# Scientific Computing (Phys 2109/Ast 3100H)
# II. Numerical Tools for Physical Scientists

SciNet HPC Consortium University of Toronto

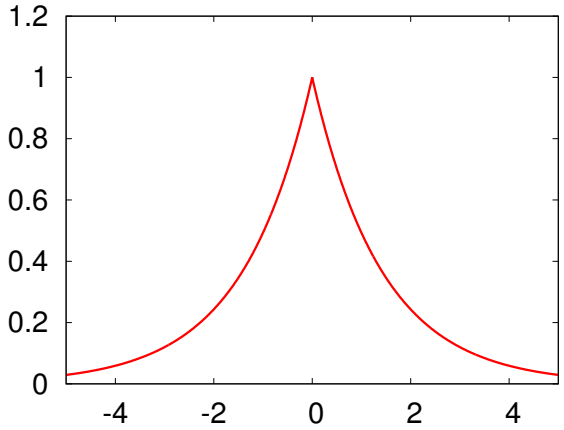
## Lecture 4: Fast Fourier Transform

January 2012

SciNet compute • calcul CANADA

# Fourier transform (FT)

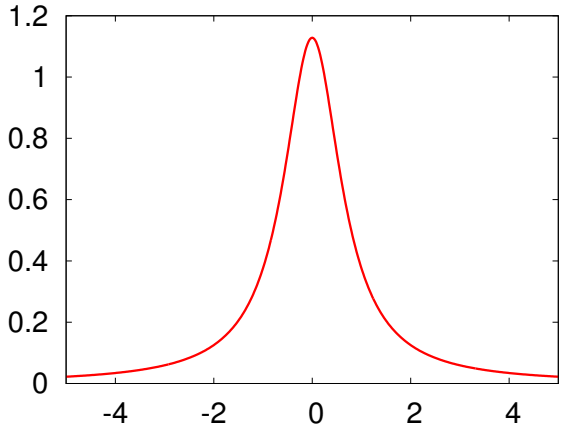
- Let $\mathbf{f}$ be a function of some variable $\mathbf{x}$.
- Transform to a function $\hat{\mathbf{f}}$ of $\mathbf{k}$:




$$\hat{\mathbf{f}}(\mathbf{k}) \propto \int \mathbf{f}(\mathbf{x})\, \mathbf{e}^{\pm \mathbf{i}\, \mathbf{k}\cdot\mathbf{x}}\, \mathbf{dx}$$

J. Fourier

- This transformation can be inverted. If $\mathbf{k}$ is continuous:




$$\hat{\mathbf{f}}(\mathbf{k}) \propto \int \hat{\mathbf{f}}(\mathbf{k})\, \mathbf{e}^{\mp \mathbf{i}\, \mathbf{k}\cdot\mathbf{x}}\, \mathbf{dk}$$

SciNet compute • calcul CANADA

# Fourier Transform (FT)

- Fourier made the claim that any function can be expressed as a harmonic series.
- The FT is a mathematical expression of that.
- Constitutes a linear (basis) transformation in function space.
- Transforms from spatial to wavenumber, or time to frequency, etc.
- Constants and signs are just convention.*

  * some restritions apply.

SciNet compute • calcul CANADA

# Application of the Fourier transform

- Many equations become simpler in the fourier basis.
- Reason: $\mathbf{exp(ik \cdot x)}$ are eigenfunctions of the $\boldsymbol{\partial/\partial x}$ operator.
- Partial diferential equation become algebraic ones, or ODEs.
- Thus avoids matrix operations.

