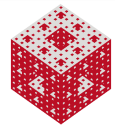
MATH 590: Meshfree Methods

Parameter Estimation and Optimization

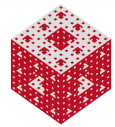
Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Fall 2014



Outline



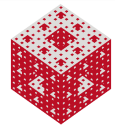
Up until now we have not discussed how to systematically choose a "good" value of the shape parameter ε present in many kernels.

Some typical observations we have made earlier are that, for decreasing values of ε,

- the kernels become wider, or "flatter",
- the condition number of the matrix K increases,
- the interpolant s resembles the test function f more accurately.

This indicates both the benefits and challenges associated with the choice of a "good" shape parameter.

We will mostly limit our discussion to finding a single "optimal" shape parameter (i.e., no anisotropic kernels, or kernels with multiple parameters).



Trial and Error

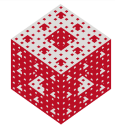
The simplest strategy is to perform a series of interpolation experiments with varying shape parameter, and then to pick the "best" one.

This strategy can be used with confidence only if we know the function f that generated the data, and therefore can calculate some sort of error for the interpolant.

Of course, if we already know f, then the exercise of finding an interpolant s may be mostly pointless.

However, this is the strategy we used for the "academic" examples earlier in this class.

This is still a very popular – and subjective – approach to picking a "good" value of ε.



Other ad hoc methods

- In one of the earliest RBF papers on (inverse) multiquadric RBF interpolation in $\mathbb{R}^2$ [Har71] suggests using

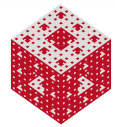
$$\varepsilon = 1/(0.815\delta),$$

  where $\delta = \frac{1}{N}\sum_{i=1}^{N} \delta_i$, and δ_i is the distance from the data point $\boldsymbol{x}_i$ to its nearest neighbor.
- [Fra82] recommends using

$$\varepsilon = \frac{0.8\sqrt{N}}{D},$$

  for 2D problems, where D is the diameter of the smallest circle containing all data points.
- [Fas02] has been quoted by some, implying that $\varepsilon = 2\sqrt{N}$ is a good choice for 2D problems.

While these strategies indeed may work reasonably well for very specific examples, they certainly cannot be taken as general guidelines.



In order to come up with a systematic approach will have to make sure that

- we specify what our criteria for "good" are,
- we estimate a value for the shape parameter from the data alone, i.e., without knowing the solution to the problem (as was always the case in our earlier ε plots),
- the estimation can be performed efficiently and accurately.

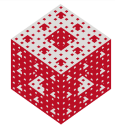
If we do not have any knowledge of f, then it becomes rather difficult to decide what "best" means.

One (non-optimal) criterion used in [Fas07, Chapter 2] is based on the trade-off principle, i.e., the fact that for small ε the error improves while the condition number grows.

"Best": smallest ε for which MATLAB does not issue a near-singular warning.

Remark

Note that this criterion only makes sense if we limit ourselves to using the standard basis.



The Power Function as Indicator for a "Good" Shape Parameter

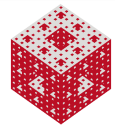
Another strategy is suggested by the standard RKHS error analysis of Chapter 8. We showed there that

$$|f(\boldsymbol{x}) - s(\boldsymbol{x})| \leq P_{K,\mathcal{X}}(\boldsymbol{x})\|f\|_{\mathcal{H}_K(\Omega)},$$

where $P_{K,\mathcal{X}}$ denotes the power function.

This estimate decouples the interpolation error into

- a component independent of the data function f
- and one depending on f.

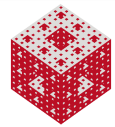


Once we have decided on a kernel K and a data set $\mathcal{X}$ we can use the power function based on scaled versions of K to optimize the error component that is independent of f.

Advantage: objective and does not depend on any knowledge of the data function

Disadvantage: will not be optimal since the second component of the error bound also depends on the kernel via the native space norm (which changes when K is scaled).

We will improve on this method later.

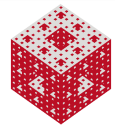


The power function can be computed via

$$P_{K,\mathcal{X}}(\boldsymbol{x}) = \sqrt{K(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}(\boldsymbol{x})^T \mathrm{K}^{-1} \boldsymbol{k}(\boldsymbol{x})},$$

where K is the interpolation matrix and $\boldsymbol{k} = [K(\cdot, \boldsymbol{x}_1), \ldots, K(\cdot, \boldsymbol{x}_N)]^T$.

This formula is implemented on lines 11–14 in the MATLAB program `Powerfunction2D.m`.

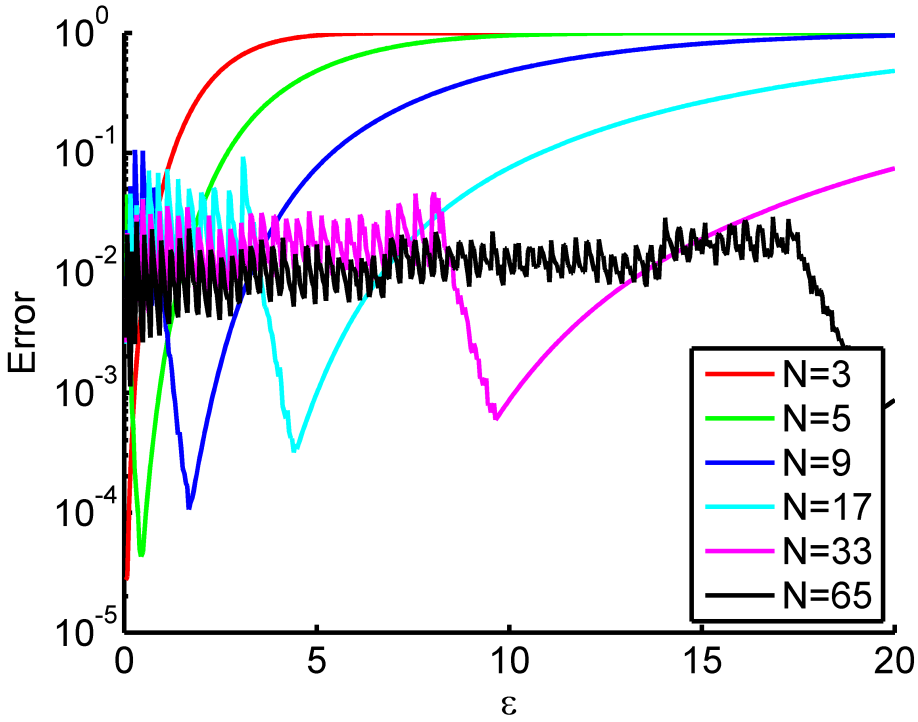


Program (Powerfunction2D.m)

```
 1  rbf = @(e,r) exp(-(e*r).^2);  mine = 0.1; maxe = 20;
 2  ne = 500; ep = linspace(mine,maxe,ne);
 3  N = 81; gridtype = 'u';
 4  dsites = CreatePoints(N,2,gridtype);  ctrs = dsites;
 5  neval = 20; M = neval^2;
 6  epoints = CreatePoints(M,2,'u');
 7  DM_k = DistanceMatrix(ctrs,epoints);
 8  DM_data = DistanceMatrix(dsites,ctrs);
 9  for i=1:length(ep)
10    IM = rbf(ep(i),DM_data);  Mk = rbf(ep(i),DM_k);
11    invIM = pinv(IM); phi0 = rbf(ep(i),0);
12    for j=1:M
13      PF(j)=real(sqrt(phi0-Mk(:,j)'*invIM*Mk(:,j)));
14    end
15    maxPF(i) = max(PF);
16  end
17  fprintf('Smallest maximum norm: %e\n', min(maxPF))
18  fprintf('at epsilon = %f\n',ep(maxPF==min(maxPF)))
19  figure; semilogy(ep,maxPF,'b');
```

Remark

- *We compute the inverse of* K *using the function* `pinv` *which is based on the singular value decomposition of* K *and therefore guarantees greater stability.*
- *Due to roundoff some of the arguments of the* `sqrt` *function on line 18 come out negative. This explains the use of the* `real` *command.*
- *The vectors* $\boldsymbol{k}(\boldsymbol{x})$ *are just columns of the evaluation matrix* `Mk` *if* $\boldsymbol{x}$ *is taken from the grid of evaluation points we used earlier for error computations and plotting purposes.*
- *Except for the loop over the shape parameter* ε *(lines 9–16) the rest of the program is similar to earlier code.*



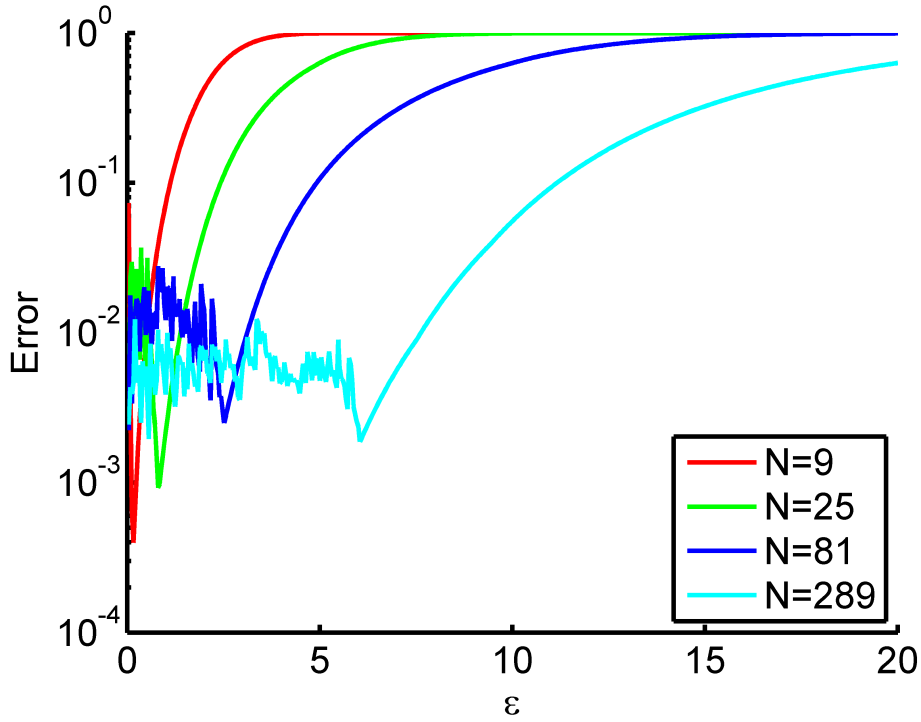


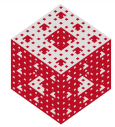
Figure: Optimal ε curves based on power functions for Gaussians in 1D (left) and 2D (right) for various choices of N uniform points.

