Kernel-Based Meshless Methods

R. Schaback and H. Wendland

Abstract

Abstract.
...??????... to be supplied.later ..??????...

Contents

1 Introduction

This article can be seen as an extension of Martin Buhmann's presentation of radial basis functions [30] in this series. But we shall take a somewhat wider view and deal with *kernels* in general, focusing on their recent applications in areas like *machine learning* and *meshless methods* for solving partial differential equations.

In their simplest form, kernels may be viewed as bell-shaped functions like Gaussians. They can be shifted around, dilated, and superimposed with weights in order to form very flexible spaces of multivariate functions having useful properties. The literature presents them under various names in contexts of different numerical techniques, for instance as *radial basis functions*, *generalized finite elements*, *shape functions* or even *particles*. They are useful both as *test functions* and *trial functions* in certain *meshless methods* for solving partial differential equations, and they arise naturally as *covariance kernels* in probabilistic models. Kernels replace radial basis functions because of their greater generality and wider applicability. In case of learning methods, sigmoidal functions within neural networks were successfully superseded by radial basis functions, but now they are both replaced by *kernel machines*[1] to implement the most successful algorithms for *Machine Learning* [81, 84]. Even the term *Kernel Engineering* has been coined recently, because efficient learning algorithms require specially tailored application-dependent kernels.

With this slightly chaotic background in mind, we survey the major application areas while focusing on a few central issues that can serve as guidelines for practical work with kernels. Section 2 (`SecKer`) starts with a general definition of kernels and provides a short account of their properties. The main reasons for using kernels at all will be described in Section 3 (`SecOR`) starting from their ability to recover functions optimally from given unstructured data. At this point, the connections between kernel methods for interpolation, approximation, learning, pattern recognition, and PDE solving become apparent. The probabilistic aspects of kernel techniques follow in Section 4 (`SecKiPM`), while practical guidelines for constructing new kernels follow in Section 5 (`SecKC`). Special application-oriented kernels are postponed to Section 6 (`SecSK`) to avoid too much detail at the beginning.

Since one of the major features of kernels is to generate spaces of trial functions with excellent approximation properties, we devote Section 7 (`SecAK`) to give a short account of the current results concerning such questions. Together with strategies to handle large and ill-conditioned systems (Section 8 (`SecHLS`)), these results are of importance to the applications that follow later.

After a short interlude on kernels on spheres in Section 9 (`SecKoS`) we start our survey of applications in Section 10 (`SecAKI`) by looking at interpolation problems first. These take advantage of the abilities of kernels to handle unstructured Birkhoff-type data while producing solutions of arbitrary smoothness and high accuracy. Then we review kernels in modern learning algorithms, but we can keep this section short because there are good books on the subject.

Section 12 (`SecMM`) surveys meshless methods [25] for solving partial differential equations. It describes the different techniques currently sailing under this flag, and it points out where and how kernels occur there. Due to an existing survey [9] in this series, we keep the generalized finite element method short here, but we incorporate meshless local Petrov-Galerkin techniques [3].

The final two sections are then focusing on purely kernel-based meshless methods. We treat applications of symmetric and unsymmetric collocation, of kernels providing fundamental and particular solutions, and provide the state-

[1] `http://www.kernel-machines.org`

of-the-art of their mathematical foundation.

Altogether, we want to keep this survey digestible for the non-expert and casual reader who wants to know roughly what happened so far in the area of application-oriented kernel techniques. This is why we omit most of the technical details and focus on the basic principles. Consequently, we have to refer as much as possible to background reading for proofs and extensions. Fortunately, there are two recent books [31, 95] which contain the core of the underlying general mathematics for kernels and radial basis functions. For kernels in learning theory, we already cited two other books [81, 84] providing further reading. If we omit pointers to proofs, these four books will usually contain what is needed.

Current books and survey articles in the area of meshless methods are focusing either on certain classes of methods or on theory or on applications. We shall cite them whenever they come handy, but it is not advisable for the reader to use just one or two of them. Likewise, the list of references cannot contain all available papers on all possible kernel applications. This forces us to select a very small subset, and our main selection criterion is how a certain reference fits into the current line of argument at a certain place of this survey. Incorporation or omission of a certain publication does not express our opinion on its importance in general.

2 Kernels

(`SecKer`)

Definition 2.1 *(`DefKer`) A* kernel *is a function of the form*

$$K \; : \; \Omega \times \Omega \to \mathbf{R}$$

where Ω can be an arbitrary nonempty set.

Some readers may consider this as being far too general. However, in the context of learning algorithms, the set Ω defines the possible learning inputs. Thus Ω should be general enough to allow Shakespeare texts or X-ray images, *i.e.* Ω should better have no predefined structure at all. Thus the kernels occurring in machine learning are extremely general, but still they take a special form which can be tailored to meet the demands of applications. We shall explain the recipe for their determination npw.

First, before a kernel is available, an application-dependent *feature map* $\Phi : \Omega \to \mathcal{K}$ with values in a Hilbert space $\mathcal{K}$ is defined. It should provide for each $x \in \Omega$ a large collection $\Phi(x)$ of *features* of x which are characteristic for x and which live in the Hilbert space $\mathcal{K}$ of high or even infinite dimension.

Guideline 2.2 *(`guifeamapstolin`) Feature maps $\Omega \to \mathcal{K}$ allow to apply linear techniques in their range $\mathcal{K}$, while their domain Ω is an unstructured set. They should be chosen carefully in an application-dependent way, capturing the essentials of elements of Ω.*

With a feature map Φ at hand, there is a kernel

eqkerler

$$K(x, y) := (\Phi(x), \Phi(y))_{\mathcal{K}} \text{ for all } x, y \in \Omega. \tag{1}$$

In another important class of cases, the set Ω consists of random variables. Then the *covariance* between two random variables x and y from Ω is a standard choice of a kernel. These and other kernels arising in nondeterministic settings will be the topic of Section 4 (SecKiPM). The connection to learning is obvious: two learning inputs x and y from Ω should be very similar, iff (1, eqkerler) takes large positive values. These examples already suggest

Definition 2.3 *(*DefKerSym*) A kernel K is* symmetric*, if $K(x, y) = K(y, x)$ holds for all $x, y \in \Omega$.*

A kernel K on Ω defines a function $K(\cdot, y)$ for all fixed $y \in \Omega$. This allows to generate and manipulate spaces of functions on Ω via

eqk0

$$\mathcal{K}_0 := \text{ span } \{K(\cdot, y) \ : \ y \in \Omega\}. \tag{2}$$

For the learning theory case, the function $K(\cdot, y) = (\Phi(\cdot), \Phi(y))_{\mathcal{K}}$ relates each other input object to a fixed object y via its essential features. But in general $\mathcal{K}_0$ just provides a handy linear space of *trial* functions on Ω which is extremely useful for most applications of kernels, *e.g.* when Ω consists of texts or images. For example, in meshless methods for solving partial differential equations, certain finite-dimensional subspaces of $\mathcal{K}_0$ are used as *trial* spaces to furnish good approximations to the solutions.

In certain other cases, the set Ω carries a measure μ, and then, under reasonable assumptions like $f, \ K(y, \cdot) \in L^2(\Omega, \mu)$, the generalized *convolution*

eqkerconv

$$K *_\Omega f := \int_\Omega f(x) K(\cdot, x) d\mu(x) \tag{3}$$

defines an integral transform $f \mapsto K *_\Omega f$ which can be very useful. Note that Fourier or Hankel transforms arise this way, and recall the rôle of the Dirichlet kernel in Fourier analysis of univariate periodic functions. The above approach to kernels via convolution works on locally compact topological groups using Haar measure, but we do not want to pursue this detour into abstract harmonic analysis too far. For space reasons, we also have to exclude complex-valued kernels and all transform-type applications of kernels here, but it should be pointed out that wavelets are special kernels of the above form, defining the *continuous wavelet transform* this way.

Note that discretization of the integral in the convolution transform leads to functions in the space $\mathcal{K}_0$ from (2, eqk0). Using kernels as trial functions can be viewed as a discretized convolution. This is a very useful fact in the theoretical analysis of kernel-based techniques.

Guideline 2.4 *(`guikercontri`) Kernels have two major application fields: they generate convolutions and trial spaces. These two are related by discretization.*

Another important aspect in all kernel-based techniques is the *scaling problem*. If the kernel K in the convolution equation (3, `eqkerconv`) is a sharp nonnegative spike with integral one, the convolution will reproduce f approximately, and the distributional "delta kernel" will reproduce f exactly. This is theoretically nice, but discretization will need a very fine spatial resolution. On the other hand, convolution with a nonnegative smooth kernel of wide or infinite support acts as a *smoothing operator* which will not have a good reproduction quality. To control this tradeoff between approximation and smoothing, many kernel applications involve a free scaling parameter, and it is a serious problem to derive good strategies for its determination. The scaling problem will come up at various places in this article.

For many applications, the space $\mathcal{K}_0$ needs more structure. In fact, it can be turned into a Hilbert space via

Definition 2.5 *(`DefKerPD`) A symmetric kernel K is* positive (semi-) definite*, if for all finite subsets $X := \{x_1, \ldots, x_N\}$ of Ω the symmetric matrices $A_{K,X}$ with entries $K(x_j, x_k)$, $1 \le j, k \le N$ are positive (semi-) definite.*

For a symmetric positive definite kernel K on Ω, the definition

`eqKKK`

$$(K(x,\cdot), K(y,\cdot))_{\mathcal{K}} = (K(\cdot,x), K(\cdot,y))_{\mathcal{K}} := K(x,y) \text{ for all } x, y \in \Omega \qquad (4)$$

of an inner product of two generators of $\mathcal{K}_0$ easily generalizes to an inner product on all of $\mathcal{K}_0$ such that

`eqKKnorm`

$$\left\| \sum_{j=1}^{N} \alpha_j K(\cdot, x_j) \right\|_{\mathcal{K}}^2 := \sum_{j,k=1}^{N} \alpha_j \alpha_k K(x_j, x_k) = \alpha^T A_{K,X} \alpha \qquad (5)$$

defines a *numerically accessible* norm on $\mathcal{K}_0$ which allows to construct a *native* Hilbert space

`eqk`

$$\mathcal{K} := \text{ clos } \mathcal{K}_0 \qquad (6)$$

as the completion of $\mathcal{K}_0$ under the above norm. In most cases, the space $\mathcal{K}$ is much richer than $\mathcal{K}_0$ and does not seem to have any explicit connection to the kernel it is generated from. For instance, Sobolev spaces $\mathcal{K} = W_2^k(\mathbf{R}^d)$ with $k > d/2$ result from the kernel

`eqSobK`

$$K(x,y) = \|x-y\|_2^{k-d/2} K_{k-d/2}(\|x-y\|_2) \qquad (7)$$

where K_ν is the Bessel function of third kind. Starting from (7, `eqSobK`) it is not at all clear that a certain closure of the span of all translates of K generates the Sobolev space $W_2^k(\mathbf{R}^d)$.

By construction, the spaces $\mathcal{K}$ and $\mathcal{K}_0$ have a nice structure now, and there is a *reproduction property*

`eqrepro`

$$f(x) := (f, K(\cdot, x))_{\mathcal{K}} \text{ for all } f \in \mathcal{K},\ x \in \Omega \tag{8}$$

for all functions in $\mathcal{K}$. At this point, we are on the classical grounds of *reproducing kernel Hilbert spaces* (RKHS) with a long history [2, 65, 7]. We shall deal with *conditionally* positive definite kernels in Section 6 (`SecSK`).

Guideline 2.6 *(*`guiposdefker`*) Positive definite kernels are* reproducing *all functions from their associated* native *Hilbert space. On the trial space* (2, `eqk0`) *of translated positive definite kernels, the Hilbert space norm can be* numerically calculated *by plain kernel evaluations, without integration or derivatives. This is particularly useful if the Hilbert space norm theoretically involves integration and derivatives,* e.g. *in case of Sobolev spaces.*

Guideline 2.7 *(*`guiinvarker`*) If the set Ω has some additional geometric structure, kernels may take a simplified form, making them* invariant *under geometric transformations on Ω.*

For instance, kernels of the form

$K(x-y)$	are *translation-invariant* on Abelian groups
$K(x^Ty)$	are *zonal* on multivariate spheres
$K(\\|x-y\\|_2)$	are *radial* on $\mathbf{R}^d$

with a slight abuse of notation. Radial kernels are also called *radial basis functions*, and they are widely used due to their invariance under Euclidean (rigid-body-) transformations in $\mathbf{R}^d$. The most important example in the *Gaussian* kernel

`eqGauss`

$$K(x,y) := \exp(-\|x-y\|_2^2) \text{ for all } x, y \in \mathbf{R}^d \tag{9}$$

which is symmetric positive definite on $\mathbf{R}^d$ for all space dimensions, and which naturally arises as a convolution kernel and a multivariate probability density.

In many applications, for instance in machine learning, the kernel value $K(x,y)$ increases with the "similarity" of x and y, like a correlation or a covariance, and bell-shaped like the Gaussian. More precisely, any symmetric positive definite kernel generates a *distance metric* $d\ :\ \Omega \times \Omega \to [0, \infty)$ via

`eqdistmet`

$$d^2(x,y) := K(x,x) - 2K(x,y) + K(y,y) \text{ for all } x, y \in \Omega \tag{10}$$

on a general set [82, 88], but this is not our major concern here, because we want to focus on applications.

Guideline 2.8 *(*`guikermet`*) Symmetric positive definite kernels introduce some kind of "geometry" on the underlying set Ω.*

The art of kernel engineering is to find a kernel that nicely models similarity or dissimilarity of two items x and y via $K(x, y)$ for a given application.

3 Optimal Recovery

(`SecOR`) One of the key advantages of kernels is the following

Guideline 3.1 *(*`guioptuse`*) Kernel-based methods can make optimal use of the given information.*

Results like this come up at various places in theory and applications, and they have a common background linking them to the interesting field of *information-based complexity*[2] [90]. In a probabilistic context, Guideline 3.1 (`guioptuse`) can be forged into an exact statement using Bayesian arguments, but we want to keep things simple first and postpone details to Section 4 (`SecKiPM`).

3.1 Recovery From Unstructured Data

For illustration, we assume that we want to model a black-box transfer mechanism

$$x \overset{f}{\mapsto} f(x)$$

...??????.... Bild...??????.....

that replies to an input $x \in \Omega$ by an output $f(x) \in \mathbf{R}$. This is the same as the reaction $f(x)$ of a well-trained individual or machine to a given stimulus x given to it. Finding a good response mechanism f can be called *learning* or *black-box modeling*. If the output should take only a finite number of possible values, this is *pattern recognition* or *classification*. We shall use the word "recovery problem" to summarize all of these situations, which mathematically require the determination of a function. But we want to stick to an application-oriented view here.

At this point we do not have any further information on the model or the intended reactions to the stimuli. But usually we have some examples of "good behaviour" that can be used. These take the form of a sequence $(x_1, y_1), \ldots, (x_N, y_N)$ of unstructured *training data*, pairing inputs $x_j \in \Omega$ with their expected responses $y_j \in \mathbf{R}$. The recovery task now is to find a function f such that

`eqfapp`

$$f(x_j) \approx y_j,\ 1 \le j \le N \tag{11}$$

and this is a standard interpolation or approximation problem, though posed on an unstructured set Ω using unstructured data.

[2] `http://www.ibc-research.org`

If we slightly change the meaning of the word "data", we can try to find a smooth function f such that

eqpoisson

$$\begin{array}{rcll} (-\Delta f)(y_j) & \approx & \varphi(y_j), & 1 \leq j \leq M \\ f(z_k) & \approx & \psi(z_k), & M+1 \leq k \leq N \end{array} \tag{12}$$

where $y_1, \ldots, y_M$ are points in a bounded domain Ω while $z_{M+1}, \ldots, z_N$ lie on the boundary. This would hopefully provide an approximate solution f to the Poisson problem

$$\begin{array}{rcll} (-\Delta f)(y) & \approx & \varphi(y), & y \in \Omega \\ f(z) & \approx & \psi(z), & z \in \partial\Omega \end{array}$$

for given functions φ on Ω and ψ on $\partial\Omega$. Note that this *collocation* technique is again a recovery problem for a function f from certain of its data, just replacing point evaluations in (11, `eqfapp`) by evaluations of certain derivatives. In general, one can replace (11, `eqfapp`) by

eqlapp

$$\lambda_j(f) \approx y_j, \ 1 \leq j \leq N \tag{13}$$

for a set of given linear *data functionals* $\lambda_1, \ldots, \lambda_N$ generalizing the point evaluation functionals $\delta_{x_1}, \ldots, \delta_{x_N}$ of (11, `eqfapp`). Tradition in Approximation Theory would call this a recovery problem from *Hermite-Birkhoff* data, if the data functionals are evaluations of derivatives at points. But there are much more general functionals, *e.g.* the ones defining *weak data* via

$$\lambda_j(f) = \int_\Omega \nabla^T f \cdot \nabla v_j$$

like in finite elements, using a *test function* v_j. This way, finite element methods for solving linear partial differential equations can be written as recovery problems (13, `eqlapp`).

For later sections of this article, the reader should keep in mind that suitable generalizations (13, `eqlapp`) of the recovery problem (11, `eqfapp`) lead to methods for solving partial differential equations. We shall stick to the simple form of (11, `eqfapp`) for a while, but when overlooking large parts of Numerical Analysis, *e.g.* finite element techniques, we have to state

Guideline 3.2 *(*`guirec`*) Many applications can be rephrased as* recovery problems *for functions from unstructured data.*

3.2 Generalization

The resulting model function f should be such that it *generalizes* well, *i.e.* it should give practically useful responses $f(x)$ to new inputs $x \in \Omega$. Furthermore, it should be *stable* in the sense that small changes in the training data do not change f too much. But these goals are in conflict with good reproduction of the training data. A highly stable but useless model would be $f = 1$, while *overfitting* occurs if there is too much emphasis on data reproduction, leading to unstable models with bad generalization properties.

Guideline 3.3 *(*`guirepgendil`*) Recovery problems are subject to the* reproduction-generalization dilemma *and need a careful balance between generalization and stability properties on one hand and data reproduction quality on the other.*

This is called the *bias-variance dilemma* under certain probabilistic hypotheses, but it also occurs in deterministic settings.

Given a recovery problem as in (11, `eqfapp`), there is not enough information to come up with a useful solution of the recovery problem. In particular, we have no idea how to define f or from which space of functions to pick it from. From a theoretical point of view, we are facing an *ill-posed problem* with plenty of indistinguishable approximate solutions. From a practical point of view, all mathematical a-priori assumptions on f are useless.

Instead, one should use additional application-dependent information about the essentials of the inputs, *e.g.* define a *feature map* $\Phi : \Omega \to \mathcal{H}$ like in (1, `eqkerler`) taking an object x to an object $\Phi(x)$ in $\mathcal{H}$ containing all essential features of x. With this additional information, we can define a kernel K using (1, `eqkerler`), and we get a space $\mathcal{K}$ of functions on Ω via (2, `eqk0`) and (6, `eqk`). Since $\mathcal{K}$ usually comes out to be rather large (see the example in (7, `eqSobK`) for Sobolev spaces), this space serves as a natural reservoir to pick f from, and if we have no other information, there is no other choice for a space defined on all of Ω. Of course, the choice of a feature map is just another way of adding hypotheses, but it is one that can be tailored perfectly for the application, using kernel engineering knowledge.

3.3 Optimality

We are now left with the problem to pick f somehow from the space $\mathcal{K}$, using our training set. If we insist on exact recovery, we get an instance of Guideline 3.1 (`guioptuse`) from

Theorem 3.4 *(*`TheBBMOR`*) Let the kernel K be symmetric positive definite. Then a function of the form*

`eqfsimp`

$$f^* := \sum_{k=1}^{N} \alpha_k K(\cdot, x_k) \tag{14}$$

is the unique minimizer of the Hilbert space norm in $\mathcal{K}$ under all functions $f \in \mathcal{K}$ with $f(x_j) = y_j$, $1 \le j \le N$. The coefficients α_k can be calculated from the linear system

`eqintsys`

$$\sum_{k=1}^{N} \alpha_k K(x_j, x_k) = y_j, \quad 1 \le j \le N. \tag{15}$$

As Section 4 (`SecKiPM`) will show, the system (15, `eqintsys`) also arises for different nondeterministic recovery problems in exactly the same way, but with different semantics.

Clearly, symmetric positive definiteness of the kernel implies positive definiteness of the *kernel matrix* $A_{K,X}$ we already saw in Definition 2.5 (`DefKerPD`).

