PADOVA UNIVERSITY PRESS

Dolomites Research Notes on Approximation

Special Issue for the “10 years of the Padua points”, Volume 8 · 2015 · Pages 37–50

Good point sets and corresponding weights for bivariate discrete least squares approximation*

Marc Van Barel[a] · Matthias Humet[a]

Abstract

An algorithm is presented to compute good point sets and weights for discrete least squares polynomial approximation on a geometry $\Omega \subset \mathbb{R}^2$. The criterion that is used is the minimisation of the Lebesgue constant of the corresponding least squares operator. In order to approximate the Lebesgue constant, we evaluate the Lebesgue function in a point set generated by a refinement method that is based on Delaunay triangulation. The algorithm is greedy in the sense that points are added where the Lebesgue function is largest. It also uses a new updating algorithm for the weights such that the corresponding Lebesgue constant is made smaller. Advantages of the method are that it works for a general geometry Ω and that the point sets are nested. Numerical experiments show that the Lebesgue constant corresponding to the least squares operator is low and grows slower than polynomially in function of the total degree of the polynomial approximation space. It follows that the computed points are point sets of a weakly admissible mesh (WAM).

1 Introduction

In [10], the problem of finding good points for discrete least squares polynomial approximation on a compact domain $\Omega \subset \mathbb{C}^n$ is considered. In this context, the authors define (weakly) admissible meshes or (W)AM's, which are sequences of point sets in function of the (total) degree that can be used to measure the uniform norm of a polynomial of degree δ on a domain Ω.

In [12, 3, 6, 2, 4, 7, 8] the problem of finding good points for polynomial interpolation on a domain $\Omega \subset \mathbb{R}^2$ is treated. The criteria that are used are, among other, maximisation of the Vandermonde determinant (Fekete points) and minimisation of the Lebesgue constant. The Lebesgue constant is equal to the uniform norm of the Lebesgue function. To approximate the Lebesgue constant, the WAM's defined in [10] are used. The domains that are considered are the square, the simplex and the disk. For a more detailed overview, we refer to the introduction in our previous paper [15].

An example of good points for polynomial interpolation on the square are the Padua points, which are known analytically and have been studied in [9, 1, 5].

Our previous paper [15] advances on the ideas of the computationally oriented paper [8]. We give alternative algorithms to compute good (and almost optimal) points for polynomial interpolation on $\Omega \subset \mathbb{R}^2$ by minimisation of the Lebesgue constant. Our algorithms are faster and also work for more general geometries, e.g., the L-shape. In order to approximate the Lebesgue constant, the MATLAB package Distmesh is used, which generates point sets of a WAM according to numerical experiments.

The goal of this paper is the computation of good points and weights for discrete least squares polynomial approximation in $\mathbb{R}^2$. The criterion that is used is the minimisation of the corresponding Lebesgue constant. In order to approximate the Lebesgue constant, we evaluate the Lebesgue function in a point set generated by a refinement method that is based on Delaunay triangulation. The algorithm is greedy in the sense that points are added where the Lebesgue function is largest. It also uses a new updating algorithm for the weights such that the corresponding Lebesgue constant is made smaller.

In Section 2 we give an overview of the theory that is needed. This includes the definitions for the discrete least squares problem, the corresponding Lebesgue constant and Lebesgue function, and WAM's. Moreover, the orthogonal polynomial basis that we use is discussed. In Section 3 we give a simple updating algorithm for the least squares weights for a given point set. Section 4 contains the presentation of our main algorithm that generates nested point sets and corresponding weights per degree, for a general geometry. A refinement method using Delaunay triangulation is discussed, which is an alternative for the Distmesh package. Finally, the numerical experiments in Section 5 indicate the quality of the point sets constructed by our algorithm. In particular, these results show that the Lebesgue constant in function of the degree is low and grows slower than polynomially. It follows that the algorithm computes the first point sets of a WAM.

*The research was partially supported by the Research Council KU Leuven, project OT/10/038 (Multi-parameter model order reduction and its applications), PF/10/002 Optimization in Engineering Centre (OPTEC), by the Fund for Scientific Research–Flanders (Belgium), G.0828.14N (Multivariate polynomial and rational interpolation and approximation), and by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization). The scientific responsibility rests with its author(s).

[a]KU Leuven, Department of Computer Science, Belgium

2 Preliminaries

2.1 Discrete least squares approximation

Let $\mathcal{C}(\Omega)$ be the space of continuous functions of two variables defined on $\Omega \subset \mathbb{R}^2$, an open connected bounded subset of $\mathbb{R}^2$ together with its closure. In the sequel, we will denote this subset as the geometry of the problem, e.g., a square, a triangle, a disk, an L-shape, ... Let $\mathcal{P}_\delta$ be the space of bivariate polynomials of total degree at most δ. Let $N = \dim \mathcal{P}_\delta$. Given the points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^L \subset \Omega$ and the weights $\mathcal{W} = \{w_i\}_{i=1}^L \subset \mathbb{R}^+ \setminus \{0\}$, $L \geq N$, we call $\mathcal{L} : \mathcal{C}(\Omega) \to \mathcal{P}_\delta$ the linear operator that maps a function f to the polynomial $p_L = \mathcal{L}(f)$ of degree δ that approximates the function f according to the discrete least squares criterion

$$\min_{p_L} \left\| f - p_L \right\|_2 = \min_{p_L} \sqrt{\sum_{i=1}^{L} w_i^2 \left| f(\boldsymbol{x}_i) - p_L(\boldsymbol{x}_i) \right|^2}. \tag{1}$$

Let $\{p_j\}_{j=1}^N$ be a basis for $\mathcal{P}_\delta$, and let $V_\mathcal{X} = \left[p_j(\boldsymbol{x}_i) \right]$ be the $L \times N$ Vandermonde matrix for this basis in the points $\mathcal{X}$. Written in this basis, the least squares approximating polynomial of a function f is

$$\mathcal{L}(f)(\boldsymbol{x}) = \sum_{j=1}^{N} c_j p_j(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{x})^T \boldsymbol{c}$$

with $\boldsymbol{p} = \left[p_1 \; \ldots \; p_N \right]^T$. Let $W = \operatorname{diag}(w_1, \ldots, w_L)$ and $\boldsymbol{f} = \left[f(\boldsymbol{x}_1) \; \ldots \; f(\boldsymbol{x}_L) \right]^T$. The coefficients $\boldsymbol{c}$ can be found as the solution of the overdetermined $L \times N$ linear system

$$W V_\mathcal{X} \boldsymbol{c} = W \boldsymbol{f}, \tag{2}$$

in the least squares sense. More precisely, $\boldsymbol{c}$ is the solution of the following optimisation problem

$$\min_{\boldsymbol{c}} \left\| W \left(V_\mathcal{X} \boldsymbol{c} - \boldsymbol{f} \right) \right\|_2 \tag{3}$$

which is the matrix notation of the optimisation problem (1). Using the pseudoinverse, we can write the solution as

$$\boldsymbol{c} = (W V_\mathcal{X})^\dagger W \boldsymbol{f}.$$

and we get

$$\mathcal{L}(f)(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{x})^T (W V_\mathcal{X})^\dagger W \boldsymbol{f}. \tag{4}$$

We will assume that the least squares problem (3) has a unique solution. It is well known that this is equivalent to the condition that $W V_\mathcal{X}$ has full rank, or equivalently, since W is invertible, that $V_\mathcal{X}$ has full rank. In this case the point set $\mathcal{X}$ is called $\mathcal{P}_\delta$-determining, a property defined in [10] stating that for every $p \in \mathcal{P}_\delta$, if $p(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathcal{X}$, then $p = 0$. The equivalent condition using the Vandermonde matrix applies, because the dimension of $\mathcal{P}_\delta$ is finite.

If $L = N$ and the point set $\mathcal{X}$ is $\mathcal{P}_\delta$-determining, then $\mathcal{X}$ is called unisolvent. In this case $V_\mathcal{X}$ is an invertible matrix and the square system (2) has a (unique) solution corresponding to the polynomial that interpolates the function f in the points $\mathcal{X}$ which is given by $\mathcal{L}(f)(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{x})^T V_\mathcal{X}^{-1} \boldsymbol{f}$.

2.2 Lebesgue constant

The ∞-norm of a function f on a set $S \subset \mathbb{R}^2$ is defined as

$$\left\| f \right\|_S = \max_{x \in S} \left| f(x) \right|.$$

If $S = \Omega$, we write $\left\| f \right\| = \left\| f \right\|_\Omega$. The Lebesgue constant $\Lambda_\mathcal{L}$ is defined as the ∞-norm of the operator $\mathcal{L}$, i.e.,

$$\begin{aligned} \Lambda_\mathcal{L} &= \min \left\{ c \geq 0 : \left\| \mathcal{L}(f) \right\| \leq c \left\| f \right\| \text{ for all } f \in \mathcal{C}(\Omega) \right\} \\ &= \max_{f \neq 0} \frac{\left\| \mathcal{L}(f) \right\|}{\left\| f \right\|} = \max_{\left\| f \right\| = 1} \left\| \mathcal{L}(f) \right\| = \max_{\left\| f \right\| \leq 1} \left\| \mathcal{L}(f) \right\|. \end{aligned} \tag{5}$$

The following inequality holds

$$\left\| \mathcal{L}(f) \right\| \leq \Lambda_\mathcal{L} \left\| f \right\|, \quad \text{for all } f \in \mathcal{C}(\Omega). \tag{6}$$

Let $p^* \in \mathcal{P}_\delta$ be the best polynomial approximation of f, which minimises $\left\| f - p^* \right\|$. (Note that p_L minimises the two-norm error $\left\| f - p_L \right\|_2$.) Using the triangle inequality we have

$$\left\| f - \mathcal{L}(f) \right\| \leq \left\| f - p^* \right\| + \left\| p^* - \mathcal{L}(f) \right\|, \quad \text{for all } f \in \mathcal{C}(\Omega)$$

and then, since $\left\| p^* - \mathcal{L}(f) \right\| = \left\| \mathcal{L}(p^* - f) \right\| \overset{(6)}{\leq} \Lambda_\mathcal{L} \left\| p^* - f \right\|$, we get the well-known result

$$\left\| f - \mathcal{L}(f) \right\| \leq (1 + \Lambda_\mathcal{L}) \left\| f - p^* \right\|, \quad \text{for all } f \in \mathcal{C}(\Omega).$$

This inequality shows how the Lebesgue constant is a measure of how good the least squares approximant is compared to the best polynomial approximation.