Remark

Clearly, even for the small data sets considered here, the numerical instability, i.e., large condition number of the interpolation matrix K*, plays a significant role.*

1D			2D		
N	ε_{opt}	cond(K)	N	ε_{opt}	cond(K)
3	0.04	1.8749e+007	9	0.16	5.3534e+009
5	0.44	5.7658e+007	25	0.84	1.0211e+011
9	1.72	6.5682e+008	81	0.04	2.0734e+019
17	4.48	6.1306e+009	289	0.56	1.2194e+020
33	9.60	5.4579e+010			
65	19.52	1.2440e+011			

Table: Optimal ε values based on power functions for Gaussians in 1D and 2D for various choices of N uniform points.

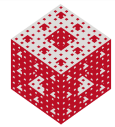


Alternatively, the power function can be computed via

$$P_{K,\mathcal{X}}(\boldsymbol{x}) = \sqrt{K(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}(\boldsymbol{x})^T \mathrm{K}^{-1} \boldsymbol{k}(\boldsymbol{x})}$$
$$= \sqrt{K(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}(\boldsymbol{x})^T \overset{\star}{\boldsymbol{u}}(\boldsymbol{x})},$$

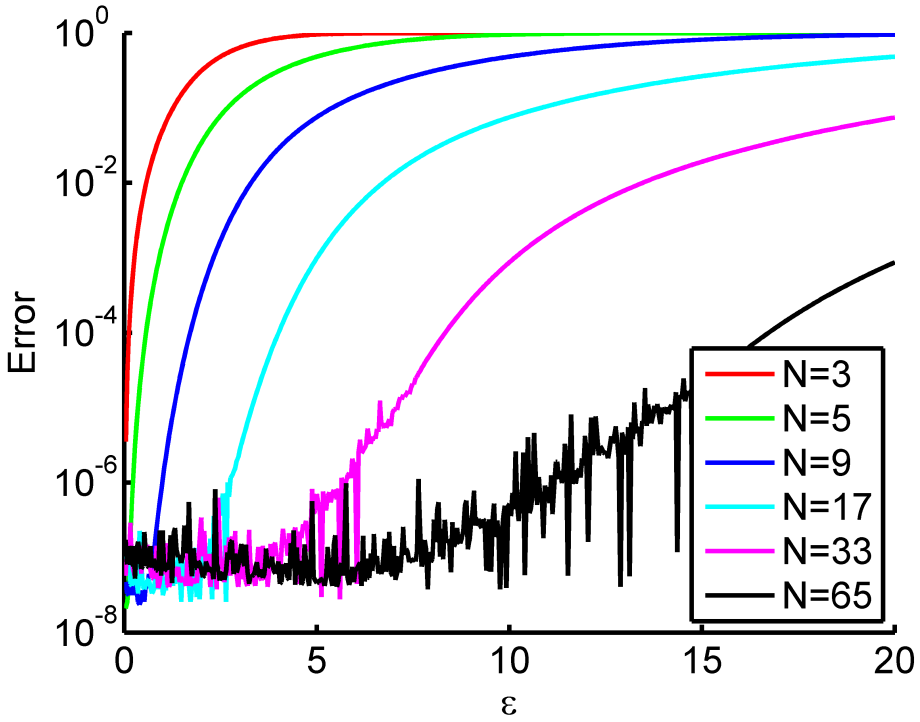
where $\boldsymbol{k}$ and K are as before and $\overset{\star}{\boldsymbol{u}}(\boldsymbol{x})$ is the vector of cardinal functions.

This formula is implemented on line 12 in the MATLAB program `Powerfunction2Dnew.m`.



Program (Powerfunction2Dnew.m)

```
 1  rbf = @(e,r) exp(-(e*r).^2);  mine = 0.1; maxe = 20;
 2  ne = 500; ep = linspace(mine,maxe,ne);
 3  N = 81; gridtype = 'u';
 4  dsites = CreatePoints(N,2,gridtype);  ctrs = dsites;
 5  neval = 20; M = neval^2;
 6  epoints = CreatePoints(M,2,'u');
 7  DM_k = DistanceMatrix(ctrs,epoints);
 8  DM_data = DistanceMatrix(dsites,ctrs);
 9  for i=1:length(ep)
10    IM = rbf(ep(i),DM_data);  Mk = rbf(ep(i),DM_k);
11    phi0 = rbf(ep(i),0); cardfuns = IM\Mk;
12    PF = real(sqrt(phi0-sum(Mk.*cardfuns,1)));
13    maxPF(i) = max(PF);
14  end
15  fprintf('Smallest maximum norm: %e\n', min(maxPF))
16  fprintf('at epsilon = %f\n',ep(maxPF==min(maxPF)))
17  figure; semilogy(ep,maxPF,'b');
```



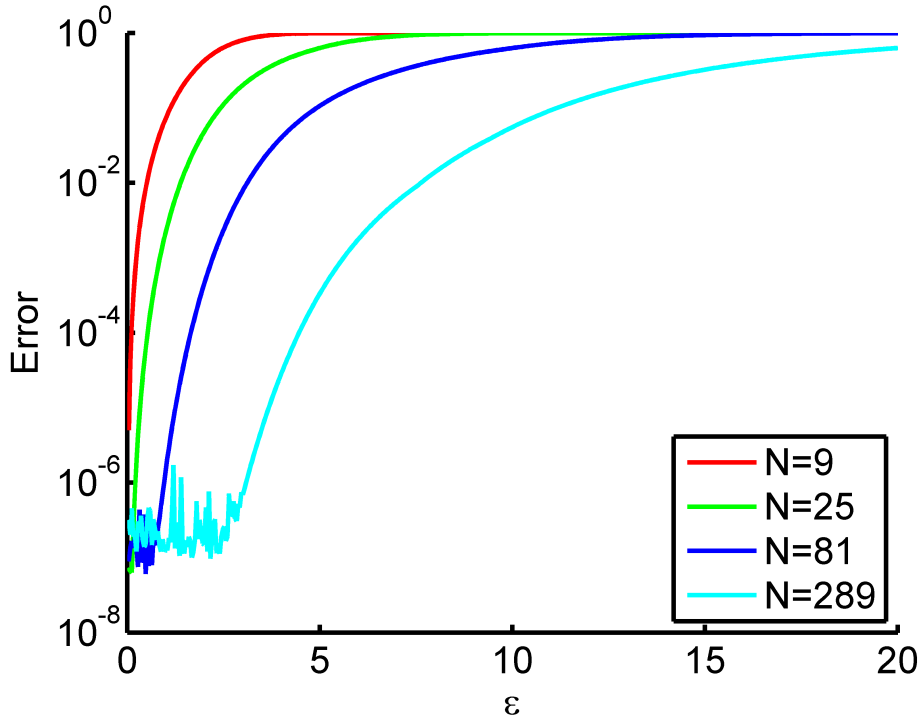


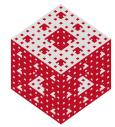
Figure: Optimal ε curves based on alternative implementation of power functions for Gaussians in 1D (left) and 2D (right) for various choices of N uniform points.

Remark

The results now are less affected by numerical instability, and we see pretty clearly that the power function used as an optimal shape parameter criterion always favors small values of ε. From our previous experiments we know that this is certainly not always the case.

1D			2D		
N	ε_{opt}	cond(K)	N	ε_{opt}	cond(K)
3	0.04	1.8749e+007	9	0.04	3.5195e+014
5	0.04	1.1671e+016	25	0.12	1.5675e+018
9	0.40	5.3461e+016	81	0.48	6.0182e+018
17	1.48	2.5020e+017	289	2.16	4.7005e+019
33	5.60	4.6681e+017			
65	14.76	4.8623e+017			

Table: Optimal ε values based on alternate implementation of power functions for Gaussians in 1D and 2D for various choices of N uniform points.



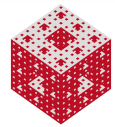
Systematic – Statistics-based – Approaches

Much more systematic approaches have been suggested in the statistics literature for a long time (see, e.g., [Wah90] and many other references).

In the radial basis community one can find papers such as [FZ07, HH99, Rip99, Sch11] that employ some of the methods we are about to explain.

We repeat some of the discussion on “good” shape parameter selection from [Fas07]. However, several other criteria – such as generalized cross validation and maximum likelihood estimation – have been investigated and/or proposed in our meshfree seminar over the past couple of years.

We therefore also add some more recent insights based on [Fas08, Hic09, Mon11, MF14].



Leave-One-Out Cross-Validation (LOOCV)

A strategy for finding an "optimal" shape parameter is to use a cross validation approach which originated in the statistics literature.

- Proposed by
  - [All74] as PRESS (Prediction Sum of Squares) for ridge regression
  - [CW79, GHW79]) for smoothing splines

  to find optimal smoothing parameter μ

$$(\mathrm{K} + \mu \mathrm{I})\boldsymbol{c} = \boldsymbol{y}$$

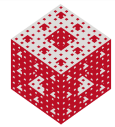
  Fundamental assumption: noisy data that require smoothed fit
- [Rip99] addressed optimization of the shape parameter of RBF interpolation systems

$$\mathrm{K}_\varepsilon \boldsymbol{c} = \boldsymbol{y},$$

  where

$$\mathrm{K}_{\varepsilon,ij} = K_\varepsilon(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\varepsilon \|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$$

  Fundamental assumption: exact data that require exact fit

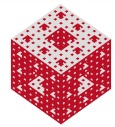


In this algorithm an "optimal" value of ε is selected by minimizing the (least squares) error for a fit to the data based on an interpolant for which one of the centers was "left out".

Advantage: dependence of the error on the data function is also taken into account.

Remark

- *Therefore, the predicted "optimal" shape parameter is closer to the one we found via the trial and error approach (for which we had to assume knowledge of the exact solution).*
- *A similar strategy was proposed earlier in [GCK96] for the solution of elliptic partial differential equations via the dual reciprocity method based on multiquadric interpolation.*



LOOCV: How it works

Let $s^{[k]}$ be the kernel interpolant to the training data $\{y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_N\}$, i.e.,

$$s^{[k]}(\boldsymbol{x}) = \sum_{\substack{j=1 \\ j \neq k}}^{N} c_j^{[k]} K(\boldsymbol{x}, \boldsymbol{x}_j),$$

such that

$$s^{[k]}(\boldsymbol{x}_i) = y_i, \qquad i = 1, \ldots, k-1, k+1, \ldots, N,$$

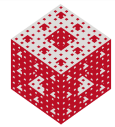
and let $e_k(\varepsilon)$ be the error

$$e_k(\varepsilon) = y_k - s^{[k]}(\boldsymbol{x}_k)$$

at the one validation point $\boldsymbol{x}_k$ not used to determine the interpolant.

Find

$$\varepsilon_{opt} = \underset{\varepsilon}{\operatorname{argmin}} \|\boldsymbol{e}(\varepsilon)\|, \qquad \boldsymbol{e} = (e_1, \ldots, e_N)^T$$



In [Rip99] the author presented examples based on use of the ℓ_1 and ℓ_2 norms.

