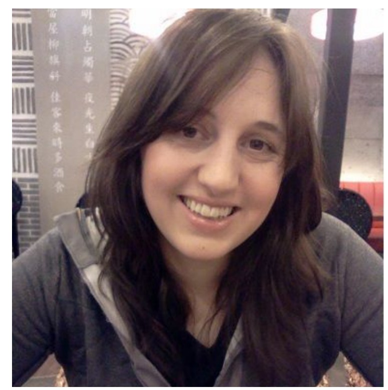
Low rank approximation of functions in polar and spherical geometries

Grady Wright

Boise State University

Alex Townsend

Cornell University



Heather Wilber

Cornell University

4th Dolomites Workshop on Constructive Approximation

12 September 2016

Where is Boise (boy-see)?

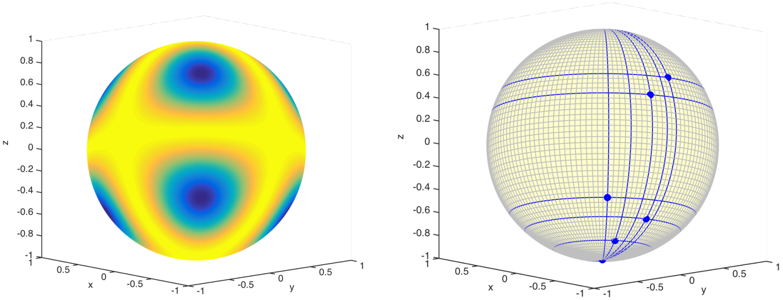
Motivation

- Approximation of functions on the disk and surface of the sphere and arise in many applications:
  - Numerical weather/climate prediction, solid earth geophysics, geodesy, geomagnetism, heliophysics, cosmology, computer graphics, biology, etc.
- No general purpose software tool for computing with functions on the disk and sphere.
  - Function manipulations, integration, differentiation, vector calculus, differential equations.
- Motivated by *Chebfun*, we set out to develop such software packages, *Spherefun* and *Diskfun*. Requirements:
  - Fast algorithms with high-order accuracy
  - All discretizations are hidden from the user (unless they are desired)
  - Open-source
  - Implemented in an integrated environment: MATLAB

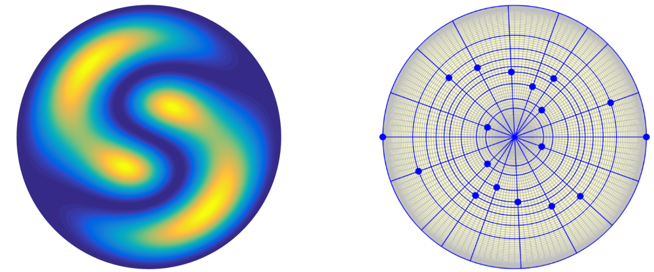
Spherefun and Diskfun Software

Townsend, Wilber, & W (2016a,b)

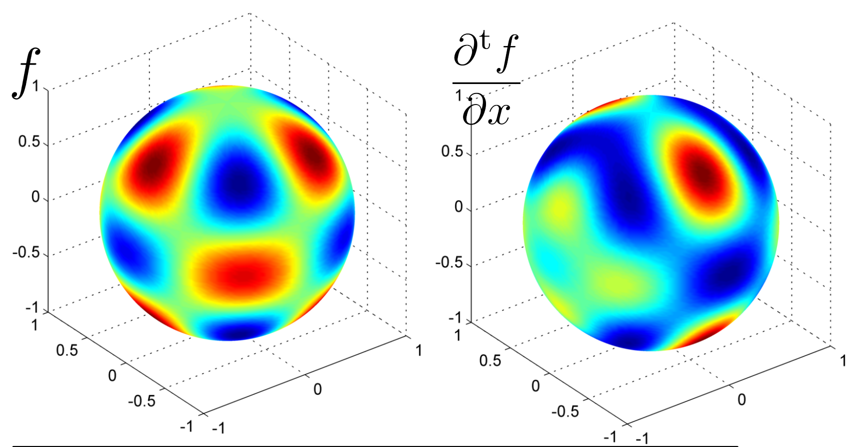
www.chebfun.org



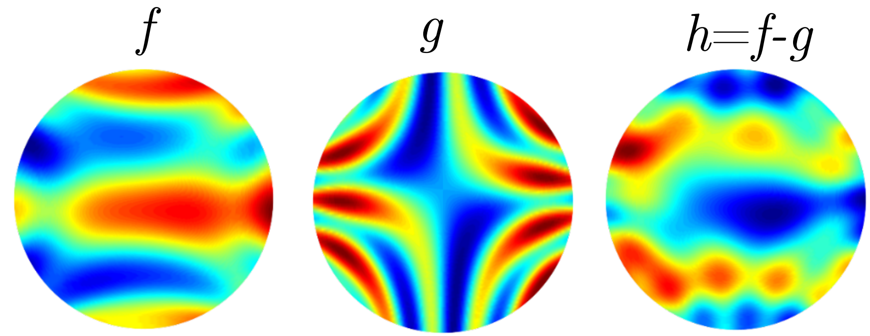
```
>>f = spherefun(@(x,y,z)cos(5*x*y*z));
>>plot(f)
```



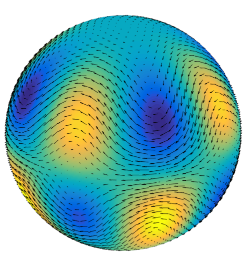
```
>>f=diskfun(@(t,r)cos(sin(pi*r).*cos(t)+…
                  sin(2*pi*r).*sin(t)));
>>plot(f)
```



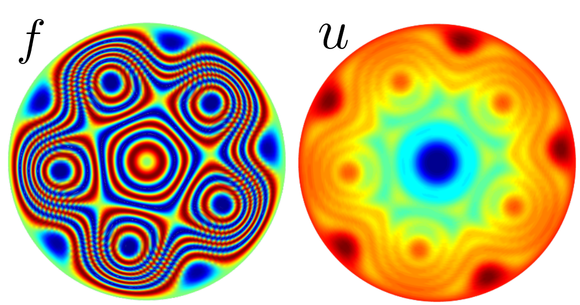

```
>>f = spherefun.sphharm(5,3);
>>plot(diff(f,1))
```




```
>>h = f-g; sum2(h)
ans =
   0.818084875090377
```



```
>>u = curl(f)
>>v = vort(u)
>>quiver(u)
>>plot(v)
```

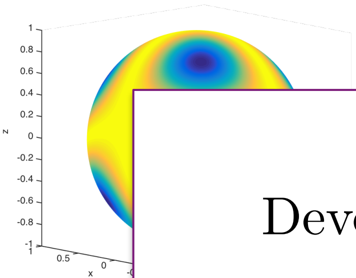
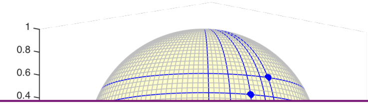
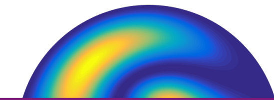
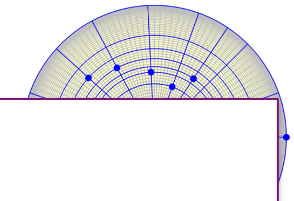



```
>>bc = @(t) 1 + 0*t;
>>u=Poisson(f,bc,1024);
>>plot(u)
```

Spherefun and Diskfun Software

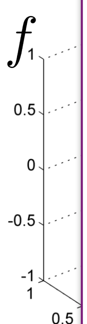
Townsend, Wilber, & W (2016a,b)

www.chebfun.org

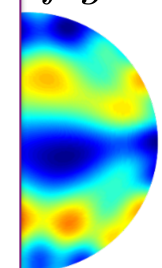





```
>>f =
>>plo
```

```
(t)+…
```



$=f\text{-}g$

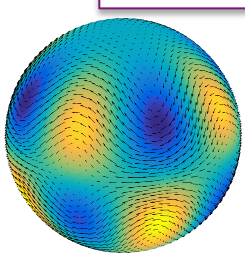


Developing such a tool requires algorithms that are

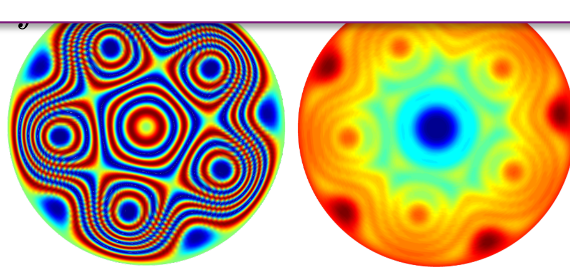
- Data-driven and adaptive
- Highly (spectral) accurate for smooth problems
- Computationally efficient
- Free of expensive pre-computational costs
- Free from issues with coordinate singularities

Double Fourier Sphere + Low rank function approximation

```
>>f
>>p
```



```
>>u = curl(f)
>>v = vort(u)
>>quiver(u)
>>plot(v)
```



```
>>bc = @(t) 1 + 0*t;
>>u=Poisson(f,bc,1024);
>>plot(u)
```

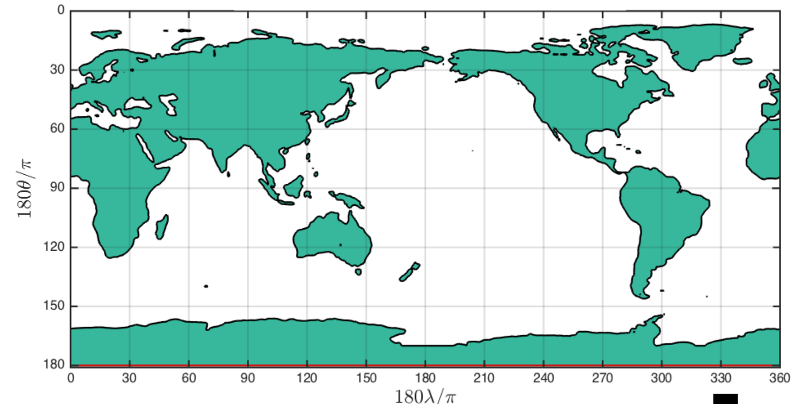
Part I: Double Fourier Sphere (DFS) method

Double Fourier Sphere (DFS) method

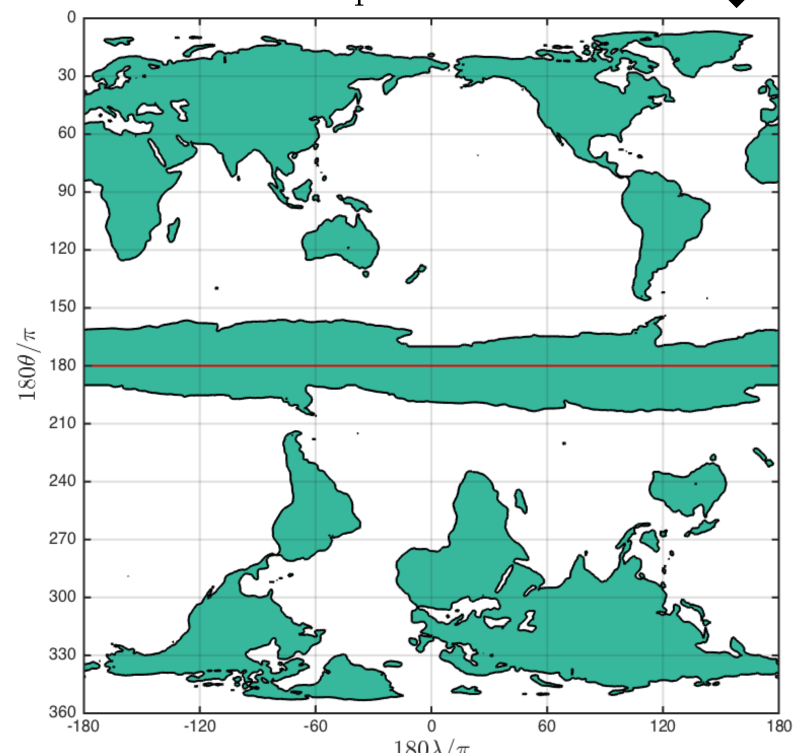
Illustration:

Cartesian coordinates

Spherical coordinates



“Doubled” spherical coordinates



Merilees (1973), Orszag (1974), Yee (1982), Fornberg (1997), Spotz et. al. (1998), Shen (1999), Cheong (2000), Ganesh & Mhaskar (2006)

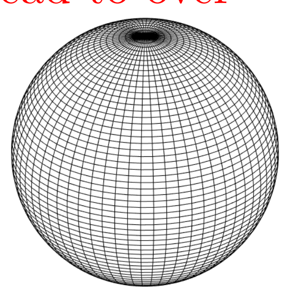
What do we gain/lose?

- The domain is doubly periodic.

  Can use FFT in both directions.

- Poles are not treated as boundaries.

- Issue: tensor product grids lead to over resolution at the poles.

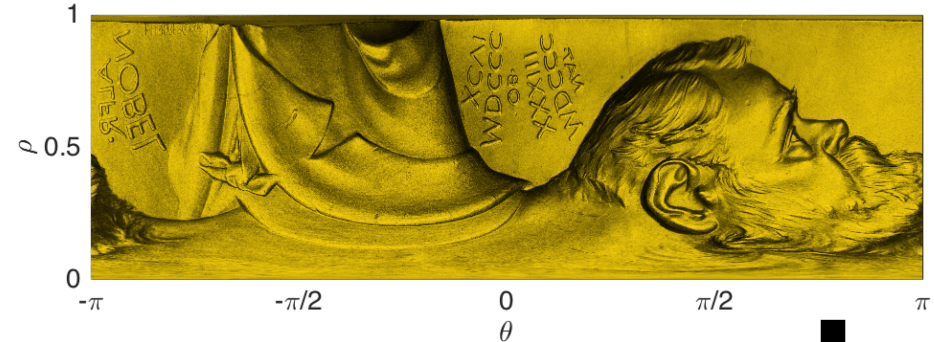


Disk analogue of the DFS method

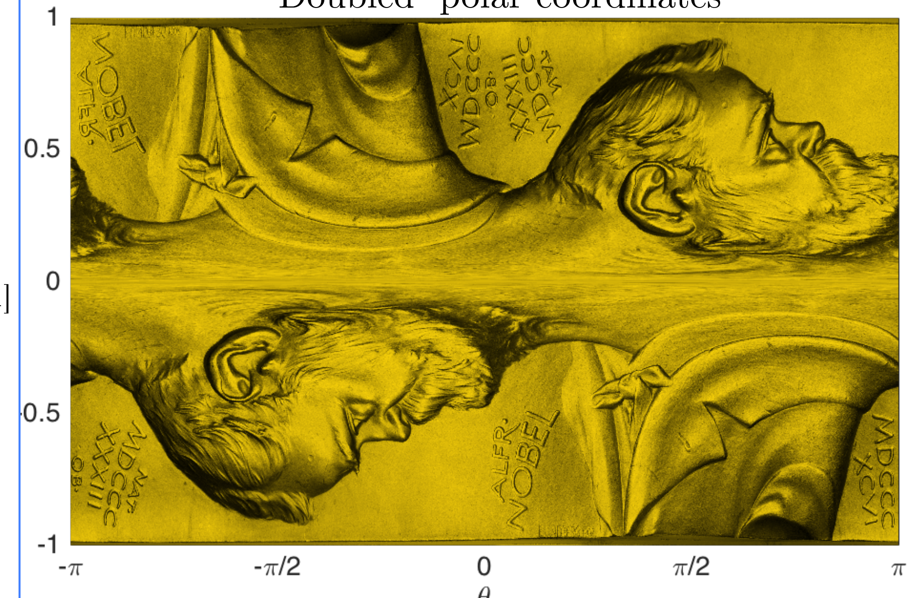
Illustration:

Cartesian coordinates

Polar coordinates



“Doubled” polar coordinates

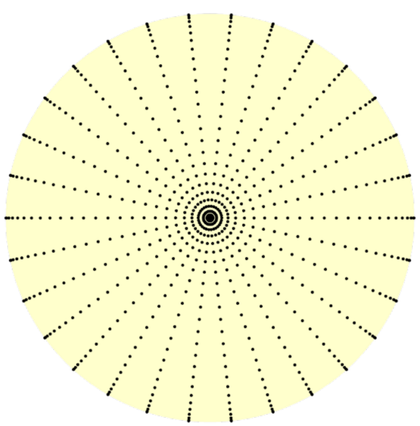


Eisen *et. al.* (1991), Fornberg (1995), Torres & Coutsias (1999), Shen (2000), Trefethen (2000).

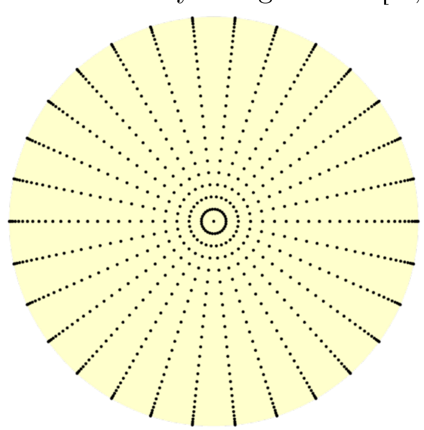
What do we gain/lose?

