The computation of sets of points with small Lebesgue constant using bivariate orthogonal polynomials

Gérard MEURANT

September 2014

Introduction

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with boundary $\partial\Omega$ and f be a function defined on $\bar{\Omega}$

We are interested in interpolating f on $\bar{\Omega}$ with bivariate polynomials of total degree d

Let $X = \{\xi_i\}_{i=1}^N = \{(x_i, y_i)\}_{i=1}^N$ with $N = (d+1)(d+2)/2$ be a unisolvent set of points in $\bar{\Omega}$

The N Lagrange polynomials ℓ_j related to these points are such that ℓ_j is equal to 1 at point ξ_j and zero at the other points
The interpolating polynomial is

$$p(\xi) = \sum_{j=1}^{N} f(\xi_j)\ell_j(\xi)$$

Interpolation error

We have

$$\|f - p\|_\infty \leq (1 + \lambda_X)\|f - p^*\|_\infty$$

where p^* is the polynomial of best approximation to f in the ∞-norm and

$$\lambda_X = \max_{\xi \in \bar{\Omega}} \Lambda_X(\xi)$$

is the Lebesgue constant,

$$\Lambda_X(\xi) = \sum_{i=1}^{N} |\ell_i(\xi)|$$

is the Lebesgue function

Hence, it makes sense to try to choose the set of points X to minimize the Lebesgue constant or, at least, to have a small Lebesgue constant

However, the problem

$$\min_{X} \max_{\xi \in \bar{\Omega}} \sum_{i=1}^{N} |\ell_i(\xi)|$$

is difficult to solve

- The function is not differentiable
- The Lebesgue function is oscillating and the max difficult to compute

Note that the number of unknowns is $2N$

Bivariate orthogonal polynomials

A way to compute the Lagrange polynomials is to use the (discrete) orthogonal polynomials associated with X

$$\langle \varphi_i, \varphi_j \rangle = \frac{1}{N} \sum_{k=1}^{N} \varphi_i(\xi_k)\varphi_j(\xi_k) = \delta_{i,j}$$

Then

$$\ell_j(\xi) = \frac{1}{N} \sum_{k=1}^{N} \varphi_k(\xi_j)\varphi_k(\xi)$$

Currently we have two ways to compute the orthogonal polynomials

- The method of Van Barel & al which is an extension of what was done by Gragg and Harrod, Reichel for the univariate case
- The Lanczos-like method from Huhtanen and Larsen

The second method suffers more from rounding errors but
A. Sommariva proposed to use it with a double reorthogonalization

Even in this case, it is much faster than the other method

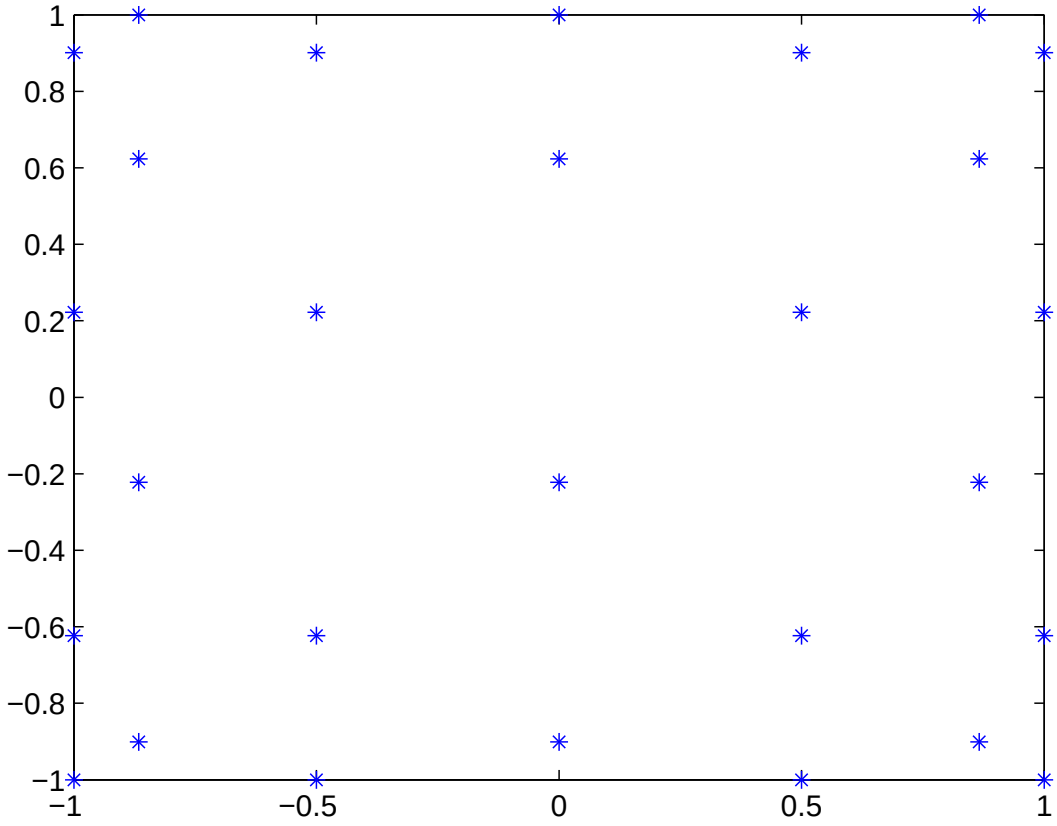
We choose an ordering of the monomials $x^i y^j$

We start from a vector of unit norm with components $1/\sqrt{N}$. Let D_x and D_y be the diagonal matrices corresponding to the coordinates of the points

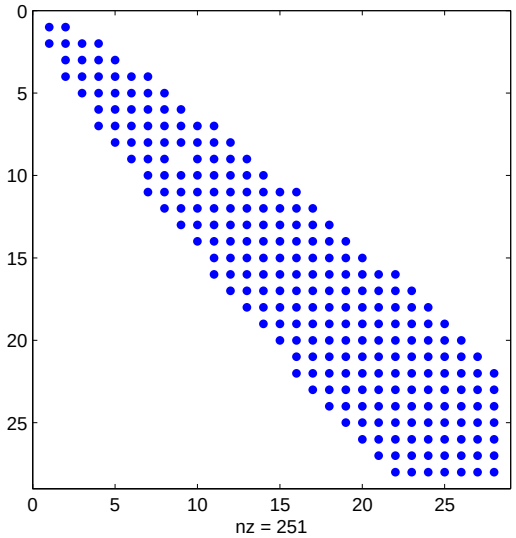
We multiply successively by D_x or D_y to obtain the monomials in the order we need and we orthogonalize with respect to the previous vectors

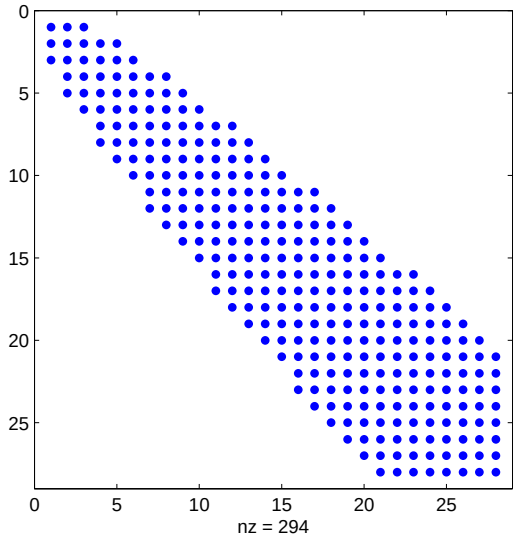
Since we generally loose orthogonality we double reorthogonalize at the end