We will mostly use the maximum norm (see line 15 in the code below).

By adding a loop over ε we can compare the error norms for different values of the shape parameter, and choose that value of ε that yields the minimal error norm as the optimal one.

Problem

This naive implementation of the leave-one-out algorithm is rather expensive (on the order of N^4 operations)

LOOCV Done Efficiently

We now derive an efficient formula for LOOCV as explained in [Hic09]. We use the following sets of centers (and coinciding data points):

$$\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}, \quad \mathcal{T} = \mathcal{X} \setminus \{\boldsymbol{x}_k\}, \quad \text{and} \quad \mathcal{V} = \{\boldsymbol{x}_k\}.$$

Then we partition

$$\mathrm{K} = \begin{bmatrix} \mathrm{K}_{tt} & \mathrm{K}_{tv} \\ \mathrm{K}_{tv}^T & \mathrm{K}_{vv} \end{bmatrix}, \mathrm{A} = \mathrm{K}^{-1} = \begin{bmatrix} \mathrm{A}_{tt} & \mathrm{A}_{tv} \\ \mathrm{A}_{tv}^T & \mathrm{A}_{vv} \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_t \\ \boldsymbol{y}_v \end{bmatrix}, \boldsymbol{c} = \begin{bmatrix} \boldsymbol{c}_t \\ \boldsymbol{c}_v \end{bmatrix}.$$

Here the subscript t corresponds to the training points in $\mathcal{T}$ and v corresponds to the validation point in $\mathcal{V}$[1], e.g., the block K_{tv} is generated using training points to evaluate and the validation point as center. For the matrix A we can't make such a direct connection. Only the sizes of the blocks match the cardinality of the sets $\mathcal{T}$ and $\mathcal{V}$.

[1] Note that there is nothing here that forces us to use only a single validation point $\boldsymbol{x}_k$, but that's how we'll think about this to keep the connection to LOOCV.

To compute the partial interpolant $s^{[k]}$ we need to solve the $(N-1) \times (N-1)$ linear system

$$\mathrm{K}_{tt} \boldsymbol{c}_t = \boldsymbol{y}_t \quad \Longrightarrow \quad \boldsymbol{c}_t = (\mathrm{K}_{tt})^{-1} \boldsymbol{y}_t.$$

Note that

$$(\mathrm{K}_{tt})^{-1} \neq \mathrm{A}_{tt} \quad \Longleftrightarrow \quad (\mathrm{K}_{tt})^{-1} \neq \left(\mathrm{K}^{-1}\right)_{tt}$$

as mentioned on the previous slide.

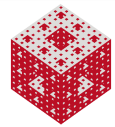
In order to get an approximation for $\boldsymbol{y}_v$ $(= f(\boldsymbol{x}_k))$ we use the evaluation matrix K_{tv}^T, i.e., evaluate the partial interpolant at the point(s) in $\mathcal{V}$:

$$\boldsymbol{y}_v \approx \mathrm{K}_{tv}^T \boldsymbol{c}_t = \mathrm{K}_{tv}^T (\mathrm{K}_{tt})^{-1} \boldsymbol{y}_t.$$

This produces the (inefficient) error formula

$$\boldsymbol{e}_k(\varepsilon) = \boldsymbol{y}_v - \mathrm{K}_{tv}^T (\mathrm{K}_{tt})^{-1} \boldsymbol{y}_t,$$

where the inverse is of size $(N-1) \times (N-1)$.

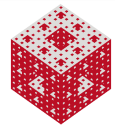


From the "full" interpolation system $\mathrm{K}\boldsymbol{c} = \boldsymbol{y}$ we know that $\boldsymbol{c} = \mathrm{K}^{-1}\boldsymbol{y} = \mathrm{A}\boldsymbol{y}$ so that

$$\begin{bmatrix} \boldsymbol{c}_t \\ \boldsymbol{c}_v \end{bmatrix} = \begin{bmatrix} \mathrm{A}_{tt}\boldsymbol{y}_t + \mathrm{A}_{tv}\boldsymbol{y}_v \\ \mathrm{A}_{tv}^T\boldsymbol{y}_t + \mathrm{A}_{vv}\boldsymbol{y}_v \end{bmatrix}.$$

In particular,

$$\boldsymbol{c}_v = \mathrm{A}_{tv}^T\boldsymbol{y}_t + \mathrm{A}_{vv}\boldsymbol{y}_v. \tag{1}$$



We also know that $AK = I$, i.e.,

$$\begin{bmatrix} A_{tt} & A_{tv} \\ A_{tv}^T & A_{vv} \end{bmatrix} \begin{bmatrix} K_{tt} & K_{tv} \\ K_{tv}^T & K_{vv} \end{bmatrix} = \begin{bmatrix} I_{tt} & \mathbf{0}_{tv} \\ \mathbf{0}_{tv}^T & I_{vv} \end{bmatrix},$$

which yields in particular

$$A_{tv}^T K_{tt} + A_{vv} K_{tv}^T = \mathbf{0}_{tv}^T \quad \text{or} \quad A_{tv}^T = -A_{vv} K_{tv}^T (K_{tt})^{-1}. \tag{2}$$

Taking (1) and (2) together we get

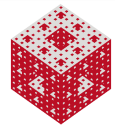
$$\boldsymbol{c}_v = -A_{vv} K_{tv}^T (K_{tt})^{-1} \boldsymbol{y}_t + A_{vv} \boldsymbol{y}_v,$$

and this is equivalent to

$$(A_{vv})^{-1} \boldsymbol{c}_v = \boldsymbol{y}_v - K_{tv}^T (K_{tt})^{-1} \boldsymbol{y}_t = e_k(\varepsilon).$$

For LOOCV the "matrix" A_{vv} is just a scalar and we get

$$e_k(\varepsilon) = \frac{\boldsymbol{c}_v}{A_{vv}} = \frac{\boldsymbol{c}_v}{\left(K^{-1}\right)_{vv}}.$$



The efficient formula for the LOOCV criterion just derived, i.e.,

$$e_k(\varepsilon) = \frac{c_k}{\mathrm{K}^{-1}_{kk}}, \qquad k = 1, \ldots, N, \tag{3}$$

with

c_k: k^{th} coefficient of full interpolant s

K^{-1}_{kk}: k^{th} diagonal element of inverse of corresponding interpolation matrix

was given by [Rip99] (and also [Wah90]).

Remark

- *Since both c_k and K^{-1} need to be computed only once for each value of ε this results in $\mathcal{O}(N^3)$ computational complexity.*
- *All entries in the error vector **e** can be computed in a single statement in* MATLAB *if we vectorize the component formula (3):*

```
errorvector = (invK*y)./diag(invK);
```

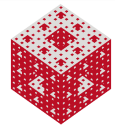
Program (LOOCV2D.m)

```
 1  K = @(e,r) exp(-(e*r).^2);
 2  mine = 0; maxe = 20; ne = 500;
 3  ep = linspace(mine,maxe,ne);
 4  N = 81; gridtype = 'u';
 5  dsites = CreatePoints(N,2,gridtype);  ctrs = dsites;
 6  neval = 20; M = neval^2;
 7  epoints = CreatePoints(M,2,'u');
 8  testfunction = @(x,y) sinc(x).*sinc(y);
 9  y = testfunction(dsites(:,1),dsites(:,2));
10  DM = DistanceMatrix(dsites,ctrs);
11  for i=1:length(ep)
12      KM = K(ep(i),DM);
13      invK = pinv(KM);
14      EF = (invK*y)./diag(invK);
15      maxEF(i) = norm(EF(:),inf);
16  end
17  fprintf('Smallest maximum norm: %e\n', min(maxEF))
18  fprintf('at epsilon = %f\n',ep(maxEF==min(maxEF)))
19  figure; semilogy(ep,maxEF,'b');
```

The sinc function used on line 8 is not a standard MATLAB function (it is part of the Signal Processing Toolbox). Therefore we provide some code for it:

Program (`sinc.m`)

```
1  function f = sinc(x)
2  f = ones(size(x));
3  nz = find(x~=0);
4  f(nz) = sin(pi*x(nz))./(pi*x(nz));
```



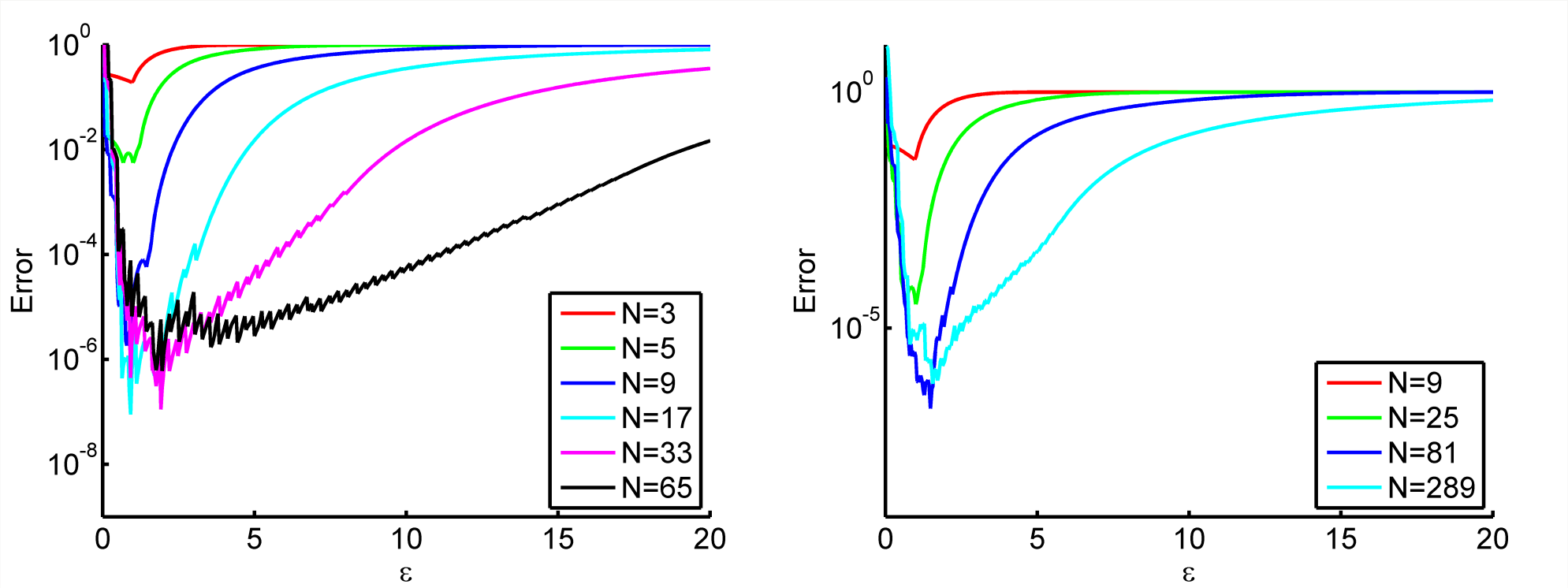
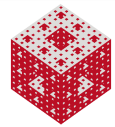
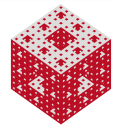


Figure: Optimal ε curves based on leave-one-out cross validation for interpolation to the sinc function with Gaussians in 1D (left) and 2D (right) for various choices of N uniform points.



