

S2Fun

Ralf Hielscher and Felix Bartel

TU Chemnitz

28 Feb 2018

1 Using S2Fun

- General idea

- Defining a S2Fun

- Plotting a S2Fun

- Calculating with S2Fun

- Multivariate S2Fun

- S2VectorField

- S2AxisField

- S2FunSym

2 Underlying mathematics

- Spherical harmonics

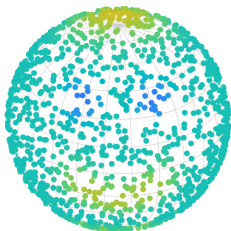
- Quadrature

- Approximation

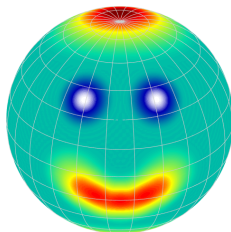
3 Applications in MTEX

- get a function which approximates the data

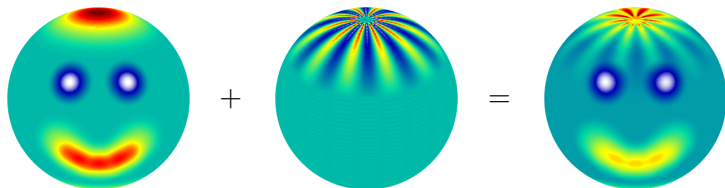
given: scattered data on \mathbb{S}^2



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



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

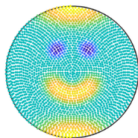


- generalize the concept to $f: \mathbb{S}^2 \rightarrow \mathbb{R}^n$, functions with symmetries, vector fields and axis fields

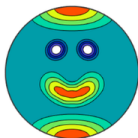
Using S2Fun

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
```



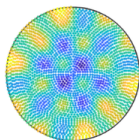
```
sF1 = interp(nodes,y,'harmonicApproximation');  
plot(sF1); % plot the spherical function
```



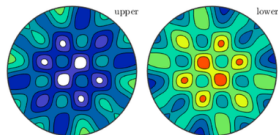
Using S2Fun

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
```



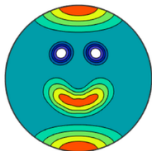
```
fhat = rand(25,1);  
sF3 = S2FunHarmonic(fhat)
```

- the Fourier coefficients are specified as a column vector

Using S2Fun

Plotting a S2Fun

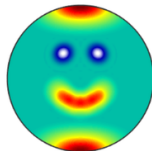
```
plot(sF1);
```



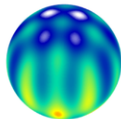
```
contour(sF1);
```



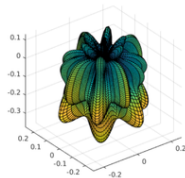
```
pcolor(sF1);
```



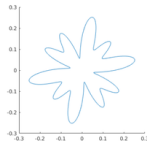
```
plot3d(sF2);
```



```
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

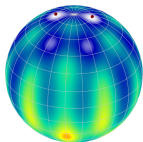
```
sF1.^sF2; 2.^sF1; sF2.^4;
```

absolute value

```
abs(sF1);
```

global minimum

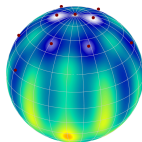
```
[val,nodes] = min(sF2);  
plot3d(sF2);  
scatter3d(nodes);
```



- val has one value

local minimas

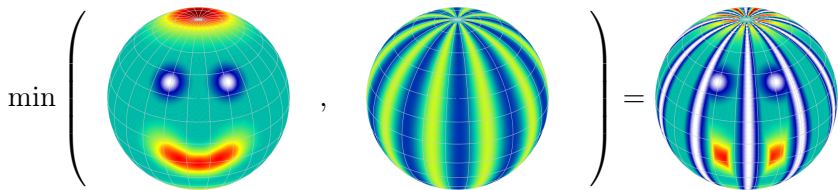
```
[val,nodes] = min(sF2, '  
    numLocal',10);  
scatter3d(nodes);
```



- val has length(nodes) values

(same conventions hold for the max command)

minimum in the pointwise sense



Using S2Fun

Calculating with S2Fun - other operations

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

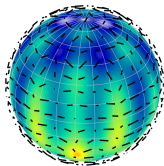
rotate

```
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 converted to S2VectorFieldHarmonic

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

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

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

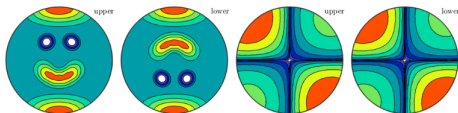
Using S2Fun

Multivariate S2Fun - definition via approximation

```
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
```



