Chapter 6

Least Squares Approximation

As we saw in Chapter 5 we can interpret radial basis function interpolation as a constrained optimization problem. We now take this point of view again, but start with a more general formulation. Let's assume we are seeking a function $\mathcal{P}f$ of the form

$$\mathcal{P}f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j), \qquad \boldsymbol{x} \in \mathbb{R}^s,$$

such that the quadratic form

$$\frac{1}{2}\boldsymbol{c}^T Q \boldsymbol{c} \tag{6.1}$$

with $\boldsymbol{c} = [c_1, \ldots, c_M]^T$ and some symmetric positive definite matrix Q is minimized subject to the linear constraints

$$A\boldsymbol{c} = \boldsymbol{f} \tag{6.2}$$

where A is an $N \times M$ matrix, and the right-hand side $\boldsymbol{f} = [f_1, \ldots, f_N]^T$ is given. Such a constrained quadratic minimization problem can be converted to a system of linear equations by introducing *Lagrange multipliers*, i.e., we consider finding the minimum of

$$\frac{1}{2}\boldsymbol{c}^T Q \boldsymbol{c} - \boldsymbol{\lambda}^T \left[A\boldsymbol{c} - \boldsymbol{f}\right] \tag{6.3}$$

with respect to $\boldsymbol{c}$ and $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_N]^T$. Since Q is a positive definite matrix, it is well known that the functional to be minimized is convex, and thus has a unique minimum. Therefore, the standard necessary condition for such a minimum (which is obtained by differentiating with respect to $\boldsymbol{c}$ and $\boldsymbol{\lambda}$ and finding the zeros of those derivatives) is also sufficient. This leads to

$$\begin{aligned} Q\boldsymbol{c} - A^T\boldsymbol{\lambda} &= \boldsymbol{0} \\ A\boldsymbol{c} - \boldsymbol{f} &= \boldsymbol{0} \end{aligned}$$

or, in matrix form,

$$\begin{bmatrix} Q & -A^T \\ A & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{f} \end{bmatrix}.$$

By applying Gaussian elimination to this block matrix (Q is invertible since it is assumed to be positive definite) we get

$$\boldsymbol{\lambda} = \left(AQ^{-1}A^T\right)^{-1}\boldsymbol{f} \tag{6.4}$$

$$\boldsymbol{c} = Q^{-1}A^T\left(AQ^{-1}A^T\right)^{-1}\boldsymbol{f}. \tag{6.5}$$

In particular, if the quadratic form represents the native space norm of the interpolant $\mathcal{P}f = \sum_{j=1}^{M} c_j\Phi(\cdot,\boldsymbol{x}_j)$, i.e.,

$$\|\mathcal{P}f\|^2_{\mathcal{N}_\Phi(\Omega)} = \sum_{i=1}^{M}\sum_{j=1}^{M} c_i c_j \Phi(\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{c}^T Q\boldsymbol{c}$$

with $Q_{ij} = \Phi(\boldsymbol{x}_i,\boldsymbol{x}_j)$ and $\boldsymbol{c} = [c_1,\ldots,c_M]^T$, and the linear side conditions are the interpolation conditions

$$A\boldsymbol{c} = \boldsymbol{f} \qquad \Longleftrightarrow \qquad \mathcal{P}f(\boldsymbol{x}_i) = f_i, \quad i = 1,\ldots,M,$$

with $A = Q$ (symmetric) and the same $\boldsymbol{c}$ as above and data vector $\boldsymbol{f} = [f_1,\ldots,f_M]^T$, then we see that the Lagrange multipliers (6.4) become

$$\boldsymbol{\lambda} = A^{-T}\boldsymbol{f} = A^{-1}\boldsymbol{f}$$

and the coefficients are given by

$$\boldsymbol{c} = \boldsymbol{\lambda}$$

via (6.5). Therefore, as we saw earlier, the minimum norm interpolant is obtained by solving the interpolation equations alone.

Since we took the more general point of view that $\mathcal{P}$ is generated by M basis functions, and N linear constraints are specified, the above formulation also covers both over- and under-determined least squares fitting where the quadratic form $\boldsymbol{c}^T Q\boldsymbol{c}$ represents an added *smoothing* (or *regularization*) term. This term is not required to obtain a unique solution of the system $A\boldsymbol{c} = \boldsymbol{f}$ in the over-determined case ($M \le N$), but in the under-determined case such a constraint is needed (cf. the solution of under-determined linear systems via singular value decomposition in the numerical linear algebra literature (e.g., [608])).

