×33 grid and the associated IPBM/F tensor-product spline for Example 3.13

Example 3.13 Repeat Example 3.12 using different degrees or different mesh sizes in the two variables.

Discussion: Here are two different ways to take account of the anisotropy.

d	$\tilde{d}$	m	n	Time	Size	Cond	emax	rms
5	5	65	33	6.96	2553	3.08e+12	2.08e−07	6.83e−08
9	5	33	33	4.80	1517	2.99e+13	9.97e−08	1.75e−08

This example shows that by taking advantage of the anisotropy, we can get the same accuracy as obtained in the last line of the table in the previous example, but with fewer degrees of freedom and faster computational times. The spline is shown in Fig. 6 (right). □

4 Examples of IPBM/C to Solve Poisson's Equation in 2D

In this section we repeat a few of the examples of Sect. 3 using IPBM/C instead of IPBM/F. For examples involving splines of degree d on triangulations we choose $\binom{d+2}{2}$ equally spaced collocation points in each triangle T, the so-called *domain points* associated with T. For rectangular partitions, for each rectangle H, we choose the collocation points to lie on an equally spaced $d \times d$ rectangular grid in H. For rectangles, it is also possible to choose points associated with tensor-product Gauss quadrature. We get similar results using fewer collocation points per triangle or rectangle, see Remark 9.18.

4.1 IPBM/C with the Spline Spaces $\mathcal{S}_5^{1,2}(\Delta)$ on Type-2 Triangulations

Example 4.1 Solve the Poisson problem on the domain Ω shown in Fig. 1 (left). with true solution $u(x, y) = \sin(10x) + \sin(10y)$ using IPBM/C with the Argyris space $\mathcal{S}_5^{1,2}(\Delta)$.

Discussion: The domain is the slit disk shown in Fig. 1 (left) and is imbedded in the unit square. We use type-2 triangulations with m grid lines in each of the variables x and y. This problem was solved with the same space using IPBM/F in Example 3.3. Here is the corresponding table of errors for a nested sequence of type-2 triangulations, where we use 324 points on the boundary and $\lambda = 1$. This table sugggests that the order of convergence is somewhere between five and six. The errors reported here are somewhat larger than those given in Example 3.3 for the IPBM/F method. The times of computation are about the same, but the condition numbers here are somewhat smaller. □

m	Time	Size	Cond	emax	rms	Rates	
3	0.01	106	1.90e+06	4.24e−02	8.36e−03		
5	0.03	350	2.95e+07	8.13e−04	2.60e−04	5.71	5.00
9	0.20	1270	1.67e+09	1.73e−05	4.02e−06	5.55	6.02
17	2.14	4838	1.77e+11	2.92e−07	8.10e−08	5.89	5.63

Example 4.2 Solve the Poisson problem of Example 3.5 with the space $\mathcal{S}_5^{1,2}(\triangle)$ but replacing IPBM/F with IPBM/C.

Discussion: The domain is shown in Fig. 3 (left) and has two holes. We use 782 boundary points and $\lambda = 1$. The following table gives results for the same nested sequence of uniform type-2 triangulations used in Example 3.5.

m	n	Time	Size	Cond	emax	rms	Rates	
5	3	0.01	192	2.65e+06	2.76e−02	6.97e−03		
9	5	0.09	666	6.56e+07	7.40e−04	2.27e−04	5.22	4.94
17	9	0.75	2478	4.33e+09	1.28e−05	3.09e−06	5.85	6.20
33	17	7.96	9558	5.12e+11	2.18e−07	6.59e−08	5.88	5.55

This table should be compared with the one in Example 3.5. Computational times are about the same, but the condition numbers are smaller while the errors are slightly larger than in the FEM case. The rate of convergence is somewhere between five and six. □

4.2 IPBM/C with C^0 Spline Spaces

As discussed in Sect. 2, to make IPBM/C work with C^0 spline spaces we will have to modify the objective function in (2.7) by adding a penalty term to make the spline (approximately) C^1. As before, given a positive number λ_s, we add the term $\lambda_s c^T E^T E c$, where E is the matrix such that $s \in C^1(D)$ if and only if $Ec = 0$.

Example 4.3 Solve the Poisson equation on the domain used in Example 3.3 using IPBM/C with the space $\mathcal{S}_5^0(\triangle)$ on type-2 triangulations of the unit square.

Discussion: The domain is the slit disk shown in Fig. 1 (left). For this experiment we take $\lambda = \lambda_s = 1$. Here is a table of the results for the same nested sequence of type-2 partitions as in Example 3.7 where IPBM/F was used.

m	Time	Size	Cond	emax	rms	Rates	
3	0.01	221	1.58e+05	2.03e−02	4.15e−03		
5	0.03	841	2.37e+06	4.98e−04	1.33e−04	5.35	4.96
9	0.12	3281	1.70e+08	1.04e−05	2.75e−06	5.57	5.60
17	2.56	12961	4.95e+10	2.33e−07	5.99e−08	5.49	5.52

The results here should be compared with those given in Example 3.3 where the IPBM/F method was used. The condition numbers here are about the same, and the errors here are only slightly larger. Also, we may be getting a little less than the optimal convergence rate of six. □

4.3 IPBM/C with Tensor-Product Splines

In this section we give two examples of IPBM/C based on tensor-product splines on a equally spaced grid covering the enclosing rectangle.

Example 4.4 Solve the Poisson problem on the domain shown in Fig. 2 (left) with true solution $u(x, y) = \sin(10x) + \sin(10y)$ using IPBM/C with the space of biquintic tensor-product splines.

Discussion: We use a nested sequence of partitions to allow us to estimate the rate of convergence. We use 394 points on the boundary with $\lambda = 1$.

m	Time	Size	Cond	emax	rms	Rates	
5	0.04	81	9.68e+05	2.99e−01	4.49e−02		
9	0.14	169	1.63e+07	6.86e−03	1.16e−03	5.44	5.27
17	0.56	441	7.39e+08	6.32e−05	7.51e−06	6.76	7.27
33	2.35	1369	1.04e+11	1.05e−06	1.12e−07	5.91	6.07

This table should be compared with the one in Example 3.9 where the same spline spaces were used with the IPBM/F method. The condition numbers here are lower than for the IPBM/F method, although the accuracy is not quite as good. The method seems to be order six, which is what we expect for quintic splines. □

Example 4.5 Repeat Example 4.4 using tensor-product splines of bidegree 6×6.

