Fourier Extensions: Algorithms and Applications

Roel Matthysen

April 14, 2016

Joint work with Daan Huybrechs
Department of Computer Science, KU Leuven

Goal

Given a function f as N data points on an equispaced grid, find an expansion in Fourier basis functions that

- converges fast
- is easily computable ($O(N)$ operations).

Classical Fourier interpolation (FFT) relies on periodicity for fast convergence.

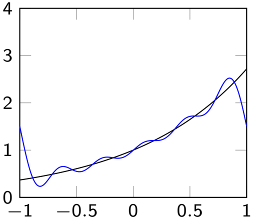
Otherwise the Gibbs phenomenon makes convergence very slow.

Goal

Given a function f as N data points on an equispaced grid, find an expansion in Fourier basis functions that

- converges fast
- is easily computable ($O(N)$ operations).

Classical Fourier interpolation (FFT) relies on periodicity for fast convergence.

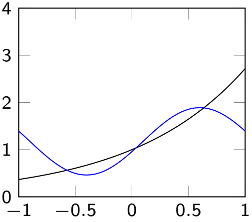


Otherwise the Gibbs phenomenon makes convergence very slow.

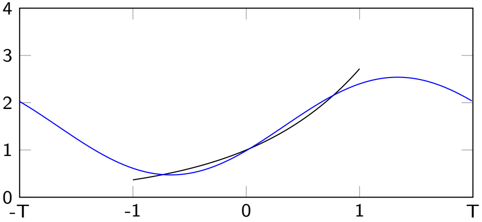
Fourier Extension

A Fourier Extension is an approximation of some function on a given domain in a Fourier Basis on a larger domain (a Fourier Frame).

Example: approximate $\exp(x)$ through interpolation and Fourier extension for increasing dof:



(a) Fourier interpolation

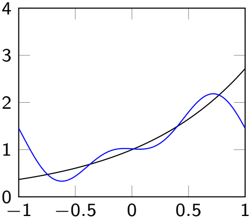


(b) Fourier extension

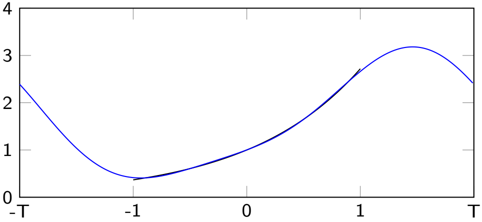
Fourier Extension

A Fourier Extension is an approximation of some function on a given domain in a Fourier Basis on a larger domain (a Fourier Frame).

Example: approximate $\exp(x)$ through interpolation and Fourier extension for increasing dof:



(a) Fourier interpolation

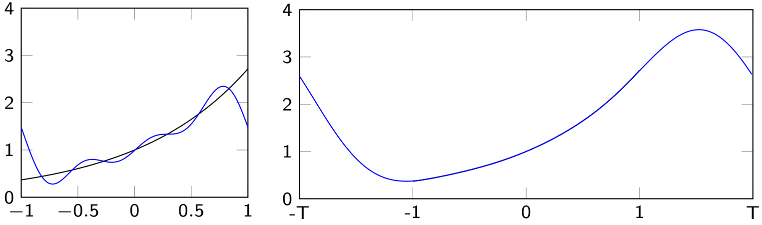


(b) Fourier extension

Fourier Extension

A Fourier Extension is an approximation of some function on a given domain in a Fourier Basis on a larger domain (a Fourier Frame).

Example: approximate $\exp(x)$ through interpolation and Fourier extension for increasing dof:

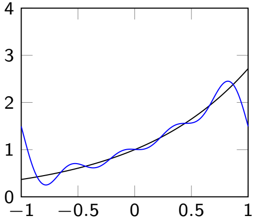


(a) Fourier interpolation (b) Fourier extension

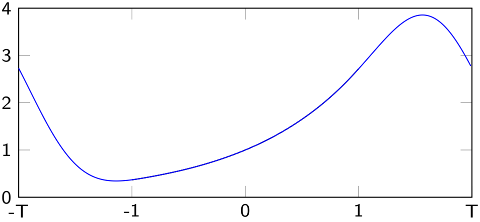
Fourier Extension

A Fourier Extension is an approximation of some function on a given domain in a Fourier Basis on a larger domain (a Fourier Frame).