From this we obtain the recursion coefficients for the orthogonal polynomials

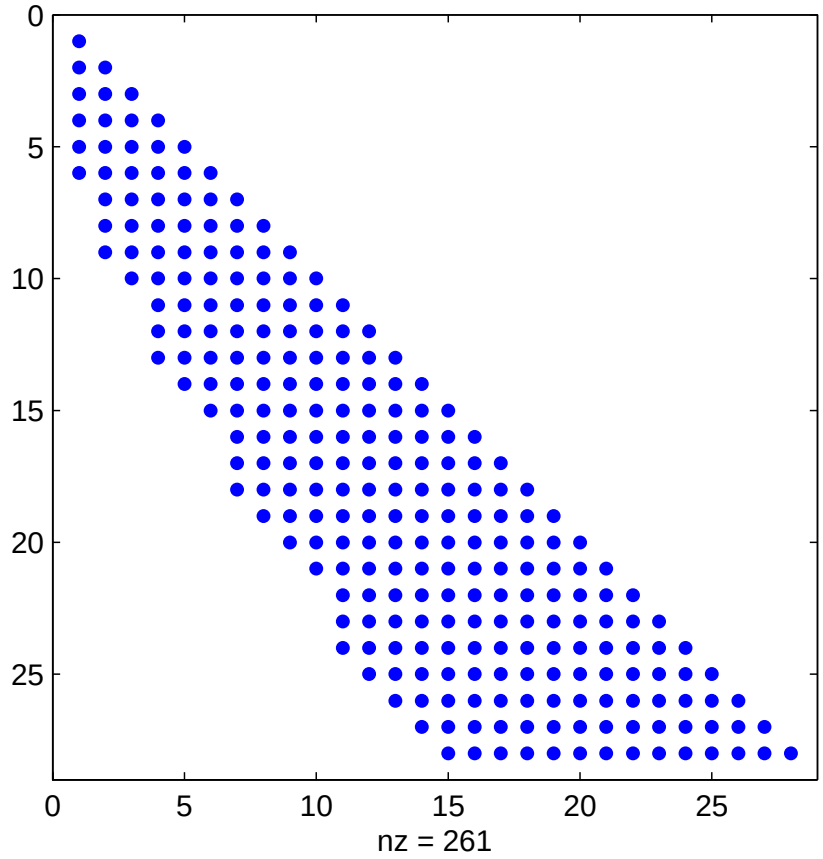


Square: Padua points for $d = 6$





Square: Padua points for $d = 6$, multiplication matrices for x and y



Square: Padua points for $d = 6$, matrix of the recurrence coefficients

Sets of points with a small Lebesgue constant

Even though the minimization of the Lebesgue constant can be done it is rather expensive for large degrees

Instead we propose to minimize the following functional

$$\min_{X,\,\xi_j\in\bar{\Omega}} I_\Omega, \quad I_\Omega = \left\{ \int_\Omega [\ell_1(x,y)^2 + \cdots + \ell_N(x,y)^2]\, dxdy \right\}^{\frac{1}{2}}$$

It is sometimes better to add a boundary integral and to minimize

$$\min_{X,\,\xi_j\in\bar{\Omega}} I_{\Omega,\partial\Omega}, \quad I_{\Omega,\partial\Omega} = \left\{ I_\Omega^2 + \mu \int_{\partial\Omega} [\ell_1(\xi)^2 + \cdots + \ell_N(\xi)^2]\, ds \right\}^{\frac{1}{2}}$$

where μ is a positive real number

For the minimization we use a translation to Matlab of the algorithm `praxis` proposed by R. Brent

Algorithms for Minimization without Derivatives, Prentice Hall (1973), reprinted by Dover (2002)

The values of the integrals are computed with cubature formulas proposed by Sommariva and Vianello (Padua university)

With a large enough number of nodes we can integrate exactly our polynomials

Refinement (1)

Even though the Lebesgue constants are not increasing too fast with the degree, they are not as good as those known in the literature for the unit square

Therefore, we add refinement algorithms after the minimization process inspired by what was done by Van Barel and al

For all the points in turn, we remove the point from the point set, compute the new approximate maximum of the Lebesgue function (using `praxis`) and put a point there. We iterate several times this process

Along this algorithm we keep the best distribution obtained so far that is, the one with the smallest Lebesgue constant

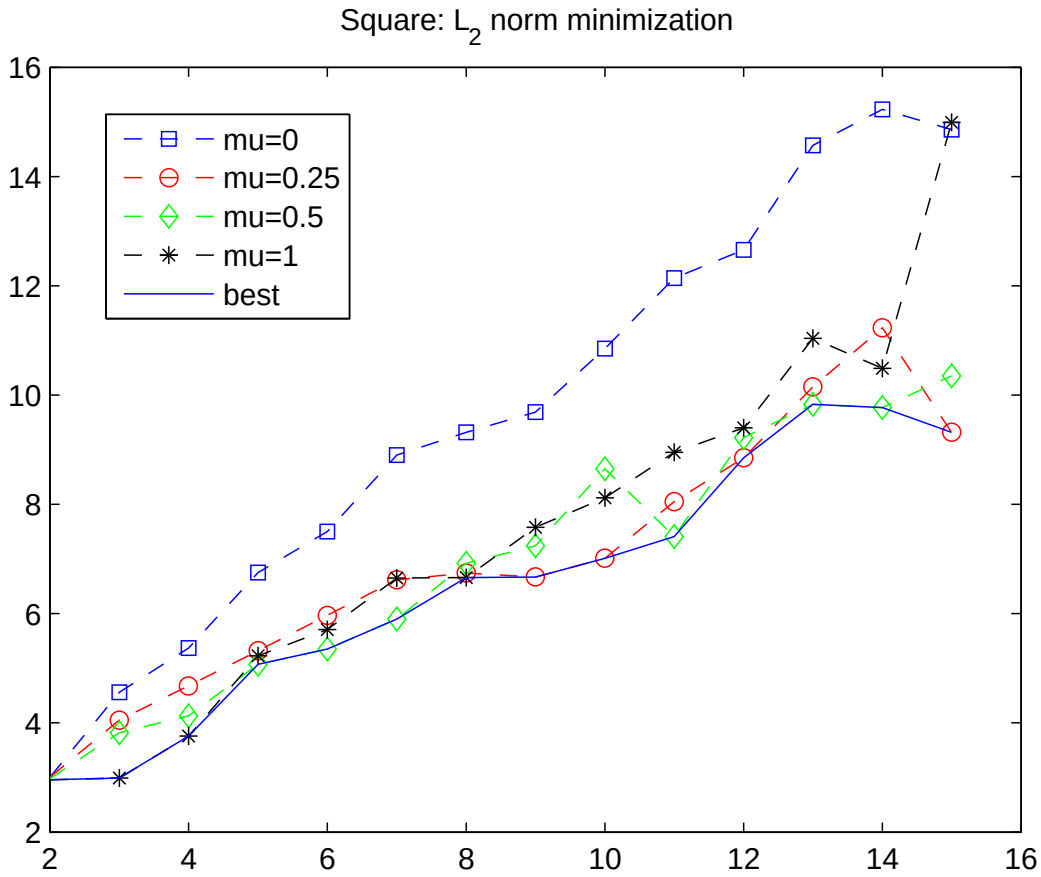
Refinement (2)

Even when using this refinement process in some cases we still have a quite large global maximum