Discussion: The domain is shown in Fig. 2. Here is the corresponding table of errors.

m	Time	Size	Cond	emax	rms	Rates	
5	0.03	100	1.20e+07	7.15e−02	1.92e−02		
9	0.14	196	1.72e+08	1.25e−03	1.78e−04	5.84	6.75
17	0.69	484	8.84e+09	5.49e−06	5.85e−07	7.83	8.25
33	2.73	1444	2.67e+12	6.97e−08	6.29e−09	6.30	6.54

The errors with $d = \tilde{d} = 6$ are much better than with $d = \tilde{d} = 5$. This example does not clearly show the convergence order, but it is clearly between six and seven. □

We conclude this section with one example with variable coefficients.

Example 4.6 Solve the boundary value problem (2.2)–(2.3) on the domain with a hole shown in Fig. 5 (left) with $a_1(x, y) = 2 - 0.1y$, $a_2(x, y) = 0.1x$, and $a_3(x, y) = 1 + 0.1y$, with right-hand side f chosen to give a true solution of $u(x, y) = \sin(20x) + xy + \sin(20y)$. Use IPBM/C with biquintic tensor-product splines on equally-spaced $m \times m$ grids.

Discussion: This problem was solved with the IPBM/F method in Example 3.11. Here is the corresponding table of errors for IPBM/C.

m	Time	Size	Cond	emax	rms	Rates	
9	0.27	169	5.29e+07	4.55e−01	1.04e−01		
17	1.06	441	4.08e+09	3.00e−03	5.15e−04	7.25	7.66
33	2.65	1369	1.02e+12	5.28e−05	6.08e−06	5.83	6.41
65	12.29	4761	3.85e+14	1.03e−06	9.56e−08	5.68	5.99

Comparing this table with the one in Example 3.11 which were computed with IPBM/F shows that the errors are slightly larger, although the condition numbers are quite a bit smaller. We are getting an optimal order rate of convergence of six. □

5 IPBM with Mixed Boundary Conditions

Given a domain Ω, suppose its boundary is split into disjoint sets $\partial\Omega_D$ and $\partial\Omega_N$ such that $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$. Now suppose we are given functions g_d and g_n defined on $\partial\Omega_D$ and $\partial\Omega_N$, respectively. For each point in $\partial\Omega_N$, let D_n denote the unit outward-pointing normal derivative to the boundary at that point. Our problem is to find a function u such that

$$Lu = f, \text{ on } \Omega \tag{5.1}$$

$$u = g_d, \text{ on } \partial\Omega_D, \tag{5.2}$$

$$D_n u = g_n, \text{ on } \partial\Omega_N. \tag{5.3}$$

Both IPBM/F and IPBM/C can be used for problems with mixed boundary conditions. We need only modify the second term in (2.6) or (2.7). Given a set of collocation points on the boundary of Ω, let $\{(\xi_i, \eta_i)\}_{i=1}^{n_D}$ be those lying in $\partial\Omega_D$ and $\{(\tilde{\xi}_i, \tilde{\eta}_i)\}_{i=1}^{n_N}$ those lying in $\partial\Omega_N$. Given positive weights λ_D and λ_N, we now replace the second term in (2.6) or (2.7) by

$$\lambda_D \sum_{i=1}^{n_D} [s(\xi_i, \eta_i) - g_d(\xi_i, \eta_i)]^2 + \lambda_N \sum_{i=1}^{n_N} [D_n s(\tilde{\xi}_i, \tilde{\eta}_i) - g_n(\tilde{\xi}_i, \tilde{\eta}_i)]^2. \tag{5.4}$$

Assuming the approximating spline s is written in terms of a basis as in (2.4), we can compute the corresponding coefficient vector c by solving a linear system of equations of the form $Gc = r$ with G a symmetric positive-definite matrix. To set up this system, we will need not only the locations of each of the chosen points on $\partial\Omega_N$, but also the unit outward-pointing normal vector at those points. For simple collections of parametric curves, these can easily be computed exactly. For NURBS curves, they can be computed from tangent vectors to the curve. In this section we discuss only the case of least-squares collocation with polynomial splines on triangulations of the enclosing rectangle.

Example 5.1 Consider the Poisson problem on the disk Ω of radius $1/2$ centered at the origin with true solution $u(x, y) = \sin(10x) + \sin(10y)$. We use mixed boundary conditions taken from u where $\partial\Omega_D$ is the upper half of the circle, and $\partial\Omega_N$ is the lower half. Use the Argyris space $\mathcal{S}_5^{1,2}(\triangle)$ defined on type-2 triangulations of the enclosing rectangle.

m	Time	Size	Cond	emax	rms	Rates	
3	0.02	106	4.05e+06	2.26e−01	1.17e−01		
5	0.05	350	3.93e+07	7.69e−03	3.25e−03	4.88	5.17
9	0.23	1270	1.37e+09	1.66e−04	6.43e−05	5.53	5.66
17	2.35	4838	2.40e+11	3.00e−06	9.74e−07	5.79	6.04

Discussion: Here is a table of results for a nested sequence of type-2 triangulations with m grid lines, where we have used 627 points on the boundary with $\lambda_D = \lambda_N = 1$. The errors are computed on 42485 points in the disk. For comparision purposes, we also present the analogous table in the case where we enforce Dirichlet boundary conditions on the entire boundary of the disk. Using mixed boundary conditions runs a bit slower and has somewhat larger condition numbers. There is only a slight loss in accuracy. □

m	Time	Size	Cond	emax	rms	Rates	
3	0.01	106	1.15e+06	1.33e−01	3.69e−02		
5	0.03	350	1.62e+07	4.35e−03	1.36e−03	4.93	4.76
9	0.20	1270	7.99e+08	8.02e−05	2.22e−05	5.76	5.93
17	1.89	4838	8.70e+10	2.21e−06	6.49e−07	5.18	5.10

6 IPBM with Splines on H-triangulations

A collection of triangles whose union covers a domain in such a way that any two triangles intersect only at points on their edges is called an *H-triangulation*. They are more general than ordinary triangulations since they allow so-called *hanging vertices*, i.e. vertices that lie in the interior of an edge of another triangle. It is straightforward to define spaces of splines on H-triangulations in the same way as for ordinary triangulations. For a detailed study of properties of such spline spaces, see [33]. For examples of H-triangulations, see Fig. 8. They were produced by the adaptive methods in Examples 6.1 and 6.2.