Guideline 3.5 *(*`gui`*) Interpolation of unstructured data using a kernel is an optimal strategy for black-box modelling and learning from noiseless information.*

The essential information on the application is built into the kernel. Once the kernel is there, things are simple, theoretically. The *generalization error* is optimal in the following sense:

Theorem 3.6 *(*`Thepwoptrec`*) Consider all possible linear recovery schemes of the form*

$$f_\alpha(\cdot) := \sum_{j=1}^{N} \alpha_j(\cdot) f(x_j)$$

which use the training data $(x_j, y_j) = (x_j, f(x_j))$, $1 \leq j \leq N$ for an unknown model $f \in \mathcal{K}$ and employ arbitrary functions α_j on Ω. Then the approximate solution f^* of Theorem 3.4 (`TheBBMOR`*) satisfies*

`eqlagrep`

$$\inf_{\alpha} \sup_{\|f\|_{\mathcal{K}} \leq 1} |f(x) - f_\alpha(x)| = \sup_{\|f\|_{\mathcal{K}} \leq 1} |f(x) - f^*(x)| \text{ for all } x \in \Omega \tag{16}$$

and it has the form $f^* = f_{u^*}$ with Lagrange-type functions $u_1^*(x), \dots, u_N^*(x)$ from span $\{K(\cdot, x_j) \;:\; 1 \leq j \leq N\}$ satisfying

`eqlagsys`

$$\sum_{j=1}^{N} u_j^*(x) K(x_j, x_k) = K(x, x_k),\ 1 \leq k \leq N,\ \text{for all } x \in \Omega. \tag{17}$$

Note that this is another instance of Guideline 3.1 (`guioptuse`). The optimality results of the previous theorems are well-known properties of univariate splines.

Guideline 3.7 *(*`guispli`*) In the context of optimal recovery, kernel methods provide natural multivariate extensions of classical univariate spline techniques.*

For later reference in Section 4 (`SecKiPM`), we should explain the connection between the linear systems (15, `eqintsys`) and (17, `eqlagsys`) on one hand, and the representations (14, `eqfsimp`) and (16, `eqlagrep`) on the other hand. Theorem 3.4 (`TheBBMOR`) works on the basis $K(\cdot, x_k)$ directly, while Theorem 3.6 (`Thepwoptrec`) produces a new basis of functions u_j^* which has the Lagrangian property $u_j^*(x_k) = \delta_{jk}$ but spans the same space. The optimal recovery solutions coincide, but have different basis representations. Transition to a local Lagrange basis is one of the possible preconditioning strategies [**?**], and approximate Lagrangian bases yield *quasi-interpolants* [**?**] which avoid solving linear systems because they provide approximate inverses. This is a promising research area.

If the recovery problem (11, `eqfapp`) is generalized to (13, `eqlapp`), there is a similar theory [100] concerning optimal recovery, replacing the kernel matrix with entries $K(x_j, x_k)$ by a symmetric matrix with elements $\lambda_j^x \lambda_k^y K(x,y)$, where we used an upper index x at λ^x to indicate that the functional λ acts with respect to the variable x. The system (15, `eqintsys`) goes over into

`eqHBlinsys`

$$\sum_{k=1}^{N} \alpha_k \lambda_j^x \lambda_k^y K(x,y) = y_j, \ \ 1 \leq j \leq N, \tag{18}$$

while (17, `eqlagsys`) will be

$$\sum_{j=1}^{N} u_j^*(x) \lambda_j^x \lambda_k^y K(x,y) = \lambda_k^y K(x,y), \ 1 \leq k \leq N, \text{ for all } x \in \Omega.$$

When using this approach for the recovery problem (12, `eqpoisson`), we get a symmetric meshless *collocation* technique for solving Poisson's equation. This will be treated in more detail in Section 15 (`SecMC`).

Let us go back to the generalization error. We shall see in Section 7 (`SecAK`) that the generalization error of kernels on $\mathbf{R}^d$ dramatically improves with their smoothness while still maintaining applicability to recovery problems with unstructured data. This is one of the key features of kernel techniques.

Guideline 3.8 *(`guismaerr`) Methods based on fixed kernels can provide recovery techniques with very small errors.*

3.4 Condition and Stability

But the small generalization error comes at a high price, because there are serious practical problems with systems of the form (15, `eqintsys`). This is in sharp contrast to the encouraging optimality properties stated so far. The systems can be very large, non-sparse and severely ill-conditioned. However, the latter is no surprise because the method solves an ill-posed problem approximatively. Thus the bad condition of the system (15, `eqintsys`) must be expected somehow. There is an apparent link between condition and scaling, since kernels with small supports will lead to approximately diagonal kernel matrices, while kernels with wide scales produce matrices with very similar rows and columns.

Guideline 3.9 *(`guismallscale`) Kernels with small scales lead to better matrix condition than kernels with wide scales.*

Since we know that kernel systems are solvable, we have

Guideline 3.10 *(`guiintreg`) Methods based on positive definite kernels have a built-in* regularization*.*

In fact, they solve the ill-posed problem (11, `eqfapp`) by providing an approximative solution minimizing the Hilbert space norm in $\mathcal{K}$ under all conceivable exact recovery schemes there, so that they act like using a regularizing penalty term of the form $\|f\|^2_{\mathcal{K}}$ which can be a Sobolev space norm for certain kernels. This regularization property will come up later when we use kernels in collocation techniques for solving partial differential equations. If (15, `eqintsys`) is viewed as an approximate solution of the integral equation

$$\int_\Omega \alpha(x)K(y,x)dx = f(y) \text{ for all } y \in \Omega$$

via a quadrature formula, we have another aspect telling us that (15, `eqintsys`) solves an ill-posed problem approximately via some regularization in the background. Note the connection to convolution (3, `eqkerconv`).

The generalization error $f(x) - f^*(x)$ and the condition of the system (15, `eqintsys`) have an unexpected connection. Theoretical results [79] and Experiments with various kernels show

Guideline 3.11 *(*`guiunc`*) Increasing smoothness of kernels on* $\mathbf{R}^d$ *decreases the recovery error but increases the condition of the system* (15, `eqintsys`)*. There are no kernels that provide small errors and good condition simultaneously.*

Guideline 3.12 *(*`guiwidescale`*) Increasing the scale of a kernel on* $\mathbf{R}^d$ *decreases the recovery error but increases the condition of the system* (15, `eqintsys`)*.*

Note that this limits the use of systems like (15, `eqintsys`) in their original form, but techniques like preconditioning [**?**] may be applied to change the systems at the cost of tolerable additional errors.

3.5 Relaxation and Complexity

Furthermore, if N is huge, the exact solution of a system (15, `eqintsys`) in the form (14, `eqfsimp`) is much too complex to be useful. This is where another general rule comes up:

Guideline 3.13 *(*`guirerereco`*) Within kernel methods, relaxation of requirements can lead to reduction of complexity.*

Under certain probabilistic hypotheses, this is another aspect of the *bias-variance dilemma* related to *overfitting.* As we already mentioned at the beginning, insisting on exact reproduction of huge amounts of data increases the complexity of the model and makes it very sensible to changes in the training data, thus less reliable as a model. Conversely, relaxing the reproduction quality will allow a simpler model. Before we turn to specific relaxation methods used in kernel-based learning, we should look back at Guideline 3.11 (`guiunc`) to see that badly conditioned large systems of the form (15, `eqintsys`) using smooth kernels will often have subsystems that provide good approximate solutions to the full system. This occurs if the generalization error is small when going over from the training data of a subset to the full training data. Thus Guideline 3.13 (`guirerereco`) may be applied by simply taking a small suitable subset of the data, relying on Guideline 3.11 (`guiunc`). As we shall see, this has serious implications for kernel-based techniques for solving partial differential equations. For simple cases, the following suffices:

Guideline 3.14 *(*`guipivot`*) Within kernel methods, large and ill-conditioned systems often have small and better conditioned subsystems furnishing good approximate solutions to the full system. Handling numerical rank loss by intelligent pivoting is useful.*

However, large problems need special treatment, and we shall deal with such cases in Section 8 (`SecHLS`).

The relaxation of (15, `eqintsys`) towards (11, `eqfapp`) can be done in several ways, and learning theory uses *loss functions* to quantify the admissible error in (11, `eqfapp`). We present this in Section 11 (`SecAL`) in more detail. Let us look at a simple special case. We allow a uniform tolerance ϵ on the reproduction of the training data, *i.e.* we impose the linear constraints

`eqlinconstr`

$$-\epsilon \leq f(x_j) - y_j \leq \epsilon,\ 1 \leq j \leq N. \tag{19}$$

We then minimize $\|f\|_{\mathcal{K}}$ while keeping ϵ fixed, or we minimize the weighted objective function $\frac{1}{2}\|f\|_{\mathcal{K}}^2 + C\epsilon$ when ϵ is varying and C is fixed. Optimization theory then tells us that the solution f^* is again of the form (14, `eqfsimp`), but the Kuhn-Tucker conditions imply that the sum only contains terms where the constraints in (19, `eqlinconstr`) are *active*, *i.e.* $\alpha_k \neq 0$ holds only for those k with $|f(x_k) - y_k| = \epsilon$. In view of principle 3.8 (`guismaerr`) these *support vectors* will often be a rather small subset of the full data, and they provide an instance of complexity reduction via relaxation along Guideline 3.13 (`guirerereco`). This roughly describes the principles behind *support vector machines* for the implementation of learning algorithms. These principles are consequences of optimization, not of statistical learning theory, and they arise in other applications as well. We explain this in some more detail in Section 11 (`SecAL`) and apply it to adaptive collocation solvers for partial differential equations in Section 15 (`SecMC`).

Furthermore, we see via this optimization argument that the exact solution of a large system (15, `eqintsys`) can be replaced by an approximative solution of a smaller subsystem. This supports Guideline 3.14 (`guipivot`) again. It is in sharpest possible contrast to the large linear systems arising in finite element theory.

Guideline 3.15 *(*`guiadaopt`*) Systems arising in kernel-based recovery problems should be solved approximatively by adaptive or optimization algorithms.*

At this point, the idea of *online learning* is helpful. It means that the training sample is viewed as a possibly infinite input sequence $(x_j, y_j) \approx (x_j, f(x_j)),\ j =$

$1, 2, \ldots$ which is used to update the current model function f_k if necessary. The connection to adaptive recovery algorithms is clear, since a new training data pair (x_{N+1}, y_{N+1}) will be discarded if the current model function f_k works well on it, *i.e.* if $f_k(x_{N+1}) - y_{N+1}$ is small. Otherwise, the model function is carefully and efficiently updated to make optimal use of the new data. Along these lines, one can devise adaptive methods for the approximate solution of partial differential equations which "learn" the solution in the sense of online learning.

Within Approximation Theory, the concept of adaptivity is closely related to the use of *dictionaries* and *frames*. In both cases, the user does not work with a finite and small set of trial functions to perform a recovery. Instead, a selection from a large reservoir of possible trial functions is made, *e.g.* by *greedy* adaptive methods or by choosing frame representations with many vanishing coefficients via certain projections. This will be a promising research area in the coming years.

The final sections of this article will review several application areas of kernel techniques. However, we shall follow the principles stated above, and we shall work out the connections between recovery, learning, and equation solving at various places. This will have to start with a look on nondeterministic recovery problems.

4 Kernels in Probabilistic Models

(`SecKiPM`) There are several different ways in which kernels arise in probability theory and statistics. We shall describe the most important ones briefly, ignoring the standard occurrence of certain kernels like the Gaussian as *densities* of probability distributions. Since Acta Numerica is aiming at readers in Numerical Analysis, we want to assume as little stochastic background as possible.

4.1 Nondeterministic Recovery Problems

If we go back to the recovery problem of Section 3 (`SecOR`) and rewrite it in a natural probabilistic setting, we get another instance of Guideline 3.1 (`guioptuse`), because kernel-based techniques again turn out to have important optimality properties. Like in Section 3 (`SecOR`) we assume that we want to find the response $f(x)$ of an unknown model function f at a new point x of a set Ω, provided that we have a sample of input-response pairs $(x_j, y_j) = (x_j, f(x_j))$ given by observation or experiment. But now we assume that the whole setting is nondeterministic, i.e. the response y_j at x_j is not a fixed function of x_j but rather a realization of a real-valued random variable $Z(x_j)$. Thus we assume that for each $x \in \Omega$ there is a real-valued random variable $Z(x)$ with expectation $E(Z(x))$ and bounded positive variance $E((Z(x) - E(Z(x))^2)$. The goal is to get information about the function $E(Z(x))$ which replaces our f in the deterministic setting. For two elements $x, y \in \Omega$ the random variables $Z(x)$ and $Z(y)$ will not be uncorrelated, because if x is close to y the random experiments described by $Z(x)$ and $Z(y)$ will often show similar behaviour. This is described by a *covariance kernel*

eqcovkerZ

$$cov(x,y) := E(Z(x) \cdot Z(y)) \text{ for all } x, y \in \Omega. \tag{20}$$

Such a kernel exists and is positive semidefinite under weak additional assumptions. If there are no exact linear dependencies in the random variables $Z(x_i)$, a kernel matrix with entries $cov(x_j, x_k)$ will be positive definite. A special case is a *Gaussian process* on Ω, where for every subset $X = \{x_1, \dots, x_N\} \subset \Omega$ the vectors $Z_X := (Z(x_1), \dots, Z(x_N))$ have a multivariate Gaussian distribution with mean $E(Z_X) \in \mathbf{R}^N$ and a covariance yielding a matrix $A \in \mathbf{R}^{N\times N}$ which has entries $cov(x_j, x_k)$ in the above sense.

Now there are several equivalent approaches to produce a good estimate for $Z(x)$ once we know data pairs (x_j, y_j) where the y_j are noiseless realizations of $Z(x_j)$. The case of additional noise will be treated later. First, *Bayesian* thinking asks for the expectation of $Z(x)$ given the information $Z(x_1) = y_1, \dots, Z(x_N) = y_N$ and write this as the expectation of a conditional probability

$$\tilde{Z}(x) := E(Z(x)|Z(x_1) = y_1, \dots, Z(x_N) = y_N).$$

Second, *Estimation Theory* looks at all linear estimators of the form

$$\tilde{Z}(x) := \sum_{j=1}^{N} u_j(x) y_j$$

using the known data to predict $Z(x)$ optimally. It minimizes the *risk* defined as

$$E((Z(x) - \sum_{j=1}^{N} u_j(x) Z(x_j))^2)$$

by choosing appropriate coefficients $u_j(x)$.

Both approaches give the same result. Furthermore, the result is computationally identical to the solution of the deterministic case using the kernel $K(x,y) = cov(x,y)$ right away, ignoring the probabilistic background completely. The system (15, `eqintsys`) has to be solved for the coefficients α_k, and the result can be written either via (14, `eqfsimp`) or Theorem 3.6 (`Thepwoptrec`). The proof of this theorem is roughly the same as the one for the Estimation Theory case in the probabilistic setting.

Guideline 4.1 (`guidetprob`) *Positive definite kernels allow a unified treatment of deterministic and probabilistic methods for recovery of functions from data.*

Guideline 4.2 (`guidetprobeq`) *Applications using kernel-based trial spaces in non-deterministic settings should keep in mind that what they do is equivalent to an estimation process for spatial random variables with a covariance described by the chosen kernel.*

This means that compactly supported or quickly decaying kernels lead to uncoupled spatial variables at larger distances. Furthermore, it explains why wide scales usually allow to get along with fewer data (see Guideline 3.12 (`guiwidescale`)). If there is a strong interdependence of local data, it suffices to use few data to explain the phenomena.

If we add a noise variable $\epsilon(x)$ at each point $x \in \Omega$ with mean zero and variance σ^2 such that the noise is independent for different points and independent of Z there, the covariance kernel with noise is

$$E((Z(x)+\epsilon(x))\cdot(Z(y)+\epsilon(y))) = cov(x,y)+\sigma^2\delta_{xy}$$

such that in presence of noise one has to add a diagonal matrix with entries σ^2 to the kernel matrix in (15, `eqintsys`). This addition of noise makes the kernel matrices positive definite even if the covariance kernel is only positive semidefinite. In a deterministic setting, this reappears as *relaxed interpolation* and will be treated in Section 7 (`SecAK`).

If there is no a-priori information on the covariance kernel and the noise variance σ, one can try to estimate these from a sufficiently large data sample. For details we refer to the literature [**?**, **?**].

If the covariance kernel is positive definite, the general theory of Section 2 (`SecKer`) applies. It turns the space spanned by functions $cov(\cdot, y)$ on Ω into a reproducing kernel Hilbert space such that the inner product of two such functions is expressible via (4, `eqKKK`) by the covariance kernel itself. This is not directly apparent from where we started. In view of learning theory, the map $x \mapsto cov(x,y)$ is a special kind of feature map which assigns to each other input x a number indicating how closely related it is to y.

4.2 Random Functions

In the above situation we had a random variable $Z(x)$ at each point $x \in \Omega$. But one can also consider random choices of functions f from a set or space $\mathcal{F}$ of real-valued functions on Ω. This requires a probability measure ρ on $\mathcal{F}$, and one can define another kind of *covariance kernel* via

`eqcovkerF`

$$\begin{array}{rcl} cov(x,y) & := & E(f(x)\cdot f(y)) \\ & = & \displaystyle\int_{\mathcal{F}} f(x)f(y)d\rho(f) \text{ for all } x,y \in \Omega \\ & = & \displaystyle\int_{\mathcal{F}} \delta_x(f)\delta_y(f)d\rho(f) \text{ for all } x,y \in \Omega. \end{array} \tag{21}$$

This is a completely different situation, both mathematically and "experimentally", because the random events and probability spaces are different.

But now the connection to Hilbert spaces and feature maps is much clearer right from the start, since the final form of the covariance kernel can be seen as a bilinear form $cov(x,y) = (\delta_x, \delta_y)$ in a suitable space. For this, we define a feature map eqphiF

$$\Phi(x) := \delta_x \ : \ f \mapsto f(x) \text{ for all } f \in \mathcal{F} \tag{22}$$

as a linear functional on $\mathcal{F}$. To a fixed input item x it assigns all possible "attributes" $f(x)$ where f varies over all random functions in $\mathcal{F}$. If we further assume that the range of the feature map is a pre-Hilbert subspace of the dual $\mathcal{F}^*$ of $\mathcal{F}$ under the inner product

$$(\lambda, \mu)_{\mathcal{F}^*} := E(\lambda(f) \cdot \mu(f)) = \int_{\mathcal{F}} \lambda(f)\mu(f)d\rho(f),$$

we are back to (1, `eqkerler`) in the form

eqcovPP

$$cov(x, y) = (\Phi(x), \Phi(y))_{\mathcal{H}} \text{ for all } x, y \in \Omega \tag{23}$$

once we take $\mathcal{H}$ as the Hilbert space completion.

If we have training data pairs $(x_i, y_i),\ i = 1, \dots, N$ as before, the y_i are simultaneous evaluations $y_i = f(x_i)$ of a random function $f \in \mathcal{F}$. A Bayesian recovery problem without noise would take the expected $f \in \mathcal{F}$ under the known information $y_i = f(x_i)$ for $i = 1, \dots, N$. Another approach is to find functions u_j on Ω such that the expectation

$$E((f(x) - \sum_{j=1}^{N} u_j(x) f(x_j))^2)$$

is minimized. Again, these two recovery problems coincide and are computationally equivalent to what we already did in Section 2 (`SecKer`) in the deterministic case, once the covariance kernel is specified.

The two different definitions for a covariance kernel cannot lead to serious confusion, because they are very closely related. If we start with random functions and (21, `eqcovkerF`), there are pointwise random variables $Z(x) := \{f(x)\}_{f \in \mathcal{F}}$ leading to the same covariance kernel via (20, `eqcovkerZ`). Conversely, starting from random variables $Z(x)$ and (20, `eqcovkerZ`) such that the covariance kernel is positive definite, a suitable function class $\mathcal{F}$ can be defined via the span of all $cov(\cdot, y)$, and point evaluations on this function class carry an inner product which allows to define a Hilbert space $\mathcal{H}$ such that (22, `eqphiF`) and (23, `eqcovPP`) hold.

