MANCHESTER
1824

The University
of Manchester

A fast algorithm for spectral interpolation of sampled data

Tang, D.F.

2012

MIMS EPrint: **2012.48**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

A fast algorithm for spectral interpolation of sampled data

Daniel Tang

April 30, 2012

Abstract

This paper describes a fast algorithm to interpolate between samples of a bandwidth-limited signal that has been sampled at regular intervals. The algorithm is most suited to situations when a small amount of error in the interpolated values is acceptable, the larger the acceptable error, the more efficient is the algorithm. This can be useful in singnal processing, image processing and as an alternative to finite difference calculations. The same algorithm can be used to calculate the value of an n^{th} degree polynomial at the $2n^{th}$ degree Chebyshev points, given its values at the n^{th} degree Chebyshev points, thus providing a fast algorithm for the approximate multiplicaltion of polynomials. This could be used in, for example, computer algebra systems. We consider signals of finite duration, but the same technique can be used to deal with infinite signals so could be used to achieve fast and power-efficient interpolation of streamed data. On a prarallel architecture, the algorithm also requires less inter-process communication than interpolation using a Fast Fourier Transform method.

1 Introduction

Consider a signal, $f(t)$, defined over the interval $0 \leq t \leq \pi$, and let $f_{0\ldots N-1}$ be a set of samples of $f(t)$ at the equi-distant points $t_n = \frac{\pi n}{N}$ such that $f_n = f(t_n)$. We assume that f is real and is bandwidth-limited below the Nyquist frequency of the sample rate, i.e. that over the interval $0 \leq t \leq \pi$, f can be expressed in the form

$$f(t) = \sum_{n=1}^{N/2-1} A_n \sin(2nt) + \sum_{n=0}^{N/2} B_n \cos(2nt) \ .$$

Given this assumption, f can also be expressed as a sum of cardinal functions (see for example Boyd, 2001)

$$f(t) = \sum_{j=0}^{N-1} f_j C_j(t)$$

where

$$C_j(t) = \frac{1}{N}\sin(N(t-t_j))\cot(t-t_j)$$

Suppose we wish to interpolate at the points, $t_{i+\frac{1}{2}} = t_i + \frac{\pi}{2N}$, that lie mid-way between the sample points. Using the identity $\sin(N(t_{i+\frac{1}{2}} - t_j)) = (-1)^{i-j}$, the value of f at the interpolation points can be written as

$$f(t^n_{i+\frac{1}{2}}) = \frac{1}{N}\sum_{j=0}^{N-1} f_j(-1)^{i-j}\cot(t_{i+\frac{1}{2}} - t_j)\ . \tag{1}$$

However, calculating this sum explicitly at all interpolation points would not be computationally efficient, taking $O(N^2)$ operations. Since this is just a discrete convolution of the signal with the cardinal function, the same calculation can be performed in $O(N\log(N))$ operations by taking the Fourier transform of the samples, padding the result with zeroes then taking the reverse Fourier transform.

However, if we are willing to accept a small error in the interpolated values (larger than around 10^{-8} of the maximum value of the signal) there exists an algorithm that is more efficient than the Fourier transform method in terms of number of arithmetic operations and communication overhead. We now present this algorithm.

2 Approximating cot with an exponential sum

We begin with the observation that $\cot(x), 0 \le x \le \pi$ can be expressed as the sum of two completely monotone decreasing functions. For example, if we let $\psi(x)$ be the digamma function then

$$\cot(x) = \frac{1}{\pi}\left(\psi\left(1-\frac{x}{\pi}\right) - \psi\left(\frac{x}{\pi}\right)\right)\ .$$

From this, we would expect that $\cot(x)$ can be approximated well by a sum of exponentials (Breass and Hackbusch, 2009).

We now show that by approximating $\cot(x)$ with a sum of exponentials, the summation in (1) can be approximated at all interpolation points in $O(N)$ operations.

