Physics Coding Club

More Fast Fourier Transforms

Phil Hasnip
phil.hasnip@york.ac.uk

Fourier Series

Recall: Any 'sensible' 1D periodic function with period L may be expressed as a Fourier series:

$$f(x) = \sum_{G=-\infty}^{\infty} c_G e^{\frac{2\pi i G x}{L}}$$

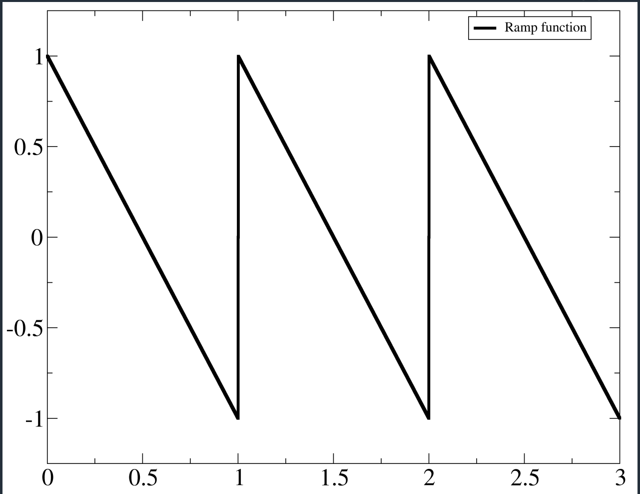
where i is the imaginary number, G is an integer and c_G are complex coefficients.

Note:

- Fourier series are continuous everywhere
- Derivatives of Fourier series are also Fourier series
  $\rightarrow$ Derivatives are continuous everywhere
  $\rightarrow$ Fourier series are differentiable everywhere

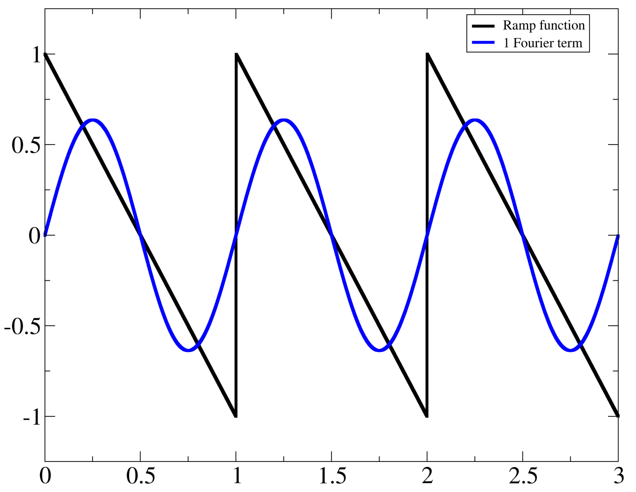
Gibbs overshoot

Where a function is not differentiable, the Fourier series struggles to match it.



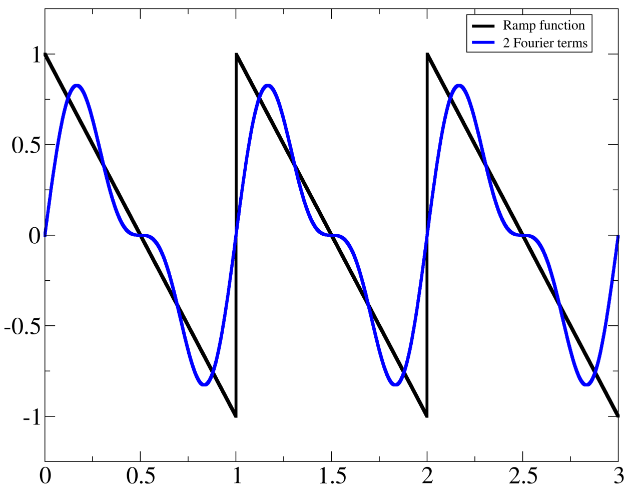
Gibbs overshoot

Where a function is not differentiable, the Fourier series struggles to match it.