- Domain is well setup for tensor product Fourier-Chebyshev expansions.
- The origin is not treated as a boundary.
- Less clustering, but it’s still an issue

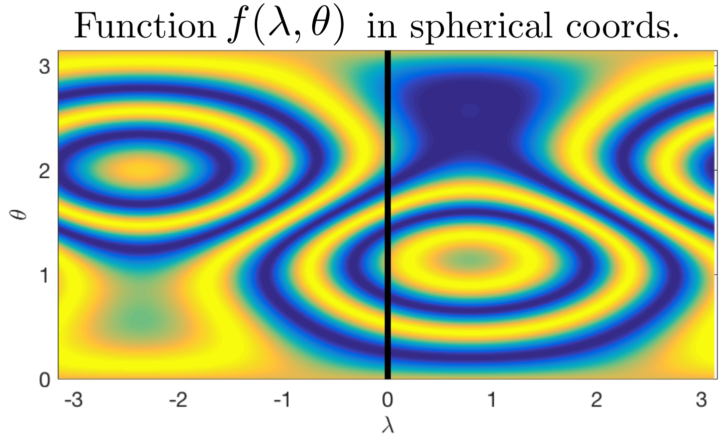
Radial Chebyshev grid over [0,1]

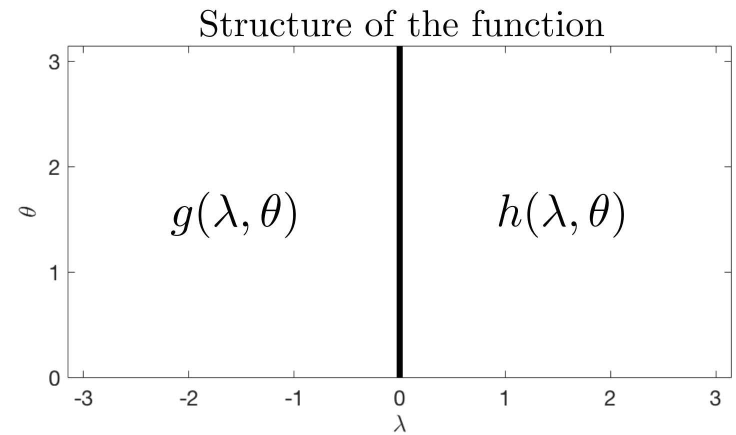


Radial Chebyshev grid over [-1,1]

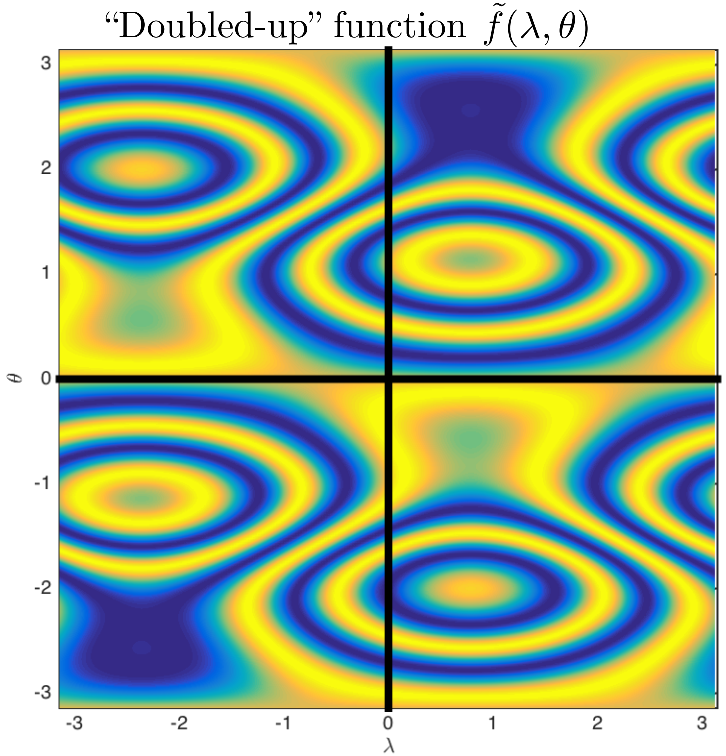


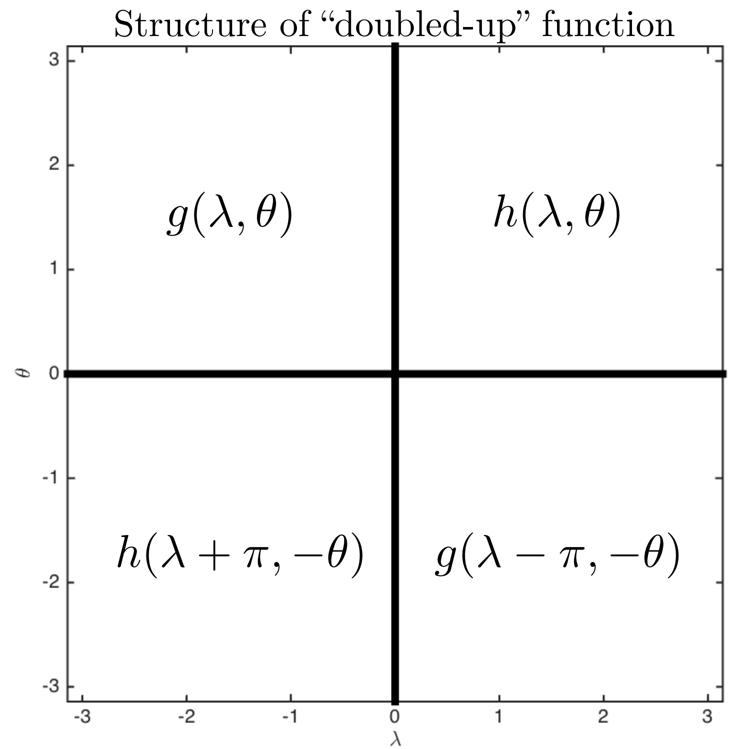
Properties of the DFS method



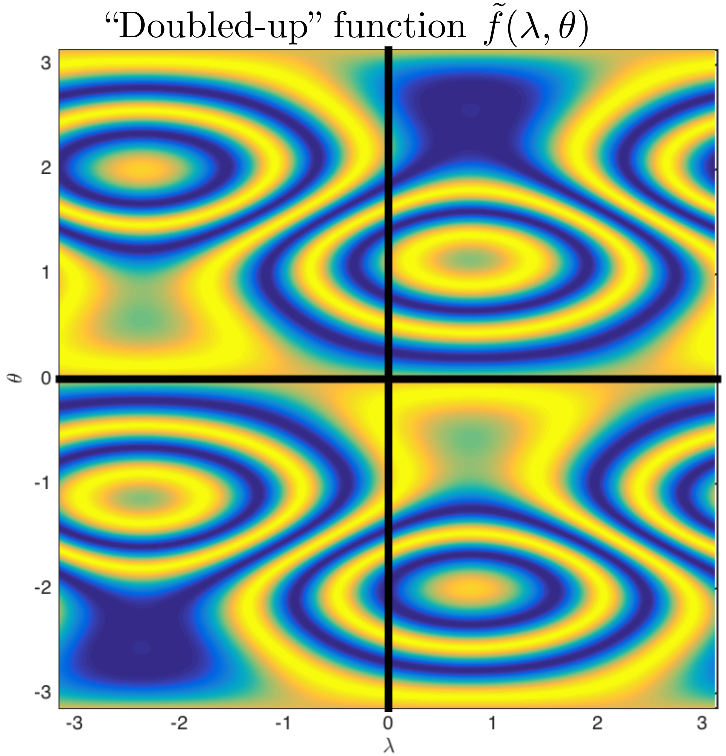


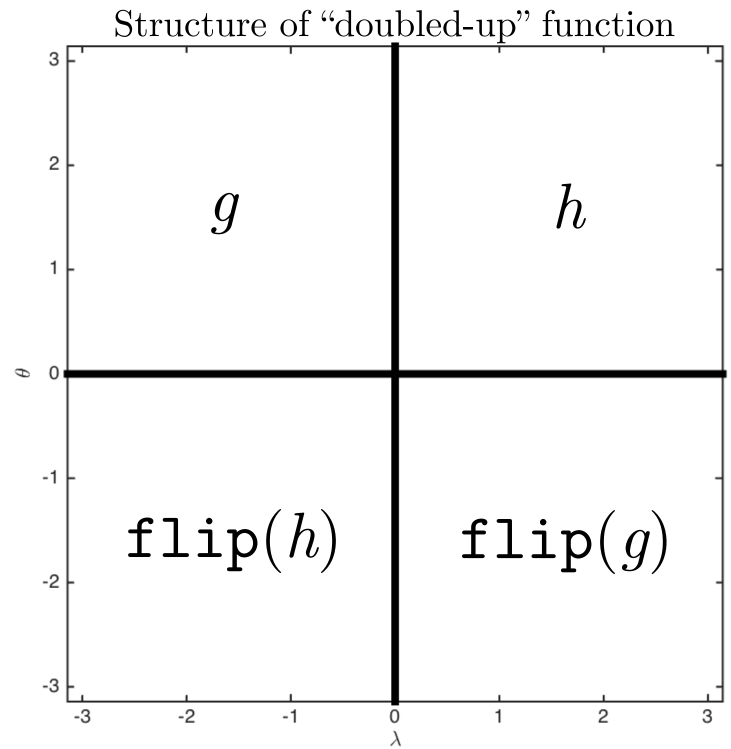
Properties of the DFS method





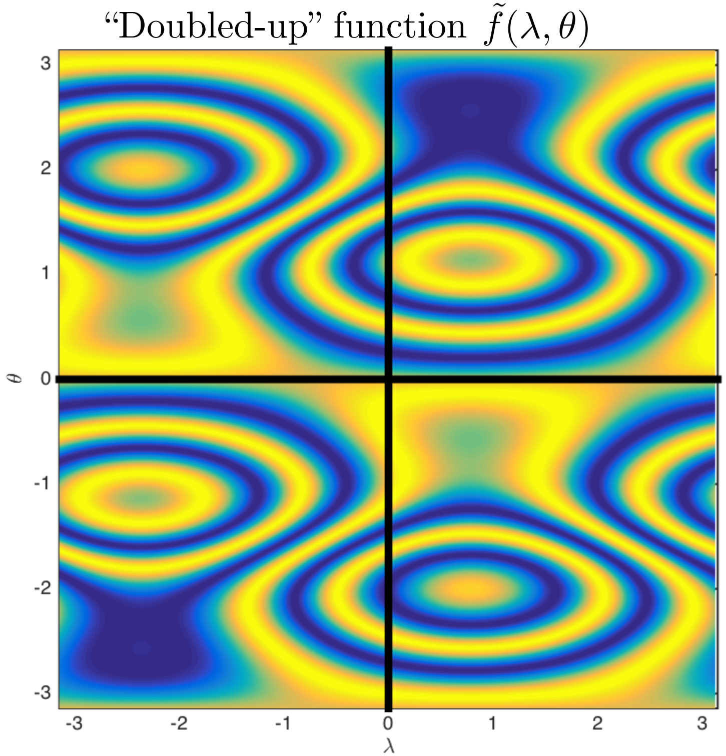
Properties of the DFS method

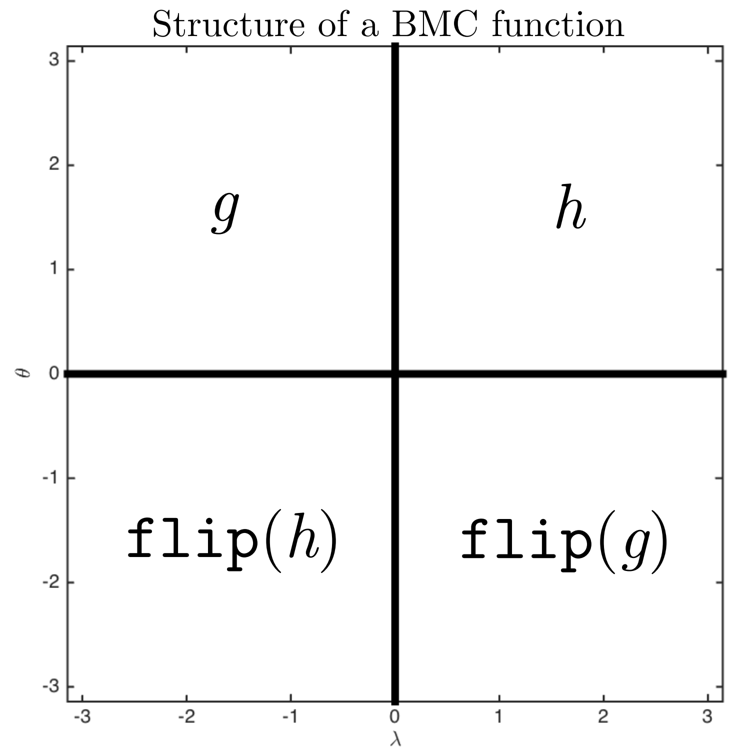




- We call functions with this structure Block Mirror Centrosymmetric (BMC).
- We refer to the "doubling up" as the BMC extension of a function.
- Similar BMC extension for functions on a disk (in radial direction).

Properties of the DFS method





Function is now doubly periodic: expand in Fourier series (using the FFT):

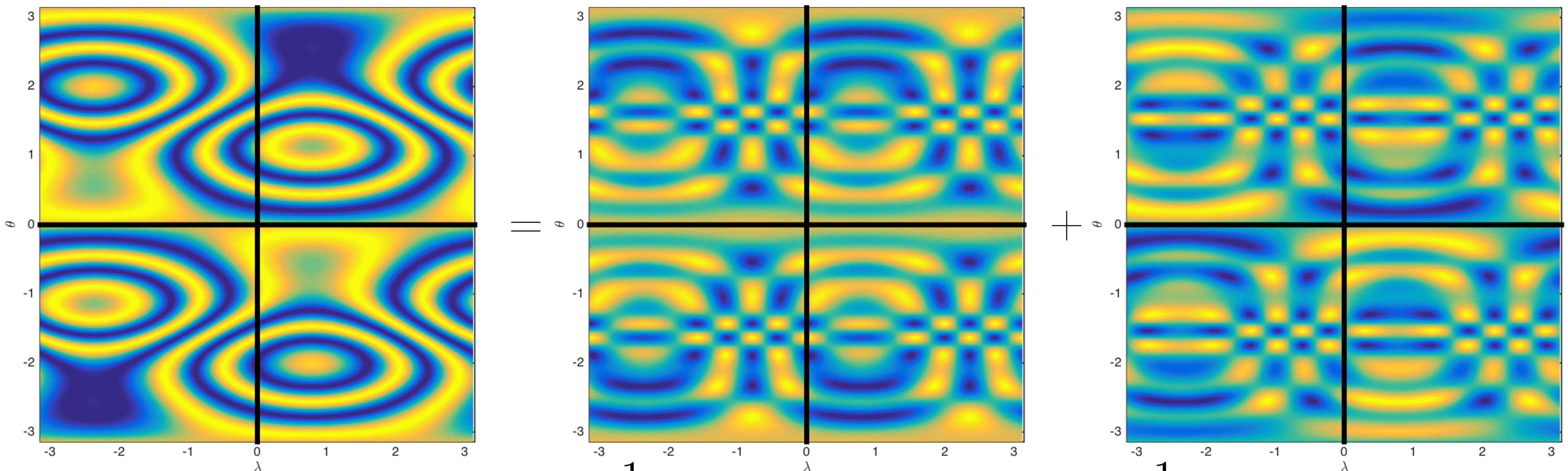
$$\tilde{f}(\lambda, \theta) \approx \sum_{j=-\frac{m}{2}}^{\frac{m}{2}-1} \sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} a_{jk} e^{ij\theta} e^{ik\lambda}, \qquad (\lambda, \theta) \in [-\pi, \pi]^2,$$

Issues:

1. Oversampling at the poles
2. Even-odd symmetries must be accounted for in approximation (Yee, 1980)

Properties of the DFS method

"Doubled-up" function $\tilde{f}(\lambda,\theta)$ | Even and π-periodic | Odd and π-antiperiodic



$\tilde{f}(\lambda,\theta)$ | $\frac{1}{2}(\tilde{f}(\lambda,\theta)+\tilde{f}(\lambda,\theta))$ | $\frac{1}{2}(\tilde{f}(\lambda,\theta)-\tilde{f}(\lambda,\theta))$

Function is now doubly periodic: expand in Fourier series (using the FFT):

$$\tilde{f}(\lambda,\theta) \approx \sum_{j=-\frac{m}{2}}^{\frac{m}{2}-1} \sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} a_{jk} e^{ij\theta} e^{ik\lambda}, \qquad (\lambda,\theta) \in [-\pi,\pi]^2,$$

Issues:

1. Oversampling at the poles
2. Even-odd symmetries must be accounted for in approximation (Yee, 1980)

Part II: Low rank function approximation

Low rank function approximation: some definitions

- A non-zero bivariate function f is called a rank 1 function if it is the product of a function in x and a function in y:

$$\text{Rank 1 function: } f(x,y) = g(y)h(x)$$

- A bivariate function f is called a rank K function if it can be written as the sum of K rank 1 functions:

$$\text{Rank } K \text{ function: } f(x,y) = \sum_{k=1}^{K} g_k(y)h_k(x)$$

- Generally functions are of infinite rank.

- But, smooth functions are typically of low finite numerical rank.

Low rank function approximation: SVD

Problem: Let $f : [-1,1]^2 \to \mathbb{R}$ be a smooth bivariate function. Find an approximation of f involving a sum of K rank 1 terms:

$$f(x,y) \approx \sum_{j=1}^{K} d_k \underbrace{c_k(y) r_k(x)}_{\text{rank 1 product}}, \quad (x,y) \in [-1,1]^2$$

This is called a low rank approximation of f.

Theorem *Let $f : [a,b] \times [c,d] \to \mathbb{R}$ be Lipschitz continuous. Then f has the following series expansion that converges absolutely and uniformly:*

$$f(x,y) = \sum_{k=1}^{\infty} \sigma_k u_k(y) v_k(x), \; (x,y) \in [a,b] \times [c,d],$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$, and $\{u_k\}$ and $\{v_k\}$ are orthonormal sets of functions on $[c,d]$ and $[a,b]$, respectively. This is called the singular value decomposition (SVD) of f.

Proof. Schmidt (1907); Hammerstein (1927) $\square$

Low rank function approximation: alternative ideas

- Truncating the SVD of f after K terms gives the best rank K approximation of f in the L^2-norm.

BUT, obtaining the SVD is computationally intensive.

- Alternative, computationally efficient methods have been actively investigated for the past 20 years:
  - Pseudoskeleton approximation (Goreinov, Tyrtyshnikov & Zamarashkin, 1997)
  - Adaptive cross approximation (ACA) (Bebendorf 2000)
  - Newton-Geddes (Carvajal, Chapman & Geddes, 2005)
  - Interpolative decomposition (Halko, Martinsson, & Tropp, 2011)
  - Gaussian elimination (GE) (Townsend & Trefethen 2013; Chebfun2)

- Data-driven, adaptive algorithms that give near-optimal low rank approximations, especially when f is smooth.