Example: approximate $\exp(x)$ through interpolation and Fourier extension for increasing dof:



(a) Fourier interpolation

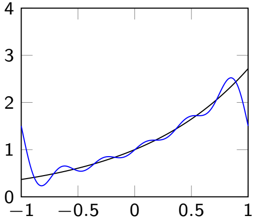


(b) Fourier extension

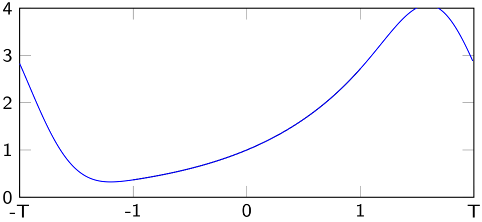
Fourier Extension

A Fourier Extension is an approximation of some function on a given domain in a Fourier Basis on a larger domain (a Fourier Frame).

Example: approximate $\exp(x)$ through interpolation and Fourier extension for increasing dof:



(a) Fourier interpolation



(b) Fourier extension

Fourier Extension

Fourier Frame on $[-T, T]$

- Basis functions $\phi_m(x) = \frac{1}{\sqrt{2T}} e^{i\frac{\pi m}{T}x}, m = -M, \dots, M$
- Approximation $g(x) = \sum_{k=-M}^{M} a_k \phi_k(x)$

Fitting the data on $[-1, 1]$

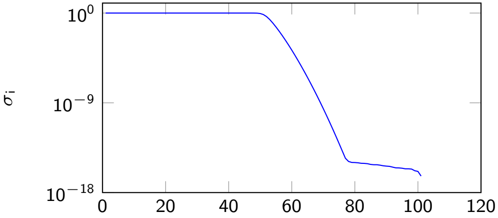
- Collocation at equidistant points $x_j = \frac{j}{N}, j = -N, \dots, N$.
- Least squares problem $A_{ij} = \phi_i(x_j)$, $b_j = f(x_j)$

$$Aa = b$$

- A is DFT-subblock
- Some oversampling $N = \eta M$ required, A is a tall matrix ($\eta \sim 2$)
- Fast solver for $Aa = b$ needed
- Fast algorithm exists for $T = 2$ (Lyon 2011)

A closer look at A

Singular values of A



- Spectrum has a plateau shape (Slepian 1978, Wilson 1987)
- Size of transition/plunge/problematic region

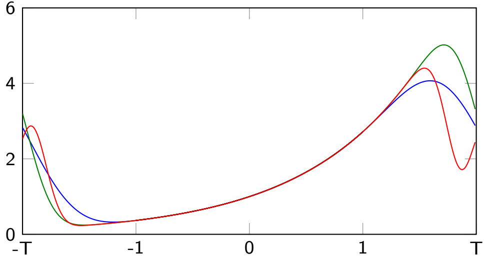
$$1 - \epsilon > \theta_i > \epsilon$$

  grows as $O(\log N)$.
- System matrix is extremely ill-conditioned

A closer look at A

Ill-conditioning explained

- *A Frame is a set of vectors that spans a vector space, but is not necessarily linearly independent.*
- Fourier basis restricted to interval constitutes a frame, a redundant basis
- Finding a representation is a very ill-conditioned problem