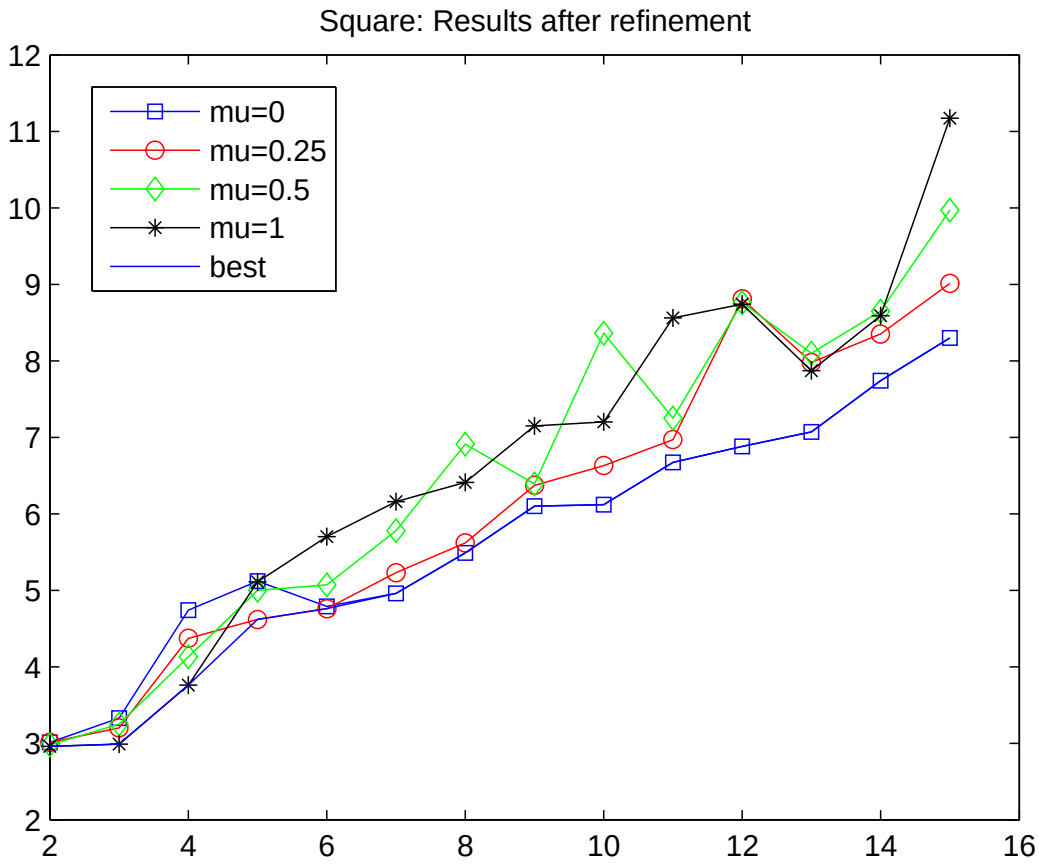
Then we look for the point closest to this maximum and minimize the Lebesgue constant as a function of the two coordinates of this point. This is done using a fine mesh to locate the maximum

We denote this as *local minimization*

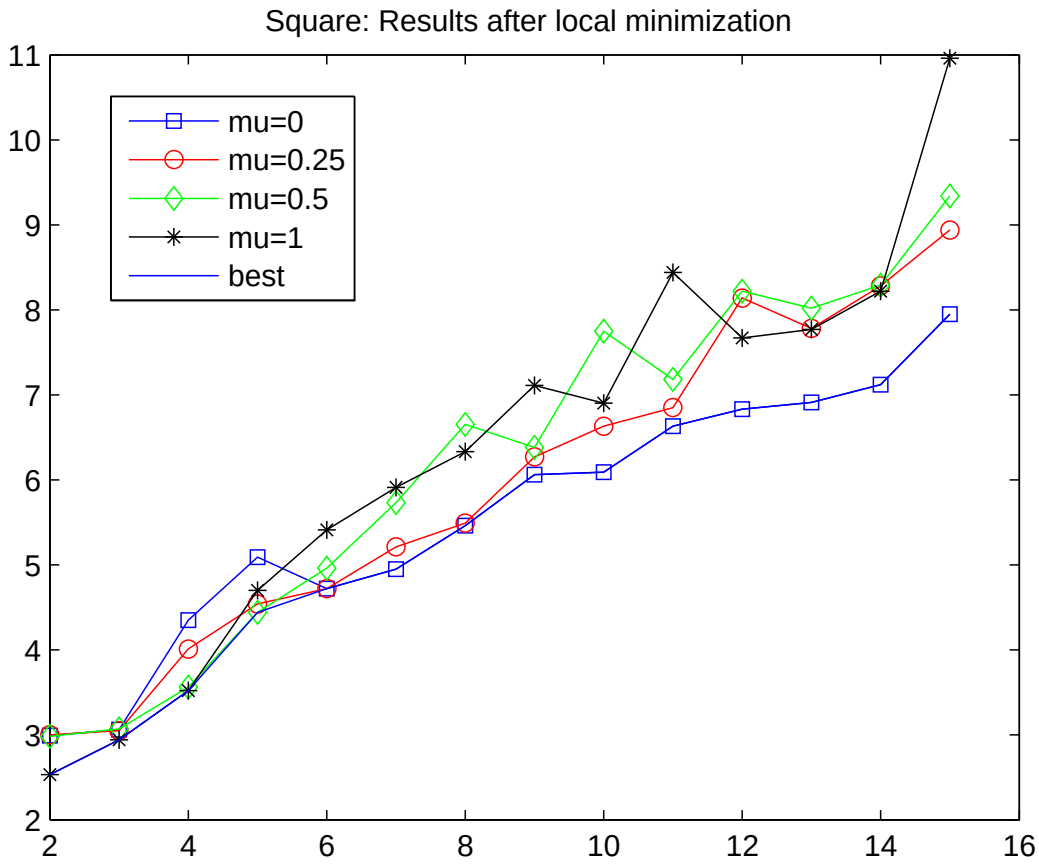
Examples



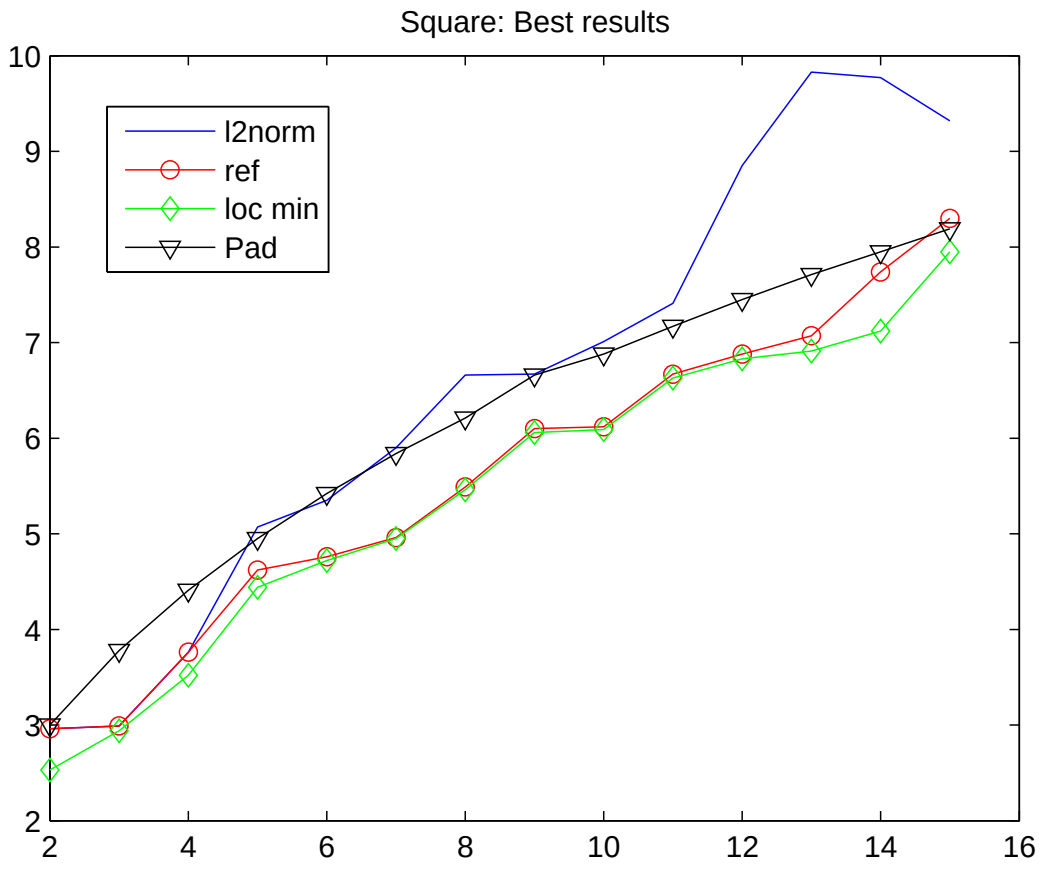
Square: Lebesgue constant with L_2-norm minimization as a function of the degree



