Greedy Sparse Linear Approximations of Functionals from Nodal Data

Robert Schaback*

July 11, 2013

Abstract

In many numerical algorithms, integrals or derivatives of functions have to be approximated by linear combinations of function values at nodes. This ranges from numerical integration to meshless methods for solving partial diffeential equations. The approximations should use as few nodal values as possible and at the same time have a smallest possible error. For each fixed set of nodes and each fixed Hilbert space of functions with continuous point evaluation, e.g. a fixed Sobolev space, there is an error–optimal method available using the reproducing kernel of the space. But the choice of the nodes is usually left open. This paper shows how to select good nodes adaptively by a computationally cheap greedy method, keeping the error optimal in the above sense for each incremental step of the node selection. This is applied to interpolation, numerical integration, and numerical differentiation. The latter case is particularly important for the design of meshless methods with sparse generalized stiffness matrices. The greedy algorithm is described in detail, and numerical examples are provided.

1 Introduction

We consider a continuous linear functional λ on a reproducing kernel Hilbert space (RKHS) H with positive definite kernel K and domain Ω, and we want to approximate it by

$$\lambda(f) \approx \sum_j \alpha_j f(x_j) = \sum_j \alpha_j \delta_{x_j}(f) \text{ for all } f \in H$$

by certain selections of points x_j from a given discrete point set $X \subset \Omega$. The representation should be sparse, i.e. we do not want to use more points from X than

*Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Lotzestraße 16–18, 37083 Göttingen, Germany, schaback@math.uni-goettingen.de

absolutely necessary. The technique to construct such approximations is briefly mentioned in [11], but we give the algorithmic details and examples here. A general paper on kernel–based approximations for the special case of numerical differentiation is [1], while [10] applies the kernel–based error analysis behind this paper to methods for PDE solving.

If the points are fixed to be from a set $X_n := \{x_1, \ldots, x_n\}$, the *error functional* is

$$\varepsilon(\lambda; \alpha_1, \ldots, \alpha_n) := \lambda - \sum_{j=1}^{n} \alpha_j \delta_{x_j}$$

and the corresponding error bound is

$$\left| \lambda(f) - \sum_{j=1}^{n} \alpha_j f(x_j) \right| \leq \|\varepsilon(\lambda; \alpha_1, \ldots, \alpha_n)\|_{H^*} \|f\|_H$$

which is sharp for the function

$$f_\varepsilon(x) := \varepsilon^y(\lambda; \alpha_1, \ldots, \alpha_n) K(x,y) = \lambda^y K(x,y) - \sum_{j=1}^{n} \alpha_j K(x, x_j)$$

in H. Here and in what follows, the notation λ^y means that λ acts with respect to the variable y.

Since we have

$$(\lambda, \mu)_{H^*} = \lambda^x \mu^y K(x,y) \text{ for all } \lambda, \mu \in H^*$$

in reproducing kernel Hilbert spaces, the norm of the error functional can be explicitly calculated via

$$\begin{aligned} \|\varepsilon(\lambda; \alpha_1, \ldots, \alpha_n)\|^2_{H^*} &= (\varepsilon(\lambda; \alpha_1, \ldots, \alpha_n), \varepsilon(\lambda; \alpha_1, \ldots, \alpha_n))_{H^*} \\ &= \varepsilon^x(\lambda; \alpha_1, \ldots, \alpha_n) \varepsilon^y(\lambda; \alpha_1, \ldots, \alpha_n) K(x,y) \\ &= \lambda^x \lambda^y K(x,y) - 2 \sum_{j=1}^{n} \alpha_j \lambda^x K(x, x_j) \\ &\quad + \sum_{j=1}^{n} \sum_{k=1}^{n} \alpha_j \alpha_k K(x_j, x_k). \end{aligned} \tag{1}$$

This implies that each approximation has an explicitly available error bound in terms of a percentage of $\|f\|_H$, and this observation can easily be used to compare approximations that come from very different backgrounds.

