Programming Derivatives of RBFs

Robert Schaback

Georg-August-Universität Göttingen
Akademie der Wissenschaften zu Göttingen

ICERM August 2017

Overview

Motivation

Examples

Theory

Remarks on Implementation

Summary and Outlook

Motivation

Motivation: Need for Derivatives

Motivation: Need for Derivatives

For unsymmetric collocation you have to take Δ

Motivation: Need for Derivatives

For unsymmetric collocation you have to take Δ

For symmetric collocation you have to take Δ and Δ^2

Motivation: Need for Derivatives

For unsymmetric collocation you have to take Δ

For symmetric collocation you have to take Δ and Δ^2

For divergence-free vector fields derived from kernels K you need $(\nabla\nabla^T - \Delta \cdot Id)K$

Motivation: Need for Derivatives

For unsymmetric collocation you have to take Δ

For symmetric collocation you have to take Δ and Δ^2

For divergence-free vector fields derived from kernels K you need $(\nabla\nabla^T - \Delta \cdot Id)K$

Students never get derivatives right

Idea: Recursion

Idea: Recursion

Observation: Derivatives of RBFs often are (modified) RBFs

Idea: Recursion

Observation: Derivatives of RBFs often are (modified) RBFs

Assume RBF family $\{\phi_p(r)\}_p$ parametrized by p

Idea: Recursion

Observation: Derivatives of RBFs often are (modified) RBFs

Assume RBF family $\{\phi_p(r)\}_p$ parametrized by p

Express derivatives via $\phi_p(r),\ \phi_{p-1}(r)$ etc.

Idea: Recursion

Observation: Derivatives of RBFs often are (modified) RBFs

Assume RBF family $\{\phi_p(r)\}_p$ parametrized by p

Express derivatives via $\phi_p(r),\ \phi_{p-1}(r)$ etc.

Observation: Strange pattern of derivative recursions

Idea: Recursion

Observation: Derivatives of RBFs often are (modified) RBFs

Assume RBF family $\{\phi_p(r)\}_p$ parametrized by p

Express derivatives via $\phi_p(r),\ \phi_{p-1}(r)$ etc.

Observation: Strange pattern of derivative recursions

Observation: The pattern comes from the *f*-form of RBFs

Idea: Recursion on f-form

Idea: Recursion on f-form

Write $\phi_p(r) = f_p(r^2/2)$ or $\phi_p(\sqrt{2s}) = f_p(s),\ s = r^2/2$

Idea: Recursion on f-form

Write $\phi_p(r) = f_p(r^2/2)$ or $\phi_p(\sqrt{2s}) = f_p(s),\ s = r^2/2$

Well-known from Bocher-Schoenberg theory

Idea: Recursion on f-form

Write $\phi_p(r) = f_p(r^2/2)$ or $\phi_p(\sqrt{2s}) = f_p(s),\ s = r^2/2$

Well-known from Bocher-Schoenberg theory

Goal: Express f_p derivatives via $f_{p-1},\ f_{p-2}$ etc.