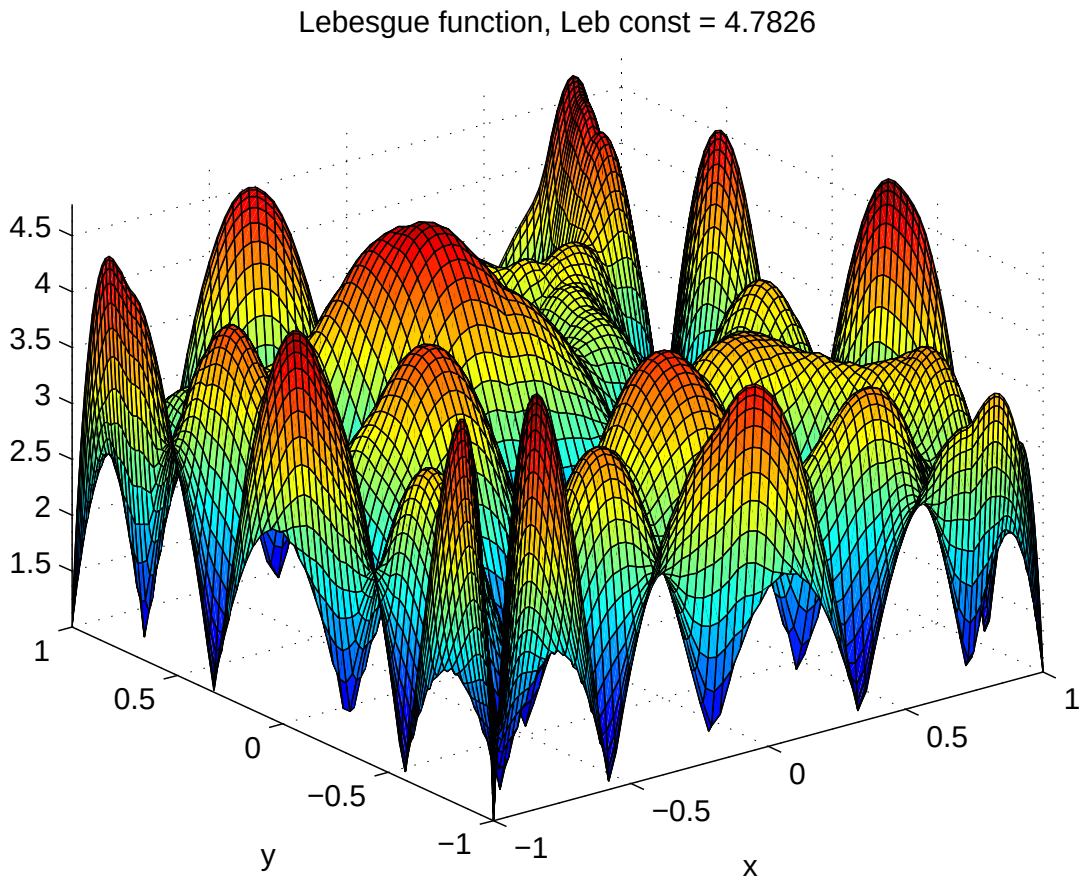
Square: Lebesgue constant with L_2-norm minimization + refinement as a function of the degree



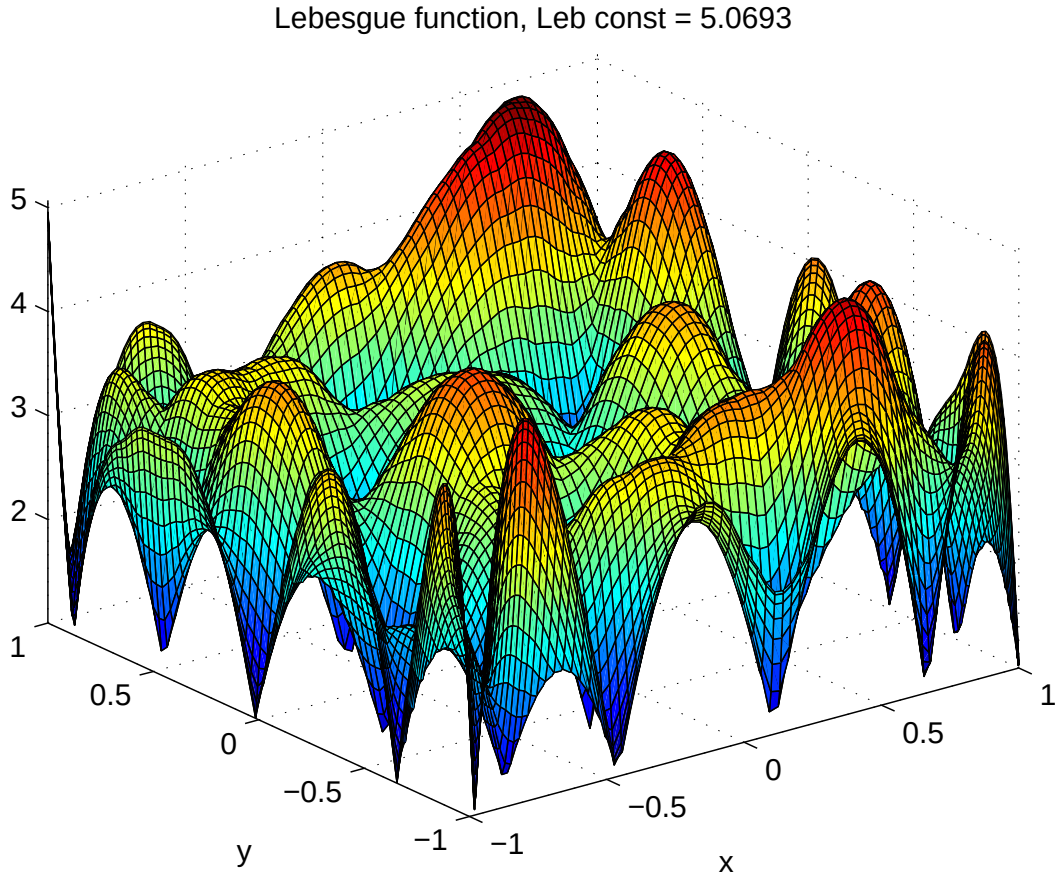
Square: Lebesgue constant with L_2-norm minimization + refinement + local min as a function of the degree