The most important example is error evaluation in Sobolev space. If we take $H = W_2^m(\mathbb{R}^d)$ with $m > d/2$ for having continuous point evaluation, the reproducing kernel is

$$K(x,y) = \frac{2^{1-m}}{\Gamma(m)} \|x-y\|_2^{m-d/2} K_{m-d/2}(\|x-y\|_2),\ x,y \in \mathbb{R}^d$$

with the modified Bessel function $K_{m-d/2}$ of the second kind. Localized Sobolev spaces $W_2^m(\Omega)$ for domains $\Omega \subset \mathbb{R}^d$ come out to be norm–equivalent to the global spaces, as long as their domains Ω satisfy a Whitney extension property or a have a piecewise smooth boundary with a uniform interior cone condition. Therefore we shall use the above kernels for evaluation of errors in Sobolev space throughout.

2 Optimal Kernel–Based Formulae

The error norm explicitly given via (1) now allows for optimization with respect to the coefficients and nodes. Fixing the nodes first, optimal coefficients are determined by solving the system

$$\lambda^x K(x,x_k) = \sum_{j=1}^{n} \alpha_j K(x_j,x_k),\ \text{ for all } k,\ 1 \le k \le n$$

by coefficients α_j^* depending on λ and all points in X_n. This follows from setting the partial derivatives of (1) to zero. The square of the optimal error norm then is the nonnegative number

$$P_n^2(\lambda) := \|\varepsilon(\lambda;\alpha_1^*(\lambda),\ldots,\alpha_n^*(\lambda))\|_{H^*}^2 = \lambda^x\lambda^y K(x,y) - \sum_j \alpha_j^*(\lambda)\lambda^y K(x_j,y).$$

Consequently, this optimal approximation outperforms all other approximations errorwise, if the nodes and the space H are fixed. Users can apply (1) to see how far their favourite approximation is from the optimum. This applies, for instance, when meshless methods approximate functionals like $\lambda_z(f) = (\Delta f)(z)$ in terms of values at nodes, using variations of Moving Least Squares techniques [3].

3 Recursive Construction

If we already have such an approximation with points $x_1,\ldots,x_n$, we now want to find the next point x_{n+1} such that $P_{n+1}^2(\lambda)$ is smallest. To this end, we have to

do some calculations that involve all available points x_j, and we use the recursive *power kernel* technique [8, 4]

$$\begin{aligned} K_0(x,y) &:= K(x,y) \\ K_{k+1}(x,y) &:= K_k(x,y) - \frac{K_k(x_{k+1},x)K_k(x_{k+1},y)}{K_k(x_{k+1},x_{k+1})} \\ \lambda^x K_0(x,y) &= \lambda^x K(x,y) \\ \lambda^x K_{k+1}(x,y) &= \lambda^x K_k(x,y) - \frac{\lambda^x K_k(x_{k+1},x)K_k(x_{k+1},y)}{K_k(x_{k+1},x_{k+1})} \end{aligned}$$

for $k \geq 0$ and the *Newton basis* calculations [5, 6] in the form

$$\begin{aligned} v_{k+1}(x) &:= \frac{K_k(x_{k+1},x)}{\sqrt{K_k(x_{k+1},x_{k+1})}} \\ K_{k+1}(x,y) &= K(x,y) - \sum_{j=1}^{k+1} v_j(x)v_j(y) \\ &= K_k(x,y) - v_{k+1}(x)v_{k+1}(y) \end{aligned}$$

where the *Newton basis* $v_1,\ldots,v_n$ is orthonormal in H, satisfies $v_k(x_j) = 0,\ 1 \leq j < k \leq n$ and spans the spaces

$$\text{span}\,\{v_1,\ldots,v_k\} = \text{span}\,\{K(\cdot,x_1),\ldots,K(\cdot,x_k)\}.$$

Then the standard *Power Function* is

$$\begin{aligned} P^2_{k+1}(\delta_x) &:= \|\varepsilon(\delta_x;\alpha^*_1(\delta_x),\ldots,\alpha^*_{k+1}(\delta_x))\|^2_{H^*} \\ &= K_{k+1}(x,x) \\ &= P^2_k(\delta_x) - v^2_{n+1}(x) \\ &= K(x,x) - \sum_{j=1}^{k+1} v^2_j(x) \end{aligned}$$

and the generalized one is

$$\begin{aligned} P^2_{k+1}(\lambda) &= \|\varepsilon(\lambda;\alpha^*_1(\lambda),\ldots,\alpha^*_{k+1}(\lambda))\|^2_{H^*} \\ &= \lambda^x\lambda^y K_{k+1}(x,y) \\ &= P^2_k(\lambda) - \lambda(v_{k+1})^2 \\ &= \lambda^x\lambda^y K(x,y) - \sum_{j=1}^{k+1} \lambda(v_j)^2. \end{aligned}$$

This follows from the fact [4] that K_k is the reproducing kernel in the Hilbert subspace of H that is orthogonal to all $K(\cdot,x_j),\ 1 \leq j \leq k$.

