A Hybrid RBF/Fourier method for computing Fourier coefficients from scattered data

Zachary Grant
University of Massachusetts Dartmouth
(zgrant@umassd.edu)

SIAM Annual Meeting 2010

Collaborations

This is joint work with

Prof. Sigal Gottlieb, Prof. Alfa Heryudono, & Sidafa Conde
UMass Dartmouth

This work was funded by NSF grants DMS-0802974 and DMS-0608844

Fourier Series

Every (nice enough) function $f(x)$, has a Fourier representation

$$f_N(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k e^{(ikx)}$$

which approximates the function $f(x)$ on the domain $x \in (-\pi, \pi)$.

We call this approximation *global* because the Fourier coefficients $\hat{f}_k$ are calculated using information about the function from the entire domain $(-\pi, \pi)$.

Calculating Fourier Coefficients

The Fourier representation is good for image deblurring and other applications, so given N function values $f(x_j)$ we want to get the Fourier coefficients $\hat{f}_k$.
One approach is by collocation, we require that

$$\sum_{k=-N}^{N} \hat{f}_k e^{(ikx_j)} = f(x_j).$$

This can be accomplished by solving a linear system

$$A\hat{\mathbf{f}}_k = \mathbf{f}$$

where $\mathbf{f}_j = f(x_j)$ and $A_{kj} = e^{-ikx_j}$.
This collocation approach requires N^2 operations.

Calculating Fourier Coefficients

A more efficient way of calculating the Fourier Coefficients is through the Fast Fourier Transform. Which is a *FAST* method of calculating Coefficients. Here you can see the times it takes to calculate the re projection given N-points.

N	Normal Collocation Times	FFT Times
101	0.0336935210	0.0062124070
201	0.0197917970	0.0102197930
401	0.4646552080	0.0318885820
801	6.8728487070	0.1178515310

Notice in the last row there is actually a 66x speed up and these gains grow with N.

The Problem

Now you can see why the FFT is the preferred choice however it only works well on equidistant points. Now what would happen if the points we sampled our data from were not equidistant but influenced by some Jittering Factor?

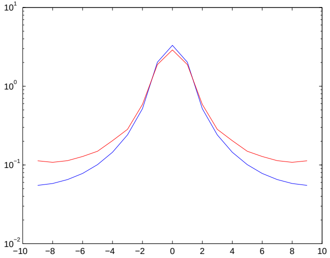
$$\begin{aligned} x_{uniform} &= (x_o + j\frac{x_N - x_0}{N}) \\ Jitter &= \frac{1}{N} * (-1)^j \\ x_{Jitter} &= x_{uniform} + Jitter \end{aligned}$$

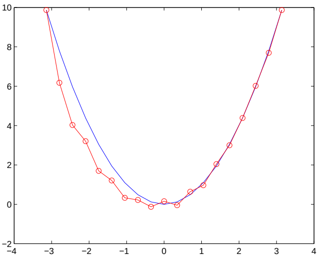
Instead of having uniformly distributed points, we have points that were slightly shifted to the right or to the left

The Problem

Left: Actual Fourier coefficients (blue) vs. the coefficients computed by FFT (red)

(Right) Actual function (blue) vs. the function reconstructed from the FFT coefficients (red)



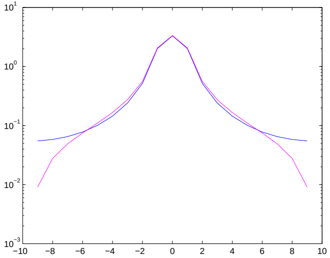


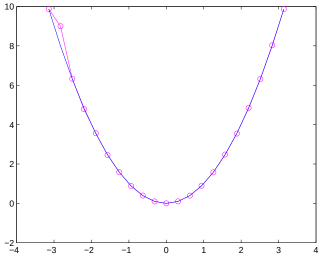
The Problem

The collocation method on non-equidistant points:

Left: Actual Fourier coefficients (blue) vs. the collocation coefficients (magenta) (semilog plot)

(Right) Actual function (blue) vs. the function reconstructed from the collocation coefficients (magenta)





The coefficients look better, but have errors which cause the function reconstruction to look bad

The Problem

Given n points $\{x_j\}$ which are not equidistant, how do we efficiently compute accurate Fourier coefficients?

Some techniques have been suggested to handle the problem of computing Fourier coefficients from non-uniform data. (Gittelson '06) (Potts, Steidl and Tasche)

We propose a technique based on radial basis functions.

Radial basis functions expansions

We start with a set of N points, $X = \{x_j\}$ and define a family of radial basis functions

$$\phi_j(x) = \sqrt{(x - x_j)^2 + \epsilon_j^2}.$$

Now, given a set of N function values $f_i = f(y_i)$, the RBF global approximation is

$$f_N(x) = \sum_{j=1}^{N} \lambda_j \phi_j(x)$$

where the RBF coeffiicients λ_j are given by the requirement that

$$f_N(y_i) = \sum_{j=1}^{N} \lambda_j \phi_j(y_i) = f_i \quad \forall i.$$

Radial basis functions expansions

To satisfy the interpolation condition we must solve a linear system

$$\mathbf{M}\Lambda = \mathbf{f}, \tag{1}$$

where $\Lambda = (\lambda_1, \cdots, \lambda_N)^T$, $\mathbf{f} = (f(y_1), \cdots, f(y_N))^T$, and the interpolation matrix $\mathbf{M}$ is given by

$$M_{ij} = \phi_j(y_i). \tag{2}$$

The expansion coefficient vector Λ is obtained by solving this system

$$\Lambda = \mathbf{M}^{-1}\mathbf{f},$$

where $\mathbf{M}$ is square and nonsingular.

