# Modul 5033 Digitale Signalverarbeitung Dozent: Rolf Vetter

# 

### *Praktische Übung No3*

# Digitale Spektralanalyse

### Ziel

* Verstehen der Beziehung zwischen der Fourier-Reihe und der diskreten Fourier-Transformation
* Umsetzten der Spektralanalyse mit Hilfe der schnellen Fourier-Transformation (FFT)
* Kenntnisnahme des Fensterungseffekt in der Spektralanalyse
* Hervorheben der Relevanz des Leistungsdichtespektrums als Hilfsmittel zur Analyse verrauschter Signale

### Übungen

### Von der Fourier-Reihe zur diskreten Fourier-Transformation

1. Schreiben Sie eine MATLAB Funktion eval\_FR(x, fa, fo, N\_harm), welche das Betrags- und Phasenspektrum des Signals x mittels einer komplexen Fourier-Reihe für N\_harm Harmonische (for Schleife und einem Skalarprodukt) ermittelt und darstellt.
   1. Schreiben Sie eine Testskript test\_eval\_FR und überprüfen Sie Ihre Funktion mit einem periodischen Sinus-, Cosinus- und Rechtecksignal. Vergleich mit Theorie? Diskussion.
   2. Modifizieren Sie die Funktion eval\_FR,so dass die resultierende Funktion eine diskrete Fourier Transformation ermittelt und darstellt (Aufruf eval\_DFT(x, fa)). Schreiben Sie ein Testskript test\_eval\_DFT und überprüfen Sie Ihre Funktion mit einem periodischen Sinus-, Cosinus-, und Rechtecksignal und mit einem nicht periodischen Rechteckimpuls. Vergleich mit Theorie? Diskussion.
   3. Fakultativ: Implementieren Sie eine Funktion, welche die DFT mittels einer Matrixoperation ermittelt.

### Fensterung und FFT

1. Entwerfen Sie eine Simulation illustration\_fenster\_fft mit Hilfe der Funktion fft() von MATLAB, die Ihnen erlaubt, den Nutzen der Fenster in der digitalen Spektralanalyse wie folgt hervorzuheben:
   1. Erzeugen Sie zwei harmonische Signale mit den normierten Frequenzen 0.07 und 0.27, einer Anzahl Samples N=50 und einer Amplitudendifferenz von 40dB. Verwenden Sie ein Rechteckfenster (boxcar()) und stellen Sie das Betragsspektrum dar (Funktion fft()). Was stellen Sie fest?
   2. Wiederholen Sie dieselbe Übung mit einem Hanning Fenster. Diskussion.
   3. Wiederholen Sie dieselbe Übung mit einer Amplitudendifferenz der Sinussignale von 80dB und vergleichen Sie die Hanning und Kaiser Fenster. Diskussion, Schlussfolgerung.

### Leistungsdichteschätzung

1. Das Ziel dieser Übung ist das Hervorheben der Relevanz der Leistungsdichteschätzung als Hilfsmittel der digitalen Spektralanalyse. Man lade daher 100 Realisierungen eines verrauschten harmonischen Signals durch den Befehl load HarmNoise01.mat in MATLAB hoch (Abtastfrequenz fa=8000Hz). Man entwickle, teste und validiere ein Algorithmus zur zuverlässigen Schätzung der Frequenz des verrauschten Signals nach folgendem Vorgehen:
   1. Stellen Sie das Konzeptschema des Algorithmus dar.
   2. Wählen Sie das adäquate Hilfsmittel der digitalen Spektralschätzung. Begründen Sie Ihre Wahl.
   3. Programmieren und testen Sie den Algorithmus.
   4. Validieren Sie den Algorithmus auf den 100 Realisierungen des stochastischen Signals. Stellen Sie als Performanceindex die Häufigkeitstabelle in Bezug auf die geschätzte Frequenz dar. Feststellungen?
   5. Diskussion, Schlussfolgerung.

