S2Fun

Felix Bartel and Ralf Hielscher

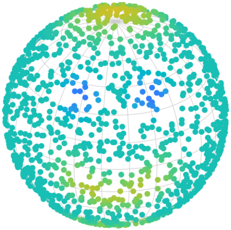
TU Chemnitz

28 Feb 2018

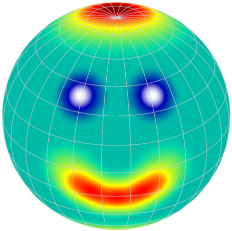
Table of contents

- get a function which approximates our data

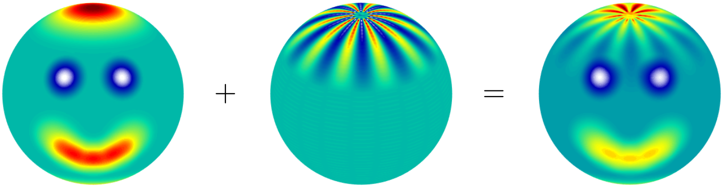
given: scattered data on $\mathbb{S}^2$



wanted: function $f : \mathbb{S}^2 \to \mathbb{R}$



- modifying and calculating with functions on $\mathbb{S}^2$



- generalize our concept to $f : \mathbb{S}^2 \to \mathbb{R}^n$, functions with symmetries, vector fields and axis fields

Using S2Fun

Defining a S2Fun via approximation

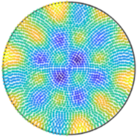
```
nodes = equispacedS2Grid('resolution',3*degree,'antipodal');
nodes = nodes(:); % define some vertices
y = smiley(nodes); % define function values
plot(nodes,y); % plot the discrete data
```

```
sF1 = interp(nodes,y,'harmonicApproximation');
plot(sF1); % plot the spherical function
```

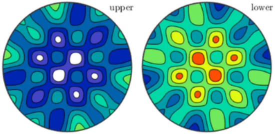
Using S2Fun

Defining a S2Fun via quadrature

```
f = @(v) 0.1*(v.theta+sin(8*v.x).*sin(8*v.y));
plot(nodes,f(nodes)); % plot the function at discrete points
```



```
sF2 = S2FunHarmonic.quadrature(f,'bandwidth',150);
plot(sF2); % plot the spherical function
```



Using S2Fun

Defining a S2Fun via the Fourier coefficients

```
fhat = rand(25,1);
sF3 = S2FunHarmonic(fhat)
```

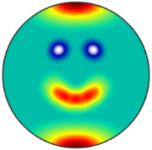
- the Fourier coefficients are specified as a column vector

Using S2Fun
Plotting a S2Fun

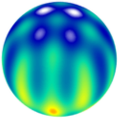
```
plot(sF1);
```

```
contour(sF1);
```

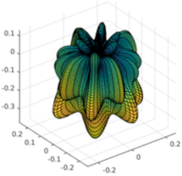
```
pcolor(sF1);
```



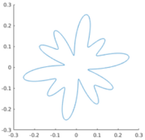
```
plot3d(sF2);
```



```
surf(sF2);
```




```
plotSection(sF2,
    zvector);
```



addition/subtraction

```
sF1+sF2; sF1-sF2;
sF1+2; sF2-4;
```

multiplication/division

```
sF1.*sF2; 2.*sF1;
sF1./(sF2+1); 2./sF2; sF2./4;
```

power

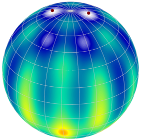
```
sF1.^sF2; 2.^sF1; sF2.^4;
```

absolute value

```
abs(sF1);
```

global minimum

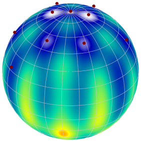
```
[val,nodes] = min(sF2);
plot3d(sF2);
scatter3d(nodes);
```



- `val` has one value

local minimas

```
[val,nodes] = min(sF2,'
    numLocal',10);
scatter3d(nodes);
```