Usually the regularized least squares approximation problem is formulated as minimization of

$$\frac{1}{2}\boldsymbol{c}^T Q\boldsymbol{c} + \omega\sum_{j=1}^{N}\left(\mathcal{P}f(\boldsymbol{x}_j) - f_j\right)^2. \tag{6.6}$$

The quadratic form controls the smoothness of the fitting function and the least squares term measures the closeness to the data. The parameter ω controls the tradeoff between these two terms. The formulation (6.6) is used in *regularization theory* (see, e.g., [185, 252]). The same formulation is also used in *penalized least squares* fitting (see, e.g., [263]), the literature on smoothing splines [528, 572], and in papers by Wahba on thin plate splines (e.g., [615, 621]). In fact, the idea of smoothing a data fitting process by this kind of formulation seems to go back to at least Whittaker [641] in 1923. In practice a penalized least squares formulation is especially useful if the data f_i cannot

be completely trusted, i.e., it is contaminated by noise. In this case, a (penalized) least squares fit is advisable. The problem of minimizing (6.6) is known as *ridge regression* in the statistics literature.

The equivalence with our formulation (6.3) above follows from

$$\begin{aligned}\frac{1}{2}\boldsymbol{c}^T Q\boldsymbol{c} + \omega \sum_{j=1}^{N} (\mathcal{P}f(\boldsymbol{x}_j) - f_j)^2 &= \frac{1}{2}\boldsymbol{c}^T Q\boldsymbol{c} + \omega[A\boldsymbol{c} - \boldsymbol{f}]^T[A\boldsymbol{c} - \boldsymbol{f}] \\ &= \frac{1}{2}\boldsymbol{c}^T Q\boldsymbol{c} - \boldsymbol{\lambda}^T[A\boldsymbol{c} - \boldsymbol{f}],\end{aligned}$$

where

$$\boldsymbol{\lambda} = -\omega[A\boldsymbol{c} - \boldsymbol{f}].$$

We are now interested in the more general setting where we still sample the given function f on the set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, but now introduce a second set $\Xi = \{\boldsymbol{\xi}_i\}_{i=1}^M$ at which we center the basis functions. Usually we will have $M \leq N$, and the case $M = N$ with $\Xi = \mathcal{X}$ recovers the traditional interpolation setting discussed thus far. Therefore,

$$\mathcal{Q}f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \Phi(\boldsymbol{x}, \boldsymbol{\xi}_j), \quad \boldsymbol{x} \in \mathbb{R}^s, \tag{6.7}$$

and the coefficients c_j can be found by minimizing $\|Qf - f\|_2^2$, where the ℓ_2-norm

$$\|f\|_2^2 = \sum_{i=1}^{N} [f(\boldsymbol{x}_i)]^2$$

is induced by the discrete inner product

$$\langle f, g \rangle = \sum_{i=1}^{N} f(\boldsymbol{x}_i) g(\boldsymbol{x}_i). \tag{6.8}$$

This approximation problem has a unique solution if the (rectangular) *collocation matrix* A with entries

$$A_{jk} = \Phi(\boldsymbol{x}_j, \boldsymbol{\xi}_k), \quad j = 1, \ldots, N, \ k = 1, \ldots, M,$$

has full rank.

Remarks:

1. If the centers in Ξ are chosen to form a subset of the data locations $\mathcal{X}$ then A has full rank provided the radial basis function is selected according to our previous chapters on interpolation. This is true, since in this case A will have an $M \times M$ square submatrix which is non-singular (by virtue of being an *interpolation matrix*).

2. The over-determined linear system $A\boldsymbol{c} = \boldsymbol{y}$ which arises in the solution of the least squares problem can be solved using standard algorithms from numerical linear algebra such as QR or singular value decomposition.

In the following section we give a brief account of theoretical results known for the general problem in which the centers and data sites differ.

