Universities of Lbeck and Chemnitz
Jens Keiner, Stefan Kunis,
and Antje Vollrath

The Nonequispaced FFT and its Applications

A Mini Course at Helsinki University of Technology

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

Solution 2 **C Library Hands On**

Exercise 1:

The routine `simple_test_nfft_1d()` initialises a plan for a one-dimensional nfft with $N = 14$ Fourier coefficients and $M = 19$ nodes, generates random nodes $x_j \in [-\frac{1}{2}, \frac{1}{2}]$, $j = 0, \ldots, 18$, and precomputes the matrix $\mathbf{B} \in \mathbb{R}^{19\times 32}$. Random Fourier coefficients $\hat{f}_k \in \mathbb{C}$, $k = -7, \ldots, 6$ are generated as well and the trigonometric polynomial $f(x) = \sum_{k=-7}^{6} \hat{f}_k \mathrm{e}^{-2\pi \mathrm{i} k x}$ is evaluated at the nodes x_j by the routine `ndft_trafo` (direct way), and by `nfft_trafo` using the approximation scheme as outlined in the lecture. Coefficients and function values are displayed. Also, the adjoint tranforms are executed - clearly showing that the nfft is, in contrast to the fft, a non-unitary transform.

```
1) computing an one dimensional ndft, nfft and an adjoint nfft


 given Fourier coefficients, vector f_hat, adr=0x804c0e8
     0. +3.0E-01+8.8E-01i,+5.3E-01+9.2E-01i,+5.2E-01+8.1E-01i,+1.9E-01+8.9E-01i,
     4. +5.7E-01+7.7E-02i,+8.2E-01+9.8E-01i,+1.2E-01+8.9E-01i,+7.8E-01+1.0E-01i,
     8. +2.5E-01+2.0E-02i,+3.8E-01+6.8E-01i,+6.8E-01+7.5E-01i,+6.2E-03+6.2E-01i,
    12. +1.3E-01+6.2E-01i,+7.7E-01+1.9E-01i,

 ndft, vector f, adr=0x804c1d0
     0. +9.1E-01+3.4E-01i,+9.1E-01+3.1E-01i,+2.5E-01+2.7E-01i,+3.2E+00-8.6E-01i,
     4. -4.7E-01-1.0E+00i,-4.8E-01+9.3E-01i,-4.6E-01+9.3E-01i,+6.7E+00+7.6E+00i,
     8. +3.1E+00+6.9E+00i,+4.7E+00+2.5E+00i,-3.9E-01-1.2E+00i,+6.2E-01-1.5E+00i,
    12. +1.3E+00+4.4E-01i,-3.9E-01+7.7E-02i,+1.3E-01+2.0E+00i,-3.4E-02+4.5E-01i,
    16. -1.7E+00-7.9E-01i,+7.8E-01+8.6E-02i,-6.2E-01-1.2E+00i,

 nfft, vector f, adr=0x804c1d0
     0. +9.1E-01+3.4E-01i,+9.1E-01+3.1E-01i,+2.5E-01+2.7E-01i,+3.2E+00-8.6E-01i,
     4. -4.7E-01-1.0E+00i,-4.8E-01+9.3E-01i,-4.6E-01+9.3E-01i,+6.7E+00+7.6E+00i,
     8. +3.1E+00+6.9E+00i,+4.7E+00+2.5E+00i,-3.9E-01-1.2E+00i,+6.2E-01-1.5E+00i,
    12. +1.3E+00+4.4E-01i,-3.9E-01+7.7E-02i,+1.3E-01+2.0E+00i,-3.4E-02+4.5E-01i,
    16. -1.7E+00-7.9E-01i,+7.8E-01+8.6E-02i,-6.2E-01-1.2E+00i,

 adjoint ndft, vector f_hat, adr=0x804c0e8
     0. +9.9E-01+1.5E+01i,+1.5E+01+1.1E+01i,+6.9E+00+1.9E+01i,+1.3E+01+2.2E+01i,
     4. +1.1E+01+1.4E+01i,+2.1E+01+2.6E+01i,+1.0E+01+1.9E+01i,+1.8E+01+1.6E+01i,
     8. +7.5E+00+1.3E+01i,+1.2E+01+2.3E+01i,+1.4E+01+1.2E+01i,+8.0E+00+3.3E+00i,
    12. +2.6E+00+1.2E+01i,+1.5E+01+4.2E+00i,

 adjoint nfft, vector f_hat, adr=0x804c0e8
     0. +9.9E-01+1.5E+01i,+1.5E+01+1.1E+01i,+6.9E+00+1.9E+01i,+1.3E+01+2.2E+01i,
     4. +1.1E+01+1.4E+01i,+2.1E+01+2.6E+01i,+1.0E+01+1.9E+01i,+1.8E+01+1.6E+01i,
     8. +7.5E+00+1.3E+01i,+1.2E+01+2.3E+01i,+1.4E+01+1.2E+01i,+8.0E+00+3.3E+00i,
    12. +2.6E+00+1.2E+01i,+1.5E+01+4.2E+00i,
```

