Adaptive Algorithms using Splines on Triangulations with Hanging Vertices

Shiying Li

December 2, 2016

Abstract

Adaptive approximation of functions is tested using polynomial splines on triangulations with hanging vertices and indicates improved efficiency as compared to ordinary triangulations. Algorithms for generating data structures needed for triangulations with hanging vertices are also developed. Adaptive mesh generation algorithms using the finite element method(FEM) for solving a model problem involving a second order elliptic PDE of the form

$$\begin{aligned} Lu := -\nabla \cdot (\kappa \nabla u) = f \qquad & \textit{on } \Omega, \\ u = g \qquad & \textit{on } \partial\Omega_D, \\ \kappa \frac{\partial u}{\partial n} = h \qquad & \textit{on } \partial\Omega_N, \end{aligned} \tag{1}$$

are discussed, where Ω is a domain in $\mathbb{R}^2$ with polygonal boundaries and $\partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$. Numerical examples using different *a posteriori* error indicators are given. An exploration of adaptively fitting images using linear splines on such triangulations is sketched also.

1 Introduction

Bivariate splines play an important role in approximation theory and solving PDE's numerically, particularly in the FEM. When approximating a function or a solution of a PDE, it is necessary that the mesh be fine enough to represent the variation in the function or solution. Adaptive refinement of a triangulation can effectively decrease the dimension of the spline spaces used to approximate a function or the size of the linear systems that arise in the Ritz-Galerkin method while producing good accuracy. Most adaptive algorithms in solving PDE numerically are directed by *a posteriori* error estimates. A mathematical theory on error estimates for adaptive finite element solutions is developed by Babuvška and Rheinboldt in [3]. See a comprehensive theory of *a posteriori* estimation in finite element analysis in [1]. In most local refinement algorithms, efforts

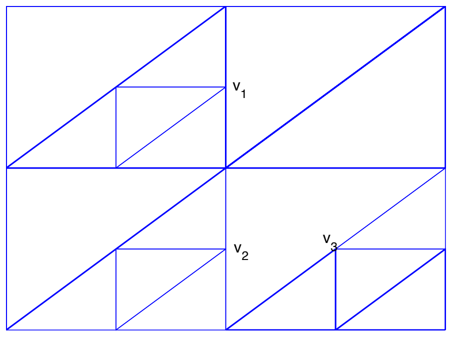


Figure 1: An H-triangulation with hanging vertices v_1, v_2, v_3.

have been made to avoid introducing hanging vertices while maintaining relatively large minimal angle in the triangulation, see [10][12]. However, allowing such vertices leads to much simpler refinement algorithms. Some recent adaptive algorithms introduce hanging nodes, see e.g. [5][18][19]. However these approaches usually involve finding mappings from a standard reference element to each element in the triangulation, which become more complicated when we want to work with splines that have C^r smoothness with $r > 0$. Our approach works directly with the B-forms(see section 3) of the splines, which eliminates the need for a reference triangle and the associated affine maps.

Dimension formulae, explicit basis functions and approximation power of spline spaces defined on triangulations with hanging vertices have been obtained in [15]. See Section 5. Similar results for spline spaces on TR-meshes and T-meshes are also obtained, see [16][17].

The rest of this paper is organized as follows. In Section 3, a brief introduction to Bernstein-Bézier representation is given. General data structure for a H-triangulation is listed in Section 4. An application to surface compression is given in Section 6. In Section 7 and 8, some basic facts about the boundary-value problem (1) and the implementation of the Ritz-Galerkin approximation using $S^0_d(\triangle)$ are presented. Our adaptive algorithms for solving the model problem (1) and numerical examples follow in Section 9. We briefly describe an exploration of image compression using linear splines on H-triangulations in Section 10.

2 Notation and definitions

See [11] [15] for more details on the following definitions.

Definition 1. *Let $\triangle := \{T_i\}_{i=1}^N$ be a collection of triangles such that the interior of the domain $\Omega := \bigcup T_i$ is connected. In addition, suppose that any pair of distinct triangles can intersect each other only at points on their edges. Then we call $\triangle$ an* **H-triangulation** *of Ω. A vertex v may lie in the interior of an* *edge of another triangle, which we call a* **hanging vertex***. See Figure 1. We refer to H-triangulations with no hanging vertices as* **ordinary triangulations***.*

Definition 2. *If $e := \langle v, w \rangle$ is a line segment of a H-triangulation $\triangle$ such that all vertices lying in the interior of e are hanging vertices, and e cannot be extended to a longer line segment with the same property, then we say e is a* **composite edge** *of $\triangle$.*

Definition 3. *We say that an H-triangulation $\triangle$ is* **regular** *provided that for every vertex v of $\triangle$, the interior of the union of triangles containing v is connected.*

Definition 4. *Suppose $w_1, ..., w_n$ is a collection of hanging vertices in an H-triangulation $\triangle$ such that for each $i = 1, ..., n$, the vertex w_i lies on a composite edge with one endpoint at w_{i+1}, where we set $w_{n+1} = w_1$. Then we say that $w_1, ..., w_n$ form a* **cycle***.*

Throughout the paper, we work with **regular H-triangulations with no cycles**.

Definition 5. *Given a H-triangulation $\triangle$ of a domain Ω, we define:*

- *$\mathcal{V}_B$= the set of boundary vertices,*
- *$\mathcal{V}_I$ = the set of interior vertices,*
- *V_H= the number of hanging vertices,*
- *V_{HB}= the number of hanging boundary vertices,*
- *V_{HI}= the number of hanging interior vertices,*
- *V_{NHB}=the number of nonhanging boundary vertices,*
- *V_{NHI}=the number of nonhanging interior vertices,*
- *E_c= the number of composite edges,*
- *N= the number of triangles,*
- *H=the number of holes in Ω,*
- *n_v=the number of edge segments ending at vertex v,*
- *m_v= the number of edge segments ending at vertex v with different slopes.*

Definition 6. *Given a positive integer d, let $\mathcal{P}_d := span\{x^i y^j\}_{0 \le i+j \le d}$ be the space of polynomials of degree d. Given $0 \le r < d$, the associated* **space of splines of degree d and smoothness r** *is defined to be the finite dimensional space*

$$S_d^r(\triangle) := \{s \in C^r(\Omega) : s|_{T_i} \in \mathcal{P}_d, \forall i = 1, ..., N\}, \tag{2}$$

where $\triangle = \{T_i\}_{i=1}^N$ is a H-triangulation of a domain Ω.