1D		2D	
N	ε_{opt}	N	ε_{opt}
3	0.96	9	0.96
5	1.00	25	1.00
9	0.80	81	1.48
17	0.92	289	1.60
33	1.92		
65	1.76		

Table: Optimal ε values based on leave-one-out cross validation for interpolation to the sinc function with Gaussians in 1D and 2D for various choices of N uniform points.

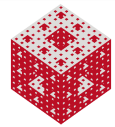


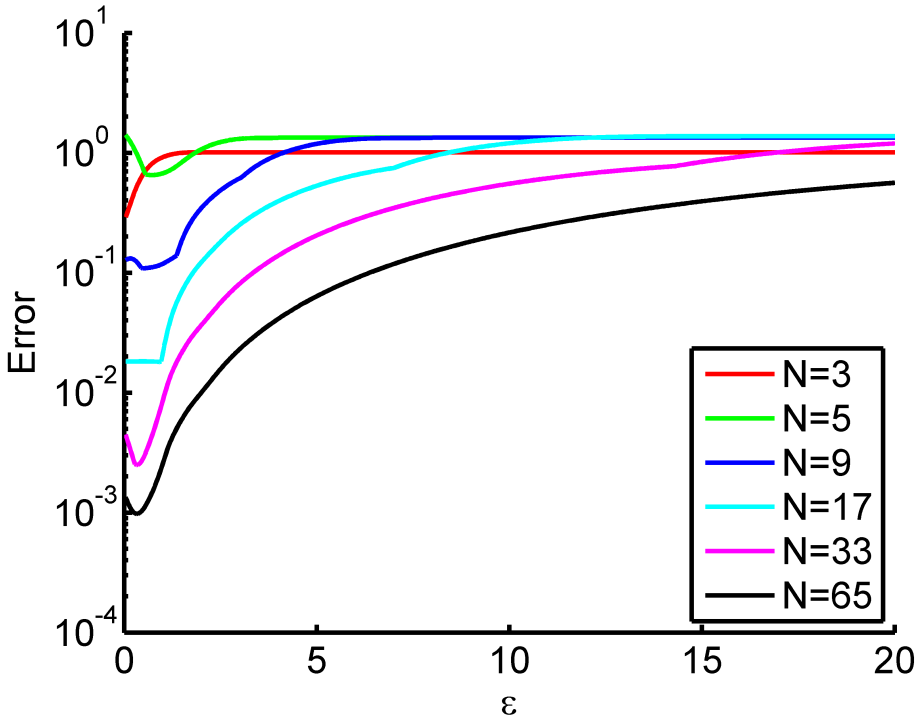
Remark

We will later see that the shape of LOOCV error curves is often quite similar to that of the true error curves. Thus, LOOCV can be recommended as a good method for selecting an "optimal" shape parameter ε since for this method no knowledge of the exact error is needed.

Similar conclusions hold in general, i.e., for

- other kernels,
- other test functions,
- other data distributions, and
- other space dimensions.





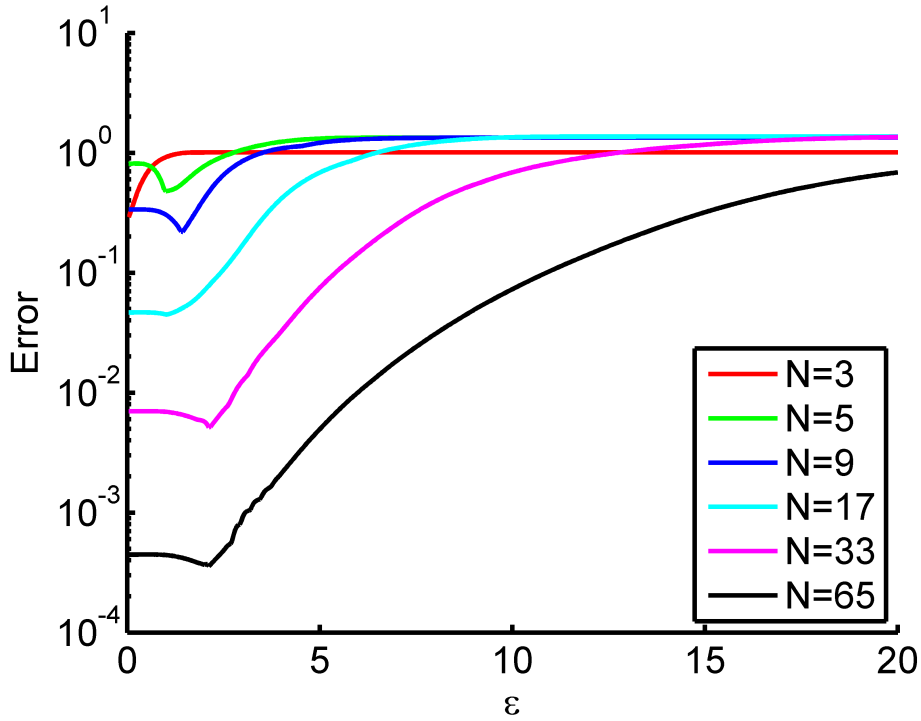


Figure: Optimal ε curves based on leave-one-out cross validation for interpolation to 1D Franke's function with Wendland's function $\varphi_{3,1}(r) = (1-\varepsilon r)_+^4(4\varepsilon r + 1)$ for various choices of N uniform points (left) and Chebyshev points (right).

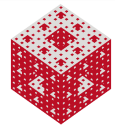
Remark

All computations are stable, and the optimal scale parameter is quite small, i.e., the support radius of the compactly supported basic function is chosen to be very large. The best results for compactly supported functions are obtained with dense matrices.

If we are not interested in the ε-curves displayed above, but only want to find a good value of the shape parameter as quickly as possible, then we can use the MATLAB function `fminbnd` to find the minimum of the cost function for ε as shown in `LOOCV2Dmin.m`.

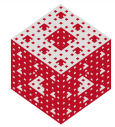
Program (`CostEpsilon.m`)

```
1  function ceps = CostEpsilon(ep,r,K,y)
2  KM = K(ep,r);
3  invK = pinv(K);
4  EF = (invK*y)./diag(invK);
5  ceps = norm(EF(:),inf);
```



Program (LOOCV2Dmin.m)

```
 1  K = @(e,r) exp(-(e*r).^2);
 2  mine = 0; maxe = 20;
 3  N = 81; gridtype = 'u';
 4  dsites = CreatePoints(N,2,gridtype);
 5  ctrs = dsites;
 6  testfunction = @(x,y) sinc(x).*sinc(y);
 7  y = testfunction(dsites(:,1),dsites(:,2));
 8  DM = DistanceMatrix(dsites,ctrs);
 9a [ep,f]=fminbnd(@(ep) CostEpsilon(ep,DM,K,y),...
 9b                      mine,maxe);
10  fprintf('Smallest maximum norm: %e\n', f)
11  fprintf('at epsilon = %f\n', ep)
```



Leave Half/Third Out Cross-Validation

If our training set is chosen to be much smaller than for LOOCV, then we can do cross-validation with

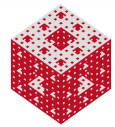
- leave half out:

$$\mathcal{T} = \text{one half of } \mathcal{X}, \qquad \mathcal{V} = \text{other half of } \mathcal{X}.$$

In MATLAB this can, e.g., be accomplished with

```
h1 = 1:2:N;
h2 = setdiff(1:N,h1);
x_train = x(h1);
y_train = f(x_train);
x_valid = x(h2);
y_valid = f(x_valid);
```

and then swapping the index sets.



Leave Half/Third Out Cross-Validation

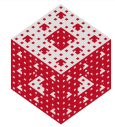
- Leave third out:

$$\mathcal{T} = \text{two thirds of } \mathcal{X}, \qquad \mathcal{V} = \text{remaining third of } \mathcal{X}.$$

In MATLAB this can, e.g., be accomplished with

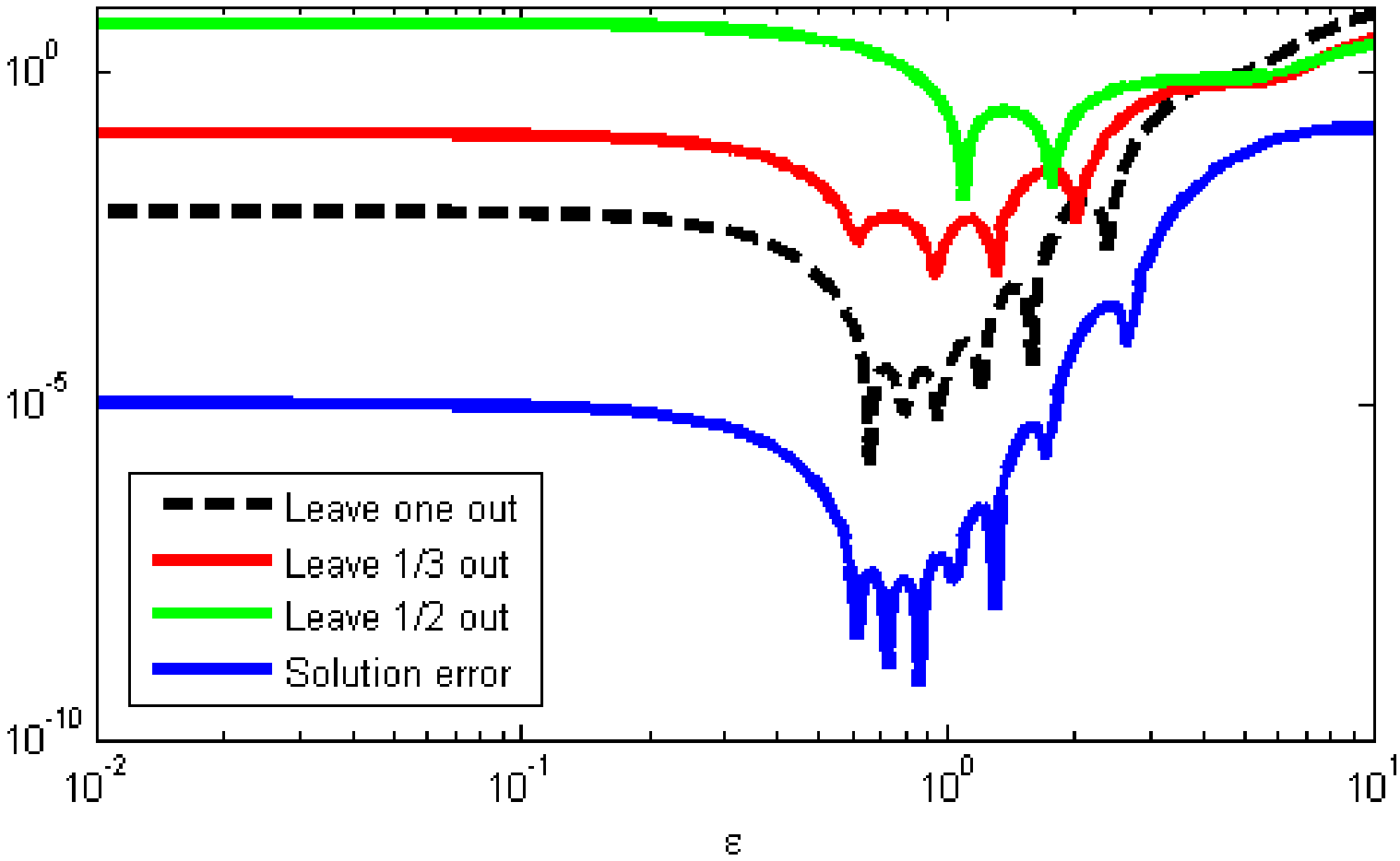
```
t1 = 1:3:N;
t2 = 2:3:N;
t3 = setdiff(1:N,[t1,t2]);
x_train = x([t1,t2]);
y_train = f(x_train);
x_valid = x(t3);
y_valid = f(x_valid);
```

and then permuting the indices.



