Numerical Algorithms for HPC

Introduction to Fourier Transforms

epcc

1

1

Overview

- The Fourier Transform
  - Who, what, why?
  - Fourier Series
  - Mathematical properties of the Fourier Transform
- Discrete Fourier Transform
  - Introduction to first exercise
- Fast Fourier Transform
  - A brief overview
  - Worked example of 4-point DFT

epcc

2

2

Fourier Transfoms

- Jean Baptiste Joseph Fourier (1768-1830) first employed what we now call Fourier Transforms whilst working on the theory of heat
  - The Fourier transform first appeared in "On the Propagation of Heat in Solid Bodies", memoir to Paris Institute, 21 Dec., 1807.
- Linear Transform which takes temporal or spatial information and converts into information which lies in the frequency domain
  - And vice versa
  - Frequency domain also known as Fourier space, Reciprocal space, or G-space -> "Spectral Methods"
- Mathematical tool which alters the problem to one which is more easily solved

Pictures of Joseph Fourier

Who would use Fourier Transforms?

- Physical Sciences
  - Cosmology (P^3M N-body solvers)
  - Fluid mechanics
  - Computational Chemistry (See L07-L09)
  - Quantum physics
  - Signal and image processing
    - Antenna studies
    - Optics
- Numerical analysis
  - Linear systems analysis
  - Boundary value problems
  - Large integer multiplication (Prime finding)
- Statistics
  - Random process modelling
  - Probability theory

Caveat: different disciplines use different notation, normalisation, and sign conventions

epcc 5

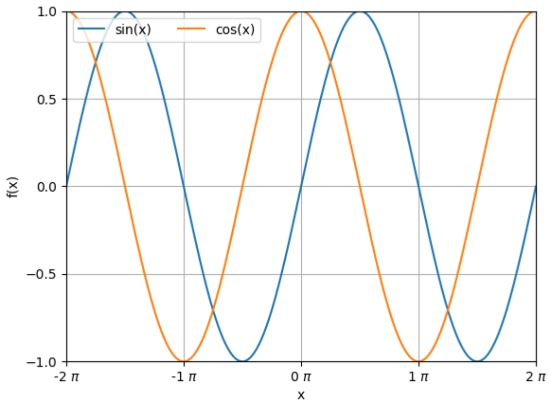
5

Periodic Functions

- A periodic function repeats itself every period *P*:

$$f(x + P) = f(x)$$

- For example, sine (odd) and cosine (even) with $P = 2\pi$.