6.1 Theoretical Results

The results mentioned here are due to Sivakumar and Ward [583], and Quak, Sivakumar and Ward [521]. The first paper deals with discrete least squares, the second with continuous least squares approximation. In both papers the authors do not discuss the collocation matrix A above, but rather base their results on the non-singularity of the coefficient matrix obtained from a system of normal equations. In the discrete setting they use the inner product (6.8) which induces the ℓ_2 norm, and then discuss non-singularity of the *Gramian* which occurs in the following system of *normal equations*

$$G\boldsymbol{c} = \boldsymbol{w}, \tag{6.9}$$

where the entries of G are the ℓ_2 inner products of the radial basis functions, i.e.,

$$G_{jk} = \langle \Phi(\cdot, \boldsymbol{\xi}_j), \Phi(\cdot, \boldsymbol{\xi}_k) \rangle = \sum_{i=1}^{N} \Phi(\boldsymbol{x}_i, \boldsymbol{\xi}_j)\Phi(\boldsymbol{x}_i, \boldsymbol{\xi}_k), \quad j,k = 1, \ldots, M,$$

and the right-hand side vector $\boldsymbol{w}$ in (6.9) is given by

$$\boldsymbol{w}_j = \langle \Phi(\cdot, \boldsymbol{\xi}_j), \boldsymbol{f} \rangle = \sum_{i=1}^{N} \Phi(\boldsymbol{x}_i, \boldsymbol{\xi}_j) f(\boldsymbol{x}_i), \quad j = 1, \ldots, M.$$

Remarks:

1. Note that in the interpolation case with $M = N$ and $\Xi = \mathcal{X}$ we have

   $$\langle \Phi(\cdot, \boldsymbol{x}_j), \Phi(\cdot, \boldsymbol{x}_k) \rangle = \langle \Phi(\cdot, \boldsymbol{x}_j), \Phi(\cdot, \boldsymbol{x}_k) \rangle_{\mathcal{N}_\Phi(\Omega)} = \Phi(\boldsymbol{x}_j, \boldsymbol{x}_k)$$

   so that G is just the interpolation matrix A.

2. Of course, this also presents an interpretation of the interpolation matrix A as the system matrix for the normal equations in the case of best approximation with respect to the native space norm – a fact already mentioned earlier in the section on optimal recovery.

In both papers, [583] as well as [521], even the formulation of the main theorems is very technical. We therefore just try to give a feel for their results.

Essentially, the authors show that the Gramian for certain radial basis functions (norm, (inverse) multiquadrics, and Gaussians) is non-singular if the centers $\boldsymbol{\xi}_k$, $k = 1, \ldots, M$, are sufficiently well distributed and the data points $\boldsymbol{x}_j$, $j = 1, \ldots, N$, are fairly evenly clustered about the centers with the diameter of the clusters being relatively small compared to the separation distance of the data points. Figure 6.1 illustrates the clustering idea.

One of the key ingredients in the proof of the non-singularity of G is to set up an interpolation matrix B for which the basis functions are centered at certain representatives of the clusters of knots about the data sites. One then splits the matrix B (which is non-symmetric in general) into a part which is symmetric and one which is

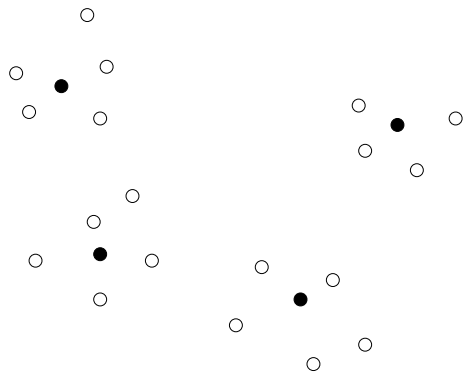

Figure 6.1: Clusters of data points ∘ around well separated centers •.

anti-symmetric, a standard strategy in linear algebra, i.e., $B = B_1 + B_2$ where B_1 and B_2 are defined by

$$\begin{aligned} B_1 &= \frac{B + B^T}{2}, \quad \text{(symmetric)}, \\ B_2 &= \frac{B - B^T}{2}, \quad \text{(anti-symmetric)}. \end{aligned}$$

Then, lower estimates for the norm of these two parts are found and used to conclude that, under certain restrictions, G is non-singular.