Gaussian elimination on matrices: a different view

- Example: GE with complete pivoting on a 6-by-6 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix}$$

Gaussian elimination on matrices: a different view

Step 1: Let a_{32} be the maximum value in magnitude of A_0:

$$A_0 = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix}$$

"Zero-out" row 3 and column 4 with rank 1 "outer product":

$$A_1 = A_0 - \underbrace{\frac{1}{a_{32}}}_{d_1} \underbrace{\begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \\ a_{42} \\ a_{52} \\ a_{62} \end{bmatrix}}_{\mathbf{c}_1} \underbrace{\begin{bmatrix} a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \end{bmatrix}}_{\mathbf{r}_1^T}$$

Gaussian elimination on matrices: a different view

Step 1: Let a_{32} be the maximum value in magnitude of A_0:

$$A_1 = \begin{bmatrix} a_{11} & 0 & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & 0 & a_{23} & a_{24} & a_{25} & a_{26} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ a_{41} & 0 & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & 0 & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & 0 & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix}$$

"Zero-out" row 3 and column 4 with rank 1 "outer product":

$$A_1 = A_0 - \underbrace{\frac{1}{a_{32}}}_{d_1} \underbrace{\begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \\ a_{42} \\ a_{52} \\ a_{62} \end{bmatrix}}_{\mathbf{c}_1} \underbrace{\begin{bmatrix} a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \end{bmatrix}}_{\mathbf{r}_1^T}$$

Gaussian elimination on matrices: a different view

Step 2: Let a_{54} be the maximum value in magnitude of A_1:

$$A_1 = \begin{bmatrix} a_{11} & 0 & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & 0 & a_{23} & a_{24} & a_{25} & a_{26} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ a_{41} & 0 & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & 0 & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & 0 & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix}$$

"Zero-out" row 5 and column 4 with rank 1 "outer product":

$$A_2 = A_1 - \underbrace{\frac{1}{a_{54}}}_{d_2} \underbrace{\begin{bmatrix} a_{14} \\ a_{24} \\ 0 \\ a_{44} \\ a_{54} \\ a_{64} \end{bmatrix}}_{\mathbf{c}_2} \underbrace{\begin{bmatrix} a_{51} & 0 & a_{53} & a_{54} & a_{55} & a_{56} \end{bmatrix}}_{\mathbf{r}_2^T}$$

Gaussian elimination on matrices: a different view

Step 3: Let a_{23} be the maximum value in magnitude of A_2:

$$A_2 = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 & a_{15} & a_{16} \\ a_{21} & 0 & a_{23} & 0 & a_{25} & a_{26} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ a_{41} & 0 & a_{43} & 0 & a_{45} & a_{46} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ a_{61} & 0 & a_{63} & 0 & a_{65} & a_{66} \end{bmatrix}$$

"Zero-out" row 2 and column 3 with rank 1 "outer product":

$$A_3 = A_2 - \underbrace{\frac{1}{a_{23}}}_{d_3} \underbrace{\begin{bmatrix} a_{13} \\ a_{23} \\ 0 \\ a_{43} \\ 0 \\ a_{63} \end{bmatrix}}_{\mathbf{c}_3} \underbrace{\begin{bmatrix} a_{21} & 0 & a_{23} & 0 & a_{25} & a_{26} \end{bmatrix}}_{\mathbf{r}_3^T}$$

Gaussian elimination on matrices: a different view

Step 4: Let a_{41} be the maximum value in magnitude of A_3:

$$A_3 = \begin{bmatrix} a_{11} & 0 & 0 & 0 & a_{15} & a_{16} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ a_{41} & 0 & 0 & 0 & a_{45} & a_{46} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ a_{61} & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

"Zero-out" row 4 and column 1 with rank 1 "outer product":

$$A_4 = A_3 - \underbrace{\frac{1}{a_{41}}}_{d_4} \underbrace{\begin{bmatrix} a_{11} \\ 0 \\ 0 \\ a_{41} \\ 0 \\ a_{61} \end{bmatrix}}_{\mathbf{c}_4} \underbrace{\begin{bmatrix} a_{41} & 0 & 0 & 0 & a_{45} & a_{46} \end{bmatrix}}_{\mathbf{r}_4^T}$$

Gaussian elimination on matrices: a different view

Step 5: Let a_{16} be the maximum value in magnitude of A_4:

$$A_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & a_{15} & a_{16} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

"Zero-out" row 1 and column 6 with rank 1 "outer product":

$$A_5 = A_4 - \underbrace{\frac{1}{a_{16}}}_{d_5} \underbrace{\begin{bmatrix} a_{16} \\ 0 \\ 0 \\ 0 \\ 0 \\ a_{66} \end{bmatrix}}_{\mathbf{c}_5} \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & a_{15} & a_{16} \end{bmatrix}}_{\mathbf{r}_5^T}$$

Gaussian elimination on matrices: a different view

Step 6: Let a_{65} be the maximum value in magnitude of A_5:

$$A_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{65} & 0 \end{bmatrix}$$

"Zero-out" row 6 and column 5 with rank 1 "outer product":

$$A_6 = A_5 - \underbrace{\frac{1}{a_{65}}}_{d_6} \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ a_{65} \end{bmatrix}}_{\mathbf{c}_6} \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & a_{65} & 0 \end{bmatrix}}_{\mathbf{r}_6^T}$$

Gaussian elimination on matrices: a different view

A_6 contains all zeros:

$$A_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

An LU factorization of A can be obtained from d_j, $\mathbf{c}_j$, and $\mathbf{r}_j$, j=1,…,6.

Additionally:

We can use d_j, $\mathbf{c}_j$, and $\mathbf{r}_j$ to write A as the following sum of rank 1 terms:

$$A = \sum_{j=1}^{6} d_j \mathbf{c}_j \mathbf{r}_j^T$$

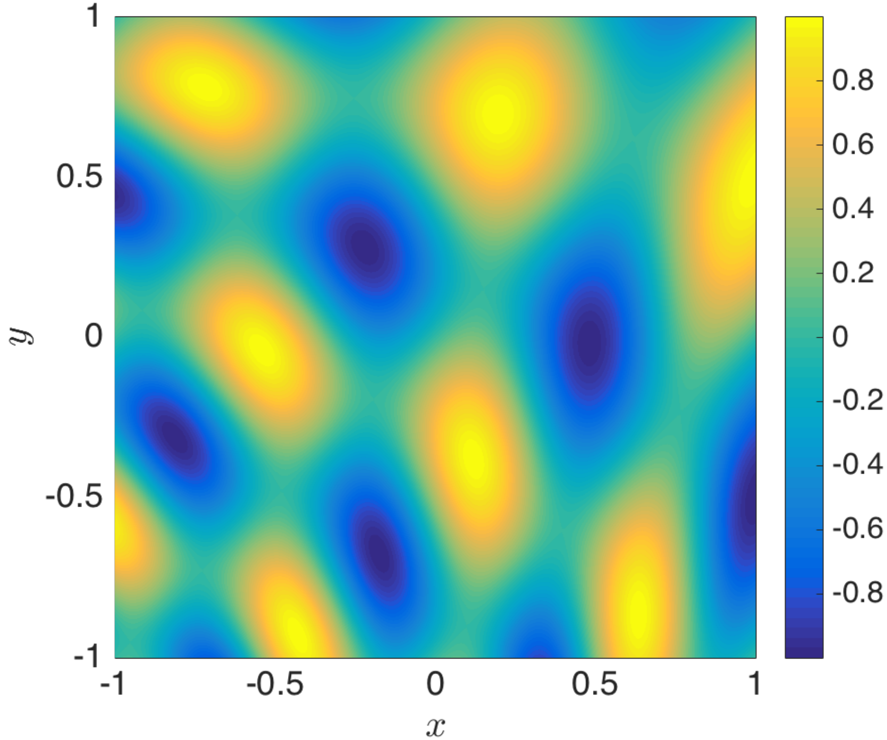
Can we truncate this sum to obtain an approximation to A?

Yes, especially when A comes form sample of a smooth function

Gaussian elimination for functions: example

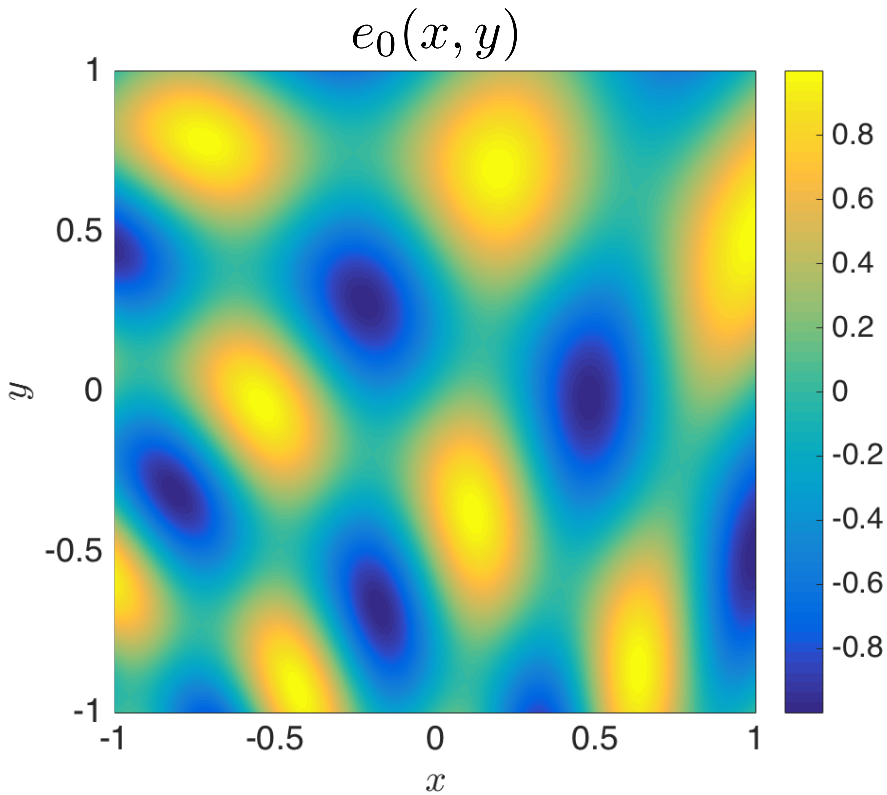
- Continuous analogue of Gaussian elimination with complete pivoting.

$f(x,y) = \cos(3(x-1)(y-2))\sin(\pi(x-y))$



Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.

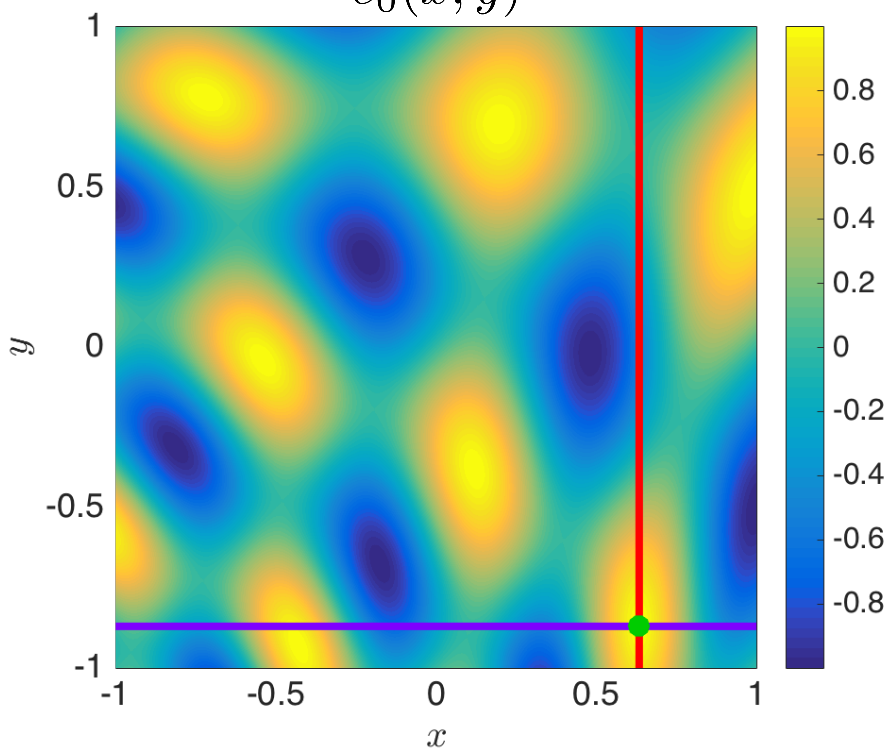


k=1

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.

$e_0(x,y)$



$|e_{k-1}(x_*, y_*)| = \|e_{k-1}\|_\infty$

$k=1$

pivot $d_k = 1/e_{k-1}(x_*, y_*)$

column slice $c_k(y) = e_{k-1}(x_*, y)$

row slice $r_k(x) = e_{k-1}(x, y_*)$

Rank 1 elimination function:

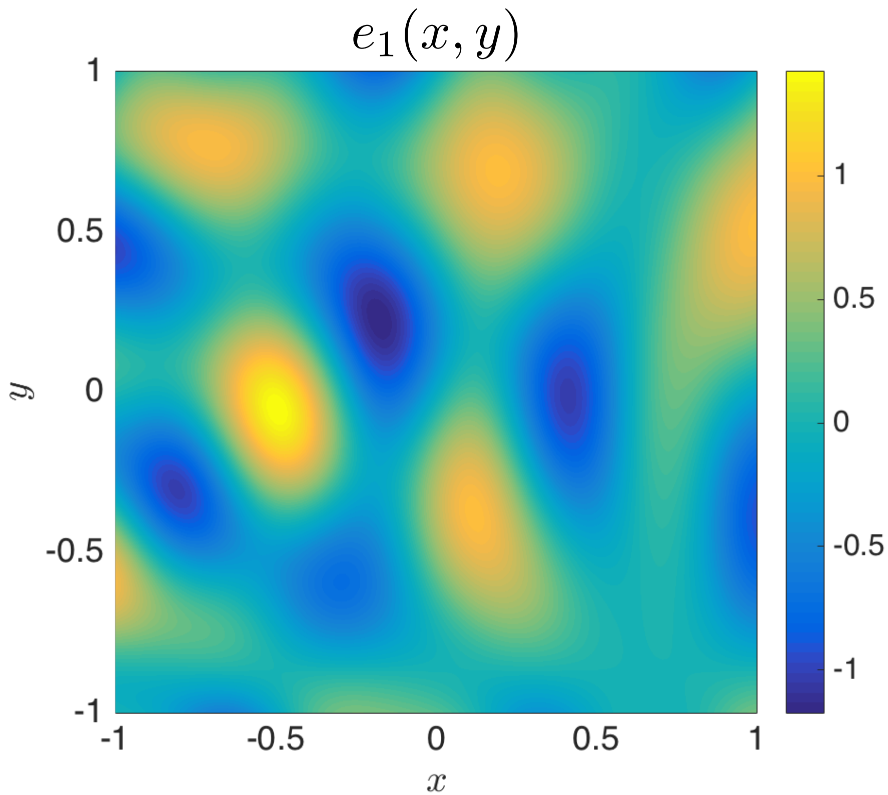
$$g_k(x,y) = d_k \; c_k(y) \; r_k(x)$$

Residual:

$$e_k(x,y) = e_{k-1}(x,y) - g_k(x,y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



$e_k(x, y_*) = 0$
$e_k(x_*, y) = 0$

k=1

pivot $d_k = 1/e_{k-1}(x_*, y_*)$

column slice $c_k(y) = e_{k-1}(x_*, y)$

row slice $r_k(x) = e_{k-1}(x, y_*)$

Rank 1 elimination function:

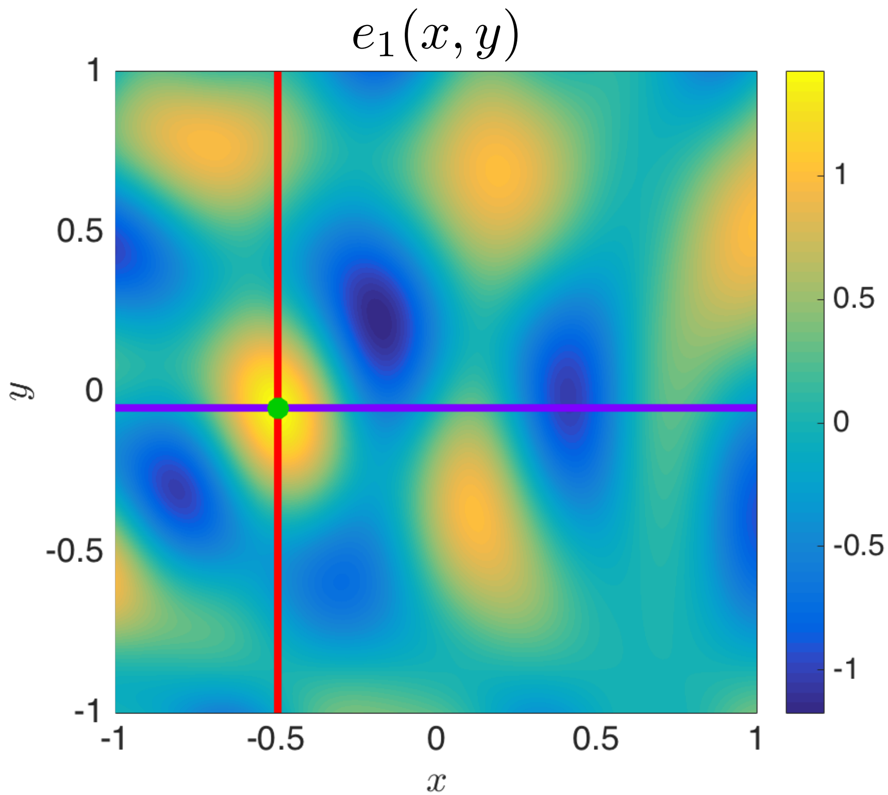
$$g_k(x,y) = d_k\ c_k(y)\ r_k(x)$$

Residual:

$$e_k(x,y) = e_{k-1}(x,y) - g_k(x,y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



$|e_{k-1}(x_*, y_*)| = \|e_{k-1}\|_\infty$

k=2

● pivot $d_k = 1/e_{k-1}(x_*, y_*)$

| column slice $c_k(y) = e_{k-1}(x_*, y)$

— row slice $r_k(x) = e_{k-1}(x, y_*)$

Rank 1 elimination function:

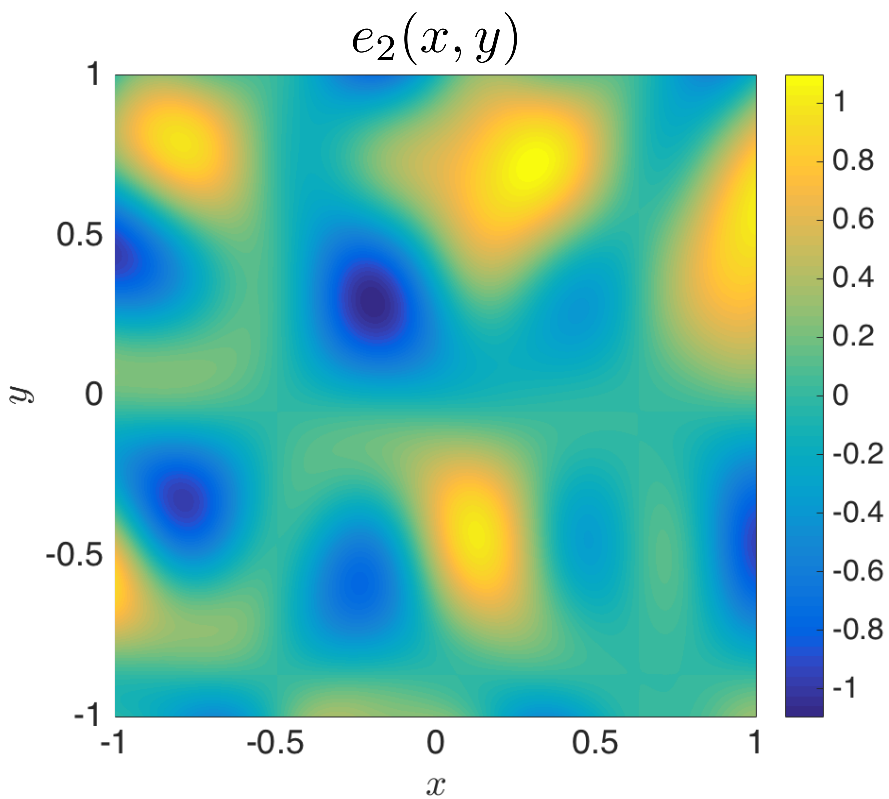
$$g_k(x,y) = d_k \; c_k(y) \; r_k(x)$$

Residual:

$$e_k(x,y) = e_{k-1}(x,y) - g_k(x,y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