Major Highlights of Radial Basis Functions

- RBFs do not require any particular structure to the points to work well.
- The RBF interpolation is not bounded to a certain domain so it is easy to rescale your points to avoid ill-conditioned matrices.

Algorithm

Given N non equidistant points x_j and the corresponding function values $f(x_j)$

1. Use RBF Interpolation on the original non-uniform points to get the RBF coefficients
2. Use the RBF coefficients to get an approximation of the function on a set of uniform points
3. Compute the corrected Fourier Coefficients from the RBF approximation using the *FFT*

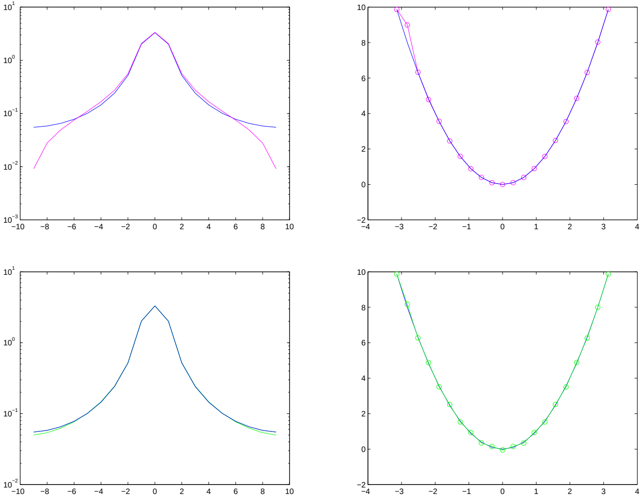
Example 1:

As before consider the function $x \in [-\pi, \pi]$,

$$f(x) = x^2$$

for $x \in [-\pi, \pi]$.
We start with 19 non-equidistant points x_j , and then reconstruct using 19 equidistant points y_j.

Collocation vs. hybrid



collocation(top) vs. hybrid (bottom)

Cost comparison

How much time does it take to run the hybrid vs. the collocation method?

N-1	collocation CPU time	hybrid CPU time	collocation error	hybrid error
16	0.0011	0.0013	6.3 (-2)	8.9 (-3)
32	0.0026	0.0027	1.7 (-2)	1.9 (-3)
64	0.0109	0.0124	4.4 (-3)	4.4 (-4)
128	0.0301	0.0368	1.1 (-3)	6.4 (-5)
256	0.3114	0.3497	2.8 (-4)	3.4 (-6)

$(-n) \equiv 10^{-n}$. The error is computed as $\frac{1}{N}\sum(\hat{f}_k - a_k)^2$ where $\hat{f}_k$ is the exact coefficient and a_k is the computed coefficient

Cost comparison

How about for larger values of N?

N	collocation CPU time	hybrid CPU time	collocation error	hybrid error
101	0.0246	0.025	1.8 (-3)	1.4 (-4)
201	0.0253	0.0255	4.6 (-4)	1.1 (-5)
401	0.4802	0.499	1.1 (-4)	2.1 (-7)
801	6.9958	6.4141	2.9 (-5)	5.6 (-10)
1601	83	77	7.3 (-6)	2.4 (-10)

$(-n) \equiv 10^{-n}$. The error is computed as $\frac{1}{N}\sum(\hat{f}_k - a_k)^2$ where $\hat{f}_k$ is the exact coefficient and a_k is the computed coefficient

350x Speed up for Comparative Error

Cost comparison

How about for larger values of N?

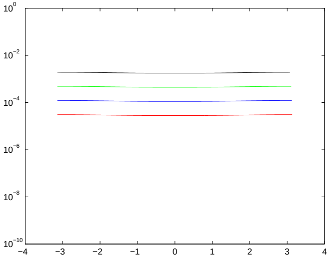
N	collocation CPU time	hybrid CPU time	collocation error	hybrid error
101	0.0246	0.025	1.8 (-3)	1.4 (-4)
201	0.0253	0.0255	4.6 (-4)	1.1 (-5)
401	0.4802	0.499	1.1 (-4)	2.1 (-7)
801	6.9958	6.4141	2.9 (-5)	5.6 (-10)
1601	83	77	7.3 (-6)	2.4 (-10)

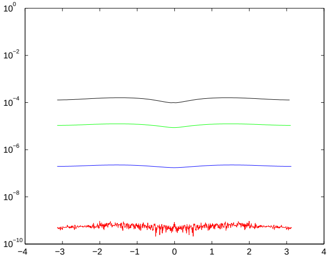
$(-n) \equiv 10^{-n}$. The error is computed as $\frac{1}{N}\sum(\hat{f}_k - a_k)^2$ where $\hat{f}_k$ is the exact coefficient and a_k is the computed coefficient

350x Speed up for Comparative Error

Convergence of Coefficient Errors

Here is a Comparison of the Errors in the Fourier coefficients for $N = 101, 201, 401, 801$ for the Fourier collocation (left) vs. hybrid method (right).