In this section we explore how IPBM/F and IPBM/C work with C^0 splines on H-triangulations. We represent our splines in terms of Bernstein basis polynomials on each triangle as discussed in [33]. It is easy to derive conditions for such splines to be in C^1. In particular, there exists a matrix E such that s is C^1 if and only if $Ec = 0$. To use splines on H-triangulations with either IPBM/F or IPBM/C, we need to add a penalty term of the form $\lambda_s c^T E^T E c$. We focus on adaptive methods based on splitting triangles into four similar pieces, guided by the size of the residuals as measured by approximations to $\|Ls - f\|_{L_1(T)}$ for all $T \in \triangle$. To improve efficiency, we refine those triangles whose residuals are in the top 30% of all residuals, cf. e.g. [24]. We are going to solve the Poisson equation with Dirichlet boundary conditions taken from a known solution u. For test purposes we take our domain to be the disk Ω inscribed in the unit square and take the Dirichlet boundary conditions from the following true solutions:

$$u(x, y) = e^{-200((x-.5)^2+(y-.5)^2)}, \tag{6.1}$$

and

$$u(x, y) = \tanh(40y - 80x^2) - \tanh(40x - 80y^2). \tag{6.2}$$

The first of these is a sharp peak Fig. 7 (left), and the second is an example used in [24] to study adaptive FEM methods on H-triangulations of the unit square. It has very large derivatives along two curves, see Fig. 7 (right) and also Fig. 9.

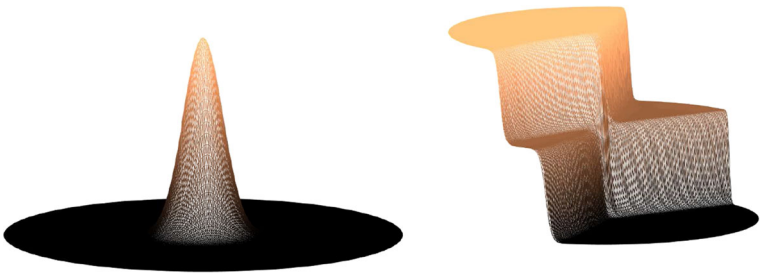

Fig. 7 Two functions for use in testing adaptive methods

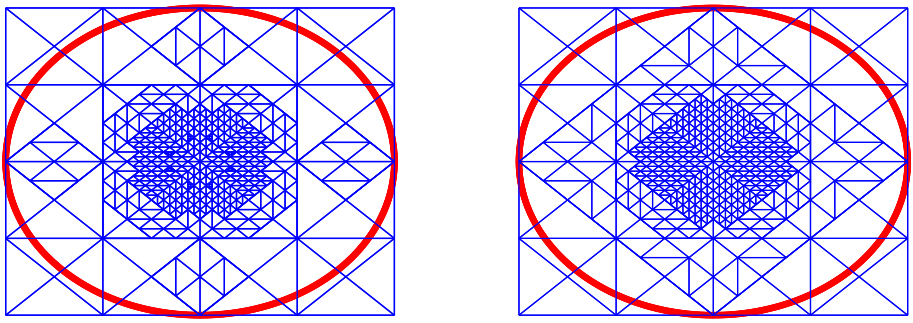

Fig. 8 H-triangulations obtained by adaptive IPBM/F and IPBM/C for Example 6.1

Example 6.1 Solve the Poisson problem on the disk Ω of radius $1/2$ centered at the point $(.5, .5)$ with true solution (6.1) using C^0 quintic splines on adaptively refined H-triangulations.

Discussion: We solve the problem using four methods: (a) adaptive IPBM/F with C^0 quintic splines starting with a 5×5 type-2 triangulation and performing 10 cycles, (b) adaptive IPBM/C with C^0 quintic splines starting with a 5×5 type-2 triangulation of the unit square with and using 11 cycles, (c) IPBM/F with biquintic tensor-product splines on a 41×41 uniform grid, (d) IPBM/C with biquintic tensor-product splines on a 41×41 uniform grid. For both (a) and (b), we adaptively refine using the residual as an error indicator. In the following table we give the size of the system matrix G and its condition number for the final refined triangulation, along with the usual max and RMS errors at boundary points on $\partial\Omega$ and at 36114 points covering the disk Ω. The resulting H-triangulations for (a) and (b) are shown in Fig. 8.

	Time	Size	Cond	emaxB	rmsB	emax	rms
(a)	9.18	8633	2.91e+14:	1.06e−08	2.32e−09	6.22e−06	1.96e−06
(b)	7.25	9057	3.78e+11	2.42e−08	9.54e−09	4.24e−05	9.50e−06
(c)	5.31	2025	5.55e+13	6.85e−17	2.44e−17	3.36e−05	1.93e−06
(d)	3.73	2025	2.69e+11	5.52e−13	3.76e−13	3.15e−05	1.94e−06

Both adaptive methods produce reasonable triangulations for this test function. However, IPBM with tensor-product splines gave comparable accuracy with fewer degrees of freedom and in less time. □

For our second example we use the true solution given in (6.2). This is a more difficult function to approximate, and will require more degrees of freedom. Due to the nature of the solution, for this example we start with a uniform type-1 triangulation rather than a type-2 triangulation.