Thus the error norm reduction when going fron n points to $n+1$ points is $\lambda(v_{n+1})^2$, but this depends on x_{n+1}, and we want the reduction as a function of x_{n+1}. Since we have the formula

$$v_{n+1}(x) := \frac{K_n(x_{n+1},x)}{\sqrt{K_n(x_{n+1},x_{n+1})}}$$

we also have

$$\lambda(v_{n+1}) = \frac{\lambda^x(K_n(x_{n+1},x))}{\sqrt{K_n(x_{n+1},x_{n+1})}}$$

as a function of x_{n+1}. Thus the error reduction by using z as x_{n+1} is

$$R_n^2(z) := \frac{(\lambda^x K_n(z,x))^2}{K_n(z,z)}$$

as a function of z, and we should choose its argmax over all available points from the full set $X_N = \{x_1,\ldots,x_N\}$ as the new point x_{n+1}. In examples below, we shall show plots of this function on fine point sets.

4 Numerical Calculation

We now describe how to calculate R_n^2 on all N points of $X_N = \{x_1,\ldots,x_N\}$, in a handy form for programming. This data vector is written as a column vector $\mathbf{R}_n \in \mathbb{R}^N$, and in the following formulae we shall always work on N–column–vectors in boldface. In particular, we need N–vectors

$$\begin{aligned} \mathbf{K}_n &:= (K_n(x_j,x_j))_{1\le j\le N} \\ \mathbf{L}_n &:= (\lambda^x K_n(x,x_j))_{1\le j\le N} \end{aligned}$$

for $n \ge 0$ and start with calculating $\mathbf{K}_0$ and $\mathbf{L}_0$ to form

$$\mathbf{R}_n := \mathbf{L}_n.^{\wedge 2}./\mathbf{K}_n$$

in MATLAB notation for $n = 0$. We find the point $x_{I(1)}$ where this is maximal, building an index set I recursively, using the notation $y_1 := x_{I(1)}$ for shorthand. Then the first Newton basis function on the N points is the vector

$$\mathbf{V}_1 := \mathbf{k}_1./\sqrt{\mathbf{K}_0}$$

with the N–vector

$$\mathbf{k}_1 := (K_0(x_{I(1)},x_j))_{1\le j\le N}.$$

For the general recursion, we assume that we already have $\mathbf{K}_n$ and $\mathbf{L}_n$, an index set I of n indices of selected points, and all vectors $\mathbf{V}_j$, $1 \le j \le n$ of the Newton

basis vectors belonging to the points $y_j = x_{I(j)}$, $1 \le j \le n$. Then we can find $y_{n+1} = x_{I(n+1)}$ by maximizing $\mathbf{R}_n = \mathbf{L}_n.^{\wedge 2}./\mathbf{K}_n$, and we are left to describe the update process.

We need $\mathbf{V}_{n+1} := \mathbf{k}_{n+1}./\sqrt{\mathbf{K}_n}$ with $\mathbf{k}_{n+1} := (K_n(y_{n+1}, x_j))_{1 \le j \le N}$. The latter can be calculated starting from $\mathbf{k}_{n+1,0} := (K(y_{n+1}, x_j))_{1 \le j \le N}$ by applying the formula

$$\begin{aligned} K_n(y_{n+1}, x_j) &= K(y_{n+1}, x_j) - \sum_{k=1}^{n} v_k(y_{n+1}) v_k(x_j), \\ \mathbf{k}_{n+1} &= \mathbf{k}_{n+1,0} - \sum_{k=1}^{n} e_{I(n+1)}^T \mathbf{V}_k \mathbf{V}_k, \\ &= \mathbf{k}_{n+1,0} - \sum_{k=1}^{n} \mathbf{V}_{k,I(n+1)} \mathbf{V}_k. \end{aligned}$$

The update of $\mathbf{K}_n$ proceeds via

$$\mathbf{K}_{n+1} = \mathbf{K}_n - \mathbf{V}_{n+1}.^{\wedge 2}$$

like in the Newton basis case. This vector, containing the values of the square of the standard power function, should be nonnegative and vanish on all points selected so far. This gives some roundoff control.

