

%AUFGABE a)

function xx = pp1a(x,richtung)

N = length(x);%laenge des vektors

if (N == 1)

xx = x; %fft im eindimensionalen fall

else

if (richtung == 0) % fuer richtung=0 rechnet das programm die inverse fft aus (bis auf faktor N)

tmp = exp(-i*2*pi/N);

else

tmp = exp(i*2*pi/N);% ansonsten rechnet das programm die normale fft der vorlesung

end

N2=N/2;

g = pp1a(x(1:2:(N-1)),richtung);%rekursiver auf der fft fuer die geraden eintraege

u = pp1a(x(2:2:N),richtung);%rekursiver auf der fft fuer die ungeraden eintraege

fac=1;

for k=1:N2%in der for-schleife werden die eintraege des ergebnis-vektors gemaess dem satz aus der vorlesung

% aus den geraden und ungeraden teilvektoren zusammengesetzt.

summand=u(k)*fac;

xx(k) = g(k) + summand;

xx(k+N2) = g(k) - summand;

fac=fac*tmp;

end

end

end

%AUFGABE b)

function fN = pp1b(f,M,N)

%leich abgewandelt von der aufgabenstellung interpoliert pp1b die funktion f an N vielen punkten und rechnet das IPP

%an M stellen wieder aus.

tmp=2*pi/N;

N1=N-1;

for k=0:N1

values(k)=f(k*tmp);%einlesen der funktionswerte an den t_j

end

fk=(1/N)*fft(values);%berechnen der f^ mithilfe der inversen fouriertrafo

tmp=2*pi/M;

for n=1:M

t=tmp*n;%zeitpunkte fuer die das IPP ausgerechnet wird

faktor=exp(1i*t);

fNtmp=0.5*faktor^(-N/2)*fk(N/2+1);

for k=-N/2+1:-1

fNtmp=fNtmp+faktor^k*fk(k+N+1);

end

for k=0:N/2-1

fNtmp=fNtmp+faktor^k*fk(k+1);

end

fNtmp=fNtmp+0.5**faktor^(N/2)*fk(N/2+1);

fN(n)=fNtmp;

end

%am besten das programm mit der folgenden eingabe in der matlab-konsole testen ;)

%N=128; M=256;

%t = 0:2*pi/M:(2*pi-(2*pi/(M+1)));

%gN = pp1b(@g,M,N);

%plot(t,[g(t);gN]);

%hN = pp1b(@h,M,N);

%plot(t,[h(t);hN]);

%AUFGABE c)

clear all

close all

X = load('Faltungsmatrix.asc');

Y = double(imread('Ausgangsmatrix.jpg'));

%die folgenden drei zeilen dienen nur zur darstellung der ausgangsmatrix als teil eines plots

hold on

subplot(2,1,1)

imshow(Y);

%start des eigentlichen algorithmus

```

X1 = fft(X);%bekommt die matlab funktion fft wie hier eine matrix als eingabe, wird fuer jede spalte die fft
ausgerechnet
X2 = fft(X1');
Y1 = fft(Y);
Y2 = fft(Y1');

Z = X2'.*Y2';
Z = ifft(Z);
Z = ifft(Z');
%ende des eigentlichen algorithmus
%die folgenden vier zeilen dienen nur zur darstellung der ergebnismatrix als teil eines plots
C = Z'/255;
imwrite(C,'Ergebnismatrix.jpg');
subplot(2,1,2)
imshow('Ergebnismatrix.jpg');

%AUFGABE d)
function x=pp1d(A,b)
N=length(b);
d=fft(A(1,1:N));%eigenwerte berechnen als fouriertransformierte der ersten zeile von A
x=ifft(b);
x=x./d;
x=fft(x);
end

%AUFGABE e)
format long
clear all
close all

N=256;
x=(2*pi/N)*[0:N-1]';           % Gitter
f=sin(x)+0.2*sin(3*x)-0.2*cos(6*x); % ungestoerte Funktion

e=0.1*randn(N,1);             % Stoe rung normalverteilt
                                % mit Streuung 0.1

b=f+e;                         % gestoerte Messwerte in b

bb=fft(b);                     % inverse FFT
n=[0:N/2-1 -N/2:-1]';
alpha=0.0001;                  % Regularisierungsparameter, der solange geraten werden muss, bis
                                % bild passt
uu=bb./(1+alpha*n.^4);         % Filter

u=ifft(uu);                    % FFT, geglaettete Daten in u

hold on
subplot(2,1,1)
plot (x,[real(u),f,b]);        % plote f,u,b als Funktionen von x
legend('geglaettet','exakt','gestoerte Werte');

fd=cos(x)+0.6*cos(3*x)+1.2*sin(6*x); % exakte Ableitung
bd=fd+e;                        % gestoerte Messwerte in bd
alpha=0.0000055;                % Regularisierungsparameter, der solange geraten werden muss, bis
                                % bild passt
uu=i*bb.*n./(1+alpha*n.^6);     % Filter
ud=ifft(uu);                    % FFT, geglaettete Daten in ud
subplot(2,1,2)
plot (x,[real(ud),fd,bd]);      % plote fd,ud,bd als Fkt. von x
legend('geglaettet','exakt');

```