To approximate $\Lambda_{\mathcal{L}}$, consider a point set $\mathcal{Y} = \{\boldsymbol{y}_i\}_{i=1}^K \subset \Omega$ and define the $K \times N$ Vandermonde matrix $V_{\mathcal{Y}} = \left[p_j(\boldsymbol{y}_i)\right]$. We have

$$\left\|\mathcal{L}(f)\right\|_{\mathcal{Y}} = \max_{\boldsymbol{x}\in\mathcal{Y}} \left|\boldsymbol{p}(\boldsymbol{x})^T (WV_{\mathcal{X}})^\dagger W\boldsymbol{f}\right| = \left\|V_{\mathcal{Y}}(WV_{\mathcal{X}})^\dagger W\boldsymbol{f}\right\|_\infty,$$

where the last norm is the infinity norm of a vector. By replacing Ω in (5) by its subset $\mathcal{Y}$, we obtain the following approximation for the Lebesgue constant

$$\Lambda_{\mathcal{L}} \approx \max_{\|f\|\le 1} \left\|\mathcal{L}(f)\right\|_{\mathcal{Y}} = \max_{\|f\|_\infty\le 1} \left\|V_{\mathcal{Y}}(WV_{\mathcal{X}})^\dagger W\boldsymbol{f}\right\|_\infty = \left\|V_{\mathcal{Y}}(WV_{\mathcal{X}})^\dagger W\right\|_\infty, \tag{7}$$

where the last norm is the infinity norm of a matrix.

2.3 Lebesgue function

From

$$\Lambda_{\mathcal{L}} = \max_{\|f\|\le 1} \left\|\mathcal{L}(f)\right\| = \max_{\|f\|\le 1} \max_{\boldsymbol{x}\in\Omega} \left|\mathcal{L}(f)(\boldsymbol{x})\right| = \max_{\boldsymbol{x}\in\Omega} \max_{\|f\|\le 1} \left|\mathcal{L}(f)(\boldsymbol{x})\right|$$

it follows that the Lebesgue constant is the maximum over Ω of the so-called Lebesgue function, i.e.,

$$\Lambda_{\mathcal{L}} = \max_{\boldsymbol{x}\in\Omega} \lambda_{\mathcal{L}}(\boldsymbol{x})$$

with

$$\lambda_{\mathcal{L}}(\boldsymbol{x}) = \max_{\|f\|\le 1} \left|\mathcal{L}(f)(\boldsymbol{x})\right|.$$

Using (4) we get

$$\lambda_{\mathcal{L}}(\boldsymbol{x}) = \max_{\|f\|_\infty\le 1} \left|\boldsymbol{p}(\boldsymbol{x})^T (WV_{\mathcal{X}})^\dagger W\boldsymbol{f}\right| = \left\|\boldsymbol{p}(\boldsymbol{x})^T (WV_{\mathcal{X}})^\dagger W\right\|_1, \tag{8}$$

where we get the 1-norm by choosing $\boldsymbol{f}(i) = \pm 1$ with the appropriate sign. Note that given a point $\boldsymbol{x} \in \mathcal{Y}$, $\lambda_{\mathcal{L}}(\boldsymbol{x})$ is the 1-norm of the corresponding row of the matrix $V_{\mathcal{Y}}(WV_{\mathcal{X}})^\dagger W$. Hence we get the same result for the approximation of $\Lambda_{\mathcal{L}}$

$$\Lambda_{\mathcal{L}} \approx \max_{\boldsymbol{x}\in\mathcal{Y}} \lambda_{\mathcal{L}}(\boldsymbol{x}) = \left\|V_{\mathcal{Y}}(WV_{\mathcal{X}})^\dagger W\right\|_\infty.$$

2.4 Weakly admissible meshes (WAM's)

Let $\mathcal{X}$ be $\mathcal{P}_\delta$-determining and let $C(\mathcal{X},\Omega)$ be the smallest constant such that

$$\left\|p\right\| \le C(\mathcal{X},\Omega)\left\|p\right\|_{\mathcal{X}}, \quad \text{for all } p \in \mathcal{P}_\delta. \tag{9}$$

A Weakly Admissible Mesh (WAM) is defined in [10] as a sequence of discrete subsets $\mathcal{X}_\delta \subset \Omega$ such that both the cardinality $|\mathcal{X}_\delta|$ and the constant $C(\mathcal{X}_\delta,\Omega)$ grow at most polynomially with δ. When $C(\mathcal{X}_\delta,\Omega)$ is bounded above, independent of δ, then the sequence $\{\mathcal{X}_\delta\}$ is an Admissible Mesh (AM).

Remember that the least squares operator $\mathcal{L}$ depends on the points $\mathcal{X}$, the weights $\mathcal{W}$ and the approximation space $\mathcal{P}_\delta$. The following equalities show that $\Lambda_{\mathcal{L}}$ is not only the smallest constant such that $\left\|\mathcal{L}(f)\right\| \le c\left\|f\right\|$ for all f, but also the smallest constant such that $\left\|\mathcal{L}(f)\right\| \le c\left\|f\right\|_{\mathcal{X}}$ for all f:

$$\begin{aligned}
\Lambda_{\mathcal{L}} &= \min\left\{c \ge 0 : \left\|\mathcal{L}(f)\right\| \le c\left\|f\right\| \text{ for all } f \in \mathcal{C}(\Omega)\right\} \\
&= \max_{f\neq 0} \frac{\left\|\mathcal{L}(f)\right\|}{\left\|f\right\|} \\
&\overset{A}{=} \max_{\|f\|=1} \left\|\mathcal{L}(f)\right\| \\
&\overset{B}{=} \max_{\|f\|_{\mathcal{X}}=1} \left\|\mathcal{L}(f)\right\| \\
&\overset{A}{=} \max_{f\neq 0} \frac{\left\|\mathcal{L}(f)\right\|}{\left\|f\right\|_{\mathcal{X}}} \\
&= \min\left\{c \ge 0 : \left\|\mathcal{L}(f)\right\| \le c\left\|f\right\|_{\mathcal{X}} \text{ for all } f \in \mathcal{C}(\Omega)\right\}
\end{aligned}$$

Equalities A follow from the linearity of $\mathcal{L}$ and the norm $\|\cdot\|_\Omega$ and equality B follows from the fact that $\mathcal{L}(f)$ only depends on the values of f in $\mathcal{X}$.

We can also interpret the least squares operator as a function $\Phi : \mathbb{R}^L \to \mathcal{P}_\delta$, because $\mathcal{L}(f)$ depends only on the vector formed by the values of f in $\mathcal{X}$. It follows that

$$\Lambda_{\mathcal{L}} = \min\left\{c \geq 0 : \|\Phi(v)\| \leq c\,\|v\|_\infty \text{ for all } v \in \mathbb{R}^L\right\}, \tag{10}$$

i.e., the Lebesgue constant is also the operator norm of Φ. The constant $C(\mathcal{X},\Omega)$ defined in (9) can be written as

$$\begin{aligned} C(\mathcal{X},\Omega) &= \min\left\{c \geq 0 : \left\|p\right\| \leq c\left\|p\right\|_{\mathcal{X}} \text{ for all } p \in \mathcal{P}_\delta\right\} \\ &= \min\left\{c \geq 0 : \|\Phi(v)\| \leq c\,\|v\|_\infty \text{ for all } v \in \mathcal{S}_\delta\right\}, \end{aligned} \tag{11}$$

where $\mathcal{S}_\delta = \left\{\begin{bmatrix} p(\boldsymbol{x}_1) & \cdots & p(\boldsymbol{x}_L)\end{bmatrix}^T : p \in \mathcal{P}_\delta\right\} \subset \mathbb{R}^L$. Hence $C(\mathcal{X},\Omega)$ is the operator norm of the restriction of Φ to the subset $\mathcal{S}_\delta$.

Since $\mathcal{S}_\delta \subset \mathbb{R}^L$, it follows from (10) and (11) that

$$C(\mathcal{X},\Omega) \leq \Lambda_{\mathcal{L}}. \tag{12}$$

If $L = N$, then the point set $\mathcal{X}$ is unisolvent, hence $\mathcal{S}_\delta = \mathbb{R}^L$ and we get the equality $C(\mathcal{X},\Omega) = \Lambda_{\mathcal{L}}$.

Two remarks can be made concerning this important upper bound. First, since the Lebesgue constant $\Lambda_{\mathcal{L}}$ depends on the weights of the least squares operator, every set of weights corresponds to an upper bound for $C(\mathcal{X}_\delta,\Omega)$. By optimising over the weights, we can try to find an upper bound that is as small as possible. In Section 3 we give such an algorithm. Second, in our paper [15] we computed $\Lambda_{\mathcal{L}}$ as an upper bound for $C(\mathcal{X}_\delta,\Omega)$ using uniform weights for the least squares operator. A similar method was used in [6].