Definition 7. *Given a positive integer $d > 0$, and a triangle $T := \langle v_1, v_2, v_3 \rangle$,*

$$D_{d,T} := \{\xi^T_{ijk} := \frac{iv_1 + jv_2 + kv_3}{d}\}_{i+j+k=d}$$

is called the set of **domain points** *asscociated with d and T.*

Definition 8. *Given a function f and a spline approximation s on a rectangular domain, define the following two types of errors which are computed over a $ng \times ng$ grid on the domain:*

$$emax = \max_{1 \le i,\ j \le ng} |s(x_i, y_j) - f(x_i, y_j)|, \tag{3}$$

and

$$rms = \left[\frac{1}{ng \times ng} \sum_{i=1}^{ng} \sum_{j=1}^{ng} (s(x_i, y_j) - f(x_i, y_j))^2\right]^{1/2}, \tag{4}$$

where $s(x_i, y_j)$ and $f(x_i, y_j)$ are the values of the spline and f on the grid point (x_i, y_j) respectively.

To compare errors on different triangles, we have several error calculations, which may also provide different perspectives about error behavior.

Definition 9. *The L_2 error between two functions u and s on a triangle T is defined as*

$$terrorL2 = \int_T |u - s|^2 dx. \tag{5}$$

The maximal error over $D_{m,T}$ on a triangle T and for a fixed positive integer m is defined as

$$termax = \max_{p_i \in D_{m,T}} |u(p_i) - s(p_i)|. \tag{6}$$

Similar errors on domain points are

$$terror1 = area(T) \times \sum_{p_i \in D_{m,T}} |u - s|(p_i)|, \tag{7}$$

and

$$terror2 = area(T) \times \sum_{p_i \in D_{m,T}} |u(p_i) - s(p_i)|^2. \tag{8}$$

3 Bernstein-Bézier Methods for Spline Spaces

See [11] for more details.
Fix a triangle T with vertices located at points (x_1, y_1), (x_2, y_2), (x_3, y_3), for each point (x, y) on the plane, there exists a unique triple (b_1, b_2, b_3) such that

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix},$$

and they are called the **barycentric coordinates** of the point relative to T.

Definition 10. *Given positive integer d, the associated* **Bernstein basis polynomials of degree d relative to a triangle T** *are defined as*

$$B^d_{ijk} := \frac{d!}{i!j!k!} b_1^i b_2^j b_3^k, \qquad i+j+k=d, \tag{9}$$

where i, j, k are nonnegative integers, and $b_1(x,y)$, $b_2(x,y)$, $b_3(x,y)$ are the linear functions giving the barycentric coordinates of point (x,y) relative to triangle T.

Basic properties of Bernstein basis polynomials:

1) On the associated triangle T, the B^d_{ijk} are nonnegative and form a **partition of unity**, i.e.,
$$0 \le B^d_{ijk} \le 1 \quad \text{for all } (x,y) \text{ in T},$$
and
$$\sum_{i+j+k=d} B^d_{ijk}(x,y) = 1 \quad \text{for all } (x,y) \text{ in T}.$$

2) For every $p \in P_d$, there is a unique set of coefficients $\{c_{ijk}\}_{i+j+k=d}$ such that
$$p = \sum_{i+j+k=d} c_{ijk} B^d_{ijk}.$$

3) Let $\triangle$ be an ordinary triangulation and $D_{d,\triangle} := \bigcup_{T\in\triangle} D_{d,T}$. There is a one-to-one correspondence from $s \in S^0_d(\triangle)$ to the associated set of B-coefficients $\{c_\xi\}_{\xi\in D_{d,\triangle}}$, where
$$s|_T = \sum_{\xi\in D_{d,T}} c_\xi B^{T,d}_\xi. \tag{10}$$

There is a very stable and efficient algorithm called the **de Casteljau algorithm** for evaluating polynomials in B-form.

Definition 11. *Let $\tilde{D}_{d,\triangle} := \bigcup_{T\in\triangle} D_{d,T}$ be the associated sets of domain points of the triangulation $\triangle$, where the union is to be understood in the sense that multiple appearances of the same point are allowed. Let $S(\triangle)$ be a subspace of $PP_d(\triangle) := \{s : s|_{T_i} \in \mathcal{P}_d, \forall i = 1,...,N\}$. Suppose $\mathcal{M}$ is a subset of $\tilde{D}_{d,\triangle}$. It is said to be a* **determining set** *for $S(\triangle)$ if for any spline $s \in S(\triangle)$, $c_\xi = 0$ for all $\xi \in \mathcal{M}$ implies $s \equiv 0$, where c_ξ is the corresponding B-coefficient of s to $\xi \in \mathcal{M}$. $\mathcal{M}$ is called a* **minimal determining set** *if there is no smaller set with this property.*

It follows from linear algebra that $\dim S(\triangle) = \#\mathcal{M}$.

In general, if $\mathcal{M}$ is a minimal determining set for a spline space $S \in S^0_d(\triangle)$, then there exists an $n_c \times N$ matrix A(usually referred to transformation matrix) such that for every $c = \{c_\xi\}_{\xi \in \tilde{D}_{d,\triangle}}$, there is a $\tilde{c} := \{\tilde{c}_\xi\}_{\xi \in \mathcal{M}}$ with $\tilde{c}_\xi = c_\xi$ for all $\xi \in \mathcal{M}$ satisfying

$$c = A\tilde{c}, \tag{11}$$

where c is the vector of B-coefficients of a spline $s \in S$, and n_c is the number of B-coefficients for the space S, see [13]. In all of our computations, we make use of the transformation matrix A instead of working with explicit forms of basis functions. Note the order we store coefficients for a splines is vertices, interior domain points on edges and then interior domain points in each triangle. The order of vertices, edges, and triangles is determined after the computation of the data structure of a triangulation and domain points are stored in **lexicographical** order.With this order, each $\xi \in \mathcal{M}$, also referred to as a degree of freedom, has a unique number corresponding to it.