Ergebnisse

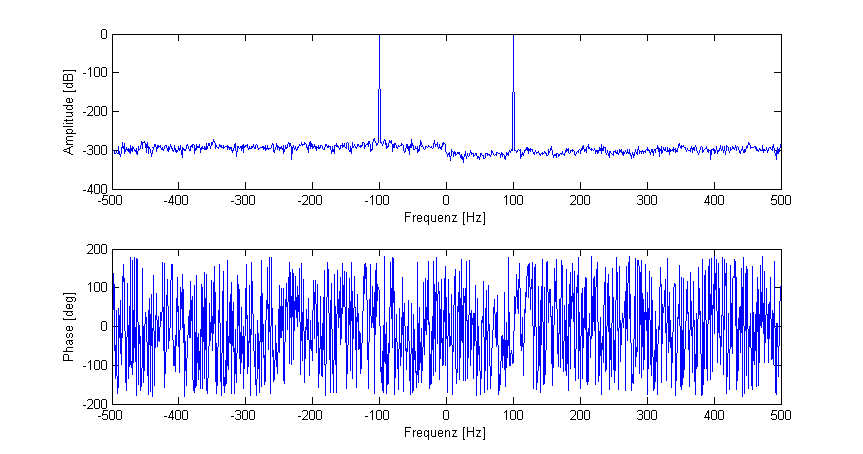
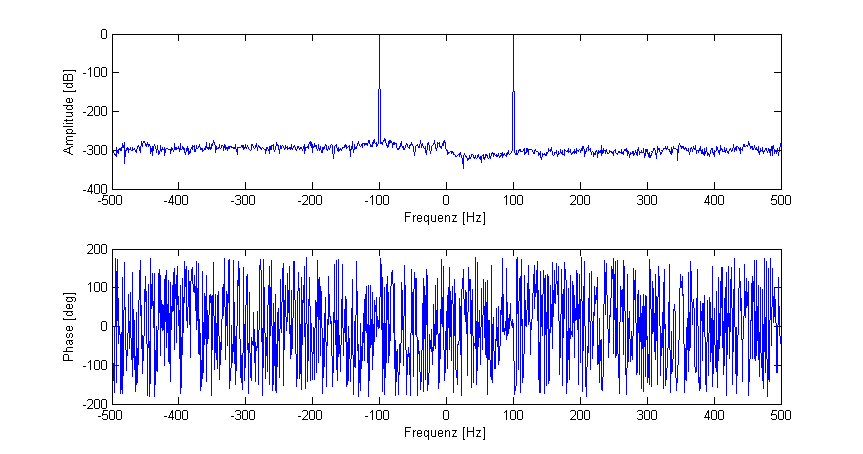
# Aufgabe 1

Matlab Code für DFT mit unterschiedlichen Signalen

|  |
| --- |
| %% Vorbereitung  fa = 1000;  f0 = 100;  N = 1000;  df = fa/N;    t = [0:N-1]'/fa;    %% Signale    x = sin(2\*pi\*f0\*t); % Sinus    % x = cos(2\*pi\*f0\*t); % Cosinus    % for i=1:length(t) % Rechteck  % x(i) = mod(i, 40)<20;  % end  % x = x';    % x = [1E3 zeros(1, length(t)-1)]'; % Dirac    %% DFT mit forschleife  for k = 0:N-1  gf = exp(-1i\*2\*pi\*k\*df\*t);  Xf(k+1)= x'\*gf/N;  end  %% Darstellung  Xf = fftshift(Xf);    subplot(2,1,1); plot([-N/2:N/2-1]\*df, 20\*log10(abs(Xf)));  xlabel('Frequenz [Hz]');  ylabel('Amplitude [dB]');    subplot(2,1,2); plot([-N/2:N/2-1]\*df, 180/pi\*(angle(Xf)));  xlabel('Frequenz [Hz]');  ylabel('Phase [deg]'); |