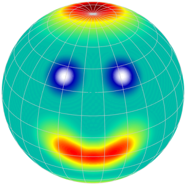
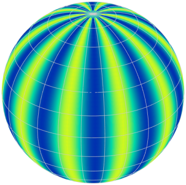
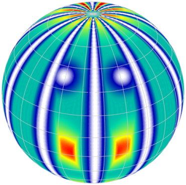
- `val` has `length(nodes)` values

(same conventions hold for the `max` command)

Using S2Fun

Calculating with S2Fun - min/max

minimum in the pointwise sense

$\min\Bigg($  ,  $\Bigg) =$ 

norm

```
norm(sF1);
```

- $L^2(\mathbb{S}^2)$ norm

mean

```
mean(sF1);
```

- mean value on $\mathbb{S}^2$

sum

```
sum(sF1);
```

- integral over the whole sphere

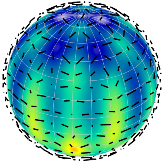
rotate

```
r=rotation('Euler',[pi/4 0 0])
rotate(sF1,r);
```

- rotates the function by `r`

calculating the gradient as `S2VectorFieldHarmonic`

```
G = grad(sF1);
```



calculating the gradient directly

```
nodes = vector3d.rand(100);
grad(sF1,nodes);
```

- is faster because it will not be convertet to `S2VectorFieldHarmonic`

$$\mathrm{sF} = \begin{pmatrix} \mathrm{sF1} & \mathrm{sF4} \\ \mathrm{sF2} & \mathrm{sF5} \\ \mathrm{sF3} & \mathrm{sF6} \end{pmatrix}$$

for a 3×2 matrix of `S2FunHarmonic` we need the following function evaluations

$$F(:,:,1) = \begin{pmatrix} f_1(v_1) & f_2(v_1) & f_3(v_1) \\ f_1(v_2) & f_2(v_2) & f_3(v_2) \\ f_1(v_3) & f_2(v_3) & f_3(v_3) \\ \vdots & \vdots & \vdots \end{pmatrix}, F(:,:,2) = \begin{pmatrix} f_4(v_1) & f_5(v_1) & f_6(v_1) \\ f_4(v_2) & f_5(v_2) & f_6(v_2) \\ f_4(v_3) & f_5(v_3) & f_6(v_3) \\ \vdots & \vdots & \vdots \end{pmatrix}$$

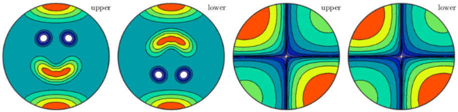
the Fourier coefficients are stored like

$$\hat{F}(:,:,1) = \begin{pmatrix} \hat{\mathbf{f}}_1 & \hat{\mathbf{f}}_2 & \hat{\mathbf{f}}_3 \end{pmatrix} \quad \text{and} \quad \hat{F}(:,:,2) = \begin{pmatrix} \hat{\mathbf{f}}_4 & \hat{\mathbf{f}}_5 & \hat{\mathbf{f}}_6 \end{pmatrix}$$

```
nodes = equispacedS2Grid('resolution',3*degree,'antipodal');
nodes = nodes(:); % define some vertices
y = [smiley(nodes),(nodes.x.*nodes.y).^(1/4)];
```

now the actual definition and plotting

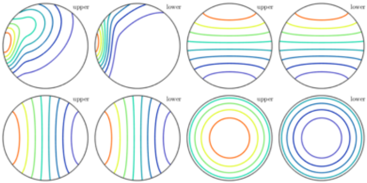
```
sF1 = S2FunHarmonic.approximation(nodes,y)
plot(sF1); % plot the spherical function
```



```
f = @(v) [exp(v.x+v.y+v.z)+50*(v.y-cos(pi/3)).^3.*(v.y-cos(pi
    /3) > 0), v.x, v.y, v.z];
```

now the actual definition and plotting

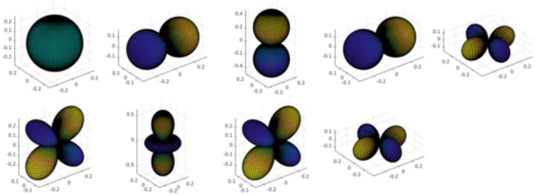
```
sF2 = S2FunHarmonic.quadrature(f,'bandwidth',50)
contour(sF2); % plot the spherical function
```