We finally need $\mathbf{L}_{n+1} = (\lambda^x K_{n+1}(x, x_j))_{1 \le j \le N}$ and apply the formulae

$$\begin{aligned} \lambda^x K_{n+1}(x, x_j) &= \lambda^x K_n(x, x_j) - \frac{\lambda^x K_n(y_{n+1}, x) K_n(y_{n+1}, x_j)}{K_n(y_{n+1}, y_{n+1})} \\ \mathbf{L}_{n+1} &= \mathbf{L}_n - \frac{\mathbf{L}_{n,I(n+1)}}{\mathbf{K}_{n,I(n+1)}} \mathbf{k}_{n+1} \end{aligned}$$

which need no additional recursive calculations.

This is enough to carry out the induction step, but we do not yet have the actual values $P_n^2(\lambda)$ bounding the RKHS error. To start with, we need the value $P_0^2(\lambda) = \lambda^x \lambda^y K(x, y)$. Then

$$P_{n+1}^2(\lambda) = P_n^2(\lambda) - R_n^2(y_{n+1})$$

is the cheap update formula. The progress of the algorithm can be monitored this way, but even without knowing $P_n^2(\lambda)$ one can stop if $R_n^2(y_{n+1})$ is smaller than a tolerance.

The above update step needs $\mathcal{O}(N \cdot n)$ operations in total, and it is the core of the whole calculation. The overall computational complexity for reaching n final

points is $\mathcal{O}(n^2N)$. Since the $\mathbf{K}, \mathbf{L}$, and $\mathbf{k}$ vectors can be overwritten, the total storage requirement is $\mathcal{O}(nN)$ for the $\mathbf{V}$ vectors.

But it is still open how to calculate the optimal weights. They can be written as $\alpha_j^* = \lambda(u_j)$ for the Lagrange basis $u_1, \ldots, u_n$ based on the selected points $y_1 = x_{I(1)}, \ldots, y_n = x_{I(n)}$, but this implies that they will change from step to step when stepping through the algorithm. Thus we refrain from giving an update formula, since updating will be as costly as an a-posteriori calculation. Instead, we propose to calculate the weights after finding the selected points. If $\tilde{\mathbf{V}}$ is the $n \times n$ matrix of the Newton basis function values on the selected points [6] (rows as points, columns as functions, i.e. a selection of rows of $\mathbf{V}$), this matrix is lower triangular and nonsingular. For the vectors and matrices below, we always use a tilde to denote restriction to the selected n points. Then the weight vector $\tilde{\mathbf{a}}$ can be obtained by solving the two triangular $n \times n$ systems

$$\begin{aligned} \tilde{\mathbf{V}}\tilde{\mathbf{z}} &= \tilde{\mathbf{L}}_0 \\ \tilde{\mathbf{V}}^T\tilde{\mathbf{a}} &= \tilde{\mathbf{z}} \end{aligned}$$

at $\mathcal{O}(n^2)$ computational complexity. This follows from the fact that the kernel matrix $\tilde{\mathbf{A}}$ on the selected points has the Cholesky factorization $\tilde{\mathbf{A}} = \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^T$, while the Lagrange form of the weights is obtained by solving $\tilde{\mathbf{A}}\tilde{\mathbf{a}} = \tilde{\mathbf{L}}_0$. Calculating the Newton basis recursively yields the $\tilde{\mathbf{V}}$ matrix for free.

A greedy algorithm for vectorial cases is in [13],and it also uses the Newton basis technique.

5 Optimal Evaluation Points

This section is a detour that turns the previous argument upside–down. Imagine that a user of a meshless method has already determined the nodes where function values are needed, selects a small set $X_n = \{x_1, \ldots, x_n\}$ of local neighbor points, and wants to find the point z where $(\Delta f)(z)$ can be best approximated using the values of f at the points of X_n. This process is repeated for all small sets of neighboring points, and the corresponding optimal z points are then used for collocation.