If we let

$$\frac{1}{N}\cot\left(\frac{\pi(i+\frac{1}{2})}{N}\right) \approx \sum_{m=1}^{2M} A_m e^{k_m i}$$

on the interval $0 \le i < N$, then since cot is periodic with a period π

$$\frac{1}{N}\cot\left(\frac{\pi(i+\frac{1}{2})}{N}\right) \approx \sum_{m=1}^{2M} A_m e^{Nk_m} e^{k_m i}$$

on the interval $-N \leq i < 0$. Substituting this into (1) gives

$$f(t^n_{i+\frac{1}{2}}) \approx \sum_{m=1}^{2M} A_m(1 - e^{Nk_m})\Phi_m(i)$$

where

$$\Phi_m(i) = (1-e^{Nk_m})^{-1} \left(\sum_{j=0}^{i} f_j(-1)^{i-j} e^{k_m(i-j)} + \sum_{j=i+1}^{N-1} f_j(-1)^{i-j} e^{Nk_m} e^{k_m(i-j)} \right) . \tag{2}$$

Let

$$S^b_{ma} = \sum_{j=a}^{b} f_j(-1)^j e^{-k_m j}$$

Substituting this into (2), noting that $S^b_{m(a+1)} = S^b_{m0} - S^a_{m0}$ and rearranging gives

$$\Phi_m(i) = (-1)^i e^{k_m i} \left(S^i_{m0} + \frac{e^{Nk_m}}{1 - e^{Nk_m}} S^{N-1}_{m0} \right) . \tag{3}$$

which leads to the recurrence relationship

$$\Phi_m(i+1) = f_{i+1} - e^{k_m}\Phi(i) . \tag{4}$$

So, in theory $\Phi_m(i)$ can be calculated for all $0 \leq i < N$ in just one subtraction and one multiplication per term by using the recurrence relation. The recurrence can get started by calculating $\Phi_m(-1)$ from (2). This will not generally require summation over all points because as the magnitude of k_m increases, the effect of more distant points will quickly decay to below our desired precision. From (4) it can be seen that $\Phi(-1)$ can be calculated to an accuracy of ϵ relative to the signal amplitude by considering only the points i that satisfy

$$i < \frac{log(\epsilon) - log(A_m)}{k_m} .$$

In practice, the recurrence relation (4) cannot be used when k_m is positive as it is numerically unstable. To deal with this, we simply reverse the recurrence to give

$$\Phi_m(i-1) = e^{-km}(f_i - \Phi(i)) \tag{5}$$

and start the recurrence from $\Phi(N-1)$.

So, the values of f at all interpolation points can be calculated using the following procedure:

- Calculate $\Phi_m(N-1)$ and $\Phi_m(-1)$ using (2), only summing over terms with significant value.
- Calculate $\Phi_m(i)$ for negative k_m at all interpolation points by sweeping forward from $i=0$ to $i=N-1$, using the recurrence relationship, (4).
- Calculate $\Phi_m(i)$ for positive k_m at all interpolation points by sweeping backward from $i=N-1$ to $i=0$ using the recurrence relationship, (5).
- At each interpolation point, multiply Φ_m by $\frac{A_m}{1-e^{Nk_m}}$ (which can be precomputed), for each m, and sum to give the final value.

3 Fitting an exponential sum to cot

Finding the parameters, A_m and k_m of an exponential sum approximant to a function is, in general, a notoriously ill conditioned problem; for a review see, for example, Holmström and Petersson (2002). What's more, the fit that minimises error in our algorithm is the fit to cot wich minimises the 1-norm error. The parameter space of the 1-norm error has many local minima and no practical method of finding the global minimum is known.

The method we use here gives bounds on the ∞-norm error and is based on that of Beylkin and Monzón (2005) (hereafter BM), but modified to exploit the anti-symmetry of the cot function. This modification was found to improve the accuracy of the approximant compared to that obtained by approximating the digamma function, as suggested in BM.

We first note that $\cot(t)$ is anti-symmetric about $t=\frac{\pi}{2}$, so $\cot(t) = -\cot(\pi - t)$. We impose the same anti-symmetry on the approximant by requring that

$$\sum_{m=1}^{2M} A_m e^{k_m i} = -\sum_{m=1}^{2M} A_m e^{k_m (N-i)} \ .$$

This is satisfied if $k_{m+M} = -k_m$ and $A_{m+M} = -A_m e^{k_m N}$ for each $0 < m \leq M$ so that the approximant can be expressed in the form

$$\sum_{m=1}^{M} A_m e^{k_m i} - \sum_{m=1}^{M} A_m e^{k_m (N-i)} \ .$$

By convention, we arrange the terms so that $k_m < 0$ for $0 < m \leq M$.