Example (Using GaussQR)

Determine the optimal ε for Gaussian interpolation using $N = 18$ evenly spaced samples from $f(x) = \cos 2\pi x$ in $[-1, 1]$.



Example (cont.)

Since the LOOCV criterion requires the inverse of K (which may be very ill-conditioned), we use the Hilbert-Schmidt SVD

$$K = \Psi \Lambda_1 \Phi_1^T,$$

accurate to within machine precision (i.e., truncation length $M > N$). Then

$$K^{-1} = \Phi_1^{-T} \Lambda_1^{-1} \Psi^{-1}.$$

- The matrices Ψ and Φ_1 are usually "well-behaved".
- But Λ_1 by itself still contains the basic ill-conditioning, i.e., potentially very small eigenvalues.
- We therefore use the pseudoinverse $\Lambda_1^\dagger$ instead of Λ_1^{-1}, i.e., we drop some of the smallest eigenvalues of the kernel K.

Remark

Note that truncating the Hilbert-Schmidt SVD is fundamentally different from performing a standard SVD of K and then truncating that.

Example (cont.)

We use the functions `gqr_solveprep` and `gqr_phi` from the GaussQR library to do the calculations needed for the LOOCV ε-curve:

```
for ep=epvec
  GQR = gqr_solveprep(0,x,ep,alpha);
  Phi = gqr_phi(GQR,x);
  Phi1 = Phi(:,1:N);
  Psi = Phi*[eye(N);GQR.Rbar];
  invPsi = pinv(Psi);
  invPhi1 = pinv(Phi1');
  nu = (2*ep/alpha)^2;
  Lambda1 = diag((nu/(2+nu+2*sqrt(1+nu))).^(1:N));
  invLambda1 = pinv(Lambda1);
  invK = invPhi1*invLambda1*invPsi;
  EF = (invK*y)./diag(invK);
  loocvvec(k) = norm(EF,1); k=k+1;
end
```

Alternatively, we can directly apply the Hilbert–Schmidt SVD in the derivation of the CV criterion. Then one can show that

$$CV(\varepsilon;\mathcal{V}) = \sum_{\boldsymbol{x}_v \in \mathcal{V}} \left\| \mathrm{B}_{vv}^{-1} \boldsymbol{b}_v \right\| = \sum_{\boldsymbol{x}_v \in \mathcal{V}} \left\| (\Psi_{vv} - \Psi_{vt}\Psi_{tt}^{-1}\Psi_{tv})^{-1} \boldsymbol{b}_v \right\|,$$

which is almost certainly more stable to compute.

Here the matrix B_{vv} is related to the HS-SVD via $\mathrm{B} = \Psi^{-1}$, i.e.,

$$\begin{pmatrix} \Psi_{tt} & \Psi_{tv} \\ \Psi_{vt} & \Psi_{vv} \end{pmatrix} \begin{pmatrix} \boldsymbol{b}_t \\ \boldsymbol{b}_v \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_t \\ \boldsymbol{y}_v \end{pmatrix} \quad \Longleftrightarrow \quad \begin{pmatrix} \boldsymbol{b}_t \\ \boldsymbol{b}_v \end{pmatrix} = \begin{pmatrix} \mathrm{B}_{tt} & \mathrm{B}_{tv} \\ \mathrm{B}_{vt} & \mathrm{B}_{vv} \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_t \\ \boldsymbol{y}_v \end{pmatrix}$$

Remark

Unfortunately, this formula is not as computationally efficient as the standard LOOCV formula since the matrix B_{vv}^{-1} still depends on the training and validation sets.

Generalization of Cross-Validation

Instead of the LOOCV cost function

$$\text{LOOCV}(\varepsilon) = \|\boldsymbol{e}(\varepsilon)\|_2 = \sqrt{\sum_{k=1}^{N} \left(\frac{c_k}{\text{K}^{-1}_{kk}} \right)^2}$$

we use the average of the diagonal elements of K^{-1}, i.e.,

$$\begin{aligned} \text{GCV}(\varepsilon) &= = \sqrt{\sum_{k=1}^{N} \left(\frac{c_k}{\frac{1}{N}\sum_{j=1}^{N} \text{K}^{-1}_{jj}} \right)^2} = \frac{N\|\boldsymbol{c}\|_2}{\text{trace}(\text{K}^{-1})} \\ &= \sqrt{\boldsymbol{y}^T \text{K}^{-2} \boldsymbol{y}}\, \mu_h\left(\lambda(\text{K})\right) \end{aligned}$$

Here we introduced μ_h, the harmonic mean.
Note that $\|\boldsymbol{c}\|_2 = \sqrt{\boldsymbol{y}^T \text{K}^{-2} \boldsymbol{y}}$ follows immediately from the interpolation system $\text{K}\boldsymbol{c} = \boldsymbol{y}$.

LOOCV via Weighted Harmonic Means

We know

$$\text{LOOCV}(\varepsilon) = \sqrt{\sum_{k=1}^{N} \left(\frac{c_k}{\text{K}^{-1}_{kk}} \right)^2}$$

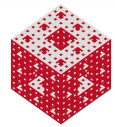
Using an (orthogonal) eigenvalue decomposition $\text{K}^{-1} = \text{U}\Lambda^{-1}\text{U}^T$

$$\sum_{k=1}^{N} \left(\frac{c_k}{\text{K}^{-1}_{kk}} \right)^2 = \sum_{k=1}^{N} \left(\frac{c_k}{\sum_{j=1}^{N} \frac{\text{U}^2_{kj}}{\lambda_j}} \right)^2 = \sum_{k=1}^{N} c_k^2 \left(\frac{\sum_{j=1}^{N} \text{U}^2_{kj}}{\sum_{j=1}^{N} \text{U}^2_{kj} \frac{1}{\lambda_j}} \right)^2$$

$$= \sum_{k=1}^{N} c_k^2 \mu^2_{h,\boldsymbol{u}_k}(\lambda(\text{K}))$$

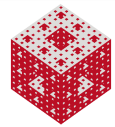
$\mu_{h,\boldsymbol{u}_k}$: $\boldsymbol{u}_k$-weighted harmonic mean, $\boldsymbol{u}_k$ is k^{th} row of U with $\|\boldsymbol{u}_k\|_2 = 1$

Note: $\sqrt{\sum_{k=1}^{N} c_k^2 \mu^2_{h,\mathbf{1}}(\lambda(\text{K}))} = \|\boldsymbol{c}\|_2 \mu_h(\lambda(\text{K})) = \text{GCV}(\varepsilon)$



A Few Claims For and Against CV

- In [CW79] it is shown that for large N, the expected mean square error using the GCV-optimal parameter for smoothing splines tends to the minimum expected mean square error. One should not expect good results for small N.
- [Wah90] warns that ill-conditioning may be a problem for CV.
- While [Wah90] states that the GCV criterion is an "amazingly good estimate" of the minimum expected mean square error, [Sto77] warns that CV may sometimes be far off the mark.
- While [Wah90] suggests that the GMLE estimate may not be as robust as GCV, [Neu98] proclaims the GMLE criterion to be "the clear winner".
- In a more careful study, [Ste90] shows that GCV has twice the asymptotic variance of GMLE for piecewise linear smoothing splines (and worse for higher-order smoothing splines). This conclusion assumes that the stochastic model is correctly specified.



Generalized Maximum Likelihood [Ste99, Wah90]

Consider f as an instance of a Gaussian process with covariance kernel K and data $(\boldsymbol{x}_i, y_i)$, i.e., $\mathrm{K}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ (see Chapter 4). The likelihood function is

$$\ell(\varepsilon) = \frac{\exp\left(-\frac{1}{2}\boldsymbol{y}^T \mathrm{K}^{-1}\boldsymbol{y}\right)}{\sqrt{(2\pi)^N \det(\mathrm{K})}}$$

Maximization of log-likelihood is equivalent to minimization of

$$\tilde{\ell}(\varepsilon) = \boldsymbol{y}^T \mathrm{K}^{-1}\boldsymbol{y} + \log\left(\det(\mathrm{K})\right)$$

Remark

- *This does not yet have any similarity with our other criteria in terms of means of eigenvalues.*
- *Moreover, statisticians like to allow a vertical scaling of K, i.e., $\widetilde{K} = \sigma^2 K$, which corresponds to the process variance (see, e.g., [Sch11] and Chapter 4). The likelihood above therefore corresponds to $\sigma^2 = 1$.*

We now try to make the criterion (with K replaced by $\widetilde{K} = \sigma^2 K$)

$$\tilde{\ell}(\varepsilon) = \boldsymbol{y}^T \widetilde{\mathrm{K}}^{-1} \boldsymbol{y} + \log\left(\det(\widetilde{\mathrm{K}})\right)$$

invariant under a vertical scaling of K. Then

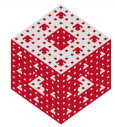
$$\tilde{\ell}(\varepsilon, \sigma^2) = \frac{1}{\sigma^2} \boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y} + N \log(\sigma^2) + \log\left(\det(\mathrm{K})\right).$$

For fixed ε, we can easily minimize this with respect to σ^2 since

$$\frac{\partial \tilde{\ell}}{\partial \sigma^2} = 0 \quad \Longrightarrow \quad -\frac{1}{\sigma^4} \boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y} + N \frac{1}{\sigma^2} = 0 \quad \Longrightarrow \quad \overset{\star}{\sigma}{}^2 = \frac{\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y}}{N}.$$