We thus consider approximations of functionals of the form

$$\lambda_z(f) := (L(f))(z) =: L_z^x f(x)$$

with a linear differential operator L evaluated at a point $z \in \Omega$, and we want to find a place z where the available data at $X_n = \{x_1, \ldots, x_n\}$ allows to reconstruct λ_z best.

Since we have

$$P_n^2(\lambda_z) = \lambda_z^x \lambda_z^y K(x,y) - \sum_{j=1}^{n} \lambda_z(v_j)^2,$$

we have to minimize this function over z, either on all of Ω or on a large discrete point set. This works with

$$\begin{aligned}
\lambda_z(v_k) &= \frac{\lambda_z(K_{k-1}(x,x_k))}{\sqrt{K_{k-1}(x_k,x_k)}},\ 1 \le k \le n \\
\lambda_z(K_{k+1}(x,y)) &= \lambda_z(K(x,y)) - \sum_{j=1}^{k+1} \lambda_z(v_j) v_j(y) \\
\lambda_z(K_{k+1}(x,x_k)) &= L_z^x K_k(x,x_k) - \frac{L_z^x K_k(x_{k+1},x) K_k(x_{k+1},x_k)}{K_k(x_{k+1},x_{k+1})}.
\end{aligned}$$

On N points as a function of z and stored as N–vectors $\mathbf{M}_j := (\lambda_z(v_j))$, $\mathbf{m} := (\lambda_z^x \lambda_z^y K(x,y))$, we have to find the minimum of the vector

$$\mathbf{m} - \sum_{j=1}^{n} \mathbf{M}_j .^{\wedge} \mathbf{M}_j,$$

which can be recast as a single MATLAB statement

$$\mathbf{m} - (sum((\mathbf{M} .* \mathbf{M})'))'$$

if $\mathbf{M}$ has the columns $\mathbf{M}_j$. To calculate

$$\mathbf{M}_n = (\lambda_z(v_n)) = \frac{\lambda_z^x K_{n-1}(x,x_n)}{\sqrt{K_{n-1}(x_n,x_n)}}$$

we define

$$\mathbf{m}_k := \lambda_z^x K_k(x,x_n)$$

and need a recursion starting from $\mathbf{m}_0$ and proceeding via

$$\begin{aligned}
\mathbf{m}_{k+1} &= \lambda_z^x K_{k+1}(x,x_n) \\
&= \lambda_z^x K_k(x,x_n) - \lambda_z(v_{k+1}) v_{k+1}(x_n) \\
&= \mathbf{m}_k - \mathbf{M}_{k+1} \mathbf{V}_{k+1,n}
\end{aligned}$$

to end up with

$$\mathbf{M}_n = \frac{\mathbf{m}_{n-1}}{\sqrt{K_{n-1}(x_n,x_n)}}.$$

This needs storage $\mathcal{O}(Nn)$ and $\mathcal{O}(Nn^2)$ operations.

6 Examples

6.1 Interpolation

We start our examples with optimal sparse recovery of $f(0)$ from values $f(x_j)$ for randomly scattered points near the origin, using radial kernels. In such cases it is clear that R_n^2 is maximal at $z = 0$ and equals $P_n^2(\delta_0)$ there, but z will usually not be among the x_j. In many cases, R_n^2 has a sharp single peak at $z = 0$, and R_n^2 vanishes at the n points selected so far, because there is no error decrease when re–selecting one of the already chosen points. This means that the algorithm will prefer neighbors of z at first, and the local behavior of R_n around $z = 0$ will not change much after all nearest neigbors are chosen. Plenty of examples show this behavior.

Figures 1 to 3 show three examples picking 15 out of 75 offered points, depicted in the middle plot with circles around the selected points. The method prefers near neighbors to $z = 0$, but does not stricly take the nearest neighbors. For kernels with finite smoothness, it turns out to make no sense to take many points, in contrast to the Gaussian case. Also, the preference of near neighbors is stronger for non–smooth kernels. In view of the theory of flat limits [2, 9], this is not surprising, because there everything behaves much like an approximation by polynomials, and good point choices should be unisolvent for high–degree polynomials. All kernels were used at scale 1 in $[-1,1]^2$.