epcc 6

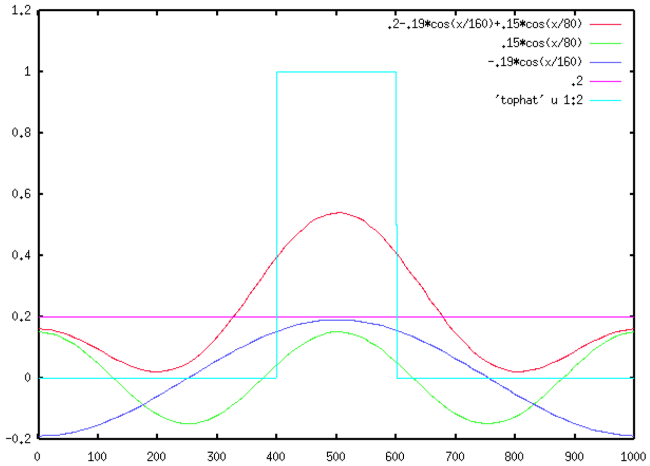
6

Fourier's Theorem

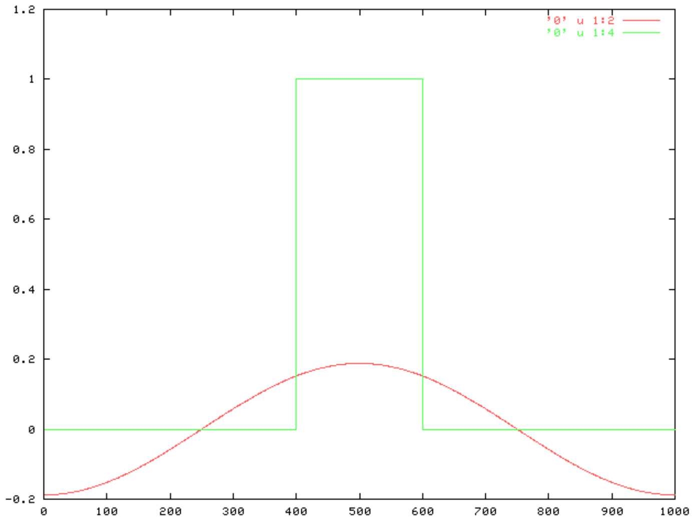
- Straightforward to see that any sum of sines and cosines gives a function which is periodic
  - Turns out the converse is true!
- Principle of superposition:
  - If waves (oscillations) meet at a point, the resulting effect is the sum of each of the individual waves
- Fourier's Theorem:
  - All periodic signals may be represented by an infinite sum of sines and cosines of different periods and amplitudes.
- The cosines and sines are associated with the symmetrical and anti-symmetric information, respectively

Example: The Top Hat Function

- The top hat function, along with the individual 1st, 2nd and 3rd Fourier components and their sum.



Example: The Top Hat Function

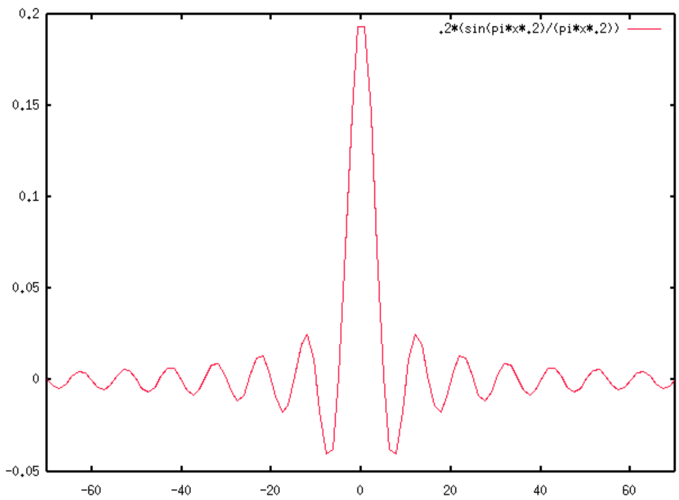


epcc

9

9

Example: The Top Hat Function



epcc

10

10

Fourier Transforms

- Fourier Transforms encode this information via Euler formula $e^{i\theta} = \cos\theta + i\sin\theta$
- NB: Any signal may be considered periodic, by replicating the non-zero part to infinity.
- Commonly used as a way of switching from time domain to frequency domain (and vice-versa)
  - E.g. given a signal as a function of time, what frequencies make up that signal
  - Time and frequency are known as a conjugate pair
  - Other examples are
    - momentum and position
    - potential and charge

epcc

Mathematics of the Fourier Transform

- The Fourier Transform of a complex function $f(x)$ is given as

$$F(s) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi xs}dx$$

- The inverse Fourier Transform is given as

$$f(x) = \int_{-\infty}^{\infty} F(s)e^{i2\pi xs}ds$$

- The Fourier pair is defined as

$$f(x) \Leftrightarrow F(s)$$

epcc

Discrete Fourier Transform

- The Discrete Fourier Transform of N complex points f_k is defined as

$$F_n = \sum_{k=0}^{N-1} f_k e^{2\pi ikn/N}$$

- The inverse Discrete Fourier Transform, which recovers the set of f_k values exactly from the F_n values is

$$f_k = \frac{1}{N}\sum_{n=0}^{N-1} F_n e^{-2\pi ikn/N}$$

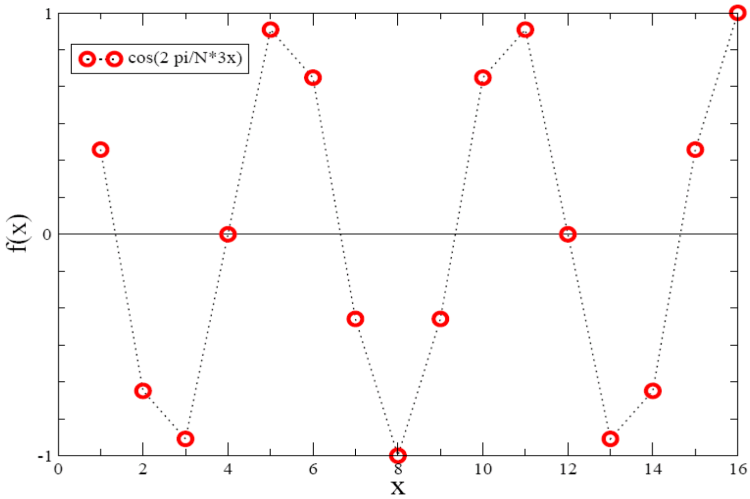
- Both the input function and its Fourier Transform are periodic