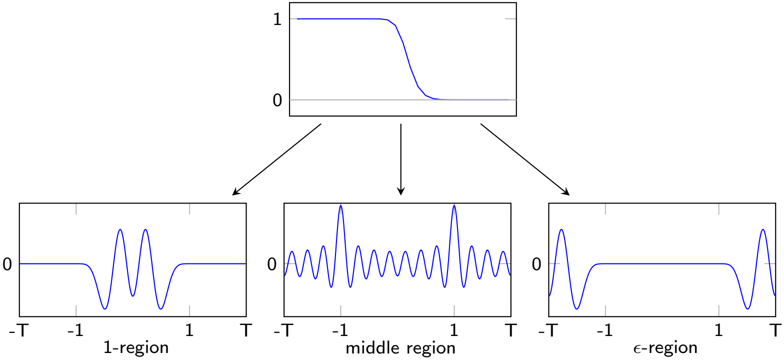


However, $||g - f||_{[-1,1]}$, $||h - f||_{[-1,1]}$, $||l - f||_{[-1,1]}$ are all small.

A closer look at A

Ill-conditioning explained

Fourier series corresponding to the singular vectors of A:



- Solving with Truncated SVD yields approximation to machine accuracy (Adcock, Huybrechs, Martin-Vaquero 2014)

Solving the least-squares system

Truncated SVD

- $A = U\Sigma V'$
- $x = V\Sigma^{-1}U'b$
- $x = \underbrace{V_1\Sigma_1^{-1}U_1'b_1}_{x_1} + \underbrace{V_{mid}\Sigma_{mid}^{-1}U_{mid}'b_{mid}}_{x_{mid}} + \underbrace{V_\epsilon\Sigma_\epsilon^{-1}U_\epsilon'b_\epsilon}_{x_\epsilon}$

Assumption: b_ϵ is negligible (discrete Picard condition)

x_1 is easy when x_{mid} is known

- $b_1 = b - b_{mid} = b - Ax_{mid}$
- $x_1 = V_1\Sigma_1^{-1}U_1'b_1 \approx V_1\Sigma_1 U_1'b_1 = A'b_1$

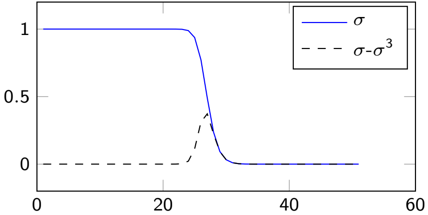
All you need is one application of A and A' (fast!)

How to find x_{mid}?

Isolating the problematic singular values

- 'Projection operator' $I - AA'$ isolates the middle singular values

$$(I - AA')A = U(\Sigma - \Sigma^3)V'$$



The numerical nullspace of $(I - AA')A$ contains both the 1 and ϵ-regions.

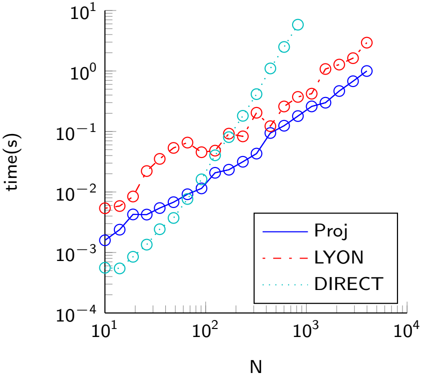
- Solving $(I - AA')Ax = (I - AA')b$ yields x_{mid}
- $(I - AA')A$ has numerical rank $O(\log N)$
- This can be solved efficiently in $O(N \log^2 N)$

Implementation

Algorithm

$$(I - AA')Ax_\beta = (I - AA')b$$
$$x_\alpha = A'(b - Ax_\beta)$$
$$x = x_\alpha + x_\beta$$

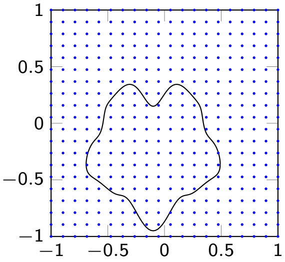
Results



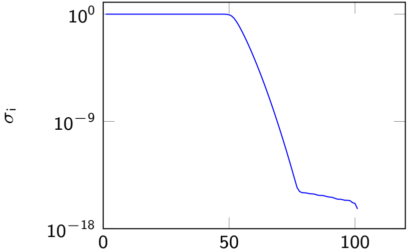
2D Frames

Fourier Frame

- A consists of selected rows of the 2D DFT matrix.