## Examples

- Periodic phenomena
- Spectral analysis
- Signal processing/filtering
- PDEs: virtually anything with a Laplacian

SciNet
compute • calcul
CANADA

# Application of the Fourier transform: examples

**Heat equation**

$$\frac{\partial \mathbf{u}}{\partial \mathbf{t}} = \alpha \nabla^2 \mathbf{u}$$

$$\Downarrow$$

$$\frac{\mathbf{d}\hat{\mathbf{u}}}{\mathbf{dt}} = -\alpha \|\mathbf{k}\|^2 \hat{\mathbf{u}}$$

**Schrödinger equation:**

$$\mathbf{i}\hbar \frac{\partial \boldsymbol{\Psi}}{\partial \mathbf{t}} = -\frac{\hbar^2}{\mathbf{2m}} \nabla^2 \boldsymbol{\Psi}$$

$$\Downarrow$$

$$\mathbf{i}\hbar \frac{\mathbf{d}\hat{\boldsymbol{\Psi}}}{\mathbf{dt}} = \frac{\hbar^2 \|\mathbf{k}\|^2}{\mathbf{2m}} \hat{\boldsymbol{\Psi}}$$

SciNet
compute • calcul
CANADA

# Discrete Fourier Transform (DFT)

C. F. Gauss

- Given a set of $\mathbf{n}$ function values on a regular grid:

$$\mathbf{f_j = f(j\Delta x)}$$

- Transform these to $\mathbf{n}$ other values $\mathbf{\hat{f}_k}$

$$\mathbf{\hat{f}_k = \sum_{j=0}^{n-1} f_j\, e^{\pm 2\pi i j k/n}}$$

- Easily back-transformed:

$$\mathbf{f_j = \frac{1}{n} \sum_{j=0}^{n-1} \hat{f}_k\, e^{\mp 2\pi i j k/n}}$$

- Negative frequencies: $\mathbf{f_{-k} = f_{n-k}}$.
- General aliasing: $\mathbf{k}$ becomes equivalent to $\mathbf{k} + \ell\mathbf{n}$. (max frequency $= \mathbf{k = n/2}$: Nyquist)

SciNet compute • calcul CANADA

# Slow Fourier Transform

- Discrete fourier transform is a linear transformation.
- In particular, it's a matrix-vector multiplication.
- Naively, costs $\mathcal{O}(\mathbf{n^2})$. Slow!
- Same scaling as many solvers.

SciNet
compute • calcul
CANADA

# slow DFT

```
typedef double complex dcomplex;
typedef unsigned long ulong;
void fftn2(ulong n,dcomplex*f,dcomplex*fhat,int dir){
   dcomplex* w = malloc(sizeof(*w)*n);
   double v = (dir<0?-2:2)*3.14159265358979323846/n;
   for (ulong j=0; j<n; j++)
      w[j] = cos(v*j) + 1i*sin(v*j);
   for (ulong k=0; k<n; k++) {
      fhat[k] = 0.0;
      for (ulong l=0; l<n;l++)
         fhat[k] += w[(k*l)%n]*f[l];
   }
   free(w);
}
```

DON'T DO IT!

Even Gauss realized this was too slow and came up with . . .

SciNet compute • calcul CANADA

# Fast Fourier Transform (FFT)

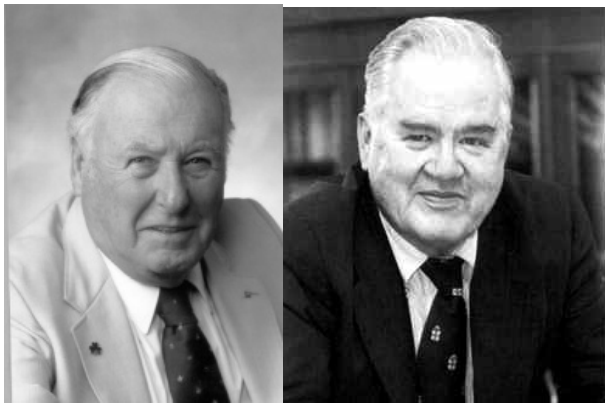
C. F. Gauss

Derived in partial form several times before and even after Gauss, because he'd just written it in his diary in 1805 (published later).

Rediscovered (in general form) by Cooley and Tukey in 1965.

## Basic idea

- Write each $\mathbf{n}$-point FT as a sum of two $\frac{\mathbf{n}}{\mathbf{2}}$ point FTs.
- Do this recursively $^{\mathbf{2}}\log \mathbf{n}$ times.
- Each level requires $\sim \mathbf{n}$ computations: $\mathcal{O}(\mathbf{n}\log \mathbf{n})$ instead of $\mathcal{O}(\mathbf{n^2})$.
- Could as easily into 3, 5, 7, ... parts.