4.3 Density Estimation by Kernels

This is again a different story, because the standard approach does not solve a linear system. The problem is to recover the density f of a multivariate distribution over a domain Ω from a large sample $x_1, \dots, x_N \in \Omega$ including repetitions. Where sampling points lie dense, the true density function must take large values. A primitive density estimate is possible via counting the samples in each cell of a grid, and to plot the resulting histogram. This yields a piecewise constant density estimate, but one can do better by using a nonnegative symmetric translation-invariant kernel K with total integral one and defining

$$\tilde{f}(x) := \frac{1}{N}\sum_{j=1}^{N} K\left(\frac{x-x_i}{h}\right)$$

as a smooth estimator. If the *bandwidth* h is taken too small, the result just shows sharp peaks at the x_i. If h is too large, the result is smoothed too much to be useful. We have another instance of the scaling problem here. There is a vast literature [**?**] on picking the "right" bandwidth and kernel experimentally, using as much observational or a-priori information as possible.

5 Kernel Construction

(`SecKC`) Before we delve into applications, we have to prepare by taking a closer and more application-oriented view at kernels. We want to give a short but comprehensive account of kernel construction techniques, making the reader able to assess features of given kernels or to construct new ones with prescribed properties.

If the domain Ω has no structure at all, the most important strategy to get a useful kernel is to construct a feature map $\Phi \; : \Omega \to \mathcal{H}$ with values in some Hilbert space $\mathcal{H}$ first, and then to use (1, `eqkerler`) for definition of a kernel. The resulting kernel is always positive semidefinite, but it will be hard to check for positive definiteness a priori, because this amounts to proving that for arbitrary different $x_j \in \Omega$ the feature vectors $\Phi(x_j) \in \mathcal{H}$ are linearly independent. However, linearly dependent $\Phi(x_j)$ lead to linearly dependent functions $K(\cdot, x_j)$, and these are useless in the representation (14, `eqfsimp`) and can be blended out by pivoting or a suitable optimization.

Guideline 5.1 *If pivoting, adaptivity, or optimization is used along Guidelines 3.14 (*`guipivot`*) and 3.15 (*`guiadaopt`*), one can safely work with positive* semi*definite kernels in practice.*

A very common special case of a feature map occurs if there is a finite or countable set $\{\varphi_i\}_{i\in I}$ of functions on Ω. In applications, this arises if $\varphi_i(x)$ is the value of feature number i on an element $x \in \Omega$. The feature map Φ then takes an element x into the set $\Phi(x) := \{\varphi_i(x)\}_{i\in I} \in \mathbf{R}$. For a set $\{w_i\}_{i\in I}$ of positive weights one can define a weighted ℓ_2 space by

$$\ell_{2,w}(I) := \left\{ \{c_i\}_{i\in I} \; : \; \sum_{i\in I} w_i c_i^2 < \infty \right\}$$

and then assume that these weights and the functions φ_i satisfy

$$\sum_{i\in I} w_i \varphi_i^2(x) < \infty$$

on all of Ω. This means that the scaling of the functions φ_i together with the weights w_i must be properly chosen such that the above series converges. Then we define $\mathcal{H} := \ell_{2,w}(I)$ and (1, `eqkerler`) yields the kernel

`eqsumker`

$$K(x,y) := \sum_{i \in I} w_i \varphi_i(x) \varphi_i(y) \text{ for all } x, y \in \Omega \tag{24}$$

dating back to early work of Hilbert and Schmidt. Such kernels are called *Mercer* kernels in the context of learning algorithms due to their connection to the Mercer theorem on positive integral operators. But note that the latter theory is much more restrictive, decomposing a given positive integral operator with kernel K into orthogonal eigenfunctions φ_i corresponding to eigenvalues w_i. For our purposes, such assumptions are not necessary.

Even outside of machine learning, many useful recovery algorithms use kernels of the above form. For instance, on spheres one can take spherical harmonics, and on tori one can take sin and cos functions. This is the standard way of handling kernels in these situations, and there is a vast literature [**?**] on such methods, including applications to geophysics [**?**].

The reader may figure out that (24, `eqsumker`) is a well-known ingredient of calculus. For instance, classical Fourier analysis on $[0, 2\pi)$ or the unit circle in the complex plane using standard trigonometric functions and fixed weights leads to the well-known *Dirichlet* kernel this way. If the functions φ_i are orthogonal univariate polynomials, the corresponding kernel is provided by the Christoffel-Darboux formula.

???????????? check ????

Guideline 5.2 (`guiortho`) *If expansion-type kernels* (24, `eqsumker`) *are used, kernel methods provide natural multivariate extensions not only of splines (see Guideline 3.7 (*`guispli`*)), but also of classical univariate techniques based on orthogonality.*

A highly interesting new class of kernels arises when the functions φ_i are scaled shifts of compactly supported refinable functions in the sense of wavelet theory. The resulting *multiscale kernels* [72] have a built-in multiresolution structure relating them to wavelets and frames. Implementing these new kernels into known kernel techniques yields efficient multiscale algorithms which are currently investigated.

Of course, one can generalize (24, `eqsumker`) to a convolution-type formula

`eqintker`

$$K(x,y) := \int_T \varphi(x,t) \varphi(y,t) w(t) dt \text{ for all } x, y \in \Omega \tag{25}$$

under certain integrability conditions and with a positive weight function w on an integration domain T. This always yields a positive semidefinite kernel, and positive definiteness follows if functions $\varphi(x, \cdot)$ are linearly independent on T for different $x \in \Omega$. Together with (24, eqsumker), this technique is able to generate compactly supported kernels, but there are no useful special cases known which were constructed along these lines.

Guideline 5.3 *(*guiweisum*) Kernels obtained by weighted positive summation or by convolution of products of other functions are positive semidefinite.*

However, the most important case arises when the underlying set Ω has more structure, in particular if it allows *transforms* of some sort.

Guideline 5.4 *(*guipostra*) Invariant kernels with positive transforms are positive semidefinite.*

We do not want to underpin this in full generality, *e.g.* for Riemannian manifolds [39] or for topological groups [52]. Instead, we focus on translation-invariant kernels on $\mathbf{R}^d$ and use Fourier transforms there, where the above result is well-known and easy to prove. In fact, positivity of the Fourier transform almost everywhere is sufficient for positive definiteness of a kernel. This argument proves positive definiteness of the Gaussian and the Sobolev kernel in (7, eqSobK), because their Fourier transforms are well-known (another Gaussian and the function $(1+\|.\|_2^2)^{-k}$, respectively, up to certain constants). By inverse argumentation, also the *inverse multiquadric* kernels of the form

eqIMQ

$$K(x-y) := (1+\|x-y\|_2^2)^{-k},\ x,y \in \mathbf{R}^d,\ k > d/2 \tag{26}$$

are positive definite.

But note that all of these kernels have infinite support, and the kernel matrices arising in (15, eqintsys) will not be sparse. To generate sparse kernel matrices, one needs explicitly known compactly supported kernels with positive Fourier transforms. This was quite a challenge for some years, but now there are classes of such kernels explicitly available via efficient representations [101, 99, 29]. If they are dilated to have support on the unit ball, they have the simple radial form

eqWendf

$$....???????HW?????. \tag{27}$$

.....????? etc. etc. for HW to do....

There are a few other construction techniques that allow to generate new kernels out of known ones.

Theorem 5.5 *Kernels obtained by weighted positive summation of positive (semi-) definite kernels on the same domain Ω are positive (semi-) definite.*

Guideline 5.6 *(*guikerop*) If a nontrivial linear operator L is applied to both arguments of a positive semidefinite kernel, chances are good to have another positive semidefinite kernel.*

This can be carried out in detail by using the representations (24, `eqsumker`) or (25, `eqintker`), if they are available. In general, one can work with (4, `eqKKK`) and assume that L can be applied inside the inner product.

Though of only theoretical importance, we note that the approach via feature maps and learning theory, as employed in early sections of this paper, is no special case of the general situation.

Theorem 5.7 *Every symmetric positive definite kernel can be generated via a suitable feature map.*

Proof: Given a symmetric positive definite kernel K, define $\Phi(x) := K(x,\cdot)$ and $\mathcal{H} := \mathcal{K}$ using (4, `eqKKK`) to get (1, `eqkerler`). QED.

There is another construction technique we ignore here. It is covered well in the literature and relies on completely monotone univariate functions. For applications, it is of minor importance, because it is restricted to radial kernels which are positive definite on $\mathbf{R}^d$ for all dimensions, and it cannot generate kernels with compact support.

6 Special Kernels

(`SecSK`) So far, we already have presented the Gaussian kernel (9, `eqGauss`), the inverse multiquadric (26, `eqIMQ`), and the Sobolev kernel (7, `eqSobK`). These have in common that they are *radial basis functions* which are globally positive and have positive Fourier transforms. Another important class of radial kernels is compactly supported and of local polynomial form, *i.e.* the Wendland functions (27, `eqWendf`). But this is not the end of all possiblities.

Guideline 6.1 *(*`guipdeker`*) There are other and somewhat more special kernels which are related to important partial differential equations.*

The most prominent case is the *thin-plate spline* [?]

`eqTPS`

$$K(x,y) = \|x-y\|_2^2 \log \|x-y\|_2 \text{ for all } x,y \in \mathbf{R}^d \tag{28}$$

which models a thin elastic sheet suspended at y as a function of x and solves the biharmonic equation $\Delta^2 u = 0$ everywhere except at y. More generally, there are *polyharmonic splines* defined as fundamental solutions of iterated Laplacians. They deserve a closer look, because they have special scaling properties, are of central importance for the meshless *Method of Fundamental Solutions* in Section 14 (`SecSWKT`), and lead naturally to the notion of *conditionally positive definite functions* below.

The *fundamental solution* for a differential operator L at some point $x \in \mathbf{R}^d$ is defined as a kernel $K(x,\cdot)$ which satisfies $LK(x,\cdot) = \delta_x$ in the distributional sense. For the iterated Laplacian $L_m := (-\Delta)^m$ we get radial kernels

$$\begin{array}{rl} r^{2m-d} & \text{for } d \text{ odd} \\ r^{2m-d}\log r & \text{for } d \text{ even} \end{array}$$

as functions of $r = \|x-y\|_2$ up to multiplicative constants and for $2m > d$. This contains the thin-plate splines of (28, `eqTPS`) for $m = d = 2$ and generalizes to positive real exponents as

`eqPHS`

$$\begin{array}{rl} r^\beta & \text{for } \beta \notin 2\mathbf{Z} \\ r^\beta \log r & \text{for } \beta \in 2\mathbf{Z} \end{array} \tag{29}$$

where now the space dimension does not appear any more.

Unfortunately, these functions increase with r, and so they are neither bell-shaped nor globally integrable. Their Fourier transforms cannot be calculated in the classical sense, and thus there are no standard Fourier transform techniques to prove positive definiteness. The same holds for *multiquadrics*

$$(1+r^2)^{\beta/2} \text{ for } \beta \notin 2\mathbf{Z},\ \beta > 0$$

which can be seen as a regularization of the polyharmonic spline r^β at zero, and which extends the inverse multiquadrics of (26, `eqIMQ`) to positive exponents, the most widely used case being $\beta = 1$.

Fortunately, these functions can be included into kernel theory by a simple generalization.

Definition 6.2 *(*`Defcondpos`*) A symmetric kernel $K\ :\ \Omega \times \Omega \to \mathbf{R}$ is* conditionally positive (semi-) definite *of order m on $\Omega \subseteq \mathbf{R}^d$, if for all finite subsets $X := \{x_1, \ldots, x_N\}$ of Ω the symmetric matrices $A_{K,X}$ with entries $K(x_j, x_k),\ 1 \le j,k \le N$ define a positive (semi-) definite quadratic form on the subspace*

`eqmomcond`

$$V_{m,X} := \{\alpha \in \mathbf{R}^N\ :\ \sum_{j=1}^N \alpha_j p(x_j) = 0 \textit{ for all } p \in P_m^d\} \tag{30}$$

of coefficient vectors satisfying certain discrete moment conditions with respect to the space P_m^d of d-variate polynomials of order at most m.

Note that (unconditional) positive definiteness is identical to conditional positive definiteness of order zero, and that conditional positive definiteness of order m implies conditional positive definiteness of any larger order. Table **??** (`tabkernel`) lists the appropriate orders of positive definiteness for special radial kernels.

Recovery problems using conditionally positive definite kernels of positive order m have to modify the trial space $\mathcal{K}_0$ to

`eqcpdtrial`

$$\begin{array}{rcl} \mathcal{K}_m & := & P_m^d + \mathcal{P}_m \\ \mathcal{P}_m & := & \text{span}\left\{\sum_{j=1}^N \alpha_j K(\cdot, x_j),\ \alpha \in V_{m,X},\ X = \{x_1, \ldots, x_N\} \subset \Omega\right\}. \end{array} \tag{31}$$

The norm (5, `eqKKnorm`) now works only on the space $\mathcal{P}_m$, and thus $\operatorname{clos}\mathcal{P}_m$ turns into a Hilbert space. The native space $\mathcal{K}$ for K then is $\mathcal{K} := P_m^d + \operatorname{clos}\mathcal{P}_m$, but the reproduction (8, `eqrepro`) of functions via the kernel K needs a modification which we suppress here.

If we have training data $(x_k, y_k),\ 1 \leq k \leq N$ for a model $f(x_k) = y_k$, we now plug these equations into our new trial space, using a basis $p_1, \ldots, p_Q$ of P_m^d and get a linear system

$$\begin{array}{rcrcll} \displaystyle\sum_{j=1}^N \alpha_j K(x_k, x_j) & + & \displaystyle\sum_{i=1}^Q \beta_i p(x_k) & = & y_k, & 1 \leq k \leq N \\ \displaystyle\sum_{j=1}^N \alpha_j p_\ell(x_j) & + & 0 & = & 0 & 1 \leq \ell \leq Q. \end{array}$$

This system has $N + Q$ equations and unknowns, and it is uniquely solvable if there is no nonzero polynomial vanishing on the set $X = \{x_1, \ldots, x_N\}$. Since the order m of conditional positive definiteness is usually rather small ($m = 1$ for standard multiquadrics and $K(x, y) = \|x - y\|_2$, while $m = 2$ for thin-plate splines) this modification is not serious, and it can be made obsolete if the kernel is changed slightly [?]. However, many engineering applications use multiquadrics or thin-plate splines without adding constant or linear polynomials, and without caring for the moment conditions in (30, `eqmomcond`). This often causes no visible problems, but is violating restrictions imposed by conditional positive definiteness.

Note that trial spaces for polyharmonic functions are independent of scaling, if they are properly defined via (31, `eqcpdtrial`). This eliminates many of the scaling problems arising in applications, but it comes at the price of limited smoothness of the kernels, thus reducing the attainable reproduction errors along Guideline 3.11 (`guiunc`).

The condition $2m > d$ for the polyharmonic functions forbids useful cases like $m = 1$ in dimensions $d \geq 2$, and thus it excludes the fundamental solutions $\log r$ and r^{-1} of the Laplacian in dimensions 2 and 3. These kernels are radial, but they have singularities at zero. They still are useful reproducing kernels in Sobolev spaces $W_2^1(\mathbf{R}^d)$ for $d = 2, 3$, but the reproduction property now reads

$$\lambda(f) = (\lambda^x K(\cdot - x), f)_{W_2^1(\mathbf{R}^d)}$$

for all $f \in W_2^1(\mathbf{R}^d),\ \lambda \in \left(W_2^1(\mathbf{R}^d)\right)^* = W_2^{-1}(\mathbf{R}^d)$. These kernels and their derivatives arise in integral equations as *single* or *double layer potentials*, and we shall encounter them again in Section 14 (`SecSWKT`) where they are used for the meshless *Method of Fundamental Solutions*.

7 Approximation by Kernels

(`SecAK`)

In studying the approximation and stability properties of meshless methods, the following two geometric quantities are usually employed. Suppose we are confronted with a bounded set $\Omega \subseteq \mathbf{R}^d$ and a finite subset $X = \{x_1, \dots, x_N\} \subseteq \Omega$. The approximation power is measured in terms of the *fill distance*, which is given by the radius of the largest data-site free ball in Ω, i.e.

`eqfilldistance`

$$h_X := h_{X,\Omega} := \sup_{x\in\Omega} \min_{1\le j\le N} \|x - x_j\|_2. \tag{32}$$

The second geometric quantity is the *separation radius*, which is half the distance between the two closest data sites, i.e.

`eqseparationradius`

$$q_X := \frac{1}{2} \min_{j\neq k} \|x_j - x_k\|_2. \tag{33}$$

Obviously, the separation radius plays an important role in the stability analysis of the interpolation process, since a small q_X means that two points, and hence two rows in the system (15, `eqintsys`) are nearly the same.

Finally, we will call a sequence of data sets $X = X_h$ *quasi-uniform* if there is a constant $c_q > 0$ independent of X such that

$$q_X \le h_{X,\Omega} \le c_q q_X.$$

The mesh ratio $\rho = \rho_{X,\Omega} := h_{X,\Omega}/q_X \ge 1$ provides a measure of how uniformly points in X are distributed in Ω.

7.1 Nonstationary versus Stationary Schemes

There is fundamental difference in studying the approximation behavior of meshless kernel methods and classical spline or finite element methods, which can be described as follows.

In classical FEM and spline theory the support of the nodal basis functions scales with the size of the mesh. For example, using classical hat functions to express a piecewise linear spline function over the node set $h\mathbf{Z}$ leads to a representation of the form

`linearSpline`

$$s_h(x) = \sum_{j\in\mathbf{Z}} \alpha_j B_1\left(\frac{x - jh}{h}\right) = \sum_{j\in\mathbf{Z}} \alpha_j B_1\left(\frac{x}{h} - j\right) \tag{34}$$

where B_1 is the standard hat function, which is zero outside $[0, 2]$ and is defined to be $B_1(x) = x$ for $0 \le x \le 1$ and $B_1(x) = 2 - x$ for $1 \le x \le 2$.

From (34, `linearSpline`) it follows that each of the basis functions $B_1(\frac{\cdot}{h} - j)$ has support $[jh, (j + 2)h]$, i.e. the support scales with the grid width. As a consequence, when setting up an interpolation system, each row in the interpolation matrix has the same number of nonzero entries (here actually only one); and this is independent of the current grid width. Hence, such a situation is usually referred to as a *stationary scheme*. Thus, for a stationary scheme, the basis function scales linearly with the grid width. In contrast, in a nonstationary scheme the basis function is kept fixed for all grid sizes h, i.e. the approximant would now take the form

`linearSplineNonStationary`

$$s_h(x) = \sum_{j \in \mathbf{Z}} \alpha_j B_1(x - jh), \tag{35}$$

resulting into a denser and denser interpolation matrix if h tends to zero. While in classical spline theory approximants of the form (35, `linearSplineNonStationary`) play no role at all, they are crucial in meshless methods for approximating and interpolating with positive definite kernels.

Guideline 7.1 *(*`GuidelineTradeOff`*) In meshless methods using positive definite kernels, approximation orders refer in general to a nonstationary setting. However, nonstationary schemes lead to ill-conditioned interpolation matrices. On the other hand, a fully stationary scheme provides in general no convergence but interpolation matrices with a condition number being independent of the fill distance.*

Guideline 7.1 describes another general trade-off or uncertainty principle in meshless methods, see also Guideline 3.11. As a consequence, when working in practice with scaled versions of one translation invariant kernel, the scale factor needs special care.

7.2 Nonstationary Interpolation

We now study approximation properties of interpolants of the form (14, `eqfsimp`) with a *fixed* kernel but for various data sets X. To denote the dependence on X and $f \in C(\Omega)$ we will use the notation

$$s_{f,X} = \sum_{j=1}^{N} \alpha_j K(\cdot, x_j)$$

where the coefficient vector is determined by the interpolation conditions $s_{f,X}(x_j) = f(x_j)$, $1 \leq j \leq N$.

First convergence results were restricted to target functions $f \in \mathcal{H}$ from the native function space associated to the employed kernel, see [60, 61, 62, 59, 102]. They are local pointwise estimates of the form

$$|f(x) - s_{f,X}(x)| \leq CF(h)\|f\|_{\mathcal{H}},$$

where F is a function depending on the kernel. For kernels of limited smoothness it is of the form $F(h) = h^\beta$, where β relates to the smoothness of K. For infinite smooth kernels like Gaussians or (inverse) Multiquadrics it has the form $F(h) = \exp(-c/h)$.

A detailed listing of kernels and their associated functions F can again be found in [97].

Recent research has concentrated on two topics. First, a return to the classical roots of splines has shown that, at least in the case of kernels generating Sobolev spaces as their native spaces, the approximation order is rather a property of the space of target functions than the kernel. This is well-covered by the second research topic, the *escape scenario*, where the target function is less smooth than the kernel, i.e. error estimates have to be established for target functions from outside the native Hilbert space.