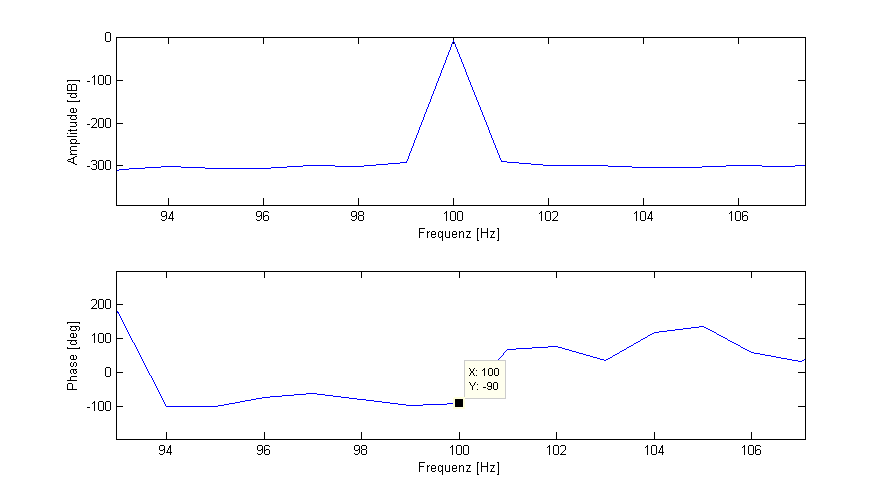
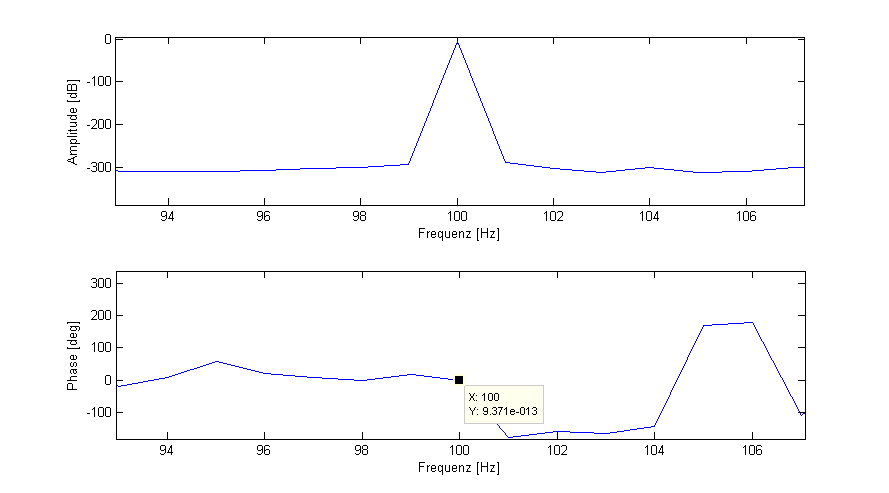
Ergebnis bei unterschiedlichen Signalen:

Sinus Cosinus

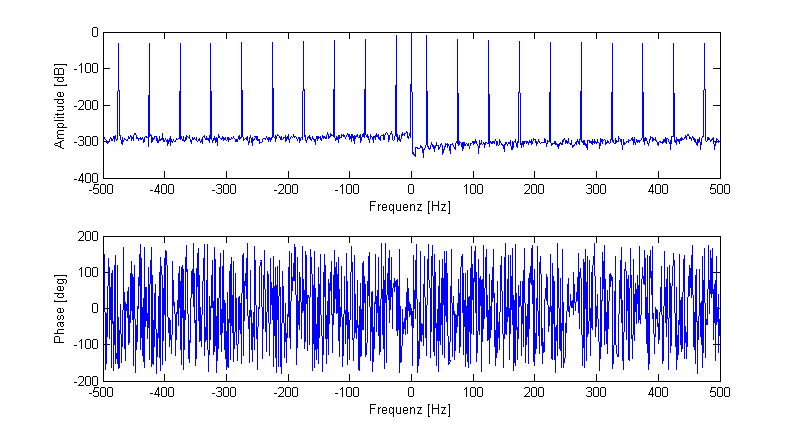
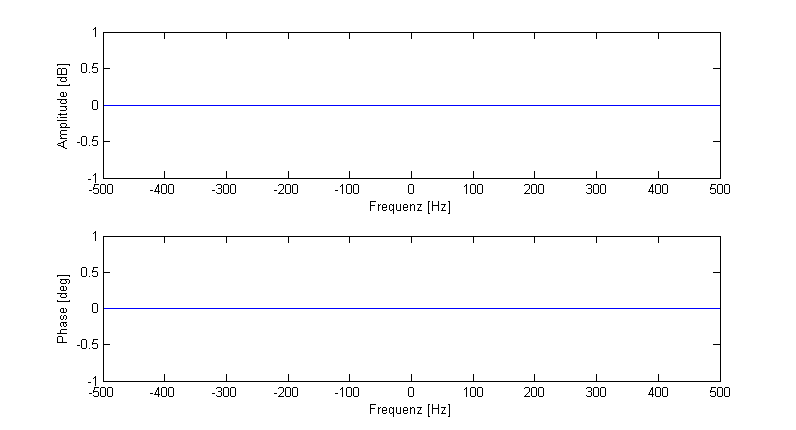
 