$e_k(x, y_*) = 0$
$e_k(x_*, y) = 0$

k=2

pivot $d_k = 1/e_{k-1}(x_*, y_*)$

column slice $c_k(y) = e_{k-1}(x_*, y)$

row slice $r_k(x) = e_{k-1}(x, y_*)$

Rank 1 elimination function:

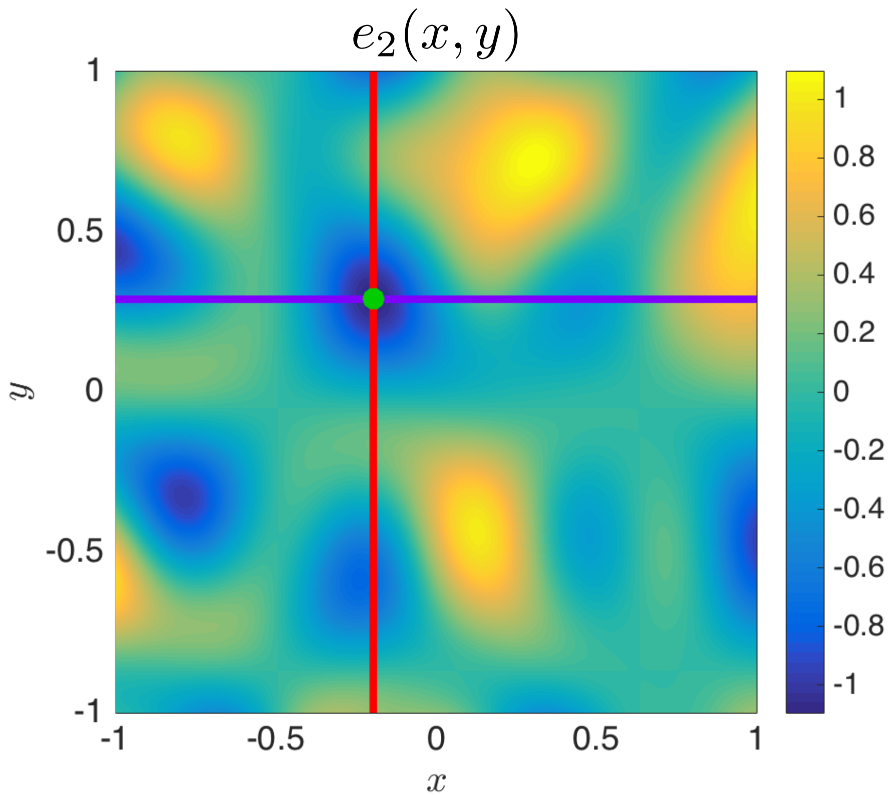
$$g_k(x,y) = d_k \; c_k(y) \; r_k(x)$$

Residual:

$$e_k(x,y) = e_{k-1}(x,y) - g_k(x,y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



$|e_{k-1}(x_*, y_*)| = \|e_{k-1}\|_\infty$

k=3

- pivot $d_k = 1/e_{k-1}(x_*, y_*)$
- column slice $c_k(y) = e_{k-1}(x_*, y)$
- row slice $r_k(x) = e_{k-1}(x, y_*)$

Rank 1 elimination function:

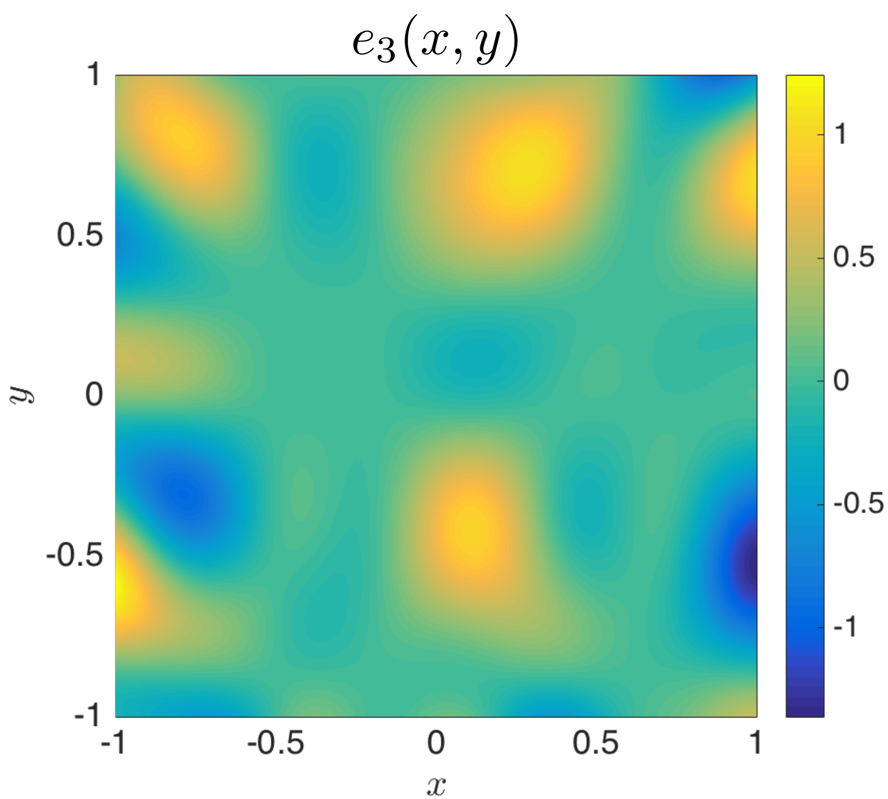
$$g_k(x,y) = d_k \; c_k(y) \; r_k(x)$$

Residual:

$$e_k(x,y) = e_{k-1}(x,y) - g_k(x,y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



$$e_k(x, y_*) = 0$$
$$e_k(x_*, y) = 0$$

$k=3$

pivot $d_k = 1/e_{k-1}(x_*, y_*)$

column slice $c_k(y) = e_{k-1}(x_*, y)$

row slice $r_k(x) = e_{k-1}(x, y_*)$

Rank 1 elimination function:

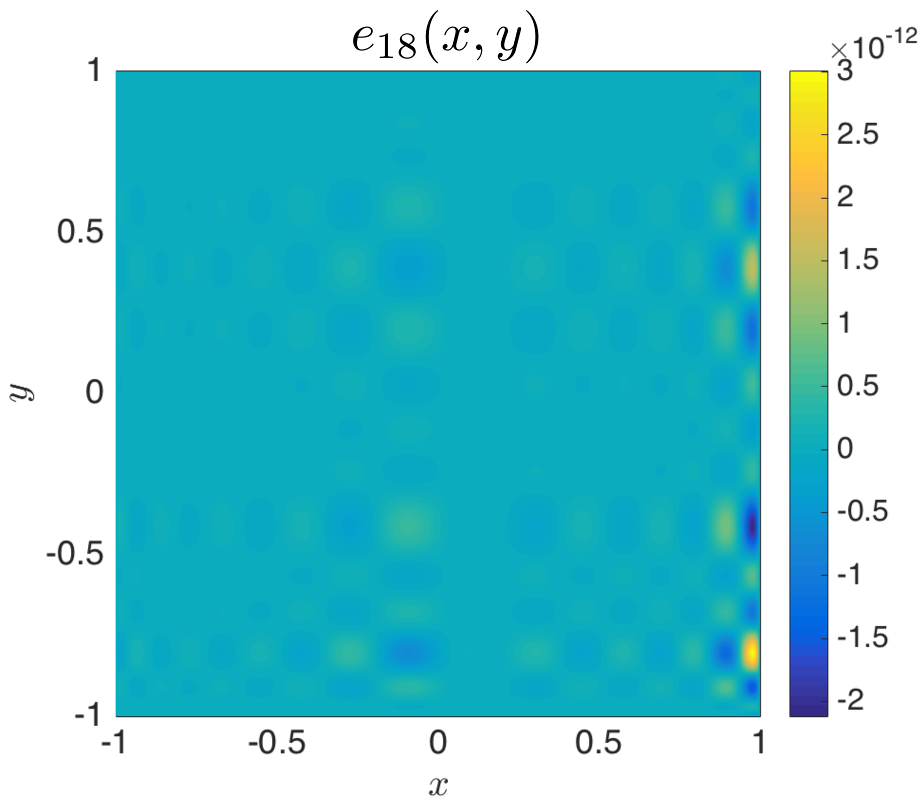
$$g_k(x,y) = d_k\ c_k(y)\ r_k(x)$$

Residual:

$$e_k(x,y) = e_{k-1}(x,y) - g_k(x,y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



$$e_k(x, y_*) = 0$$
$$e_k(x_*, y) = 0$$

Repeat to k=18

pivot $d_k = 1/e_{k-1}(x_*, y_*)$

column slice $c_k(y) = e_{k-1}(x_*, y)$

row slice $r_k(x) = e_{k-1}(x, y_*)$

Rank 1 elimination function:

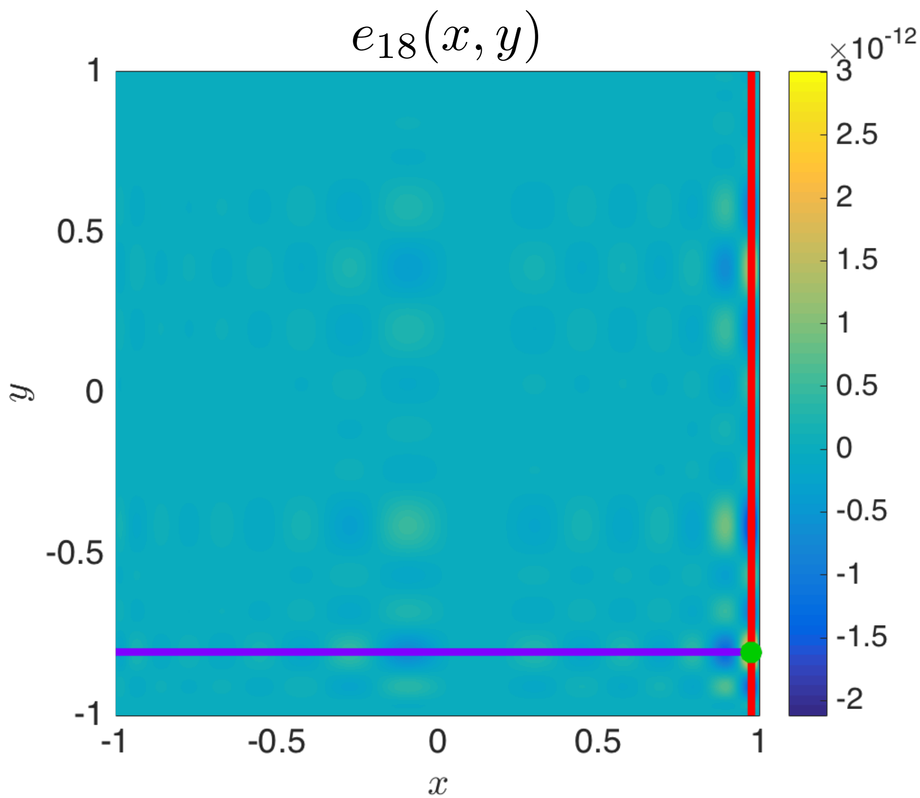
$$g_k(x,y) = d_k \; c_k(y) \; r_k(x)$$

Residual:

$$e_k(x,y) = e_{k-1}(x,y) - g_k(x,y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



$|e_{k-1}(x_*,y_*)| = \|e_{k-1}\|_\infty$

k=19

pivot $d_k = 1/e_{k-1}(x_*,y_*)$

column slice $c_k(y) = e_{k-1}(x_*,y)$

row slice $r_k(x) = e_{k-1}(x,y_*)$

Rank 1 elimination function:

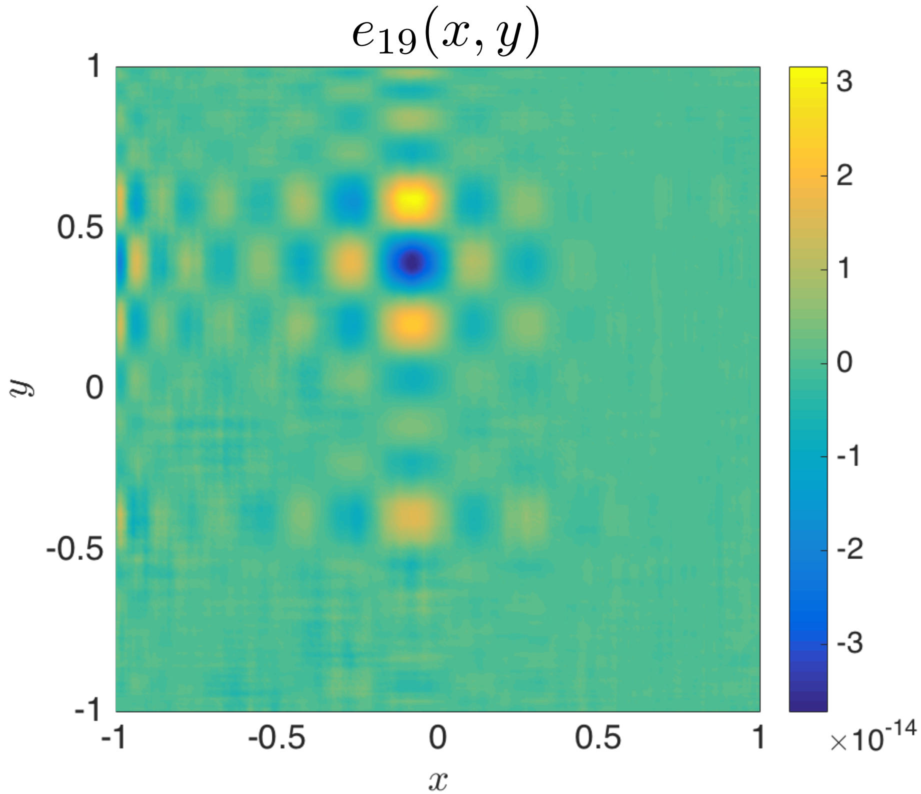
$$g_k(x,y) = d_k\ c_k(y)\ r_k(x)$$

Residual:

$$e_k(x,y) = e_{k-1}(x,y) - g_k(x,y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



$e_k(x, y_*) = 0$
$e_k(x_*, y) = 0$

k=19

- pivot $d_k = 1/e_{k-1}(x_*, y_*)$
- column slice $c_k(y) = e_{k-1}(x_*, y)$
- row slice $r_k(x) = e_{k-1}(x, y_*)$

Rank 1 elimination function:

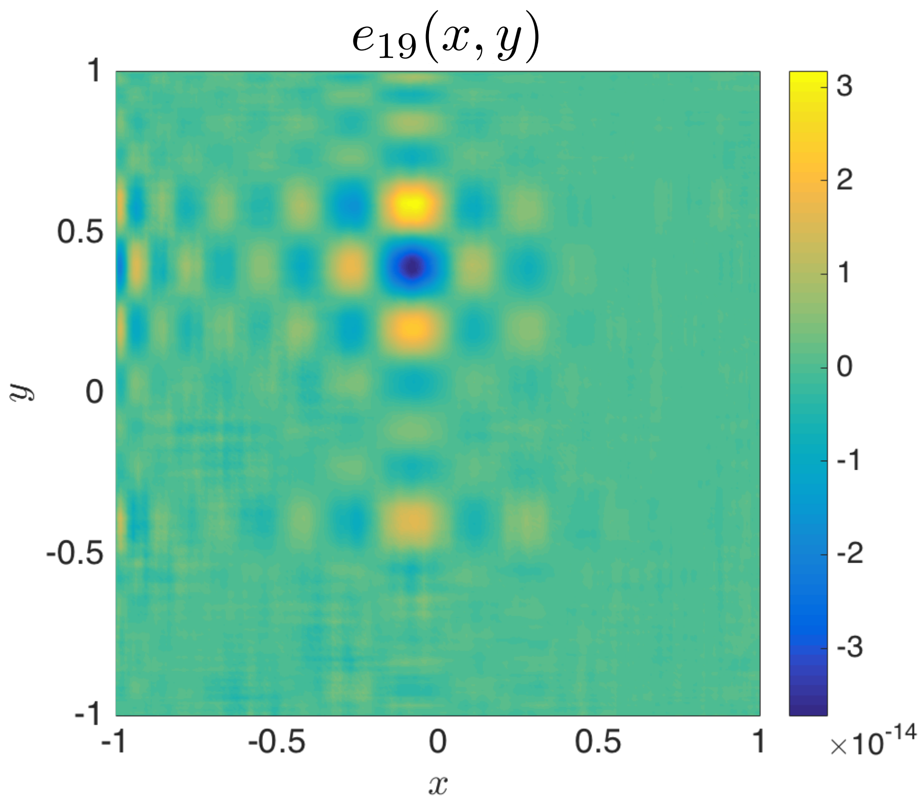
$$g_k(x, y) = d_k \; c_k(y) \; r_k(x)$$

Residual:

$$e_k(x, y) = e_{k-1}(x, y) - g_k(x, y)$$

Gaussian elimination for functions: example

- Continuous analogue of Gaussian elimination with complete pivoting.