Exercise 2:
The initialisation with no precomputation of **B**, compressed storage, and explicit storage are given by

```
nfft_init_guru(&p, 2, N, N[0]*N[1], n, 4,
               PRE_PHI_HUT| MALLOC_F_HAT| MALLOC_X| MALLOC_F |
               FFTW_INIT| FFT_OUT_OF_PLACE,
               FFTW_ESTIMATE| FFTW_DESTROY_INPUT);
```

```
nfft_init_guru(&p, 2, N, N[0]*N[1], n, 4,
               PRE_PHI_HUT| PRE_PSI| MALLOC_F_HAT| MALLOC_X| MALLOC_F |
               FFTW_INIT| FFT_OUT_OF_PLACE,
               FFTW_ESTIMATE| FFTW_DESTROY_INPUT);
```

and

```
nfft_init_guru(&p, 2, N, N[0]*N[1], n, 4,
               PRE_PHI_HUT| PRE_FULL_PSI| MALLOC_F_HAT| MALLOC_X| MALLOC_F |
               FFTW_INIT| FFT_OUT_OF_PLACE,
               FFTW_ESTIMATE| FFTW_DESTROY_INPUT);
```

For a transform with $N_0 = 70$ and $N_1 = 50$, i.e. $N_0N_1 = 3500$ Fourier coefficients, and $M = 3500$ evaluation nodes, the computation times are something like

transform	cpu time (secs.)
ndft	1.74e+00
nfft, no precomputation	8.00e-02
nfft, PRE_PSI	1.60e-02
nfft, PRE_FULL_PSI	4.00e-03

Exercise 3:
An exemplatory Matlab-Script that computes the error as specified in Exercise 3:

```
function err = projection()
% threshold
kappa = 1000;
% polynomial degrees
emin = 0; emax = 9; Nv = 2.^(emin:emax);
% saves errors
err = [Nv',zeros(length(Nv),1)];
% uniform random nodes
Me = 1000; xe = [2*pi*rand(1,Me);acos(2*rand(1,Me)-1)]; fe = ff(xe);
% precomputation
nfsft_precompute(max(Nv),kappa);

% loop over polynomial degrees
j = 1;
```

```
for N = Nv
  % projection using Gauss-Legendre points
  [x,w] = gl(N);
  M = size(x,2);
  plan = nfsft_init_advanced(N,M,NFSFT_NORMALIZED);
  nfsft_set_x(plan,x);
  nfsft_precompute_x(plan);
  f = ff(x).*w;
  nfsft_set_f(plan,f);
  nfsft_adjoint(plan);
  fh = f_hat(nfsft_get_f_hat(plan));
  nfsft_finalize(plan);

  % evaluation at random nodes
  plan = nfsft_init_advanced(N,Me,NFSFT_NORMALIZED);
  nfsft_set_x(plan,xe);
  nfsft_precompute_x(plan);
  nfsft_set_f_hat(plan,double(fh));
  nfsft_trafo(plan);
  fa = nfsft_get_f(plan);
  err(j,2) = norm(fe-fa)/norm(fe);
  j = j + 1;
  nfsft_finalize(plan);
end

% delete precomputed data
nfsft_forget();

% error plot
figure;
loglog(Nv,err(:,2));

end

% the function f
function y = ff(x)
n = size(x,2);
y = ones(1,n);
j = x(2,:) > pi/2;
y(j) = 1./sqrt(1 + 3*cos(x(2,j)).^2);
end
```

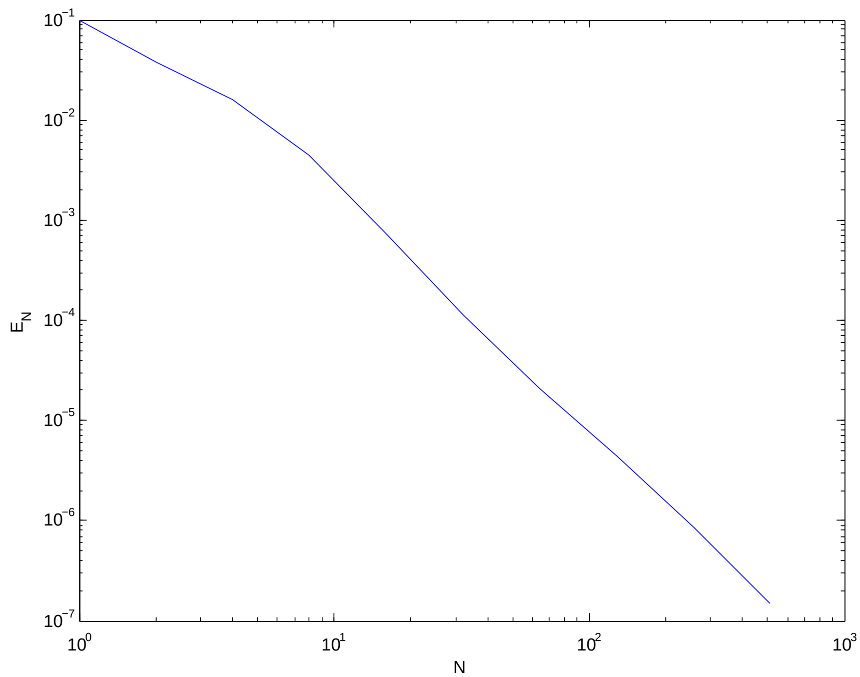

Figure 1: The relative error E_N plotted against the polynomial degree N (the bandwidth).