The goal of this paper is to compute a sequence of point sets $\mathcal{X}_\delta$ having small Lebesgue constants $\Lambda_{\mathcal{L}}$, such that the cardinality $|\mathcal{X}_\delta|$ of $\mathcal{X}_\delta$ grows polynomially. Our numerical experiments indicate that $\Lambda_{\mathcal{L}}$ grows (slower than) polynomially, i.e., that our point sets form (the first point sets of) a WAM.

2.5 Orthogonal polynomial basis

The formulas for the Lebesgue constant in (7) and the Lebesgue function in (8) make use of the pseudoinverse. In practice, to evaluate these expressions, we solve a linear least squares problem with L right hand side vectors. It is well known that when using a backward stable algorithm to solve a linear least squares problem, the relative accuracy of the computed solution is determined by the condition number of the coefficient matrix (see [11, Theorem 5.3.1]), in this case the weighted Vandermonde matrix $WV_{\mathcal{X}}$. Hence the condition number of $WV_{\mathcal{X}}$ must not be too large.

In this subsection we review some good bases for certain geometries and briefly introduce the orthonormal bases of [14] which we used in our numerical experiments and also in our previous paper [15]. Details about the computation of the orthonormal bases are left out and we refer the reader to Section 4 of [15].

Let $\mathcal{M} = \{m_i\}_{i=1}^N$ be a monomial basis for $\mathcal{P}_\delta$, which depends on how the monomials are ordered. In this paper, we use the graded lexicographic order

$$\mathcal{M} = \{1, y, x, y^2, xy, x^2, \ldots\},$$

in the variables x and y. The monomial basis is not a good basis for least squares in general, because the corresponding weighted Vandermonde matrix can have a large condition number. For particular geometries, suitable bases may be available, e.g., in [8] the authors use product Chebyshev polynomials for the square, Dubiner polynomials for the simplex and Koornwinder type II polynomials for the disk in the context of polynomial interpolation. However, if we consider the general setting, we need to be able to build such good bases.

An optimal basis for $\mathcal{P}_\delta$ for our least squares problem is an orthogonal polynomial basis $\{p_j\}_{j=1}^N$ with respect to the discrete inner product

$$\langle p, q\rangle = \sum_{i=1}^{L} w_i^2 \overline{p(\boldsymbol{x}_i)} q(\boldsymbol{x}_i), \tag{13}$$

i.e., $\langle p_i, p_j\rangle = \delta_{i,j}$ and $p_j = \sum_{i=1}^j u_{i,j} m_i$, with $u_{j,j} \neq 0$. Indeed, if $V_{\mathcal{X}} = \begin{bmatrix} p_j(\boldsymbol{x}_i)\end{bmatrix}$ is the $L \times N$ Vandermonde matrix corresponding to this orthogonal basis and the points $\mathcal{X}$, and if $W = \operatorname{diag}(w_i)$, then $WV_{\mathcal{X}}$ is an orthogonal matrix with condition number 1. In our final algorithm, we will iteratively change the weights without updating the basis, but if the weights do not change too much, the condition number of $WV_{\mathcal{X}}$ will remain relatively low. Details on how to compute such a basis are given in [15].

Remark 1. Note that for the square $[-1,1]^2$ the product Chebyshev basis is an example of an orthogonal polynomial basis with respect to the discrete inner product (13) where the points $\mathcal{X}$ are the Padua points with specific weights that are given in [1, Theorem 1].[1]

[1]Note that if L is the number of Padua points, then there are $L-1$ product Chebyshev polynomials that are orthonormal w.r.t. the discrete inner product, but not L.

3 Optimising the discrete least squares weights

The Lebesgue constant for polynomial interpolation depends on the point set $\mathcal{X}$, the approximation space $\mathcal{P}_\delta$ and the geometry Ω. For least squares approximation, the Lebesgue constant also depends on the weights $\mathcal{W}$. In this section, we introduce Algorithm 3.1, a simple updating algorithm that improves the weights $\mathcal{W}$ for a given point set $\mathcal{X}$. In each iteration, each weight is multiplied by the value of the Lebesgue function in the corresponding point, and the weights are normalised. By updating the weights in this way, relatively more weight is added to the points in $\mathcal{X}$ where the Lebesgue function is larger. The idea is that this will result in lowering the Lebesgue constant. To make the algorithm robust, we use the following stopping criterion: stop if either

(i) the Lebesgue constant does not decrease or

(ii) the relative change of the Lebesgue constant is smaller than some threshold ϵ_Λ.

Algorithm 3.1 Least squares weight updating

```
Input: geometry $\Omega$, degree $\delta$, basis for $\mathcal{P}_\delta$, $k_{max}$, $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^L \subset \Omega$
Output: weights $\mathcal{W}$
  $\mathcal{W} \leftarrow \{w_i = 1/\sqrt{L},\ i = 1,\dots,L\}$, normalised equal weights
  $V_\mathcal{X} \leftarrow$ evaluate basis functions in points $\mathcal{X}$
  for $k = 1,\dots,k_{max}$ do
      Evaluate Lebesgue function in points $\mathcal{X}$ using (8):
        $\lambda_i \leftarrow \left\|\boldsymbol{p}(\boldsymbol{x}_i)^T (W V_\mathcal{X})^\dagger W\right\|_1,\ i = 1,\dots,L$
      Update weights $\mathcal{W}$:
        $w_i \leftarrow w_i \cdot \lambda_i,\ i = 1,\dots,L$
      Normalise weights
      if Stopping criterion then
          Break
      end if
  end for
```

We illustrate the effectiveness of the algorithm with the following numerical experiments. Let $\mathcal{X}$ be an equispaced mesh of size 100×100 on the square $\Omega = [-1,1]^2$.

- We run the algorithm for $\delta = 10$, with $\epsilon_\Lambda = 10^{-3}$. The algorithm stops after 18 iterations because the relative change of the Lebesgue constant is lower than ϵ_Λ. The resulting Lebesgue function with the optimised weights is shown in Figure 1 and the weights are shown in Figure 2. Note the particular pattern of the Lebesgue function. Compare this with the Lebesgue function for equal weights in Figure 3.
- We run the algorithm for the degrees $\delta = 1,\dots,20$, with $\epsilon_\Lambda = 10^{-3}$. In Figure 4 the Lebesgue constant, here denoted as Λ_δ in function of the degree, is shown for each iteration step of the algorithm for the degrees $\delta = 5, 10$ and 20. In Figure 5 the resulting Lebesgue constant reached at the end of the algorithm is plotted in function of the degree δ. For every degree the algorithm stops because the relative change of the Lebesgue constant is lower than ϵ_Λ.

We conclude that by updating the weights with Algorithm 3.1 we get a much lower Lebesgue constant, which improves the general quality of the least squares approximation operator. Moreover, note that when using equal weights, the Lebesgue constant for the least squares operator corresponding to the equispaced mesh $\mathcal{X}$ is a large overestimation of the constant $C(\mathcal{X},\Omega)$ in (9). Using Algorithm 3.1 we obtain weights for which the Lebesgue constant is much lower and hence a much better bound for $C(\mathcal{X},\Omega)$.

Comparing Figures 1 and 3, we note that our computed weights push all the local maxima of the Lebesgue function to almost the same level, obtaining some sort of equi-oscillating effect. Note that a similar result was obtained with our global optimisation algorithm in [15] and for univariate polynomial approximation, the equi-oscillation property is a necessary and sufficient condition to obtain the best polynomial approximant in the uniform norm, see [13, Theorem 10.1]. This indicates that the computed weights are close to optimal and that the Lebesgue constant can probably not be made much smaller by further optimising the weights. Note that the computed weights in Figure 2 are larger near the boundaries and especially near the corners of the square. These weights give an indication of the distribution of a good point set for this geometry.

Finally, we observe in Figure 4 that the value of the Lebesgue constant decreases a lot in the first iterations. It follows that in practice, the stopping criteria can be chosen rather tight, since not many iterations are needed to get a rather low value of the Lebesgue constant.

4 Good point sets for discrete least squares polynomial approximation

In this section we present an algorithm that computes a sequence of point sets $\mathcal{X}_\delta$ and corresponding weights $\mathcal{W}_\delta$, for $\delta = 1, 2, \dots, \delta_{\max}$, with the following properties.