J. W. Cooley  J. Tukey

SciNet compute • calcul CANADA

# If $\mathcal{O}(n \log n)$ versus $\mathcal{O}(n^2)$ does not impress you...

| $n$ | $n \log_2 n$ | $n^2$ | ratio |
|---:|---:|---:|---:|
| 32 | 160 | 1,024 | 6× |
| 128 | 896 | 16,384 | 18× |
| 512 | 4,608 | 262,144 | 57× |
| 2,048 | 22,528 | 4,194,304 | 186× |
| 8,192 | 106,496 | 67,108,864 | 630× |

SciNet
compute • calcul
CANADA

# Fast Fourier Transform: How can you do that?

- Define $\omega_n = e^{2\pi i/n}$.
  Note that $\omega_n^2 = \omega_{n/2}$.
- DFT takes form of matrix-vector multiplication:

$$\hat{f}_k = \sum_{j=0}^{n-1} \omega_n^{kj} f_j$$

- With a bit of rewriting (assuming $n$ is even):

$$\hat{f}_k = \underbrace{\sum_{j=0}^{n/2-1} \omega_{n/2}^{kj} f_{2j}}_{\text{FT of even samples}} + \omega_n^k \underbrace{\sum_{j=0}^{n/2-1} \omega_{n/2}^{kj} f_{2j+1}}_{\text{FT of odd samples}}$$

- Repeat, until the lowest level (for $n = 1$, $\hat{f} = f$).
- Note that a fair amount of shuffling is involved.

SciNet
compute • calcul
CANADA

# Fast Fourier Transform: How do you really do that?

Do not write your own: use existing libraries.

- Because getting all the pieces right is tricky;
- Getting it to compute fast requires intimate knowledge of how processors work and access memory;
- Because there are libraries available.
  Examples: fftw, intel mkl.
- Because you have better things to do.

SciNet
compute • calcul
CANADA

# FFTW

## Example

```
#include <fftw.h>
typedef double complex dcomplex;
typedef unsigned long ulong;
void fftw(ulong n,dcomplex*f,dcomplex*fhat,int dir) {
   fftw_plan p = fftw_plan_dft_1d(n,f,fhat,
      dir<0?FFTW_BACKWARD:FFTW_FORWARD, FFTW_ESTIMATE);
   fftw_execute(p);
   fftw_destroy_plan(p);
}
```

SciNet
compute • calcul
CANADA

## Hands-on:

- Type this in and make sure it compiles.
- Given an 1d input signal: a discretized **sinc(x) = sin(x)/x** with 16384 points on the interval [-30:30].
- Get the linking straighened out.
- Perform forward transform
- Write to file
- Continous FT of **sinc(x)** is

$$\mathbf{rect(f)} = \begin{cases} \mathbf{a} & \text{if } |\mathbf{f}| \leq \mathbf{b} \\ \mathbf{0} & \text{if } |\mathbf{f}| > \mathbf{b} \end{cases}$$

- Does that match?

SciNet
compute • calcul
CANADA

# Symmetries

## Real data

- All arrays were complex so far.
- If input $\mathbf{f}$ is real, this can be exploited.

$$\mathbf{f}_j^* = \mathbf{f}_j \leftrightarrow \hat{\mathbf{f}}_k = \hat{\mathbf{f}}_{n-k}^*$$

- Each complex number holds two real numbers, but for the input $\mathbf{f}$ we only need $\mathbf{n}$ real numbers.
- If $\mathbf{n}$ is even, the transform $\hat{\mathbf{f}}$ has real $\hat{\mathbf{f}}_0$ and $\hat{\mathbf{f}}_{n/2}$, and the values of $\hat{\mathbf{f}}_k > \mathbf{n/2}$ can be derived from the complex valued $\hat{\mathbf{f}}_{0<k<n/2}$: again $\mathbf{n}$ real numbers need to be stored.
- Beware of implement dependent storage patterns.

