TU Berlin Fakultät IV
Institut für Energie und Automatisiertungstechnik
Fachgebiet Elektronische Mess- und Diagnosetechnik
Praktikum Messdatenverarbeitung

Praktikum Messdatenverarbeitung
Termin 7

Özgü Dogan (326 048)
Timo Lausen (325 411)
Boris Henckell (325 779)

5. Juli 2012

Gruppe: G1 Fr 08-10

Betreuer: Jürgen Funk

Inhaltsverzeichnis

1 Vorbereitungsaufgaben

1.1 Vorbereitungsaufgaben zu Termin 7

1.1.1 Chirp-Signal erzeugen

Als erstes sollte anhand Matlab ein chirp-Signal erzeugt und untersucht werden. Dieser wurde mit dem Aufruf chirp() erzeugt, dem man einen Zeitvektor, die Startfrequenz und weitere Angaben über den Verlauf geben konnte. Bei unserem Signal sollte ein linearer Frequenzanstieg erfolgen.

Als Beispiel wurde folgendes Signal erstellt:
t = $0 : 0.001 : 2$
chirp(t,0,1,100)

Das Chirp-Signal und das Spektrogram dazu sehen so aus:

Abb. 1: erzeugtes Chirp-Signal über der Zeit

Abb. 2: Spektrogramm des erzeugten Chirp-Signals

Man sieht einen Sinusverlauf, dessen Frequenz mit der Zeit immer größer wird. Wir vermuten einen linearen Abstieg der Frequenz. Im Spektrogramm kann man deutlich sehen, dass nach einer Sekunde die Frequenz den erwünschten Wert von $100Hz$ annimmt.

Weiterhin kann man die Auswirkung der Eingabevariablen des Spektrogrammaufrufs auf das entstehende Spektrogramm untersuchen. Hier sind zwei Beispiele, in denen einmal die Überlappungsfläche zwischen zwei Segmenten verkleinert wird und einmal die verwendete Fenstergröße vergrößert wird.

Abb. 3: Überlappung zwischen den Segmenten im Spektrogram wird kleiner gewählt

Abb. 4: verwendete Fensterfolge wird größer gewählt

Innerhalb des gewählten Fensters, kann das verwendete Signal als stationär angenommen werden. Wird das Beobachtungsfenster kürzer, nimmt auch die Frequenzauflösung ab. Wird das Fenster zu groß, kann wiederum das Signal innerhalb des Fensters nicht mehr als stationär angenommen werden. Die Verschlechterung und Ungenauigkeiten im Spektrogramm kann man in den Plots oben gut erkennen.

1.1.2 Matlab-Funktion: Frequenzverlau über der Zeit

Als nächstes sollte eine Matlab-Funktion erstellt werden, die im Zeitbereichdie momentane Frequenz ermittelt. Dieser berechnete Frequenzverlauf über der Zeit sollte geplottet und mit dem erwarteten Verlauf verglichen werden. In die erstellte Funktion kann in den Codes am Ende des Protokolls eingesehen werden. Der entstandene Plot ist hier zu sehen:

Abb. 5: Chirp-Signal und der dazugehörige Frequenzanstieg, berechnet aus dem Zeitsignal

Da beim Erstellen des Chirpsignals ein linearer Ansteig der Frequenz vorausgesetzt war, wurde auch ein linearer Frequenzverlauf erwartet. Das Ergebnis bestätigt die Erwartung.

1.1.3 Matlab-Funktion: Frequenzverlauf anhand Spektrogram

Nun sollte der gleiche Frequenzverlauf anhand des Spektrogramms ermittelt werden. Dafür implementierten wir eine weitere Matlab-Funktion, welche in den Codes zu sehen ist. Das erwartete Ergebnis war wieder ein positiver linearer Anstieg.