- The point sets $\mathcal{X}_\delta$ are nested, i.e., $\mathcal{X}_{\delta-1} \subset \mathcal{X}_\delta$ for $\delta = 2, 3, \dots, \delta_{\max}$.
- The number of elements in each point set $\mathcal{X}_\delta$ grows linearly in the dimension of the vector space $\mathcal{P}_\delta$, i.e.,

$$|\mathcal{X}_\delta| = \mathrm{round}(\alpha N_\delta) + c, \quad \text{with} \quad N_\delta = \dim \mathcal{P}_\delta,$$

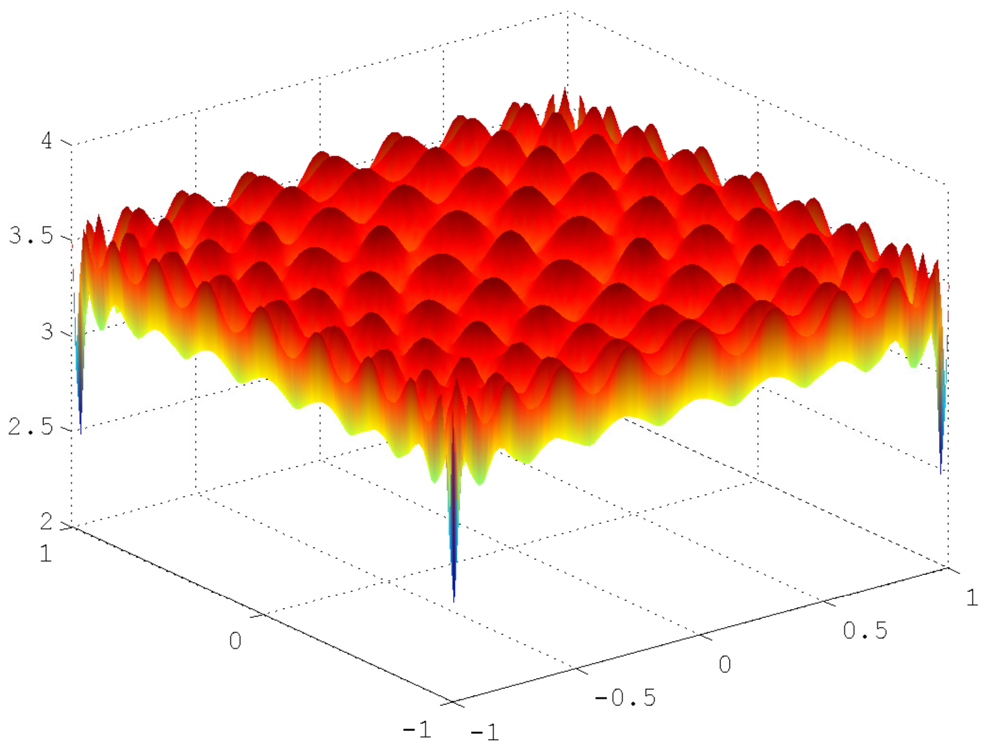

Figure 1: Lebesgue function for $\delta = 10$ after 18 iterations.

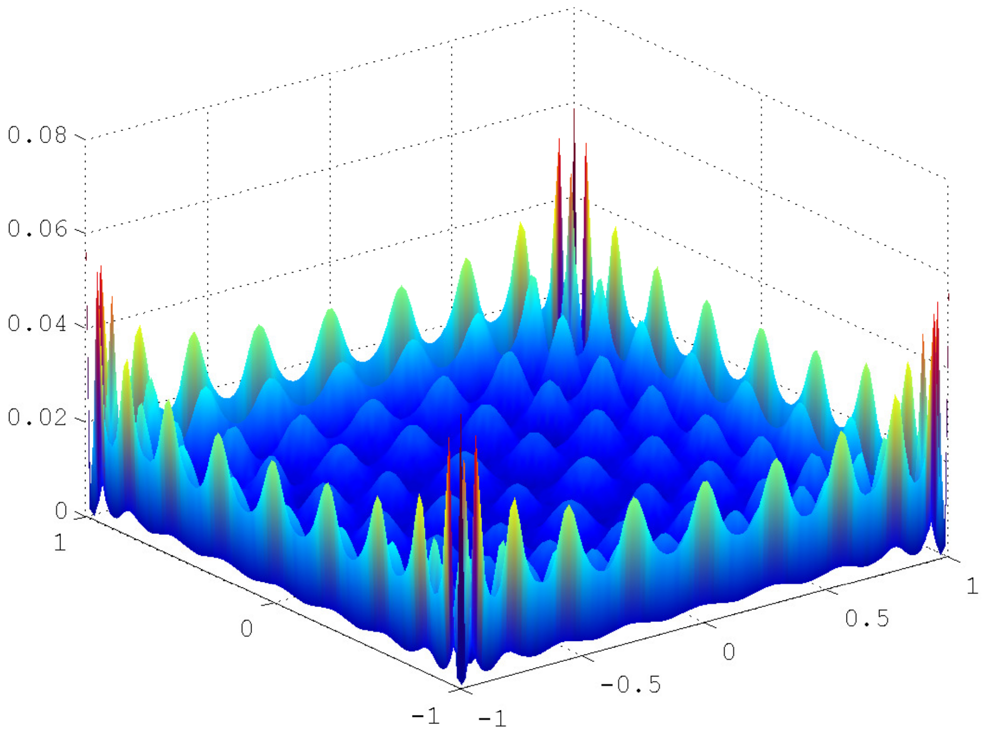

Figure 2: Resulting weights for $\delta = 10$ after 18 iterations.

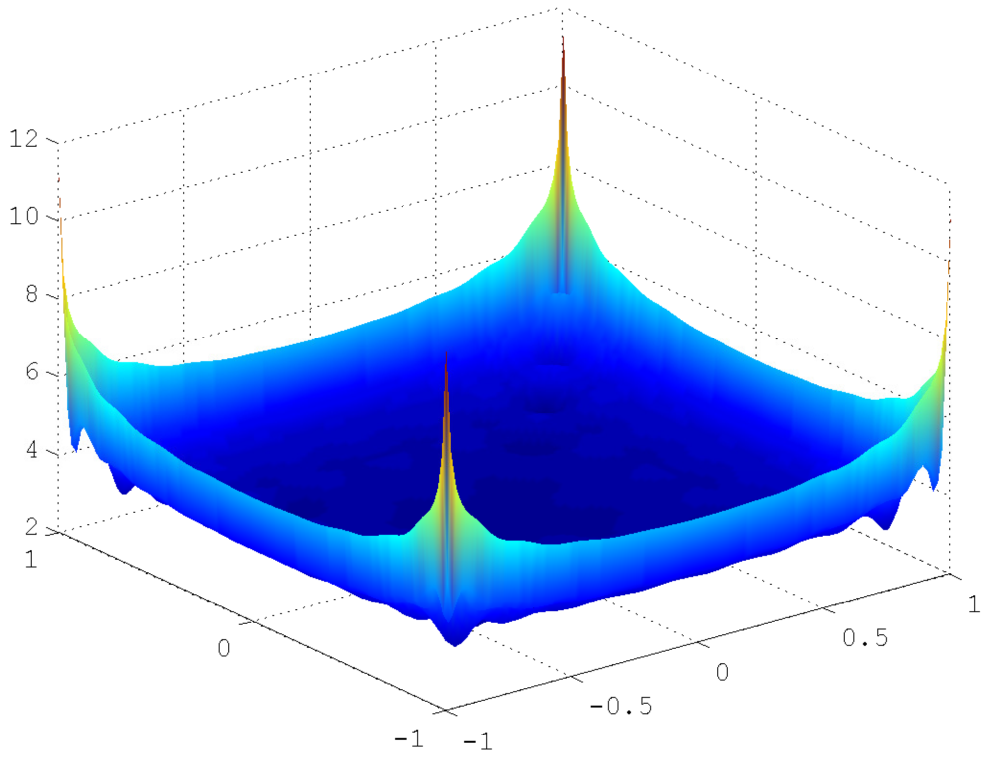

Figure 3: Lebesgue function for $\delta = 10$ for equal weights.

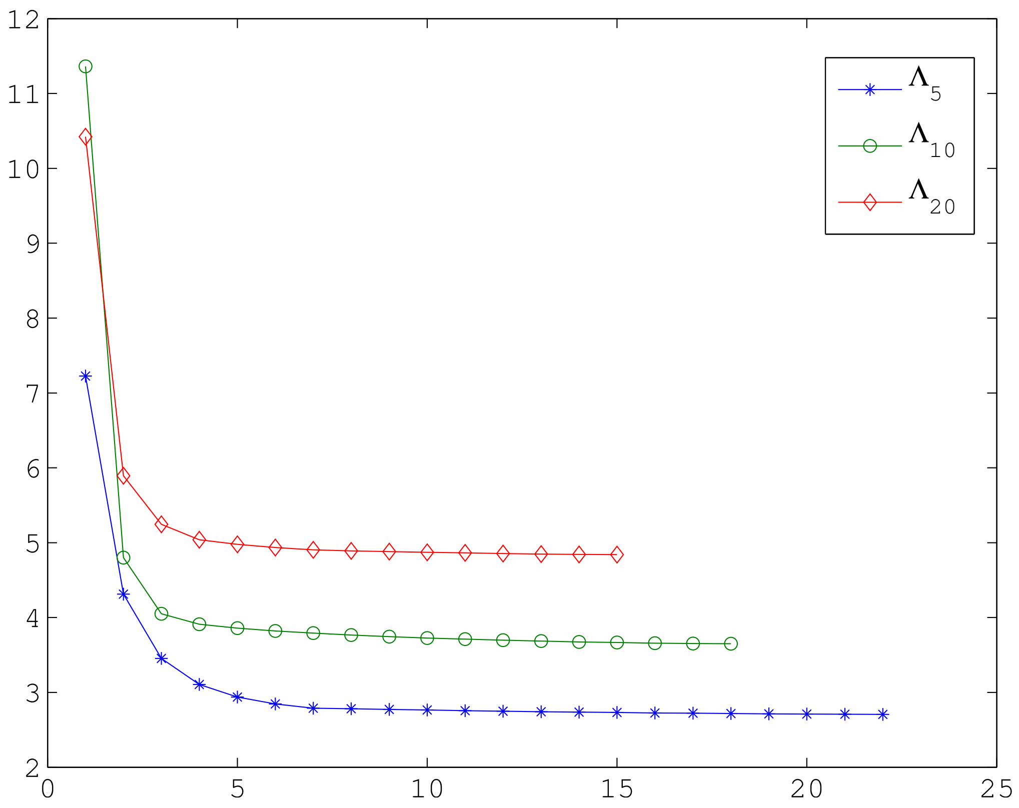

Figure 4: Λ_δ in each iteration step of the algorithm for degrees $\delta = 5, 10$ and 20.