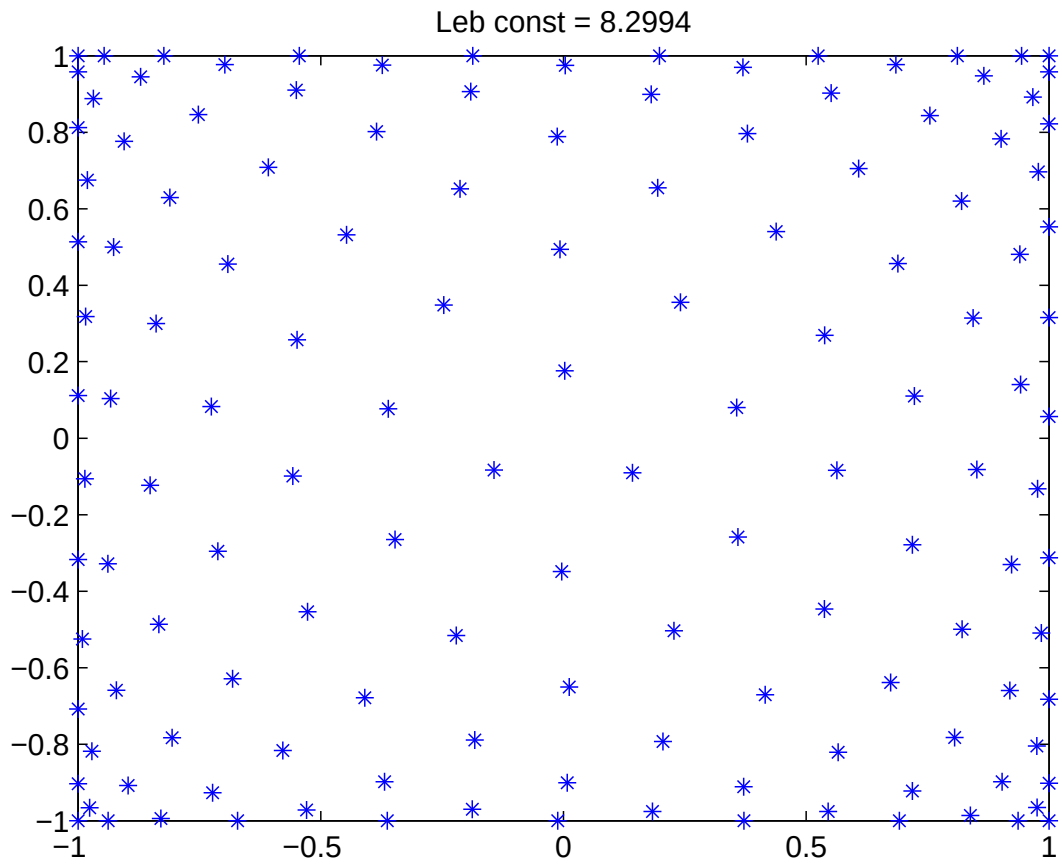
Square: Lebesgue constant, best results



Square: Lebesgue function, $d = 6$, $\mu = 0$



Square: Lebesgue function, $d = 6$, $\mu = 0.5$



Square: The distribution of points for degree 15

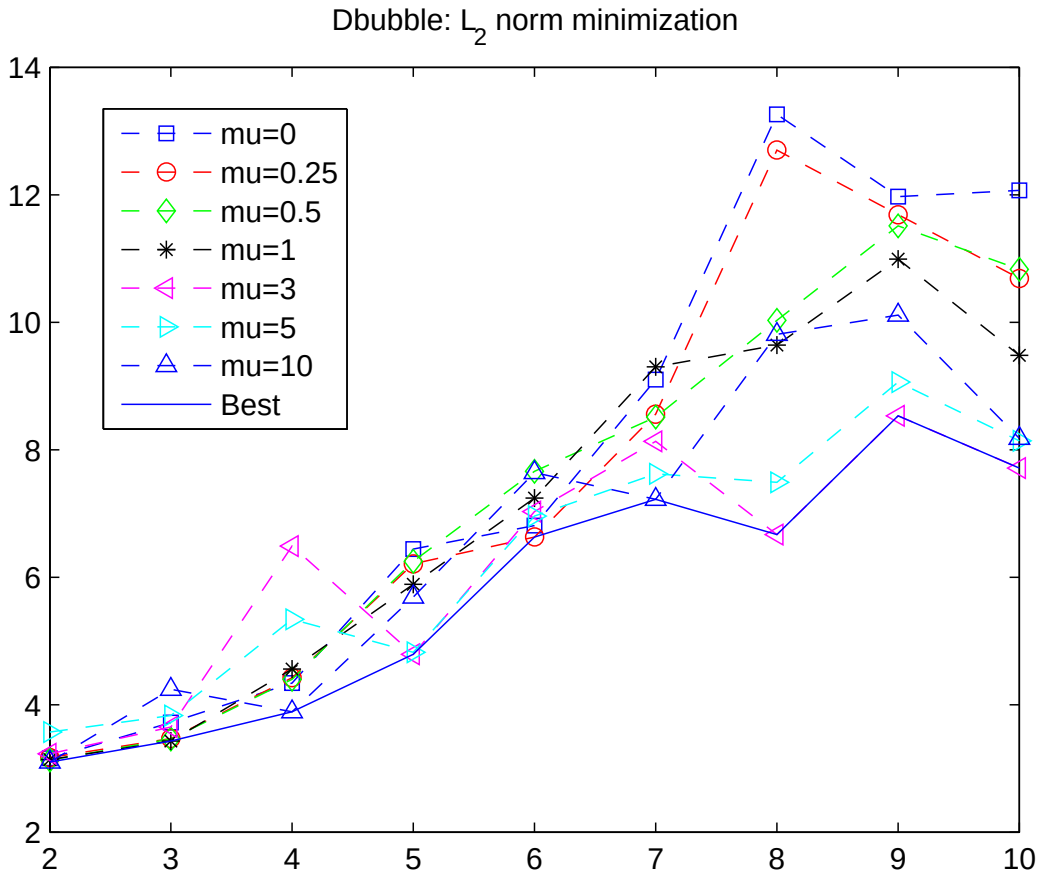
Double bubble

The domain is the union of 2 disks:

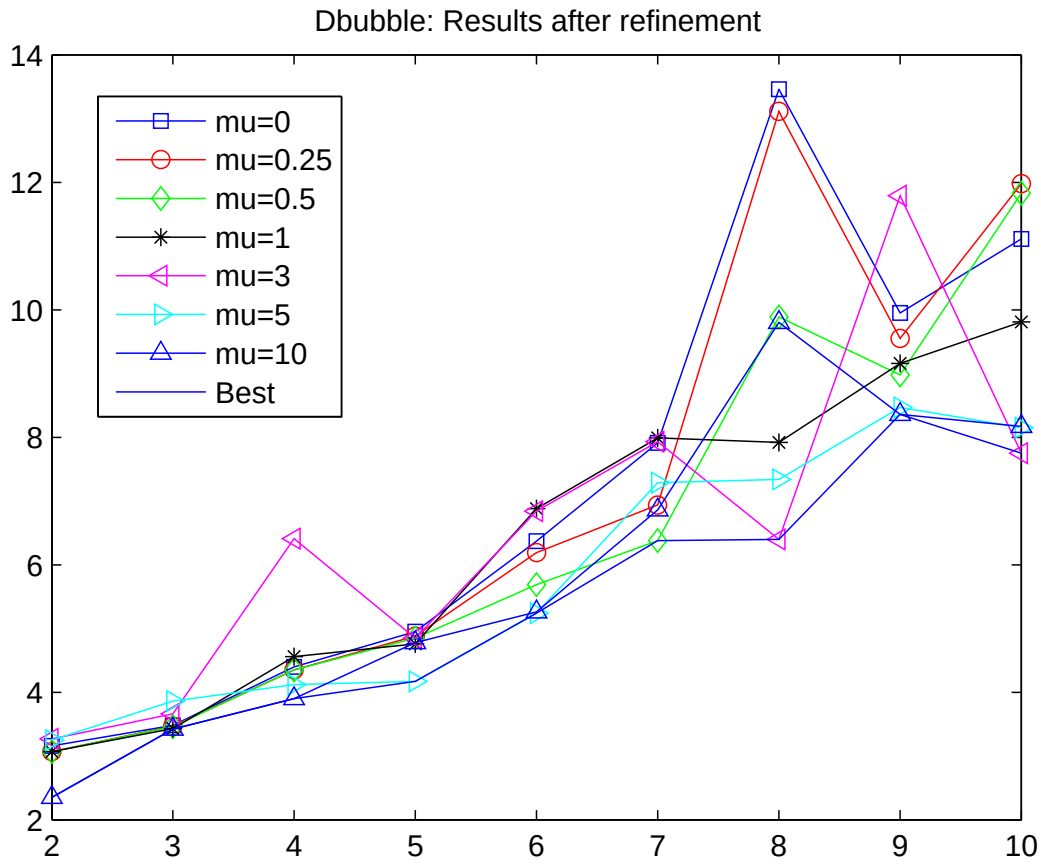
center (0,0), radius 5

center (6,0), radius 3

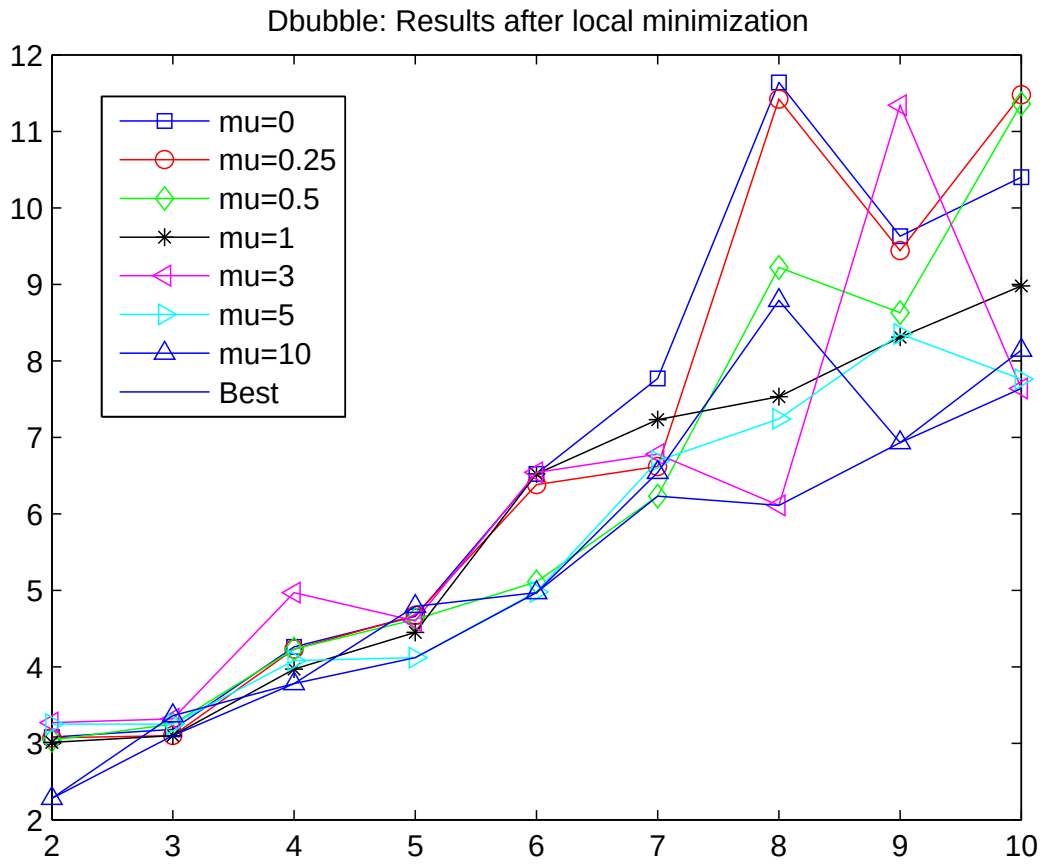
Double bubble



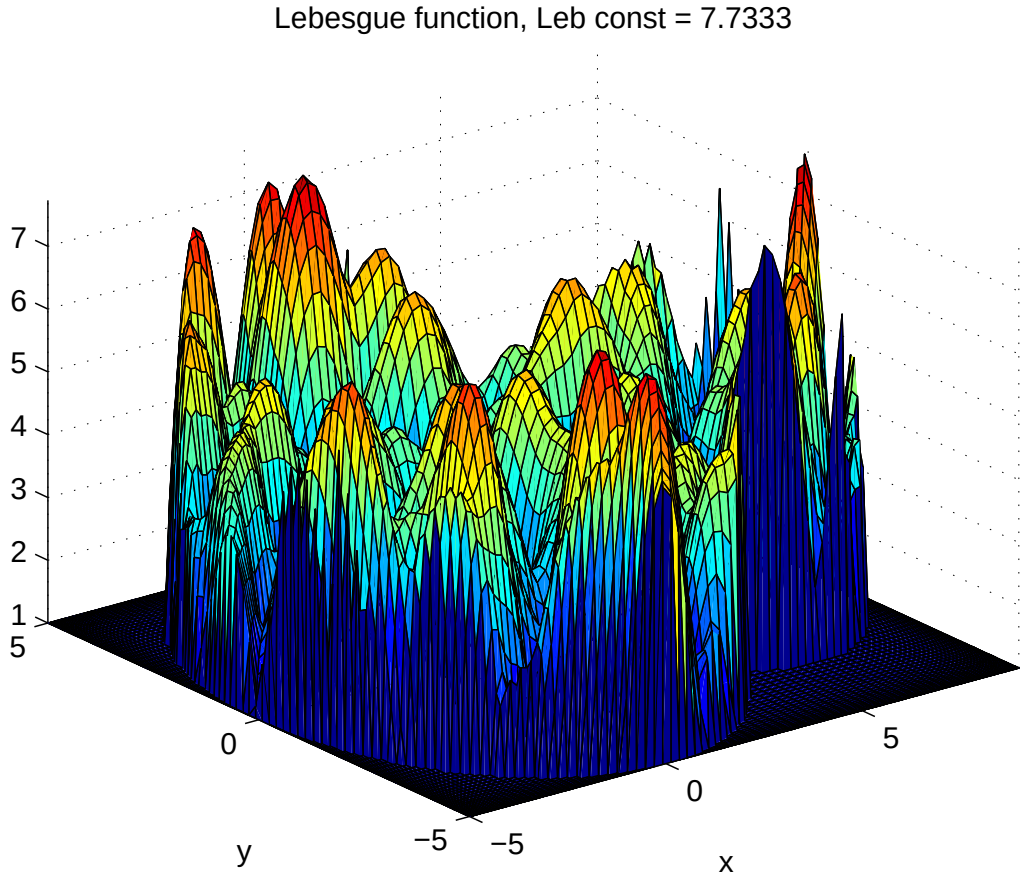
Double bubble: Lebesgue constant with L_2-norm minimization as a function of the degree