Using S2Fun

Multivariate S2Fun - definition via Fourier coefficients

```
sF3 = S2FunHarmonic(eye(9));
surf(sF3);
```

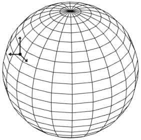


```
sF4 = [sF1; sF2]; % concatenation
sF4(2:3); % indexing
conj(sF1); % complex conjugate
sF1.'; % transpose
sF1'; % complex conjugate transpose
length(sF1); % number of functions in sF1
size(sF2); % length of each dimensions
sF3 = reshape(sF3,3,[]); % reshapes the size
sum(sF1); % integral for each element
sum(sF3,2); % sum over the second dimension
min(sF3); % pointwise minimum over the first non singelton
    dimension
```

definition via multivariate `S2FunHarmonic`

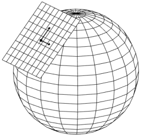
x-, y- and z-component

```
sF=S2FunHarmonic(rand(10,3));
sVF1=S2VectorFieldHarmonic(sF)
```



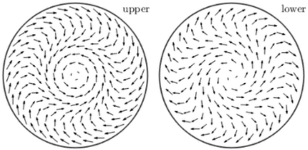
ϑ- and ρ-component

```
sF=S2FunHarmonic(rand(10,2));
sVF2=S2VectorFieldHarmonic(sF)
```



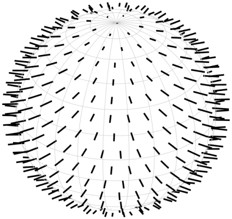
definition via approximation

```
nodes = equispacedS2Grid('points',1e5);
nodes = nodes(:);
y = vector3d('polar',sin(3*nodes.theta),nodes.rho+pi/2);
sVF1 = S2VectorFieldHarmonic.approximation(nodes,y)
plot(sVF1);
```



definition via quadrature

```
f = @(v) vector3d(v.x,v.y,0*v.x);
sVF2 = S2VectorFieldHarmonic.quadrature(@(v) f(v))
quiver3(sVF2);
```



addition/substraction

```
sVF1+sVF2;
sVF1+vector3d(1,0,0);
sVF1-sVF2;
sVF2-vector3d(1/2,1/2,0);
```

multiplication/division

```
2.*sVF1;
sVF1./4;
```

dot product

```
dot(sVF1,sVF2);
dot(sVF1,vector3d(0,0,1));
```

cross product

```
cross(sVF1,sVF2);
cross(sVF1,vector3d(0,0,1));
```

mean

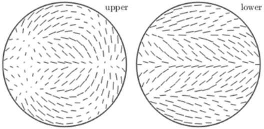
```
mean(sVF1);
```

norm

```
norm(sVF1);
```

Using S2Fun
S2AxisField

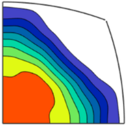
```
nodes = equispacedS2Grid('points',1e5);
nodes = nodes(:);
y = vector3d(sin(5*nodes.x),1,nodes.y,'antipodal');
sAF1 = S2AxisFieldHarmonic.approximation(nodes,y);
plot(sAF1);
```