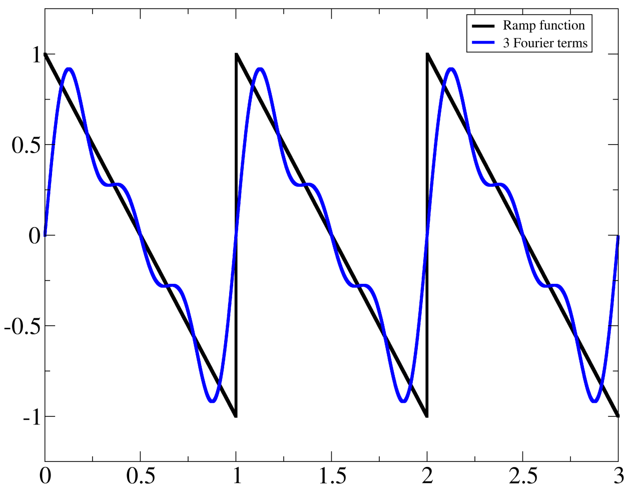
Gibbs overshoot

Where a function is not differentiable, the Fourier series struggles to match it.



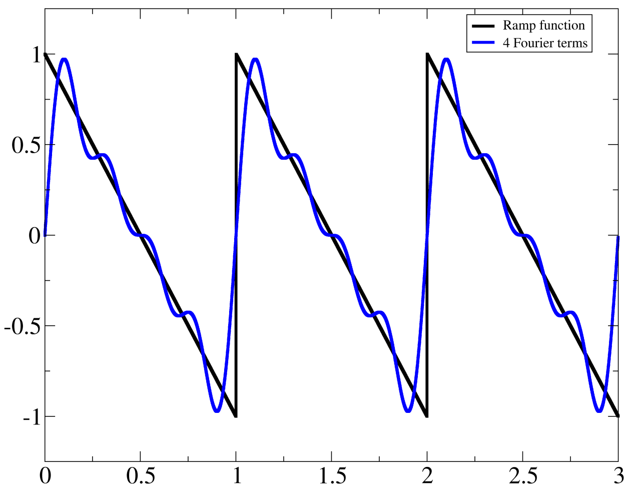
Gibbs overshoot

Where a function is not differentiable, the Fourier series struggles to match it.



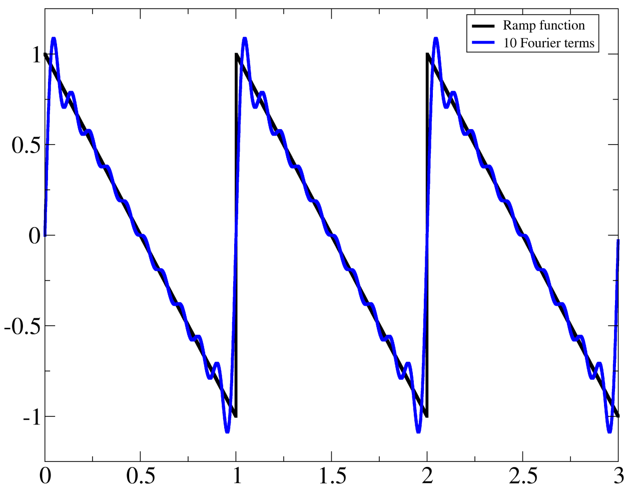
Gibbs overshoot

Where a function is not differentiable, the Fourier series struggles to match it.