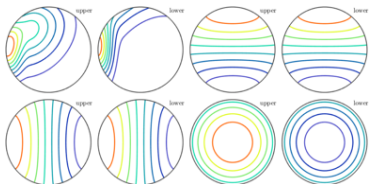
Using S2Fun

Multivariate S2Fun - definition via quadrature

```
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

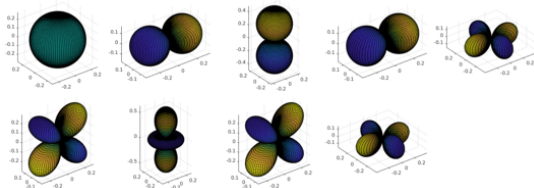
```
sF2 = S2FunHarmonic.quadrature(f, 'bandwidth', 50)  
contour(sF2); % plot the spherical function
```



Using S2Fun

Multivariate S2Fun - definition via Fourier coefficients

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



Using S2Fun

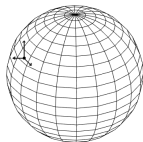
Multivariate S2Fun - Matlab's matrix operations

```
sF4 = [sF1; sF2]; % concatenation
sF4(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(sF3,1); % pointwise minimum over the specified dimension
```

definition via multivariate S2FunHarmonic

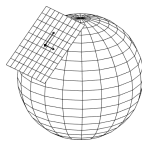
x -, y - and z -component

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



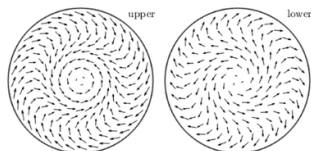
ϑ - and ρ -component

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



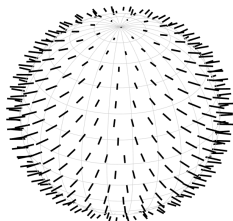
definition via approximation

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

multiplication/division

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

dot product

```
dot(sVF1,sVF2);  
dot(sVF1,zvector);
```

cross product

```
cross(sVF1,sVF2);  
cross(sVF1,yvector);
```

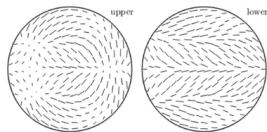
mean

```
mean(sVF1);
```

norm

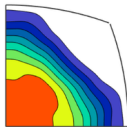
```
norm(sVF1);
```

```
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`

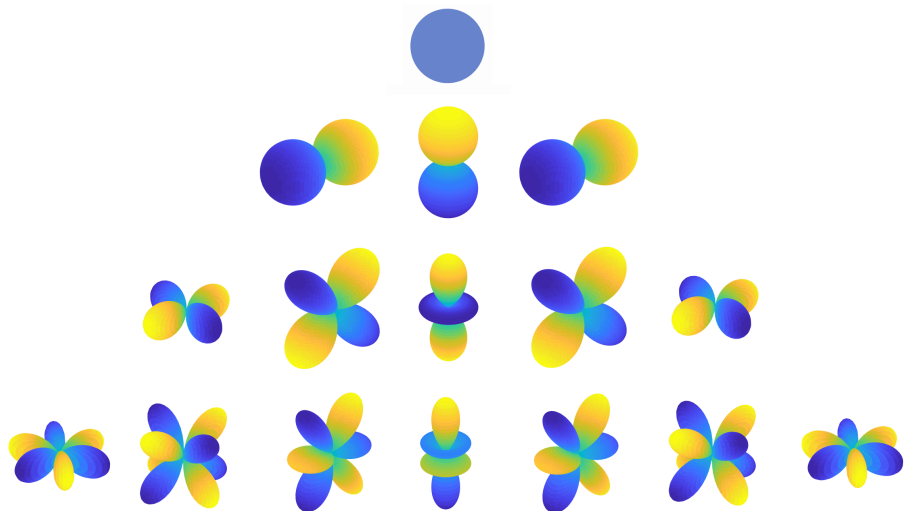
```
sF = S2FunHarmonic.quadrature(@(v) smiley(v))  
cs = crystalSymmetry('432');  
  
sFs = symmetrise(sF,cs);  
plot(sFs);
```



- 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

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

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

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

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

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

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

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

with Fourier coefficients

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

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

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

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

Underlying mathematics

Spherical harmonics

frequency domain

Fourier coefficients

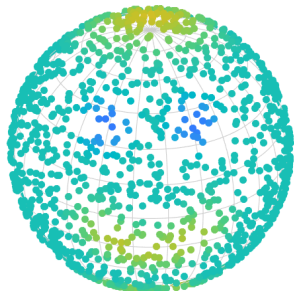
$$\begin{array}{cccc} & & & \hat{f}_{M,-M} \\ & & \dots & \\ & \hat{f}_{1,-1} & & \\ \hat{f}_{0,0} & \hat{f}_{1,0} & \dots & \vdots \\ & \hat{f}_{1,1} & & \\ & & \dots & \\ & & & \hat{f}_{M,M} \end{array}$$