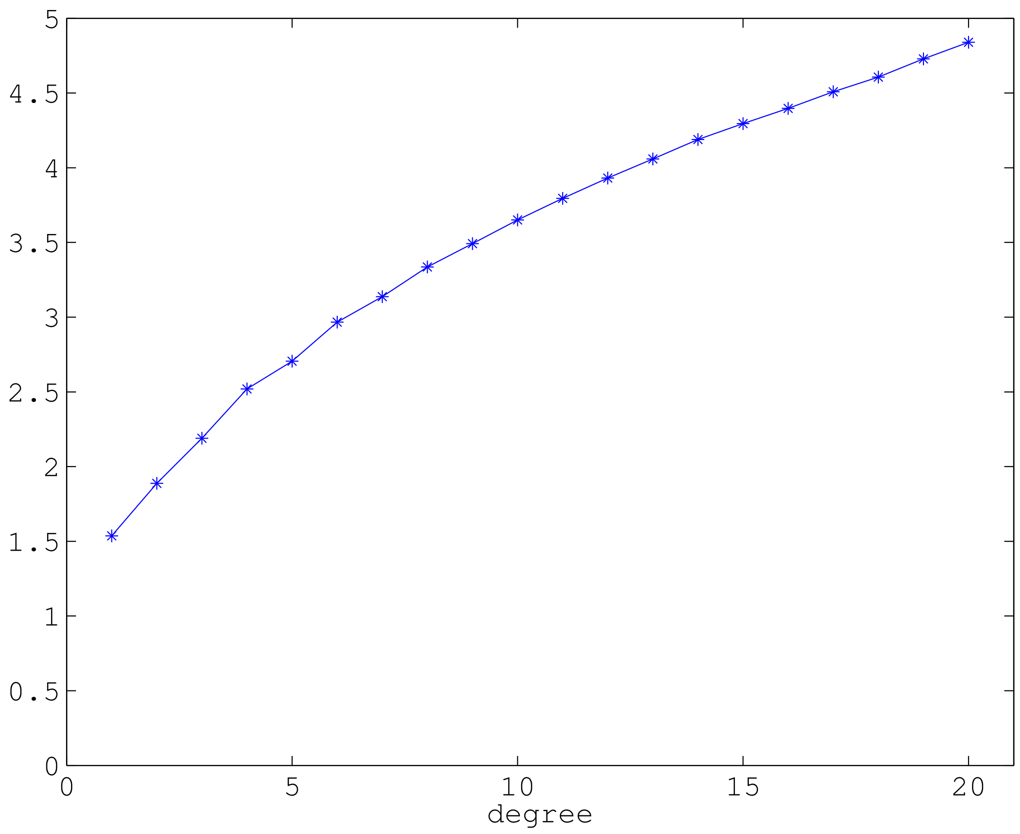

Figure 5: Λ_δ reached at the end of the algorithm in function of the degree δ.

and c is a small integer constant.

- The weights W_δ are the resulting weights of Algorithm 3.1 corresponding to $\mathcal{X}_\delta$.
- The corresponding Lebesgue constant Λ_δ for discrete least squares approximation over $\mathcal{P}_\delta$ grows slowly with respect to δ. From our numerical experiments we observe that on average it grows slower than δ.

We explain the algorithm for a polygon domain Ω. Later we consider non-polygon domains. As an example, we consider the L-shape polygon defined by the nodes

$$\mathcal{G} = \{(-1,-1),(-1,1),(0,1),(0,0),(1,0),(1,-1)\}.$$

The dimension N_δ of $\mathcal{P}_\delta$ satisfies

$$N_\delta = \binom{\delta+2}{2} = N_{\delta-1} + n_{\delta-1}, \quad \text{with} \quad n_{\delta-1} = \delta + 1.$$

Since the point sets are nested and

$$|\mathcal{X}_\delta| = \alpha N_\delta + c = |\mathcal{X}_{\delta-1}| + \alpha n_{\delta-1},$$

it follows that $\mathcal{X}_\delta$ is constructed by adding $\alpha n_{\delta-1}$ additional points to $\mathcal{X}_{\delta-1}$, i.e.,

$$\mathcal{X}_\delta = \mathcal{X}_{\delta-1} \cup \mathcal{X}_{\delta-1}^{\text{add}}, \quad \text{with} \quad |\mathcal{X}_{\delta-1}^{\text{add}}| = \alpha n_{\delta-1}.$$

These additional $\alpha n_{\delta-1}$ points $\mathcal{X}_{\delta-1}^{\text{add}}$ are chosen one by one from a larger point set $\mathcal{X}_{\delta-1}^{\text{ext}}$. This point set $\mathcal{X}_{\delta-1}^{\text{ext}}$ is constructed by Delaunay triangulation of the point set $\mathcal{X}_{\delta-1}$ and refining the triangles by dividing each edge in γ equal parts and removing the points $\mathcal{X}_{\delta-1}$. In Figure 6 we show the points $\mathcal{X}_{10}$ with the triangulation and refinement for $\gamma = 4$ and $\alpha = 2$.

The first new point is chosen as follows. The Lebesgue function is computed for discrete least squares approximation over $\mathcal{P}_{\delta-1}$ using the point set $\mathcal{X}_{\delta-1}$ and corresponding weights $\mathcal{W}_{\delta-1}$. The point of $\mathcal{X}_{\delta-1}^{\text{ext}}$ where the Lebesgue function is largest is added to $\mathcal{X}_{\delta-1}$, with the corresponding weight equal to the maximum of the weights $\mathcal{W}_{\delta-1}$. The Lebesgue function is updated and in the same way the other points are added one by one to form $\mathcal{X}_\delta$. Once the point set $\mathcal{X}_\delta$ is constructed, we compute the corresponding weights $\mathcal{W}_\delta$ by using Algorithm 3.1.

At the start of the algorithm the point set $\mathcal{X}_1$ is needed. This point set consists of the points $\mathcal{G}$ defining the polygon and some possible additional points. The additional points guarantee that the refined point set $\mathcal{X}_1^{\text{ext}}$ is large enough for the algorithm to start. In Section 5 we specify the point set $\mathcal{X}_1$ for the other geometries, i.e., the square, the triangle and the disk. The corresponding weights $\mathcal{W}_1$ are computed by Algorithm 3.1.

The refined mesh $\mathcal{X}_{\delta-1}^{\text{ext}}$ is constructed as follows:

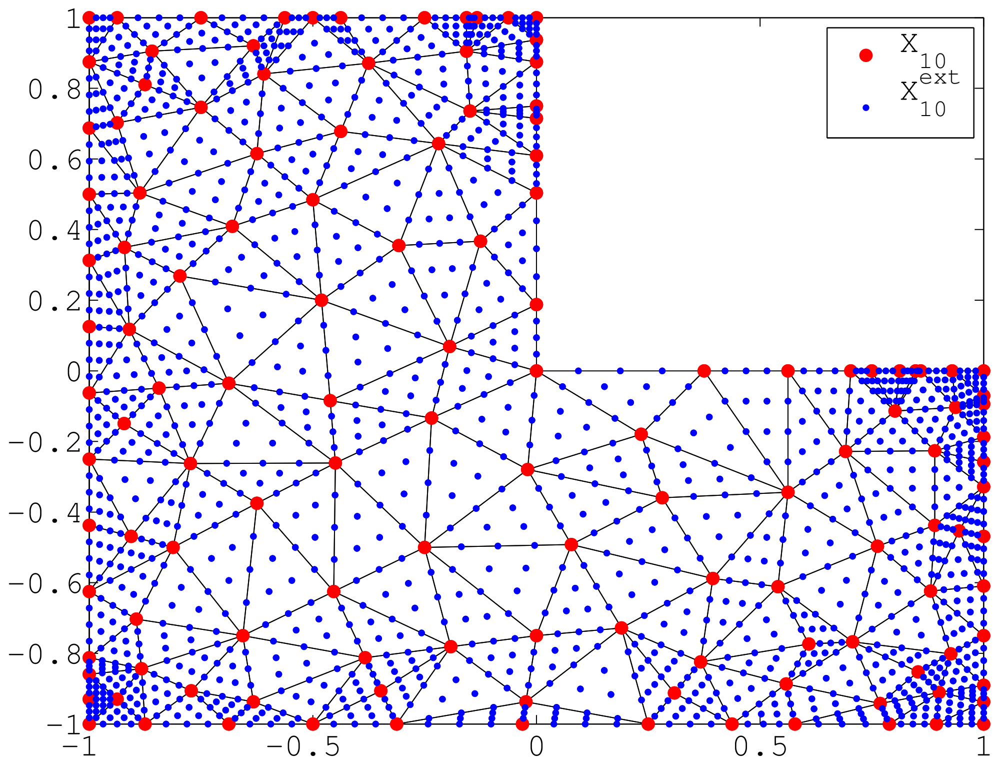

Figure 6: The points $\mathcal{X}_{10}$ and the refined mesh $\mathcal{X}_{10}^{\text{ext}}$ for $\gamma = 4$ and $\alpha = 2$.