Abb. 6: Chirp-Signal und der dazugehörige Frequenzanstieg, berechnet aus dem Spektrogramm

1.1.4 Drehzahl-Berechnung anhand Amplitudenspektrum des Motorstroms

Zuletzt wurde die Drehzahl des verwendeten Motors berechnet. Dafür wurden uns drei Messreihen mit jeweils Strom- und Tachomesswerten vorgegeben. Zunächst sollten die Amplitudenspektren der Motorströme erstellt werden, welche uns durch die DFT der Stromwerte gelang. Die erhöhten Amplitudenwerte entsprachen dabei den einfachen Vielfachen der Drehfrequenz des Motors mit der jeweiligen Versorgungsspannung. Die deutlich herausstehenden Peaks dagegen traten nur bei dem 18- oder 36-fachen Vielfachen der Drehfrequenz auf. Dieses Wissen nutzen wir, indem wir anhand eines Matlabalgorithmuses den höhsten Peak mit Index ausgaben lassen (die höchsten Peaks waren stets jene, die am nächsten zu der null standen), um daraus eine Frequenz zu bestimmen und durch 18 zu teilen. So berechneten wir die drei geforderten Drehfrequenzen. Der Algorithmus steht in den Codes, die drei Amplitudenspektren sehen folgendermaßen aus:

Abb. 7: Spektrum des Motorstroms bei 10V

Abb. 8: Spektrum des Motorstroms bei 20V

Abb. 9: Spektrum des Motorstroms bei 30V

Die berechneten Drehfrequenzen betragen für die erste Messung $(10V) = 2.1389Hz$, für die zweite Messung $(20V) = 6.3056Hz$ und für die dritte Messung $(30V) = 10.4723Hz$.

TODO:

Hier sind die Werte falsch, weil wir die falschen Paeks rausgesucht haben, Fehler muss noch korrigiert werden

Außerdem wurden die Drehfrequenzen auch aus den jeweiligen Tachosignalen bestimmt. Dafür wurde genauso vogegangen wie bei den Motorströmen. Die Spektren sind hier:

Abb. 10: Spektrum des Tachosignals bei 10V

Abb. 11: Spektrum des Tachosignals bei 20V

Abb. 12: Spektrum des Tachosignals bei 30V

Die anhand der Tachosignale berechneten Drehfrequenzen betragen für die erste Messung $(10V)$ = $19.5627Hz$, für die zweite Messung $(20V)$ = $56.8131Hz$ und für die dritte Messung $(30V)$ = $94.3134Hz$.
Es fällt auf, dass die Drehzahlen, welche über die Ströme berechnet werden, ca. 4mal kleiner sind als die Drehzahlen über die Tachowerte. Ein konstanter Faktor von 4 kann angenommen werden.

1.2 Vorbereitungsaufgaben zu Termin 8

1.2.1 Zerlegung des Signals mittels Haar-Tranformation

In der ersten Vorbereitungsaufgabe des 8.Praktikumstermins wird eine vorgegebene strom.m Datei verwendet, welche einen angechnittenen Sinus im Bereich $[0, 8\pi]$ darstellt. Dieses Signal wird mit der Schnellen Haar-Transformation zerlegt. Außerdem wird die Approximation und die Details dür die Skalierungen $m = 1 \dots 5$ berechnet werden, wofür die drei Funktionen haardec.m, haardeclevel.m und getAppDet.m implementiert werden. Diese stehen unter den Codes.

1.2.2 Darstellung der Approximationen

Als nächstes werden die Approximationen und die Details des angeschnittenen Sinus-Signals dargestellt und das stationäre Spektrum sowie das Spektrogramm berechnet. Diese spektralen Darstellungen werden mit den Darstellungen mittels Wavelets verglichen. Am Ende soll ermittelt werden, welche Darstellung bestimmte Informationen über das verwendete Signal besser veranschaulicht.