4 General Data Structure for a H-triangulation

We need certain lists of data similar to those used for ordinary triangulations, cf. [15]. Let

- nb = number of boundary vertices;
- n = number of vertices,
- ne = number of edges,
- nt = number of triangles,
- bdy = a vector marking the boundary vertices,
- v1, v2, v3 = lists of indices of the vertices of each triangle,
- e1, e2, e3 = lists of indices of the edges of each triangle,
- area = a vector giving the area of each triangle,
- ie1, ie2 = lists of endpoints of each edge.

The following lists storing information related to hanging vertices are also computed:

- hv = a vector marking a hanging vertex with the composite edge containing it,
- se2 = a vector of containing edges for each edge,
- be = a vector marking the boundary edges.

This information can be computed by a matlab code called `tlists`(Larry Schumaker). The degrees of freedom and transformation matrix are calculated by a matlab code called `mds0d`(Larry Schumaker).

5 Dimension formulae for $S^r_d(\triangle)$

We now present a recent theoretical result in [15].

Theorem 1. *Suppose $\triangle$ is a regular H-triangulation without cycles, and let $d \geq 4r+1$. Then*

$$\begin{aligned} dim S^r_d(\triangle) &= \frac{d^2+r^2-r+d-2dr}{2} V_{NHB} \\ &+ \frac{d^2+3r^2-4dr-d-r}{2} V_{HI} + (d-r)(d-2r)V_{NHI} \\ &+ \frac{-2d^2+6rd-3r^2+3r+2}{1-H} + \sum_{v\in\mathcal{V}_I} \sigma_v, \end{aligned} \tag{12}$$

where

$$\sigma_v := \sum_{j=1}^{r} (r+j+1-jm_v)_+.$$

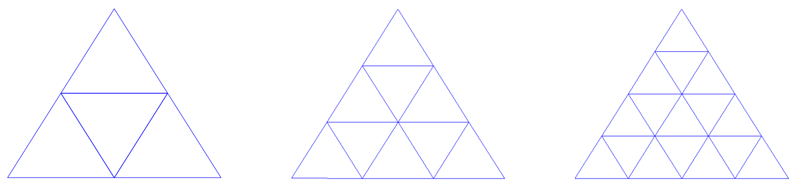
6 Surface Compression

In this section we give an algorithm for adaptively fitting a function using splines defined on an H-triangulation.

The key steps of the algorithm are:

1. Input a degree d for the spline space S^0_d and the number of refinement steps. Compute the data structure using `trilists` (see [13]) for a coarse ordinary triangulation $\mathcal{T}$.
2. Interpolate the function with a spline of degree d on $\mathcal{T}$ and calculate all B-coefficients, which are stored in a coefficient vector c.
3. Compute the discrete L_1 or L_2 error calculated over domain points $D_{m,T}$ for some fixed degree m on each triangle T. Select the triangle with the largest error.
4. Subdivide the selected triangle and locally update the data structure for the new triangulation. In addition, c is updated locally and so is the list of triangles on which the error needs to be recalculated.
5. Update errors and select the triangle with the largest error. Go back to 4 until the number of refinement steps is reached.

A stable algorithm by Larry Schumaker refines a group of triangles with errors larger than a fixed ratio of the maximal error at each loop, and then updates the data structure using `tlists` and computes the minimal determining set and transformation matrix with the code `mds0d`. The coefficient vector c is computed as in (11).



(a) $m = 2$ (b) $m = 3$ (c) $m = 4$

Figure 2: m-refinement.

Let T be a triangle, given a positive integer m, a m-refinement of T divides T into m^2 similar triangles. A few examples are given in Figure 2. In step 4, different m can be chosen for subdivision according to the specific problem. For adaptive FEM codes, we compute and update the transformation matrix A and degrees of freedom in step 2 and step 4 respectively, since a linear system of equations needs to be solved at each refinement step.

One advantage of our adaptive methods lies in the local update of the data structure, transformation matrix and degrees of freedom for each refinement. Many adaptive methods try to eliminate hanging vertices after refining an triangle by algorithms such as `Newest-node bisection` and `Longest-edge bisection`, which have to be implemented with some care, in order to avoid very thin triangles [10]. In order to trace the list of triangles on which the errors have changed, a way of comparing the levels of hanging vertices/triangles is needed to get the chains of relations as in (15) and the chains of triangles of which some B-coefficients are forced to change after a refinement. Hence after each refinement only a small number of coefficients and errors are recomputed, which adds some efficiency to the program. We have written a matlab code called `Padapta` to test this adaptive algorithm for function interpolation. A motivating example is interpolating a function with a sharp peak. One would expect more refinements in the region where the peak(sharp change) arises. Adaptively refining the coarse triangulation will not only reduce the computational time, but also reduce the number of degrees of freedom effectively.

Example 1. *Consider the function*

$$f = e^{-500[(x-.375)^2+(y-.375)^2]},$$

which is defined on the rectangular domain $[0,1] \times [0,1]$*. See Figure 3. Fit the surface generated by* f *using continuous splines.*

Discussion: We start with a triangulation `type2.25` on the domain $[0,1] \times [0,1]$(see figure 4) and use degree 4 splines to approximate the function. Note the `emax` and `rms` in the following tables are computed over a 91×91 grid on the domain. In the following tables, the number of refinement steps and the number of coefficients are denoted as `nr` and `nc` respectively. We can see using adaptive refinement that better accuracy can be achieved with many fewer coefficients.

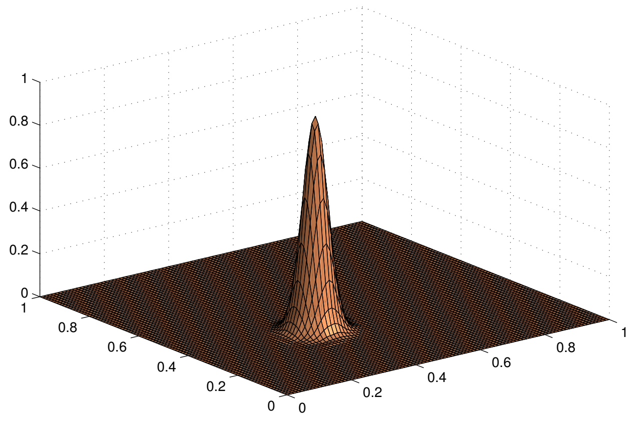


Figure 3: Plot for $f = e^{-500[(x-.375)^2+(y-.375)^2]}$

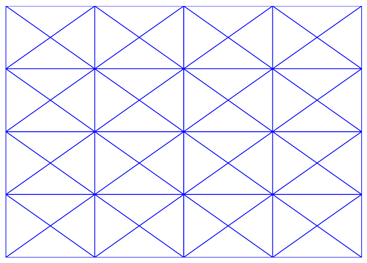


Figure 4: type2.25

Two plots of triangulations after refining `type2.25` (using 2-refinement, see figure 2) are shown in Figure 5 and Figure 6.

nr	nc	emax	rms
1	2113	1.53(-2)	1.05(-3)
2	8321	1.18(-3)	4.80(-5)

Table 1: Table of errors for uniform refinement.