(a) Masked Grid

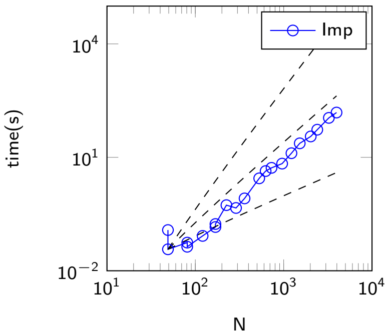


(b) Singular values of A

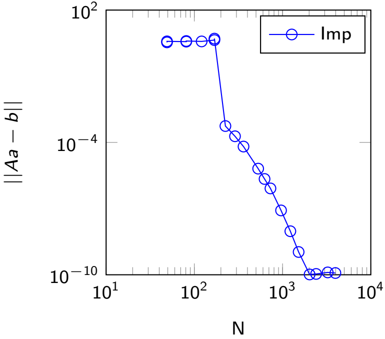
- Partially proven: the problematic region grows as $O(\sqrt{N})$ for N total points.
- The algorithm complexity becomes $O(N^2)$

2D Frames

Experimental results for 2D extensions:



(a) Timings



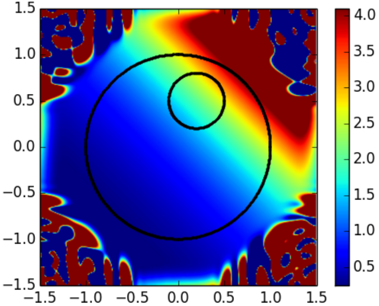
(b) Error

- Complexity is $O(N^2)$, as expected
- Convergence speed is seemingly conserved

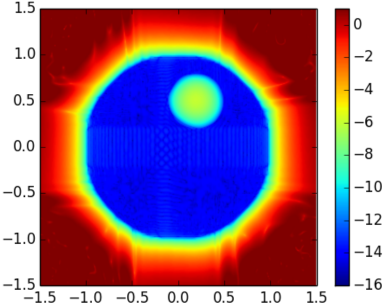
2D Fourier Frame

2D Fourier Extension

$$f(x,y) = exp(x+y)$$



(a) Fourier⊗Fourier Extension

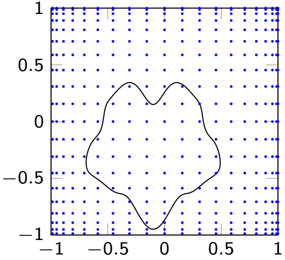


(b) $\log_{10}(error)$

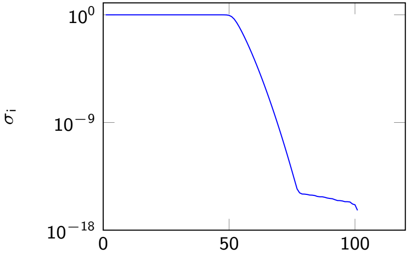
2D Frames

Chebyshev Frame

- A consists of selected rows of the 2D Chebyshev interpolation matrix.