Guideline 7.2 *The approximation order depends locally rather on the smoothness of the target function than on the smoothness of the kernel.*

To make this more precise, let us state two recent results from [71, 70].

As usual we let $W_p^k(\Omega)$ denote the Sobolev space of measurable functions having weak derivatives up to order k in $L_p(\Omega)$. Furthermore, we will employ *fractional* order Sobolev spaces $W_p^\tau(\Omega)$, which can, for example, be introduced using interpolation theory.

Theorem 7.3 (`thzeros`) *Let k be a positive integer, $0 < s \le 1, \tau = k + s, 1 \le p < \infty, 1 \le q \le \infty$ and let $m \in \mathbf{N}_0$ satisfy $k > m + d/p$ or, for $p = 1, k \ge m + d$. Let $X \subset \Omega$ be a discrete set with mesh norm $h_{X,\Omega}$ where Ω is a compact set with Lipschitz boundary which satisfies an interior cone condition. If $u \in W_p^\tau(\Omega)$ satisfies $u|_X = 0$, then*

$$|u|_{W_q^m(\Omega)} \le C h_{X,\Omega}^{\tau - m - d(1/p - 1/q)_+} |u|_{W_p^\tau(\Omega)},$$

where C is a constant independent of u and $h_{X,\Omega}$, and $(x)_+ = \max\{x, 0\}$.

Theorem 7.3 describes mainly the structure of Sobolev functions having lots of zeros. It is entirely independent of any reconstruction method.

For interpolation by kernels we can set $u = f - s_{f,X}$ and, if the kernel generates the Sobolev space $W_2^\tau(\Omega)$ employ Theorem 7.3 to derive error estimates, for example, of the form

$$\begin{aligned} |f - s_{f,X}|_{W_2^m(\Omega)} &\le C h_{X,\Omega}^{\tau - m} \|f\|_{W_2^\tau(\Omega)} \\ |f - s_{f,X}|_{W_\infty^m(\Omega)} &\le C h_{X,\Omega}^{\tau - m - d/2} \|f\|_{W_2^\tau(\Omega)}. \end{aligned}$$

This still covers only the situation of target functions from the native Hilbert space. The next result is concerned with the situation that the kernel generates a smooth Sobolev space $W_2^\tau(\Omega)$ while the target function comes from a rougher Sobolev space $W_2^\beta(\Omega)$. It employs the mesh ratio $\rho_{X,\Omega} = h_{X,\Omega}/q_X$.

Theorem 7.4 *If $\tau \geq \beta$, $\beta = k + s$ with $0 < s \leq 1$ and $k > d/2$, and if $f \in W_2^\beta(\Omega)$, then*

$$\|f - s_{f,X}\|_{W_2^\mu(\Omega)} \leq C h_{X,\Omega}^{\beta-\mu} \rho_{X,\Omega}^{\tau-\mu} \|f\|_{W_2^\beta(\Omega)}, \qquad 0 \leq \mu \leq \beta.$$

In particular, if X is quasi-uniform, this yields

$$\|f - s_{f,X}\|_{W_2^\mu(\Omega)} \leq C h_{X,\Omega}^{\beta-\mu} \|f\|_{W_2^\beta(\Omega)}, \qquad 0 \leq \mu \leq \beta.$$

7.3 Approximation via Relaxed Interpolation

Classical interpolation works fine in the context of quasi-uniform and noiseless data. However, if a data set is highly nonquasi-uniform, ill-conditioning becomes an issue. This is due to the fact that the condition number of the interpolation matrix $A_{K,X} = (K(x_i, x_j))$ depends mainly on the smallest eigenvalue of $A_{K,X}$. For this eigenvalue it is known [11, 12, 28, 66, 67, 68, 69, 78, 97] that it can be bounded from below by

$$\lambda_{\min}(A_{K,X}) \geq cG(q_X)$$

where unfortunately, in accordance with Guideline 3.11, it often holds that $G(q) = F(q^2)$.

Hence, the best approximation error with the most stable system is achieved by using quasi-uniform sets. However, sorting out nearly coalescing points might be problematic in particular if noisy data are considered.

One possible remedy to both problems, coalescing points and noisy data, is to relax the interpolation condition and to solve instead the following smoothing problem:

`eqsplineminprob`

$$\min \left\{ \sum_{j=1}^{N} [f(x_j) - s(x_j)]^2 + \lambda \|f\|_{\mathcal{H}}^2, : s \in \mathcal{H} \right\} \tag{36}$$

where $\lambda > 0$ is a certain smoothing parameter balancing the resulting approximant between interpolation and approximation. This problem already occurred in a probabilistic setting in Section 4 (`SecKiPM`). It is also intensively studied in the context of kernel learning, see for example [36, 37, 83]) and in the theory of regularization networks (see for example [40]).

The fundamental theorem of learning theory is the following generalization of Theorem 3.4:

Theorem 7.5 *(*`Thesplineminprob`*) Suppose K is the reproducing kernel of $\mathcal{H}$. Then, the solution to* (36, `eqsplineminprob`) *is given by a function of the form* (14, `eqfsimp`)*, where the coefficient vector $\alpha = \{\alpha_j\}$ now can be calculated by solving the linear system*

$$(A_{K,X} + \lambda I)\alpha = f|X.$$

Guideline 7.6 *(*`guidetprobrelax`*) Relaxed interpolation along the lines of* (36, `eqsplineminprob`) *is computationally equivalent to recovery from noisy observations. The relaxation parameter λ is connected to the noise variance σ by $\lambda = \sigma^2$.*

Theorem 7.5 (`Thesplineminprob`) shows that the ill-conditioning problem is simply addressed by moving the eigenvalues of the interpolation matrix away from zero by an offset given by the smoothing parameter $\lambda > 0$.

However, this immediately introduces the problem of how to choose the smoothing parameter. There have been thorough investigations mainly motivated by probabilistic approaches along the lines of Section 4 (`SecKiPM`), see for example [75, 76, 35, 91, 92, 34, 74, 93].

Instead of going into details on this, we follow a recent deterministic approach [98] which is based upon the following simple observation. The solution s_λ of (36, `eqsplineminprob`) allows the following two bounds

- $|f(x_j) - s_\lambda(x_j)| \le \sqrt{\lambda}\|f\|_{\mathcal{H}}$ for all $1 \le j \le N$,
- $\|s_\lambda\|_{\mathcal{H}} \le \|f\|_{\mathcal{H}}$.

Both can easily be verified, since $f \in \mathcal{H}$ is feasible in (36, `eqsplineminprob`). Hence, if λ is considered to be small, the error function $u = f - s_\lambda$ is approximately zero on X and its $\mathcal{H}$-norm can be bounded by the twice the $\mathcal{H}$-norm of f.

Theorem 7.7 *Assume that all assumption of Theorem 7.3 hold, except for $u|X = 0$. Then the following generalized estimate holds:*

`eqapproxint`

$$|u|_{W_q^m(\Omega)} \le C\left(h_{X,\Omega}^{\tau-m-d(1/p-1/q)_+}|u|_{W_p^\tau(\Omega)} + h_{X,\Omega}^{-m}\|u|X\|_\infty\right). \qquad (37)$$

This gives in particular for our smoothing problem the estimate

$$\|f - s_\lambda\|_{L_\infty(\Omega)} \le C\left(h_{X,\Omega}^{\tau-d/2} + \sqrt{\lambda}\right)\|f\|_{\mathcal{H}}.$$

Keeping in mind that in this particular situation $F(h) = h^{\tau-d/2}$ and $G(q) = q^{2\tau-d}$, we have

Guideline 7.8 *If the smoothing parameter $\lambda > 0$ is chosen as $\lambda = Ch^{2\tau-d}$, then the best possible approximation order is achieved with an "interpolation matrix" having a largest possible smallest eigenvalue.*

7.4 Moving Least Squares

(`SubSecMLS`) While our analysis in the previous subsections dealt with nonstationary approximation schemes based on kernel methods, we will now discuss a particular *stationary* scheme. Approximation by moving least squares has its origin in the early papers [55, 63, 64, 85, 41, 42, 43]. They have become popular again, in approximation theory [56, 94], in computer graphics [], and in meshless methods for solving partial differential equations [24].

The idea of moving least squares approximation is to solve for every point x a locally weighted least squares problem. This appears to be quite expensive at first sight, but actually it is a very efficient method. Moreover, in many applications one is interested only in a few evaluations. For such applications the moving least squares are even more attractive, because it is not necessary to set up and solve a large system.

The influence of the data points is governed by a weight function $w : \Omega \times \Omega \to \mathbf{R}$, which becomes smaller the farther its arguments are away from each other. Ideally, w vanishes for arguments $x, y \in \Omega$ with $\|x - y\|_2$ greater than a certain threshold. Such a behavior can be modeled by using a translation invariant weight function. This means that w is of the form $w(x, y) = \Phi_\delta(x - y)$ with the scaled version $\Phi_\delta = \Phi(\cdot/\delta)$ of a compactly supported function $\Phi : \mathbf{R}^d \to \mathbf{R}$.

Definition 7.9 *For $x \in \Omega$ the value $s_{f,X}(x)$ of the moving least squares approximant is given by $s_{f,X}(x) = p^*(x)$ where p^* is the solution of*

$$\min \left\{ \sum_{i=1}^{N} (f(x_i) - p(x_i))^2 w(x, x_i) \; : \; p \in \pi_m(\mathbf{R}^d) \right\}. \tag{38}$$

Here, $\pi_m(\mathbf{R}^d)$ denotes the space of all d-variate polynomials of degree at most m. But it is not at all necessary to restrict oneself to polynomials. It is, for example, even possible to incorporate singular functions into the finite dimensional function space.

The minimization problem (38, `mls`) can be seen as a discretized version of the continuous problem

$$\min \left\{ \int_{\mathbf{R}^d} |f(y) - p(y)|^2 w(x, y) dy : p \in \pi_m(\mathbf{R}^d) \right\},$$

where the integral is supposed to be restricted by the support of the weight function to a region around the point x.

The simplest case of (38, `mls`) is given by choosing only constant polynomials, i.e. $m = 0$. In this situation, the solution of (38, `mls`) can easily be computed to the explicit form

`eqshepard`

$$s_{f,X}(x) = \sum_{j=1}^{N} f(x_j) \frac{w(x, x_j)}{\sum_{k=1}^{N} w(x, x_k)}, \tag{39}$$

which is also called *Shepard approximant.* From the explicit form (39, `eqshepard`), one can already read off some specific properties, which hold more generally also for moving least squares. First of all, since the weight function $w(x, y)$ is supposed to be nonnegative, so are the "basis" functions

$$a_j(x) = \frac{w(x, x_j)}{\sum_{k=1}^{N} w(x, x_k)}, \qquad 1 \le j \le N.$$

Moreover, these functions form a *partition of unity*, i.e. they satisfy

$$\sum_{j=1}^{N} a_j(x) = 1.$$

Nonnegativity and partition of unity guarantee already linear convergence, if the weight functions are of the form $w(x,y) = \Phi((x-y)/h)$, since we have for all $p \in \pi_0(\mathbf{R}^d)$:

$$\begin{aligned} |f(x) - s_{f,X}(x)| &\le |f(x) - p(x)| + |p(x) - s_{f,X}(x)| \\ &\le |f(x) - p(x)| + \sum_{j=1}^{N} a_j(x)|p(x) - f(x_j)| \\ &\le 2\|f - p\|_{L_\infty(B(x,h))} \\ &\le C_f h. \end{aligned}$$

To derive a similar result for the general moving least squares approximation scheme, it is important to rewrite the approximant in form of a quasi-interpolant

$$s_{f,X} = \sum_{j=1}^{N} a_j^m(x) f(x_j).$$

This is, under mild assumptions on the data sites, always possible. Though the basis functions $a_j^m(x)$ are in general not nonnegative, they satisfy a constraint minimization problem, which leads to a uniform bound of the ℓ_1-norm of $\{a_j^m(x)\}_{j=1}^N$. From this, convergence orders can be derived. The following results summarizes this discussion. Its proof can be found in [94].

Theorem 7.10 *Suppose the data set $X \subseteq \Omega$ is quasi-uniform and $\pi_m(\mathbf{R}^d)$-unisolvent. If the support radius δ of the compactly supported, nonnegative weight function $w(x,y) = \Phi((x-y)/\delta)$ is chosen proportional to the fill distance $h_{X,\Omega}$ and if $f \in C^{m+1}(\mathbf{R}^d)$ is the target function, then the error can be bounded by*

$$\|f - s_{f,X}\|_{L_\infty(\Omega)} \le C_f h_{X,\Omega}^{m+1}.$$

It is remarkable, that this result is actually a local one, i.e. if there are regions, where the target function is less smooth, the associated approximation order is automatically achieved.

Moreover, the assumption on the data set to be quasi-uniform is negligible if the support radius is continuously adapted to the local fill distance.

Finally, if for a point $x \in \Omega$ the data sites in the ball of radius $\delta = Ch_{X,\Omega}$ are known in advance, the minimization problem can be solved and hence the moving least squares approximation can be computed in constant time. Locating the relevant data sites can be done by employing an "intelligent" data structure in at most $\mathcal{O}(\log N)$ time, if an additional $\mathcal{O}(N \log N)$ time is allowed to built the data structure. This, of course is only relevant if a substantial number of evaluations is necessary. For only a few evaluations all relevant data sites can be found by brute force methods in linear time.

8 Large and Ill-conditioned Kernel Systems

(SecHLS)
Section 7 (SecAK) already indicated that approximation by kernels may lead to large, non-sparse systems, which are often highly ill-conditioned. This will become even more evident in Section 10 (SecAKI). Hence, it is now time to discuss efficient methods for solving large and often dense systems.

There are mainly three different approaches here:

- multipole expansion with domain decomposition methods,
- partiton of unity methods,
- multilevel with compactly supported kernels.

Each of these methods has its strengths and drawbacks and it depends on the users to decide which one suits their application best.

8.1 Multipole Expansions

We start with the discussion of multipole expansions. They are, in the first place, only a tool to approximately evaluate sums of the form

eqsum

$$s(x) = \sum_{j=1}^{N} \alpha_j K(x, x_j). \tag{40}$$

in a fast way. As a matter of fact, they have been developed in the context of the N-body problem, which appears in a diverse number of scientific fields (see [13, 1, 50]).

It is important to realize that for large N a system of the form (15, eqintsys) cannot be solved by any direct method. Instead, iterative methods have to be employed. No matter which iterative method is used, the main operation is a matrix by vector multiplication, which is nothing but the evaluation of N sums of the form (40, eqsum).

Hence, not only for a fast evaluation of the interpolant or approximant but also for solving the linear equations (15, eqintsys) it is crucial to know how to form such matrix by vector products efficiently.

To derive a sufficiently fast evaluation of (40, eqsum), for every evaluation point x the sum is splitted in the form

eqssplit

$$s(x) = \sum_{j \in I_1} \alpha_j K(x, x_j) + \sum_{j \in I_2} \alpha_j K(x, x_j), \tag{41}$$

where I_1 contains the indices of those points x_j that are close to x, while I_2 contains the inidices of those points x_j that are far away from x. Both sums can now be replaced by approximations to them. In the first case, since $\|x - x_j\|_2$

is small for $x_j \in I_1$, the associated sum can, for example, be approximated by a Taylor polynomial. This is sometimes called a *near field expansion.* More important is a proper approximation to the second sum, which is done by a *unipole* or *farfield* expansion.

The main idea of such an expansion is based upon a kernel expansion of the form (24, `eqsumker`). Incorporating the weights w_i into the function φ_i and allowing also different functions for the two arguments, this can more generally be written as

`eqgenker`

$$K(x,t) = \sum_{i=1}^{\infty} \varphi_i(x)\psi_i(t) \tag{42}$$

and one usually refers to t in $\Phi(x,t)$ as a *source point* while x is called an *evaluation point.*

Now suppose that the source points x_j, $j \in I_2$, are located in a *panel* with center t_0, which is sufficiently far away from the evaluation point, i.e. panel and evaluation point are *well-separated.* Suppose further, (42, `eqgenker`) can be split into

`eqphisplit`

$$K(x,t) = \sum_{k=1}^{p} \phi_k(x)\psi_k(t) + R_p(x,t) \tag{43}$$

with a remainder R_p that tends to zero for $\|x - t_0\|_2 \to \infty$ or for $p \to \infty$ if $\|x - t_0\|_2$ is sufficiently large. Then, (43, `eqphisplit`) allows us to evaluate the second sum s_2 in (41, `eqssplit`) by

$$\begin{aligned} s_2(x) &:= \sum_{j\in I_2} \alpha_j K(x,x_j) \\ &= \sum_{j\in I_2} \alpha_j \sum_{k=1}^{p} \phi_k(x)\psi_k(x_j) + \sum_{j\in I_2} \alpha_j R(x,x_j) \\ &= \sum_{k=1}^{p} \phi_k(x) \sum_{j\in I_2} \alpha_j \psi_k(x_j) + \sum_{j\in I_2} \alpha_j R(x,x_j) \\ &=: \sum_{k=1}^{p} \beta_k \phi_k(x) + \sum_{j\in I_2} \alpha_j R(x,x_j). \end{aligned}$$

Hence, if we use the approximation $\widetilde{s_2}(x) = \sum_{k=1}^{p} \beta_k \phi_k(x)$ we have an error bound

$$|s_2(x) - \widetilde{s_2}(x)| \le \|\alpha\|_1 \max_{j\in I_2} |R(x,x_j)|.$$

which is small if x is far enough away from the sources x_j, $j \in I_2$. Moreover, each coefficient β_k can be computed in advance in linear time. Thus, if p is much smaller than N, we can consider it as constant and we need only constant time for each evaluation of $\widetilde{s_2}$.

So far, we have developed an efficient method for evaluating a sum of the form (40, `eqsum`) for one evaluation point or, more generally, for evaluation points from the same panel, which is well separated from the panel containing the source points. To derive a fast summation formula for arbitrary evaluation points $x \in \Omega$, the idea has to be refined. To this end the underlying region of interest Ω is subdivided into cells or panels. To each panel a far field and a near field expansion is assigned. For evaluation, all panels are visited and, depending whether the panel is well-separated from the panel which contains the evaluation point or not, the near field or far field expansion is used.

The decomposition of Ω into panels can be done either uniformly or data-dependently. A uniform decomposition makes a near field expansion indispensable since the cardinality of I_1 cannot be controled. However, its simple structure makes it easy to implement and hence it has been and still is often used. An adaptive decomposition is often based on a tree like data structure where the panels are derived by recursive subdivision of space. More details can be found in the literature. In any case, since we now have to implement a unipole expansion for every panel, the resulting technique is called *multipole expansion.*

Unfortunately, the multipole expansion has to be precomputed for each kernel separately. However, for translation invariant kernels $K(x, y) = K(x - y)$, it suffices to know the far field expansion around zero. Because this gives the far field expansion around any t_0 simply by

$$\begin{aligned} K(x-t) &= K((x-t_0)-(t-t_0)) \\ &= \sum_{k=1}^{p} \phi_k(x-t_0)\psi_k(t-t_0) + R(x-t_0, t-t_0). \end{aligned}$$

The far field expansion around zero can often be calculated using Laurent expansions of the translation invariant kernel. Details can be found in [14, 16, 17, 18, 19, 20, 22, 23, 51, 77].

8.2 Domain Decomposition

Having a fast evaluation procedure for functions of the form (40, `eqsum`), different iterative methods for solving the linear system (15, `eqintsys`) can be applied. However, the reader should be aware of the fact that the far field expansion may now lead to a nonsymmetric situation [15].

Here, we want to describe a domain decomposition method [21], which can be extended to generalized interpolation problems [96] and has already been used in the context of meshless methods for partial differential equations [**?**]. However, the name is rather distracting since the domain is not decomposed but the approximation or trial space. The method is an iterative projection method.