SciNet compute • calcul CANADA

# Inverse

## Inverse DFT

- Inverse DFT is similar to forward DFT, up to a normalization: almost just as fast.

$$f_j = \frac{1}{n} \sum_{k=0}^{n-1} \hat{f}_k \, e^{\mp 2\pi i j k/n}$$

  Many implementations (almost all in fact) leave out the $1/n$ normalization.
- FFT allows quick back-and-forth between **x** and **k** domain (or e.g. time and frequency domain).
- Allows parts of the computation and/or analysis to be done in the most convenient or efficient domain.

SciNet compute • calcul CANADA

# Multidimensional transforms

In principle a straighforward generalization:

- Given a set of $\mathbf{n} \times \mathbf{m}$ function values on a regular grid:

$$\mathbf{f_{ab} = f(a\Delta x, b\Delta y)}$$

- Transform these to $\mathbf{n}$ other values $\hat{\mathbf{f}}_\mathbf{k}$

$$\hat{\mathbf{f}}_{\mathbf{kl}} = \sum_{\mathbf{a=0}}^{\mathbf{n-1}} \sum_{\mathbf{b=0}}^{\mathbf{m-1}} \mathbf{f_{ab}}\, \mathbf{e}^{\pm\, 2\pi \mathbf{i}\,(\mathbf{a\,k + b\ l})/\mathbf{n}}$$

- Easily back-transformed:

$$\mathbf{f_{ab}} = \frac{\mathbf{1}}{\mathbf{nm}} \sum_{\mathbf{k=0}}^{\mathbf{n-1}} \sum_{\mathbf{l=0}}^{\mathbf{m-1}} \hat{\mathbf{f}}_{\mathbf{kl}}\, \mathbf{e}^{\mp\, 2\pi \mathbf{i}\,(\mathbf{a\,k + b\ l})/\mathbf{n}}$$

- Negative frequencies: $\mathbf{f_{-k,-l} = f_{n-k,m-l}}$.

SciNet
compute • calcul
CANADA

# Multidimensional FFT

- We could successive apply the FFT to each dimension
- This may require transposes, can be expensive.
- Alternatively, could apply FFT on rectangular patches.
- Mostly should let the libraries deal with this.
- FFT scaling still $\mathbf{n \log n}$.
- Real transform even more convoluted.

SciNet
compute • calcul
CANADA

# FFTW

**2! = 3**

## Capabilities

- Complex one-dimensional transforms
- Complex multi-dimensional transforms.
  (Needs contiguous arrays)
- Real-to-half-complex array transforms
- Format real transforms different in 1d and nd.
- Read the manual!

## Notes:

- Always create a plan first. Plans can be reused in the program, and even saved on disk!
- Works with doubles by default.

SciNet
compute • calcul
CANADA

# Homework

## Trigonometric interpolation

Trigometric interpolation uses a **n** point Fourier series to find values at intermediate points. It is one way of "downscaling" data, and was a motivation for Gauss, to be applied to planetary motion. The way it works is:

- You fourier transform your data
- You add frequecies above the Nyquist frequency (in absolute values), but set all the amplitudes of the new frequencies to zero.
- Note that the frequencies are stored such that eg. $\hat{\mathbf{f}}_{\mathbf{n-1}}$ is a low frequency $-\mathbf{1}/\mathbf{n}$.
- The resulting 2n array can be back transformed, and now gives an

SciNet compute • calcul CANADA

# Homework

## Assignment 1

Write an application that will read in an image as a binary file with a 2d array, in double precision, and creates an image twice the size in all directions.
Use a real-to-real version of the fftw.

## PPM image format

The image format to be used is ppm, which goes as follows:

- first line: "P6\n"
- second line: "width height\n"
- Subsequently triplets of 3 (rgb) byte values per pixel.

SciNet
compute • calcul
CANADA

# Homework

**Assignment 2**

Write an application which reads an image and performs a low pass filter on the image, i.e., any fourier components with mangitudes **k** larger than n/4 are to be set to zero, after which the fourier inverse is taken and the image is to be printed out.

Due next Thursday at noon!

SciNet
compute • calcul
CANADA