Therefore, with this optimal value of $\overset{\star}{\sigma}{}^2$, the criterion $\tilde{\ell}(\varepsilon)$ becomes

$$\tilde{\ell}(\varepsilon, \overset{\star}{\sigma}{}^2) = N - N \log(N) + N \log(\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y}) + \log\left(\det(\mathrm{K})\right).$$



Since

$$\tilde{\ell}(\varepsilon, \overset{\star}{\sigma}^2) = N - N\log(N) + N\log(\boldsymbol{y}^T \mathrm{K}^{-1}\boldsymbol{y}) + \log\left(\det(\mathrm{K})\right)$$

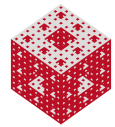
is only modified by a constant if we replace K by $\widetilde{\mathrm{K}}$ we can consider only

$$\tilde{\tilde{\ell}}(\varepsilon) = \log\left(\boldsymbol{y}^T \mathrm{K}^{-1}\boldsymbol{y}\right) + \frac{1}{N}\log\left(\det(\mathrm{K})\right),$$

which is essentially the so-called profile log-likelihood or concentrated log-likelihood.
Therefore we use

$$\mathrm{GMLE}(\varepsilon) = \left(\boldsymbol{y}^T \mathrm{K}^{-1}\boldsymbol{y}\right) \sqrt[N]{\det(\mathrm{K})} = \left(\boldsymbol{y}^T \mathrm{K}^{-1}\boldsymbol{y}\right) \mu_g\left(\lambda(\mathrm{K})\right)$$

where μ_g denotes the geometric mean.

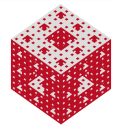


How the Hilbert–Schmidt SVD helps with MLE [MF14]

In order to stably compute $\log(\det(K))$ we can use $K = \Psi\Lambda_1\Phi_1^T$ to get

$$\log(\det(K)) = \log\det\Psi + \log\det\Lambda_1 + \log\det\Phi_1^T.$$

- The very small eigenvalues can be handled safely by taking their logarithms (since Λ_1 is diagonal).
- Φ_1^T gets inverted while forming the stable basis, and
- Ψ gets inverted while computing an interpolant, so the cost of computing $\log(\det(K))$ is negligible.



To compute the term $\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y}$ stably we recall (using $\Psi \boldsymbol{b} = \boldsymbol{y}$)

$$\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y} = (\Psi \boldsymbol{b})^T (\Psi \Lambda_1 \Phi_1^T)^{-1} \Psi \boldsymbol{b} = \boldsymbol{b}^T \Psi^T \Phi_1^{-T} \Lambda_1^{-1} \boldsymbol{b}.$$

Since

$$\Psi = \begin{pmatrix} \Phi_1 & \Phi_2 \end{pmatrix} \begin{pmatrix} \mathrm{I}_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix}$$

we can write $\Psi^T = \Phi_1^T + \Lambda_1^{-1} \Phi_1^{-1} \Phi_2 \Lambda_2 \Phi_2^T$, and so

$$\begin{aligned} \boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y} &= \boldsymbol{b}^T \Psi^T \Phi_1^{-T} \Lambda_1^{-1} \boldsymbol{b} \\ &= \boldsymbol{b}^T \Lambda_1^{-1} \boldsymbol{b} + \boldsymbol{b}^T \Lambda_1^{-1} \Phi_1^{-1} \Phi_2 \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \boldsymbol{b}. \end{aligned}$$

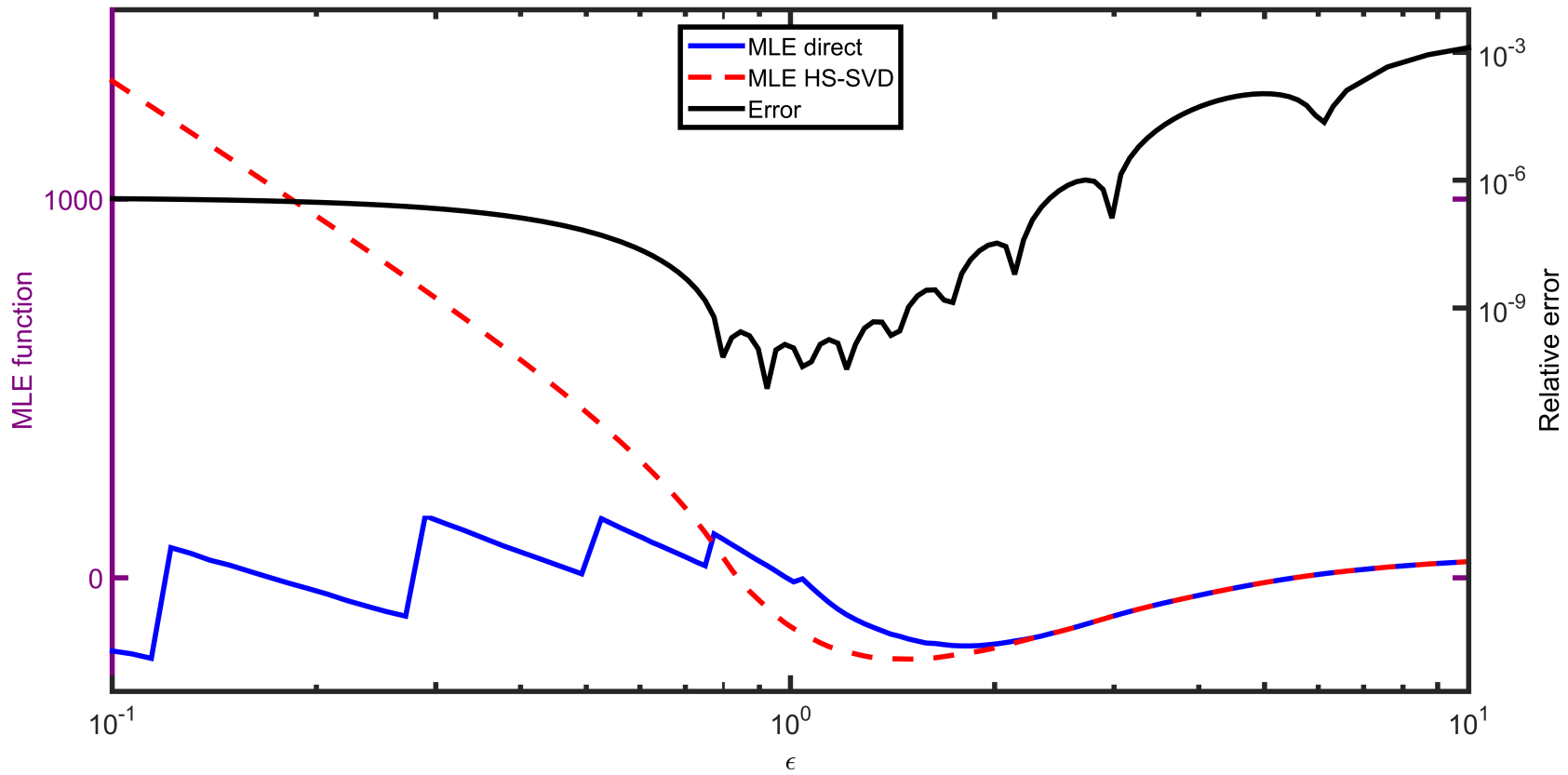
Remark

The second term above can be computed efficiently via

$$\boldsymbol{b}^T \Lambda_1^{-1} \Phi_1^{-1} \Phi_2 \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \boldsymbol{b} = \left\| \Lambda_2^{-1/2} (\Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1}) \boldsymbol{b} \right\|_2^2.$$

Example

$N = 24$ evenly spaced samples from $f(x) = \cos(3\pi x)$ in $[-1, 1]$.



MLE direct loses accuracy for $\varepsilon < 3$ and completely breaks down for $\varepsilon < 1$ due to ill-conditioning.

Hölder Means

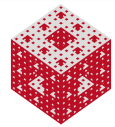
We can view GCV and GMLE as being at two ends of the spectrum in terms of the eigenvalues of K:

- GCV uses their harmonic mean,
- GMLE the geometric mean.

Using so-called Hölder means of the eigenvalues and p-type norms of the coefficient vector we can further generalize GCV and GMLE to a two-parameter family of shape parameter criteria:

$$\text{Crit}_{p,q}(\varepsilon) = \left(\boldsymbol{y}^T \text{K}^{-p} \boldsymbol{y}\right)^{1/p} \left(\frac{1}{N} \sum_{k=1}^{N} \lambda_k^q(\text{K})\right)^{1/q}$$

with $\text{GCV} = \text{Crit}_{2,-1}$ and $\text{GMLE} = \text{Crit}_{1,0}$.



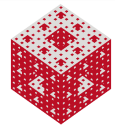
Remark

We note that

- *large eigenvalues are penalized for positive values of q*
- *and small eigenvalues are penalized for $q < 0$.*

In particular,

$$\left(\frac{1}{N}\sum_{k=1}^{N}\lambda_k^q(\mathrm{K})\right)^{1/q} \text{ corresponds to } \begin{cases}\max\left(\lambda(\mathrm{K})\right) & \text{for } q=\infty\\ \min\left(\lambda(\mathrm{K})\right) & \text{for } q=-\infty\end{cases}$$



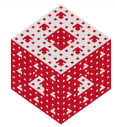
Error Bound Criterion

Remember the standard error bound for kernel interpolation from Chapter 8.
Using the representation $\check{\boldsymbol{u}}(\boldsymbol{x}) = \mathrm{K}^{-1}\boldsymbol{k}(\boldsymbol{x})$ of the cardinal functions we have for any $\boldsymbol{x} \in \Omega$

$$
\begin{aligned}
|f(\boldsymbol{x}) - s(\boldsymbol{x})| &= \left| f(\boldsymbol{x}) - \sum_{j=1}^{N} f(\boldsymbol{x}_j)\check{u}_j(\boldsymbol{x}) \right| = \left| \langle f, K(\cdot,\boldsymbol{x})\rangle_{\mathcal{H}_K(\Omega)} - \sum_{j=1}^{N} \langle f, K(\cdot,\boldsymbol{x}_j)\rangle_{\mathcal{H}_K(\Omega)} \check{u}_j(\boldsymbol{x}) \right| \\
&= \left| \langle f, K(\cdot,\boldsymbol{x}) - \sum_{j=1}^{N} K(\cdot,\boldsymbol{x}_j)\check{u}_j(\boldsymbol{x})\rangle_{\mathcal{H}_K(\Omega)} \right| = \left| \langle f, K(\cdot,\boldsymbol{x}) - \boldsymbol{k}^T(\cdot)\mathrm{K}^{-1}\boldsymbol{k}(\boldsymbol{x})\rangle_{\mathcal{H}_K(\Omega)} \right| \\
&\leq \|f\|_{\mathcal{H}_K(\Omega)} \left\| K(\cdot,\boldsymbol{x}) - \boldsymbol{k}^T(\cdot)\mathrm{K}^{-1}\boldsymbol{k}(\boldsymbol{x}) \right\|_{\mathcal{H}_K(\Omega)} = \|f\|_{\mathcal{H}_K(\Omega)} P_{K,\mathcal{X}}(\boldsymbol{x}),
\end{aligned}
$$

with $\boldsymbol{k}(\cdot) = (K(\cdot,\boldsymbol{x}_1),\ldots,K(\cdot,\boldsymbol{x}_N))^T$, and power function

$$
P_{K,\mathcal{X}}(\boldsymbol{x}) = \sqrt{K(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}(\boldsymbol{x})^T\mathrm{K}^{-1}\boldsymbol{k}(\boldsymbol{x})}.
$$