Examples

Example: Gaussian

$$\phi(r) = \exp(-r^2/2)$$

Example: Gaussian

$$\phi(r) = \exp(-r^2/2)$$

$$f(s) = \exp(-s)$$

Example: Gaussian

$$\phi(r) = \exp(-r^2/2)$$

$$f(s) = \exp(-s)$$

$$f'(s) = -f(s)$$

Example: Multiquadrics

$$\phi_m(r) = (1 + r^2/2)^{-m}$$

Example: Multiquadrics

$$\phi_m(r) = (1 + r^2/2)^{-m}$$

$$f_m(s) = (1 + s)^{-m}$$

Example: Multiquadrics

$$\phi_m(r) = (1 + r^2/2)^{-m}$$

$$f_m(s) = (1 + s)^{-m}$$

$$f'_m(s) = -m\, f_{m+1}(s)$$

Example: Powers

$$\phi_m(r) = r^m$$

Example: Powers

$$\phi_m(r) = r^m$$

$$f_m(s) = (\sqrt{2s})^m$$

Example: Powers

$$\phi_m(r) = r^m$$

$$f_m(s) = (\sqrt{2s})^m$$

$$\frac{d}{ds}\sqrt{2s} = 1/\sqrt{2s}$$

Example: Powers

$$\phi_m(r) = r^m$$

$$f_m(s) = (\sqrt{2s})^m$$

$$\frac{d}{ds}\sqrt{2s} = 1/\sqrt{2s}$$

$$f'_m(s) = m\,(\sqrt{2s})^{m-1}/\sqrt{2s} = m\,f_{m-2}(s)$$

Example: Thin-Plate Splines

$$\phi_{2m}(r) = r^{2m} \log r$$

Example: Thin-Plate Splines

$$\phi_{2m}(r) = r^{2m} \log r$$

$$f_{2m}(s) = (\sqrt{2s})^{2m} \log(\sqrt{2s})$$

Example: Thin-Plate Splines

$$\phi_{2m}(r) = r^{2m} \log r$$

$$f_{2m}(s) = (\sqrt{2s})^{2m} \log(\sqrt{2s})$$

$$\begin{aligned} f'_{2m}(s) &= 2m(\sqrt{2s})^{2m-1} \log(\sqrt{2s})/\sqrt{2s} + (\sqrt{2s})^{2m}/(2s) \\ &= 2m\, f_{2m-2}(s) + \underbrace{(2s)^{m-1}}_{= \text{polynomial}} \end{aligned}$$

Example: Thin-Plate Splines

$$\phi_{2m}(r) = r^{2m} \log r$$

$$f_{2m}(s) = (\sqrt{2s})^{2m} \log(\sqrt{2s})$$

$$\begin{aligned} f'_{2m}(s) &= 2m(\sqrt{2s})^{2m-1} \log(\sqrt{2s})/\sqrt{2s} + (\sqrt{2s})^{2m}/(2s) \\ &= 2m\, f_{2m-2}(s) + \underbrace{(2s)^{m-1}}_{=\ \text{polynomial}} \end{aligned}$$

The polynomial part vanishes
in the conditional positive definite setting

Example: Thin-Plate Splines

$$\phi_{2m}(r) = r^{2m} \log r$$

$$f_{2m}(s) = (\sqrt{2s})^{2m} \log(\sqrt{2s})$$

$$\begin{aligned} f'_{2m}(s) &= 2m(\sqrt{2s})^{2m-1} \log(\sqrt{2s})/\sqrt{2s} + (\sqrt{2s})^{2m}/(2s) \\ &= 2m\, f_{2m-2}(s) + \underbrace{(2s)^{m-1}}_{=\text{ polynomial}} \end{aligned}$$

The polynomial part vanishes
in the conditional positive definite setting

Dealing with powers is clear

Matérn-Sobolev Kernels

$$\phi_\nu(r) = r^\nu K_\nu(r)$$

Matérn-Sobolev Kernels

$$\phi_\nu(r) = r^\nu K_\nu(r)$$

$$f_\nu(s) = (\sqrt{2s})^\nu K_\nu(\sqrt{2s})$$

Matérn-Sobolev Kernels

$$\phi_\nu(r) = r^\nu K_\nu(r)$$

$$f_\nu(s) = (\sqrt{2s})^\nu K_\nu(\sqrt{2s})$$

$$\frac{d}{dz}(z^\nu K_\nu(z)) = -z^\nu K_{\nu-1}(z)$$

Matérn-Sobolev Kernels

$$\phi_\nu(r) = r^\nu K_\nu(r)$$

$$f_\nu(s) = (\sqrt{2s})^\nu K_\nu(\sqrt{2s})$$

$$\frac{d}{dz}(z^\nu K_\nu(z)) = -z^\nu K_{\nu-1}(z)$$

$$f'_\nu(s) = -(\sqrt{2s})^\nu K_{\nu-1}(\sqrt{2s})/\sqrt{2s} = -f_{\nu-1}(s)$$

Matérn-Sobolev Kernels

$$\phi_\nu(r) = r^\nu K_\nu(r)$$

$$f_\nu(s) = (\sqrt{2s})^\nu K_\nu(\sqrt{2s})$$

$$\frac{d}{dz}(z^\nu K_\nu(z)) = -z^\nu K_{\nu-1}(z)$$

$$f'_\nu(s) = -(\sqrt{2s})^\nu K_{\nu-1}(\sqrt{2s})/\sqrt{2s} = -f_{\nu-1}(s)$$

This would not work without $s = r^2/2$

Matérn-Sobolev Kernels

$$\phi_\nu(r) = r^\nu K_\nu(r)$$

$$f_\nu(s) = (\sqrt{2s})^\nu K_\nu(\sqrt{2s})$$

$$\frac{d}{dz}(z^\nu K_\nu(z)) = -z^\nu K_{\nu-1}(z)$$

$$f'_\nu(s) = -(\sqrt{2s})^\nu K_{\nu-1}(\sqrt{2s})/\sqrt{2s} = -f_{\nu-1}(s)$$

This would not work without $s = r^2/2$

$\nu = m - d/2 \Rightarrow \nu - 1$
means $m \Rightarrow m - 1$ or $d \Rightarrow d + 2$