6.2 Integration

The next series of examples concerns integration. If we take λ to be the global integral over all of $\mathbb{R}^2$, and if we use globally integrable radial kernels, the algorithm will work. The integration formula always yields the exact integral over the interpolant, and this always exists, though the functional λ is not in the dual of the Hilbert space, since $\|\lambda\|_H^2 = \lambda^x\lambda^y K(x,y) = P_0^2(\lambda)$ is infinite. The R_n function is still defined, and one can pick the maximum of R_n over all available points. Formally, the algorithm is used with the finite constant $K(x,x)$ as a replacement for $P_0^2(\lambda)$, and since the R_n values are large and subtracted from this, the "error norm" in the following figures gets negative.

In these cases, the selected points spread over the domain, and the R_n function is large near the boundary. A point in the upper left corner is desperately needed (but not available), and using more points in the interior will not pay off.

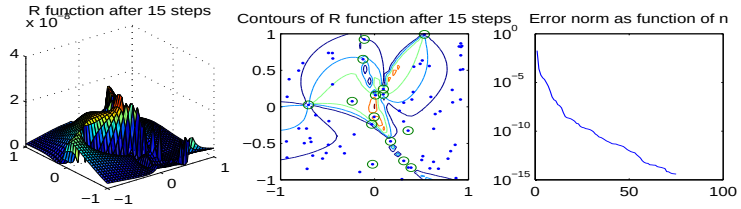

Figure 1: Gaussian kernel, interpolation, 15 out of 75 points

The next series of experiments in Figures 7 to 9 uses the finite integral

$$\lambda(f) = \int_{-1}^{+1} \int_{-1}^{+1} f(x,y)dxdy.$$

The results are similar to the case with the infinite integral. The examples were all bound to work with only 15 selected points in order not to overload the graphics. Of course, 15 points are not enough to yield a good integration accuracy.

6.3 Differentiation

Again, we use the same 75 offered data locations and the same three kernels, but this time we approximate the functional $\lambda(f) := (\Delta f)(0)$. The results are in figures 10 to 12. For the Wendland kernel. we had to go C^4 and scale 10. Note that sparsity of final collocation matrices does not necessarily require compactly supported kernels. It suffices to use sparse approximations to each functional. This is particularly important for localized meshless methods fo solving PDEs. Examples for such techniques are in [7, 12, 14, 15, 16], for instance.

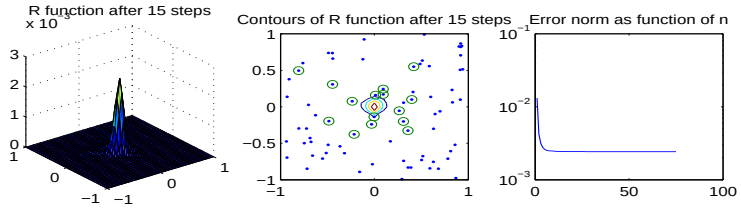

Figure 2: Wendland C^2 kernel, interpolation, 15 out of 75 points

7 Conclusion

The proposed greedy selection algorithm for nodes that yield good approximations to linear functionals is computationally efficient and yields optimal results, error–wise. Of course, a total minimization of (1) with respect to weights *and* nodes would yield even smaller errors, but it is computationally much too involved.

For interpolation and evaluation of derivatives, the strong localization of the R_n functions near the evaluation point z supports an observation already made in [11], namely that choosing nearest neighbors is a good suboptimal strategy.

References

[1] O. Davydov and R. Schaback. Error bounds for kernel-based numerical differentiation. Draft, 2013.

[2] T. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Comput. Math. Appl.*, 43:413–422, 2002.

[3] D. Mirzaei, R. Schaback, and M. Dehghan. On generalized moving least squares and diffuse derivatives. *IMA J. Numer. Anal.*, 32, No. 3:983–1000, 2012. doi: 10.1093/imanum/drr030.

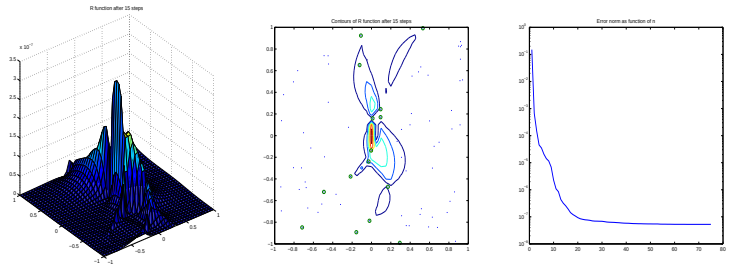

Figure 3: Matern kernel $m = 5$, interpolation, 15 out of 75 points

[4] M. Mouattamid and R. Schaback. Recursive kernels. *Analysis in Theory and Applications*, 25:301–316, 2009.