The standard error bound

$$|f(\boldsymbol{x}) - s(\boldsymbol{x})| \leq \|f\|_{\mathcal{H}_K(\Omega)} P_{K,\mathcal{X}}(\boldsymbol{x})$$

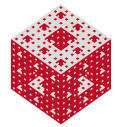
can be improved (see [GW59]) to

$$|f(\boldsymbol{x}) - s(\boldsymbol{x})| \leq \|f - s\|_{\mathcal{H}_K(\Omega)} P_{K,\mathcal{X}}(\boldsymbol{x})$$

since $f - s$ is orthogonal (in the Hilbert space inner product) to s, i.e.,

$$\begin{aligned}\|f\|^2 &= \|f - s + s\|^2 = \langle (f-s) + s, (f-s) + s \rangle \\ &= \|f - s\|^2 + 2 \underbrace{\langle f - s, s \rangle}_{=0} + \|s\|^2.\end{aligned}$$

This tighter error bound does not seem to play a significant role in the RBF literature.



Since $\|f\|_{\mathcal{H}_K(\Omega)}$ usually is not computable (remember, we do not even know f, but want to reconstruct it from the data) the standard error bound is not very useful for practical situations.
On the other hand, if we assume that our approximation s is not too bad, i.e.,

$$\|f - s\|_{\mathcal{H}_K(\Omega)} \leq \delta \|s\|_{\mathcal{H}_K(\Omega)}$$

for some not too large constant δ, then the Golomb-Weinberger improved error bound yields a computable error bound

$$|f(\boldsymbol{x}) - s(\boldsymbol{x})| \leq \delta \|s\|_{\mathcal{H}_K(\Omega)} P_{K,\mathcal{X}}(\boldsymbol{x})$$

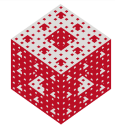
This is indeed computable since $\|s\|_{\mathcal{H}_K(\Omega)} = \sqrt{\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y}}$:

$$\begin{aligned}\|s\|^2_{\mathcal{H}_K(\Omega)} &= \langle \boldsymbol{y}^T \boldsymbol{u}^\star(\cdot), \boldsymbol{y}^T \boldsymbol{u}^\star(\cdot) \rangle_{\mathcal{H}_K(\Omega)} = \langle \boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{k}(\cdot), \boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{k}(\cdot) \rangle_{\mathcal{H}_K(\Omega)} \\ &= \boldsymbol{y}^T \mathrm{K}^{-1} \langle \boldsymbol{k}(\cdot), \boldsymbol{k}(\cdot) \rangle_{\mathcal{H}_K(\Omega)} \mathrm{K}^{-1} \boldsymbol{y} = \boldsymbol{y}^T \mathrm{K}^{-1} \mathrm{K} \mathrm{K}^{-1} \boldsymbol{y}.\end{aligned}$$

Therefore we have

$$\mathrm{EB}(\varepsilon) = \sqrt{\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y}} \|P_{K,\mathcal{X}}\|_\infty,$$

where we compute the max-norm of the power function on a discrete evaluation grid with high resolution.



Connection to Kriging Variance

Earlier we concluded that the power function (or kriging variance) by itself is not a good criterion for finding an optimal shape parameter. However, if we include the process variance in the discussion of the power function/kriging variance, i.e., if we replace K by $\widetilde{K} = \sigma^2 K$, then we get

$$\begin{aligned} P^2_{\widetilde{K},\mathcal{X}}(\boldsymbol{x}) &= \widetilde{K}(\boldsymbol{x},\boldsymbol{x}) - \widetilde{\boldsymbol{k}}(\boldsymbol{x})^T \widetilde{\mathrm{K}}^{-1} \widetilde{\boldsymbol{k}}(\boldsymbol{x}) \\ &= \sigma^2 K(\boldsymbol{x},\boldsymbol{x}) - \sigma^2 \boldsymbol{k}(\boldsymbol{x})^T (\sigma^2 \mathrm{K})^{-1} \sigma^2 \boldsymbol{k}(\boldsymbol{x}) \\ &= \sigma^2 \left(K(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}(\boldsymbol{x})^T \mathrm{K}^{-1} \boldsymbol{k}(\boldsymbol{x}) \right) = \sigma^2 P^2_{K,\mathcal{X}}(\boldsymbol{x}). \end{aligned}$$

If we now replace σ^2 by the optimal value $\overset{\star}{\sigma}{}^2$ obtained for the MLE, then

$$P^2_{\widetilde{K},\mathcal{X};\overset{\star}{\sigma}{}^2}(\boldsymbol{x}) = \frac{\boldsymbol{y}^T \mathrm{K}^{-1} \boldsymbol{y}}{N} P^2_{K,\mathcal{X}}(\boldsymbol{x}).$$

Note that this is almost identical to the error bound criterion EB(ε).

Summary

LOOCV	$\sqrt{\sum_{k=1}^{N}\left(\frac{c_k}{\mathrm{K}^{-1}_{kk}}\right)^2}$
EB	$\sqrt{\boldsymbol{y}^T\mathrm{K}^{-1}\boldsymbol{y}}\left\|\sqrt{K(\cdot,\cdot)-\boldsymbol{k}^T(\cdot)\mathrm{K}^{-1}\boldsymbol{k}(\cdot)}\right\|_\infty$
GCV	$\frac{N\|\boldsymbol{c}\|_2}{\mathrm{trace}(\mathrm{K}^{-1})}=\sqrt{\boldsymbol{y}^T\mathrm{K}^{-2}\boldsymbol{y}}\,\mu_h(\lambda(\mathrm{K}))$
GMLE	$\left(\boldsymbol{y}^T\mathrm{K}^{-1}\boldsymbol{y}\right)\sqrt[N]{\det(\mathrm{K})}=\left(\boldsymbol{y}^T\mathrm{K}^{-1}\boldsymbol{y}\right)\mu_g(\lambda(\mathrm{K}))$
$\mathrm{Crit}_{p,q}$	$\left(\boldsymbol{y}^T\mathrm{K}^{-p}\boldsymbol{y}\right)^{1/p}\left(\frac{1}{N}\sum_{k=1}^{N}\lambda_k^q(\mathrm{K})\right)^{1/q}$

Note: GCV $=\mathrm{Crit}_{2,-1}$, GMLE $=\mathrm{Crit}_{1,0}$

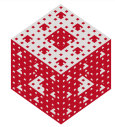
Implementation of Criteria

- all criteria require computation of K^{-1}
- some require computation of eigenvalues of K
- MLE requires computing $\log(\det(K))$

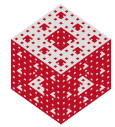
Challenge: K may be very ill-conditioned for small values of ε

Remedy: Need stable (approximate) factorization of K

- Here discussed with SVD
- also example using Riley's algorithm [Ril55]
- RBF-QR not yet implemented for all criteria
- A special algorithm to compute $\log(\det(K))$ is developed in [MF14].



```
function [c1 c2 c3 c4] = CostFuns(ep,r,rbf,rhs,DM_k)
K = rbf(ep,r);
[U,S,V] = svd(K);
lambda = diag(S); n = length(lambda);
lambda((lambda<10*eps)) = [];  nt = length(lambda);
z = zeros(1,n-nt);
invK = V*diag([1./lambda; z'])*U';
newrhs = U(:,1:nt)'*rhs;
approxNSnorm = (newrhs'*diag(1./lambda)*newrhs);
c1 = norm((invK*rhs)./diag(invK));   % LOOCV
c2 = n*(norm(invK*rhs))/(trace(invK));   % GCV
c3 = prod(nthroot(lambda,n)) * approxNSnorm;   % MLE
Mk = rbf(ep,DM_k);   % Now compute power function
K0 = rbf(ep,0);   cardfuns = invK*Mk;
powfun = real(sqrt(K0-sum(Mk.*cardfuns,1)));
c4 = max(max(powfun)) * sqrt(approxNSnorm);   % EB
```



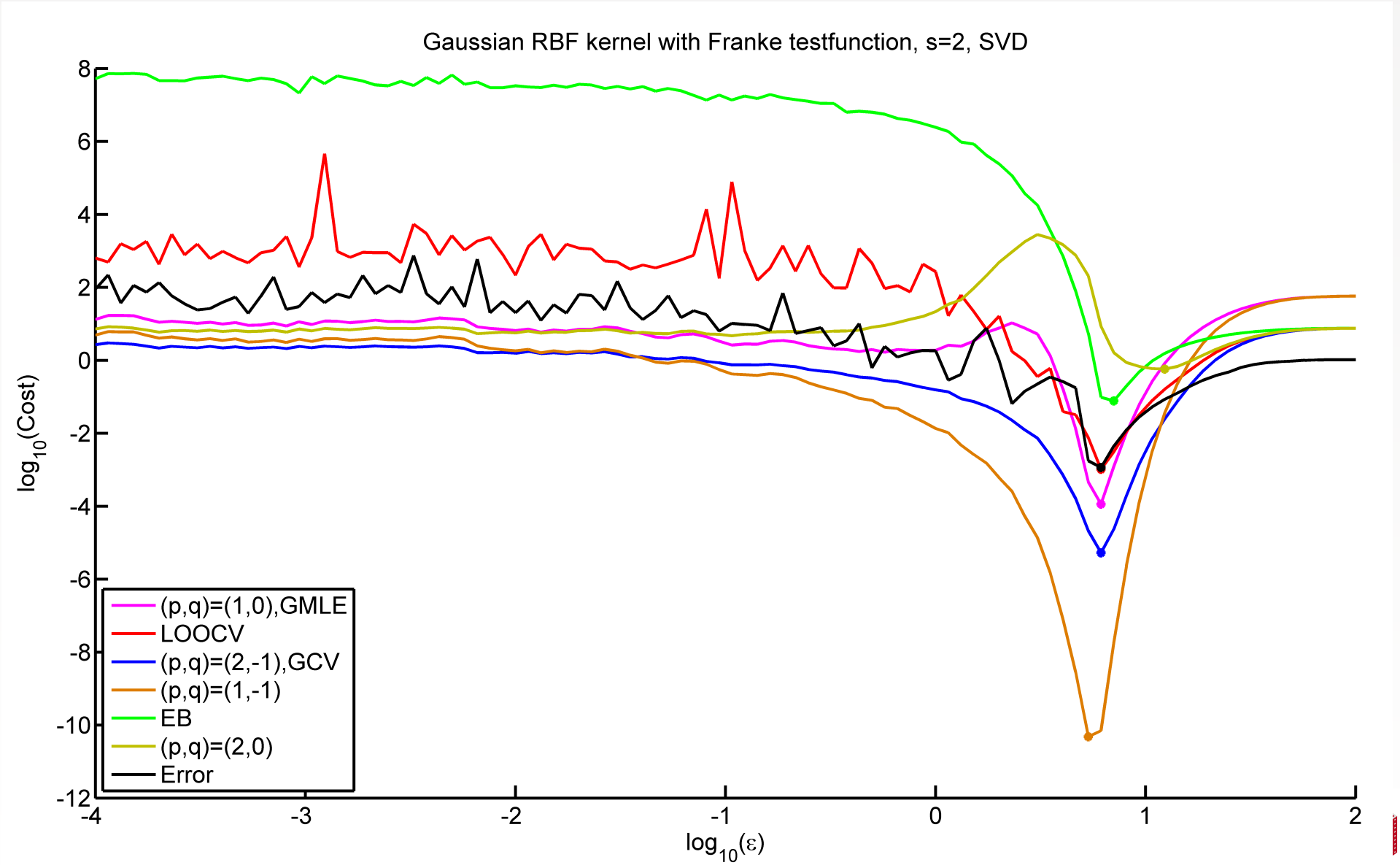
MATLAB code for all criteria.