$(M+1)^2$

time domain

point evaluations

$$\underbrace{f(\xi_1), \dots, f(\xi_N)}_N$$



Underlying mathematics

Spherical harmonics

frequency domain

Fourier coefficients

$$\begin{array}{cccc} & & & \hat{f}_{M,-M} \\ & & \dots & \\ & \hat{f}_{1,-1} & & \\ \hat{f}_{0,0} & \hat{f}_{1,0} & \dots & \vdots \\ & \hat{f}_{1,1} & & \\ & & \dots & \\ & & & \hat{f}_{M,M} \end{array}$$

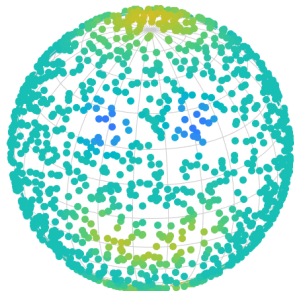
$(M+1)^2$

$\mathbf{Y}_{\mathcal{X},M}$
NFSFT

time domain

point evaluations

$$\underbrace{f(\xi_1), \dots, f(\xi_N)}_N$$



Underlying mathematics

Spherical harmonics

frequency domain

Fourier coefficients

$$\underbrace{\begin{array}{cccc} & & & \hat{f}_{M,-M} \\ & & \dots & \\ & \hat{f}_{1,-1} & & \\ \hat{f}_{0,0} & \hat{f}_{1,0} & \dots & \vdots \\ & \hat{f}_{1,1} & & \\ & & \dots & \\ & & & \hat{f}_{M,M} \end{array}}_{(M+1)^2}$$

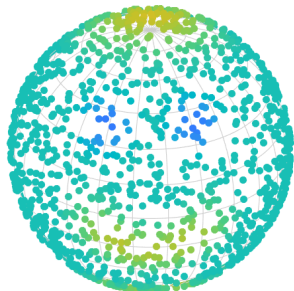
$\mathbf{Y}_{\mathcal{X},M}$
NFSFT

quadratur
lsqr

time domain

point evaluations

$$\underbrace{f(\xi_1), \dots, f(\xi_N)}_N$$



Definition

A quadrature formula $Q_{\mathcal{X}, \mathbf{W}}$ for the vertices $\mathcal{X} = (\xi_1, \dots, \xi_N)^\top$ and weights $\mathbf{W} = \text{diag}(w_1, \dots, w_n)$ is **exact** upto polynomial degree M , iff

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

Definition

A quadrature formula $Q_{\mathcal{X}, \mathbf{W}}$ for the vertices $\mathcal{X} = (\xi_1, \dots, \xi_N)^T$ and weights $\mathbf{W} = \text{diag}(w_1, \dots, w_N)$ is **exact** upto polynomial degree M , iff

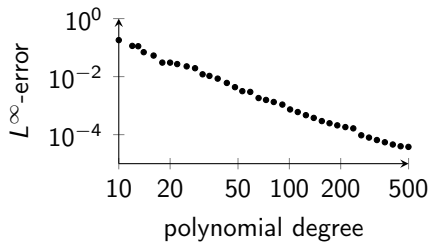
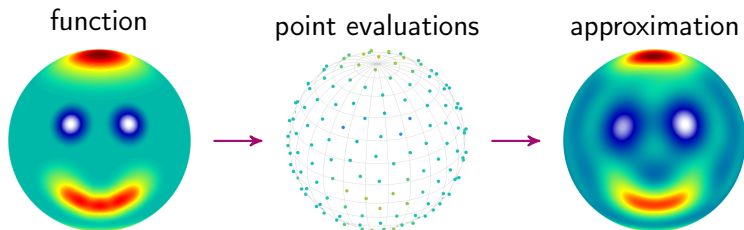
$$Q_{\mathcal{X}, \mathbf{W}} f := \sum_{n=1}^N w_n f(\xi_n) = \int_{\mathbb{S}^2} f(\xi) d\omega(\xi) \quad \forall f \in \mathbb{P}_M(\mathbb{S}^2).$$

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

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

Underlying mathematics

Quadrature



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

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

with weights $\mathbf{W} = \text{diag}(w_1, \dots, w_n)$

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

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

with weights $\mathbf{W} = \text{diag}(w_1, \dots, w_n)$

- apply lsqr on normal equation of first kind

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

- use the voronoi weights in \mathbf{W}