[5] S. Müller and R. Schaback. A Newton basis for kernel spaces. *Journal of Approximation Theory*, 161:645–655, 2009. doi:10.1016/j.jat.2008.10.014.

[6] M. Pazouki and R. Schaback. Bases for kernel-based spaces. *Computational and Applied Mathematics*, 236:575–588, 2011.

[7] B. Šarler. From global to local radial basis function collocation method for transport phenomena. In *Advances in meshfree techniques*, volume 5 of *Comput. Methods Appl. Sci.*, pages 257–282. Springer, Dordrecht, 2007.

[8] R. Schaback. Reconstruction of multivariate functions from scattered data. Manuscript, available via http://www.num.math.uni-goettingen.de/schaback/research/group.html, 1997.

[9] R. Schaback. Limit problems for interpolation by analytic radial basis functions. *J. Comp. Appl. Math.*, 212:127–149, 2008.

[10] R. Schaback. A computational tool for comparing all linear PDE solvers. submitted, http://www.num.math.uni-goettingen.de/schaback/research/group.html, 2013.

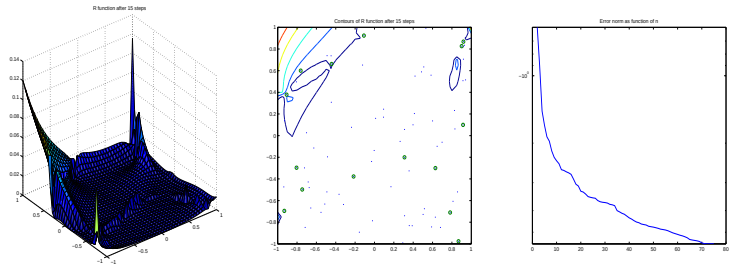

Figure 4: Gaussian kernel, infinite integration, 15 out of 75 points

[11] R. Schaback. Direct discretizations with applications to meshless methods for PDEs. submitted, http://www.num.math.uni-goettingen.de/schaback/research/group.html, 2013.

[12] R. Vertnik and B. Šarler. Local collocation approach for solving turbulent combined forced and natural convection problems. *Adv. Appl. Math. Mech.*, 3(3):259–279, 2011.

[13] D. Wirtz and B. Haasdonk. A vectorial kernel orthogonal greedy algorithm. Preprint, Stuttgart Research Centre for Simulation Technology, 2012.

[14] G. Yao, B. Šarler, and C. S. Chen. A comparison of three explicit local meshless methods using radial basis functions. *Eng. Anal. Bound. Elem.*, 35(3):600–609, 2011.

[15] G. Yao, S. ul Islam, and B. Šarler. A comparative study of global and local meshless methods for diffusion-reaction equation. *CMES Comput. Model. Eng. Sci.*, 59(2):127–154, 2010.

[16] G. Yao, S. ul Islam, and B. Šarler. Assessment of global and local meshless methods based on collocation with radial basis functions for parabolic partial differential equations in three dimensions. *Eng. Anal. Bound. Elem.*, 36(11):1640–1648, 2012.