Let $h = (h_0 ... h_{N-1})$ be a vector of values we wish to approximate as a sum of exponentials. Let H be the $\lceil \frac{N}{2} \rceil$ column Hankel matrix such that $H_{ij} = h_{i+j}$:

$$H = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\lceil \frac{N}{2} \rceil - 1} \\ h_1 & \ddots & & h_{\lceil \frac{N}{2} \rceil} \\ \vdots & & \ddots & \vdots \\ h_{\frac{N}{2}} & h_{\frac{N}{2}+1} & \cdots & h_{N-1} \end{bmatrix} .$$

Note that in the case when N is even the matrix is not square, with one more rows than columns.

For any vector $q \in \mathbb{R}^M$, define the *characteristic polynomial* of q to be the polynomial

$$P_q(x) = \sum_{n=0}^{M-1} q_n x^n$$

and define the *characteristic sum* of q to be the sum

$$S_q(A, i) = \sum_{m=0}^{M-1} A_m \Theta_m^i .$$

where $\Theta_0 ... \Theta_{M-1}$ are the zeroes of the characteristic polynomial of q, $P_q(\Theta)$.

We know from Prony's method that if we solve

$$Hq = 0 .$$

then there exists an A such that $h_i = S_q(A, i)$. That is, h can be represented exactly as a sum of exponentials by forming the characteristic sum of q and solving for A, which is equivalent to solving a Vandermonde system.

The method presented in BM extends Prony's method to allow the approximation of h as a sum in fewer than $\lceil \frac{N}{2} \rceil$ terms in a numerically stable way. Their method is based on three observations:

1. If q_n is the n^{th} eigenvector of H, ordered by descending eigenvalue, the error between the characteristic sum, $S_{q_n}(A, i)$, and h_i is bounded by the magnitude of the n^{th} eigenvalue, σ_n.

2. On solving for A, only n terms in the characteristic sum have magnitudes greater than the n^{th} eigenvalue.

3. The eigenvalues of H decay exponentially to zero for many functions of interest.

However, the method in BM cannot be used to find antisymmetric approximants of antisymmetric functions. To prove this, we begin by defining an exponential

sum $S_q(A,i)$ to be antisymmetric if there exists an A such that $S_q(A,i) = S_q(A,-i)$ for all i. This includes all functions that are antisymmetric about any point as a shift in origin can be achieved by changing A.

For any vector $q \in \mathbb{R}^M$ define $q^r \in \mathbb{R}^M$ to be the element-wise reverse of q such that $q^r_{M-n} = q_n$. A vector is said to be symmetric if $q = q^r$ and antisymmetric if $q = -q^r$.

Theorem 1 *For any vector, q, if its characteristic sum, S_q, is antisymmetric then q is either symmetric or antisymmetric depending on whether P_q has an even or odd number of zeroes at 1, respectively.*

Proof 1 *Notice that for an exponential sum to be antisymmetric, there exists an A such that*

$$\sum_m A_m \Theta_m^i = \sum_m A_m \Theta_m^{-i}$$

which is true everywhere only if for every Θ_m there exists an n such that $\Theta_n = \Theta_m^{-1}$. Let q be the vector whose characteristic sum $S_q(A,i)$ is antisymmetric. From the antisymmetry, if Θ_m is a zero of the characteristic polynomial, $P_q(\Theta)$, then Θ_m^{-1} must also be a zero. But if we let $a(x) = (\Theta_m - x)(\Theta_m^{-1} - x) = 1 - (\Theta_m + \Theta_m^{-1})x + x^2$ and notice that $a(x) = x^2 a(x^{-1})$ then, by induction, any N^{th} degree polynomial, $P(x)$, that can be reduced to the product of pairs of the form $(\Theta_m - x)(\Theta_m^{-1} - x)$ has the property that $P(x) = x^N P(x^{-1})$.

The characteristic polynomial may also have an odd number of zeroes at ± 1, since these points are their own reciprocal. An unpaired zero at -1 maintains the property $P(x) = x^N P(x^{-1})$ since $(1+x) = x(1+x^{-1})$, an unpaired zero at 1 changes this to $P(x) = -x^N P(x^{-1})$ since $(1-x) = -x(1-x^{-1})$.