TODO:

entsprechende Ergebnisse einfügen

1.2.3 Daubechies-Wavelets

Nun wird das angeschnitte Sinus-Signal mit Hilfe von Daubechies-Wavelets zerlegt. Dazu verwenden wir die Matlab-Fuktionen wavedec, appcoef und detcoaf und variieren die Anzahl der verschwindenen Momente der Wavelets.

TODO:

Ergebnisse einfügen

1.2.4 Vergleich der Zerlegungen mittels Daubechies-Wavelets und Haar-Wavelets

Zuletzt wird in der Vorbereitung des Praktikums die Zerlegungen mittels Daubechies-Wavelets mit den Zerlegungen mittels Haar-Wavelets verglichen. Die Unterschiede sind folgende:

TODO:

Unterschiede untersuchen und einfügen

2 Durchführungen

2.1 Durchführung zu Termin 7

2.2 Durchführung zu Termin 8

3 Auswertung

3.1 Auswertung Termin 7

3.2 Auswertung Termin 8

4 Quellcodes

4.1 Codes aus Termin 7

4.1.1 Frequenzverlauf über der Zeit

Listing 1: frequenz$_{i}mZeitbereich_{a}usSignal$

```
%Funktion zum Errechnen der Frequenz aus dem Zeitsignal

function frequenz_imZeitbereich_ausSignal (u, t)

p = length(t);
q = max(t)/p;

i = 0;
for k = 1:1:length(u)-1      %i = Anzahl der Nulldurchgnge
    if (u(k) < 0 && u(k+1) >=0) || (u(k) > 0 && u(k+1) <=0)
        i = i+1;
```

```
        end
    end

    nulldurchgang = ones(1,i)*-9;      %Vektor der L nge i

    j = 1;
    for indx = 1:1:length(u)-1
        if (u(indx) < 0 && u(indx+1) >=0) || (u(indx) > 0 && u(indx+1) <=0)
            if abs(u(indx)) > abs(u(indx+1))
                nulldurchgang(j) = indx+1;
            else
                nulldurchgang(j) = indx;          %Index, welcher am  n chsten an
            end
            j = j+1;
        end
    end

    %Vektor f r Frequenzeintrge
    v = length(nulldurchgang);
    frequenzen = ones(1,v)*-9;

    for n = 1:1:v-4                    %4 Nulldurchgnge, also 2 Perioden werden
        diff = nulldurchgang(n+4)-nulldurchgang(n);
        frequenzen(n) = 1/(diff*0.5*q);
    end

    figure(1);
    [AX H1 H2] = plotyy(t,u,nulldurchgang*(max(t)/p),frequenzen,'plot','stem')
    % AXIS([0 2 -1.1 1.1]);
    xlabel('Zeitachse [s]');
    set(get(AX(1),'Ylabel'),'String','Amplitude [V]');
    set(get(AX(2),'Ylabel'),'String','Frequenz [Hz]');
    title('\bf u-Signal und Frequenzanstieg  ber die Zeit');
    end
```

4.1.2 Frequenzverlauf aus dem Spektrogram

Listing 2: frequenz$_{durchspektogramm}$

```
% Funktion zur Errechnung der Frequenzen durch das Spektogramm

function frequenz_durch_Spektogramm(x_t,t)


N=length(t);
fs=N/max(t);

wnsize=256;
wnoverlap= 250;
nr_abtastwerte_frequenz= 256;

[S,F,T]=spectrogram(x_t,wnsize,wnoverlap,nr_abtastwerte_frequenz,fs);

%spectrogram(x_t,wnsize,wnoverlap,nr_abtastwerte_frequenz,fs);
```

```
groesse = size(S);
maxfrequ = ones(1,groesse(1))*-9;
maxfrequ_umgerechnet = ones(1,groesse(1))*-9;
maxindx = ones(1,groesse(1))*-8;
maxindx_umgerechnet = ones(1,groesse(1))*-8;

for i=1:groesse(2)
    [maxfrequ(i) maxindx(i)] = max(abs(S(:,i)));
    maxfrequ_umgerechnet(i) = F(round(maxindx(i)+1));
    maxindx_umgerechnet(i) = T(round(i));
end

maxindx_umgerechnet

figure(204);
[AX H1 H2] = plotyy(t,x_t,maxindx_umgerechnet,maxfrequ_umgerechnet,'plot'
% AXIS([0 2 -1.1 1.1]);
xlabel('Zeitachse [s]');
set(get(AX(1),'Ylabel'),'String','Amplitude [V]');
set(get(AX(2),'Ylabel'),'String','Frequenz [Hz]');
title('\bf Chirp-Signal und Frequenzanstieg ber die Zeit');
end
```