Wendland Kernels

$\phi_{d,k}$ in C^{2k}, SPD on $\mathbb{R}^d$, minimal degree $\lfloor d/2 \rfloor + 3k + 1$

$$
\begin{aligned}
\phi_\ell(r) &:= (1-r)_+^\ell \\
(I\phi)(r) &:= \int_r^\infty t\phi(t)dt \\
\phi_{d,k}(r) &:= I^k \phi_{\lfloor d/2 \rfloor + k + 1}(r) \\
f_{d,k}(s) &:= I^k \phi_{\lfloor d/2 \rfloor + k + 1}(\sqrt{2s}) \\
(I\phi)'(r) &= -r\phi(r) \\
f'_{d,k}(s) &= -\sqrt{2s}\, I^{k-1} \phi_{\lfloor d/2 \rfloor + k + 1}(\sqrt{2s})/\sqrt{2s} \\
&= -I^{k-1} \phi_{\lfloor (d+2)/2 \rfloor + k - 1 + 1}(\sqrt{2s}) \\
&= -f_{d+2,k-1}(s)
\end{aligned}
$$

This would not work without $s = r^2/2$

Laplacian

$$
\begin{aligned}
\Delta\phi(r) &= \phi''(r) + (d-1)\frac{\phi'(r)}{r} \text{ (singular!)} \\
\phi(r) &= f(r^2/2) \\
\phi'(r) &= r\, f'(r^2/2) \\
\phi''(r) &= r^2\, f''(r^2/2) + f'(r^2/2) \\
\Delta\phi &= r^2\, f''(r^2/2) + d\, f'(r^2/2) = 2sf''(s) + df'(s) \\
\Delta^2\phi &= 4s^2 f^{(4)}(s) + 4sdf^{(3)}(s) + d^2 f''(s)
\end{aligned}
$$

No visible singularities in f-form
Other derivatives via e.g.

$$\frac{d}{dx}\phi(r) = \phi'(r)\frac{x}{r} = r\, f'(r^2/2)\frac{x}{r} = xf'(s)$$

Theory

General Result

Theorem (Dimension walk)

Radial Fourier transform F_d on $\mathbb{R}^d$ implies $F_{d+2} f_p' = -F_d f_p$

General Result

Theorem (Dimension walk)
Radial Fourier transform F_d on $\mathbb{R}^d$ implies $F_{d+2} f_p' = -F_d f_p$
Closedness Assumption between $\{f_p\}_p$ and $\{g_q\}_q$

$$F_d f_p = g_{A(d,p)}, \quad F_d g_q = f_{B(d,q)}$$

General Result

Theorem (Dimension walk)
Radial Fourier transform F_d on $\mathbb{R}^d$ implies $F_{d+2} f_p' = -F_d f_p$
Closedness Assumption between $\{f_p\}_p$ and $\{g_q\}_q$

$$F_d f_p = g_{A(d,p)}, \quad F_d g_q = f_{B(d,q)}$$

Theorem:

$$f_p' = -F_{d+2} F_d f_p = -f_{B(d+2,A(d,p))}$$

General Result

Theorem (Dimension walk)

Radial Fourier transform F_d on $\mathbb{R}^d$ implies $F_{d+2} f_p' = -F_d f_p$

Closedness Assumption between $\{f_p\}_p$ and $\{g_q\}_q$

$$F_d f_p = g_{A(d,p)}, \quad F_d g_q = f_{B(d,q)}$$

Theorem:

$$f_p' = -F_{d+2} F_d f_p = -f_{B(d+2,A(d,p))}$$

No separate derivative program needed

General Result

Theorem (Dimension walk)

