Universities of Lübeck and Chemnitz Helsinki University of Technology
Jens Keiner, Stefan Kunis, Special Year in Numerics
and Antje Vollrath

The Nonequispaced FFT and its Applications

A Mini Course at Helsinki University of Technology

`http://math.tkk.fi/numericsyear/NFFT`

Lab 2 **C Library Hands On**

Exercise 1:
Browse through the NFFT homepage

`http://www.tu-chemnitz.de/~potts/nfft`

Then, download the NFFT package and build the library in your home directory, i.e.,

1. `tar xfvz nfft-3.1.0.tar.gz`
2. `cd nfft-3.1.0`
3. `./configure`
4. `make`

Lookup and open the source file `simple_test.c` found in `nfft-3.1.0/examples/nfft`. Skim through the subroutine `simple_test_nfft_1d()`. Try to understand what it does. Then, run the actual executable `simple_test`.

Exercise 2:
Using matrix-vector notation as in the lecture, the NFFT algorithm corresponds to using the approximation

$$\mathbf{A}\hat{\mathbf{f}} \approx \mathbf{BFD}\hat{\mathbf{f}},$$

where $\mathbf{B}$ denotes the real $M \times n$ sparse matrix

$$\mathbf{B} := \left(\tilde{\psi}\left(x_j - \frac{l}{n}\right)\right)_{j=0,\dots,M-1;\, l=-n/2,\dots,n/2-1}.$$

We propose different methods for the compressed storage and application of the matrix $\mathbf{B}$ which are all available in the NFFT library by choosing different precomputation flags. These methods do not yield a different asymptotic performance but yet lower the constant hidden in the $\mathcal{O}$ notation.

Compare the situation with no precomputation (that is, no precomputation flags set) with the usage of the flags `PRE_PSI` and `PRE_FULL_PSI` in the routine `simple_test_nfft_2d`. Modify the call to `nfft_init_guru` as necessary. There should be an observeable performance difference.

Exercise 3:
The NFFT library has a simple interface to compute discrete spherical Fourier transforms (NDSFTs) in Matlab. While the C interface is very similar to the plain NFFT routines, the Matlab interface is more convenient to use.

Run the `configure` script with the option `--with-matlab=<path/to/matlab>` where `<path/to/matlab>` should be replaced with a valid path to the local Matlab installation. Then, recompile the NFFT library. Observe that there will be newly created binaries in the directory `nfft-3.1.0/matlab/nfsft`.

Start up Matlab and change to said directory. Browse the script file `simple_test.m` and try to understand what it does. Then, run it.

Write a Matlab `function y = f(x)` that evaluates the function

$$f(\vartheta, \varphi) = \begin{cases} 1, & \text{if } \vartheta \in [0, \pi/2], \\ (1 + 3\cos^2\vartheta)^{-1/2}, & \text{if } \vartheta \in (\pi/2, \pi]. \end{cases}$$

The input should be a matrix $\mathbf{x} \in \mathbb{R}^{2 \times M}$ with spherical coordinates $\vartheta_j \in [0, \pi]$, $\varphi_j \in [0, 2\pi)$, $j = 1, 2, \ldots, M$, i.e.,

$$\mathbf{x} = \begin{pmatrix} \varphi_1 & \varphi_2 & \ldots & \varphi_M \\ \vartheta_1 & \vartheta_2 & \ldots & \vartheta_M \end{pmatrix}.$$

Then, take approximations to the function f by computing projections onto spherical harmonics expansions up to degrees $N = 8, 16, 32, \ldots, 512$. Compare the approximations to the original function by evaluating them on a set of points on the sphere of your choice and calculating the relative error

$$E_N := \frac{\|\mathbf{f} - \mathbf{f}_N\|_2}{\|\mathbf{f}\|_2}.$$

Here, $\mathbf{f}$ contains the exact function values and $\mathbf{f}_N$ contains the values of the approximation of degree N.

Hint: You can use the script `simple_test.m` as a starting point. But note that you will have to change the order in which transformations are computed.