- operations and other definition methods similar to `S2VectorFieldHarmonic`

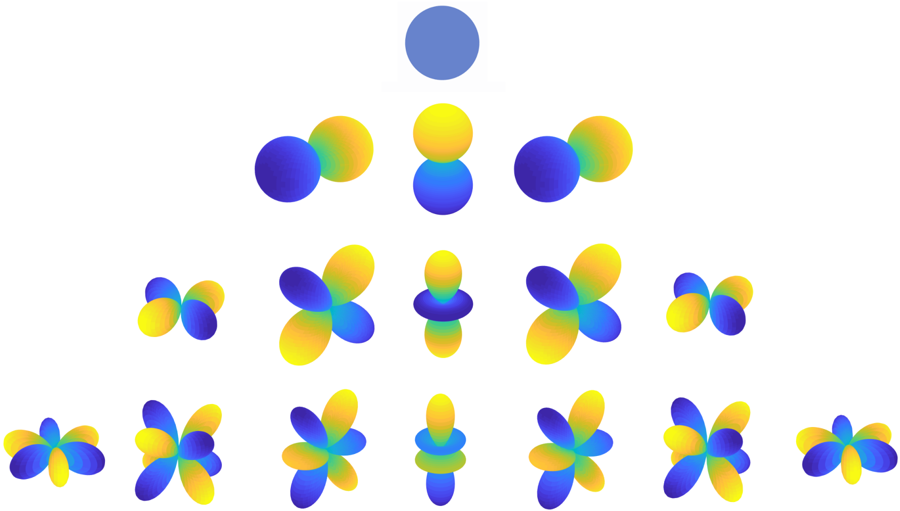
```
sF = S2FunHarmonic.quadrature(@(v) smiley(v))
cs = crystalSymmetry('432');

sFs = symmetrise(sF,cs);
plot(sFs);
```



- this symmetrises the function and gives back the result with the symmetry attached
- the operations for `S2FunSym` are equivalent to these for `S2Fun`

Underlying mathematics
Spherical harmonics



- spherical harmonic of degree m and order l

$$Y_{m,l}(\vartheta, \rho) = \sqrt{\frac{2m+1}{4\pi}} P_{m,|l|}(\cos \rho) \mathrm{e}^{\mathrm{i} l \vartheta}$$

- $\{Y_{m,l}\}_{m \in \mathbb{N}_0, |l| \leq m}$ form a complete orthonormalbasis of $L^2(\mathbb{S}^2)$
- we define space of spherical harmonics upto degree M

$$\mathbb{P}_M\left(\mathbb{S}^2\right) := \operatorname{span}\{Y_{m,l}\}_{m=0,\ldots,M,|l| \leq m}$$

series expansion for $f \in L^2(\mathbb{S}^2)$:

$$f(\boldsymbol{\xi}) = \sum_{m=0}^{\infty} \sum_{l=-m}^{m} \hat{f}_{m,l} Y_{m,l}(\boldsymbol{\xi})$$

with Fourier coefficients

$$\begin{aligned} \hat{f}_{m,l} &= (f, Y_{m,l})_{L^2(\mathbb{S}^2)} \\ &= \int_{\mathbb{S}^2} f(\boldsymbol{\xi}) \overline{Y_{m,l}(\boldsymbol{\xi})} \, \mathrm{d}\omega(\boldsymbol{\xi}) \end{aligned}$$

series expansion for $f \in L^2(\mathbb{S}^2)$:

$$f(\boldsymbol{\xi}) = \sum_{m=0}^{\infty} \sum_{l=-m}^{m} \hat{f}_{m,l} Y_{m,l}(\boldsymbol{\xi})$$

with Fourier coefficients

$$\begin{aligned} \hat{f}_{m,l} &= (f, Y_{m,l})_{L^2(\mathbb{S}^2)} \\ &= \int_{\mathbb{S}^2} f(\boldsymbol{\xi}) \overline{Y_{m,l}(\boldsymbol{\xi})} \, \mathrm{d}\omega(\boldsymbol{\xi}) \end{aligned}$$

approximation through $f \in C(\mathbb{S}^2)$:

$$f(\boldsymbol{\xi}) = \sum_{m=0}^{M} \sum_{l=-m}^{m} \tilde{f}_{m,l} Y_{m,l}(\boldsymbol{\xi})$$

with a suitable choice of M and a sheme for calculating $\tilde{f}_{m,l}$

frequency domain

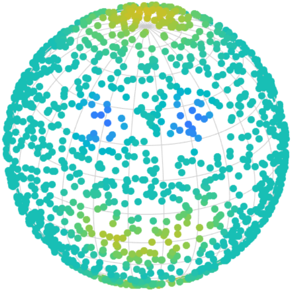
Fourier coefficients

$$\underbrace{\begin{array}{ccccc} & & & & \hat{f}_{M,-M} \\ & & & \iddots & \\ & \hat{f}_{1,-1} & & & \\ \hat{f}_{0,0} & \hat{f}_{1,0} & \cdots & & \vdots \\ & \hat{f}_{1,1} & & & \\ & & & \ddots & \\ & & & & \hat{f}_{M,M} \end{array}}_{(M+1)^2}$$