Example 2

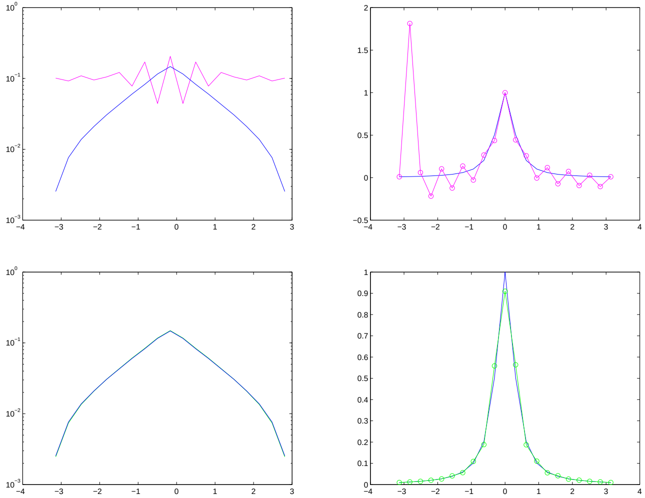
Example 2:

As before consider the function $x \in [-\pi, \pi]$,

$$f(x) = \frac{1}{10x^2 + 1}$$

We start with 19 non-equidistant points x_j , and then reconstruct using 19 equidistant points y_j.

Collocation vs. hybrid



collocation(top) vs. hybrid (bottom)

Cost comparison

How much time does it take to run the hybrid vs. the collocation method?

N-1	collocation CPU time	hybrid CPU time	collocation error	hybrid error
16	0.0169	0.0178	1.2 (-1)	6.4 (-4)
32	0.0125	0.0141	1.0 (-2)	5.2 (-5)
64	0.0168	0.0293	7.0 (-5)	2.3 (-6)
128	0.0535	0.0632	1.1 (-6)	1.7 (-7)
256	0.3101	0.6926	2.8 (-7)	8.8 (-9)

$(-n) \equiv 10^{-n}$. The error is computed as $\frac{1}{N}\sum(\hat{f}_k - a_k)^2$ where $\hat{f}_k$ is the exact coefficient and a_k is the computed coefficient

Cost comparison

N	collocation CPU time	hybrid CPU time	collocation error	hybrid error
101	0.0405	0.0624	2.1 (-6)	3.8 (-7)
201	0.2345	0.0897	4.7 (-7)	3.0 (-8)
401	1.2633	1.2034	1.2 (-7)	5.3 (-10)
801	10.7240	9.6358	2.9 (-8)	1.4 (-12)
1601	80	77	7.4 (-9)	6.1 (-13)

$(-n) \equiv 10^{-n}$. The error is computed as $\frac{1}{N}\sum(\hat{f}_k - a_k)^2$ where $\hat{f}_k$ is the exact coefficient and a_k is the computed coefficient

150x Speed up for Comparative Error

Cost comparison

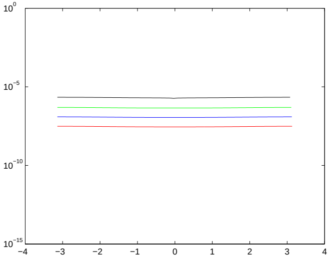
N	collocation CPU time	hybrid CPU time	collocation error	hybrid error
101	0.0405	0.0624	2.1 (-6)	3.8 (-7)
201	0.2345	0.0897	4.7 (-7)	3.0 (-8)
401	1.2633	1.2034	1.2 (-7)	5.3 (-10)
801	10.7240	9.6358	2.9 (-8)	1.4 (-12)
1601	80	77	7.4 (-9)	6.1 (-13)

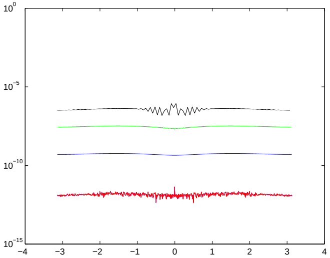
$(-n) \equiv 10^{-n}$. The error is computed as $\frac{1}{N}\sum(\hat{f}_k - a_k)^2$ where $\hat{f}_k$ is the exact coefficient and a_k is the computed coefficient

150x Speed up for Comparative Error

Convergence of Coefficient Errors

Here is a Comparison of the Errors in the Fourier coefficients for $N = 101, 201, 401, 801$ for the Fourier collocation (left) vs. hybrid method (right).





Conclusions

- Comparisons with fast Fourier methods for non-equidistant points are needed.
- Our proposed method uses RBF ideas to compute Fourier Coeficients from non-uniform data
- The hybrid method has the coefficients converge rapidly to the true coefficients
- The hybrid method cost is only a small increase in the amount of time compared to collocation method