- For a convex polygon domain it suffices to use standard Delaunay triangulation. If the polygon domain is non-convex, then constrained Delaunay triangulation is used, where the boundary of the domain defines the constraint. This method makes sure that only triangles inside the domain are generated, and it is available in MATLAB.
- For convex non-polygon domains, e.g., the disk, using Delaunay triangulation alone is not enough, because the boundary of such a domain lies outside the convex hull of the point set $\mathcal{X}_{\delta-1}$. To overcome this problem for the disk, for each degree we generate a boundary set of equispaced points on the boundary that is added to $\mathcal{X}_{\delta-1}^{\text{ext}}$. The number of points on the boundary is taken equal to $\gamma N_{\delta-1}$. Experimentally we observed that the resulting sets $\mathcal{X}_{\delta-1}$ constructed by Algorithm 4.1 have a large fraction ϕ of points on the boundary. It follows that $\mathcal{X}_{\delta-1} \cup \mathcal{X}_{\delta-1}^{\text{ext}}$ contains about $\phi\gamma N_{\delta-1}$ points on the edges of the triangles near the boundary. Hence the cardinality of our boundary set is such that it has about the same scale of refinement.
- For non-convex non-polygon domains a similar method can be used. After creating a fine mesh based on Delaunay triangulation, first the points that lie outside of the domain are removed, and second a boundary set of equispaced points on the boundary is added.

The final method is described as Algorithm 4.1 and numerical experiments are given in Section 5.

Algorithm 4.1 Good point sets and corresponding weights for discrete least squares approximation

```
Input: geometry $\Omega$, $\mathcal{X}_1 \subset \Omega$, $\alpha$, $\gamma$, $\delta_{\max}$
Output: point sets $\mathcal{X}_i$ and corresponding weights $\mathcal{W}_i$, $i = 1, 2, \ldots, \delta_{\max}$
  Compute weights $\mathcal{W}_1$ corresponding to $\mathcal{X}_1$ using Algorithm 3.1
  for $\delta = 2, \ldots, \delta_{\max}$ do
      Triangulate $\mathcal{X}_{\delta-1}$
      Generate the set $\mathcal{X}_{\delta-1}^{\text{ext}}$ by dividing the triangle edges in $\gamma$ equal parts
      Extend $\mathcal{X}_{\delta-1}$ to $\mathcal{X}_\delta$ by adding $\alpha n_{\delta-1}$ points from $\mathcal{X}_{\delta-1}^{ext}$.
          - The points are added one by one where the Lebesgue function $\lambda_{\delta-1}$
            is largest.
          - The Lebesgue function is updated after each point is added.
      Compute weights $\mathcal{W}_\delta$ corresponding to $\mathcal{X}_\delta$ using Algorithm 3.1
  end for
```

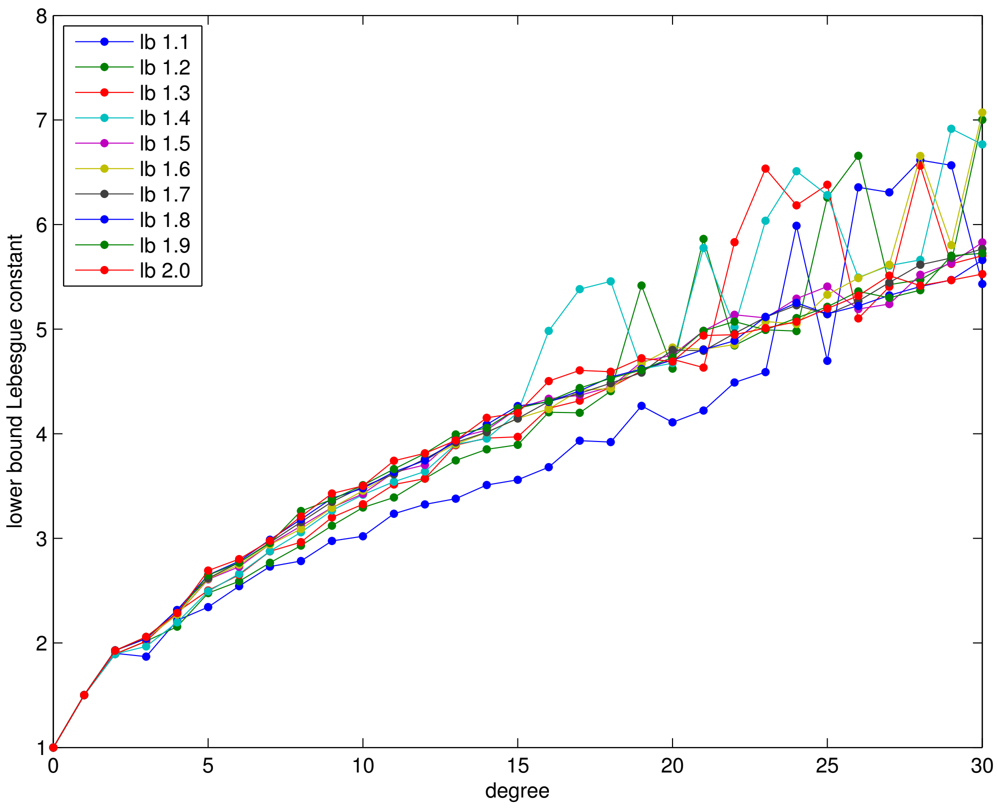

Figure 7: Lower bound of the Lebesgue constant for $\alpha = 1.1, 1.2, \ldots, 1.9, 2.0$.

5 Numerical experiments

This section contains results of Algorithm 4.1 for the following geometries: the square, the simplex, the disk and the L-shape. For each geometry a suitable point set $\mathcal{X}_1$ is created. This point set must have the property that the refined point set $\mathcal{X}_1^{\text{ext}}$ from which points are added to $\mathcal{X}_1$ to obtain $\mathcal{X}_2$, is large enough. Besides the original 4 points and 3 points of $\mathcal{G}$ defining the geometry of the square and the triangle, respectively, the Delaunay triangulation is taken and refined by dividing all edges in 2 equal parts. For the L-shape, it suffices to take the 6 points defining the geometry as $\mathcal{X}_1$. For the unit disk, we take the following points for $\mathcal{X}_1$: on the x-axis, $(-1,0), (-0.5,0), (0,0), (0.5,0), (1,0)$, and on the y-axis, $(0,-1), (0,-0.5), (0,0.5), (0,1)$.

The values of the parameters are $\delta_{\max} = 30$ and $\gamma = 4$ and we have considered the following values for α: $1.1, 1.2, \ldots, 1.9, 2.0, 2.5, 3.0, \ldots, 5.5, 6.0$. Remember that γ determines the resolution of the refined point set $\mathcal{X}_\delta^{\text{ext}}$ and α determines the number of points of $\mathcal{X}_\delta$ compared to the dimension of $\mathcal{P}_\delta$.

A lower bound for the Lebesgue constant Λ_δ is obtained when computing good weights $\mathcal{W}_\delta$ corresponding to $\mathcal{X}_\delta$, i.e., $\Lambda_\delta \geq \max_{x \in \mathcal{X}_\delta} \lambda_\delta(\boldsymbol{x})$. For the square, this lower bound is plotted in Figure 7 for the values of $\alpha = 1.1, 1.2, \ldots, 1.9, 2.0$. Note the non-steady behaviour for $\alpha = 1.1, 1.2, \ldots, 1.6$. An estimate for the Lebesgue constant for the values of $\alpha = 1.1, 1.2, \ldots, 1.9, 2.0$ is given in Figure 8. We use the estimate $\Lambda_\delta \approx \max_{x \in \mathcal{Y}} \lambda_\delta(\boldsymbol{x})$, where $\mathcal{Y}$ is the result of our Delaunay refinement dividing the edges in 7 equal parts. For the values of $\alpha = 1.7, 1.8, 1.9, 2.0$, the lower bound as well as the estimate are plotted in Figure 9. A similar plot for values of $\alpha = 2.0, 2.5, 3.0, 4.0, 5.0, 6.0$ is given in Figure 10. From Figure 8 we observe that the Lebesgue constant decreases for increasing values of α. This effect is more pronounced if α is close to 1. Figure 9 shows that there is a gap between the lower bound and the more accurate approximation of the Lebesgue constant. By increasing α, this gap decreases as we can see in Figure 10.

The time for adding point set $\mathcal{X}_\delta$ behaves as $\mathcal{O}(\delta^5)$ as illustrated in Figure 11.

We get similar plots for the other geometries, i.e., for the triangle, the L-shape and the disk.

In Figure 12 we show the point sets $\mathcal{X}_{30}$, containing 995, 992, 992 and 995 points, respectively, for the square, the triangle, the L-shape and the disk, with $\alpha = 2.0$. In Figure 13 the magnitude of the weights corresponding to the square are shown. For the other geometries, the magnitude of the weights is very similar.

6 Conclusion

In this paper, we have developed an algorithm to compute nested point sets and corresponding weights having a small Lebesgue constant for the discrete least squares approximation for a given geometry. Compared to the number of points N_δ needed for interpolation with a polynomial of total degree δ, the number of points in the point set for least squares approximation by a polynomial of total degree δ is $\alpha N_\delta + c$ with $\alpha > 1$ and c a small constant. The numerical experiments indicate that even for