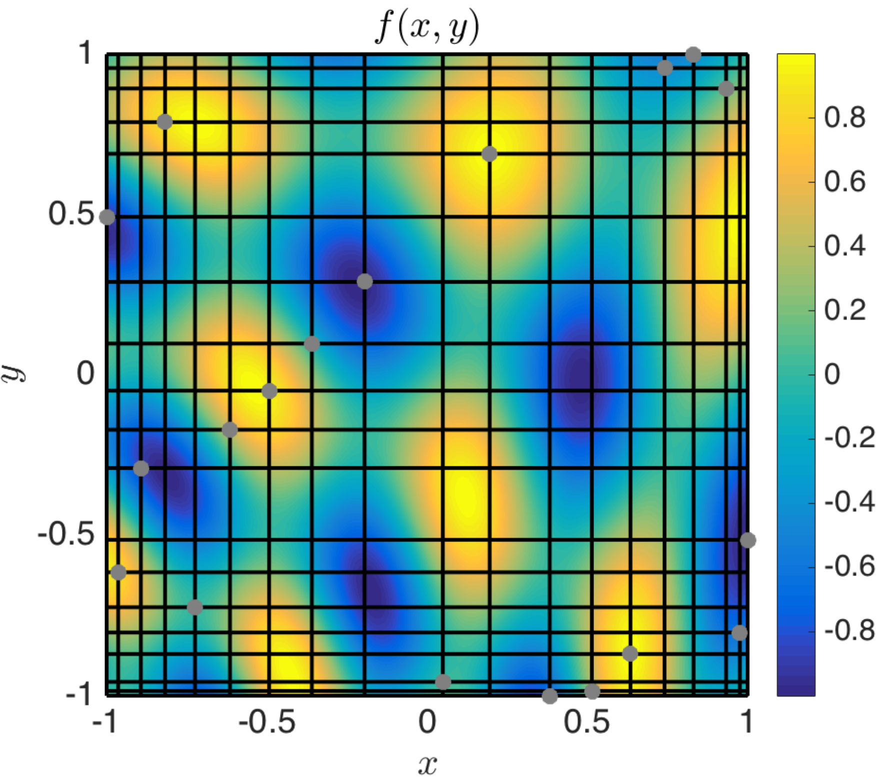
k=19

A rank 19 approximation with a max-norm error of $O(10^{-14})$ has now been obtained:

$$f(x, y) \approx \sum_{k=1}^{19} d_k c_k(y) r_k(x)$$

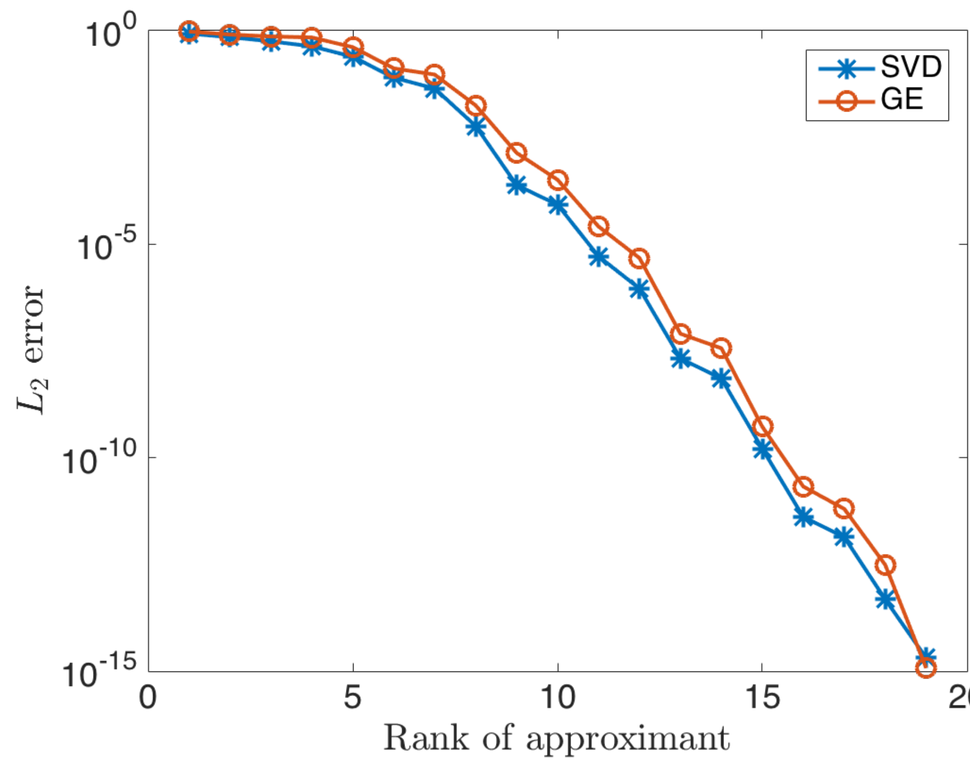
Gaussian elimination for functions: example

"Skeleton" showing where f is sampled



$$f(x, y) = \cos(3(x-1)(y-2))\sin(\pi(x-y))$$
$$\approx \sum_{k=1}^{19} d_k c_k(y) r_k(x)$$

Comparison to the truncated SVD of f

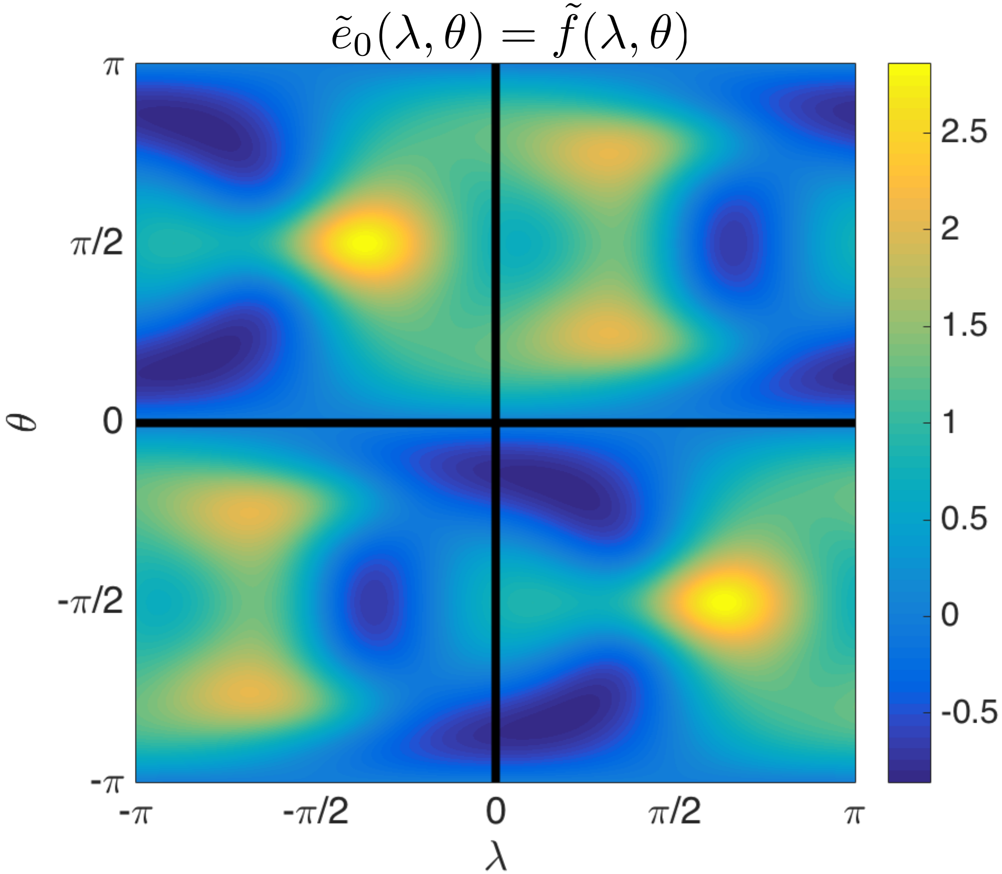


The SVD gives the optimal rank K approximation to f.

- Algorithmic details and convergence theory are given in Townsend & Trefethen (2013, & 2015)

Gaussian elimination for functions: sphere example

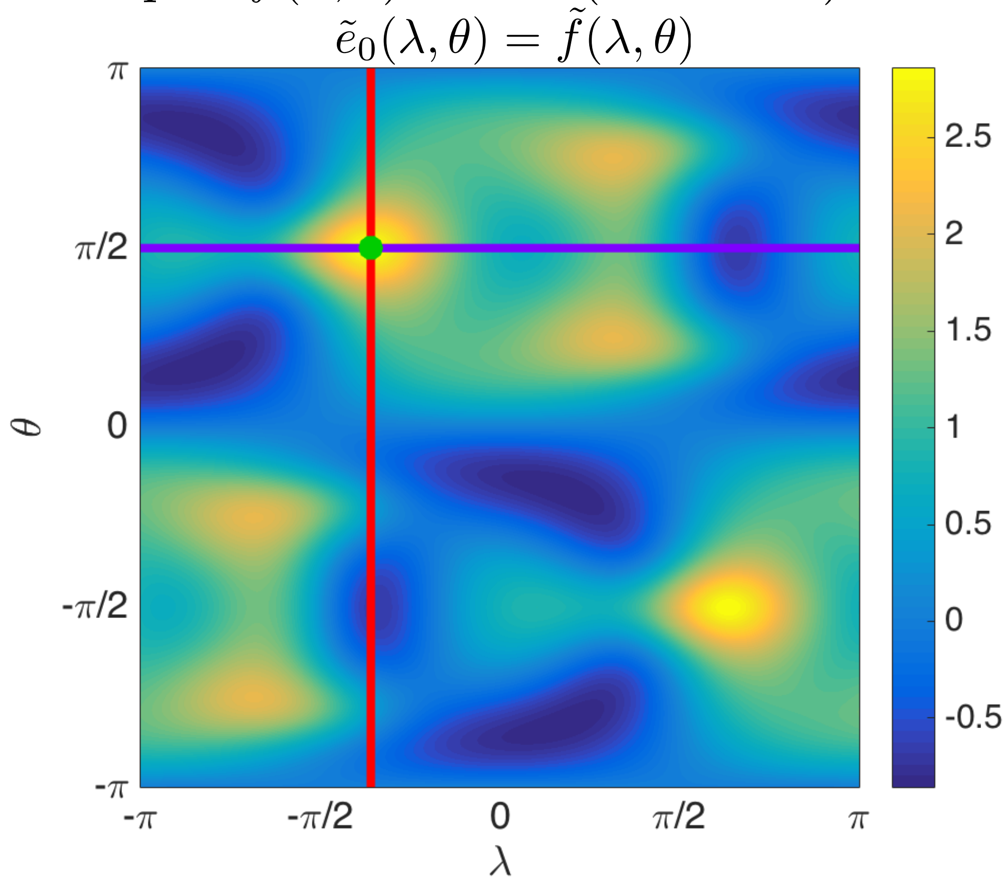
Example: $f(\lambda,\theta) = \tanh(1-\cos^2\theta)e^{\sin\lambda\sin\theta(5\cos^2\theta-1)} + \sin(\pi(\cos\lambda\sin\theta))$



Note structure from BMC extension

Gaussian elimination for functions: sphere example

Example: $f(\lambda,\theta) = \tanh(1-\cos^2\theta)e^{\sin\lambda\sin\theta(5\cos^2\theta-1)} + \sin(\pi(\cos\lambda\sin\theta)$



$k=1$

pivot $d_k = 1/\tilde{e}_{k-1}(\lambda_*,\theta_*)$

column slice $c_k(\theta) = \tilde{e}_{k-1}(\lambda_*,\theta)$

row slice $r_k(\lambda) = \tilde{e}_{k-1}(\lambda,\theta_*)$

Rank 1 elimination function:

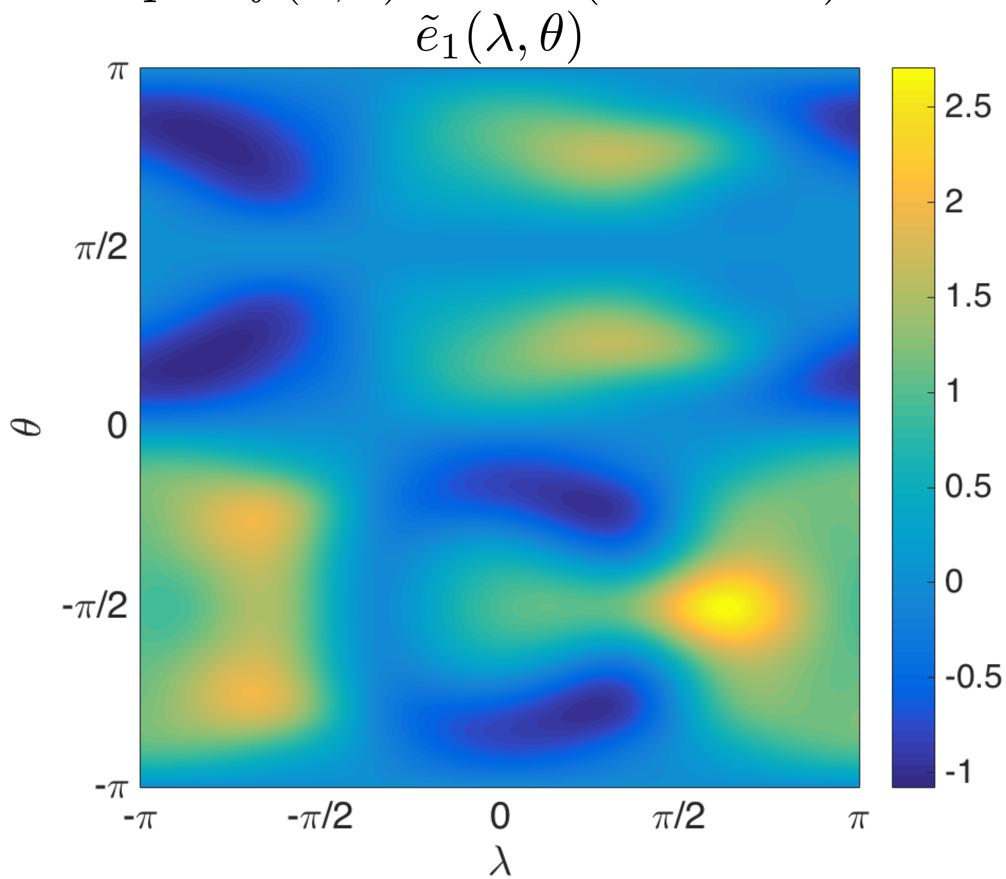
$$\tilde{g}_k(\lambda,\theta) = d_k\ c_k(\theta)\ r_k(\lambda)$$

Residual:

$$\tilde{e}_k(\lambda,\theta) = \tilde{e}_{k-1}(\lambda,\theta) - \tilde{g}_k(\lambda,\theta)$$

Gaussian elimination for functions: sphere example

Example: $f(\lambda,\theta) = \tanh(1-\cos^2\theta)e^{\sin\lambda\sin\theta(5\cos^2\theta-1)} + \sin(\pi(\cos\lambda\sin\theta)$



$k{=}1$

pivot $d_k = 1/\tilde{e}_{k-1}(\lambda_*,\theta_*)$

column slice $c_k(\theta) = \tilde{e}_{k-1}(\lambda_*,\theta)$

row slice $r_k(\lambda) = \tilde{e}_{k-1}(\lambda,\theta_*)$

Rank 1 elimination function:

$$\tilde{g}_k(\lambda,\theta) = d_k\ c_k(\theta)\ r_k(\lambda)$$

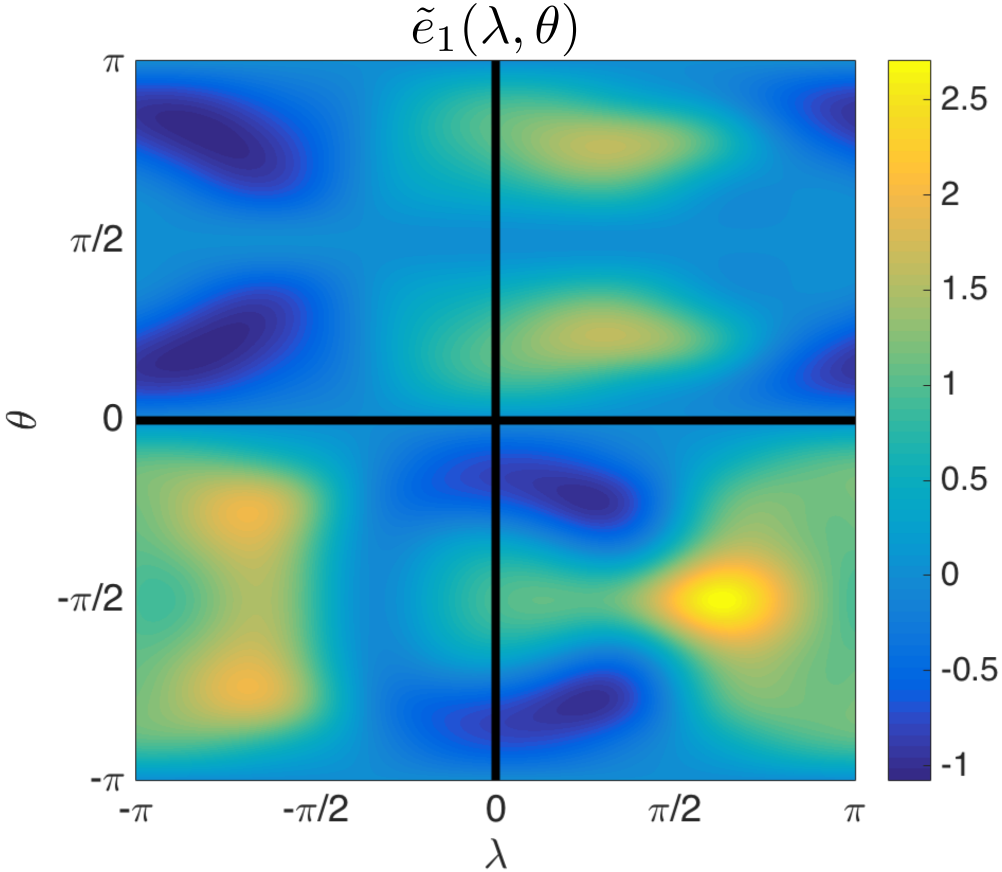
Residual:

$$\tilde{e}_k(\lambda,\theta) = \tilde{e}_{k-1}(\lambda,\theta) - \tilde{g}_k(\lambda,\theta)$$

The BMC structure has been lost!