If a vector $q = (q_0 ... q_{N-1})$ has a characteristic polynomial that satisfies $P_q(x) = x^N P_q(x^{-1})$ then q must be symmetric since by equating powers of x, $x^{N-n} = x^N x^{-n}$ so $q_{N-n} = q_n$. Similarly in the case $P_q(x) = -x^N P_q(x^{-1})$, $q_{N-n} = -q_n$ so q is antisymmetric.

□

However, a Hankel matrix, H_h, of an antisymmetric vector h cannot have a symmetric or antisymmetric eigenvector. If

$$H_h q = u$$

then

$$u_k = \sum_{n=0}^{N} h_{k+n} q_n$$

but also

$$u_{N-k} = \sum_{n=0}^{N} h_{N-k+n} q_n = \sum_{n=0}^{N} h_{-k-n} q_{N-n} = -\sum_{n=0}^{N} h_{k+n} q_{N-n}$$

by the antisymmetry of h. However, if q is symmetric then

$$u_{N-k} = -\sum_{n=0}^{N} h_{k+n} q_n = -u_k$$

so u is antisymmetric. If q is antisymmetric then

$$u_{N-k} = \sum_{n=0}^{N} h_{k+n} q_n = u_k$$

so u is symmetric. So it cannot be true that $q = \sigma u$ for some constant σ, so q cannot be a symmetric or antisymmetric eigenvector so the method in BM cannot produce antisymmetric approximants.

The following modified method can be used to find antisymmetric approximants of antisymmetric vectors:

Theorem 2 *Let $h = (h_0 ... h_{N-1})$ be an antiymmetric vector and let H' be the $\frac{N}{4} \times \frac{N}{4}$ matrix whose elements are given by*

$$H'_{ij} = \sum_{n=0}^{i+j} h_n + \sum_{n=0}^{\frac{N}{2}-1+i-j} h_n \ .$$

If q is an eigenvector of H' with eigenvalue σ then there exists a vector A such that

$$h_i = \sum_{n=0}^{M} \left(A_n \Theta_n^i + A_n \Theta_n^{N-1-i} \right) + \epsilon_i$$

where each Θ_n is a zero of $P_q(\Theta) + x^{\frac{N}{2}-1} P_q(\Theta^{-1})$, $M \leq \frac{N}{4}$ and ϵ_i is bounded by 2σ for all i.

Proof 2 *If $H'q = \sigma q$ then*

$$\sum_{j=0}^{\frac{N}{4}} \left(\sum_{n=0}^{i+j} h_n + \sum_{n=0}^{\frac{N}{2}-1+i-j} h_n \right) q_j = \sigma q_i$$

so

$$\sum_{j=0}^{\frac{N}{4}} \left(\sum_{n=0}^{i+j} h_n q_j + \sum_{n=0}^{\frac{N}{4}+i+j} h_n q_j^r \right) = \sigma q_i$$

so

$$\sum_{j=0}^{\frac{N}{2}} \sum_{n=0}^{i+j} h_n (q|q^r)_j = \sigma q_i \tag{6}$$

where $(q|q^r)$ is the vector formed from the elementwise concatenation of q and its reverse.

Let s be the vector

$$s_n = \sum_{m=0}^{n} h_n \ .$$

It follows from (6) that

$$P_s(x)P_{(q|q^r)}(x^{-1}) = \sigma\left(P_q(x) + x^{\frac{N}{4}}P_r(x)\right)$$

for some residual vector r. If we let $r = q^r$ and notice that $(1-x)P_s(x) = P_h(x)$ then we have

$$P_h(x)P_{(q|q^r)}(x^{-1}) = \sigma(1-x)P_{(q|q^r)}(x)$$

Now let $h_i = a_i + \epsilon_i$ where

$$a_i = \sum_{n=0}^{2M} A_n\Theta_n^i$$

is the proposed approximant such that Θ_n are chosen from the zeroes of $P_{(q|q^r)}(\Theta) = P_q(\Theta) + \Theta^{\frac{N}{2}-1}P_q(\Theta^{-1})$. Since $P_{a+\epsilon}(x) = P_a(x) + P_\epsilon(x)$ and we know from Prony's method that there exists an amplitude vector A such that $P_a(x)P_{(q|q^r)}(x^{-1}) = 0$ for powers of x^n, $0 \le n < N$ then