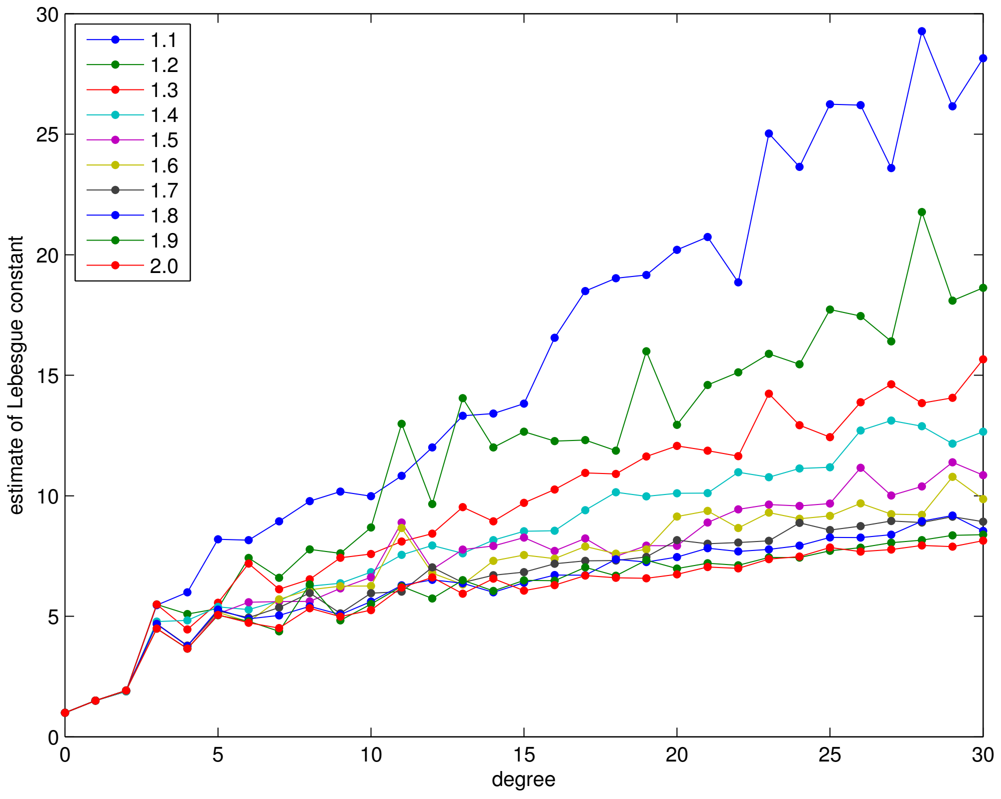

Figure 8: Estimate of the Lebesgue constant for $\alpha = 1.1, 1.2, \ldots, 1.9, 2.0$.

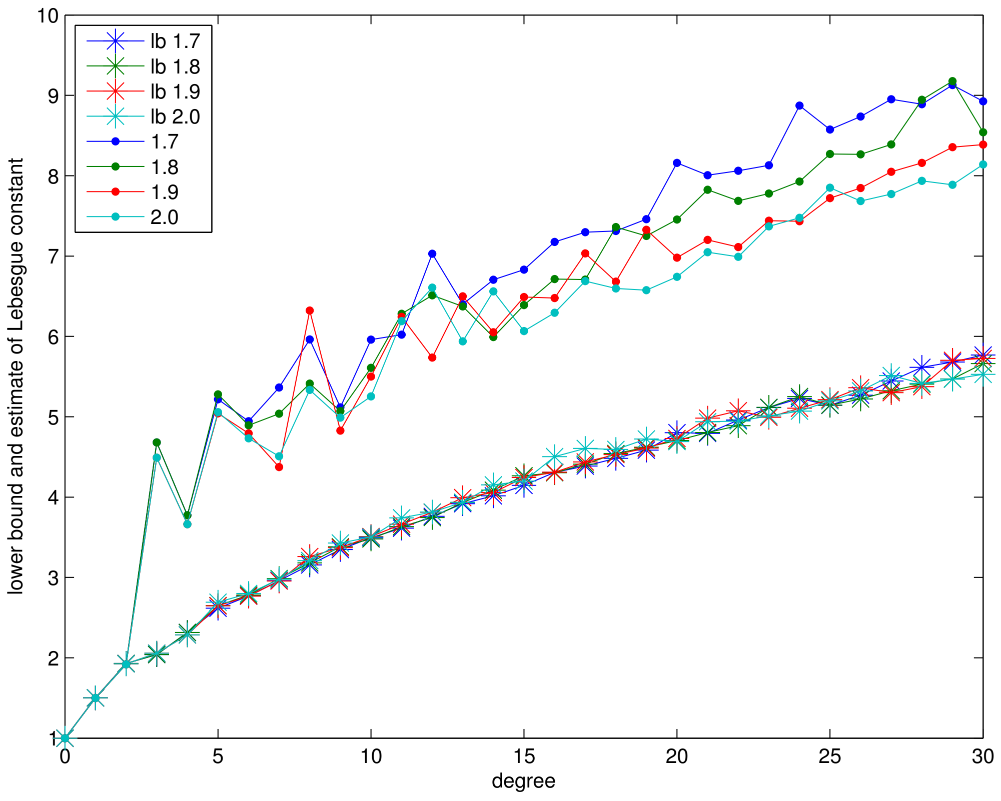

Figure 9: Lower bound and estimate of the Lebesgue constant for $\alpha = 1.7, 1.8, 1.9, 2.0$.

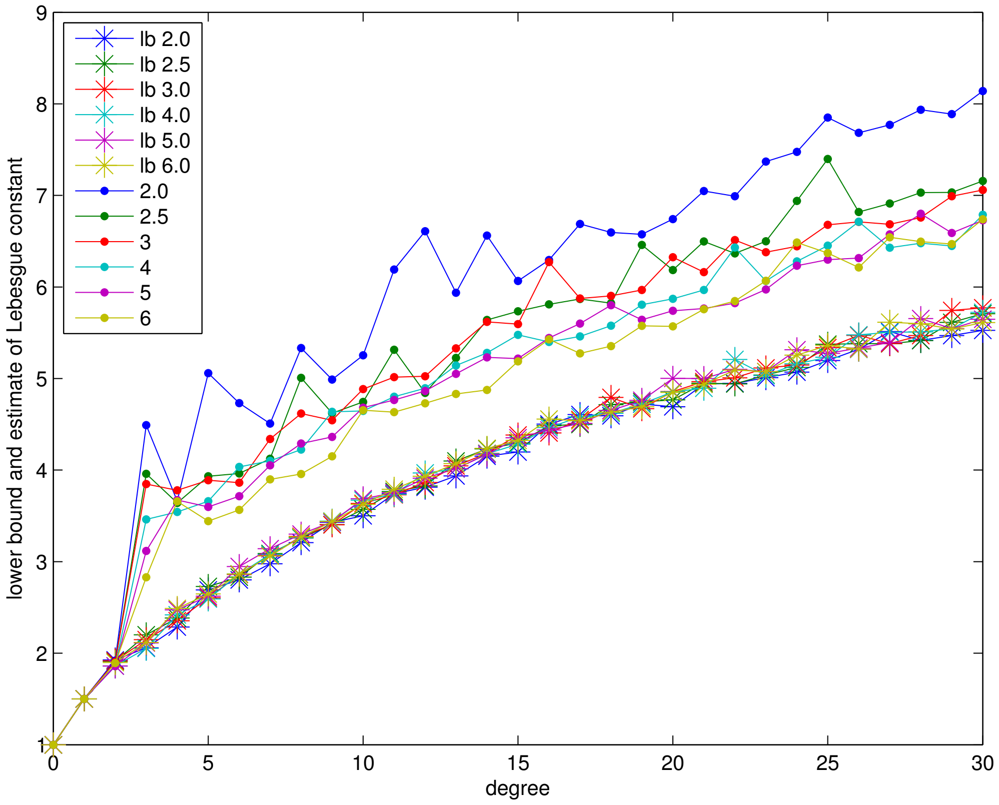

Figure 10: Lower bound and estimate of the Lebesgue constant for $\alpha = 2.0, 2.5, 3.0, 4.0, 5.0, 6.0$.

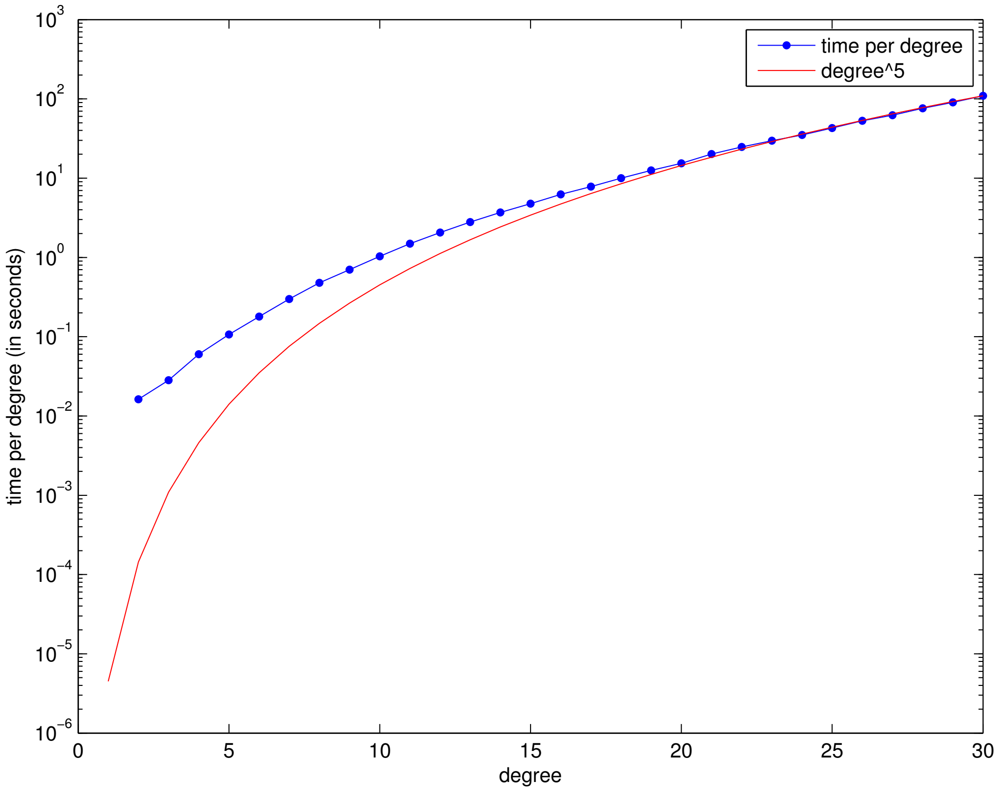

Figure 11: Time in function of degree δ for $\alpha = 2.0$.