Example 6.2 Solve the Poisson problem on the disk Ω of radius $1/2$ centered at the point $(.5, .5)$ with true solution (6.2) using C^0 quintic splines on adaptively refined H-triangulations.

Discussion: We solve this problem in the following different ways: (a) using IPBM/F with $\mathcal{S}_5^0(\triangle)$ and an adaptive algorithm starting with a 10×10 type-1 triangulation and applying

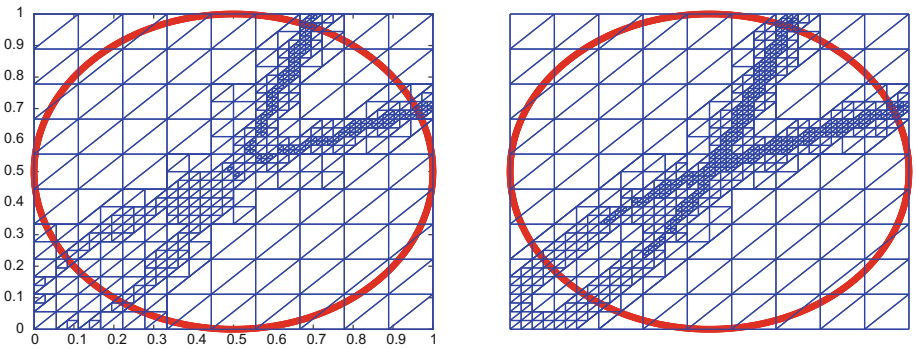

Fig. 9 H-triangulations obtained by adaptive IPBM/C and IPBM/F for Example 6.2

15 cycles of uniform refinement, (b) using IPBM/C with $\mathcal{S}_5^0(\triangle)$ and an adaptive algorithm starting with a 10×10 type-1 triangulation and applying 20 cycles of uniform refinement, (c) IPBM/F with degree five tensor-product splines on a 121×121 uniform mesh, and (d) IPBM/C with degree five tensor-product splines on a 121×121 uniform mesh. The following table shows the same values as in the previous example.

	Time	Size	Cond	emaxB	rmsB	emax	rms
(a)	25.20	11898	2.72e+16	6.44e−04	8.10e−05	5.88e−03	6.51e−04
(b)	32.80	16293	7.90e+16	1.91e−03	3.26e−04	1.08e−02	1.66e−03
(c)	155.87	15625	1.57e+18	1.06e−02	8.75e−04	1.16e−02	7.02e−04
(d)	103.25	15625	2.82e+15	1.55e−02	2.71e−03	1.55e−02	8.09e−04

The H-triangulations produced by the adaptive IPBM/C methods (a) and (b) are shown in Fig. 9—they are quite similar to each other. We can see how the refinements are concentrated along the curves where u has very steep derivatives. Here we can clearly see the advantage of using H-triangulations. We got the best accuracy in the least time with the adaptive IPBM/F method. □

7 IPBM with Splines on T-Meshes

A T-mesh is similar to an H-triangulation except that instead of using triangles we use rectangles. More precisely, a *T-mesh* of a domain Ω is a union of axis-aligned rectangles that cover Ω such that two rectangles touch each either only at points on their edges. T-meshes are more general than tensor-product meshes since they allow hanging vertices, and moreover the domain need not be a rectangle. For examples of T-meshes, see Fig. 10.

Given a T-mesh $\triangle$, a spline s of bidegree $d \times \tilde{d}$ is a piecewise function defined on $\triangle$ such that in each subrectangle of $\triangle$, s is a tensor-product polynomial of bidegree $d \times \tilde{d}$. Theoretical properties of spaces of splines on T-meshes have been studied in a number of papers in the literature, see e.g. [32,34] and references therein. We will work with splines on T-meshes in terms of the coefficients of the Bernstein–Bézier representation of the tensor-product polynomial pieces of the spline. We can assemble these coefficients in a vector c

of length $n_r(d+1)(\tilde{d}+1)$, where n_r is the number of rectangles in the mesh. It is easy to construct a matrix E so that such a piecewise polynomial on the mesh is continuous and continuously differentiable if and only if $Ec = 0$. So instead of working with splines that are exactly C^1, in carrying out the IPBM/C and IPBM/F methods using splines on T-meshes we can deal with smoothness as a penalty term as described in Sect. 2.

To illustrate the use of IPBM with splines on T-meshes, we now redo Examples 6.1 and 6.2 replacing H-triangulations by T-meshes. As in those examples, we take our domain Ω to be the disk of radius 1/2 inscribed in the unit square. Our aim is to solve the Laplace equation with Dirichlet boundary conditions taken from the functions (6.1) and (6.2) shown in Fig. 7.

As for H-triangulations, we begin with a simple partition, in this case a tensor-product grid, and then iteratively refine it by splitting certain rectangles. The decision of which rectangles to split can be made using an estimate for the residuals $\|Ls - f\|_{L_1(H)}$ associated with each rectangle H in the mesh. In the following experiments we refine those rectangles whose residuals are in the top 30% of all residuals, and then repeat the process by computing the new spline and associated residual and performing another round of splits.

A given T-mesh can be refined in a variety of ways, see Remark 9.25. Here we focus on *uniform refinement* in which each rectangle H is split into four subrectangles by inserting one new vertex at the center of H, and additional hanging vertices at the centers of each edge.

Example 7.1 Solve the Poisson problem (3.1) on the disk Ω of radius $1/2$ centered at the point (.5, .5) using adaptive C^0 splines on T-meshes, where the true solution u is the peak function defined in (6.1).

Discussion: We solve the problem using the following approaches: (a) adaptive IPBM/F on T-meshes with splines of bidegree 6 starting with a 5×5 tensor-product grid and executing 11 cycles, (b) adaptive IPBM/C with degree 6 splines starting with a 5×5 tensor-product grid and executing 10 cycles, (c) IPBM/F with tensor-product bidegree 6 splines on a 45×45 uniform grid, and (d) IPBM/C degree 6 tensor-product splines on a 45×45 uniform grid. In the following table we give the size of the system matrix G and its condition number for the final refined partition, along with the usual max and RMS errors at boundary points on $\partial\Omega$ and at 36,114 points covering the disk Ω. The T-meshes produced in (a) and (b) are shown in Fig. 10.