(a) Masked Grid



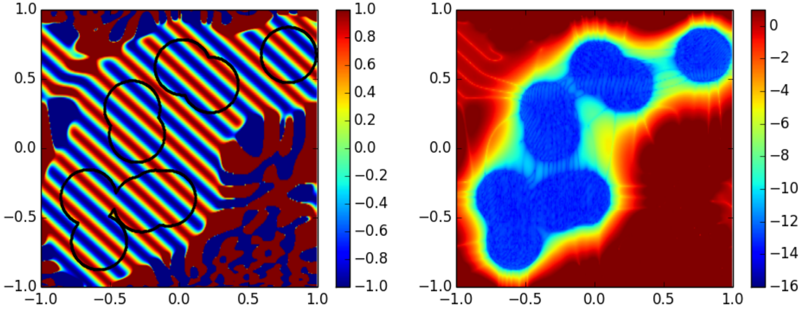
(b) Singular values of A

- The algorithm complexity becomes $O(N^2)$

2D Chebyshev Frame

Combining Fourier and Chebyshev basis functions

$$f(x, y) = \cos(20x + 22y)$$

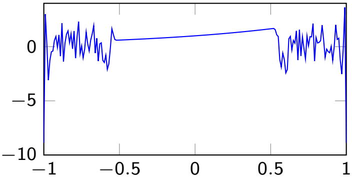


(a) Fourier⊗Chebyshev Extension (b) $\log_{10}(error)$

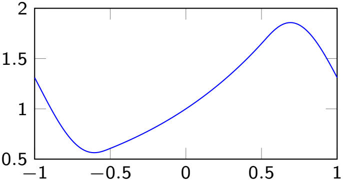
Smoothed Extensions

Manipulating the nullspace

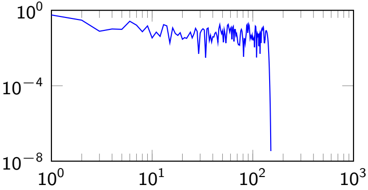
At the same complexity, the coefficients can be made to have a certain decay rate.



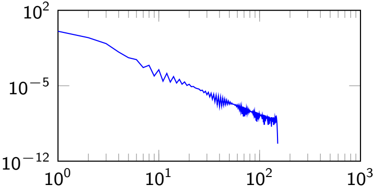
(a) Regular solution



(b) Smoothed solution



(c) Regular coefficients



(d) Smoothed coefficients

Smoothed Extensions

Extension convergence

When smoothed, the extension converges to some fixed function.

- Example: when smoothing the second derivative ($O(n^{-2})$ decay rate), the extension converges to a third degree polynomial that interpolates $f(a), f(b), f'(a), f'(b)$.

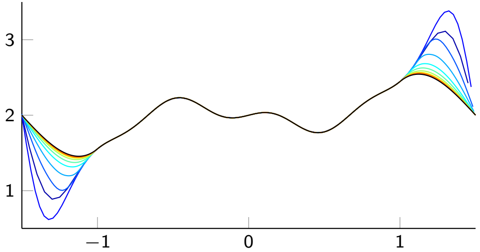
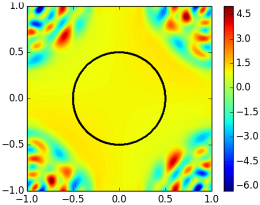


Figure: Smoothed extension for increasing dof.

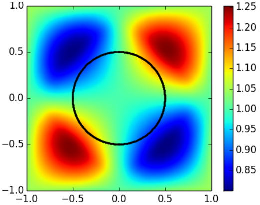
Smoothed Extensions

2D smoothing

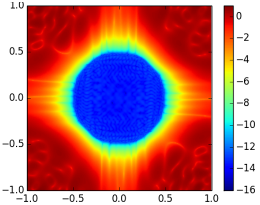
2D functions can also easily be smoothed, example $f(x, y) = e^{x+y}$



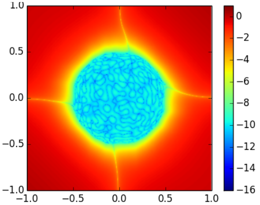
(a) Fourier Extension



(b) Smoothed Extension



(c) $\log_{10}(error)$



(d) $\log_{10}(error)$

Differential Equations

Derivative is diagonal operator

$$f(x) = \sum_{m=-M}^{M} a_m \phi_m(x), \qquad f'(x) = \sum_{m=-M}^{M} \frac{a_m i \pi m}{T} \phi_m(x)$$

A differential equation

$$\Delta A + k^2 A = f$$

becomes

$$(D_x^2 + D_y^2 + k^2 I)\boldsymbol{c}_A = \boldsymbol{c}_f$$

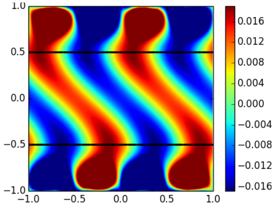
Boundary conditions

- Number of additional equations grows as the dimension of the boundary.
- When implemented correctly, this does not destroy the plateau/plunge region singular value distribution.
- Solving a differential equation has the same complexity as a FE.