Gaussian elimination for functions: sphere example

Example: $f(\lambda,\theta) = \tanh(1-\cos^2\theta)e^{\sin\lambda\sin\theta(5\cos^2\theta-1)} + \sin(\pi(\cos\lambda\sin\theta)$



k=1

pivot $d_k = 1/\tilde{e}_{k-1}(\lambda_*,\theta_*)$

column slice $c_k(\theta) = \tilde{e}_{k-1}(\lambda_*,\theta)$

row slice $r_k(\lambda) = \tilde{e}_{k-1}(\lambda,\theta_*)$

Rank 1 elimination function:

$$\tilde{g}_k(\lambda,\theta) = d_k\ c_k(\theta)\ r_k(\lambda)$$

Residual:

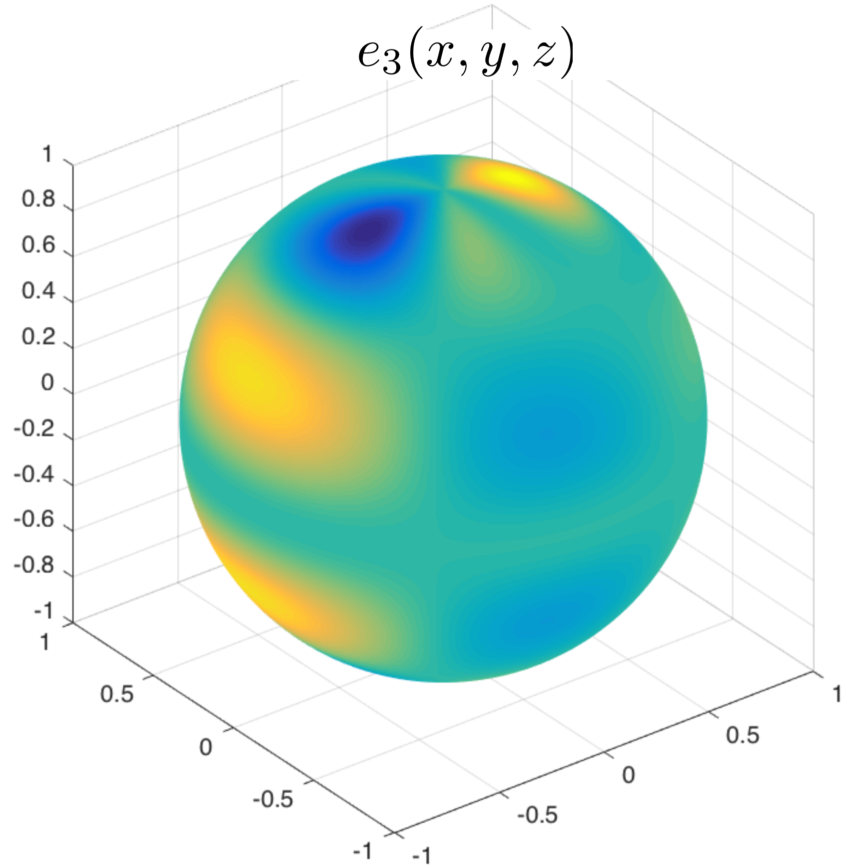
$$\tilde{e}_k(\lambda,\theta) = \tilde{e}_{k-1}(\lambda,\theta) - \tilde{g}_k(\lambda,\theta)$$

The residual is not smooth over the poles

Gaussian elimination for functions: sphere example

Example: $f(\lambda, \theta) = \tanh(1 - \cos^2\theta)e^{\sin\lambda\sin\theta(5\cos^2\theta - 1)} + \sin(\pi(\cos\lambda\sin\theta)$

Plot of the residual for k=3 on the sphere

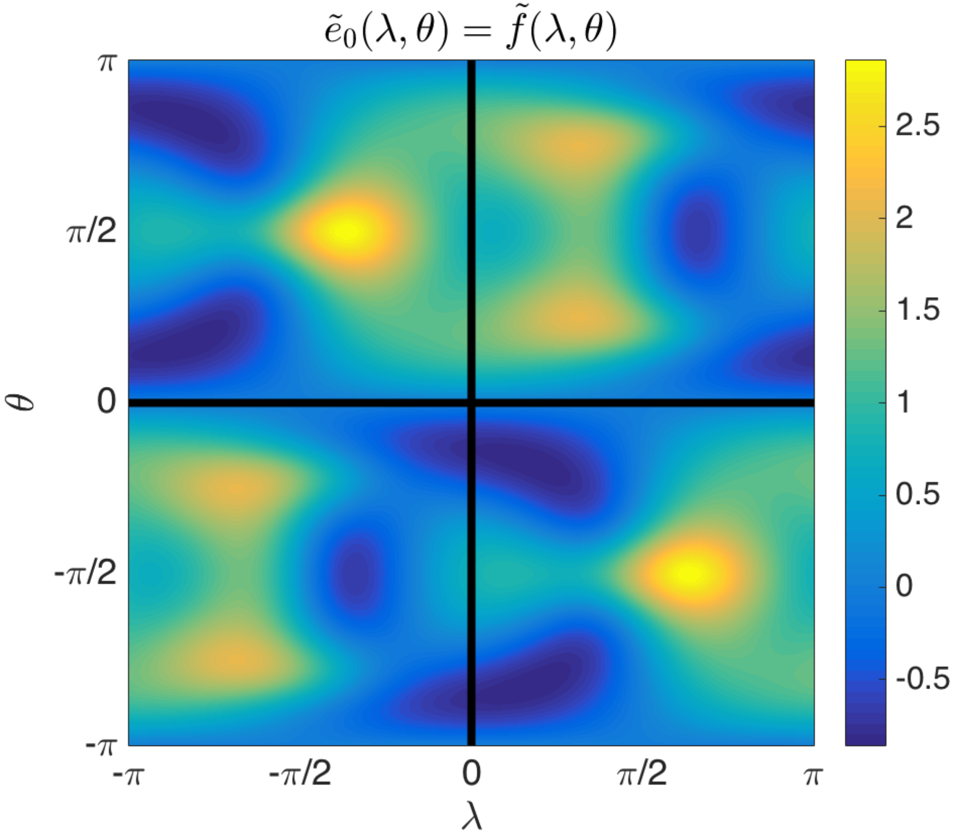


- What is wrong with this approach?

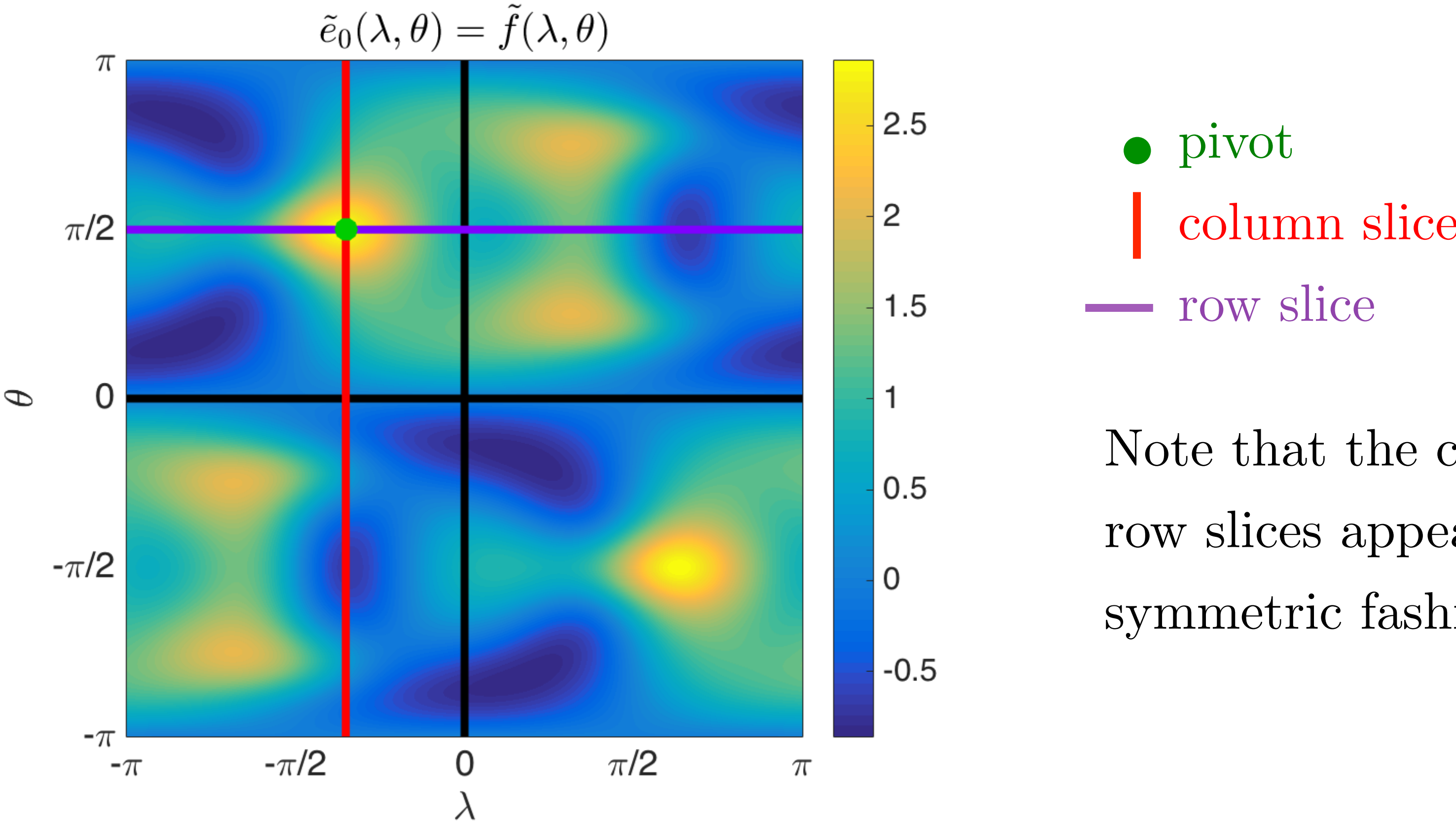
Residual step: $\tilde{e}_k(\lambda, \theta) = \tilde{e}_{k-1}(\lambda, \theta) - \tilde{g}_k(\lambda, \theta)$

To preserve the BMC structure, $\tilde{g}_k(\lambda, \theta)$, must also be a BMC function

Gaussian elimination that preserves BMC structure

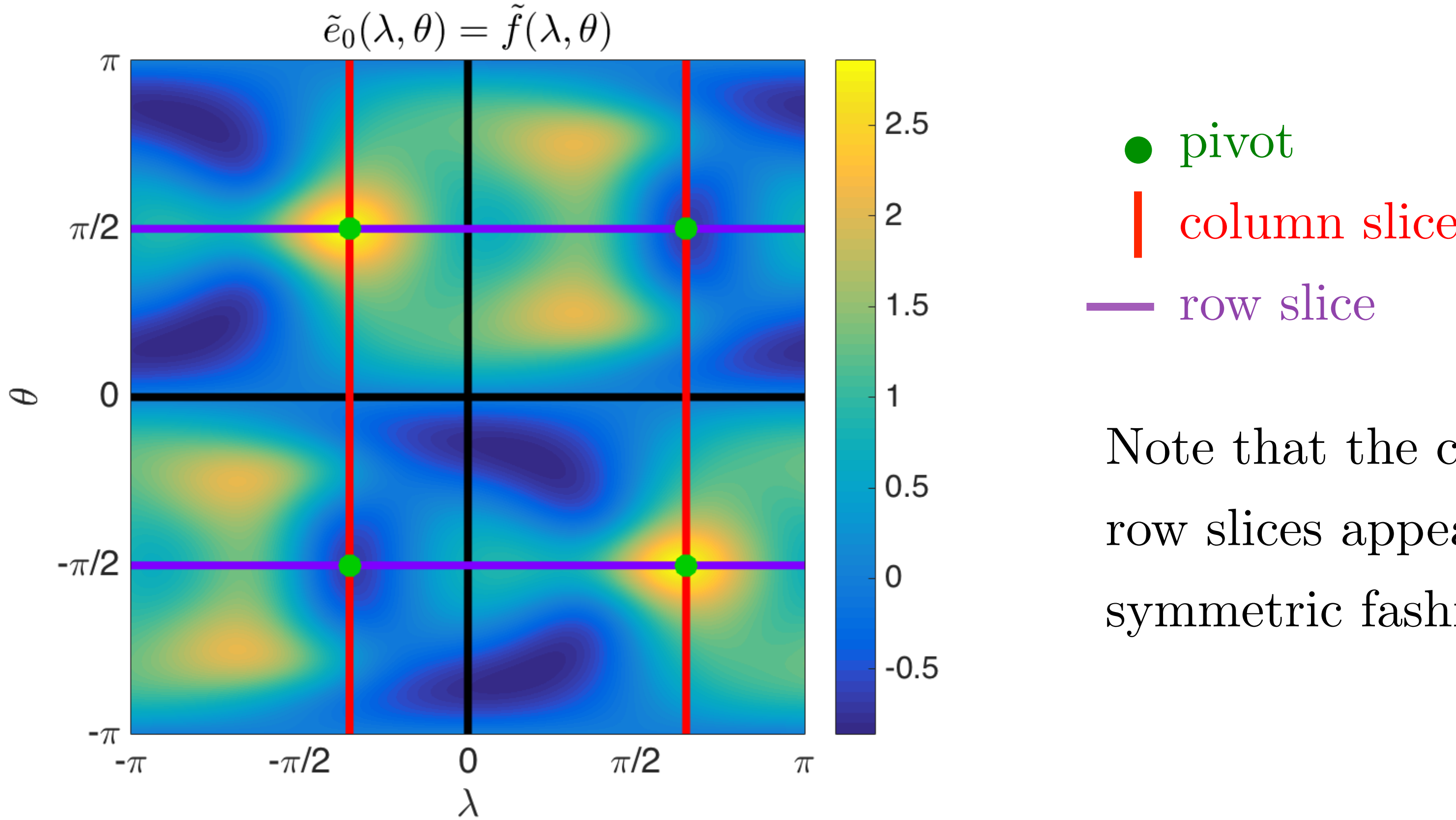


Gaussian elimination that preserves BMC structure



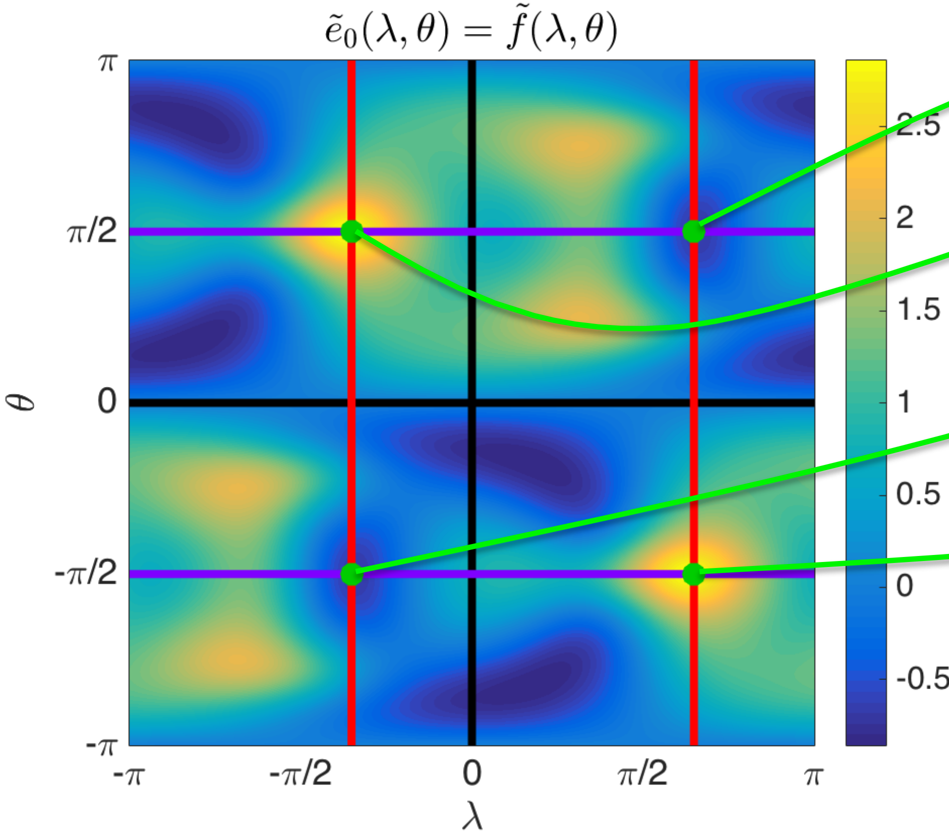
Note that the column and row slices appear in a symmetric fashion.

Gaussian elimination that preserves BMC structure



Note that the column and row slices appear in a symmetric fashion.

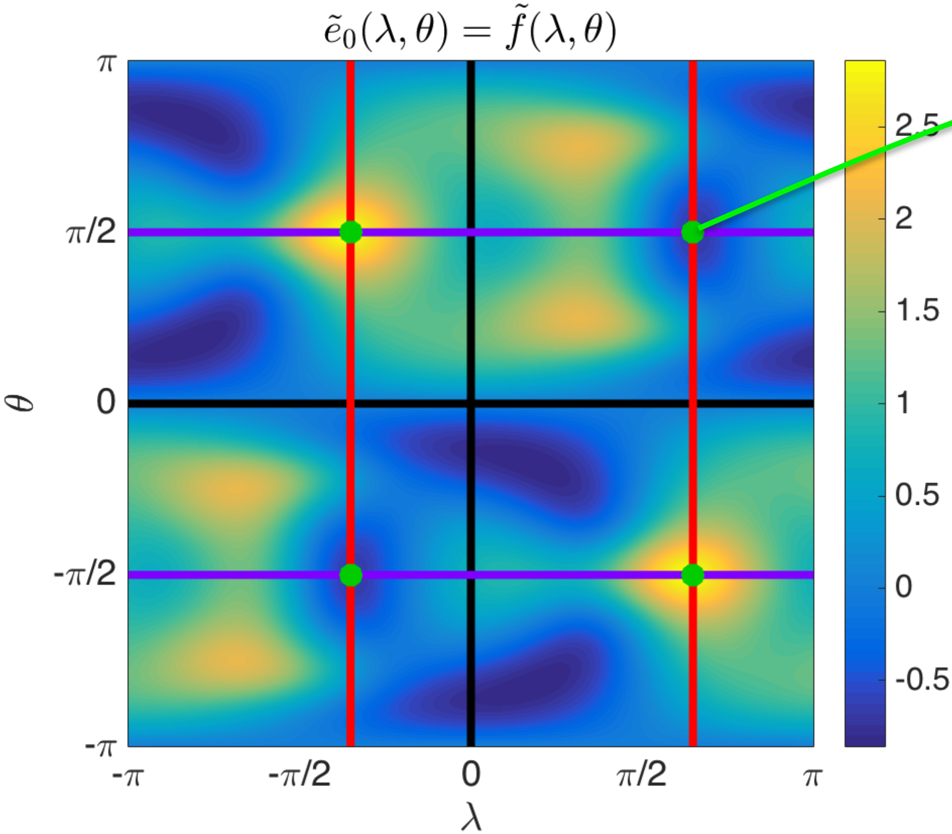
Gaussian elimination that preserves BMC structure



Define

$$M_1 = \begin{bmatrix} \tilde{e}_0(\lambda_* - \pi, \theta_*) & \tilde{e}_0(\lambda_*, \theta_*) \\ \tilde{e}_0(\lambda_*, \theta_*) & \tilde{e}_0(\lambda_* - \pi, \theta_*) \end{bmatrix}$$