$$P_\epsilon(x)P_{(q|q^r)}(x^{-1}) = \sigma(1-x)P_{(q|q^r)}(x)$$

for these powers of n. So we set $P_\epsilon(x)$ to be the $(N-1)^{th}$degree Padé approximant of

$$\sigma(1-x)\frac{P_{(q|q^r)}(x)}{P_{(q|q^r)}(x^{-1})} \ .$$

However, the magnitudes of the coefficients of a power series are bounded by the magnitude of the series' value on the unit circle. On the unit circle $|(1-x)| \le 2$ and $\left|\frac{P_{(q|q^r)}(x)}{P_{(q|q^r)}(x^{-1})}\right| = 1$ so for all i, $\epsilon_i \le 2\sigma$.

Since ϵ is formed from the characteristic polynomial of an antisymmetric vector, $1-x = P_{(1,-1)}$, multiplied by that of a symmetric vector, $\frac{P_{(q|q^r)}(x)}{P_{(q|q^r)}(x^{-1})}$, ϵ must itself be antisymmetric. Since $a = h - \epsilon$ and ϵ and h are both antisymmetric, then the approximant a must also be antisymmetric, so can be expressed in the form

$$a_i = \sum_{n=0}^{M}\left(A_n\Theta_n^i + A_n\Theta_n^{N-1-i}\right) \ .$$

Since $(q|q^r)$ has $\frac{N}{2}$ zeroes, $M \le \frac{N}{4}$, and as found in BM we find that some of the A_n are smaller than the error so can be neglected.

□

The above method is a robust method of finding good approximants of antisymmetric functions. However, a slight modification can sometimes give better approximants. This involves finding an eigenvector, q, of H' where

$$H'_{ij} = h_{i+j} + h_{\frac{N}{2}-1+i-j}$$

and forming the approximant from the zeroes of $P_q(\Theta) + \Theta^{\frac{N}{2}-1} P_q(\Theta^{-1})$. Following a similar line of reasoning to that given in the proof above, the error can be expressed as

$$P_\epsilon(x) = \sigma \frac{P_q(x) - x^{\frac{N}{4}+1} P_q(x^{-1})}{P_q(x^{-1}) + x^{-\frac{N}{2}+1} P_q(x)} \tag{7}$$

where the negative $P_q(x^{-1})$ term is necessary to ensure the antisymmetry of ϵ. It is not immediately clear that the power-series coefficients of the $(N-1)^{th}$ degree Padé approximant of P_ϵ (which are equal to the error we're interested in) can be bounded in full generality. However, since ϵ is antisymmetric, P_ϵ can be expressed in the form $P_\epsilon(x) = P_g(x) - x^{N-1} P_g(x^{-1})$. A solution to

$$P_g(x) = \frac{P_q(x)}{P_q(x^{-1})} \left(\sigma - x^{1-\frac{N}{2}} P_g(x) \right)$$

would then provide a solution to (7). From this, P_g can be expressed as an infinite series

$$P_g(x) = \sigma \sum_{n=0}^{\infty} (-1)^n x^{n(1-\frac{N}{2})} \left(\frac{P_q(x)}{P_q(x^{-1})} \right)^{n+1} .$$

Equating powers of x gives an expression for the coefficients of P_g (i.e. the elements of g)

$$g_i = \sigma \sum_{n=0}^{\infty} (-1)^n \gamma^{n+1}_{n(\frac{N}{2}-1)+i} \tag{8}$$

where γ^m is the vector defined by

$$P_{\gamma^m}(x) = \left(\frac{P_q(x)}{P_q(x^{-1})} \right)^m .$$

So, if q is such that this sum in (8) converges and can be bound by β then the error, ϵ, can be bound by 2β. In the case of approximants of $\cot(x)$ it was found that this method tended to produce approximants with a smaller 1-norm error, so was the method chosen to create the approximants for the interpolation algorithm.