4.1.3 Algorithmus zur Drehfrequenzberechnung

Listing 3: Algorithmus zur Drehfrequenzberechnung

```
%MDV Praktikum 7 Vorbereitungsaufgabe 4 - Testprogramm

clear all; clc; close all;

%Messwerte laden
messung1 = load('MotorStrom_10V_100kS.mat');
messung2 = load('MotorStrom_20V_100kS.mat');
messung3 = load('MotorStrom_30V_100kS.mat');

% S t r me und Tachos identifizieren
strom1 = messung1.strom;
tacho1 = messung1.tacho;
strom2 = messung2.strom;
tacho2 = messung2.tacho;
strom3 = messung3.strom;
tacho3 = messung3.tacho;

%Versorgungsspannungen
A1 = 10;
A2 = 20;
A3 = 30;

f_T = 100000;
T_ges = 1/f_T;

%plottet Spektren der str me
```

```
[y_DFT_abs_10V_strom f_DFT_10V_strom]= MotorStrom_Amplitudenspektrum(strom
[y_DFT_abs_20V_strom f_DFT_20V_strom]= MotorStrom_Amplitudenspektrum(strom
[y_DFT_abs_30V_strom f_DFT_30V_strom]= MotorStrom_Amplitudenspektrum(strom

%findet Index vom h chsten Peak
[maxwert10_strom maxind10_strom] = max(y_DFT_abs_10V_strom);
[maxwert20_strom maxind20_strom] = max(y_DFT_abs_20V_strom);
[maxwert30_strom maxind30_strom] = max(y_DFT_abs_30V_strom);

%berechnet entsprechende Drehzahl
Drehzahl_Motor_10V_strom = abs(f_DFT_10V_strom(maxind10_strom))/18
Drehzahl_Motor_20V_strom = abs(f_DFT_20V_strom(maxind20_strom))/18
Drehzahl_Motor_30V_strom = abs(f_DFT_30V_strom(maxind30_strom))/18

%plottet Spektren der tachos
[y_DFT_abs_10V_tacho f_DFT_10V_tacho]= MotorStrom_Amplitudenspektrum(tacho
[y_DFT_abs_20V_tacho f_DFT_20V_tacho]= MotorStrom_Amplitudenspektrum(tacho
[y_DFT_abs_30V_tacho f_DFT_30V_tacho]= MotorStrom_Amplitudenspektrum(tacho

%findet Index vom h chsten Peak
[maxwert10_tacho maxind10_tacho] = max(y_DFT_abs_10V_tacho);
[maxwert20_tacho maxind20_tacho] = max(y_DFT_abs_20V_tacho);
[maxwert30_tacho maxind30_tacho] = max(y_DFT_abs_30V_tacho);


%berechnet entsprechende Drehzahl
Drehzahl_Motor_10V_tacho = abs(f_DFT_10V_tacho(maxind10_tacho))/8
Drehzahl_Motor_20V_tacho = abs(f_DFT_20V_tacho(maxind20_tacho))/8
Drehzahl_Motor_30V_tacho = abs(f_DFT_30V_tacho(maxind30_tacho))/8


% Drehzahl_Motor_10V_tacho/Drehzahl_Motor_10V_strom
% Drehzahl_Motor_20V_tacho/Drehzahl_Motor_20V_strom
% Drehzahl_Motor_30V_tacho/Drehzahl_Motor_30V_strom

Drehzahl_Motor_10V_tacho/Drehzahl_Motor_10V_strom
Drehzahl_Motor_20V_tacho/Drehzahl_Motor_20V_strom
Drehzahl_Motor_30V_tacho/Drehzahl_Motor_30V_strom
```