To decompose the trial space it suffices to decompose the set of centers X (or more generally the set of functionals). To be more precise, let us decompose

X into subsets $X_1, \dots, X_k$. These subsets need not be disjoint but their union must be X. Then the algorithm starts to interpolate on the first set X_1, forms the residual, interpolates this on X_2 and so on. After k steps one cycle of the algorithm is complete and it starts over again. A more formal description is

```
1. Set $f_0 = f$, $s_0 = 0$.
2. For $n = 0, 1, 2, \dots$
        For $r = 1, \dots, k$
            $f_{nk+r} = f_{nk+r-1} - I_{X_r} f_{nk+r-1}$
            $s_{nk+r} = s_{nk+r-1} + I_{X_r} f_{nk+r-1}$
        If $\|f_{(n+1)k}\|_{L_\infty(X)} < \epsilon$ stop.
```

This algorithm approximates the interpolant $I_X f = f^*$ from (14, `eqfsimp`) up to the specified accuracy. The convergence result is based upon the fact that the interpolant $s_0 = I_X f$ is also the *best approximant* to f from the subspace

$$V_X := \left\{ \sum_{j=1}^{N} \alpha_j K(x, x_j) : \alpha \in \mathbf{R}^N \right\}.$$

Convergence is achieved under very mild assumptions on the decomposition. The data sets X_j have to be *weakly disjoint* meaning that $X_j \neq Y_j$ and $Y_{j+1} \neq Y_j$ for each $1 \leq j \leq k-1$, where $Y_j = \cup_{i=j}^{k} X_i$, $1 \leq j \leq k$. This is, for example, satisfied, if each X_j contains at least one data site, which is not contained in any other X_i.

Theorem 8.1 *Let $f \in \mathcal{K}$ be given. Suppose $X_1, \dots, X_k$ are weakly distinct subsets of $\Omega \subseteq \mathbf{R}^d$. Set $Y_j = \cup_{i=j}^{k} X_i$, $1 \leq j \leq k$. Denote with $s^{(j)}$ the approximant after j completed cycles. Then there exists a constant $c \in (0,1)$ so that*

$$\|f^* - s^{(n)}\|_{\mathcal{K}} \leq c^n \|f\|_{\mathcal{K}}.$$

For a proof of this theorem and for a more thorough discussion on how the subsets X_j have to be chosen we refer the reader to [21, 97].

For an efficient implementation one needs not only the far field or multipole expansion of the kernel. Since the coefficients of the sum (40, `eqsum`) are now changing with every iteration, one also needs intelligent update formulas. Finally, the decomposition of X into $X_1, \dots, X_k$ has to be done in such a way that the local interpolants and the (global) residuals can be computed efficiently.

8.3 Partition of Unity

Any iterative method for solving sysyem (15, `eqintsys`) leads to a solution of the form (14, `eqfsimp`). Hence, every data site x_k has influence on every evaluation point x unless compactly supported kernels are used. Though, by employing multipole methods this influence has already been reduced. On the other hand, in the situation of moving least squares, only near-by data sites are used to form the approximant. It is now our goal to derive a similar result by a *partition of unity approach.*

This time, we really decompose the domain Ω into small subdomains Ω_k in an overlapping manner: $\Omega \subseteq \cup_{j=1}^{M} \Omega_j$. Associated to this covering $\{\Omega_j\}$ we choose a partition of unity, i.e. a family of weight functions $w_j : \Omega_j \to \mathbf{R}$, which are nonnegative, supported in Ω_j, and satisfy

$$\sum_{j=1}^{M} w_j(x) = 1, \qquad x \in \Omega.$$

There weight functions are conveniently chosen as translates of *kernels* which are smooth and compactly supported, but not necessarily positive definite.

Finally, we associate to each cell Ω_j an approximation space V_j and an approximation process, which maps a function $f : \Omega_j \to \mathbf{R}$ to an approximation $s_j : \Omega_j \to \mathbf{R}$. This approximation process can, for example, be given by local interpolants using only the data sites $X_j = X \cap \Omega_j$. However, the whole procedure works for arbitrary approximation processes. In the end, the global approximant is formed from the local approximants by weighting:

$$s(x) = \sum_{j=1}^{M} w_j(x) s_j(x), \qquad x \in \Omega.$$

From the partition of unity property, we can immediately read-off that

$$\begin{aligned} |f(x) - s(x)| &= \left| \sum_{j=1}^{M} [f(x) - s_j(x)] w_j(x) \right| \\ &\le \sum_{j=1}^{M} |f(x) - s_j(x)| w_j(x) \\ &\le \max_{1 \le j \le M} \|f - s_j\|_{L_\infty(\Omega_j)}. \end{aligned}$$

Hence, we have

Guideline 8.2 (`guipum`) *The partition of unity approximant is at least as good as its worst local approximant.*

More sophisticated error estimates can be found in [8, 97], including also bounds on the derivatives (simultanous approximation). In the latter case additional assumptions on the partitions and the weight functions have to be imposed. However, for an efficient implementation of the partition of unity method, these are, in general, automatically satisfied. To control the complexity of *evaluating* the partition of unity approximant, the cells must not overlap too much, i.e. every $x \in \Omega$ has to be contained in only a small number of cells and these cells must be easily determinable. Moreover, each local approximant has to be evaluated efficiently. Keeping Guideline 8.2 in mind, this often goes hand in hand with the fact that the regions are local meaning that their diameter is of the size of the fill distance or the separation distance. For example, if the local approximation process employs polynomials a diameter $\mathcal{O}(h_{X,\Omega})$ guarantees good approximation properties of the local approximants by a Taylor polynomial argument. If interpolation by kernels is employed, it is more important that the number of centers in each cell can be considered constant when compared to the global number of centers. In each case one has roughly to assume that the number of cells is proportional to the number of data sites. In this situation, all local interpolants can be *computed* in linear time provided that the local centers are known. Hence, everything depends upon a good data structure for both the centers and the cells, which can be provided by tree-like constructions again.

Finally, the easiest way to construct the partition of unity weight functions w_j is by employing MLS in its simplest form, namely Shepard approximants (see Section 7 (`SecAK`)).

8.4 Multilevel and Compactly Supported Kernels

We now turn to a method taylored in particular for compactly supported kernels. We know from Section 7 (`SecAK`) that interpolation in the *stationary* setting will not lead to convergence. Moreover, to guarantee solvability, the same support radius for all basis functions has to be used. This, contradicts, in a certain way:

Guideline 8.3 *(`guilocalglobal`) Resolve coarse features by using a large support radius and finer features with a smaller support radius.*

To obey Guideline 8.3 (`guilocalglobal`), the following multilevel scheme is useful. We first split our set X into a nested sequence

$$X_1 \subseteq X_2 \subseteq \ldots \subseteq X_k = X.$$

If X is quasi-uniform meaning that q_X has comparable size to $h_{X,\Omega}$, then the subsets X_j should also be quasi-uniform. Moreover, they should satisfy $q_{X_{j+1}} \approx c_a q_{X_j}$ and $h_{X_{j+1},\Omega} \approx c_a h_{X_j,\Omega}$ with a fixed constant c_a.

Now, the *multilevel method*, introduced in [44], is simply one cycle of the domain decomposition method. But this time we use compactly supported basis functions with a different support radius at each level. We could even use different basis functions at different levels. Hence, a general formulation goes as follows. For every $1 \leq j \leq k$ we choose a kernel K_j and form the interpolant

$$I_j f := I_{X_j,K_j} f = \sum_{x_j \in X_j} c_{x_j}(f) K_j(\cdot, x_j)$$

by using kernel K_j on level j. We have in mind to take $K_j(x,y)$ as $K((x-y)/\delta_j)$ with a compactly supported basis function K and scaling parameter δ_j proportional to $h_{X_j,\Omega}$. The idea behind this algorithm is that one starts with a very thin, widely spread set of points and uses a smooth basis function to recover the global behavior of the function f. In the next level a finer set of points is used and a less smooth function possibly with a smaller support is employed to resolve more details and so on.

As said before, the algorithm performs one cycle of the domain decomposition algorithm. This means

1. set $f_0 = f$ and $s_0 = 0$.
2. for $1 \leq j \leq k$:

$$\begin{aligned} s_j &= s_{j-1} + I_j f_{j-1}, \\ f_j &= f_{j-1} - I_j f_{j-1}. \end{aligned}$$

8.5 Preconditioning

?????????? Weight kernels occurring in PoU and MLS. –¿ Section 7 (SecAK).

9 Kernels on Spheres

(SecKoS)
??????????

10 Applications of Kernel Interpolation

(SecAKI) Here, we review some practical application areas for kernel techniques which neither fit into Section 11 (SecAL) on Machine Learning nor into the final sections on solving partial differential equations. These techniques perform generalized interpolation of smooth functions using unstructured data. The background was described in Section 3 (SecOR) on optimal recovery, with conditional positive definiteness added from Section 6 (SecSK). Finally, special techniques for handling large-scale problems from Section 8 (SecHLS) will occur at certain places. We group the applications by certain general features that are general enough to enable the reader to insert new applications into the right context. The citations will usually not cover the whole application area. We confine ourselves with an early article and maybe a few illustrative contemporary ones.

10.1 Exotic Data Functionals

This application area uses the fact that kernel techniques can recover functions from very general kinds of "data" which need not be structured in any way. Any linear functional λ acting on multivariate functions is allowed, provided that the kernel K is chosen smooth enough to make $\lambda^x \lambda^y K(x, y)$ meaningful.

Guideline 10.1 (`guigenfunctional`) *Kernel methods can handle generalized recovery problems when the data are given by rather exotic linear functionals.*

A typical example [89, 53, 32] concerns postprocessing the output of *finite volume methods.* These calculate a set of values f_j of an unknown function f which are not evaluations of f at certain nodes x_j, but rather integrals of f over a certain small "volume" V_j. Thus the functionals in (13, `eqlapp`) are

$$\lambda_j(f) := \int_{V_j} f(t)dt,\ 1 \leq j \leq N.$$

Usually, the domains V_j form a non-overlapping decomposition of a domain Ω. Then any recovery $\tilde{f}$ of f along the lines of Sections 3 (`SecOR`) and 6 (`SecSK`) will have the same local integrals as f, and also the global integral of f is reproduced. Thus a postprocessing of a finite-volume calculation produces a smooth function with correct local "finite volumes". These functions can then be used for further postprocessing, *e.g.* calculation of gradients, pressure, contours etc.

This technique can be used in quite a general fashion. In fact, one can always add interpolation conditions of the above type to any other recovery problem, and the result will have the required conservation property.

Guideline 10.2 (`guiconlaw`) *Within kernel-based reconstruction methods, it is possible to maintain conservation laws.*

Another similar case occurs when a certain algorithm produces an output function which has not enough smoothness to be the input of a subsequent algorithm. An intermediate kernel-based interpolation will help.

Guideline 10.3 (`guireplnonsm`) *Using kernel-based techniques, one can replace a non-smooth function by a smooth one, preserving any finite number of data which are expressible via linear functionals.*

We stated this in the context of conservation here, but it will occur again later with a different focus.

A somewhat more exotic case is the recovery of functions from *orbital derivatives* along trajectories $X(t) \in \mathbf{R}^d$ of a dynamical system

[49]

have the semantics of an

problems

?? (`MM`) on Meshless Methods, **??** (`AL`) on machine Learning,

??????????

Terrain Modeling

CAGD recovery on scattered point clouds (Beatson, HW)

Local interpolants for finite volume methods (Sonar/Iske)

Ljapounov basins (Giesl)

Transition between different FEM systems (HW)

Data Mining (Hegland)???

Pointer to MLS

??????????

11 Kernels in Machine Learning

(`SecAL`) The older literature on radial basis functions was dominated by applications in *neural networks*, in which sigmoid response functions were gradually replaced by radial basis functions over the years. We do not want to explain this machinery in detail here, because *kernels* provide a much more general and flexible technique replacing classical neural networks in learning algorithms. There are close connections of Machine Learning to Pattern Recognition and Data Mining, but we have to be brief here and prefer to focus on Learning.

11.1 Problems in Machine Learning

Before we go into details, we provide an introduction to the basic notions of Machine Learning, starting from the recovery problems in Section 3 (`SecOR`). These are subsumed under *Supervised Learning*, because the expected response y_j to an input $x_j \in \Omega$ is provided by the unknown "supervisor" function $f : \Omega \to \mathbf{R}$. If the target data y_j can take non-discrete real values, the supervised recovery problem is called *regression*, while the case of discrete values is called *classification.* In the latter case the input set Ω is divided into the equivalence classes defined by the different target values. After learning, the resulting function $\tilde{f} \approx f$ should be able to classify arbitrary inputs $x \in \Omega$ by assigning one of the finitely many possible target values. For instance, a classification between "good" and "bad" inputs x_j^+ and x_j^- can be done by finding a hyperplane in feature space which separates the features $\Phi(x_j^+)$ and $\Phi(x_j^-)$ of "good" and "bad" inputs in a best possible way. This can be done by linear algebra or linear optimization, and is an instance of Guideline 2.2 (`guifeamapstolin`).

In many applications, classification is reduced to regression by

1. embedding the discrete target values into the real numbers,
2. solving the resulting regression problem by some function $\tilde{f}$,
3. classifying new inputs x by assigning the discrete target value closest to $\tilde{f}(x)$.

Thus we shall focus on regression problems later, ignoring special techniques for classification.

Unsupervised Learning has inputs $x_j \in \Omega$ but no given target responses y_j associated to them. The goal for learning is given semantically instead. A frequent case is *clustering*, which is classification with just a few target values whose calculation is part of the problem. Another unsupervised technique is the determination of anomalies, outliers, or novelties. This can be seen as a classification where only the "normal" inputs are known beforehand, while future "abnormal" inputs have to be detected.

11.2 Linear Algebra Methods in Feature Space

Many pattern recognition or learning techniques apply a linear algebra technique in feature space, and thus they use Guideline 2.2 (`guifeamapstolin`). Since the kernel matrix contains all geometric information on the learning sample, the algorithms are based on the kernel matrix. A simple novelty detection could, for instance, just check how far a new feature vector $\Phi(x)$ is away from the mean of the "normal" feature vectors $\Phi(x_j)$ and declare it "abnormal" if it is "too far away". Of course, there are statistical background arguments to support certain decision rules.

Primitive binary classification can take the means μ^+ and μ^- of the feature vectors of the "good" samples x_j^+ and the "bad" samples x_k^-, and then classify a new input x by checking whether $\Phi(x)$ is closer to μ^+ or μ^-. Of course, there are more sophisticated methods with statistical foundations, but the upshot is that a kernel defined via a feature map is all that is needed to start a linear algebra machinery, ending up with certain statistical decision rules.

A very important background technique for many pattern recognition and learning algorithms is to attempt a complexity reduction of the input data first. If this is possible, anomalies can be detected if they do not fit properly into the reduction pattern for the "normal" data. The most widely used method for complexity reduction proceeds via *principal component analysis*, which in case of kernel-based methods boils down to a singular-value decomposition of the kernel matrix followed by projection onto the eigenspaces associated to large singular values.

11.3 Optimization Methods in Feature Space

But the most important numerical methods in Machine Learning are optimizations, not linear algebra techniques. For illustration, we take a closer look at unsupervised learning in the regression case, which in Section 3 (`SecOR`) was called a recovery problem. The *reproduction-generalization* dilemma stated in Guideline 3.3 (`guirepgendil`) is observed in Machine Learning by minimizing both a *loss function* penalizing the reproduction error and a *regularization* term penalizing instability and assuring generalization. These two penalty terms arise in various forms and under various assumptions, deterministic and non-deterministic, and they can be balanced by taking a weighted sum as an objective function for joint minimization. A typical deterministic example is (36, `eqsplineminprob`) summing a least-squares loss function and a native space norm penalty term. Another case is the sup-norm loss function

$$\epsilon := \max_j |y_j - f(x_j)|$$

arising indirectly in (19, `eqlinconstr`) and added to the native space norm to define the objective function $\frac{1}{2}\|f\|_{\mathcal{K}}^2 + C\epsilon$ to be minimized.

Both cases, as many others in Machine Learning, boil down to quadratic optimization, because (5, `eqKKnorm`) allows explicit and efficient calculation of

the native space norm on the trial space (2, `eqk0`) via the kernel matrix defined for the training data. This applies to all techniques using the quadratic penalty
`eqquadpen`

$$\alpha \in \mathbf{R}^N \mapsto \alpha^T A_{K,X} \alpha = \|f\|_{\mathcal{K}}^2 \tag{44}$$

to guarantee stability and generalization. For large training samples, the resulting quadratic programming problems have to cope with huge positive definite kernel matrices in their objective function, calling for various additional numerical techniques like principal component analysis to keep the complexity under control. Of course one can also get away with linear optimization if the quadratic term is replaced by minimization of terms like $\|A_{K,X}\alpha\|_\infty$ or $\|\sqrt{A_{K,X}}\alpha\|_\infty$ with a similar penalty effect. Again, the kernel matrix is the essential ingredient.

But this technique is not limited to learning algorithms. One can use it for regularizing many other methods, because one has a cheap grip on high derivatives.

Guideline 11.1 *(*`guilinlos`*) Quadratic penalty terms* (44, `eqquadpen`) *using the square of the native space norm of a kernel-based trial function are convenient for regularizing ill-posed problems.*

Since this only requires the trial space to consist of translates of a single positive definite kernel, and since such trial spaces have good approximation properties, kernel-based methods are good candidates for solving ill-posed and inverse problems [**?**].

11.4 Loss Functions

After looking at the penalty for instability, we have to focus on the *loss* function, while we assume an at least quadratic optimization using (44, `eqquadpen`) as part of the objective function. There are various ways to define loss, but they have seriously different consequences, not only from a statistical, but also from a numerical viewpoint. We ignore the vast literature on Statistical Learning Theory here and focus on computationally relevant questions with implications for other kernel-based techniques.

The quadratic least-squares loss in (36, `eqsplineminprob`) has the consequence to add a constant diagonal to the kernel matrix. This is the old Levenberg-Marquardt regularization of least-squares problems, but it has the disadvantage that the solution will not have a reduced complexity. The resulting coefficient vector $\alpha \in \mathbf{R}^N$ for N traing samples will not necessarily have many zeros, so that the kernel-based model (14, `eqfsimp`) has full $\mathcal{O}(N)$ complexity.

On the other hand, Guidelines 3.13 (`guirerereco`) and 3.14 (`guipivot`) tell us that a complexity reduction should be possible, using only $n << N$ terms in the solution (14, `eqfsimp`). This is achieved by using *linear* loss constraints like (19, `eqlinconstr`) instead of quadratic ones. Then Kuhn-Tucker theory restricts the optimal solution via the *active* constraints. In the literature on

Machine Learning, this is the *Support Vector Machine* philosophy, because the feature vectors $\Phi(x_j)$ for the "active" indices j with $|f(x_j) - y_j| = \epsilon$ are called "support vectors" for some reason or other.

Guideline 11.2 *(*`guilinlos`*) Complexity reduction via linear loss constraints is useful for most recovery situations, deterministic or non-deterministic.*

Since many numerical methods can be reformulated as recovery problems, this has an unexpectedly wide range of possible applications. We use it for adaptive meshless collocation methods in Section 15 (`SecMC`). There are good chances that future methods for PDE solving will take the form of adaptive optimization routines with linear loss constraints leading to complexity reduction.

11.5 Kernels in Learning Theory

Theoretical research on Learning has close connections to Approximation Theory, and it is naturally focusing on kernels [87, 37, 83, 86, 73] Most of this is based on Statistical Learning Theory. Since we want to stay on the Numerical Analysis side, we only present the most important connection to approximation by kernels.

A central question in supervised learning is to have bounds for the necessary number N of training data (x_j, y_j) to guarantee the availability of a trained model $\tilde{f}$ based on these data which has a small generalization error $\|f - \tilde{f}\|_\Omega \leq \epsilon$ in some norm $\|.\|_\Omega$ over the input domain Ω. This problem can be handled using Theorem 7.4 (`thml2`) from Section (7, `SecAK`). In particular, if the true model f lies in some Sobolev space $W_2^\beta(\Omega)$ containing the native space for our kernel, and if X is a quasi-uniform sample set of N points in Ω with fill distance $h_{X,\Omega}$, we can find an exact data reproduction $s_{f,X}$ based on X such that Theorem 7.4 (`thml2`) provides an error bound of order $h_{X,\Omega}^{\beta-\mu}\|f\|_{W_2^\beta(\Omega)}$ for the generalization error in the Sobolev norm $\|.\|_{W_2^\mu}(\Omega)$. Thus the necessary number $N \approx h_{X,\Omega}^{-1/d}$ of training samples to handle all nonzero unknown functions $f \in W_2^\beta(\Omega)$ to an error $\|f - \tilde{f}\|_{W_2^\mu(\Omega)} \leq \epsilon$ behaves like

$$N \geq C \cdot \left(\frac{\epsilon}{\|f\|_{W_2^\beta(\Omega)}} \right)^{\frac{-d}{\beta-\mu}}$$

for $0 \leq \mu < \beta$. Guideline 3.11 (`guiunc`) comes up here again, because smoothness of the kernel and the model pays off. There are similar bounds in other norms, but we do not go into details. Unfortunately, there are no deterministic results yet which support Guideline 3.13 (`guirerereco`) in a quantitative way, reducing N if the reproduction quality is relaxed.