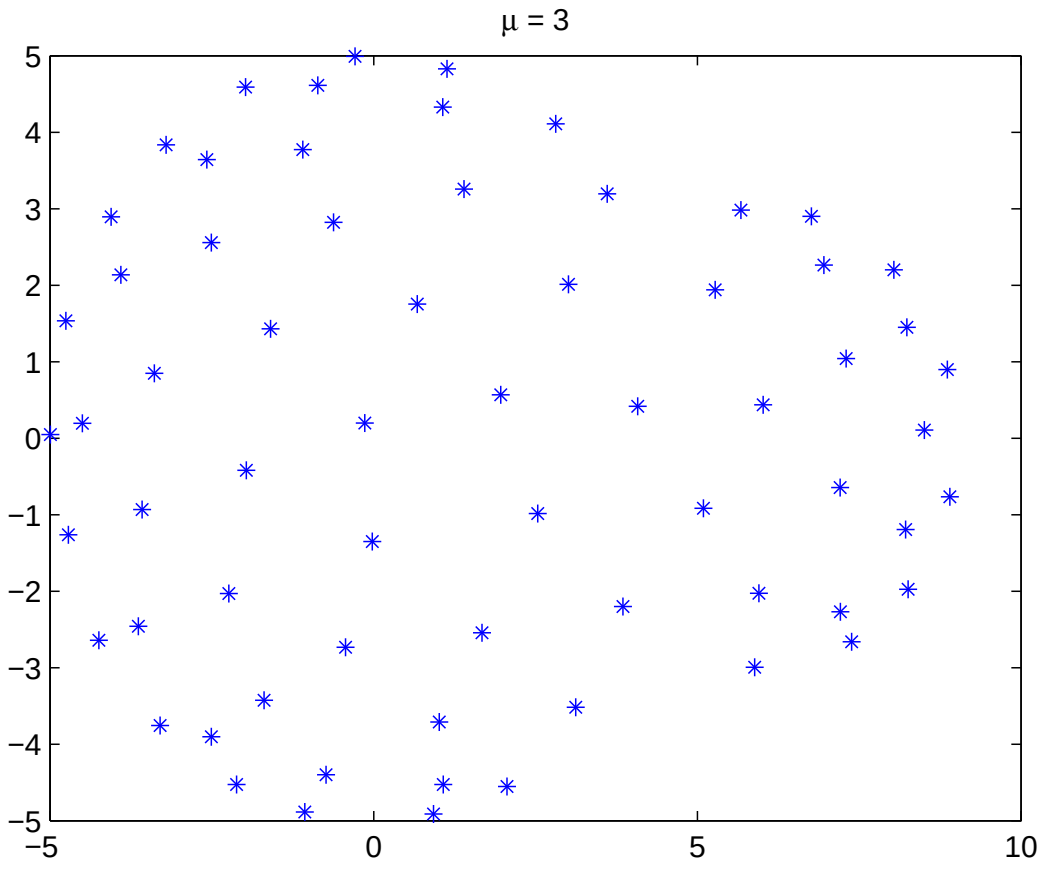
Double bubble: Lebesgue constant with L_2-norm minimization + refinement as a function of the degree



Double bubble: Lebesgue constant with L_2-norm minimization + refinement + local min as a function of the degree



Double bubble: Lebesgue function, $d = 10$, $\mu = 3$



Double bubble: The distribution of points for degree 10, $\mu = 3$

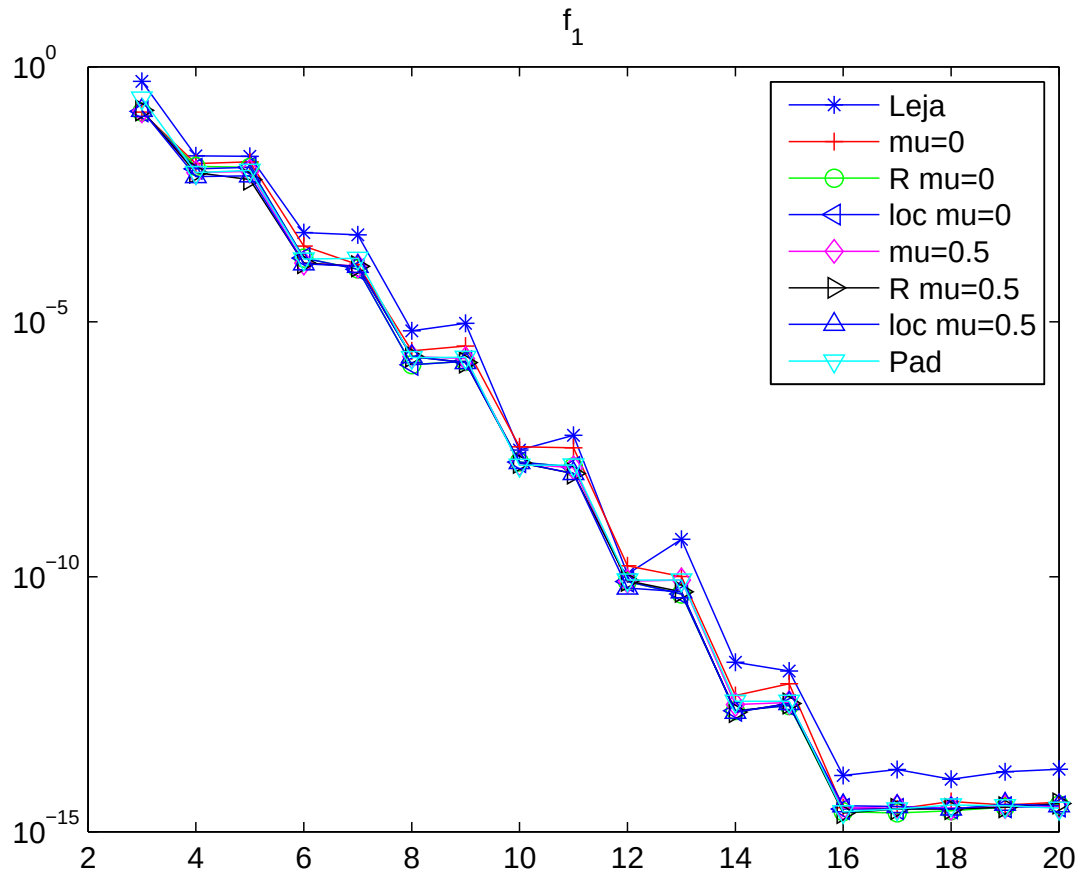
Interpolation

We use the orthogonal polynomials to compute the Lagrange polynomials and the interpolates of the functions:

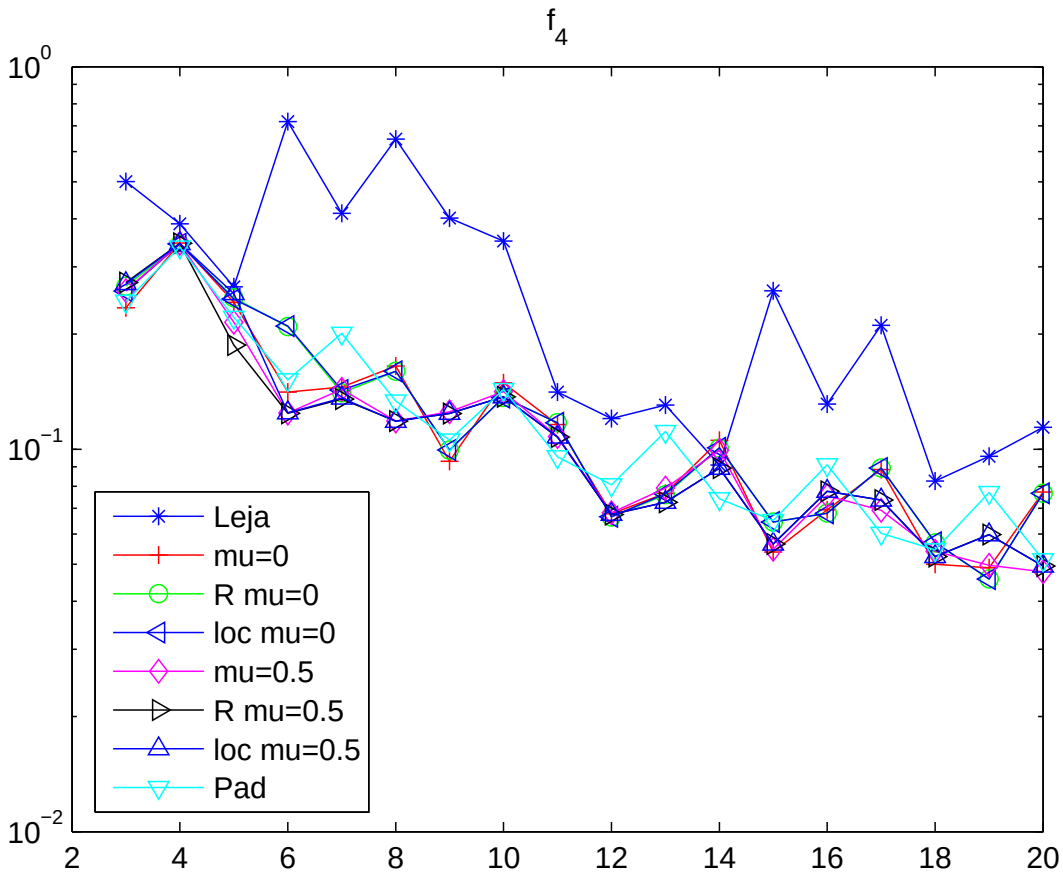
$$f_1 = \cos(x + y),$$

$$f_4 = |x - 1/2| + |y - 1/2|$$

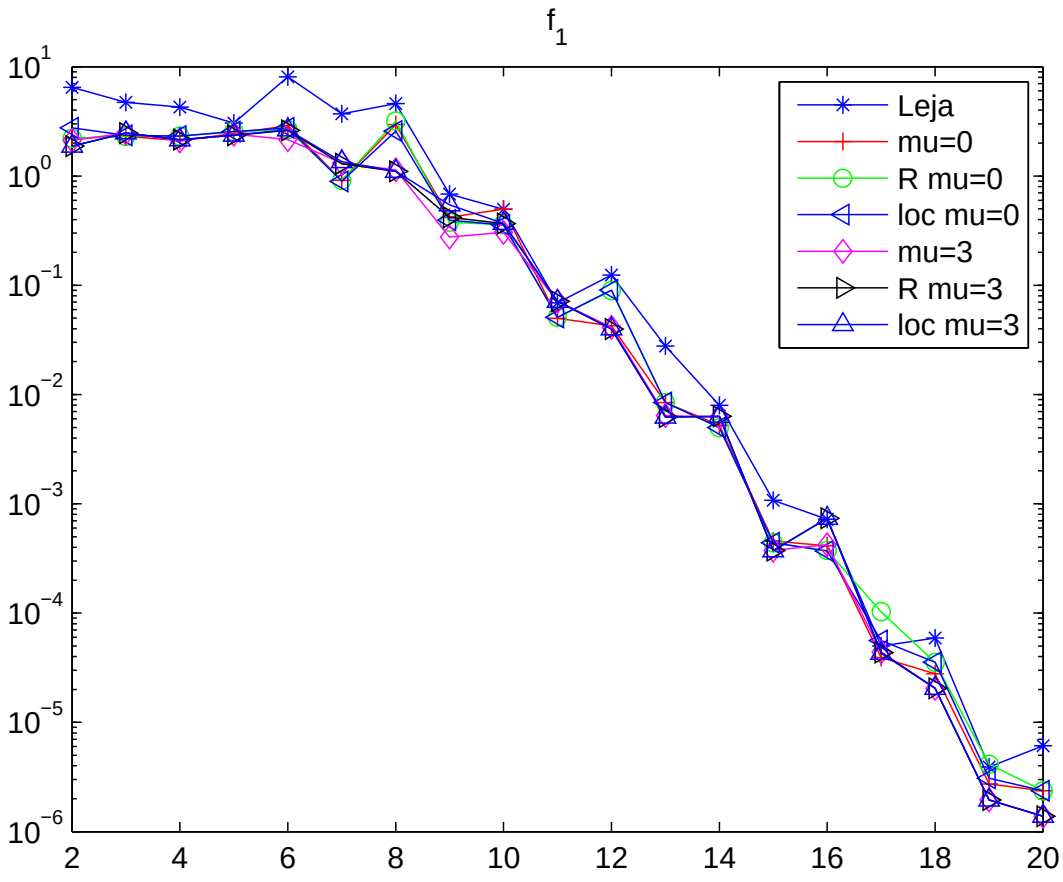
We consider the square and the double bubble



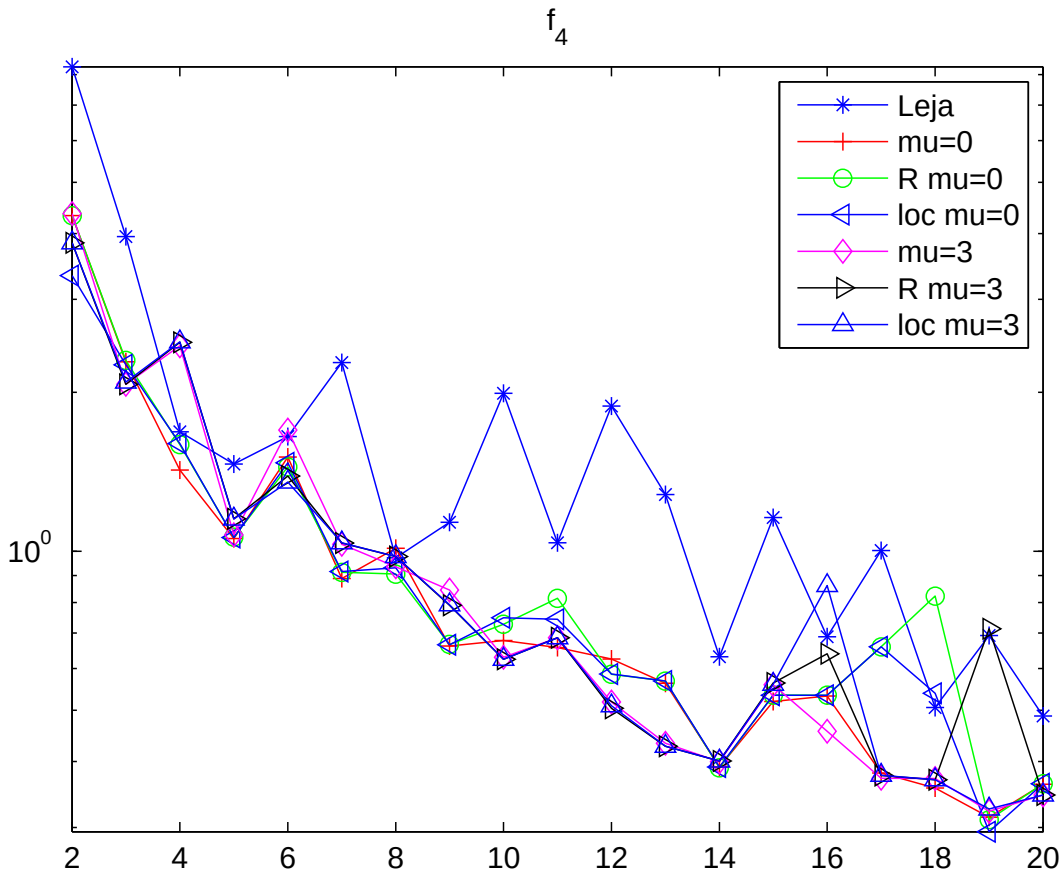
Square: interpolation error as a function of the degree, function f_1



Square: interpolation error as a function of the degree, function f_4



Double bubble: interpolation error as a function of the degree, function f_1



Double bubble: interpolation error as a function of the degree, function f_4

Conclusion

- We obtain sets of points with small Lebesgue constants using minimization of an integral
- The values of the functional are computed using bivariate orthogonal polynomials
- We use cubature formulas to compute the integrals
- This can be done for any bounded 2D domain for which we have an indicating function and a cubature formula