4.2 Codes aus Termin 8

4.2.1 Funktion haardec.m

Listing 4: Funktion haardec.m

```
%MDV Praktikum 8 Vorbereitungsaufgabe 1
%Funktion haardec

function [u, v] = haardec(x)

% f hrt einen Zerlegungsschritt der schnellen Haartransformation durch
% input :       x ? zu zerlegendes Signal
% output :      u ? Approximationen
%               v ? Details
```

```
% Bemerkung :   u,v sind halb so lang wie x


%Haar−Matrix sieht wie folgt aus:
% haar_matrix = ones(2,2);
% haar_matrix(4) = −1;
% haar_matrix = haar_matrix*(1/sqrt(2));

n = length(x);              %L nge des Eingangssignals

u_u = x(1:2:n−1);           %ungerade Komponenten des Signals
u_g = x(2:2:n);             %gerade Komponenten des Signals

u = (u_u + u_g)/sqrt(2);    %Approximationen
v = (u_u − u_g)/sqrt(2);    %Details

plot(x)
hold on
plot(u, 'r')
hold on
plot(v, 'g')
hold off

end
```

4.2.2 Funktion haardeclevel.m

Listing 5: Funktion haardeclevel.m

```
%MDV Praktikum 8 Vorbereitungsaufgabe 1
%Funktion haardeclevel

function [S] = haardeclevel(x, lvl)

% f hrt die Schnelle Haar−Transformation bis zu einem
% vorgegebenen Sakalierungslevel durch
% input :       x − zu zerlegendes Signal
%              lvl − Skalierungslevel
% output :      S − Matrix mit Skalierungen und Details
%                   Dimensionen: lvl+1:Signall nge
% Bemerkung :   Jede Zeile enth lt die Approximationen gefolgt von den
%               Details eines Levels
%               Skalierungslevel ist eine Potenz von zwei = ganzzahlig

n1 = length(x);
S = ones(lvl+1, n1)*−4;          %Hat lvl+1 Zeilen und n Spalten
u = x;

S(1,:) = x;                      %erste Zeile ist das Originalsignal

for i=2:(lvl+1)
    [u, v] = haardec(u)
    n2 = length(u);
    S(lvl,:) = [u, v, S(i−1,((2*n2)+1):n1)];
```

```
end
```

4.2.3 Funktion getAppDet.m

Listing 6: Fuktion getAppDet.m

```
%MDV Praktikum 8 Vorbereitungsaufgabe 1
%Funktion getAppDet

 function [u, v] = getAppDet(S, lvl)

% extrahiert die Approximaionen und Details eines Levels
% input :       S - Matrix mit Signalzerlegung
%              lvl - Skalierungslevel
% output :      u - Aproximationen
%               v - Details
% Bemerkung :   Jede Zeile in S enthlt die Approximationen gefolgt von de
%               Details eines Levels
%               Skalierungslevel ist eine Potenz von zwei = ganzzahlig >0

m = length(S(lvl ,:));       %gibt Lnge der lvl. Zeile wieder


    u = S((lvl+1) , (1:m/(2^lvl)));
    v = S((lvl+1) , ((m/(2^lvl))+1):(m/(2^lvl))*2);

%   v = S((lvl+1) , (m/(2^lvl))+1:m)
%Details aus der gesamten Zeile
der Zeile


% disp(['Approximationen u im gewhlten Level ',num2str(lvl)
 ' : ',num2str(u)]);
% disp(['Details v im gewhlten Level ',num2str(lvl) ' : ',num2str(v)]);

end
```