12 Meshless Methods

(`SecMM`) Here, we start considering applications of kernels within methods solving partial differential equations. These are published in abundance, mainly in journals focusing on computational techniques in engineering and sciences, and this paper should help the user to sort them out properly. To this end, we derive some guidelines for using kernels in numerical methods, but this will need some general considerations first. To set the stage properly, we recall the fundamental dichotomies between

- strong and weak problem formulations
- test and trial functions
- stationary and nonstationary scales of trial spaces
- implicit or explicit shape functions
- symmetric and unsymmetric methods

and consider

- regularity of solutions
- consistency, *i.e.* reproduction of polynomials
- adaptivity
- necessity of global spatial discretizations
- numerical integration.

These issues are intimately related, as we shall see.

12.1 Strong and Weak Problems

Strong problems define solutions as functions satisfying a partial differential equation and certain boundary conditions *pointwise*, employing evaluations of functions and classical derivatives. *Weak problems* replace point evaluations by local integrations against *test functions* or (weak) derivatives thereof, introducing numerical integrations. Both apply "tests" to check whether a "trial" function is a solution. Their difference is not on the "trial" side, but on the "test" side. We shall come back to this later.

Strong methods can be called "integration-free", and this sometimes seems to be more important than the notion of "mesh-free". As far as point evaluations are concerned, there is no big difference between weak and strong methods, since almost all methods for solving weak problems apply numerical integration techniques using strong function values. The crucial point of weak formulations is to apply integration by parts to the integrals of derivatives against test functions, thus reducing the necessary order of differentiability and allowing Hilbert space methods like Dirichlet's Principle.

Strong formulations imply stronger regularity assumptions, *i.e.* classical differentiability with Hölder continuity of the highest derivatives occurring in the differential equations. Weak formulations can get away with lower regularity and lower-order derivatives, but the derivatives are not classical [3] ones. While this argument is independent of numerical methods, regularity is closely connected to them, since convergence orders usually increase with regularity.

Guideline 12.1 *If the PDE problem will have a rather regular solution, the user should apply techniques that make use of this regularity, and can choose between weak and strong problem formulations. If the solution will definitely have low regularity, the user should first try to convert the problem to another with more regularity,* e.g. *by giving expected singularities or discontinuities a special treatment. If the final problem still leads to a solution with low regularity, the user is forced to pose a weak problem, but must expect poor numerical performance of any numerical method.*

If there is enough regularity to have a choice between weak and strong problems, the connection of the problem formulation to numerical integration becomes important. Weak formulations introduce additional numerical integrations which are not necessary for strong formulations. These numerical integrations increase the algorithmic complexity and introduce a possibly avoidable source of numerical errors.

Guideline 12.2 *Strong problem formulations avoid certain numerical integrations, but they have to assume higher regularity than weak formulations.*

The integration error can be quite serious [33, 10] and needs a careful selection of integration techniques. In particular, if regularity is high to allow high-order methods like *hp* finite elements in a weak formulation, the integration error must be increased properly to adjust to the convergence order, so that the final error is not dominated by the one induced by numerical integration. This makes it questionable to go for a weak problem formulation in case of high regularity.

12.2 Trial Functions

If we rule out purely discrete techniques like plain finite differences, the approximate solutions of partial differential equations are usually represented as linear combinations of *trial functions*. These come in a great variety, *e.g.* as polynomials, piecewise polynomials (splines, box splines, or finite elements), shape functions, particle functions, generalized finite elements, wavelets, or kernel translates. Furthermore, they do not come single, but usually as a whole *scale* of spaces, and then the question of *stationary* or *nonstationary* scaling comes up as in Section 7 (`SecAK`). Let us have a closer look at trial spaces in general in order to see where kernels are useful.

[3] Though politically incorrect, it should be pointed out that nobody can perform a numerical calculation of a weak derivative or a weak solution. Consequently, "weak" notions are useful for theoretical analysis only, not for calculations.

Guideline 12.3 *(`giutrialprop`) Trial functions should*

- *provide a good approximation to the solution,*
- *be effectively evaluable,*
- *easy to modify, and*
- *easy to integrate numerically, in case of weak problems.*

They should only in the latter situation be dependent on the test side. We now shall look at these properties one by one, starting with approximation properties.

In many cases, *e.g.* for finite elements, scales of trial spaces attain their approximation power via a *geometric domain discretization* of the underlying domain up to some granularity h describing something like the maximum diameter of a local polyhedral support of a trial function. Certain methods using shape functions or translated kernels do not split the domain geometrically, but use a cloud of points that "fills" the domain so that h is a "fill distance" like (32, `eqfilldistance`) which measures the radius of the largest ball within the domain but without one of the points. In both cases, there is s *domain discretization* involved.

But as far as approximation power is concerned, it is by no means mandatory that a scale of trial spaces requires a geometric global domain discretization of any kind.

Guideline 12.4 *(`guismalltrial`) If the expected solution of a problem has a good approximation from a low-dimensional space of global functions, the trial space should be selected accordingly, without discretizing the domain at all. If singularities of known form and place are to be expected, they should be included into the trial space, no matter what the actual numerical method is.*

Note that a missing space discretization for the trial space is just one aspect when looking at "meshless methods". There may be integration nodes in certain cases, and there may be a space discretization for the test side which we have not yet looked at. Currently, most meshless methods are still using global space discretizations, but allow to add adaptive local refinement when necessary. However, the user should keep in mind that spectral methods [45] or general trial spaces without space discretization are to be considered as alternatives when the expected properties of the solution allows them.

Guideline 12.5 *High approximation* orders *are not related to domain discretization, but to smoothness. They are achievable if the solution of the problem is sufficiently smooth. This is independent of the trial space. But they also require a trial space that can make use of that smoothness.*

Such spaces must have higher smoothness themselves, as in the p-version of the finite element method. A trial space with good approximation properties should thus have p-adaptivity in the sense that it guarantees the highest possible approximation order attainable for the (unknown) smoothness of the solution. By

Section 7 (SecAK) we know that nonstationary scales kernel-based trial spaces have such a p-adaptivity, but theory there still requires a space discretization with a small fill distance h. It is a future challenge to provide a sound mathematical basis for h-type adaptivity like the support vector technology within machine learning. Adaptive optimization strategies for PDE solving should select spatial resolutions locally where needed, and automatically yield optimal local approximation orders depending on the smoothness of the solution.

Some applications require good approximations of higher derivatives of the solution, *e.g.* if pressure or stress is to be evaluated from displacements. This calls for smooth trial functions.

Guideline 12.6 *Because the node connectivity problems of piecewise polynomials increase dramatically with smoothness requirements and space dimension, it is much easier for meshless kernel-based methods than for finite elements to generate smooth trial spaces, in particular for higher space dimensions.*

Standard results concerning numerical methods for solving ODEs suggest that good convergence orders are obtained by high *consistency* orders, provided that *stability* is satisfied. This does not generalize easily to PDE problems, and is not directly related to the approximation power of trial spaces. However, the term *consistency* is present in quite a number of papers on meshless methods, and we shall later describe its questionable use there.

We now leave approximation quality and focus on evaluation efficiency of trial functions. Though not standard in the literature, we distinguish between *explicit* and *implicit* evaluation of trial functions. For *explicit* evaluation, there is a simple formula like $\exp(-0.3 * \|x-x_j\|_2^2)$ for each trial function, and there is no need to look up a number of other nodes or to evaluate geometric data. This is the standard technique for kernel-based trial spaces. *Implicit* evaluation means that each trial function value is the result of a subroutine call to a function that depends on multiple data in a somewhat complex and geometry-dependent way. This applies to finite elements and all "shape functions" which are the result of pointwise local optimizations like *moving least squares*. If applications need to evaluate the solution on extremely many points, implicit trial spaces may not be the best choice. It often happens that the calculation of the parameters of a solution is faster than the generation of all values needed for visualization, such that evaluation becomes more important than solving. A-posteriori display of a scattered-data interpolant to the actual solution along the lines of Section 7 (SecAK) is always possible, of course, but it is a problem of its own and induces additional errors.

Guideline 12.7 *If an approximate solution composed of trial functions must be evaluated on very many points, explicit meshless representations have an advantage.*

Another efficiency argument comes up when the dimension of the trial space is large. This should be avoided following Guideline 12.4 (guismalltrial), but it always occurs if the trial space is using a space discretization with fine granularity. Even if there is not too much connectivity between geometric information, *i.e.* if the method is meshless, one needs to have a fast method for range queries retrieving neighbors of nodes. Similar problems always come up when trial spaces need some *localization.* There are various ways to cope with it, *e.g.* wavelets, multipole and multilevel methods, but they all seem to be closely connected to the choice of a useful basis, either a-priori or adaptively. This brings us to the next issue: the adaptivity properties of trial spaces.

The really serious situations for the choice of the trial space occur when singularities will arise, but at places not known beforehand. This is the case for certain fluid dynamics, advection-diffusion, or crack propagation problems. However, it does not make sense to use a fine global space discretization when there will be just a local effect that calls for a finer local resolution. This is observed by plenty of *adaptive* methods. They sometimes just re-mesh a global space discretization locally where necessary, or they add new and more flexible elements into the fixed basic triangulation, but both of these tasks are not easy. Particle- or kernel-based methods using clouds of scattered points can adapt by adding or deleting points where necessary, but they usually do not need to update geometric contingency information that arises with meshes or triangulations. This is the punchline when meshless methods are characterized [25] as "constructing the approximation entirely in terms of nodes". The cited article considers "meshless approximations based on

- moving least squares
- kernels
- partitions of unity"

and states that these "three methods are in most cases identical except for the important fact that partitions of unity enable p-adaptivity to be achieved". Furthermore, kernels occur in all of these three, and this is another reason why kernels are a central tool in meshless methods. Some authors even talk of "truly meshless methods" when they want to stress that they do not need numerical integration, but we suggest to state precisely to which extent spatial discretizations need to be maintained, and whether the trial functions can be accessed explicitly or implicitly.

12.3 Kernel-Based Trial Spaces

(`SecKBTS`) At this point, we should show how "representability in terms of nodes" is understood in meshless methods and how it is related to kernel-based trial functions, establishing a very close connection of nearly all meshless methods to kernels. The idea of "nodes" is roughly the same as the "centers" for standard kernel approximations as in Section 7 (`SecAK`). In the simplest case, the trial space should be spanned by multivariate functions $\varphi_i(x, x_i),\ i \in I$, which are functions of $x \in \mathbf{R}^d$ depending on a single "node" or "center" or "particle position" $x_i \in \mathbf{R}^d$. This function can be seen as a "smoothed particle" as in *smoothed particle hydrodynamics* (SPH), and it is called *shape function* or *particle function* in the literature. For a meshless method, there should be no complicated geometric connection between nodes like a triangulation of the convex hull of the nodes with the nodes as vertices (this could then be called a "mesh"). It should be easy to extend the trial space by adding some new nodes and associated trial functions (this is called "h-adaptivity" in FEM terms) without updating the connectivity information. In this sense, meshless methods can be seen as an alternative to adaptive finite element methods.

For many good reasons, the functions $\varphi_i(x, x_i)$ in meshless methods should be

- translation-invariant and
- compactly supported around the node x_i.

This implies that they should necessarily have the form

`eqphiker`

$$\varphi_i(x, x_i) := K\left(x - x_i\right),\ i \in I \tag{45}$$

with a compactly supported translation-invariant kernel K of small support.

Guideline 12.8 *Translation-invariant trial functions for meshless methods are always kernel-based, if they are dependent on a single node.*

This implies that trial spaces spanned by functions of the form (14, `eqfsimp`) occur canonically in meshless methods, and the previous sections have accumulated quite some information on those spaces.

But the literature on meshless methods uses also "shape functions" defined implicitly via local processes like moving least squares. Then the resulting trial functions depend on more than one node, though this is often ignored in the notation. In fact, for each node x_i there is a trial function φ_i depending on x_i and some if its neighbors, as far as they fall into the support of the weight function associated to the node x_i. Kernels occur here only via the weight functions used, and they need not be positive definite. For scattered nodes, the resulting trial functions will not be translation-invariant.

For MLS-based shape functions, we know from Section 7 (`SecAK`) that polynomial reproduction

`eqpolrepgen`

$$\sum_{i\in I} p(x_i)\varphi_i(x) = p(x) \text{ for all } x \in \mathbf{R}^d,\ p \in P_m^d \tag{46}$$

can be achieved under mild additional assumptions, where P_m^d stands for the space of d-variate polynomials of order (= degree $+1$) at most m. Note that the "partition of unity" property of Section 8 (`SecHLS`) coincides with polynomial reproduction of order one or degree zero.

Polynomial reproduction properties are sometimes called *consistency conditions*, and very many papers seem to understand *reproducing kernels* via the above reproduction property, not via (8, `eqrepro`) in Hilbert spaces. Some also seem to assume that convergence follows as soon as there is a consistency condition of some nonnegative order in the above sense, but this argument has no solid foundation, since the usual Lax-type theory understands consistency differently and is modeled for discretizations of time-dependent problems. Mathematicians will find plenty of open questions concerning convergence and error bounds of meshless methods, while many engineers seem to believe to be on solid ground once they have what they call consistency.

Sometimes the notion of *completeness* is used in the sense of *convergent approximate polynomial reproduction*, e.g. *linear completeness* meaning convergent approximate reproduction of linear functions [25]. This is different from the usual notion of completeness in mathematics, and it must be used with extreme care, in particular when assuming that it implies convergent approximate reproduction of *piecewise* linear functions.

Guideline 12.9 *Within meshless methods, the notions of* consistency *and* completeness *should be used with caution.*

Anyway, the polynomial reproduction property (46, `eqpolrepgen`) appears in many meshless methods. In fact, recent surveys [57, 58, 48] of meshless methods focus entirely on methods with exact polynomial reproduction. However, it must be stated clearly that exact polynomial reproduction is not necessary for convergence, as is shown, for example, by the rigorous convergence analysis of the generalized finite element method [9], and the symmetric [46] and unsymmetric [80] meshless collocation methods. Polynomial reproduction appears to be popular because it is necessary in convergence arguments using Strang-Fix conditions or the Bramble-Hilbert lemma, but it is not mandatory to use these tools, as follows from Section 7 (`SecAK`). By Theorem **??** (`TheSobIntCon`), optimal approximation orders in Sobolev spaces are attained without it, and in very general situations, not only for interpolation from nonstationary scales of kernel-based trial spaces.

In view of these remarks, future work should remove exact polynomial reproduction from the assumptions of many meshless methods. Instead, care must be taken to conserve physical properties like mass and momentum in applications. This is only loosely related to polynomial reproduction.

After this detour into polynomial reproduction we still have to look at a class of methods that arrives at meshless trial spaces via a slightly different approach. *Smoothed Particle Hydrodynamics* (SPH) use spatial kernel approximations that we called *discretized kernel convolutions* in Section 7 (`SecAK`) with their main convergence result given by Theorem **??** (`Theconvconv`). This means that a suitably scaled and normalized kernel K is chosen such that

`eqcontconv`

$$f \approx K * f \tag{47}$$

holds, and a discretization of the convolution integral implies

$$f(x) \approx \sum_{i \in I} w_i K(x, x_i) f(x_i) \text{ for all } x \in \mathbf{R}^d$$

with integration weights w_i at integration nodes x_i. The linear unknowns here are $f(x_i)$, while the points x_i are interpreted as *particle positions* and can be considered as nonlinear parameters whose number and value can change. The name of the technique is derived from the fact that the right-hand side writes a function or vector field as a sum over the local kernel-controlled influences of discrete particles at the points x_i. The logic of SPH thus does not directly aim at trial spaces, but rather parameterizes fields describing flows in the form (14, eqfsimp) we had in the beginning, by using the right-hand side of the above approximation. All other operations, e.g. setting up momentum equations, are performed using the parametrized flow. Since the background problems are time-dependent, the above spatial discretizations lead to large systems of ordinary differential equations, where time discretization is another issue we do not address here.

To achieve a good approximation in the continuous convolution error (47, eqcontconv), Theorem **??** (Thecontconv) tells us that the kernel should satisfy certain moment conditions, *i.e.* it should have unit integral and vanishing higher moments. If the integration scheme is exact for low-order polynomials, this implies the partition-of-unity property for the trial functions $w_iK(\cdot, x_i)$, but there will be no reproduction of higher-order polynomials. This problem can be removed by dropping the philosophy of discretizing a convolution integral, and going radically over to functions (45, eqphiker) with exact or approximate polynomial reproduction. This is called the *reproducing kernel particle method*, when the rest of the SPH is maintained, *i.e.* when discretized systems are derived from parametrized kernel-based field representations some way or other. We refer the reader to a recent survey article [57] and a book [58] on SPH and RKPM techniques, containing long lists of references, and describing many variations induced by additional physical constraints.

12.4 Residuals, Test Functionals and Functions

(SecRTfF) After looking at the trial side, we should now focus on the test side. If we assume that the trial side has somehow produced some trial function which is a candidate for an approximate solution of the partial differential equation and the boundary conditions, we want to conclude that this trial function is close to the real solution. This is the job of the test side, and it is quite natural that here the necessity of a space discretization is much more obvious than on the trial side.

But we postpone discretization on the test side for a while. If we rewrite the differential equation and the boundary conditions as differences $L(u) - f = 0$ which should be zero for the exact solution u, an approximate solution $\tilde{u}$ should make the *residuals* $L(\tilde{u}) - f$ small everywhere. Usually, to conclude that the error $u - \tilde{u}$ is small, it suffices to make sure that the residuals are small, because the solution of any well-posed linear problem will be continuously dependent on the data, implying

$$\tilde{u} - u \text{ small, if all residuals } L(\tilde{u}) - L(u) = L(\tilde{u}) - f \text{ are small}$$

where "all" means residuals of differential equation(s) and boundary condition(s) altogether, as many as present in the problem. This means that "testing" usually should make sure that the residuals are zero or at least small *globally.* Numerical techniques aiming at globally small residuals are often called *methods of weighted residuals.*

Guideline 12.10 *Small residuals imply small errors for well-posed linear problems,* i.e. *if the solution is continuously dependent on the data. But one must make sure that the notions of "well-posedness" and "small" are consistently defined.*

In fact, if we pack differential equations and boundary conditions into one single linear operator L, continuous dependence requires fixing spaces U and F for the solution u and the data f of the problem $L(u) = f$ such that

$$\|u\|_U \leq C\|L(u)\|_F$$

holds, *i.e.* L has a continuous inverse taking the data into a solution having these data. Then one must make sure that "small" residuals for an approximate solution $\tilde{u}$ means $\|L(u) - L(\tilde{u})\|_F$ to be small, nothing else. Thus, even when discretization of residuals is not an issue, the choice of a *residual norm* is important.

This is closely connected to the distinction between strong and weak problems. For strong problems, the residual norm usually is something like a $\|.\|_\infty$ norm, while weak problems will use "weaker" norms like $\|.\|_2$. But in most cases small residuals in the $\|.\|_\infty$ norm will be small also in the $\|.\|_2$ norm, such that even if the $\|.\|_2$ norm is the correct one for continuous dependence, users are safe if they minimize $\|.\|_\infty$ instead, *i.e.* solving a strong instead of a weak problem. This requires the trial space and the data f to have enough smoothness to make $\|L(\tilde{u}) - f\|_\infty$ to be well-defined, but this usually is not a big problem in many applications.

Guideline 12.11 *If trial functions and data are smooth enough, users can often use a strong formulation even if a corresponding weak formulation has continuous dependence.*

There is a natural class of numerical methods related to weighted residuals, *i.e.* methods that globally optimize residuals in the correct residual norm. These will always lead to an optimization problem instead of a linear system. In case of L_2 residual minimization, this is the well-known *method of (continuous) least squares*, and there the optimization problem is quadratic and boils down again to a linear system of equations. With weak problems it shares the disadvantage of requiring integration, while it has the additional disadvantage of working with higher-order derivatives than weak techniques. It also requires additional regularity in excess of L_2 to conclude that numerical integration of residuals has a controllable error.

For L_∞ residual minimization of residuals of problems in strong form, one gets a semi-infinite linear optimization problem. Most users will not know that there are good numerical techniques for solving such problems. Furthermore, Kuhn-Tucker conditions will help to reduce complexity, as for learning algorithms via support vector machines, and adaptivity is easy to implement. Thus there is some hope that linear optimization codes will be very helpful in the future when it comes to solve partial differential equation problems.