Radial Fourier transform F_d on $\mathbb{R}^d$ implies $F_{d+2}f_p' = -F_d f_p$

Closedness Assumption between $\{f_p\}_p$ and $\{g_q\}_q$

$$F_d f_p = g_{A(d,p)}, \quad F_d g_q = f_{B(d,q)}$$

Theorem:

$$f_p' = -F_{d+2}F_d f_p = -f_{B(d+2,A(d,p))}$$

No separate derivative program needed

Derivatives and dimensions may be fractional

Proof of Dimension Walk

Radial Fourier transform F_ν for $\nu = (d-2)/2$:

$$
\begin{aligned}
(F_\nu f_p)(t) &= \int_0^\infty f_p(s) s^\nu H_\nu(st) ds \\
f_p(s) &= \int_0^\infty (F_\nu f_p)(t) t^\nu H_\nu(ts) dt \\
(z/2)^{-\nu} J_\nu(z) &= H_\nu(-z^2/4) = \textstyle\sum_{k=0}^\infty \frac{(-z^2/4)^k}{k!\Gamma(k+\nu+1)} \\
H_\nu' &= -H_{\nu+1}, \qquad d \Rightarrow d+2 \\
f_p'(s) &= \int_0^\infty (F_\nu f_p)(t) t^\nu t H_\nu'(ts) dt \\
&= -\int_0^\infty (F_\nu f_p)(t) t^{\nu+1} H_{\nu+1}'(ts) dt \\
&= -F_{\nu+1}^{-1} F_\nu(f_p)(s) \\
F_{\nu+1} f_p' &= -F_\nu f_p
\end{aligned}
$$

Remarks on Implementation

Matrix Formulation

Kernel matrix $\phi(\|x_j - y_k\|_2) = f(\|x_j - y_k\|_2^2/2)$

```
function dsqh=distsqh(X, Y)
% X and Y are matrices with points as rows
nX=length(X(:,1));nY=length(Y(:,1));
Xsh=sum((X.*X)')/2; Ysh=sum((Y.*Y)')/2;
dsqh=repmat(Xsh',1,nY)+repmat(Ysh,nX,1)-X*Y';
```

Matrix Formulation

Kernel matrix $\phi(\|x_j - y_k\|_2) = f(\|x_j - y_k\|_2^2/2)$
Write f as program on a matrix $S = (\|x_j - y_k\|_2^2/2)$

```
function dsqh=distsqh(X, Y)
% X and Y are matrices with points as rows
nX=length(X(:,1));nY=length(Y(:,1));
Xsh=sum((X.*X)')/2; Ysh=sum((Y.*Y)')/2;
dsqh=repmat(Xsh',1,nY)+repmat(Ysh,nX,1)-X*Y';
```

Matrix Formulation

Kernel matrix $\phi(\|x_j - y_k\|_2) = f(\|x_j - y_k\|_2^2/2)$

Write f as program on a matrix $S = (\|x_j - y_k\|_2^2/2)$

Use $\|x_j - y_k\|_2^2/2 = \|x_j\|_2^2/2 + \|y_k\|^2/2 - (x_j, y_k)$

```
function dsqh=distsqh(X, Y)
% X and Y are matrices with points as rows
nX=length(X(:,1));nY=length(Y(:,1));
Xsh=sum((X.*X)')/2; Ysh=sum((Y.*Y)')/2;
dsqh=repmat(Xsh',1,nY)+repmat(Ysh,nX,1)-X*Y';
```

Matrix Formulation

Kernel matrix $\phi(\|x_j - y_k\|_2) = f(\|x_j - y_k\|_2^2/2)$

Write f as program on a matrix $S = (\|x_j - y_k\|_2^2/2)$

Use $\|x_j - y_k\|_2^2/2 = \|x_j\|_2^2/2 + \|y_k\|^2/2 - (x_j, y_k)$

No square roots, no loops for this

```
function dsqh=distsqh(X, Y)
% X and Y are matrices with points as rows
nX=length(X(:,1));nY=length(Y(:,1));
Xsh=sum((X.*X)')/2; Ysh=sum((Y.*Y)')/2;
dsqh=repmat(Xsh',1,nY)+repmat(Ysh,nX,1)-X*Y';
```

Calculation of *f*-Form

```
S=distsqh(X,Y);
```

Calculation of *f*-Form

`S=distsqh(X,Y);`

`F=frbf(S,k)` calculates $f^k(S)$ on matrix S

Calculation of *f*-Form

`S=distsqh(X,Y);`

`F=frbf(S,k)` calculates $f^k(S)$ on matrix S

RBF type, scale, parameters controlled by globals

Calculation of *f*-Form

`S=distsqh(X,Y);`

`F=frbf(S,k)` calculates $f^k(S)$ on matrix S

RBF type, scale, parameters controlled by globals

E.g.: Laplacian is `d*frbf(S,1)+2*S.*frbf(S,2)`

Summary and Outlook

Summary

You don't need to program derivatives,
if you program a whole family of RBFs
that is closed under double Fourier transforms
wrt. different dimensions