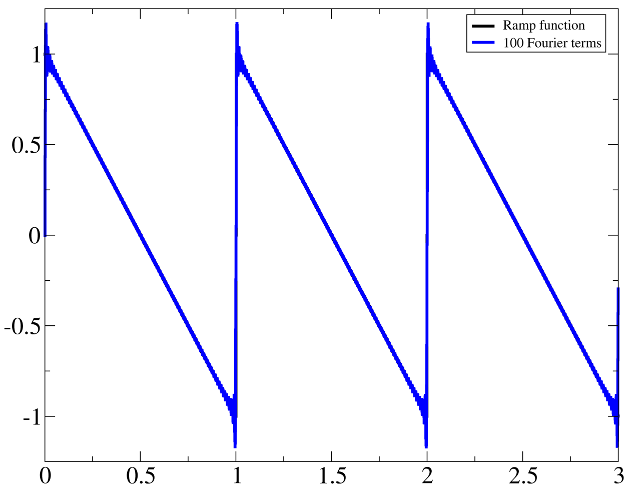
Gibbs overshoot

Where a function is not differentiable, the Fourier series struggles to match it.



Gibbs overshoot

Where a function is not differentiable, the Fourier series struggles to match it.



Square 100

Where a function is not differentiable, the Fourier series struggles to match it.

Square 10

Where a function is not differentiable, the Fourier series struggles to match it.

Square 5

Where a function is not differentiable, the Fourier series struggles to match it.

Square 1

Where a function is not differentiable, the Fourier series struggles to match it.

Squares

Where a function is not differentiable, the Fourier series struggles to match it.

3D

Any 3D periodic function may be expressed as a Fourier series:

$$f(\mathbf{r}) = \sum_{n=-\infty}^{\infty} c_n e^{i\mathbf{G_n}\cdot\mathbf{r}}$$

where c_n are (usually) complex and $\mathbf{G_n}$ are the Fourier wavevectors.

A 3D FFT may be performed as 3 separate sets of 1D FFTs, e.g. transforming the entire data in z, then y and then x.

Beyond the FFT

- Parallel FFT
- Non-uniform FFT
- Fractional FFT
- FFT with derivative information?

We'll only look at the first two here.

Parallel FFT

- Each core only has some of the Fourier components
- Each core only has some of the real-space points
- Fourier transform: *all* Fourier components contribute at *all* points in real-space.

Fourier component parallelism

- 3D transform can be performed as 3 1D transforms
- Give each core all Fourier components in a column in z
- Each core does transform in z
- All cores swap data so they have y-column data
- Each core does transform in y
- All cores swap data so they have x-column data
- Each core does transform in x

  $\longrightarrow$ Each core ends up with real-space data in x

3D FFT in parallel

- Actual (1D) transforms distribute well
- Transpositions are a problem
- Every core has to communicate with every other core!
  $\longrightarrow$ as N_{core} increases, Fourier transform will dominate.
  $\longrightarrow$ when the communication time is comparable to the compute time, there's no point using more cores.

Parallel bottlenecks

- Ultimate scaling dominated by all-to-all communications
- Number of messages increases as N_{cores}^2
- Size of messages decreases as $\frac{1}{N_{cores}^2}$
- Particularly problematic because of limited NICs/node

Non-uniform DFTs

What if we don't sample with a regular grid?

Broadly classified as 3 types of non-uniform Fourier transform (NUFT):

1. Space sampled uniformly, but want non-uniform frequencies
2. Space sampled non-uniformly, but want uniform frequencies
3. Space sampled non-uniformly, want non-uniform frequencies

Because the Fourier transform and inverse Fourier transform are closely related (via complex conjugates), Types 1 and 2 are closely related.

Non-uniform DFTs

Like ordinary DFTs, the obvious way to do an N-point NUFT scales as N^2. There are many ways people try to do NUFTs in a fast way – NUFFTs – but they almost all end up being ways to map the non-uniform problem onto a uniform one (or a set of uniform ones), and using a standard FFT:

- Oversampling + interpolation
  E.g. fit splines to the non-uniform regions and sample onto uniform grid
- Approximate (low-rank) transforms
  These are quick to apply but only approximate the NUFT. The accuracy of the method depends on the rank of the approximation, so it is controllable.

`http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/PIRODDI1/NUFT/NUFT.html`
`https://cims.nyu.edu/cmcl/nufft/nufft.html`
`https://www.math.nyu.edu/faculty/greengar/glee_nufft_sirev.pdf`