time domain

point evaluations

$$\underbrace{f(\xi_1), \ldots, f(\xi_N)}_{N}$$



Underlying mathematics
Spherical harmonics

frequency domain

Fourier coefficients

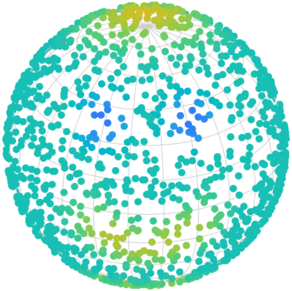
$$\underbrace{\begin{array}{ccccc} & & & & \hat{f}_{M,-M} \\ & & & \iddots & \\ & \hat{f}_{1,-1} & & & \\ \hat{f}_{0,0} & \hat{f}_{1,0} & & \cdots & \vdots \\ & \hat{f}_{1,1} & & & \\ & & & \ddots & \\ & & & & \hat{f}_{M,M} \end{array}}_{(M+1)^2}$$

$$\xrightarrow[\text{NFSFT}]{\boldsymbol{Y}_{\mathcal{X},M}}$$

time domain

point evaluations

$$\underbrace{f(\xi_1), \ldots, f(\xi_N)}_{N}$$



Underlying mathematics

Spherical harmonics

frequency domain

Fourier coefficients

$$\underbrace{\begin{array}{ccccc} & & & & \hat{f}_{M,-M} \\ & & & \iddots & \\ & \hat{f}_{1,-1} & & & \\ \hat{f}_{0,0} & \hat{f}_{1,0} & \cdots & & \vdots \\ & \hat{f}_{1,1} & & & \\ & & & \ddots & \\ & & & & \hat{f}_{M,M} \end{array}}_{(M+1)^2}$$

$\boldsymbol{Y}_{\mathcal{X},M}$

NFSFT

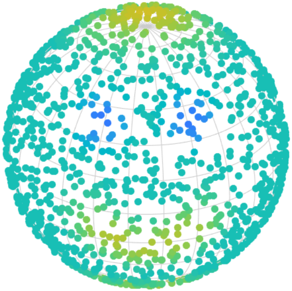
quadratur

lsqr

time domain

point evaluations

$$\underbrace{f(\xi_1), \ldots, f(\xi_N)}_{N}$$



Definition

A quadrature formula $Q_{\mathcal{X},\boldsymbol{W}}$ for the vertices $\mathcal{X} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N)^\top$ and weights $\boldsymbol{W} = \operatorname{diag}(w_1, \ldots, w_n)$ is **exact** upto polynomial degree M, iff

$$Q_{\mathcal{X},\boldsymbol{W}} f \coloneqq \sum_{n=1}^{N} w_n f(\boldsymbol{\xi}_n) = \int_{\mathbb{S}^2} f(\boldsymbol{\xi}) \, \mathrm{d}\omega(\boldsymbol{\xi}) \quad \forall f \in \mathbb{P}_M \left(\mathbb{S}^2\right).$$