Gaussian elimination that preserves BMC structure



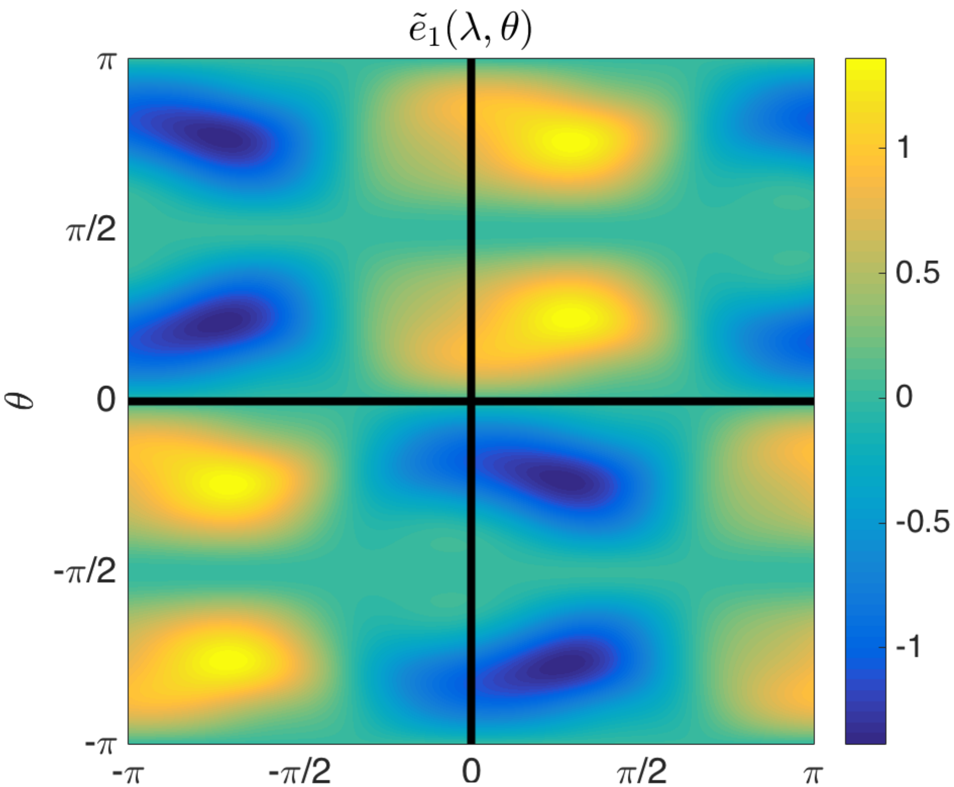
Define

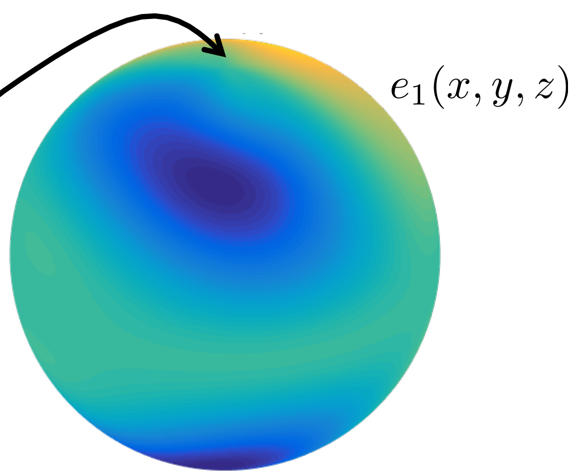
$$M_1 = \begin{bmatrix} \tilde{e}_0(\lambda_* - \pi, \theta_*) & \tilde{e}_0(\lambda_*, \theta_*) \\ \tilde{e}_0(\lambda_*, \theta_*) & \tilde{e}_0(\lambda_* - \pi, \theta_*) \end{bmatrix}$$

$c_1(\theta) = \tilde{e}_0(\lambda_*, \theta)$ and $r_1(\lambda) = \tilde{e}_0(\lambda, \theta_*)$

$$\boxed{\tilde{g}_1(\lambda,\theta) = \begin{bmatrix} c_1(-\theta) & c_1(\theta) \end{bmatrix} M_1^{-1} \begin{bmatrix} r_1(\lambda) \\ r_1(\lambda - \pi) \end{bmatrix}}$$

Gaussian elimination that preserves BMC structure





Smooth at the poles.

Define

$$M_1 = \begin{bmatrix} \tilde{e}_0(\lambda_* - \pi, \theta_*) & \tilde{e}_0(\lambda_*, \theta_*) \\ \tilde{e}_0(\lambda_*, \theta_*) & \tilde{e}_0(\lambda_* - \pi, \theta_*) \end{bmatrix}$$

$c_1(\theta) = \tilde{e}_0(\lambda_*, \theta)$ and $r_1(\lambda) = \tilde{e}_0(\lambda, \theta_*)$

$$\tilde{g}_1(\lambda,\theta) =$$

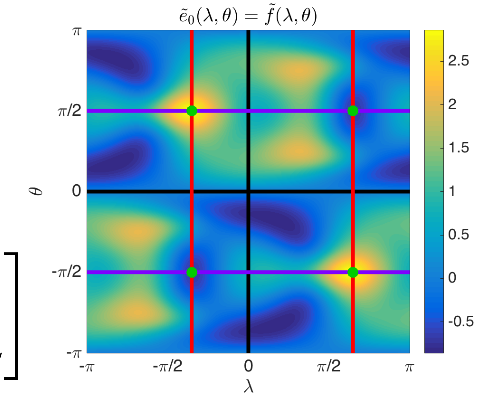
$$\begin{bmatrix} c_1(-\theta) & c_1(\theta) \end{bmatrix} M_1^{-1} \begin{bmatrix} r_1(\lambda) \\ r_1(\lambda - \pi) \end{bmatrix}$$

$$\boxed{\tilde{e}_1(\lambda,\theta) = \tilde{e}_0(\lambda,\theta) - \tilde{g}_1(\lambda,\theta)}$$

BMC function!

Repeat this process on $\tilde{e}_k(\lambda,\theta)$, $k = 1, 2, \ldots$

Some details on the algorithm



- How do we choose the pivot matrices?

  Find $(\lambda_*, \theta_*) \in [0, \pi]^2$ such that

$$M_k = \begin{bmatrix} \tilde{e}_{k-1}(\lambda_* - \pi, \theta_*) & \tilde{e}_{k-1}(\lambda_*, \theta_*) \\ \tilde{e}_{k-1}(\lambda_*, \theta_*) & \tilde{e}_{k-1}(\lambda_* - \pi, \theta_*) \end{bmatrix} = \begin{bmatrix} a & b \\ b & a \end{bmatrix}$$

  has maximal singular value $\sigma_1(M_k) = \max\{|a+b|, |a-b|\}$.

  $\Longrightarrow$ This is the 2×2 pivot analogue of complete pivoting.

- M_k may be singular or numerically ill-conditioned.

  Solution: Replace M_k^{-1} by ϵ-pseudoinverse: $M_k^{-1} \longrightarrow M_k^{+\epsilon}$

- Algorithm does not produce decoupled scalars d_k, and columns and rows $c_k(\theta)$ and $r_k(\lambda)$ such that $\tilde{f}(\lambda, \theta) \approx \sum_{k=1}^{K} d_k c_k(\theta) r_k(\lambda)$

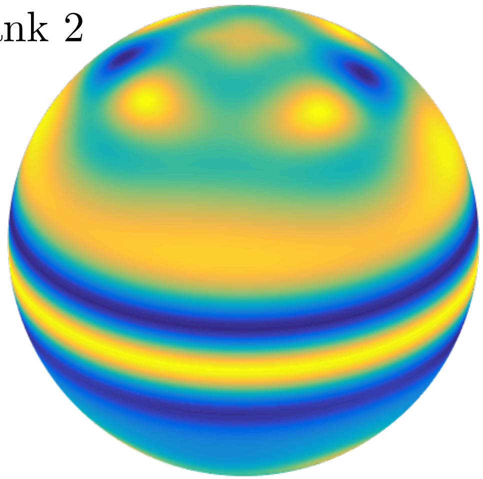
  Solution: Use spectral decomposition of each M_k to decouple the rows and columns from $\tilde{g}_k$.

Why Preserving BMC structure is important

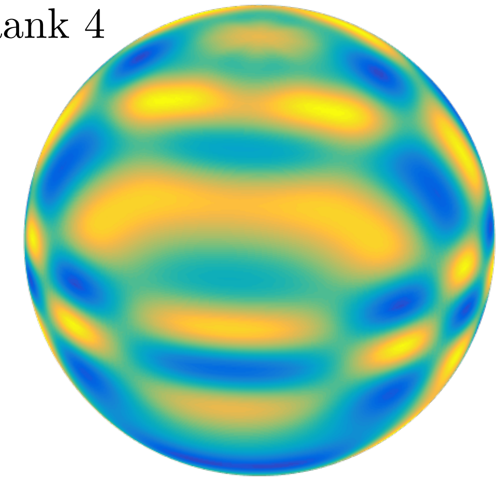
Example: $f(x, y, z) = \cos(1 + 2\pi(x + y) + 5\sin(\pi z))$

<u>GE preserving BMC structure:</u> no pole singularity!

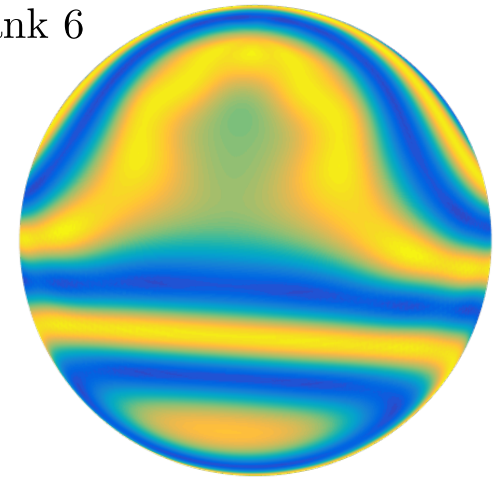
Rank 2



Rank 4

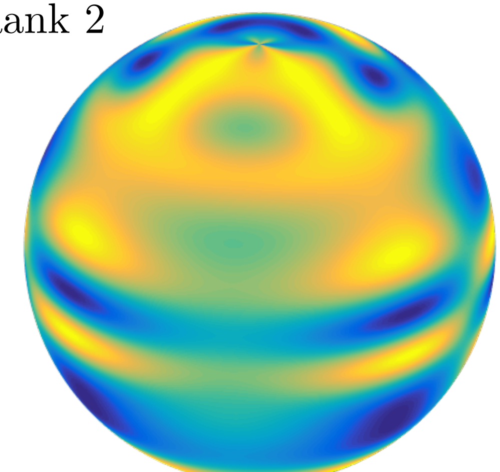


Rank 6

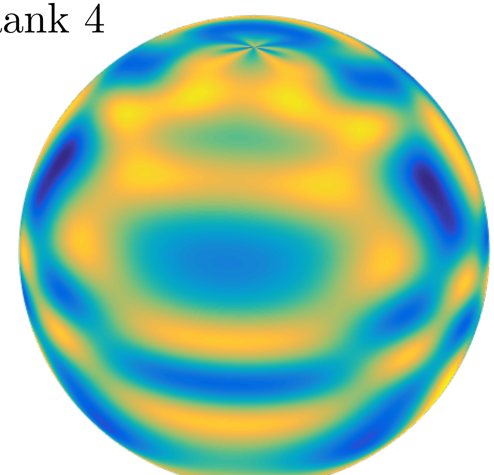


<u>Standard GE:</u> pole singularity

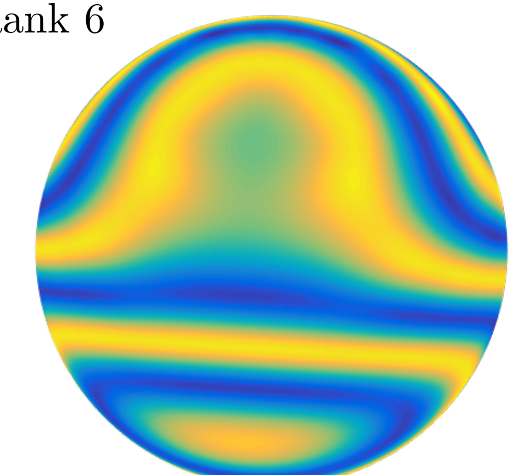
Rank 2



Rank 4

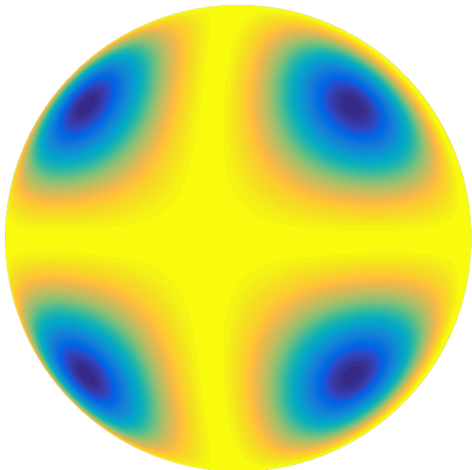


Rank 6

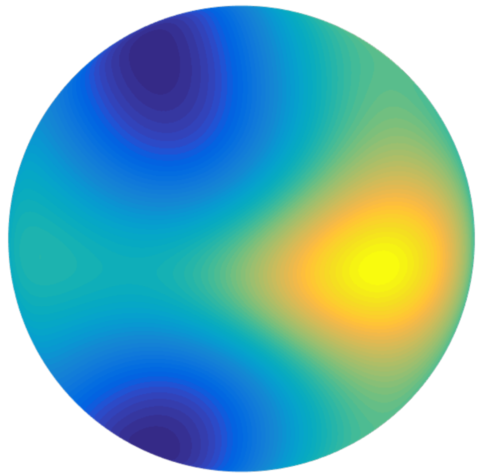


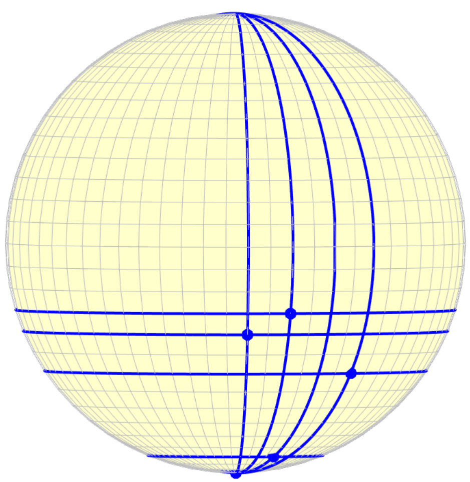
Example “skeletons” from low rank approximations

$\cos(xyz)$

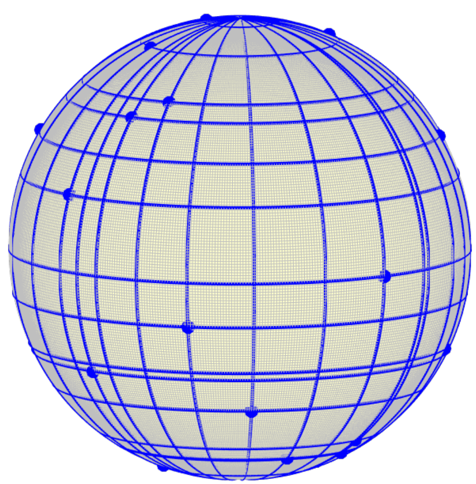


$\tanh(1-z^2)e^{(5z^2-1)y} + \sin(\pi x)$





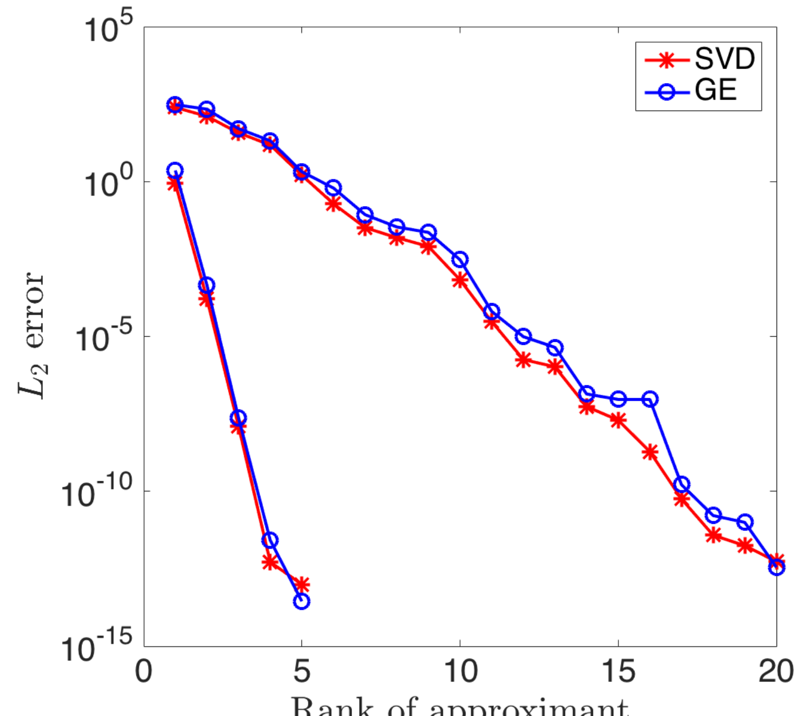
Rank K=5



Rank K=20

GE algorithm ameliorates oversampling at the poles.

SVD is optimal, but GE is near-optimal



Approximation theory for the GE algorithm

Theorem *Let* $f : \mathbb{S}^2 \to \mathbb{R}$ *and* $\tilde{f} : [-\pi, \pi]^2 \to \mathbb{R}$ *be its BMC extension. Then the following holds for our Gaussian elimination algorithm:*

1) *Structure preserving. BMC structure is preserved at every step and each rank one update is smooth on the sphere.*

Proofs and more details in Townsend, Wilber, & W, 2016a,b.

Approximation theory for the GE algorithm

Theorem *Let $f : \mathbb{S}^2 \rightarrow \mathbb{R}$ and $\tilde{f} : [-\pi, \pi]^2 \rightarrow \mathbb{R}$ be its BMC extension. Then the following holds for our Gaussian elimination algorithm:*

1) *Structure preserving. BMC structure is preserved at every step and each rank one update is smooth on the sphere.*

2) *Exact recovery. All band-limited functions on the sphere are exactly recovered.*

Proofs and more details in Townsend, Wilber, & W, 2016a,b.

Approximation theory for the GE algorithm

Theorem *Let $f : \mathbb{S}^2 \to \mathbb{R}$ and $\tilde{f} : [-\pi, \pi]^2 \to \mathbb{R}$ be its BMC extension. Then the following holds for our Gaussian elimination algorithm:*

1) *Structure preserving. BMC structure is preserved at every step and each rank one update is smooth on the sphere.*

2) *Exact recovery. All band-limited functions on the sphere are exactly recovered.*

3) *Convergence. If $\tilde{f}(\lambda, \theta)$ is analytic in both variables separately in a sufficiently large region, then $\|\tilde{e}_k\|_\infty \to 0$ geometrically as $k \to \infty$.*

Proofs and more details in Townsend, Wilber, & W, 2016a,b.

Approximation theory for the GE algorithm: sphere

Theorem *Let $f : \mathbb{S}^2 \to \mathbb{R}$ and $\tilde{f} : [-\pi, \pi]^2 \to \mathbb{R}$ be its BMC extension. Then the following holds for our Gaussian elimination algorithm:*

1) *Structure preserving. BMC structure is preserved at every step and each rank one update is smooth on the sphere.*

2) *Exact recovery. All band-limited functions on the sphere are exactly recovered.*

3) *Convergence. If $\tilde{f}(\lambda, \theta)$ is analytic in both variables separately in a sufficiently large region, then $\|\tilde{e}_k\|_\infty \to 0$ geometrically as $k \to \infty$.*

4) *Symmetries. The low rank approximation to $\tilde{f}(\lambda, \theta)$ can be written as:*

$$\tilde{f} \approx \sum_{k=1}^{K^{\mathrm{e}}} d_k^{\mathrm{e}} \begin{bmatrix} c_k^{\mathrm{e}} \\ \mathtt{flip}(c_k^{\mathrm{e}}) \end{bmatrix} \begin{bmatrix} r_k^{\mathrm{e}} & r_k^{\mathrm{e}} \end{bmatrix} + \sum_{k=1}^{K^{\mathrm{o}}} d_k^{\mathrm{o}} \begin{bmatrix} c_k^{\mathrm{o}} \\ -\mathtt{flip}(c_k^{\mathrm{o}}) \end{bmatrix} \begin{bmatrix} r_k^{\mathrm{o}} & -r_k^{\mathrm{o}} \end{bmatrix}$$

where $c_k^{\mathrm{e}} / c_k^{\mathrm{o}}$ are even/odd and $r_k^{\mathrm{e}} / r_k^{\mathrm{o}}$ are π-periodic/π-antiperiodic over $[0, \pi]$.