Summary

You don't need to program derivatives,
if you program a whole family of RBFs
that is closed under double Fourier transforms
wrt. different dimensions

Available as Technical Report:
http://num.math.uni-goettingen.de/schaback/
research/papers/MPfKBM.pdf

Summary

You don't need to program derivatives,
if you program a whole family of RBFs
that is closed under double Fourier transforms
wrt. different dimensions

Available as Technical Report:
http://num.math.uni-goettingen.de/schaback/research/papers/MPfKBM.pdf

Version of 2011, dating back to 2008

Caveat

For low order Wendland functions:

Caveat

For low order Wendland functions:

numerical problems at zero

Caveat

For low order Wendland functions:

numerical problems at zero

Calculating Δ needs $f'_{d,k} = -f_{d+2,k-1},\ f''_{d,k} = f_{d+4,k-2}$

Caveat

For low order Wendland functions:

numerical problems at zero

Calculating Δ needs $f'_{d,k} = -f_{d+2,k-1},\ f''_{d,k} = f_{d+4,k-2}$

For $k = 1$ the function $f_{d,k}$ is in $C^2 = C^{2k}$, but calculation goes down to $f_{d+4,-1}$

Caveat

For low order Wendland functions:

numerical problems at zero

Calculating Δ needs $f'_{d,k} = -f_{d+2,k-1},\ f''_{d,k} = f_{d+4,k-2}$

For $k = 1$ the function $f_{d,k}$ is in $C^2 = C^{2k}$, but calculation goes down to $f_{d+4,-1}$

Example: $d = 2,\ k = 1$

$$\phi_{6,-1}(r) = I^{-1}\phi_3(r) = -\frac{1}{r}\frac{d}{dr}(1-r)_+^3 = \frac{3(1-r)_+^2}{r}$$

Caveat

For low order Wendland functions:

numerical problems at zero

Calculating Δ needs $f'_{d,k} = -f_{d+2,k-1},\ f''_{d,k} = f_{d+4,k-2}$

For $k = 1$ the function $f_{d,k}$ is in $C^2 = C^{2k}$, but calculation goes down to $f_{d+4,-1}$

Example: $d = 2,\ k = 1$

$$\phi_{6,-1}(r) = I^{-1}\phi_3(r) = -\frac{1}{r}\frac{d}{dr}(1-r)_+^3 = \frac{3(1-r)_+^2}{r}$$

Theory is OK: Cancellation of singularities at zero

Caveat

For low order Wendland functions:

numerical problems at zero

Calculating Δ needs $f'_{d,k} = -f_{d+2,k-1}, \ f''_{d,k} = f_{d+4,k-2}$

For $k = 1$ the function $f_{d,k}$ is in $C^2 = C^{2k}$, but calculation goes down to $f_{d+4,-1}$

Example: $d = 2, \ k = 1$

$$\phi_{6,-1}(r) = I^{-1}\phi_3(r) = -\frac{1}{r}\frac{d}{dr}(1-r)_+^3 = \frac{3(1-r)_+^2}{r}$$

Theory is OK: Cancellation of singularities at zero

Laplacian needs $f''_{2,1} = r^2\phi_{6,-1}(r)$

Open Problems

Deal with Wendland case properly

Open Problems

Deal with Wendland case properly

Make routines more efficient,
e.g. Laplacian via `d*frbf(S,1)+2*S.*frbf(S,2)`

Open Problems

Deal with Wendland case properly

Make routines more efficient,
e.g. Laplacian via `d*frbf(S,1)+2*S.*frbf(S,2)`

Dear with sparsity properly

Open Problems

- Deal with Wendland case properly
- Make routines more efficient,
  e.g. Laplacian via `d*frbf(S,1)+2*S.*frbf(S,2)`
- Dear with sparsity properly
- Implement basis transformations

Open Problems

- Deal with Wendland case properly
- Make routines more efficient,
  e.g. Laplacian via `d*frbf(S,1)+2*S.*frbf(S,2)`
- Dear with sparsity properly
- Implement basis transformations
- Extend to a general toolkit

Thank You!

For references, see

http://www.num.math.uni-goettingen.de/schaback/research.html