epcc 13

13

Example: Cosine function

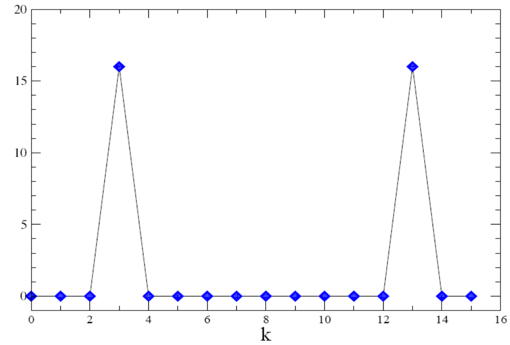
3 periods for N=16:



epcc 14

14

Example: Cosine function



- FT is generally complex – Figure shows real part only
- Peaks of height N at $k = 3$ and $k = N - 3$. This second spike represents the (non-physical) negative frequency – k
- For Fourier transforms of real functions, don't worry about the 2nd half.
- The highest frequencies are at the centre, and lowest at the edges.

epcc

Discrete Fourier Transform

- The DFT can be rewritten as

$$F_n = a_0 + \sum_{k=1}^{N-1} \left(a_k \cos\left(2\pi k \frac{n}{N}\right) + b_k i \sin\left(2\pi k \frac{n}{N}\right) \right)$$

- Thus, the DFT essentially returns real number values for a_k and b_k, stored in a complex array
  - a_k and b_k are functions of f_k
  - remaining trigonometric constants (twiddle factors) may be pre-computed for a given N
- Mathematical properties of the continuous transform also hold for the discrete case.

epcc

Fast Fourier Transform

- What is the computational cost of the DFT?
  - Each of the N points of the DFT is calculated in terms of all the N points in the original function: $\mathcal{O}(N^2)$

$$F_n = \sum_{k=0}^{N-1} f_k e^{2\pi i k n/N}$$

  - Very expensive to compute, even for moderate N

epcc

Fast Fourier Transform

- In 1965, J.W. Cooley and J.W. Tukey published a DFT algorithm which is of $\mathcal{O}(N \log N)$
  - Fast Fourier Transform (FFT)
  - N is a power of 2
  - FFTs in general are not limited to powers of 2, however, the order may resort to $\mathcal{O}(N^2)$
  - Essentially a divide-and-conquer algorithm (details to follow)
  - In hindsight, faster than $\mathcal{O}(N^2)$ algorithms were previously, independently discovered
    - Gauss was probably first to use such an algorithm in 1805

epcc

Fast Fourier Transform

- FFT is an efficient method for computing the DFT
  - Orders of magnitude faster, even for small values of N

N	N^2	$N \log_2(N)$
128	16384	896

- For further reading, implementation details consult:
  - *Numerical Recipes. The Art of Scientific Computing, 3rd Edition*, 2007, Cambridge University Press (www.nr.com)

epcc 19

19

FFT Implementation

- Algorithm based on Danielson & Lanczos (1942)

$$F_n = \sum_{k=0}^{N-1} f_k e^{2\pi ikn/N}$$

$$F_n = \underbrace{\sum_{k=0}^{\frac{N}{2}-1} f_{2k} e^{2\pi i(2k)n/N}}_{\text{even } k} + \underbrace{\sum_{k=0}^{\frac{N}{2}-1} f_{2k+1} e^{2\pi i(2k+1)n/N}}_{\text{odd } k}$$

$$F_n = \sum_{k=0}^{\frac{N}{2}-1} f_{2k} e^{2\pi ikn/(N/2)} + e^{2\pi in/N} \sum_{k=0}^{\frac{N}{2}-1} f_{2k+1} e^{2\pi ikn/(N/2)}$$

