博客-分数阶傅里叶变换数值计算研究

The FrFT, which is a generalization of the ordinary Fourier transform.

我最近在研究frft-分数阶傅里叶变换的数值计算问题，主要是Ozaktas提出的采样型算法，以及Pei 等提出的采样型算法。

前一种算法，一般都把信号模型建立在[-N/2,N//2]的时间范围内，而时间中的信号都是从零开始的，假如我的信号是时间区间为[0,8N],每T时间内的区间都要做FRFT变换，怎样把[0，N]，(N,2N],.......,(7N,8N] 区间内的信号转换到 [-N/2,N/2]就成了困扰我的难题。补充：由于信号模型是s(t)=exp(j\*2\*π\*f0\*n/fs+j\*π\*u\*t^2)，所以不能直接将（kN，(k+1)N】内的信号直接通过移位实现.

请求处理过的人给与指导

Pei采样型算法虽然从公式上看起来很好理解，可是我编出来的程序，对于上面的信号s(t)愣是不出峰值。另外，这个方法如何进行信号s(t)的参数f0和u的估计。

希望有解决了我这些问题或之一的的人能够指教一二，小女子感激不尽啊！！**分数阶傅里叶变换frft数值计算，求助！！**

参考Ozaktas提出的采样型算法，我的matlab程序如下：

在程序中的三种信号形式下，只有第一种情况能够准确的估计出信号s(t)的f0 和u，第二、三中情况下只能估计出u，f0就不正确了，思考了很久业不知道问题出在哪里，求高人指点。

matlab M文件程序

N=511;

fi=0;

f0=20; %初始频率0;%

fs=97.6;%600; % 抽样频率1e6;%

k=8; %调频率3e4;%

f=zeros(1,N);%频率，每个frft值代表的频率（分正负）

t=zeros(1,N);%时间，每个时域信号样点代表的时间（分正负）

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*\*\*\*\*第一种情况：信号时间区间为【-N/2，N/2】

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for n=-fix(N/2):fix(N/2)

x(n+ceil(N/2))=exp(j\*fi+j\*2\*pi\*f0\*n/fs+j\*pi\*k\*((n/fs)\*(n/fs)));%generate signal t=n/fs

t(n+ceil(N/2))=n/fs;%time

f(n+ceil(N/2))=n\*fs/N;%freq

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*\*\*\*\*第二种情况：信号时间区间为【0，N-1】

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for n=0:N-1

x(n+1)=exp(j\*fi+j\*2\*pi\*f0\*n/fs+j\*pi\*k\*((n/fs)\*(n/fs)));%generate signal t=n/fs

t(n+1)=n/fs;%time

f(n+1)=(n-fix(N/2))\*fs/N;%freq

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*\*\*\*\*第三种情况：信号时间区间为【-N/2，N/2】+N

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for n=-fix(N/2)+N:fix(N/2)+N

x(n+ceil(N/2)-N)=exp(j\*fi+j\*2\*pi\*f0\*n/fs+j\*pi\*k\*((n/fs)\*(n/fs)));%generate signal t=n/fs

t(n+ceil(N/2)-N)=n/fs;%time

f(n+ceil(N/2)-N)=(n-N)\*fs/N;%freq

end

X=awgn(x,0);

r=0.01;

a=[0:r:2];%fractional power

G=zeros(length(a),length(X)); %for different power do frft

h=0;

for l=1:length(a)

T=frft(X,a(l));

G(l,:)=abs(T(:));

if(h<=max(abs(T(:))))

h=max(abs(T(:)));

p1=a(l); %阶数

end

end

for m=1:length(a);

y(m)=max(abs(G(m,:)));

end

tt=t(N)-t(1);%截取信号的持续时间

kk=tt/(fs);

kr=-cot(p1\*pi/2)/kk % k参数的估计值,参考公式（4-30）

FF=frft(x,p1);

[aa,bb]=max(abs(FF));

bb

u0=f(bb);

ff0=u0\*csc(p1\*pi/2) % 中心频率f0的估计值

figure=figure('Color',[1 1 1]);

plot(f\*csc(p1\*pi/2),abs(FF),'r');

function Faf = frft(f, a)

% The fast Fractional Fourier Transform快速分数傅里叶变换

% input: f = samples of the signal

% a = fractional power分数幂

% output: Faf = fast Fractional Fourier transform

error(nargchk(2, 2, nargin));

f = f(:);

N = length(f);

shft = rem((0:N-1)+fix(N/2),N)+1;

sN = sqrt(N);

a = mod(a,4);

% do special cases

if (a==0), Faf = f; return; end;

if (a==2), Faf = flipud(f); return; end;

if (a==1), Faf(shft,1) = fft(f(shft))/sN; return; end

if (a==3), Faf(shft,1) = ifft(f(shft))\*sN; return; end

% reduce to interval 0.5 < a < 1.5

if (a>2.0), a = a-2; f = flipud(f); end

if (a>1.5), a = a-1; f(shft,1) = fft(f(shft))/sN; end%即将参与frft的信号变为原始信号f的傅立叶变换

if (a<0.5), a = a+1; f(shft,1) = ifft(f(shft))\*sN; end%即将参与frft的信号变为原始信号f的逆傅立叶变换

% the general case for 0.5 < a < 1.5

alpha = a\*pi/2;

tana2 = tan(alpha/2);

sina = sin(alpha);

f = [zeros(N-1,1) ; interp(f) ; zeros(N-1,1)];

% chirp premultiplication

chrp = exp(-i\*pi/N\*tana2/4\*(-2\*N+2:2\*N-2)'.^2);

f = chrp.\*f;

% chirp convolution

c = pi/N/sina/4;

y=exp(i\*c\*(-(4\*N-4):4\*N-4)'.^2);

Faf = fconv(exp(i\*c\*(-(4\*N-4):4\*N-4)'.^2),f);

Faf = Faf(4\*N-3:8\*N-7)\*sqrt(c/pi);%去卷积结果的中间1/3总长度的数据点数

% chirp post multiplication

Faf = chrp.\*Faf;

% normalizing constant

Faf = exp(-i\*(1-a)\*pi/4)\*Faf(N:2:end-N+1);

function xint=interp(x)

% sinc interpolation

N = length(x);

y = zeros(2\*N-1,1);

y(1:2:2\*N-1) = x;

xint = fconv(y(1:2\*N-1), sinc([-(2\*N-3):(2\*N-3)]'/2));

xint = xint(2\*N-2:end-2\*N+3);%取卷积结果的中间1/3总长度的数据

function z = fconv(x,y)

% convolution by fft

N = length([x(:);y(:)])-1;

P = 2^nextpow2(N);

z = ifft( fft(x,P) .\* fft(y,P));

z = z(1:N);