Subject: Re: Non-uniform FFT?
Posted by Kenneth P. Bowman on Tue, 05 Apr 2011 17:54:55 GMT
View Forum Message <> Reply to Message

In article
<9cb6bd9c-a2b4-487a-b039-a9e636ba55af@c26g2000vbq.googlegroups.com>,
Eric Hudson <ehudson@mit.edu> wrote:

> Hi,
>
> I was wondering if anyone has implemented a non-uniform FFT algorithm
> in IDL. We have non-regularly spaced real space data that we need to
> Fourier transform, and it is painfully slow to do the discrete
> transform. I have found several c algorithms online (e.g.
> http://www-user.tu-chemnitz.de/~potts/nfft/download.php) but before
> launching into either converting them or figuring out how to run C
> code from within IDL thought maybe someone else had already gone to
> the trouble.
>
> Thanks,
> Eric

The approach could depend on just how non-uniform your data are.

Do you need the whole spectrum, or do you know in advance
which wavenumbers are of interest?

You can do the DFT using least squares (regression), but that will
be slow if you need the full spectrum.

If you only need low wavenumbers, you could interpolate to
a regular grid and then use least squares or the FFT.

Ken Bowman

Subject: Re: Non-uniform FFT?
Posted by Eric Hudson on Wed, 06 Apr 2011 15:32:11 GMT
View Forum Message <> Reply to Message

On Apr 5, 1:54 pm, "Kenneth P. Bowman" <k-bow...@null.edu> wrote:
> In article
> < 9cb6bd9c-a2b4-487a-b039-a9e636ba5...@c26g2000vbq.googlegroup s.com >,
> Eric Hudson <ehud...@mit.edu> wrote:
>
>> Hi,
>
>> I was wondering if anyone has implemented a non-uniform FFT algorithm

>> in IDL. We have non-regularly spaced real space data that we need to
>> Fourier transform, and it is painfully slow to do the discrete
>> transform. I have found several c algorithms online (e.g.
>> http://www-user.tu-chemnitz.de/~potts/nfft/download.php) but before
>> launching into either converting them or figuring out how to run C
>> code from within IDL thought maybe someone else had already gone to
>> the trouble.
>
>> Thanks,
>> Eric
>
> The approach could depend on just how non-uniform your data are.
>
> Do you need the whole spectrum, or do you know in advance
> which wavenumbers are of interest?
>
> You can do the DFT using least squares (regression), but that will
> be slow if you need the full spectrum.
>
> If you only need low wavenumbers, you could interpolate to
> a regular grid and then use least squares or the FFT.
>
> Ken Bowman

Hi Ken,

Thanks for the response. Unfortunately I need the whole spectrum (I have 2D data, slightly irregularly gridded, and want the equivalent of what you'd see if you did a 2D FFT on regularly gridded data). I had thought of doing interpolation and then the standard FFT, which I guess is to an extent what they are doing in these NFFT algorithms, but it seems they are a little more clever than that, which is why I was hoping someone had coded the NFFT routine in IDL. For now I am just directly integrating A(r) exp(i*q*r) over the whole image for each q, which is painfully slow because I have to loop on q (I don't have enough memory to make the whole q*r array in one go).

Eric

Subject: Re: Non-uniform FFT?
Posted by R.G.Stockwell on Fri, 15 Apr 2011 00:33:41 GMT
View Forum Message <> Reply to Message

>
>
> "Eric Hudson" wrote in message
> news:624b5c87-ab22-4a11-b96f-a8d70b4f4ab0@s33g2000vbb.google groups.com...

>
> On Apr 5, 1:54 pm, "Kenneth P. Bowman" <k-bow...@null.edu> wrote:
>> In article
>> < 9cb6bd9c-a2b4-487a-b039-a9e636ba5...@c26g2000vbq.googlegroup s.com >,
>> Eric Hudson <ehud...@mit.edu> wrote:
>>
>>> Hi,
>>
>>> I was wondering if anyone has implemented a non-uniform FFT algorithm
>>> in IDL. We have non-regularly spaced real space data that we need to
>>> Fourier transform, and it is painfully slow to do the discrete
>>> transform. I have found several c algorithms online (e.g.
>>> http://www-user.tu-chemnitz.de/~potts/nfft/download.php) but before
>>> launching into either converting them or figuring out how to run C
>>> code from within IDL thought maybe someone else had already gone to
>>> the trouble.
>>
>>> Thanks,
>>> Eric
>>
>> The approach could depend on just how non-uniform your data are.
>>
>> Do you need the whole spectrum, or do you know in advance
>> which wavenumbers are of interest?
>>
>> You can do the DFT using least squares (regression), but that will
>> be slow if you need the full spectrum.
>>
>> If you only need low wavenumbers, you could interpolate to
>> a regular grid and then use least squares or the FFT.
>>
>> Ken Bowman
>
> Hi Ken,
>
> Thanks for the response. Unfortunately I need the whole spectrum (I
> have 2D data, slightly irregularly gridded, and want the equivalent of
> what you'd see if you did a 2D FFT on regularly gridded data). I had
> thought of doing interpolation and then the standard FFT, which I
> guess is to an extent what they are doing in these NFFT algorithms,
> but it seems they are a little more clever than that, which is why I
> was hoping someone had coded the NFFT routine in IDL. For now I am
> just directly integrating A(r) exp(i*q*r) over the whole image for
> each q, which is painfully slow because I have to loop on q (I don't
> have enough memory to make the whole q*r array in one go).
>
> Eric

I seriously doubt any algorithm can actually solve this problem. There is lomb and lomb scargle, those algorithms produce a spectrum but they do not address the underlying fundament problem of spectral analysis of non-uniform data.

The question (for all fourier transform analysis, regular and non-uniform) is properly framed as a least squares equation, Ax = b where x is the spectrum. To solve it, one must invert the matrix. For regular FFTs, the inverse is the hermitian transpose and you directly get x = A^Tb. The Fast of FFT simply solves x = A^Tb very efficiently. There is almost no reason to ever perform a slow DFT, or to perform a direct integration approximation.

In the non-uniform case, A is not, in general, invertible. The sinusoidal basis functions are not orthogonal, and probably not even independent. Any method I have ever seen (like Lomb) basically ignores this point.

My suggestion, do a loess interpolation/smooth into a downsampled regular-sampled image. by a factor of two for instance. Then FFT.

cheers,
bob