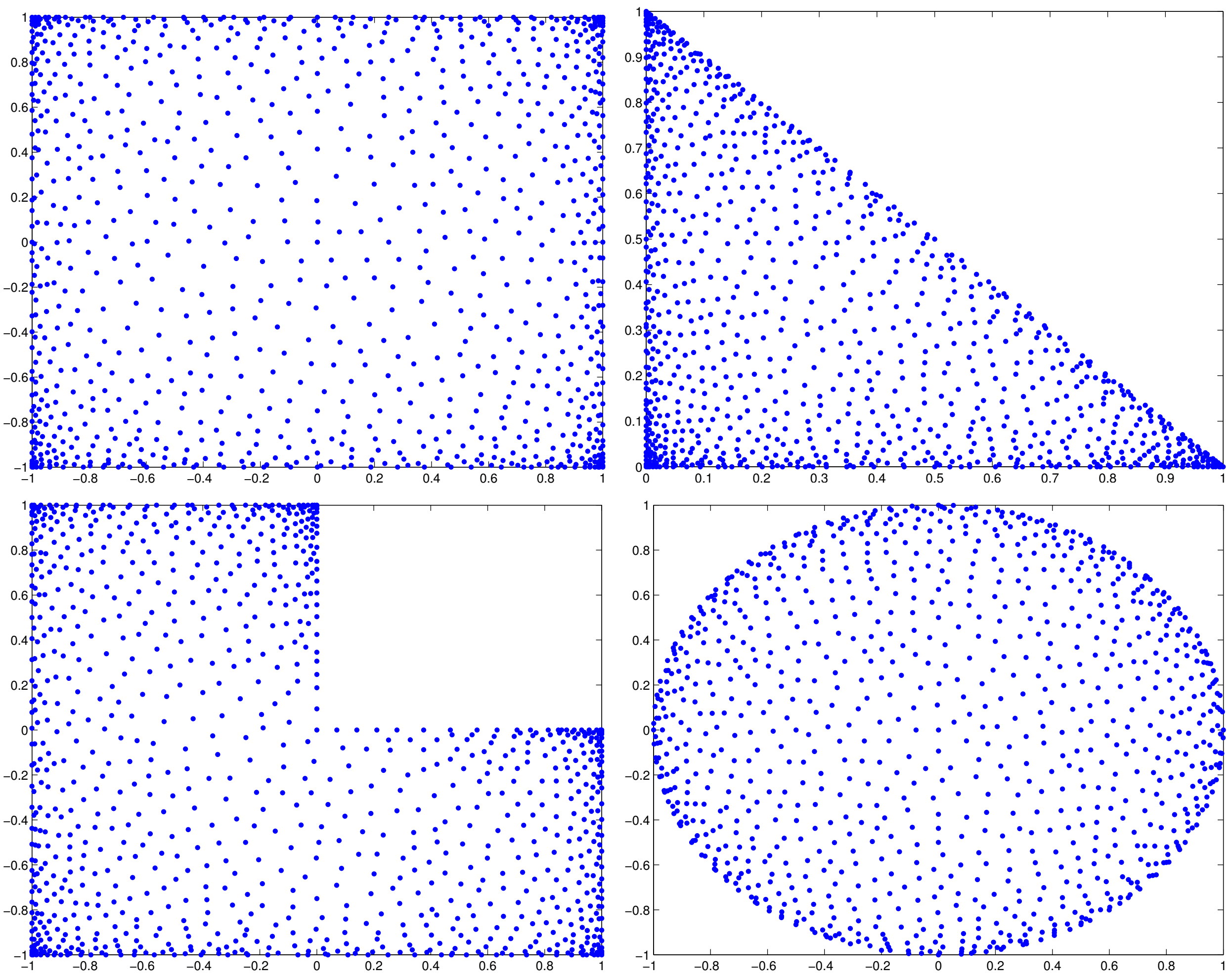

Figure 12: Point sets $\mathcal{X}_{30}$ for the square, the triangle, the L-shape and the disk, for $\alpha = 2.0$, containing $995, 992, 992$ and 995 points, respectively. The estimated Lebesgue constants are equal to 8.14, 8.96, 8.56 and 9.23, respectively.

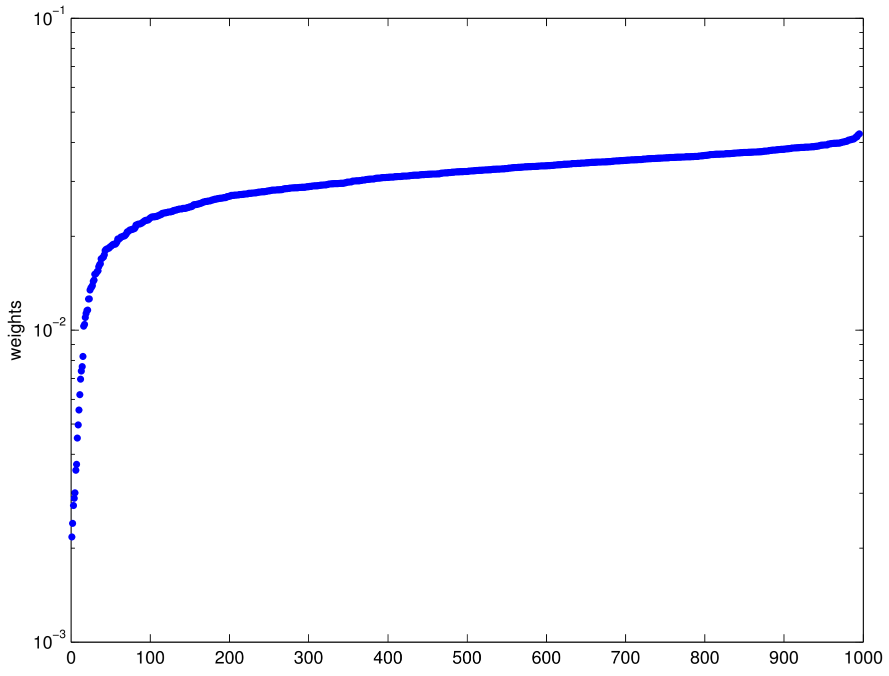

Figure 13: Weights $\mathcal{W}_{30}$ for the square domain with $\alpha = 2.0$ containing 995 points.

small values of α we obtain point sets with reasonably low Lebesgue constant and this for several geometries, i.e., the square, the triangle, the L-shape and the disk. E.g., for $\alpha = 2$, the Lebesgue constant grows less than polynomial with respect to the degree for each of these geometries. Hence, empirically, the point sets that we compute corresponding to each of the geometries considered, are the first sets in a sequence of point sets forming a Weakly Admissible Mesh for the corresponding geometry.

References

[1] L. Bos, M. Caliari, S. De Marchi, M. Vianello, and Y. Xu. Bivariate Lagrange interpolation at the Padua points: the generating curve approach. *Journal of Approximation Theory*, 143:15–25, 2006.

[2] L. Bos, J.-P. Calvi, N. Levenberg, A. Sommariva, and M. Vianello. Geometric weakly admissible meshes, discrete least squares approximations and approximate Fekete points. *Mathematics of Computation*, 80(275):1623–1638, 2011.

[3] L. Bos, S. De Marchi, A. Sommariva, and M. Vianello. Computing multivariate Fekete and Leja points by numerical linear algebra. *SIAM Journal on Numerical Analysis*, 48(5):1984–1999, 2010.

[4] L. Bos, S. De Marchi, A. Sommariva, and M. Vianello. Weakly admissible meshes and discrete extremal sets. *Numerical Mathematics: Theory, methods and applications*, 4:1–12, 2011.

[5] L. Bos, S. De Marchi, M. Vianello, and Y. Xu. Bivariate Lagrange interpolation at the Padua points: the ideal theory approach. *Numerische Mathematik*, 108:43–57, 2007.

[6] L. Bos, A. Sommariva, and M. Vianello. Least-squares polynomial approximation on weakly admissible meshes: Disk and triangle. *Journal of Computational and Applied Mathematics*, 235(3):660–668, 2010.

[7] L. Bos and M. Vianello. Low cardinality admissible meshes on quadrangles, triangles and disks. *Mathematical Inequalities and Applications*, 15(1):229–235, 2012.

[8] M. Briani, A. Sommariva, and M. Vianello. Computing Fekete and Lebesgue points: Simplex, square, disk. *Journal of Computational and Applied Mathematics*, 236:2477–2486, 2012. Not yet available on JCAM website...

[9] M. Caliari, S. De Marchi, and M. Vianello. Bivariate polynomial interpolation on the square at new nodal sets. *Applied Mathematics and Computation*, 165(2):261–274, 2005.

[10] J.-P. Calvi and N. Levenberg. Uniform approximation by discrete least squares polynomials. *Journal of Approximation Theory*, 152:82–100, 2008.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, USA, third edition, 1996.

[12] A. Sommariva and M. Vianello. Computing approximate Fekete points by QR factorizations of Vandermonde matrices. *Computers & Mathematics with Applications*, 57(8):1324–1336, 2009.

[13] L. N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2012.

[14] M. Van Barel and A. A. Chesnokov. A method to compute recurrence relation coefficients for bivariate orthogonal polynomials by unitary matrix transformations. *Numerical Algorithms*, 55:383–402, 2010.

[15] M. Van Barel, M. Humet, and L. Sorber. Approximating optimal point configurations for multivariate polynomial interpolation. *ETNA*, 42:41–63, 2014.