For both cases, there is a trial function with small residuals, if the true solution u has a good approximation $\hat{u}$ from the trial space U_{trial}. But the latter is just a question from approximation theory which is dependent on the solution u, the trial space U_{trial}, and the norm $\|.\|_U$ in the solution space U only, not on any partial differential equation. Thus user should keep the first part of Guideline **??** (`guitrialprop`) in mind without looking at the partial differential equation. Then the numerical method for solving a PDE problem, in weak or strong form, just has to make sure not to discard the existing unknown good approximation $\hat{u}$, while it produces another approximation $\tilde{u} \in U_{trial}$ based on PDE data which is not too much worse. For residual minimization algorithms, this means that there exists an admissible trial function yielding small residuals $\|L(u)-L(\hat{u})\|_F$, such that the final optimal solution cannot have worse residuals. Error bounds and convergence results will then follow the simple estimates

$$\begin{array}{rcl}\|u-\tilde{u}\|_U & \leq & C\|L(u)-L(\tilde{u})\|_F\\ & = & C \inf\limits_{v\in U_{trial}} \|L(u)-L(v)\|_F\\ & \leq & C\|L(u)-L(\hat{u})\|_F.\end{array}$$

Guideline 12.12 *Residual minimization works, if the problem is well-posed and if the trial space contains a good approximation to the solution.*

Because exact and global residual minimization cannot be carried out numerically, we now look at *discretization* on the test side. It means that only finitely many "tests" are performed. Discretization of a strong problem means taking a finite subset of points where the differential equation or boundary conditions are satisfied. This is the standard technique of *collocation.* For weak problems, discrete testing means to take inner products of the residuals with finitely many test functions, and then the residuals are not zero or small, but orthogonal to the *test space* spanned by *test functions.* In both cases one has to make sure that small results of discrete testing lead to small results in (theoretical) infinite testing. This is the most serious part of any error or convergence analysis for numerical methods. It usually takes the form of a *stability* condition relating the test and the trial space, and making sure that a small discrete residual on the trial space implies a small full residual on the trial space. We shall see examples later, but we can already state at this point that there should be no nonzero trial function $\hat{u}$ with vanishing test residuals, if we want to have error bounds, because all functions $\tilde{u}+\alpha\cdot\hat{u}$ for arbitrary $\alpha \in \mathbf{R}$ would have the same discrete test residuals and spoil the error bound.

Guideline 12.13 *The discretized residual norm on the test side should at least work like a norm on the trial space.*

???????????????

12.5 Symmetric and Unsymmetric Methods

Consequently, the test side may need more attention than the trial side, and this leads us to the distinction between *symmetric* and *unsymmetric* methods. Symmetric methods use discretizations with

- the same degree of freedom on the trial and test side,
- closely related test functionals and trial functions,
- square and possibly positive definite matrices.

For weak problems, this means that trial and test *functions* coincide, and usually the standard Galerkin method is employed, yielding a positive definite square matrix. This applies to finite elements and several generalizations, *e.g.* the GFEM [9] described in detail in this series. The GFEM is a meshless method which enlarges the admissible trial spaces far beyond classical piecewise polynomial finite elements, but it still uses the basic symmetric Hilbert space formulation of the finite element method. In its actual form, the GFEM uses stationary scales of trial spaces spanned by a partition of unity. Since it is a symmetric Galerkin technique, the trial functions and the test functions coincide. Compactly supported kernels occur naturally in the partition of unity, but they need not be positive definite. Since the current theory uses stationary approximations (see Section 7 (`SecAK`)) in its scales of local trial spaces, the only kernels providing useful approximation orders are conditionally positive definite with infinite support, like multiquadrics or thin-plate splines. When local trial spaces are generated by moving least squares (see Section 7 (`SecAK`)), weight kernels occur again. But most applications just augment finite element spaces by useful additional trial functions, e.g. for treating singularities. However, the overall axiomatic structure of the GFEM theory [9] suggests that it should be possible to extend the theory of the GFEM to allow nonstationary scales of kernel-based trial spaces with high approximation orders.

For strong problems, the test side contains point evaluation functionals and there are no test functions. But there also is a symmetric method taking the trial functions as results when these functionals are applied to one argument of a positive definite kernel. This establishes a close relation

$$\lambda \leftrightarrow v_\lambda := \lambda^x K(x,\cdot)$$

between test functionals λ and trial functions v_λ which is only possible because kernels are involved. We call this *symmetric collocation* and deal with it in Section 15 (`SecMC`).

Both kinds of symmetric methods can be rewritten as an approximation or optimization problem in Hilbert space, and their theoretical foundation strongly relies on this fact. This comes close to Guideline 3.15 (`guiadaopt`), because the problem itself is a quadratic optimization problem solved via a linear system.

?????????? Something on the convergence proof technique for symmetric problems?

Let us now look at unsymmetric methods. In the strong case, collocation [54] using nonstationary scales of trial spaces of radial basis functions, in particular multiquadrics occurs in many applications.

????? Survey?

Theoretical support was only given recently [80], proving high convergence rates depending on the regularity assumptions. We provide details in Section 15 (`SecMC`).

Unsymmetric methods for weak problems usually take the form of Petrov-Galerkin schemes, where trial and test functions differ. Their basic theory [38] was established for trial spaces spanned by multivariate polynomial splines and for elliptic problems, making use of coercivity. More modern applications [26, 27] have the same theoretical basis, but also do not apply kernel techniques.

A more radical approach to solve weak problems by an unsymmetric Petrov-Galerkin technique is the *Meshless Local Petrov-Galerkin* (MLPG) technique developed by S.N. Atluri and his collaborators [4] with a very readable short and recent survey [5] and two books [3, 6]. It uses a variety of test and trial functions, and due to its general form it can claim to formally include many other methods. However, there is no general convergence proof or error estimate available unless the method is restricted to well-known special cases like symmetric finite element techniques.

??????????????????????????

12.6 Numerical Integration

(`SecNI`)

Let us finally focus on numerical integration questions, and let us look at weak problems first. The integrals for stiffness matrix entries within weak problems usually contain products of test and trial functions or derivatives thereof. To make integration easy and precise, test and trial functions have to be chosen carefully and should be closely related. The standard choice of piecewise polynomial trial and test functions in the finite element method achieves this, since the integrals can be done exactly in case of polyhedral domains, though one has to keep track of the polyhedra carefully. The integration of test functions against arbitrary functions is required for the inhomogeneities, but this is an issue of the test side, not of the trial side. Anyway, integrating piecewise polynomials on polyhedra needs some domain triangulation first (the primary *mesh*), and then a careful choice of interpolation nodes (or transformation to standard elements) for the integration (the *integration mesh*). Even "meshless" methods, if they require integration, may sometimes need an integration mesh and are subject to influences of integration error, if the are applied to weak problems.

Using translates of radial kernels on both the trial and test side of weak problems can be equally efficient as finite elements are, if the integration domains do not interfere with boundaries, because the integrals are univariate radial functions which are either analytically known or can be pretabulated. Certain variations of the MLPG method can take advantage of this. Integration of "test" kernels against given functions may be simplified by first representing the function in terms of translates of a "trial" kernel, followed by integrations of kernels against kernels, which again is easy if no boundaries are in the way. In presence of nontrivial boundaries, all trial and test functions cause problems, unless the real boundary is replaced piecewise by boundaries of supports of trial and test functions.

For problems in strong form, this discussion is not necessary. The trial functions can be chosen freely to satisfy the first three properties. These properties are independent of PDE solving. We shall take a closer look at them, but from a more general point of view.

12.7 Classification of Meshless Methods

(`SecCoMM`)
????????????????????

12.8 Meshless Local Petrov-Galerkin Method

(`SecMLPG`)
MLPG
??????????

12.9 Local Garbage

The actual numerical calculations are carried out with these functions, using basis coefficients or function values or (strong) derivative values. The choice of good bases is a crucial issue, as is the approximation power of the *trial space* spanned by the trial functions. We should note here that one possible definition of *meshless methods* [25] marks them as "constructing the approximation entirely in terms of nodes", which is a statement concerning the representability of the basis of the trial space, ignoring other issues like weak or strong problem formulation or necessity of numerical integration.

???????????????????

Furthermore, the literature on meshless methods uses the term "Kronecker delta property" if

$$\varphi_i(x_j, x_i) = \delta_{ij} \text{ for all } i, j \in I$$

holds, *i.e.* if the basis is Lagrangian with respect to the nodes.

In both cases, any discretization induces the problem to generalize from finitely many conditions or equations to all necessary equations.

13 Kernel Methods for Inverse Problems?

(`SecKiIP`)
????????????????????

14 Special Meshless Kernel Techniques

(SecSWKT) Following Guideline 6.1 (guipdeker) along the techniques of Section 6 (SecSK), Kernel Engineering can provide kernels which are closely connected to standard differential equations. This is used by certain numerical methods to be described in this section.

14.1 Dual Reciprocity Method

(SecDRM) This misleading name stands for a technique [**?**] arisen from boundary element methods. The basic idea is to split the problem into an inhomogeneous and a homogeneous subproblem with respect to the differential equation. A problem $L(u) = f$ with a linear differential operator L and linear boundary conditions $B(u) = g$ is treated first by constructing a *particular* solution u_P with $L(u_P) = f$ without regard of boundary values. Then the *homogeneous* problem $L(u) = 0$ is solved by some function u_H under the boundary conditions $B(u) = g - B(u_P)$ to get the final solution as $u := u_P + u_H$.

The first problem uses trial spaces of known particular solutions. These are easy to construct for kernel-based trial functions. The second problem makes use of a-priori information on homogeneous solutions either via *integral equations* or *fundamental solutions*, providing trial spaces of homogeneous solutions via a special kernel called *the* fundamental solution of the differential operator L. Due to this close connection to kernels, we have to treat this technique in some detail.

Guideline 14.1 *The Dual Reciprocity Method can be applied to well-posed linear problems with well-known fundamental and particular solutions which have good approximation properties.*

14.2 Method of Particular Solutions

(SecMPS) To find a *particular* solution u_P with $L(u_P) = f$ without regard of boundary values, one can use trial functions u_i whose images $f_i := L(u_i)$ under L are well-known and numerically available. Then the right-hand side f of the differential equation is approximated by a linear combination

$$\tilde{f} := \sum_i \alpha_i f_i$$

of the f_i to some small error $\|f - \tilde{f}\|_F$ in some suitable function space F, and the approximation

$$\tilde{u}_P := \sum_i \alpha_i u_i$$

is the canonical approximation to a particular solution u_P. Note that this part of the algorithm is an approximation problem which is completely independent of partial differential equations. After construction of $\tilde{f}$ we know that the *residual*

$L(u_P - \tilde{u}_P) = f - \tilde{f}$ is small, but we have to postpone a thorough error analysis until we have looked at the homogeneous problem and boundary conditions.

Of course, there are plenty of ways to produce good approximations $\tilde{f}$ to f, provided that the approximation properties of the functions f_i are well-known. But it may be a problem to find functions f_i which are particular solutions and have good approximation properties. Starting from well-approximating multivariate functions f_i like finite elements, it is often hard or impossible to find the functions u_i with $L(u_i) = f_i$. On the other hand, starting with nice functions u_i will only rarely lead to functions $f_i = L(u_i)$ with good approximation properties.

But things can be easy if kernels are used. The simplest way is to take a smooth symmetric translation-invariant positive definite kernel K and define

$$u_i := K(\cdot - x_i) \text{ and } f_i := LK(\cdot - x_i)$$

for trial centers x_i. If the operator L is elliptic with constant coefficients, the resulting kernel LK for the f_i will be positive definite again, as inspection of Fourier transforms shows. Now all techniques of Section 7 (SecAK) can be applied to reconstruct f approximately using the trial functions f_i.

If the operator is not elliptic, the kernel LK will not be positive definite. In such cases, the reverse strategy can be helpful, starting with $f_i := K(\cdot - x_i)$ using a positive definite kernel K and finding another kernel K_L such that $L(K_L) = K$. This new kernel need not be positive definite, but since it is not used for approximation, there is no problem here.

Guideline 14.2 (guiDRM) *A natural kernel-based strategy for the Method of Particular Solutions is to have pairs u_i, f_i with $f_i = L(u_i) = K(\cdot - x_i)$ such that one can perform approximation of f by the standard translates of the kernel K.*

The literature contains many such pairs, and we cite [**?**] for a selection.

14.3 Method of Fundamental Solutions

(SecMFS) Once the problem $L(u) = f$ with boundary data $B(u) = g$ is transformed into homogeneous form $L(u) = 0$, $B(u) = g - B(\tilde{u}_P) =: g_H$ by the method of *particular solutions*, the method of *fundamental solutions* takes over. It uses a special kernel F called *the* fundamental solution of $L(u) = 0$ such that $LF(\cdot, x) = \delta_x$ in the distributional sense. These kernels are well-known for a number of linear operators, and we presented those for the iterated Laplacian in Section 6 (SecSK), *i.e.* the *thin-plate spline* of (28, eqTPS) and the *polyharmonic splines* of (29, eqPHS). This can be generalized to linear elliptic differential operators with constant coefficients, but we do not want to go into details and refer the reader to the literature on Fourier methods in partial differential equations [**?**] and on special fundamental solutions [**?**]. However, the kernel F providing the fundamental solution will have a singularity "on the diagonal", *i.e.* for $F(x, x)$ or derivatives thereof. For second-order equations in dimension 2 or more, F itself is already singular, while for higher order one gets singularities in the derivatives of F. Singular kernels are not directly covered by the standard theory of positive definite kernels, but they work fine in a generalized sense avoiding point evaluation functionals.

Once a fundamental solution F is at hand, there are various ways to generate trial functions solving the homogeneous differential equation. Before we describe these techniques, we want to look at the error and convergence analysis. The trial functions are used for approximating the prescribed boundary values g_H on the boundary. If a numerical scheme comes up with a trial function $\tilde{u}_H$ satisfying $L(\tilde{u}_H) = 0$ and with a small residual $B(\tilde{u}_H) - g_H = B(\tilde{u}_H) - B(u) + B(\tilde{u}_P)$, we use $\tilde{u} := \tilde{u}_H + \tilde{u}_P$ for our full solution and residuals

$$\begin{array}{rcl} \|L(\tilde{u}) - f\|_F & = & \|L(\tilde{u}_H + \tilde{u}_P) - f\|_F \\ & = & \|L(\tilde{u}_P) - f\|_F \\ & = & \|\tilde{f} - f\|_F \\ \|B(\tilde{u}) - g\|_G & = & \|B(\tilde{u}_H + \tilde{u}_P) - g\|_G \\ & = & \|B(\tilde{u}_H) - g_H\|_G \end{array}$$

for a suitable norm on a space G where the boundary values live. If the problem is continuously dependent on the data in the sense that an a-priori inequality
eqcontdep

$$\|u\|_U \le C(\|L(u)\|_F + \|B(u)\|_G) \tag{48}$$

holds, and if the exact solution u exists and lies in U, then there is an error bound

$$\begin{array}{rcl} \|\tilde{u} - u\|_U & \le & C(\|L(\tilde{u} - u)\|_F + \|B(\tilde{u} - u)\|_G) \\ & = & C(\|L(\tilde{u}) - f\|_F + \|B(\tilde{u}) - g\|_G) \\ & = & C(\|\tilde{f} - f\|_F + \|B(\tilde{u}_H) - g_H\|_G) \end{array}$$

reducing the overall error to the error of the residuals. Thus the Dual Reciprocity Method has a solid mathematical foundation once continuous dependence holds and the residuals are small. This confirms Guideline 14.2 (`guiDRM`). For elliptic operators satisfying a maximum principle, these error bounds can be improved, provided that the spaces F and G are chosen appropriately.

However, it remains to prove that certain approximation schemes in the methods of particular and fundamental solutions lead to small residuals in the correct spaces needed for continuous dependence. If methods of Section 7 (`SecAK`) based on positive definite kernels are applied within the method of particular solutions, there are no serious problems, because there are good error estimates like (**??**, `eqSobBound`) in Sobolev spaces on bounded domains. We thus are left with the analysis of the approximation power of the method of fundamental solutions.

A particularly simple way to generate trial functions satisfying the homogeneous problem $L(u) = 0$ is to proceed like in Section 7 (`SecAK`) by taking linear combinations of translates $F(\cdot, x_i)$ of the fundamental solution. This is an approximation problem eqdiscrMFS

$$g_H(t) \approx \sum_j \alpha_j F(t, x_j),\ t \in \Gamma \tag{49}$$

to be posed on the boundary Γ of the domain. But in order to avoid singularities of trial functions inside the domain or on the boundary, the trial centers x_i should be placed outside the domain. But then the theory of Section 7 (`SecAK`) does not apply, because the approximation domain does not contain the centers and there is no notion like a “fill distance”. However, this method performs very well in practice [**?**] if the outside centers are placed with care. For very special domains and smooth boundary data the method can be proven to have spectral convergence [**?**], but a general theory is still missing.

A well-known and much older approach is to place infinitely many trial centers right on the boundary and to take a weighted sum over all such translates of the fundamental solution. This leads to the singular *single-layer potential* integral equation

$$g_H(t) = \int_\Gamma \alpha(x) F(t, x) dx,\ t \in \Gamma.$$

Note that this is a non-discrete form of (49, `eqdiscrMFS`). Due to the singularities of F, this equation cannot be solved strongly, but it can be solved weakly *e.g.* via finite elements on the boundary. Such techniques are called *boundary element methods* and have a rich literature [**?**].

???????? Guidelines for practical work

Variations of this approach are possible by replacing $F(\cdot, x)$ by certain linear functionals acting on $F(\cdot, x)$ with respect to the second argument x. These new kernels, like the normal derivative $\frac{\partial F(\cdot,x)}{\partial n}$ will usually preserve the property that action of L on the first argument results in zero. The standard case is the integral equation of the *double-layer potential*, but there are plenty of other possibilities that are yet unexploited, *e.g.* replacing $F(t, x)$ by local integrals around x of $F(t, s)$ with respect to s in order to remove the singularities. A special case of the is the recent *boundary knot method* [**?**, **?**].

14.4 Divergence-free Kernels

(`SecDFK`)

....

????????????

15 Meshless Collocation

(`SecMC`) Within the classification of meshless methods in Section 12 (`SecMM`), the techniques of this section solve partial differential equations in *strong* form, using *collocation* on the test side and avoiding *numerical integration* completely.

On the trial side, they use *nonstationary* scales of *explicit* kernel-based trial functions. They come in a *symmetric* and an *unsymmetric* form.

In both cases, a given partial differential equation $L(u) = f$ and various boundary conditions of the form $B(u) = g$ are discretized by point evaluations of both sides in certain *collocation* nodes. For instance, a Poisson problem on a domain Ω with Dirichlet conditions $u = g_D$ on $\Gamma_D \subseteq \Gamma := \partial\Omega$ and Neumann conditions $\frac{\partial u}{\partial n} = g_N$ on $\Gamma_N \subset \Gamma$ can be discretized by a set $\Lambda := \{\lambda_1, \ldots, \lambda_N\}$ of *test functionals* consisting of three parts

$$\begin{array}{rcll} \Lambda & = & \Lambda_1 \cup \Lambda_2 \cup \Lambda_3 & \\ \Lambda_1 & := & \{\lambda_1, \ldots, \lambda_{N_1}\} & \\ \lambda_j(u) & := & -\Delta u(x_j), & x_j \in \overline{\Omega},\ 1 \leq j \leq N_1, \\ \Lambda_2 & := & \{\lambda_{1+N_1}, \ldots, \lambda_{N_2}\} & \\ \lambda_j(u) & := & u(x_j), & x_j \in \Gamma_D,\ 1 + N_1 \leq j \leq N_2, \\ \Lambda_3 & := & \{\lambda_{1+N_2}, \ldots, \lambda_{N_3}\} & \\ \lambda_j(u) & := & \frac{\partial u}{\partial n}(x_j), & x_j \in \Gamma_N,\ 1 + N_2 \leq j \leq N_3 =: N. \end{array}$$

If the evaluation points within the three sets $\Lambda_1,\ \Lambda_2,\ \Lambda_3$ of functionals are different, all linear test functionals in $\Lambda = \Lambda_1 \cup \Lambda_2 \cup \Lambda_3$ are linearly independent.

This specifies the *test* part of the problem for both the symmetric and unsymmetric methods. In general, there may be several differential operators and several boundary conditions in any kind of mixture, provided that everything is linear in u and the test functionals are linearly independent. There is no numerical integration, no test functions, and up to now there are no kernels.