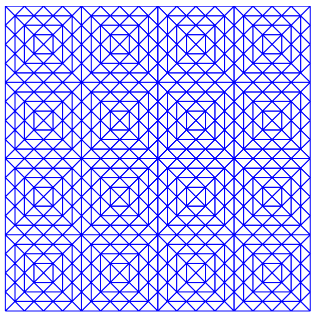


Figure 5: The triangulation after 2 uniform refinements.

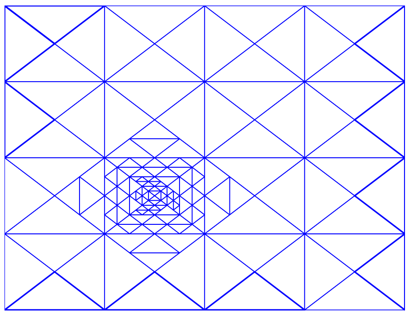


Figure 6: The triangulation after 35 adaptive refinements.

nr	nc	emax	rms
15	961	1.18(-3)	6.37(-5)
25	1240	1.18(-3)	4.51(-5)
35	1520	3.76(-4)	1.74(-5)

Table 2: Table of errors for adaptive refinement.

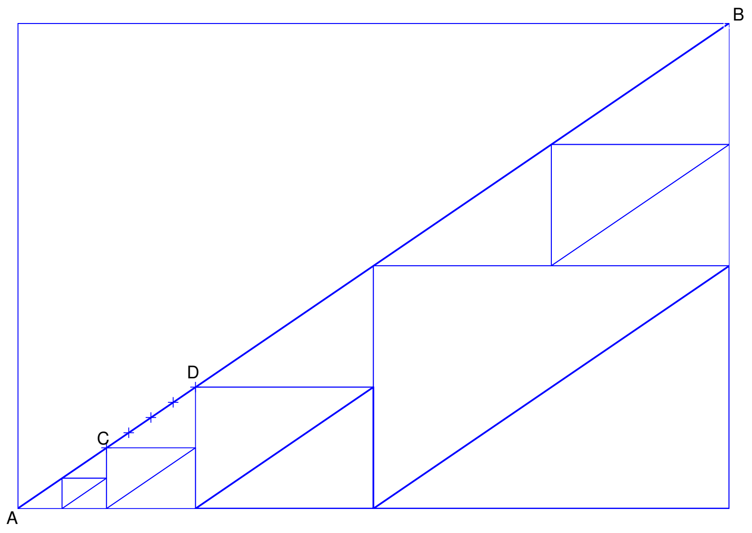


Figure 7: An example of part of a triangulation.

Now we give more details on how to enforce continuity for splines defined on H-triangulations. To maintain the continuity across the edges, domains points on the interior of an edge contained in a composite edge and hanging vertices cannot be the degrees of freedom and their B-coefficients have to be constrained by those of the degrees of freedom. To determine the B-coefficients of domain points on a constrained edge, we interpolate on the domain points using the information of the containing composite edge. For example, in Figure 6, let $\{\xi_i\}_{i=1}^{d+1}$ be the domain points on edge CD, and $\{c_i\}_{i=1}^{d+1}$ be the coefficients associated with there domain points. We know that on edges the Bernstein basis polynomials reduce to the 1-dimensional form:

$$B_{i,d}(x) = \begin{pmatrix} d \\ i \end{pmatrix} x^i (1-x)^{d-i}, \tag{13}$$

where $i = 0, ..., d$. Let $\{x_i\}_{i=1}^{d+1}$ and $\{z_i\}_{i=1}^{d+1}$ be the corresponding parameter in (13) for $\{\xi_i\}_{i=1}^{d+1}$, relative to edge CD and edge AB respectively. If $\{C_i\}_{i=1}^{d+1}$ are the B-coefficients corresponding to the $d+1$ domain points on edge AB, then the following equations are satisfied,

$$\sum_{j=0}^{d} c_j B_{j,d}(x_i) = \sum_{j=0}^{d} C_j B_{j,d}(z_i), \tag{14}$$

for $i = 1, ..., d+1$. For the moment, denote $B_{i,d+1}$ as B_i. Suppose

$$X = \begin{bmatrix} B_1(x_1) & B_2(x_1) & \cdots & B_d(x_1) \\ B_1(x_2) & B_2(x_2) & \cdots & B_d(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ B_1(x_{d+1}) & B_2(x_{d+1}) & \cdots & B_d(x_{d+1}) \end{bmatrix},$$

and

$$Z = \begin{bmatrix} B_1(z_1) & B_2(z_1) & \cdots & B_d(z_1) \\ B_1(z_2) & B_2(z_2) & \cdots & B_d(z_2) \\ \vdots & \vdots & \ddots & \vdots \\ B_1(z_{d+1}) & B_2(z_{d+1}) & \cdots & B_d(z_{d+1}) \end{bmatrix},$$

and $T = X^{-1}Z$. Writing (14) in matrix form , we get

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{d+1} \end{bmatrix} = T \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_{d+1} \end{bmatrix}. \tag{15}$$

7 Boundary-Value Problems and Finite Element Method

Assume in problem (1), $\kappa \in L^\infty(\Omega)$ such that there exists a positive constant c_0 with $\kappa(x,y) \geq c_0$ for all $(x,y) \in \Omega$, $f \in L^2(\Omega)$, $g \in H^1(\partial\Omega_D)$ and $h \in L^2(\partial\Omega_N)$. A weak formulation of the problem is : Find $u \in H^1(\Omega)$ with $u|_{\partial\Omega_D} = g$ and

$$\int_\Omega \kappa \nabla u \cdot \nabla v = \int_\Omega gv + \int_{\partial\Omega_N} hv, \tag{16}$$

for all $v \in V := \{v \in H^1(\Omega) \mid v|_{\partial\Omega_D} = 0\}$. Define the following bilinear and linear forms:

$$a : V \times V \to \mathbb{R}, \qquad a(u,v) = \int_\Omega \kappa \nabla u \cdot \nabla v, \tag{17}$$

$$l : V \to \mathbb{R}, \qquad l(v) = \int_\Omega gv + \int_{\partial\Omega_N} hv. \tag{18}$$