Parameters to play with:

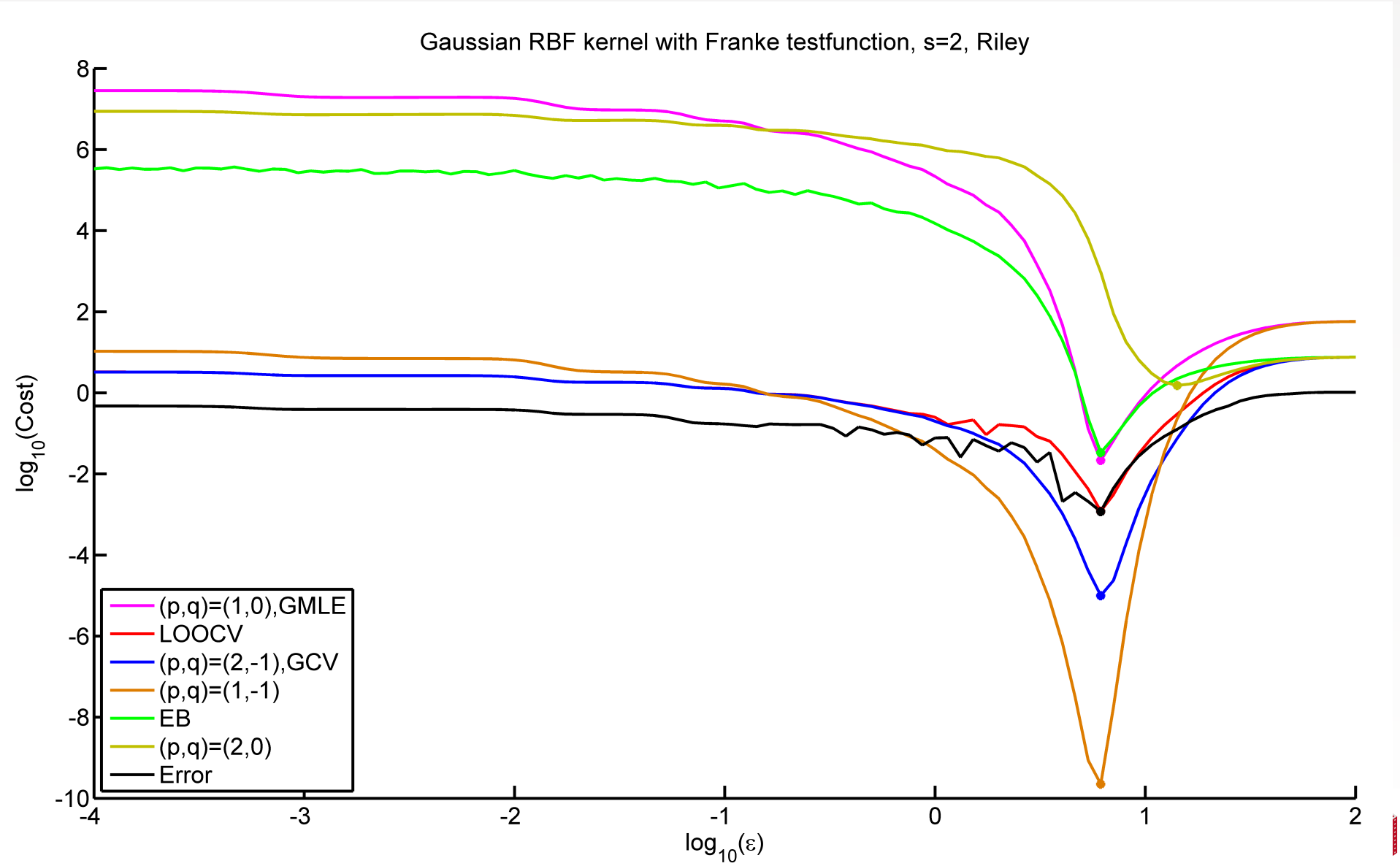
- different values of p and q along with other criteria
- different kernels, e.g.,
  - Gaussian $e^{-(\varepsilon r)^2}$
  - C^2 Matérn $(1 + \varepsilon r)e^{-\varepsilon r}$
- different space dimensions
- different test functions, e.g.,
  - Franke
  - borehole
- different data locations, e.g.,
  - uniformly gridded
  - gridded Chebyshev
  - low-discrepancy (Halton, Sobol, digital nets, etc.)
- different sizes of data sets (density of data)
- different solution algorithms, e.g., SVD, Riley, RBF-QR

Investigating this carefully is a high-dimensional approximation problem!

Example using SVD and Riley

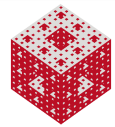


Example using SVD and Riley



Remark

- *The figure suggests that – at least for this example – all but the* $\text{Crit}_{2,0}$ *criterion perform equally well.*
- *They all locate the value of* ε *for which the actual error (black curve) is minimized quite accurately.*
- *Further investigations of shape parameter and kernel selection are reported in [MF14, Mon11].*
- *Related criteria ("sequential CV", "geometry weighted CV" and "partial MLE") are derived in [Sch11].*



References I

[All74] D. M. Allen, *The relationship between variable selection and data agumentation and a method for prediction*, Technometrics **16** (1974), no. 1, 125–127.

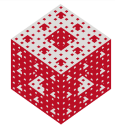
[CW79] Peter Craven and Grace Wahba, *Smoothing noisy data with spline functions*, Numerische Mathematik **31** (1979), no. 4, 377–403.

[Fas02] G. E. Fasshauer, *Newton iteration with multiquadrics for the solution of nonlinear PDEs*, Computers & Mathematics with Applications **43** (2002), no. 3–5, 423–438.

[Fas07] _____, *Meshfree Approximation Methods with* MATLAB, Interdisciplinary Mathematical Sciences, vol. 6, World Scientific Publishing Co., Singapore, 2007.

[Fas08] _____, *Tutorial on Meshfree Approximation Methods with Matlab*, Dolomites Research Notes On Approximation **1** (2008).

[Fra82] Richard Franke, *Scattered data interpolation: Tests of some method*, Mathematics of Computation **38** (1982), no. 157, 181–200.



References II

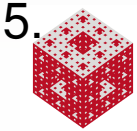
[FZ07] Gregory E. Fasshauer and Jack G. Zhang, *On choosing "optimal" shape parameters for RBF approximation*, Numerical Algorithms **45** (2007), no. 1-4, 345–368.

[GCK96] M. A. Golberg, C. S. Chen, and S. R. Karur, *Improved multiquadric approximation for partial differential equations*, Eng. Anal. with Bound. Elem. **18** (1996), 9–17.

[GHW79] G. H. Golub, M. Heath, and G. Wahba, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics **21** (1979), no. 2, 215–223.

[GW59] M. Golomb and H. F. Weinberger, *Optimal approximation and error bounds*, On Numerical Approximation (R. E. Langer, ed.), University of Wisconsin Press, 1959, pp. 117–190.

[Har71] Rolland L. Hardy, *Multiquadric equations of topography and other irregular surfaces*, Journal of Geophysical Research **76** (1971), no. 8, 1905–1915.



References III

[HH99] Fred J. Hickernell and Y. C. Hon, *Radial basis function approximations as smoothing splines*, Applied Mathematics and Computation **102** (1999), no. 1, 1–24.

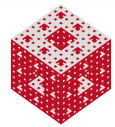
[Hic09] F. J. Hickernell, *Shape parameter problem*, 2009.

[MF14] Michael McCourt and G. E. Fasshauer, *Stable likelihood computation for Gaussian random fields*, 2014.

[Mon11] M. Mongillo, *Choosing basis functions and shape parameters for radial basis function methods*, SIAM Undergraduate Research Online **4** (2011), 190–209.

[Neu98] Arnold Neumaier, *Solving ill-conditioned and singular linear systems: A tutorial on regularization*, SIAM Review **40** (1998), no. 3, 636–666.

[Ril55] James D. Riley, *Solving systems of linear equations with a positive definite, symmetric, but possibly ill-conditioned matrix*, Mathematical Tables and Other Aids to Computation **9** (1955), no. 51, 96–101.



References IV

[Rip99] S. Rippa, *An algorithm for selecting a good value for the parameter c in radial basis function interpolation*, Adv. Comput. Math. **11** (1999), no. 2-3, 193–210.

[Sch11] Michael Scheuerer, *An alternative procedure for selecting a good value for the parameter c in RBF-interpolation*, Advances in Computational Mathematics **34** (2011), no. 1, 105–126.

[Ste90] Michael L. Stein, *A comparison of generalized cross validation and modified maximum likelihood for estimating the parameters of a stochastic process*, The Annals of Statistics **18** (1990), no. 3, 1139–1157.

[Ste99] M. L. Stein, *Interpolation of Spatial Data: Some theory for Kriging*, Springer-Verlag, New York, 1999.

[Sto77] M. Stone, *Asymptotics for and against cross-validation*, Biometrika **64** (1977), no. 1, 29–35.

[Wah90] Grace Wahba, *Spline Models for Observational Data*, SIAM, Philadelphia, 1990.