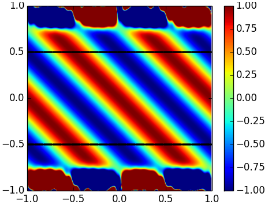
Differential Equations

Poisson Equation

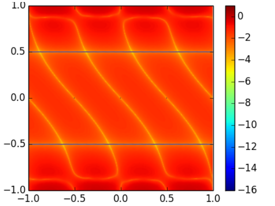
$$\Delta A = \cos(2\pi(x+y)), \qquad \delta A(x,y)/\delta y = 0, \quad (x,y) \in \delta\Omega$$



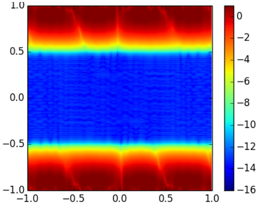
(e) A



(f) ΔA



(g) $\log_{10}(\text{error}_{boundary})$

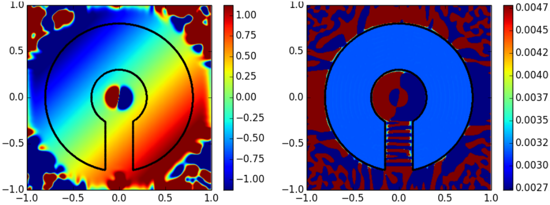


(h) $\log_{10}(\text{error}_{domain})$

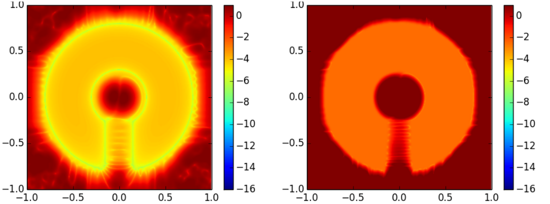
Differential Equations

Laplace Equation

$$\Delta A = 0, \qquad A(x, y) = x - y, \quad (x, y) \in \delta\Omega$$



(a) A (b) ΔA

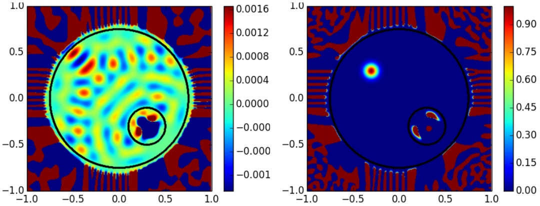


(c) $\log_{10}(\text{error}_{boundary})$ (d) $\log_{10}(\text{error}_{domain})$

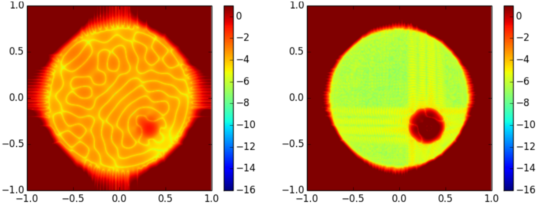
Differential Equations

Helmholtz Equation

$$\Delta A + 35^2 A = e^{-200((x-0.3)^2+(y+0.3)^2)}, \qquad A(x,y) = 0, \quad (x,y) \in \delta\Omega$$



(a) A (b) $\Delta A + k^2 A$



(c) $\log_{10}(\text{error}_{boundary})$ (d) $\log_{10}(\text{error}_{domain})$

The end

Julia code available at `https://github.com/daanhb/Framefuns.jl`