	Time	Size	Cond	emaxB	rmsB	emax	rms
(a)	11.10	10813	2.69e+13	1.63e−06	3.91e−07	1.63e−06	2.15e−07
(b)	13.48	16758	2.73e+09	7.21e−07	2.13e−07	1.85e−05	9.92e−06
(c)	9.74	2500	2.39e+15	6.20e−16	1.51e−16	3.80e−06	2.44e−07
(d)	5.77	2500	1.98e+13	2.16e−15	1.29e−15	3.88e−06	2.44e−07

For this test function the adaptive methods do not show any advantage as both the tensor-spline IPBM/F and IPBM/C methods get better accuracy in less time and with fewer coefficients. For this problem the tensor-product collocation was fastest. □

For our second example we use the true solution given in (6.2), see Fig. 7 (right). We start the adaptive methods with a uniform 5×5 grid on the enclosing rectangle.

Example 7.2 Solve the Poisson problem on the disk Ω in Fig. 10 (left) with true solution (6.2) using splines on T-meshes.

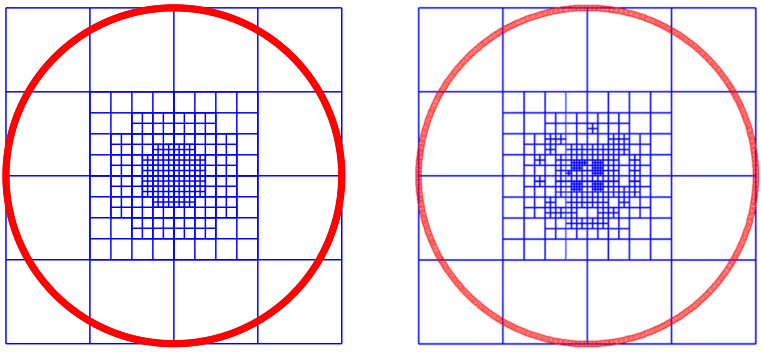

Fig. 10 T-meshes obtained by adaptive IPBM/F and IPBM/C for Example 7.1

Discussion: We solve this problem in the following ways: (a) using IPBM/F with splines of bidegree 5 and an adaptive algorithm using 10 cycles starting with a 10×10 type-1 triangulation. (b) using IPBM/C with splines of bidegree 6 and an adaptive algorithm starting with a 10×10 tensor-product grid and applying uniform refinement with 15 cycles, (c) using IPBM/F with tensor-product degree six splines on a 81×81 uniform mesh, and (d) using IPBM/C with tensor-product degree six splines on a 81×81 uniform mesh. For (a) and (b) we used $\lambda_s = 1000$, while for (c) and (d) we used $\lambda = 10$.

	Time	Size	Cond	emaxB	rmsB	emax	rms
(a)	34.71	20327	1.33e+15	5.86e−02	1.09e−02	5.86e−02	2.33e−03
(b)	38.08	23627	3.32e+16	5.60e−04	8.25e−05	2.00e−02	2.86e−03
(c)	60.48	7396	9.01e+19	2.81e−02	2.96e−03	4.95e−02	4.30e−03
(d)	26.57	7396	3.81e+15	7.71e−02	8.84e−03	7.71e−02	4.48e−03

T-meshes corresponding to methods (a) and (b) are shown in Fig. 11. We can clearly see how the refinement was concentrated along the curves where u has very steep derivatives. We got comparable errors with all four methods, although the tensor-product splines have far fewer coefficients. They are also the smoothest.

□

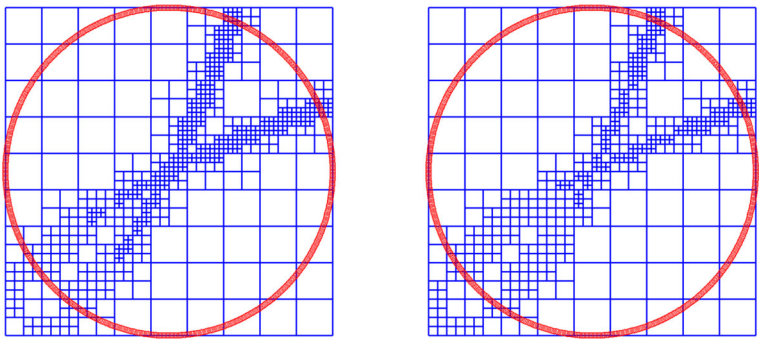

Fig. 11 The T-meshes produced by adaptive IPBM/F and IPBM/C in Example 7.2

8 The 3D-Case

In this section we explore the performance of IPBM in the case where the domain of interest lies in $\mathbb{R}^3$. Although we could use 3D analogs of all of the approximating spaces discussed above, we shall limit our discussion to C^0 polynomial splines on a tetrahedral partition of an enclosing rectangular box D. Our examples involve solving the Laplace equation with Dirichlet boundary conditions. For each example we take the true solution to be

$$u(x, y, z) = \sin(5x) + \sin(5y) + \sin(5z). \tag{8.1}$$

Example 8.1 Solve Poisson's problem on the domain shown in Fig. 12 (left) with true solution (8.1) using the IPBM/F method with C^0 quintic splines on tetrahedral partitions as shown in the figure.