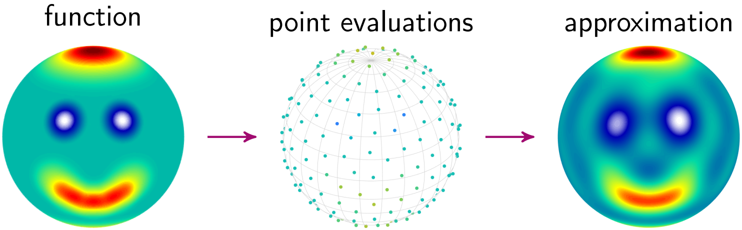
Definition

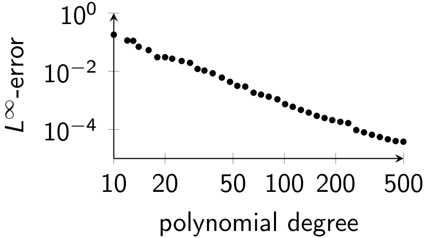
A quadrature formula $Q_{\mathcal{X},\boldsymbol{W}}$ for the vertices $\mathcal{X} = (\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_N)^\top$ and weights $\boldsymbol{W} = \operatorname{diag}(w_1, \dots, w_n)$ is **exact** upto polynomial degree M, iff

$$Q_{\mathcal{X},\boldsymbol{W}} f := \sum_{n=1}^{N} w_n f(\boldsymbol{\xi}_n) = \int_{\mathbb{S}^2} f(\boldsymbol{\xi}) \,\mathrm{d}\omega(\boldsymbol{\xi}) \quad \forall f \in \mathbb{P}_M\left(\mathbb{S}^2\right).$$

If $f \in \mathbb{P}_M(\mathbb{S}^2)$ and $Q_{\mathcal{X},\boldsymbol{W}}$ is exact upto to polynomial degree $2M$, then for $m \leq M, |l| \leq m$ holds

$$\hat{f}_{m,l} = \int_{\mathbb{S}^2} \underbrace{f(\boldsymbol{\xi}) \overline{Y_{m,l}(\boldsymbol{\xi})}}_{\in \mathbb{P}_{2M}(\mathbb{S}^2)} \,\mathrm{d}\omega(\boldsymbol{\xi}) = \sum_{n=1}^{N} w_n f(\boldsymbol{\xi}_n) \overline{Y_{m,l}(\boldsymbol{\xi}_n)} = \left(\boldsymbol{Y}_{\mathcal{X},M}^{\mathrm{H}} \boldsymbol{W} \boldsymbol{f}\right)_{(m,l)}$$


function
point evaluations
approximation


10^{0}
10^{-2}
10^{-4}
L^∞-error
10
20
50
100
200
500
polynomial degree

- number of vertices N bigger than $\dim(\mathbb{P}_M(\mathbb{S}^2)) = (M+1)^2$
- minimization problem

$$\left\|\boldsymbol{f} - \boldsymbol{Y}_{\mathcal{X},M}\tilde{\boldsymbol{f}}\right\|_{\boldsymbol{W}}^2 = \sum_{n=1}^{N} w_n \left| f(\boldsymbol{\xi}_n) - \left(\boldsymbol{Y}_{\mathcal{X},M}\tilde{\boldsymbol{f}}\right)_n \right|^2 \xrightarrow{\tilde{\boldsymbol{f}}} \min$$

with weights $\boldsymbol{W} = \operatorname{diag}(w_1, \dots, w_n)$

- number of vertices N bigger than $\dim(\mathbb{P}_M(\mathbb{S}^2)) = (M+1)^2$
- minimization problem

$$\left\|\boldsymbol{f} - \boldsymbol{Y}_{\mathcal{X},M}\tilde{\boldsymbol{f}}\right\|_{\boldsymbol{W}}^2 = \sum_{n=1}^{N} w_n \left| f(\boldsymbol{\xi}_n) - \left(\boldsymbol{Y}_{\mathcal{X},M}\tilde{\boldsymbol{f}}\right)_n \right|^2 \xrightarrow{\tilde{\boldsymbol{f}}} \min$$

with weights $\boldsymbol{W} = \operatorname{diag}(w_1, \dots, w_n)$

- apply lsqr on normal equation of first kind

$$\boldsymbol{Y}_{\mathcal{X},M}^{\mathrm{H}} \boldsymbol{W} \boldsymbol{Y}_{\mathcal{X},M} \tilde{\boldsymbol{f}} = \boldsymbol{Y}_{\mathcal{X},M}^{\mathrm{H}} \boldsymbol{W} \boldsymbol{f}$$

- use the voronoi weights in $\boldsymbol{W}$

Applications in MTEX