$$F_n = F_n^e + W_N^n F_n^o \qquad W_N = e^{2\pi i/N}$$

epcc 20

20

FFT Implementation

- Can continue to break down into smaller and smaller FFTs

$$F_n = F_n^e + W_N^n F_n^o$$

$$F_n = F_n^{ee} + W_{N/2}^n F_n^{eo} + W_N^n F_n^{oe} + W_{N/2}^n W_N^n F_n^{oo}$$

$$F_n = F_n^{ee} + W_N^{2n} F_n^{eo} + W_N^n F_n^{oe} + W_N^{3n} F_n^{oo}$$

- When an *F* becomes a one-point transform it just equals an *f:*

$$F_n^{eo} = f_k$$

- For a 4 element DFT ($N = 4$), each of the remaining 1-element DFTs must be one of the f_k we started with – but which ones?

epcc 21

21

FFT Implementation

- Bit reversal
- Set *e*=0, *o*=1, and reverse the order in binary to find the k corresponding to the sequence of es and os.

$F_n^{ee} = f_{00} = f_0$ $\quad$ $F_n^{eo} = f_{10} = f_2$ $\quad$ etc.

- Each split of the data into odd and even was checking the value of the least-significant bit of *n* in binary.
- Swap elements by bit reversal to the order needed in F_n; the cost is $\mathcal{O}(N)$
- Now build up the F_n by combining the reordered f_k values

epcc 22

22

FFT Implementation

- Recall $F_n = F_n^e + W_N^n F_n^o$

- i.e. we can find all the components of an N-length DFT via 2 $N/2$-length DFTs – these are periodic with period $N/2$ so

$$F_n^e = F_{n-N/2}^e \qquad F_n^o = F_{n-N/2}^o \qquad W_N^n = -W_N^{n-N/2}$$

$$F_n = \begin{cases} F_n^e + W_N^n F_n^o & if \quad n < N/2 \\ F_{n-N/2}^e - W_N^{n-\frac{N}{2}} F_{n-\frac{N}{2}}^o & if \ n \geq N/2 \end{cases}$$

epcc

23

23

FFT Implementation

- So first combine DFTs pairwise to make two N=2 FFTs, with $W_2 = -1$:

f_0	f_2	f_1	f_3

becomes

$f_0+W^0f_2$	$f_0-W^0f_2$	$f_1+W^0f_3$	$f_1-W^0f_3$

=

f_0+f_2	f_0-f_2	f_1+f_3	f_1-f_3

epcc

24

24

FFT Implementation

- Then combine DFTs in even/odd pairs with $W_4 = i$:

f_0+f_2	f_0-f_2	f_1+f_3	f_1-f_3

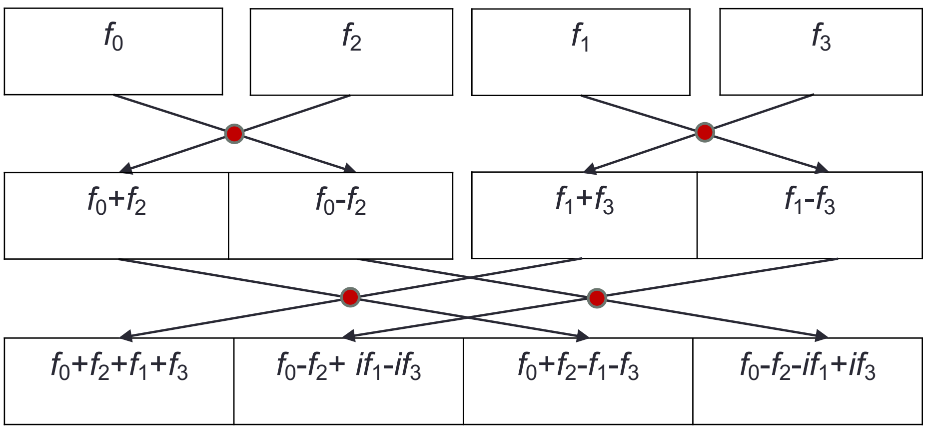
becomes

$f_0+f_2+W^0(f_1+f_3)$	$f_0-f_2+W^1(f_1-f_3)$	$f_0+f_2-W^0(f_1+f_3)$	$f_0-f_2-W^1(f_1-f_3)$

=

$f_0+f_2+f_1+f_3$	$f_0-f_2+ if_1-if_3$	$f_0+f_2-f_1-f_3$	$f_0-f_2-if_1+if_3$

Butterfly diagram



Go left: left box + W x right box. Go right: left box – W x right box.