Discussion: We solve the problem for a sequence of nested partitions corresponding to $m = 3, 5$ and 9 grid lines in each direction. Since we are using C^0 splines, we must add a penalty term to make the splines approximately C^1. The following table reports the computational times, numbers of degrees of freedom, condition number of the system, l_2 norm of the smoothness conditions, and the max and RMS errors on the boundary and in the interior of Ω.

m	Time	Size	Cond	Smooth	MaxB	RMSB	Max	RMS
3	0.97	1331	1.32e+07	2.26e−05	9.88e−05	1.76e−05	4.44e−04	7.69e−05
5	9.49	9261	2.69e+09	4.85e−07	2.27e−06	5.17e−07	9.90e−06	1.51e−06
9	315.68	68921	1.31e+12	6.02e−09	5.12e−08	1.11e−08	1.66e−07	2.41e−08

More experimentation is needed to determine the order of convergence of this method, but this table is showing approximately six, which would be optimal for quintic splines. □

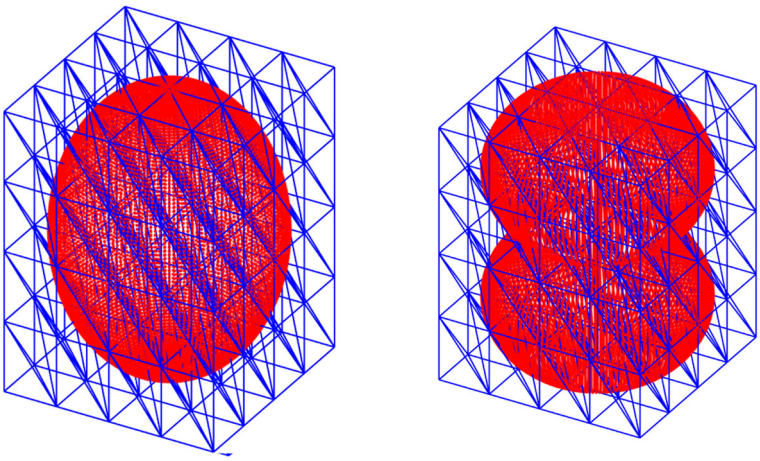

Fig. 12 Two 3D domains imbedded in tetrahedral partitions of a rectangular box

Example 8.2 Repeat Example 8.1 for the domain shown in Fig. 12 (right).

Discussion: This domain is constructed from two partial spheres joined together. We solve the problem for a sequence of nested partitions corresponding to $m = 3, 5$ and and 9 grid lines in each direction. The following table shows the same information as in the previous example.

m	Time	Size	Cond	Smooth	MaxB	RMSB	Max	RMS
3	0.64	1331	3.77e+06	1.20e−04	4.94e−04	1.43e−04	3.44e−03	1.14e−03
5	8.4	9261	3.18e+08	2.07e−06	2.00e−05	4.09e−06	8.66e−05	2.71e−05
9	333.01	68921	1.21e+11	4.11e−09	1.03e−07	2.28e−08	4.17e−07	9.40e−08

More experimentation is needed to determine the order of convergence of this method, but this table is showing approximately six, which would be optmal for quintic splines. □

9 Remarks

Remark 9.1 The idea of solving a boundary problem by immersing the domain in a larger set has been heavily studied in the literature, mostly as relates to solving problems in fluid dynamics where moving boundaries are involved, or for so-called interface problems, see e.g. [27] or the book [25] and references therein. They have also been used for problems with discontinuous coefficients. Other names used in the literature include *ghost fluid method* and *fictitious domain method*.

Remark 9.2 The immersion idea has been used to solve boundary-value problems on domains with curved boundaries using so-called WEB splines, see [1], [19] and the book [18]. This method works with modified tensor-product splines on an enclosing rectangle, and requires constructing a weight function that vanishes on the boundary.

Remark 9.3 There are a number of papers in the literature that deal with so-called *boundary penalty methods* or *penalty boundary methods*, see e.g. [2,9]. Second order elliptic problems with Dirichlet boundary conditions are treated with a finite-element method involving the immersion idea and imposing the boundary condition weakly via a penalty term involving the integral over the boundary.

Remark 9.4 An extensive survey of least-squares methods for solving PDE's (as of 1976) can be found in [13]. Of 170 papers referenced, only one is listed as involving discrete least-squares terms for both the differential equation and the boundary conditions, see [14], and it deals with two-point boundary-value problems for ODE's.

Remark 9.5 The referee kindly pointed out recent work of Bochev and Gunzburger on least-squares methods for PDE, see e.g. [3] and the book [4]. In Sect. 5.1 of [3] they formulate a collocation method similar to our (2.7) except that they do not use immersion and so their collocation points must be chosen in Ω itself. Immersion with a discrete least-squares formulation was used in [22], albeit for an advection-diffusion problem.

Remark 9.6 Methods for solving elliptic boundary-value probems with Dirichlet boundary conditions on curved boundaries based on a version of the Ritz–Galerkin method that eliminates the need to impose boundary conditions have been studied in several papers, see e.g. [5,7,8,15,36]. One approach is to use Nitsche's formulation of the variational problem which requires computing integrals along the boundary. In some of these papers the methods are called *fictitious domain* methods.

Remark 9.7 There are also related papers on the so-called *finite-cell method*, see [28,30]. These methods require adaptive quadrature rules to compute the needed integrals when dealing with problems on domains with curved boundaries.

Remark 9.8 In our formulation of the IPBM/F method we work with a least-squares problem involving the integrals of $Ls - f$ against the basis functions ϕ_i. As an alternative, we could replace the first term in (2.6) by $\int_D (Ls - f)^2$. Our experience is that using (2.6) works better.

Remark 9.9 Another way to handle boundary-value problems on domains with curved boundaries is to use the classical FEM approach, but working with approximating spaces defined on meshes which include curved triangles. If the boundary is piecewise conic, this can be done using appropriate piecewise polynomial spline spaces, see [11]. Alternatively, one can work with piecewise rationals instead, see [37].

Remark 9.10 Collocation is frequently used in the meshfree radial basis function (RBF) approach to solving boundary-value problems, see the book [16] or the survey article [23] and references therein. This approach typically collocates both the differential equation and the boundary conditions.

Remark 9.11 Classical finite-element methods based on polynomal splines work on partitions with polygonal boundaries. They can be used for problems involving curved boundaries by creating partitions whose boundaries approxate the curved boundary. But then the boundary conditions have to be imposed on the polygonal boundary, which have to be estimated from nearby points on the true boundary, with a resulting loss of accuracy, see e.g. Example 9.14 in [31].