Guideline 15.1 *To give certain test functionals special importance, one should apply constant factors.*

For example, boundary test functionals in Poisson problems should get a factor of about 1000 over the differential equation test functionals. Exact rules for this are not known, but the background is provided by continuous dependence inequalities like (48, `eqcontdep`) where the parts of the right-hand side should carry different weights.

15.1 Symmetric Meshless Collocation

(`SecSMC`) The difference between symmetric and unsymmetric meshless collocation shows up when looking at the trial side. For *unsymmetric* collocation, a standard nonstationary scale of kernel-based trial spaces is used, where the translates $K(\cdot, y_k)$ are taken with *trial* nodes y_k that are independent of the *test functionals* in Λ. This method goes back to Kansa [**?**] and will be analyzed later.

In the symmetric case, there must be a strong connection between trial functions and test functionals. This is done by taking the trial functions $\lambda_j^x K(\cdot, x),\ 1 \leq j \leq N$ for a sufficiently smooth kernel K guaranteeing that all test functionals lie in the dual of its native space. This is a special case of general Hermite-Birkhoff interpolation as described in Section 3 (`SecOR`). Under mild additional assumptions, this leads to a symmetric nonsingular linear system (18, `eqHBlinsys`) and error bounds along the lines of Theorem **??** (TheHBbounds). A detailed theoretical analysis of symmetric collocation can be found in the literature [100, 47, 46], while reports on applications are somewhat scattered [**?**] and limited to small problems with regular solutions. For such cases, the method gives quick and useful results, provided that the general guidelines on scaling in Section 3 (SecOR) are observed. Future work should apply special techniques of Section 8 (SecHLS) for handling large-scale and ill-conditioned systems.

15.2 Unsymmetric Meshless Collocation

(SecUMC) is much more popular than the symmetric case, because it is easier to handle. The matrix entries $\lambda_i^x\lambda_j^y K(x,y)$ of the symmetric case apply all derivatives twice, while the unsymmetric case with trial functions $K(\cdot,y_k)$ involves only $\lambda_i^x K(x,y_k)$ which is simpler to program. There are very many papers on practical applications of this technique [**?**], but a thorough theoretical analysis was missing for about 20 years [80]. We summarize the relevant issues in

Guideline 15.2 *(guiunsymmcoll) The mathematical foundation of unsymmetric collocation requires four ingredients:*

1. *a linear and well-posed PDE problem*
2. *a nonstationary scale of meshless trial spaces with good approximation properties and spanned by sufficiently smooth kernel translates $K(\cdot,y_k)$*
3. *a scale of test discretizations via sets Λ of collocation functionals which is fine enough to guarantee at least a full rank of the unsymmetric linear systems with entries $\lambda_i^x K(x,y_k)$*
4. *an approximate solution of this linear system with small discrete residuals.*

Items 1-3 above are sufficient to guarantee approximate solvability in the final step. It can be implemented by various techniques including linear or least-squares optimization or greedy adaptive methods [**?**] described below. Of course, guidelines of Section 7 (SecAK) concerning scaling must be observed at all times. If the sup norm of residuals is minimized, the method reduces to linear optimization, and it can be implemented via the revised simplex method. By Kuhn-Tucker theory, the final result will then be based only on a small finite set of test functionals. This is a connection to *support vector machines.*

15.3 Adaptive Collocation Solvers

(SecACSC) In finite elements, there is a vast recent literature on adaptivity controlled by efficient error estimation techniques. Meshless kernel-based collocation methods can implement this in a very simple way by inspecting residuals of the differential equation and the boundary conditions. Since evaluations of trial functions are explicitly possible and very cheap, one can always evaluate the residuals on a large set of background test points, using only a few of these to define the test functionals entering the calculations. The general recipe is

1. Start with $\tilde{u}$ being the zero trial function and set $N := 0$.
2. Iteration:
   Assume that there is a trial function $\tilde{u}$ which is a linear combination of N trial functions $u_1, \ldots, u_N$ such that the $N \times N$ system withe entries $\lambda_j(u_k)$ for N test functionals $\lambda_1, \ldots, \lambda_N$ is non-singular and has $\tilde{u}$ as an approximate solution.
   (a) Find a point in the domain or on the boundary where there is a large or maximal residual. Stop if none can be found.
   (b) Use this point to define a new test functional λ_{N+1} for further calculations.
   (c) (`itemtrialselect`) Add a new trial function u_{N+1} such that the enlarged system still is nonsingular.
   (d) Solve the new system approximatively for a new trial function $\tilde{u}$.

If candidates for test functionals and trial functions are chosen from a large reservoir satisfying the background theory for unsymmetric calculations as described in Guideline 15.2 (`guiunsymmcoll`), this is an adaptive bootstrapping technique that automatically selects useful subsets of trial functions and test functionals without ever forming a huge matrix defined by all possible trial functions and test functionals. Connections to the notions of *dictionaries* [?] in Approximation Theory and to *greedy algorithms* [?] are apparent.

This technique works fine for small problems [?] but needs further theoretical and numerical research if N gets large and the systems get ill-conditioned. In particular, step 2c (`itemtrialselect`) of the algorithm can be implemented in various ways, and it is not clear how to assess the performance for small N. The symmetric case can also be handled adaptively by omitting step 2c (`itemtrialselect`), taking the new test point and the corresponding functional to define a new trial function. There is a theoretical background for this symmetric *greedy* strategy [?].

?????????????????

16 Special Kernel Techniques for Time-dependent Problems?

(`SecSKTfTdP`)

?????????????????

Acknowledgement

sfg

References

[1] A. Appel. An efficient program for many-body simulation. *SIAM J. Sci. Statist. Comput.*, 8:85, 1985.

[2] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

[3] S. N. Atluri and S. Shen. The Meshless Local Petrov-Galerkin (MLPG) method. *Tech Science Press, Encino, CA, 2002*, 2002.

[4] S. N. Atluri and T. L. Zhu. A new meshless local Petrov-Galerkin (MLPG) approach to nonlinear problems in computer modeling and simulation. *Computer Modeling and Simulation in Engineering*, 3:187–196, 1998.

[5] Satya N. Atluri and Shengping Shen. The basis of meshless domain discretization: the meshless local Petrov-Galerkin (MLPG) method. *Adv. Comput. Math.*, 23(1-2):73–93, 2005.

[6] S.N. Atluri. *The meshless method (MLPG) for domain and BIE discretizations*. Tech Science Press, 2005.

[7] M. Atteia. *Hilbertian kernels and spline functions*. North-Holland, Amsterdam, 1992. Studies in Computational Mathematics (4).

[8] I. Babuska and J. M. Melenk. The partition of unity method. *Int. J. Numer. Methods Eng.*, 40:727–758, 1997.

[9] I. Babuška, U. Banerjee, and J. Osborn. Survey of meshless and generalized finite element methods a unified approach. In *Acta Numerica*, pages 1–215. 2003.

[10] I. Babuška, U. Banerjee, and J.E. Osborn. Meshless and generalized finite element methods: A survey of some major results. In *Meshfree Methods for Partial Differential Equations, Lecture Notes in Computational Science and Engineering 26*, pages 1–20. Springer, 2002.

[11] K. Ball. Eigenvalues of Euclidean distance matrices. *J. Approx. Theory*, 68:74–82, 1992.

[12] K. Ball, N. Sivakumar, and J. D. Ward. On the sensitivity of radial basis interpolation to minimal data separation distance. *Constr. Approx.*, 8:401–426, 1992.

[13] J. E. Barnes and P. Hut. A hierarchical $\mathcal{O}(N \log N)$ force-calculation algorithm. *Nature*, 324:446–449, 1986.

[14] R. K. Beatson and E. Chacko. Fast evaluation of radial basis functions: A multivariate momentary evaluation scheme. In A. Cohen, C. Rabut, and L. L. Schumaker, editors, *Curve and Surface Fitting: Saint-Malo 1999*, pages 37–46, Nashville, 2000. Vanderbilt University Press.

[15] R. K. Beatson, J. B. Cherrie, and C. T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration. *Adv. Comput. Math.*, 11:253–270, 1999.

[16] R. K. Beatson, J. B. Cherrie, and D. L. Ragozin. Polyharmonic splines in $\mathbf{R}^d$: Tools for fast evaluation. In A. Cohen, C. Rabut, and L. L. Schumaker, editors, *Curve and Surface Fitting: Saint-Malo 1999*, pages 47–56, Nashville, 2000. Vanderbilt University Press.

[17] R. K. Beatson, J. B. Cherrie, and D. L. Ragozin. Fast evaluation of radial basis functions: Methods for four-dimensional polyharmonic splines. *SIAM J. Math. Anal.*, 32:1272–1310, 2001.

[18] R. K. Beatson, G. Goodsell, and M. J. D. Powell. On multigrid techniques for thin plate spline interpolation in two dimensions. *Lect. Appl. Math.*, 32:77–97, 1996.

[19] R. K. Beatson and L. Greengard. A short course on fast multipole methods. In M. Ainsworth, J. Levesley, W. Light, and M. Marletta, editors, *Wavelets, multilevel methods and elliptic PDEs. 7th EPSRC numerical analysis summer school, University of Leicester, Leicester, GB, July 8–19, 1996*, pages 1–37. Clarendon Press, Oxford, 1997.

[20] R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: Methods for two-dimensional polyharmonic splines. *IMA J. Numer. Anal.*, 17:343–372, 1997.

[21] R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput.*, 22:1717–1740, 2000.

[22] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: I. *Comput. Math. Appl.*, 24:7–19, 1992.

[23] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: Moment-based methods. *SIAM J. Sci. Comput.*, 19:1428–1449, 1998.

[24] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Comput. Methods in Appl. Mechanics and Engineering*, 139:3–47, 1996.

[25] T. Belytschko, Y. Krongauz, D.J. Organ, M. Fleming, and P. Krysl. Meshless methods: an overview and recent developments. *Computer Methods in Applied Mechanics and Engineering, special issue*, 139:3–47, 1996.

[26] B. Bialecki and G. Fairweather. Orthogonal spline collocation methods for partial differential equations. *J. Comput. Appl. Math.*, 128(1-2):55–82, 2001. Numerical analysis 2000, Vol. VII, Partial differential equations.

[27] B. Bialecki, M. Ganesh, and K. Mustapha. A Petrov-Galerkin method with quadrature for elliptic boundary value problems. *IMA J. Numer. Anal.*, 24(1):157–177, 2004.

[28] P. Binev and K. Jetter. Estimating the condition number for multivariate interpolation problems. In D. Braess et al., editors, *Numerical methods in approximation theory. Vol. 9: Proceedings of the conference held in Oberwolfach, Germany, November 24-30, 1991*, volume 105 of *Int. Ser. Numer. Math.*, pages 41–52, Basel, 1992. Birkhäuser.

[29] M. D. Buhmann. Radial functions on compact support. *Proc. Edinb. Math. Soc. II*, 41:33–46, 1998.

[30] M.D. Buhmann. Radial basis functions. *Acta Numerica*, 10:1–38, 2000.

[31] M.D. Buhmann. *Radial Basis Functions*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004.

[32] Tom Cecil, Jianliang Qian, and Stanley Osher. Numerical methods for high dimensional hamilton-jacobi equations using radial basis functions. *J. Comput. Phys.*, 196(1):327–347, 2004.

[33] P. G. Ciarlet. Basic error estimates for elliptic problems. In *Handbook of numerical analysis, Vol. II*, Handb. Numer. Anal., II, pages 17–351. North-Holland, Amsterdam, 1991.

[34] D. D. Cox. Multivariate smoothing spline functions. *SIAM J. Numer. Anal.*, 21:789–813, 1984.

[35] P. Craven and G. Wahba. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31:377–403, 1979.

[36] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, 2000.

[37] F. Cucker and S Smale. On the mathematical foundation of learning. *Bull. Amer. Math. Soc.*, 39:1–49, 2001.

[38] Jim Douglas, Jr., Todd Dupont, Henry H. Rachford, Jr., and Mary F. Wheeler. Local H^{-1} Galerkin procedures for elliptic equations. *RAIRO Anal. Numér.*, 11(1):3–12, 111, 1977.

[39] N. Dyn, F.J. Narcowich, and J.D. Ward. Variational principles and sobolev–type estimates for generalized interpolation on a riemannian manifold. *Constructive Approximation*, 15(2):174–208, 1999.

[40] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Adv. Comput. Math.*, 13:1–50, 2000.

[41] R. Farwig. Multivariate interpolation of arbitrarily spaced data by moving least squares methods. *J. Comput. Appl. Math.*, 16:79–83, 1986.

[42] R. Farwig. Multivariate interpolation of scattered data by moving least squares methods. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*, pages 193–211, Oxford, 1987. Clarendon Press.

[43] R. Farwig. Rate of convergence of moving least squares interpolation methods: The univariate case. In P. Nevai and A. Pinkus, editors, *Progress in Approximation Theory*, pages 313–327. Academic Press, Boston, 1991.

[44] M. S. Floater and A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *J. Comput. Appl. Math.*, 73:65–78, 1996.

[45] Bengt Fornberg and David M. Sloan. A review of pseudospectral methods for solving partial differential equations. In *Acta numerica, 1994*, Acta Numer., pages 203–267. Cambridge Univ. Press, Cambridge, 1994.

[46] C. Franke and R. Schaback. Solving partial differential equations by collocation using radial basis functions. *Appl. Math. Comp.*, 93:73–82, 1998.

[47] Carsten Franke and R. Schaback. Convergence order estimates of meshless collocation methods using radial basis functions. *Adv. Comput. Math.*, 8(4):381–399, 1998.

[48] Th.-P. Fries and H.-G. Matthies. Classification and overview of meshfree methods. Informatikbericht Nr. 2003-3, revised, Scientific Computing Univ. Braunschweig, 2004.

[49] P. Giesl. Construction of global lyapunov functions using radial basis functions, 2005. Habilitationsschrift.

[50] L Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325–348, 1987.

[51] L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. Sci. Statist. Comput.*, 12:79–94, 1991.

[52] T. Gutzmer. Interpolation by positive definite functions on locally compact groups with application to SO(3). *Result. Math.*, 29:69–77, 1996.

[53] A. Iske and T. Sonar. On the structure of function spaces in optimal recovery of point functionals for ENO-schemes by radial basis functions. *Numer. Math.*, 74:177–201, 1996.

[54] E. J. Kansa. Application of Hardy's multiquadric interpolation to hydrodynamics. In *Proc. 1986 Simul. Conf., Vol. 4*, pages 111–117, 1986.

[55] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Math. Comput.*, 37:141–158, 1981.

[56] D. Levin. Stable integration rules with scattered integration points. *J. Comput. Appl. Math.*, 112:181–187, 1999.

[57] Shaofan Li and Wing Kam Liu. Meshfree and particle methods and applications. *Appl. Mech. Rev.*, 55:1–34, 2002.

[58] Shaofan Li and Wing Kam Liu. *Meshfree particle methods*. Springer-Verlag, Berlin, 2004.

[59] W. A. Light and H. Wayne. On power functions and error estimates for radial basis function interpolation. *J. Approx. Theory*, 92:245–266, 1998.

[60] W. R. Madych and S. A. Nelson. Multivariate interpolation and conditionally positive definite functions. *Approximation Theory Appl.*, 4:77–89, 1988.

[61] W. R. Madych and S. A. Nelson. Multivariate interpolation and conditionally positive definite functions II. *Math. Comput.*, 54:211–230, 1990.

[62] W. R. Madych and S. A. Nelson. Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation. *J. Approx. Theory*, 70:94–114, 1992.

[63] D. H. McLain. Drawing contours from arbitrary data points. *Comput. J.*, 17:318–324, 1974.

[64] D. H. McLain. Two dimensional interpolation from random data. *Comput. J.*, pages 178–181, 1976.

[65] H. Meschkowski. *Hilbertsche Räume mit Kernfunktion*. Springer, Berlin, 1962.

[66] F. J. Narcowich and J. D. Ward. Norms of inverses and condition numbers for matrices associated with scattered data. *J. Approx. Theory*, 64:69–94, 1991.

[67] F. J. Narcowich and J. D. Ward. Norms of inverses for matrices associated with scattered data. In P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker, editors, *Curves and Surfaces*, pages 341–348, Boston, 1991. Academic Press.

[68] F. J. Narcowich and J. D. Ward. Norm estimates for the inverse of a general class of scattered-data radial-function interpolation matrices. *J. Approx. Theory*, 69:84–109, 1992.

[69] F. J. Narcowich and J. D. Ward. On condition numbers associated with radial-function interpolation. *J. Math. Anal. Appl.*, 186:457–485, 1994.

[70] F. J. Narcowich, J. D. Ward, and H. Wendland. Sobolev error estimates and a Bernstein inequality for scattered data interpolation via radial basis functions. College Station, 2004.

[71] F. J. Narcowich, J. D. Ward, and H. Wendland. Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting. *Math. Comput.*, 74:643–763, 2005.

[72] R. Opfer. Multiscale kernels. *Advances in Computational Mathematics*, 2005.

[73] T. Poggio and S. Smale. The mathematics of learning: dealing with data. *American Mathematical Society*, 50(5):537–544, 2003.

[74] D. L. Ragozin. Error bounds for derivative estimates based on spline smoothing of exact or noisy data. *J. Approx. Theory*, 37:335–355, 1983.

[75] C. H. Reinsch. Smoothing by spline functions. *Numer. Math.*, 10:177–183, 1967.

[76] C. H. Reinsch. Smoothing by spline functions II. *Numer. Math.*, 16:451–454, 1971.

[77] G. Roussos. *Computation with radial basis functions.* PhD thesis, Imperial College of Science Technology and Medicine, University of London, 1999.

[78] R. Schaback. Error estimates and condition number for radial basis function interpolation. *Adv. Comput. Math.*, 3:251–264, 1995.

[79] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3:251–264, 1995.

[80] R. Schaback. Convergence of unsymmetric kernel–based meshless collocation methods. Preprint Göttingen, 2005.

[81] B. Schoelkopf and A. Smola. *Learning with Kernels.* MIT Press, Cambridge, 2002.

[82] I. J. Schoenberg. On certain metric spaces arising from Euclidean spaces by a change of metric and their imbedding in Hilbert space. *Annals of Math.*, 38:787–793, 1937.

[83] B. Schölkopf and A. J. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, Massachusetts, 2002.

[84] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[85] D. Shepard. A two dimensional interpolation function for irregularly spaced data. In *Proceedings of the ACM national conference*, pages 517–524, 1968.

[86] Steve Smale and Ding-Xuan Zhou. Estimating the approximation error in learning theory. *Anal. Appl. (Singap.)*, 1(1):17–41, 2003.

[87] A. J. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22(1-2):211–231, 1998.

[88] J. Stewart. Positive definite functions and generalizations, an historical survey. *Rocky Mountain J. Math.*, 6:409–434, 1976.

[89] 16 T. Sonar, *Optimal recovery using thin plate splines in finite volume methods for the numerical solution of hyperbolic conservation laws.* 549–581. 1996.

[90] J. F. Traub and A. G. Werschulz. *Complexity and Information.* Oxford University Press, Oxford, UK, 1998.

[91] G. Wahba. Smoothing noisy data by spline functions. *Numer. Math.*, 24:383–393, 1975.

[92] G. Wahba. *Spline Models for Observational Data.* CBMS-NSF, Regional Conference Series in Applied Mathematics. Siam, Philadelphia, 1990.

[93] T. Wei, Y.C. Hon, and Y. B. Wang. Reconstruction of numerical derivatives from scattered noisy data. *Inverse Problems*, 21:657 – 672, 2005.

[94] H. Wendland. Local polynomial reproduction and moving least squares approximation. *IMA J. Numer. Anal.*, 21:285–300, 2001.

[95] H. Wendland. *Scattered Data Approximation.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, 2004.

[96] H. Wendland. Solving large generalized interpolation problems efficiently. In M. Neamtu and E. B. Saff, editors, *Advances in Constructive Approximation*, pages 509–518, Brentwood, TN, 2004. Nashboro Press.

[97] H. Wendland. *Scattered Data Approximation.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, UK, 2005.

[98] H. Wendland and C. Rieger. Approximate interpolation with applications to selecting smoothing parameters. to appear in Numerische Mathematik, 2004.

[99] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(4):389–396, 1995.

[100] Z. Wu. Hermite–Birkhoff interpolation of scattered data by radial basis functions. *Approximation Theory and its Applications*, 8/2:1–10, 1992.

[101] Z. Wu. Multivariate compactly supported positive definite radial functions. *Advances in Computational Mathematics*, 4:283–292, 1995.

[102] Z. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. *IMA J. Numer. Anal.*, 13:13–27, 1993.