Proofs and more details in Townsend, Wilber, & W, 2016a,b.

Approximation theory for the GE algorithm: disk

Theorem *Let $f : \mathbb{D} \to \mathbb{R}$ and $\tilde{f} : [-\pi, \pi] \times [-1, 1] \to \mathbb{R}$ be its BMC extension. Then the following holds for our Gaussian elimination algorithm:*

1) *Structure preserving. BMC structure is preserved at every step and each rank one update is smooth on the sphere.*

2) *Exact recovery. All band-limited functions on the disk are exactly recovered.*

3) *Convergence. If $\tilde{f}(\theta, \rho)$ is analytic in both variables separately in a sufficiently large region, then $\|\tilde{e}_k\|_\infty \to 0$ geometrically as $k \to \infty$.*

4) *Symmetries. The low rank approximation to $\tilde{f}(\theta, \rho)$ preserves the even/odd symmetries of the disk.*

Proofs and more details in Townsend, Wilber, & W, 2016a,b.

Complexity of the GE algorithm

Theorem Complexity

If $\tilde{f} : [-\pi, \pi]^2 \to \mathbb{R}$ can be approximated to machine precision by a rank K function then the algorithm takes $\boxed{O(K^3 + K^2(m+n))}$ *operations where m and n are the maximum samples required to resolve the column and row samples, respectively, with a discrete Fourier series of these sizes.*

If $\tilde{f} : [-\pi, \pi] \times [-1, 1] \to \mathbb{R}$ can be approximated to machine precision by a rank K function then the algorithm takes $\boxed{O(K^3 + K^2(m+n))}$ *operations where m and n are the maximum samples required to resolve the column and row samples, respectively, with a discrete Fourier series and Chebyshev interpolant of these respective sizes.*

Part III: Operations in Spherefun and Diskfun

Separable operations: Spherefun example

Idea: 2-D separable operations can exploit 1-D technology

$$\tilde{f}(\lambda, \theta) \approx \sum_{k=1}^{K} d_k \underbrace{c_k(\theta)}_{2\pi\text{-periodic}} \underbrace{r_k(\lambda)}_{2\pi\text{-periodic}}$$

Integration: `sum2(f)`

$$\int_{\mathbb{S}^2} f(x,y,z) d\Omega \approx \sum_{k=1}^{K} d_k \int_0^{\pi} c_k(\theta) \sin\theta d\theta \int_{-\pi}^{\pi} r_k(\lambda) d\lambda,$$

Differentiation: `diff(f)`

$$\frac{\partial^{\mathrm{t}} f}{\partial x} \approx -\sum_{k=1}^{K} d_k \left(\frac{c_k(\theta)}{\sin\theta}\right) \left(\sin\lambda \frac{\partial r_k(\lambda)}{\partial \lambda}\right) + \sum_{k=1}^{K} d_k \left(\cos\theta \frac{\partial c_k(\theta)}{\partial \theta}\right) \left(\cos\lambda \, r_k(\lambda)\right)$$

Others include:

`f(x,y,z), sample(f), laplacian(f), grad(f), curl(f), coeffs2(f)`

Vector calculus: Spherefunv and Diskfunv

Idea: represent vector fields on the sphere or disk with respect to Cartesian coordinates:

$$\mathbf{u}(x,y,z) = \begin{bmatrix} u(x,y,z) & v(x,y,z) & w(x,y,z) \end{bmatrix} \quad (x,y,z) \in \mathbb{S}^2$$

$$\mathbf{u}(x,y) = \begin{bmatrix} u(x,y) & v(x,y) \end{bmatrix} \quad (x,y) \in \mathbb{D}$$

Each component is approximated using a low rank approximant and combined into a `spherefunv` or `diskfunv` object.

Example: `div(u)`

`Spherefunv:` $$\nabla_{\mathbb{S}^2} \cdot \mathbf{u} = \frac{\partial^{\mathrm{t}} u}{\partial x} + \frac{\partial^{\mathrm{t}} v}{\partial y} + \frac{\partial^{\mathrm{t}} w}{\partial z}$$

`Diskfunv:` $$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

Others operations for `spherefunv` and `diskfunv` objects:

`curl(u), vort(u), cross(u,v), normal(u), tangent(u), quiver(u)`

Fast Poisson solver for the sphere

We also have new fast, optimal complexity Poisson/Helmholtz solvers.

- Fourier/ultraspherical spectral methods using BMC extension
- Low rank representation exploited to compute spectral expansion coefficients of the right-hand side.

Poisson equation on the sphere:

$$\Delta_{\mathbb{S}^2} u = f, \quad \text{subject to: } \int_{\mathbb{S}^2} f \, d\Omega = 0$$

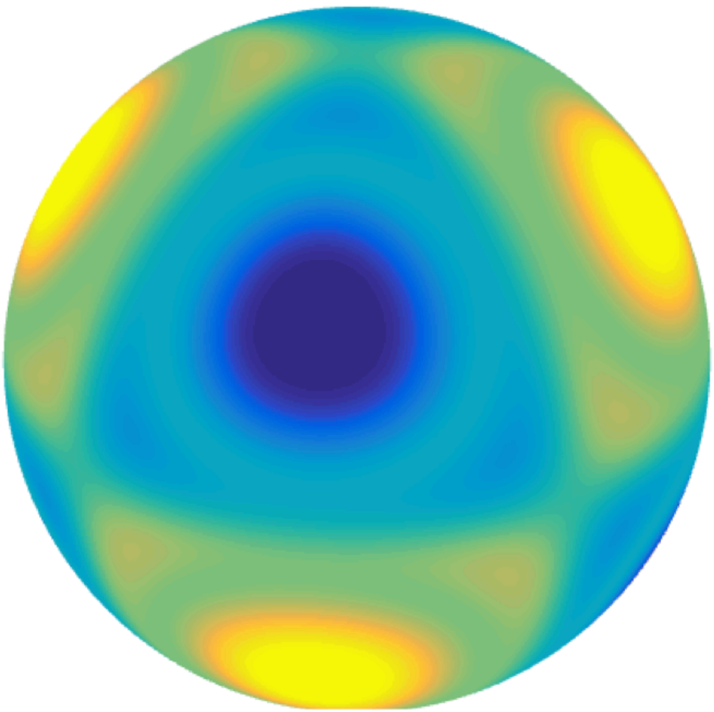
Poisson equation on the disk:

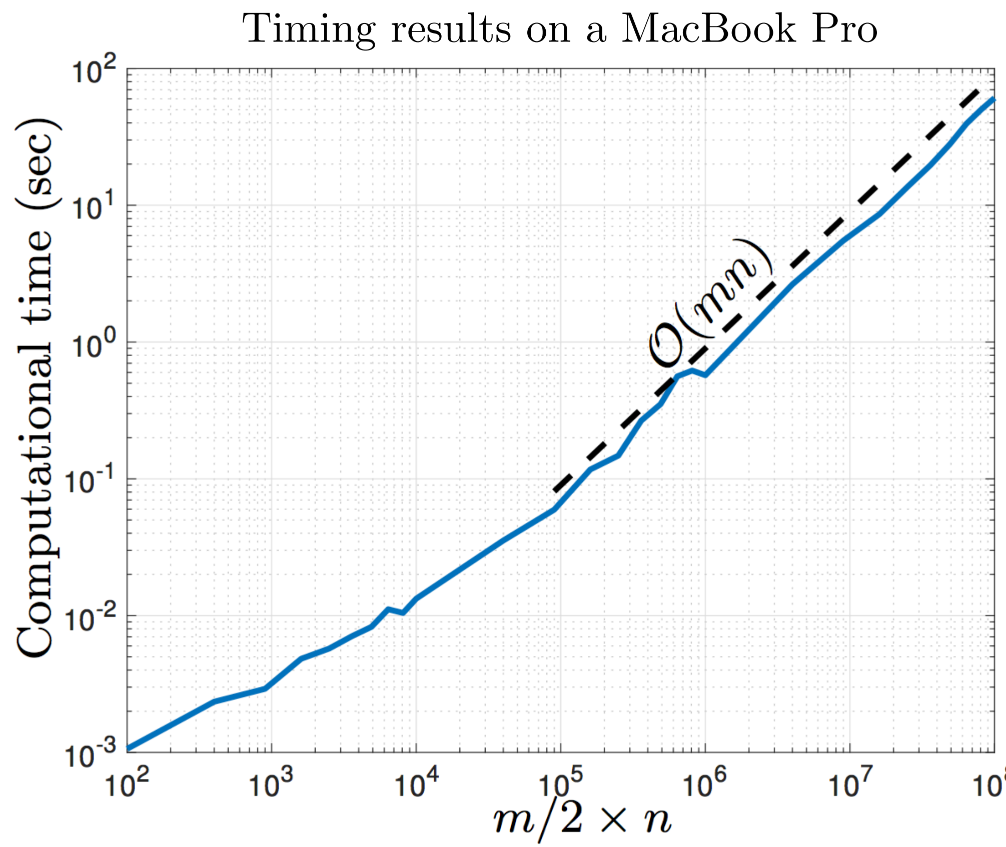
$$\Delta u = f, \quad \text{subject to: } u(\theta, 1) = g(\theta), \; -\pi \leq \theta \leq \pi$$

Fast Poisson solver for the sphere

Example: $\Delta_{\mathbb{S}^2} u = \sin(50xyz)$, $(x, y, z) \in \mathbb{S}^2$

$u(x, y, z)$

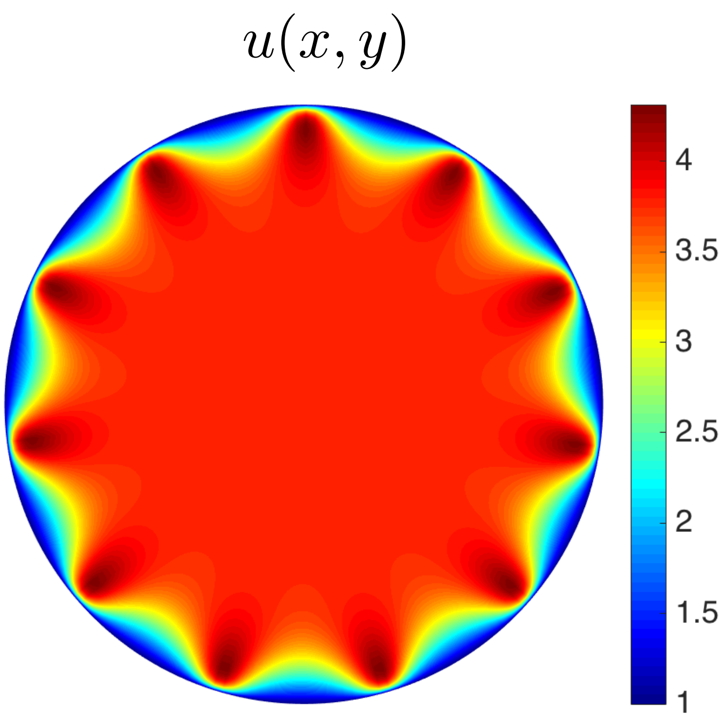


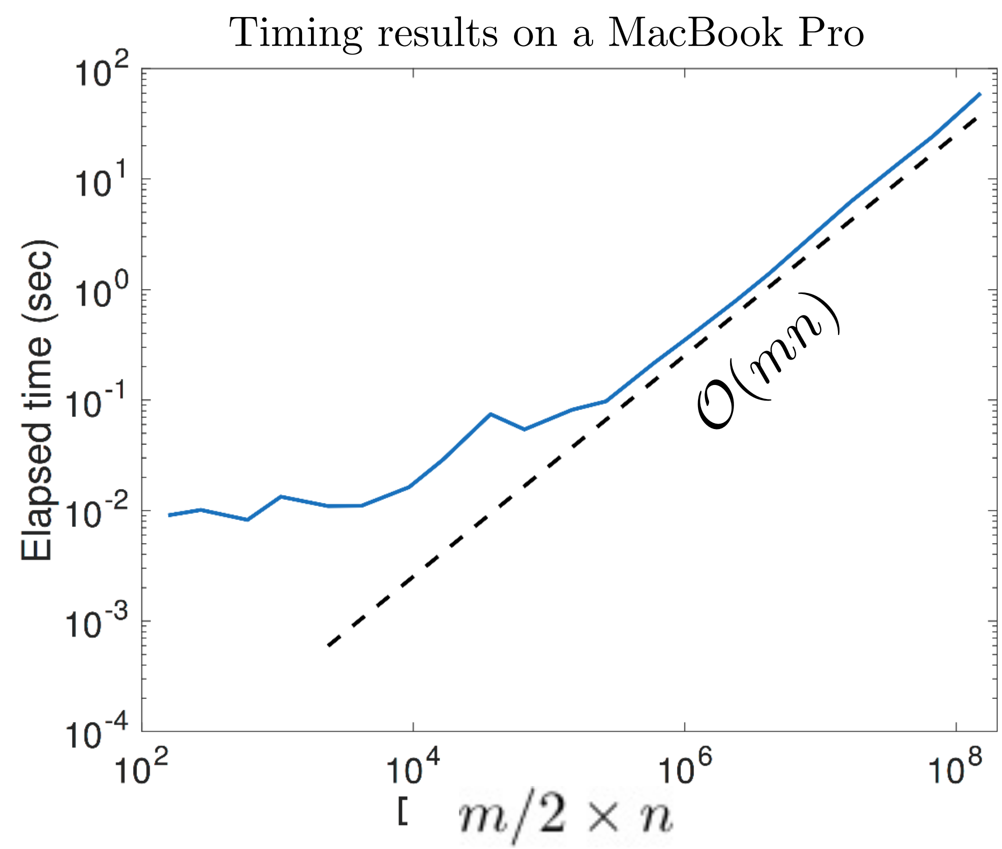



```
f = spherefun(@(x,y,z) sin(50*x.*y.*z));
u = spherefun.poisson(f,0,m,n);
```

Fast Poisson solver for the disk

Ex: $\Delta u = -e^{-40(\rho^2-1)^4}\sinh(5(1-\rho^{11}\cos(11(\theta-1/\sqrt{2})),\ u(\theta,1)=0$






```
g = chebfun(@(t) 0*t,[-pi, pi],'trig');
u = diskfun.poisson(f,g,m,n);
```

Application: spiral waves in excitable media

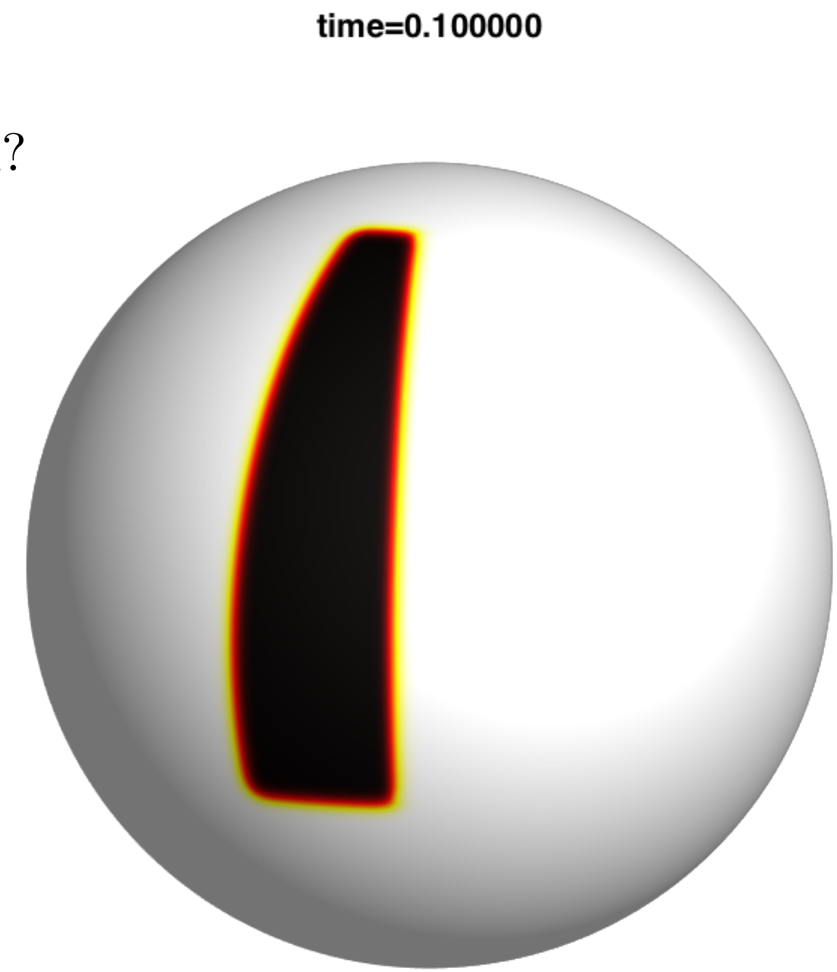
Barkley (1991) model (simplification of FitzHugh-Nagumo model)

$$\frac{\partial u}{\partial t} = \delta_u \Delta u + \frac{1}{\epsilon} u (1-u) \left(u - \frac{v+b}{a} \right)$$

$$\frac{\partial v}{\partial t} = \delta_v \Delta v + u - v$$

u=activator species
v=inhibitor species

How does geometry effect the wave pattern?



- Visualization of the u (activator) component
- Black=1, white = 0
- Uses 3$^{\text{rd}}$ order semi-implicit BDF scheme.
- Fast Helmholtz solver for implicit terms.

Conclusion

Spherefun & Diskfun: computing with functions on the sphere and disk

- Combines the Double Fourier Sphere method with a structure preserving Gaussian Elimination algorithm for functions.
  - ➢ Allows use of the FFT in both coordinate directions.

    Sphere: Fourier/Fourier   Disk: Fourier/Chebyshev
  - ➢ Ameliorates oversampling at the poles (sphere) and origin (disk)
  - ➢ Avoids numerical issues with computing derivatives at the poles.

- Includes a new, fast (optimal) spectral method for Poisson and Helmholtz equations.

- Code is now a part of Chebfun (www.chebfun.org).

Future:

- Fully low rank Poisson solver.
- Cylinder and Solid sphere.
- More development on the PDE side.

Grazie per la vostra attenzione.