Remark 9.12 In the examples presented here we have chosen the immersing domain to be the smallest rectangle containing the actual domain Ω. When using non-tensor splines, we can often choose smaller enclosing polygonal domains whose boundaries more closely follow the boundary of Ω. This reduces the number of degrees of freedom, but otherwise does not seem to have a major impact on the performance of the methods.

Remark 9.13 Depending on how the immersing domain is chosen, we will usually be working with some collocation points that fall outside the domain Ω where the problem is formulated. Assuming the problem has a solution on the larger domain, we do not find that this causes any problems. If desired, one could reduce the influence of collocation points falling outside of Ω by assigning small weights to the corresponding equations, but we did not find this useful in our examples.

Remark 9.14 To use the IPBM approach in practice, we need to select reasonably spaced points on the boundary to be used in forming the penalty terms in (2.6) and (2.7). This is a relatively easy task for 2D problems. If $\partial\Omega$ is given by a collection of simple curves (such as lines and circles) with arc-length parametrizations, it is easy to find points which are exactly equally spaced. For other boundary representations such as NURBs where we may not have

arc-length parametrizations, we often can get almost equally spaced points by taking equally spaced points in the parameter domain. Alternatively, one can write an algorithm which starts with an initial point and a desired spacing h, then computes the next point on the curve that is approximately h away, and then iterates. The examples in this paper involving nurbs curves were done in this way.

Remark 9.15 Finding reasonably spaced points on the boundary of a 3D domain is more difficult. Again the simplest approach is to take well-spaced points in the parameter domain and use the corresponding points on $\partial\Omega$.

Remark 9.16 If we work with piecewise polynomials on a partition of an immersing domain D in $\mathbb{R}^2$, then the second derivative of a basis function will be of size $\mathcal{O}(h^2)$ where h is the mesh size. Assuming we are working with local basis functions of height $\mathcal{O}(1)$, it follows that terms of the form $\int_D \phi_i L\phi_j$ will be of size $\mathcal{O}(1)$. This implies that λ in (2.6) can be thought of as a dimensionless parameter, i.e., its size will not have to be scaled as we scale the problems.

Remark 9.17 However, the situation is different for the IPBM/C method. Here we must work with point values of the products $\phi_i L\phi_j$, and not their integrals. Assuming we are working with a second order differential operator L, in this case to make λ dimensionless we need to multiply these products by h^4, where h is the mesh size. In our implementations we choose w in (2.7) in this way.

Remark 9.18 All of our experiments involving collocation were done using equally spaced collocation points in the mesh triangles or rectangles. For triangles, we use the $\binom{d+2}{2}$ domain points, while for rectangles we use a grid of $(d+1)(\tilde{d}+1)$ points. Collocation does not require this many points, and certainly the assembly of the collocation equations will run faster if we use fewer points. For smaller problems the reduction in running time is barely noticable, but could make a difference for very large problems.

Remark 9.19 T-splines are a generalization of polynomial splines on T-meshes that are piecewise rational rather than piecewise polynomial, see [35]. The IPBM approach can also be used with such splines. But for our purposes there does not seem to be any clear benefit in using them. We also do not have to worry about working with so-called analysis suitable meshes.

Remark 9.20 All of the results shown here are based on writing the linear systems as normal equations and using the built-in Matlab system solver. This accounts for why the condition numbers are sometimes rather large, although we got very good accuracy anyway. The performance of the algorithms can certainly be improved by using better least-squares solvers.

Remark 9.21 In solving the boundary value problems we do not need a mesh defined on the domain Ω. However, we do create such a mesh using a grid of interior points in Ω to construct a constrained Delaunay triangulation for the purpose of displaying the spline surfaces.

Remark 9.22 All of the programs used here were written in Matlab and run on a standard iMac. The times reported here are only for comparison purposes. No effort was made to optimize the code.

Remark 9.23 Although we introduced a basis in (2.4) to formulate the IPBM methods, no global basis functions are ever computed. Everything is done in terms of Bézier coefficients relative to a triangle or rectangle. Since we are not using any parametric map, no Bézier extraction techniques are required.

Remark 9.24 Adaptive methods for solving boundary-value problems have been studied in the literature in a variety of papers, including recently in the IGA literature [12,30]. The test function (6.2) was used in [20,24] to explore adaptive methods with splines on rectangles. In [20] the refinement strategy works only with ordinary triangulations, while in [24] triangulations with hanging vertices are used.

Remark 9.25 In the adaptive methods using H-triangulations we have chosen to split triangles into four similar subtriangles. An alternative would be to split them into two subtriangles. Similarly, for our adaptive methods using splines on T-meshes, we have split rectangles equally into four subrectangles, but of course there are other splitting algorithms that yield only two subrectangles for example.

Remark 9.26 In the adaptive methods of Sect. 6 we have opted to consider all triangles in the partition of D for possible refinement, not just those that overlap Ω. Limiting the refinement to only those triangles that overlap Ω does not seem to lead to better refinements. The same comment applies to the adaptive methods of Sect. 7 where we refine rectangles.

Remark 9.27 Heirarchical refinement of tensor-product meshes seems to be fashionable in the IGA community, and of course we can use that refinement strategy here too. This would amount to splitting a rectangle into a larger number of congruent subrectangles.

Remark 9.28 Here we have illustrated IPBM methods only for second order PDE's. They are of course also applicable to higher order problems such as the biharmonic equation. The simplest approach would be to use tensor-product splines on an enclosing rectangle. This way we can have whatever smoothness is required just by choosing the degree of the splines sufficiently large.