In this setting, we want to find $u \in H^1(\Omega)$ such that

$$a(u,v) = l(v) \qquad \forall v \in V. \tag{19}$$

We know the existence and uniqueness of the weak solution under certain conditions are guaranteed by the famous Lax-Milgram theorem:

Theorem. *(Lax-Milgram) Given a Hilbert space$(V, (\cdot,\cdot))$, a continuous, coercive bilinear form $a(\cdot,\cdot)$ and a continuous linear functional $l \in V'$, there exists a unique $u \in V$ such that*

$$a(u,v) = l(v) \qquad \forall v \in V.$$

A bilinear form $a : V \times V \to \mathbb{R}$ is continuous if there exists a positive constant C such that $|a(u,v)| \leq C\|u\|\|v\|$, $\forall u, v \in V$. It is said to be coercive if there exists a positive α such that $a(v,v)| \geq \alpha\|v\|^2$,$\forall v \in V$, where the norm here is the one associated with the inner product on V.

In practice, we use a finite dimensional space $V_h \subset V$ in place of V, which leads to the so called **Ritz-Galerkin approximation** to the solution of (19): Find $u_h \in V_h$ such that

$$a(u_h, v_h) = l(v_h) \qquad \forall v_h \in V_h. \tag{20}$$

Céa's lemma tells us u_h is the best approximation to u in V_h up to some constant:

Theorem. *(Céa) Let $a : V \times V \to \mathbb{R}$ be a continuous and coercive bilinear form. Suppose u and u_h are solution of (19) and(20) respectively, then*

$$\|u - u_h\| \leq \frac{C}{\alpha} \min_{v_h \in V_h} \|u - v_h\|,$$

where C and α are the continuity and coercivity constants respectively as in their definitions.

See more results related to finite element analysis in [1][3][10].

8 Implementation of the Ritz-Galerkin approximation using $S_d^0(\triangle)$

In this paper we choose V_h to be $S_d^0(\triangle)$, and will try to develop similar algorithms for smoother spline spaces in the future.

In [13], the Ritz-Galerkin approximation has been implemented on an ordinary triangulation using $S_d^0(\triangle)$ and several macro-element spaces to solve the model problem (1). Now let $\triangle$ be a H-triangulation, the set-up of the approximation is similar.

Given problem (1) and assumptions on κ, f, g and h as in Section 5, let $U_0 := \{s \in S_d^0(\triangle) : s(x,y) = 0, \quad \forall (x,y) \in \partial\Omega_D\}$. Suppose $\{\phi_i\}_{i=1}^{n_0}$ is a basis for U_0, and the stiffness matrix M is given by

$$M = [\langle \phi_i, \phi_j \rangle_G]_{i,j=1}^{n_0}, \tag{21}$$

where

$$\langle \phi_i, \phi_j \rangle_G := \sum_{T \in \triangle} \langle \phi_i, \phi_j \rangle_{G,T}, \quad \langle \phi_i, \phi_j \rangle_{G,T} = \int_T \kappa \nabla \phi_i \cdot \nabla \phi_j dxdy.$$

Similarly, define

$$\langle \phi_i, \phi_j \rangle_2 := \sum_{T \in \triangle} \langle \phi_i, \phi_j \rangle_{2,T}, \quad \langle \phi_i, \phi_j \rangle_{2,T} = \int_T \phi_i(x,y) \phi_j(x,y) dxdy.$$

According to (20) and using Green's identities, we know if M is nonsingular, then the approximate solution of the form

$$s = \sum_{i=1}^{n_0} c_i \phi_i + s_b, \tag{22}$$

satisfies

$$Mc = r, \tag{23}$$

where $c = (c_1, ..., c_{n_0})^T$, $r = (r_1, ..., r_{n_0})$, $r_i = \langle f, \phi_i \rangle_2 + \langle h, \phi_i \rangle_{2,\partial\Omega_N} - \langle s_b, \phi_i \rangle_G$, and where s_b is an approximation of g on $\partial\Omega_D$, see also [13].

Let $\mathcal{M}$ be a minimal determining set for U_0. It follows from linear algebra that dim U_0=#$\mathcal{M}$. In [15], an explicit construction of minimal determining sets for $S_d^r(\triangle)$ is presented for $d \geq 4r+1$. In this paper, we make use of the spline space $S_d^0(\triangle)$ for interpolation and FEM, and implementation of $S_d^r(\triangle)$ is planned for the future.

Let $\mathcal{V}_{NI}$ be the set of non-hanging interior vertices, $\mathcal{V}_{BN}$ be the set of boundary vertices that are on $\partial\Omega_N$, $\mathcal{E}_{CI}$ be the set of interior composite edges and $\mathcal{E}_{BN}$ be the set of boundary edges that are on $\partial\Omega_N$. Let $\mathcal{M}_e$ be the set of interior domain points associated with edge e, and $\mathcal{M}_T$ be the set of interior domain points associated with triangle T. Then

$$\mathcal{M} := \mathcal{V}_{NI} \cup \mathcal{V}_{BN} \cup \bigcup_{e \in \mathcal{E}_{CI} \cup \mathcal{E}_{BN}} \mathcal{M}_e \cup \bigcup_{T \in \triangle} \mathcal{M}_T, \tag{24}$$

is a minimal determining set for U_0. It's not hard to see that for each $\xi \in \mathcal{M}$, there exists a unique spline $\psi_\xi \in U_0$ such that

$$\gamma_\eta \psi_\xi = \delta_{\xi,\eta}, \quad \forall \eta \in \mathcal{M}, \tag{25}$$

where γ_η is the linear functional such that for every $s \in U_0$, $\gamma_\eta s$ is the B-coefficients of s associated with the domain point η. Moreover, $\{\psi_\xi\}_{\xi \in \mathcal{M}}$ is a stable basis for U_0, see [15].