Remarks:

1. As a by-product of this argumentation the authors obtain a proof for the non-singularity of *interpolation* matrices for the case in which the centers of the basis functions are chosen different from the data sites, namely as small perturbations thereof.

2. The discussion of the continuous case is very similar to that of the discrete one.

6.2 Adaptive Least Squares using Knot Insertion

In this and in the following section we mention some strategies for an algorithm for solving the least squares problem in an adaptive fashion. When fitting data by linear combinations of certain basis functions, it is a classical technique to improve the quality of a given initial approximation by increasing the number of basis functions used for the fit, i.e., by refining the space from which we are approximating. Since every radial basis function is associated with one particular center (or *knot*), this can be achieved by adding new knots to the existing ones. This idea was explored for multiquadrics on $\mathbb{R}^2$ in [237, 238], and for radial basis functions on spheres in [192].

We will now describe an algorithm which adaptively adds knots to a radial basis function approximant in order to improve the ℓ_2 error.

Let us assume we are given a large number, N, of data and we want to fit them with a radial basis expansion to within a given tolerance. The idea is to start with very

few initial knots, and then to repeatedly insert a knot at that data location whose ℓ_2 error component is largest. This is done as long as the least squares error exceeds a given tolerance. The following algorithm may be used.

Algorithm: Knot insertion

(1) Let data sites $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, data f_i, $i = 1, \ldots, N$, and a tolerance ε be given.

(2) Choose M initial knots $\Xi = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}$.

(3) Calculate the least squares fit

$$\mathcal{Q}f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \Phi(\boldsymbol{x}, \boldsymbol{\xi}_j)$$

with its associated error

$$e = \sum_{i=1}^{N} [f_i - \mathcal{Q}f(\boldsymbol{x}_i)]^2.$$

While $e > \varepsilon$ **do**

(4) "Weight" each data point $\boldsymbol{x}_i$, $i = 1, \ldots, N$, according to its error component, i.e., let

$$w_i = |f_i - \mathcal{Q}f(\boldsymbol{x}_i)|, \quad i = 1, \ldots, N.$$

(5) Find the data point $\boldsymbol{x}_\nu \notin \Xi$ with maximum weight w_ν and insert it as a knot, i.e.,

$$\Xi = \Xi \cup \{\boldsymbol{x}_\nu\} \quad \text{and} \quad M = M + 1.$$

(6) Recalculate fit and associated error.

Remarks:

1. We note that we have to solve one linear least squares problem for every knot we add. This can be done employing standard techniques such as QR or SVD factorization. The size of these problems increases by one at each step. In order to improve the runtime of this algorithms an updating QR factorization (see e.g., [265]) could be used. However, neither [237, 238] nor [192] have found this necessary since the problems they considered begin with very small systems (and therefore fast solutions), and the desired accuracy was achieved with fairly few additional knots.

2. If the initial knots are chosen to lie at data sites, the process described in the above algorithm will always have a full rank collocation matrices A. This is guaranteed, since we only add data sites as new knots, and we make sure in step (5) that no multiple knots are created (which would obviously lead to a rank deficiency).

6.3 Adaptive Least Squares using Knot Removal

The idea of knot removal was primarily motivated by the need for data reduction, but it can also be used for the purpose of adaptive approximation (for a survey of knot removal see, e.g., [412]). The basic idea is to start with a good fit (e.g., an interpolation to the data), and then successively reduce the number of knots (and therefore basis functions) used until a certain given tolerance is reached.

Specifically, this means we will start with an initial fit and then use some kind of weighting strategy for the knots, so that we can repeatedly remove those contributing least to the accuracy of the fit. The following algorithm performs this task.

Algorithm: Knot removal

(1) Let data points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, data f_i, $i = 1, \ldots, N$, and a tolerance ε be given.

(2) Choose M initial knots $\Xi = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}$.