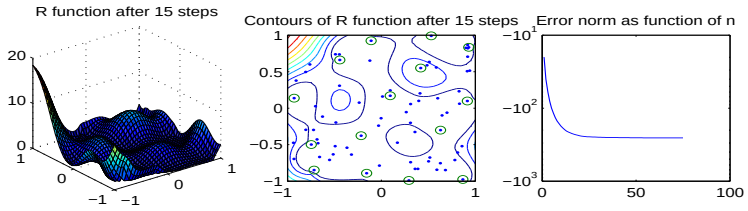

Figure 5: Wendland C^2 kernel, infinite integration, 15 out of 75 points

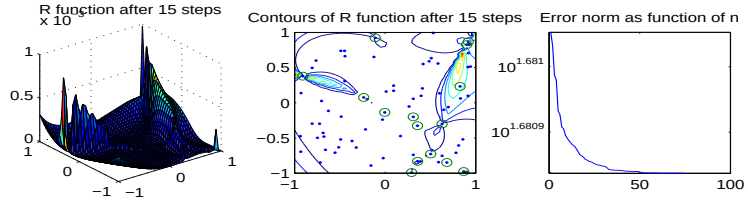

Figure 6: Matern kernel $m = 5$, infinite integration,15 out of 75 points

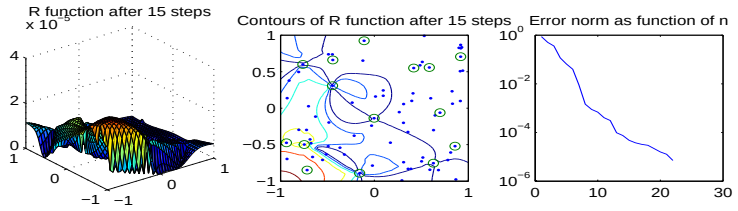

Figure 7: Gaussian kernel, finite integration, 15 out of 75 points

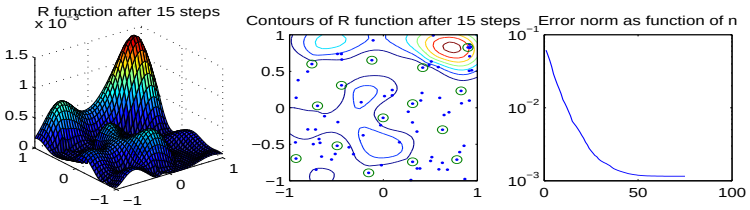

Figure 8: Wendland C^2 kernel, finite integration, 15 out of 75 points

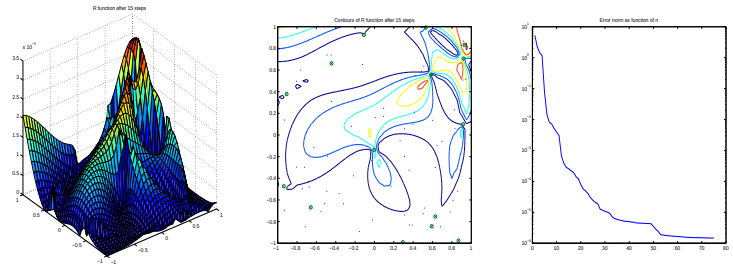

Figure 9: Matern kernel $m = 5$, finite integration, 15 out of 75 points

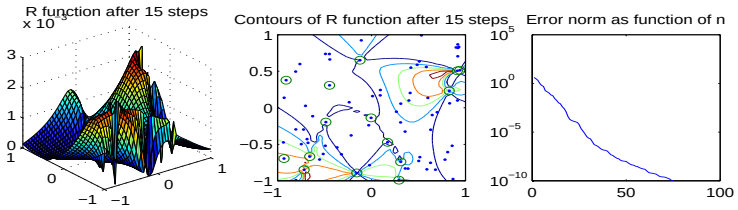

Figure 10: Gaussian kernel, Laplacian, 15 out of 75 points

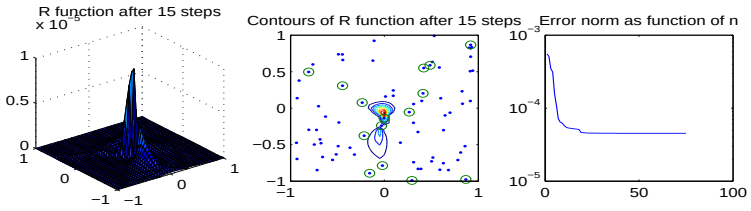

Figure 11: Wendland C^4 kernel, scale 10, Laplacian, 15 out of 75 points

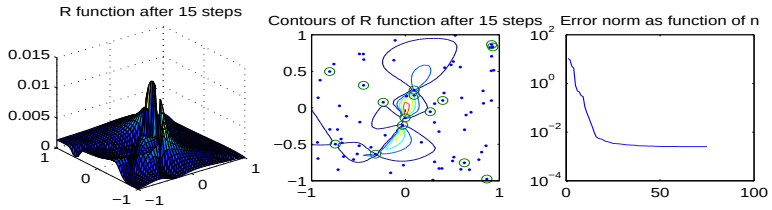

Figure 12: Matern kernel $m = 5$, Laplacian, 15 out of 75 points