References

1. Apprich, C., Höllig, K., Hörner, J., Reif, U.: Collocation with WEB-splines. Adv. Comput. Math. **42**, 823–842 (2016)
2. Barrett, J.W., Eilliot, C.M.: Finite element approximation of the Dirichet problem using the boundary penalty method. Numer. Math. **49**, 343–366 (1986)
3. Bochev, P., Gunzburger, M.: Least-squares finite element methods. International Congress Mathematicians, vol. III, pp. 1137–1162. European Mathematical Society, Zürich (2006)
4. Bochev, P.B., Gunzburger, M.D.: Least-Squares Finite Element Methods. Applied Mathematical Sciences, vol. 166. Springer, New York (2009)
5. Bramble, J.: Rayleigh–Ritz–Galerkin methods for Dirichlet's problem using subspaces without boundary condiitions. Commun. Pure Appl. Math. **23**, 653–675 (1970)
6. Buffat, M., Penven, L.: A spectral fictitious domain method with internal forcing for solving elliptic PDEs. J. Comput. Phys. **230**, 2433–2450 (2011)
7. Burman, E., Hansbo, P.: Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method. Comput. Methods Appl. Mech. Eng. **199**, 2680–2686 (2010)
8. Burman, E., Hansbo, P.: Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method. Appl. Numer. Math. **62**, 328–341 (2012)
9. Clark, B.W., Anderson, D.C.: The penalty boundary method. Finite Elem. Anal. Des. **39**, 387–401 (2003)
10. Cottrell, J.A., Hughes, T.J.R., Bazilevs, Y.: Isogeometric Analysis: Toward Integration of CAD and FEA. Wiley, New York (2009)
11. Davydov, O., Kostin, G., Saeed, A.: Polynomial finite element method for domains enclosed by piecewise conics. Comput. Aid. Geom. Des. **45**, 48–72 (2016)
12. Dörfel, M.R., Simeon, B., Jüttler, B.: Adaptive isogeometric analysis by local h-refinement with T-splines. Comput. Methods Appl. Mech. Eng. **199**, 264–275 (2010)
13. Eason, E.D.: A review of least-squares methods for solving partial differential equations. Int. J. Numer. Methods Eng. **10**, 1021–1046 (1976)

14. Eason, E.D., Mote, C.D.: Solution of non-linear boundary value problems by discrete least squares. Int. J. Numer. Methods Eng. **11**, 641–652 (1977)
15. Embar, A., Dolbow, J., Harari, I.: Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements. Int. J. Numer. Methods Eng. **83**, 877–898 (2010)
16. Fasshauer, G.: Meshfree Approximation Methods with MATLAB. World Scientific, Singapore (2007)
17. Glowinski, R., Pan, T., Periaux, J.: A fictitious domain method for Dirichlet problem and applications. Comput. Method. Appl. Mech. Eng. **111**, 283–303 (1994)
18. Höllig, K.: Finite Element Methods with B-splines. SIAM, Philadelphia (2003)
19. Höllig, K., Reif, U., Wipper, J.: Weighted extended B-spline approximation of Dirichlet problems. SIAM J. Numer. Anal. **39**, 442–462 (2001)
20. Lai, M.-J., Mersmann, C.: An adaptive triangulation method for bivariate spline solutions of PDE's. In: Fasshauer, G., Schumaker, L. (eds.) Approximation Theory XV: San Antonio 2016, pp. 155–175. Springer, New York (2017)
21. Lai, M.J., Schumaker, L.L.: Spline Functions on Triangulations. Cambridge University Press, Cambridge (2007)
22. Laible, J.P., Pinder, G.F.: Least squares collocation solution of differential equations on irregularly shaped domains using orghogonal meshes. Numer. Method PDEs **5**, 347–361 (1989)
23. Larsson, E., Fornberg, B.: A numerical study of some radial function based solution methods for PDE's. Comput. Math. Appl. **46**, 891–902 (2003)
24. Li, S., Schumaker, L.L.: Adaptive computation with splines on triangulations with hanging vertices. In: Fasshauer, G., Schumaker, L. (eds.) Approximation Theory XV: San Antonio 2016, pp. 197–218. Springer, New York (2017)
25. Li, Z., Ito, K.: The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains. SIAM, Philadelphia (2006)
26. Li, X., Deng, J., Chen, F.: Polynomial splines over general T-meshes. Vis. Comput. **26**, 277–286 (2010)
27. Peskin, C.: The immersed boundary method. Acta Numer. **11**, 1–39 (2002)
28. Rank, E., Ruess, M., Kollmannsberger, S., Schillinger, D., Düster, A.: Geometric modeling, isogeometric analysis and the finite cell method. Comput. Methods Appl. Mech. Eng. **249–252**, 104–115 (2012)
29. Sanches, R., Bornemann, P., Cirak, F.: Immersed B-spline (I-spline) finite element method for geometrically complex domains. Comput. Methods Appl. Mech. Eng. **200**, 1432–1445 (2011)
30. Schillinger, D., Dedè, L., Scott, M.A., Evans, J.A., Borden, M.J., Rank, E., Hughes, T.J.R.: An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. Comput. Methods Appl. Mech. Eng. **249–252**, 116–150 (2012)
31. Schumaker, L.L.: Spline Functions: Computational Methods. SIAM, Philadelphia (2015)
32. Schumaker, L.L., Wang, L.: Spline spaces on TR-meshes with hanging vertices. Numer. Math. **118**, 531–548 (2011)
33. Schumaker, L.L., Wang, L.: Splines on triangulations with hanging vertices. Constr. Approx. **36**, 487–511 (2012)
34. Schumaker, L.L., Wang, L.: Approximation power of polynomial splines on T-meshes. Comput. Aid. Geom. Des. **29**, 599–612 (2012)
35. Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCSs. ACM Trans. Graph. **22**(3), 477–484 (2003)
36. Serbin, S.M.: Computational investigations of least-squares type methods for the approximate solution of boundary value problems. Math. Comput. **29**, 777–793 (1975)
37. Wachspress, E.L.: A Rational Finite Element Basis. Academic Press, New York (2012)
38. Yao, G.: Immersed Boundary Method for CFD: Focusing on its Implementation. CreateSpace (Amazon) (2016)
39. Zhu, T., Atluri, S.N.: A modified collocation method and a penalty formulation for enforcing the essential boundary conditions in the element free Galerkin method. Comput. Mech. **21**, 211–222 (1998)