(3) Calculate an initial fit

$$\mathcal{Q}f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \Phi(\boldsymbol{x}, \boldsymbol{\xi}_j)$$

with its associated least squares error

$$e = \sum_{i=1}^{N} [f_i - \mathcal{Q}f(\boldsymbol{x}_i)]^2.$$

While $e < \varepsilon$ **do**

(4) "Weight" each knot $\boldsymbol{\xi}_j$, $j = 1, \ldots, M$, according to its least squares error, i.e., form

$$\Xi^* = \Xi \setminus \{\boldsymbol{\xi}_j\},$$

and calculate the weights

$$w_j = \sum_{i=1}^{N} \left[f_i - \mathcal{Q}^* f(\boldsymbol{x}_i)\right]^2,$$

where

$$\mathcal{Q}^* f(\boldsymbol{x}) = \sum_{j=1}^{M-1} c_j \Phi(\boldsymbol{x}, \boldsymbol{\xi}_j^*)$$

is the approximation based on the reduced set of knots Ξ^*.

(5) Find the knot $\boldsymbol{\xi}_\mu$ with lowest weight $w_\mu < \varepsilon$ and permanently remove it, i.e.

$$\Xi = \Xi \setminus \{\boldsymbol{\xi}_\mu\} \quad \text{and} \quad M = M - 1.$$

(6) Recalculate fit and associated error.

Again we would like to comment on the algorithm.

Remarks:

1. As far as computational times are concerned, this algorithm is *much* slower than the knot insertion algorithm, since the weight for every knot is determined by the solution of a least squares problem, i.e., in every iteration we solve M least squares problems. These problems, however, become increasingly smaller.

2. This approach should be especially beneficial when the number of evaluations of the constructed approximant is much larger than its actual computation, since, for the same tolerance, one would expect knot removal to result in fewer knots than knot insertion.

3. If the initial knots are chosen at the data sites, then again there will be no problems with the collocation matrix becoming rank deficient.

4. In [192, 193] some alternatives to this knot removal strategy were also considered. One of them is the removal of certain groups of knots at one time in order to speed up the process. We used this version of knot removal algorithm in our examples in the next subsection. Another is based on choosing the weights based on the size of the coefficients c_j in the expansion of $\mathcal{Q}f$, i.e., to remove that knot whose associated coefficient is smallest.

5. A further variation of the adaptive algorithms was considered in both [237] and in [192]. Instead of treating only the coefficients of the expansion of $\mathcal{Q}f$ as parameters in the minimization process, one can also include the knot locations themselves and possibly the parameters which are inherent in the definition of some of the radial functions used in practice. This however, leads to *nonlinear* least squares problems. We will not discuss this topic further here.

6. Buhmann, Derrien, and Le Méhauté [88], and Le Méhauté [366] also discuss knot removal. Their approach is based on an *a priori estimate* for the error made when removing a certain knot. These estimates depend on the specific choice of radial basis function, and so far they only cover the inverse multiquadric type, i.e.,

$$\varphi(r) = (r^2 + \alpha^2)^{\beta/2}, \quad -s \leq \beta < 0, \ \alpha > 0.$$

7. Iske [317] suggests an alternative knot removal strategy for least squares approximation. His removal heuristics are based on so-called *thinning algorithms*. In particular, in each iteration a point is removed if it belongs to a pair of points in Ξ with minimal separation distance. The thinning phase of the algorithm is then enhance by an exchange phase in which points can be "swapped back in" if this process reduces the fill-distance of Ξ.

	μ	ρ	# knots used	time
KI	8.074767e-02	1.773359e-04	30	4 sec
KR	6.948009e-02	1.488779e-04	36	153 sec

Table 6.1: Comparison of adaptive algorithms for Franke's function ($\varepsilon = 0.01$).

6.4 Some Numerical Examples

For the following tests we consider Franke's test function

$$\begin{aligned} f(x,y) &= \frac{3}{4}e^{-1/4((9x-2)^2+(9y-2)^2)} + \frac{3}{4}e^{-(1/49)(9x+1)^2-(1/10)(9y+1)^2} \\ &\quad + \frac{1}{2}e^{-1/4((9x-7)^2+(9y-3)^2)} - \frac{1}{5}e^{-(9x-4)^2-(9y-7)^2}. \end{aligned} \tag{6.10}$$

The range of this function on the grid $\mathcal{G}$ described below is

$$\text{range } f = [0.003280, 1.220000].$$

The graph of Franke's function is shown in Figure 6.2. We choose the set $\mathcal{X}$ of data sites as the grid G_{64} of 8×8 equally spaced points in the unit square $[0,1] \times [0,1]$ of $\mathbb{R}^2$.