9 Adaptive algorithms in Solving Elliptic PDEs

Our finite element solver `fem0dv` (by Larry Schumaker) using continuous splines on a H-triangulation deals with the boundary conditions and assembles the stiffness matrix similarly as in [13].

Unlike function interpolation or surface fitting, we usually do not have the true solution of a PDE for the approximation solution to compare with. Hence a good *a posteriori* error indicator is needed to choose the triangles to be refined at each step in order to get a good approximation of the true solution.

Consider our model problem (1) and suppose the spline space S^0_d is used in the Ritz-Galerkin approximation. Before introducing the error indicators, we define the following:

Definition 12. *Let u_h be the FEM solution of a PDE of the form (1), and let the residual of this problem be $Lu_h - f$.*

Definition 13. *The* **energy norm** *on a domain Ω associated with our model problem is defined by*

$$\|u\|_E = \big(\int_\Omega \kappa \nabla u \cdot \nabla u\big)^{1/2}. \tag{26}$$

We also recall the H^1-norm of a function u on a domain Ω is

$$\|u\|_{H^1} = \left(\|u\|_{L_2(\Omega)} + \|u'\|_{L_2(\Omega)} \right)^{1/2}.$$

Given a FEM solution u_h and the true solution U of a PDE, we introduce the following notations:

$$\text{energy error} := \|u_h - U\|_E\,, \tag{27}$$

$$errL2 := \left\{ \sum_{T \in \triangle} \int_T |u_h - U|^2 dx \right\}^{1/2}, \tag{28}$$

$$relH1 := \frac{\|u_h - U\|_{H^1}}{\|U\|_{H^1}} \times 100\%. \tag{29}$$

The following possible error indicators have been considered:

a. After each refinement, compare the "more accurate" FEM solution in the spline space S^0_{d+1} with the one obtained in S^0_d. Then error calculations on triangles for the difference of the two solutions can be used to get different error indicators.

b. After each refinement, compute the `terror2` of the residual evaluated over $D_{m,T}$ on each triangle T, for some fixed positive number m as an error indicator.

c. After each refinement, compute the `terrorL2` of the residual on each triangle T.

As in Example 1, we have two similar adaptive schemes:

- The key steps of the first scheme are

  1. Input a degree d for the spline space S^0_d and the number of refinement loops. Compute the data structure for H-triangulation using `tlists` for the initial coarse ordinary triangulation.
  2. Calculate the transformation matrix A and degrees of freedom using `mds0d`, and use `fem0dv` to get a FEM solution associated with the triangulation.
  3. Use one of our *a posteriori* error indicators to select a group of triangles with errors above .9 of the maximum value of the chosen error indicator.
  4. Subdivide the selected group of triangles and locally update the data structure iteratively using a matlab code `trefinem`.
  5. Go back to 2 until the number of refinement loops is reached.

- The key steps of the second scheme are

  1. Input a degree d for the spline space S^0_d and the number of refinement steps. Compute the data structure for H-triangulation using `tlists` for the initial coarse ordinary triangulation.
  2. Input the initial transformation matrix as identity matrix, and degrees of freedom as all domain points. Then use `fem0dv` to get a FEM solution associated with the triangulation.
  3. Use one of our *a posteriori* error indicators to select the triangle with the largest error.
  4. Subdivide the selected triangle and locally update the data structure, transformation matrix and degrees of freedom using a matlab code `upAsd`.
  5. Use `fem0dv` to get a FEM solution associated with the new triangulation. Go back to 3 until the number of refinement steps is reached.

Note that other stopping criterion can be chosen also, for example, when a specific type of error reaches a certain threshold.

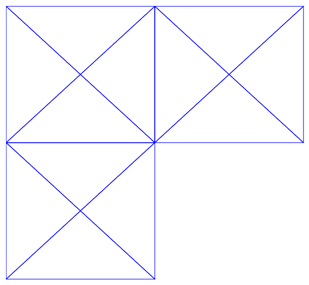


Figure 8: triL.11.

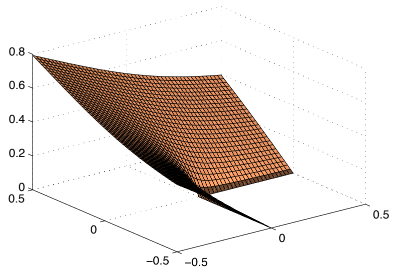


Figure 9: The true solution U.

Example 2. *Find a C^0 spline approximation to the solution of the Laplace equation on a L-shaped domain Ω with Dirichlet boundary condition given by the true solution U, i.e.,*

$$-\Delta u = 0 \qquad on\ \Omega, \tag{30}$$

$$u = U \qquad on\ \partial\Omega, \tag{31}$$

where $\Omega = [-0.5, 0.5]^2 \setminus \{(0, 0.5] \times [-0.5, 0)\} \in \mathbb{R}^2$, and

$$U(x, y) = r(x, y)^{2/3} sin(\frac{2}{3}\theta(x, y)),$$

where $(r(x, y), \theta(x, y)$ are the polar coordinates of (x, y) ($\theta \in [0, \frac{3}{2}\pi]$). See Figure 9.

Discussion: We start with a triangulation called 'triL.11', see Figure 8. Here we give results by applying the first adaptive scheme with error indicator **b** and use cubic splines. In the following table, *nr* denotes the number of refinement loops, *nd* denotes the number of degrees of freedom, *emax* and *rms* are computed over the a 51×51 grid points inside the domain. A plot of the triangulation after 15 adaptive refinement loops is also given, see Figure 10.

nr	nd	emax	rms	energy error	errL2	relH1
5	238	6.06(-3)	3.12(-4)	2.21(-2)	2.84(-4)	2.41
10	412	1.88 (-4)	6.44(-5)	6.72 (-3)	5.69(-5)	7.35(-1)
15	676	1.70(-4)	4.09(-5)	2.86(-3)	3.51(-5)	3.12(-1)
20	1030	8.02(-5)	1.40(-5)	1.29(-3)	1.23(-5)	1.41(-1)
25	1468	3.12(-5)	6.28(-6)	7.19(-4)	5.42(-6)	7.87(-2)

Table 3: Table of errors for adaptive refinements.