4 Algorithmic complexity

The proposed algorithm takes $7MN - N + 2e$ arithmetic operations to calculate all interpolation points, where e is the total number of operations needed to calculate $\Phi(-1)$. This can be seen directly from the algorithm: the backward and forward sweeps require $4MN$ arithmetic operations and the final multiplication and addition requires $3MN - N$ operations since, by symmetry, there will be only M distinct values of $\frac{A_m}{1-e^{Nk_m}}$.

The method described in the previous section was used to find exponential sum approximants of cot for different values of M (number of terms in the exponential sum) and for $N = 2^n, 3 \leq n \leq 13$. For each M, N pair, the resulting 1-norm error was calculated along with the total number of arithmetic operations needed to calculate the values of all interpolation points to that error.

A least squares fit of the resulting data was performed and it was found that the total number of arithmetic operations, c, could be described as a function of the number of points N and the 1-norm error ϵ by the relation

$$c = (-0.348 \log_2(\epsilon) + 1.8 \pm 0.13) N \log_2(N)$$

Interpolation using two fast Fourier transforms, using the split-radix FFT algorithm, woud require $12N \log_2(N) - 10N + 16$ operations (Duhamel and Hollman, 1984). So, this algorithm can out-perform the split-radix FFT algorithm at up to 29 bits of accuracy, slightly better than single precision. As the desired accuracy decreases so does the computational requirement of this algorithm.

5 Inter-process communication

The proposed algorithm would also require less inter-process communication if executed on a parallell architecture. For example, a split-radix N-point FFT distributed across p processes would require $O(N \log(p))$ floating point numbers to be transferred between processes, whereas the proposed algorithm would require $2Mp$ where M is the number of terms in the exponential sum. By least squares fitting, it was found that M could be described in terms of the number of points N and the 1-norm error ϵ by the relation

$$M = (0.308 - 0.0503 \log_2(\epsilon)) \log_2(N) + 0.0951 \log_2(\epsilon) \pm 0.159$$

So, the total amount of inter-process communication would be $O(p \log(N))$. Since p is necessarily less than N, this is always less than the split-radix FFT.

6 Conclusion

A new algorithm for calculating the spectral interpolation of a sampled signal has been presented. This algorithm has been shown to require fewer computations than using a split-radix FFT if the required precision is less than single precision and less inter-process communication if implemented on a parallel architecture. The algorithm can easily be modified to perform cosine transformations and so can be used to quickly multiply polynomials (to the given precision) that are represented as samples at the Chebyshev points. The algorithm can also be modified to find the rate of change of a sampled signal at the sample points and so could be used as an alternative to finite difference methods on gridded data.

A new method of finding exponential-sum approximants to antisymmetric functions was also presented. This was based on an algorithm presented in Beylkin and Monzón (2005) but modfied to give antisymmetric approximants. The new method has the added advantage that, by taking advantage of the antisymmetry, it requires fewer computations to find the approximant. The method finds approximants close to the minimum ∞-norm but does not find the global minimum of the 1-norm error which would minimise the error in the interpolation algorithm. A slight improvement to the interpolation algorithm's efficiency could be made, therefore, if a method of finding approximants with better 1-norm error could be found.

The software used to create the approximants, along with approximants of $\frac{1}{N}\cot(\frac{\pi(i+\frac{1}{2})}{N})$ for a range of N can be found at `http://github.com/danftang/ExpFit`.

References

[1] Boyd, J.P., **2001**: Chebyshev and Fourier spectral methods (2^{nd} edition). Dover publications.

[2] Beylkin, G. and Monzón, L. **2005**: On approximation of functions by exponential sums. Appl. Comput. Harmon. Anal. 19: 17-48.

[3] Braess, D. and Hackbusch, W., **2009**: On the efficient computation of high-dimensional integrals and the approximation by exponential sums. DeVore, R.A., Knuth, A. (eds) Multiscale, nonlinear and adaptive approximation, Springer-Verlag, Berlin.

[4] Duhamel, P. and Hollman, H. **1984**: Split-radix FFT algorithms. Electronics Letters, 20, 14-16.

[5] Holmström, K. and Petersson, J. **2002**: A review of the parameter estimation problem of fitting positive exponential sums to empirical data. Applied Math. and Comp. 126: 31-61