For the evaluation and rendering of our test examples we use a grid $\mathcal{G}$ of 30×30 equally spaced points in the unit square. On this grid we compute maximum errors

$$\mu := \|f - \mathcal{Q}f\|_\infty = \max_{\boldsymbol{x}\in\mathcal{G}} |f(\boldsymbol{x}) - \mathcal{Q}f(\boldsymbol{x})|,$$

and mean-square errors

$$\rho := \frac{1}{900}\sum_{\boldsymbol{x}\in\mathcal{G}} |f(\boldsymbol{x}) - \mathcal{Q}f(\boldsymbol{x})|^2,$$

where f is the known test function, and $\mathcal{Q}f$ is a radial basis function approximant to it.

For the results shown in Table 6.1 we have used the multiquadric without any constant added to its expansion. The value of $\alpha = 0.3$, and the tolerance ε was chosen to be 0.01. We compare the knot insertion algorithm (KI) with a version of the knot removal algorithm (KR) which removes certain groups of knots all at once before a new weighting step is performed, rather than re-weighting after the removal of each individual knot. The effect of this is a considerable speedup in execution time with some loss of accuracy (we point out that the numerical experiments were performed in 1996, and therefore the times are only to be used as relative measures of performance). We do not consider this loss of accuracy to be so crucial, since there does not exist a strategy which would prescribe the optimal order in which to remove knots one at a time, either. The knot insertion algorithm is initialized with a single random knot in the unit square, the knot removal algorithm begins with an interpolant to all 64 data values.

Figure 6.2 shows the original function, and Figures 6.3 and 6.4 the approximation via knot insertion and the fit obtained by performing knot removal along with the grid

	μ	ρ	# knots used	time
KI	4.191418e-02	4.630199e-05	52	18 sec
KR	4.1404723-02	4.618084e-05	49	399 sec

Table 6.2: Comparison of adaptive algorithms for Franke's function ($\varepsilon = 0.0001$).

of data sites and the knots used for the fit. The color shading of the graphs of the approximations is such that a dark color reflects small maximum errors, and a light color indicates large errors.

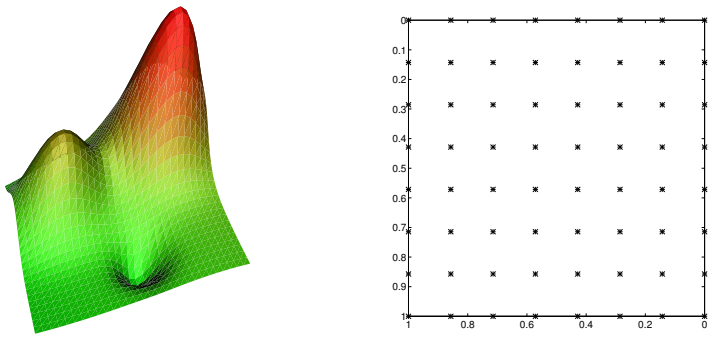

Figure 6.2: (a) Franke's function, and (b) 8×8 grid.

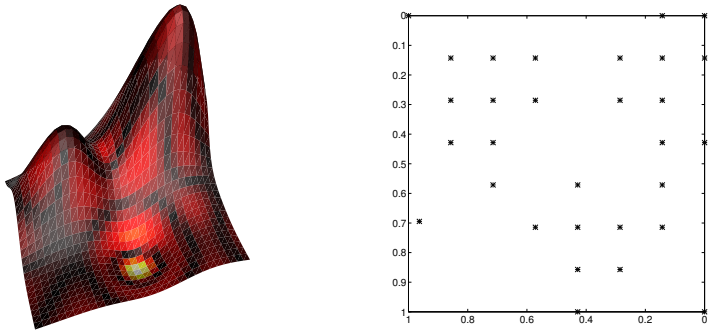

Figure 6.3: (a) Knot insertion fit, and (b) selected knots for $\varepsilon = 0.01$.

Table 6.2 along with Figures 6.5 and 6.6 show the approximation via knot insertion and the fit obtained by performing knot removal using a tighter tolerance of $\varepsilon = 0.0001$ along with the respective knot sets.