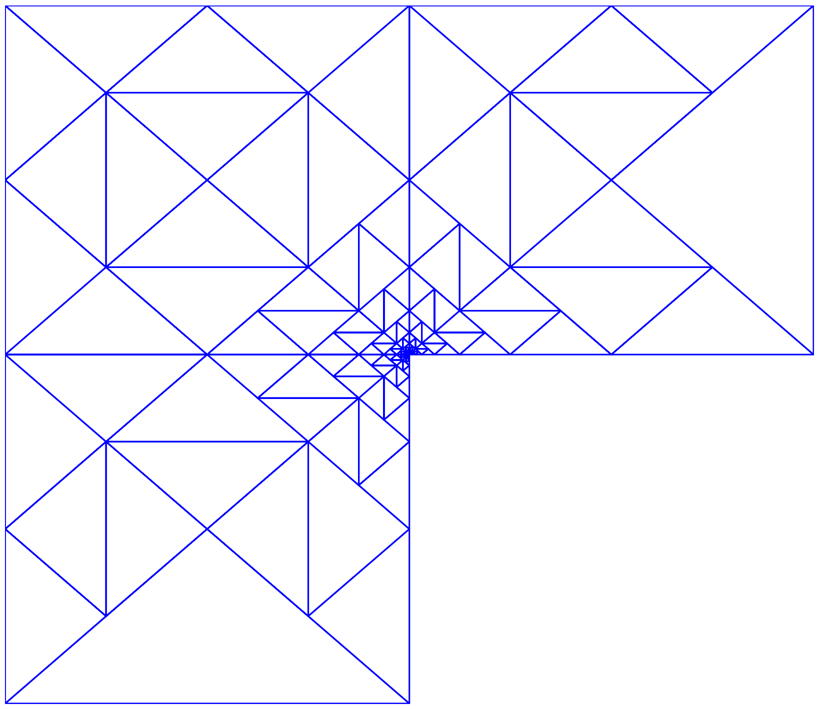


Figure 10: The triangulation after 15 refinement loops.

As expected, the table shows improvement in all types of errors as we refine the triangulations. For this problem we have a continuous boundary condition, but the true solution U has unbounded derivatives as we approach the reentrant corner from the interior of the domain. So we would expect the subdivision to focus on the reentrant corner where larger errors may occur . As we can see from Figure 10 our error indicator **b** works pretty well in selecting the triangles to refine for this problem. However for this problem, error indicator **a** seems to not work well since the splits are often away from the reentrant corner in our numerical tests. Error indicator **c** gives nonuniform splits near the reentrant corner, which in our tests seldom subdivides the triangles on the upper right. Also the improvement of errors are also worse compared to that got by error indicator **b**.

We also give a similar plot as in [18] relating the relative H^1 error defined in (29) to the number of degrees of freedom. Starting with the triangulation 'triL.11', and using cubic splines, we get Figure 11.

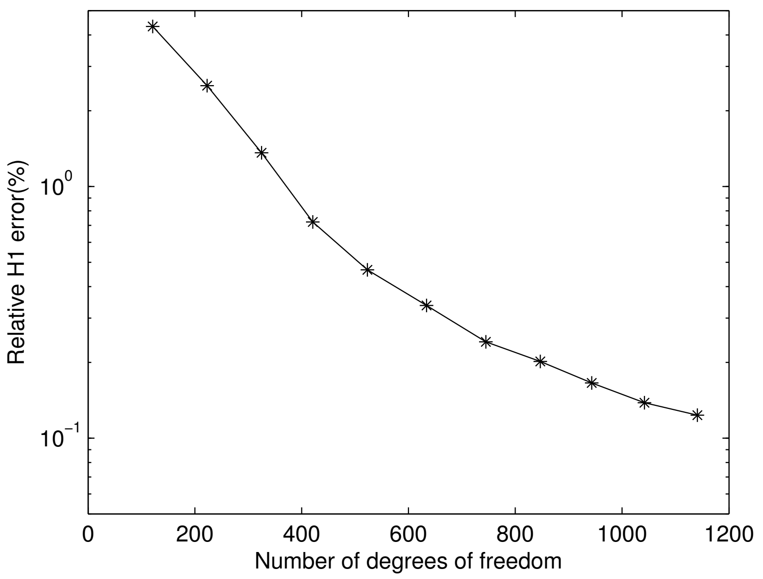


Figure 11: Convergence history for the L-shape domain problem.

10 An Exploration of Image Compression

A digital image is represented by a rectangular grid of pixels, where each pixel bears a color value or a greyscale luminance. Today grayscale images are commonly stored with 8 bits per sampled pixel, which allows 256 different intensities from 0 to 255. Sixteen bits per sample may be needed for more technical use such as medical imaging. The dimension of an image is the number of the horizontal and vertical samples in the rectangular grid representing the image. For example, a 512×512 image contains a total of 262,144 pixels. Here we restrict our discussion to greyscale images. For color images, we first convert it to a greyscale image by a Matlab function `rgb2gray`.

Because of the large number of pixels in images, a lot of triangles are used for a good approximation to the image. For adaptive algorithms, a large number of splits are made and updating the transformation matrix can be expensive if continuity of the approximating spline is required. Hence for efficiency purposes discontinuous linear splines are considered in our adaptive algorithms for image compression.

A well-known quality measure for the evaluation of image compression schemes is the *Peak Signal to Noise Ratio*(PSNR),

$$PSNR = 10 \times log_{10}\left(\frac{2^r \times 2^r}{\eta^2}\right), \tag{32}$$

where r is the number of bits per sample pixel which in our case is 8, and

$$\eta^2 = \frac{1}{N} \sum_{x \in G} |s(x) - f_x|^2, \tag{33}$$

where G denotes the collection of grid points bearing the pixels of the image, s denotes the linear spline approximating the image and f_x denotes the greyscale value at the grid point x.

Recall the Bernstein-Bézier representation for linear splines, it is easy to see that the number of coefficients of the approximating linear spline is equal to the number of vertices of the triangulation being used. For the calculation of the coefficients associated with a vertex, we take the average value of four grid points nearby. For an error indicator on a triangle T, we take the discrete l_1 norm of the difference between the true and approximate grayscale over all grid (sample) points inside the triangle T.