FFT Implementation

- Try e.g. taking the transform of (1, 2, 3, 4)
- Gives (10, -2-2i, -2, -2+2i)

- Compare with e.g. FFT Calculator
  - http://www.random-science-tools.com/maths/FFT.htm
  - Or implement your own using FFTW (see later)

- Correct answer, (modulo choice of sign for imaginary part)

epcc 27

To summarise:

- Input data are the f_k
- Start by reordering via bit reversal
- Then start to build the full set of transformed F_n at the same time:
- Pairwise add the reordered f_k from the $< N/2$ subset, and pairwise subtract from the $\geq N/2$ subset (from DFT periodicity).
- Then do the same again, this time multiplying the second operand by W_4^n for $< N/2$, and by $W_4^{n-\frac{N}{2}}$ for $\geq N/2$.
- For *N*=4, this is complete.

epcc 28

FFT Implementation

- Pseudocode example given in Num. Recipes Ch. 12 (this is also a good resource in general)
- Key Points
  - $\mathrm{Log}_2(\mathrm{N})$ steps for each element F_n
  - Each step we update N elements
  - Overall runtime is $O(N \log N)$
  - This is a real pain to implement (either by hand or in code)
  - You don't want to ever do this!
  - Use a library!

epcc

Next few slides

- The following slides have more information.
- Mathematical properties of Fourier transforms:
  - Scaling the original and transformed function
  - Shifting the original and transformed function
  - Convolution and correlation
- Why we saw that second peak in the Fourier transform of the cosine.
- Not examinable!

epcc

Properties: Scaling

- Time scaling

$$f(at) \Leftrightarrow \frac{1}{|a|} F\left(\frac{s}{a}\right)$$

- Frequency scaling

$$\frac{1}{|b|} f\left(\frac{t}{b}\right) \Leftrightarrow F(bs)$$

epcc

31

31

Properties: Shifting

- Time shifting

$$f(t - t_0) \Leftrightarrow F(s)e^{2\pi i s t_0}$$

- Frequency shifting

$$f(t)e^{-2\pi i s_0 t} \Leftrightarrow F(s - s_0)$$

epcc

32

32

Properties: Convolution Theorem

- Say we have two functions, $g(t)$ and $h(t)$, then the convolution of the two functions is defined as

$$g \otimes h = \int_{-\infty}^{\infty} g(\tau)h(t-\tau)d\tau$$

- The Fourier Transform of the convolution is simply the product of the individual Fourier Transforms

$$g \otimes h \Leftrightarrow G(s)H(s)$$

epcc 33

33

Properties: Correlation

- The correlation of the two functions is defined by

$$Corr(g,h) = \int_{-\infty}^{\infty} g(\tau+t)h(\tau)d\tau$$

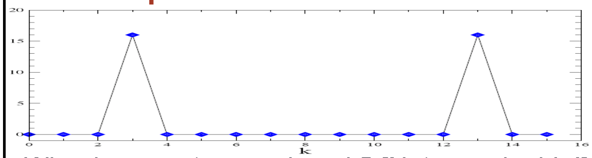
- The Fourier Transform of the correlation is simply

$$Corr(g,h) \Leftrightarrow G(s)H(-s)$$

epcc 34

34

Example: Cosine function – 2nd peak



- Why do we get second peak? [Not examinable!]
- If f is real (i.e. $f_n^* = f_n$) then $F[k]^* = F[-k]$ by definition of F
- As F is periodic with period N, then $F[-k] = F[-k+N] = F[N-k]$
- So $F[N-k] = F[k]^*$ or $\mathrm{Re}\, F[N-k] = \mathrm{Re}\, F[k]$
- i.e. if you get a peak at k then you'll get one at $N - k$
- and graph is symmetrical about the middle
  - imaginary part would be anti-symmetrical
- This second spike represents the (non-physical) negative frequency $-k$
- For Fourier transforms of real functions, 2nd half can be ignored
  - Can be thought of as representing –ve frequencies which don't really have physical meaning

epcc

35

35