It is clearly apparent that the knot insertion algorithm is *much* faster than the one for knot removal. However, we should remark that the implementation of the method used for the solution of the least squares systems uses SVD, which emphasizes this discrepancy even more. The knot removal algorithm for the tighter tolerance performed relatively more exact than that with tolerance $\varepsilon = 0.01$. This can be seen when one observes the size of the groups of knots which were removed. For tolerance

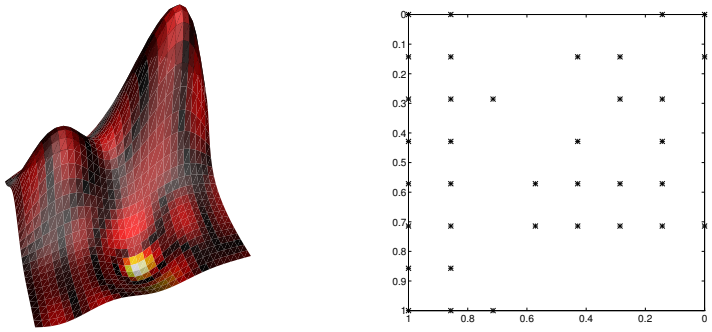

Figure 6.4: (a) Knot removal fit, and (b) selected knots for $\varepsilon = 0.01$.

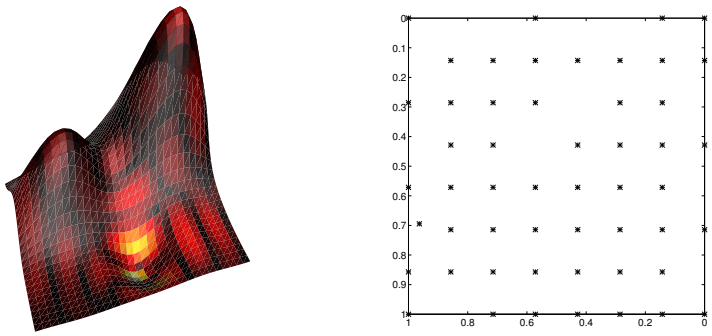

Figure 6.5: (a) Knot insertion fit, and (b) selected knots for $\varepsilon = 0.0001$.

$\varepsilon = 0.01$ first 16 and then 12 knots were removed, whereas for $\varepsilon = 0.0001$ the algorithm was not quite as radical and removed knots in groups of size 3, 4, 4, 4, and 2. This is also evident from the fact that for $\varepsilon = 0.0001$ the KR algorithm ended up using fewer knots than the KI algorithm, which was not the case for $\varepsilon = 0.01$. In general, one should expect the KR algorithm to be able to produce fits with a comparable accuracy to that of a KI fit, but using fewer knots. This should be so since the KR algorithm removes redundancies from an ideal initial fit, whereas the KI algorithm starts with almost no information and has to find a good fit from there.

It is also interesting to note the actual sets of knots chosen by the two methods. They certainly are not the same, but quite a few features are in common.

Remark: Since we developed the native space theory in Chapter 5 it is clear that the above algorithms can also be performed in an analogous way with respect to best approximation in the native space norm. Now, the power function can be used as an error indicator. This idea is pursued in recent papers by Schaback and Wendland [562, 563] to design so-called *greedy algorithms*.

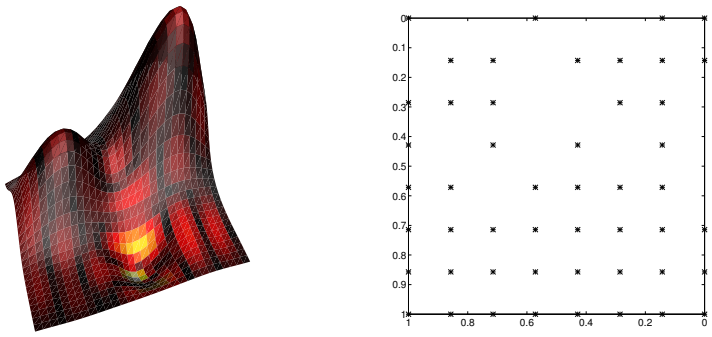

Figure 6.6: (a) Knot removal fit, and (b) selected knots for $\varepsilon = 0.0001$.