The compression rate is defined as follows if we quantize coefficients in $[0, 7]$ as 0, $[8, 15]$ as 8, etc.:

$$CR = \frac{np}{ns + nc/3}, \tag{34}$$

where np is the number of grid points, ns is the number of splits done in the adaptive process, and nc is the number of coefficients of the linear spline approximating the image. This quantization improves the compression rate and only slightly degrades $PSNR$.

Example 3. *We consider one popular test image* **Pepper**. *This image is of dimension* 497×499. *Choosing* $ns = 12000$*, we get* $PSNR = 27.925$ *and* $CR = 12.4387$. *The final triangulation after* 12000 *adaptive splits is shown in Figure 13.*

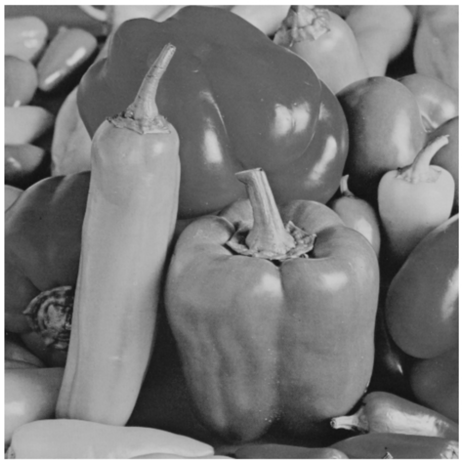


Figure 12: A 497× 499 image **Pepper**.

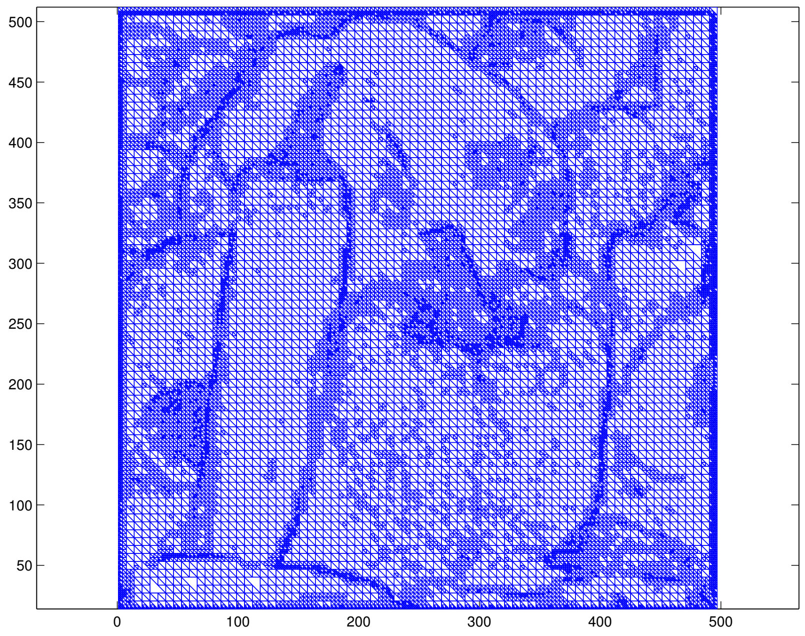


Figure 13: The final triangulation for fitting **Pepper** with $ns = 12000$.

References

[1] Ainsworth M., J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*, Wiley, New York, 2000.

[2] Arandiga, F., A. Cohen, R. Donat, N. Dyn, B. Matei, Approximation of piecewise smooth functions and images by edge-adapted (ENO-EA) nonlinear multiresolution techniques, Appl. Comput. Harmon. Anal. **24**(2) (2008), 225–250.

[3] Babuvška, I., W. C. Rheinboldt, Error estimates for adaptive finite element computations, SIAM Journal on Numerical Analysis 1978 **15**(4), 736–754.

[4] Bank, R. E., R. K. Smith, A posteriori error estimates based on hierarchical bases, SIAM Journal on Numerical Analysis **30**(4) (1993), 921–935.

[5] Carstensen, C., J. Hu, and A. Orlando, Framework for the a posteriori error analysis of nonconforming finite elements, SIAM Journal on Numerical Analysis **45**(1) (2007), 68–82.

[6] Cohen, A., N. Dyn, F. Hecht, J. Mirebeau, Adaptive multiresolution analysis based on anisotropic triangulations, Mathematics of Computation **81**(278)(2012), 789–810.

[7] Demaret, Laurent, N. Dyn, and A. Iske, Image compression by linear splines over adaptive triangulations, Signal Process **86**(7) (July 2006), 1604–1616.

[8] Erath, Christoph, D. Praetorius, A posteriori error estimate and adaptive mesh refinement for the cell-centered finite volume method for elliptic boundary value problems, SIAM J. Numer. Anal., **47**(1), 109–135.

[9] Iske, Armin, *Multiresolution methods in scattered data modelling*, Vol. 37, Springer Science & Business Media, 2004.

[10] Gockenbach, M. S., *Understanding and Implementing the Finite Element Method*, SIAM(Philadelphia), 2006.

[11] Lai, M. J., L. L. Schumaker, *Spline Functions on Triangulations*, Cambridge University Press, Cambridge, 2007.

[12] Mitchell, William F. , A comparison of adaptive refinement techniques for elliptic problems, ACM Transactions on Mathematical Software (TOMS) **15**.4 (1989), 326–347.

[13] Schumaker, L. L., *Spline Functions: Computational Methods*, SIAM, Philadelphia, 2015.

[14] Schumaker, L. L., *Spline Functions: Basic Theory*, Wiley, New York, 1981.

[15] Schumaker, L. L., L. Wang, Splines on triangulations with hanging vertices, Comput. Aided Geom. Design **30**(3)(2013), 263–275.

[16] Schumaker, L. L., L. Wang, Spline spaces on TR-meshes with hanging vertices, Numerische Mathematik **118**(3)(2011), 531–548.

[17] Schumaker, L. L., L. Wang, Approximation power of polynomial splines on T-meshes, Computer Aided Geometric Design **29**(8)(2012), 599–612.

[18] Šolń, P. , J. Červený , I. Doležel, Arbitrary-level hanging nodes and automatic adaptivity in the hp-FEM, Mathematics and Computers in Simulation **77** (1), 117–132, February, 2008.

[19] Zhao, X., Z. Shi, Q. Du, Constraint-free adaptive FEMs on quadrilateral nonconforming meshes, Journal of Scientific Computing **59**(1)(2014), 53–79.