Noch mit Zoom, zum Darstellen der unterschiedlichen Phase beim positiven Träger:

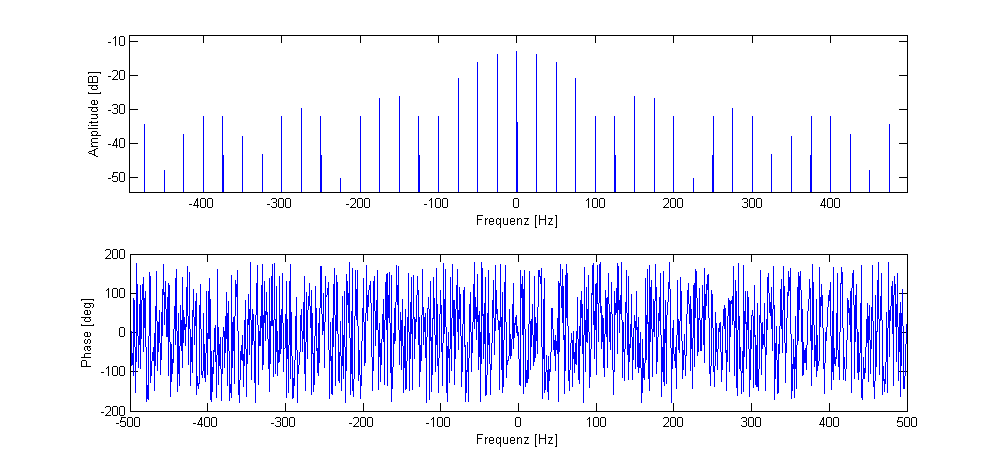
Sinus Cosinus

Rechteck Dirac Impuls

Unsymmetrischer Rechteck (9 zu 31)



Die Ergebnisse sind zufriedenstellend – Genau diese Diagramme haben wir erwartet.

Bemerkung: Amplitude mit **set(gca, `ylim`, [-80 0])** beschränken

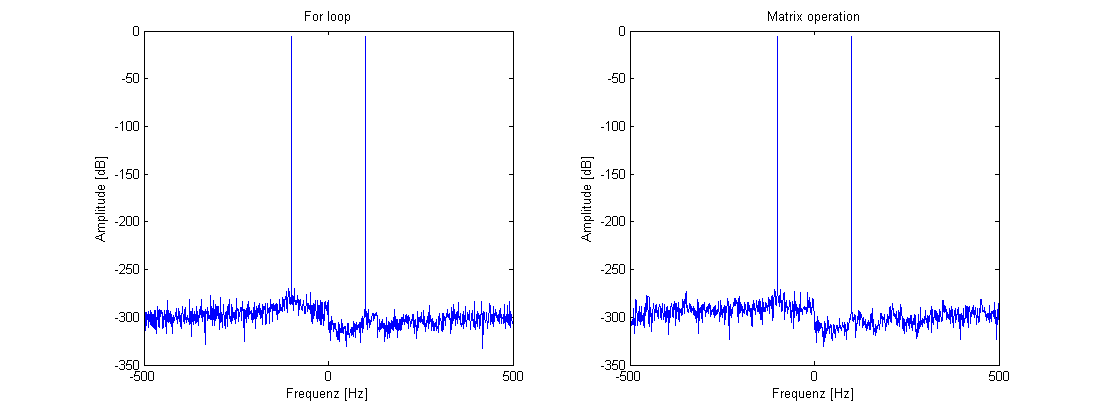
Bemerkung: Phase an nichtsaussagenden Stellen auf 0 setzen mit **X.\*(abs(X)>get(gca, `ylim`))** beschränken

Bemerkung: Figures von Matlab als EPS für LaTeX speichern: **print –depsxc <path>**

Matlab Code für DFT mit Matrixoperationen

|  |
| --- |
| %% DFT mit forschleife  for k = 0:N-1  gf = exp(-1i\*2\*pi\*k\*df\*t);  Xf(k+1)= x'\*gf/N;  end    %% DFT mit Matrixoperation  k = 0:N-1;  gf = exp(-1i\*2\*pi\*df\*t\*k);  Xf = x'\*gf/N; |

Ergebnis:



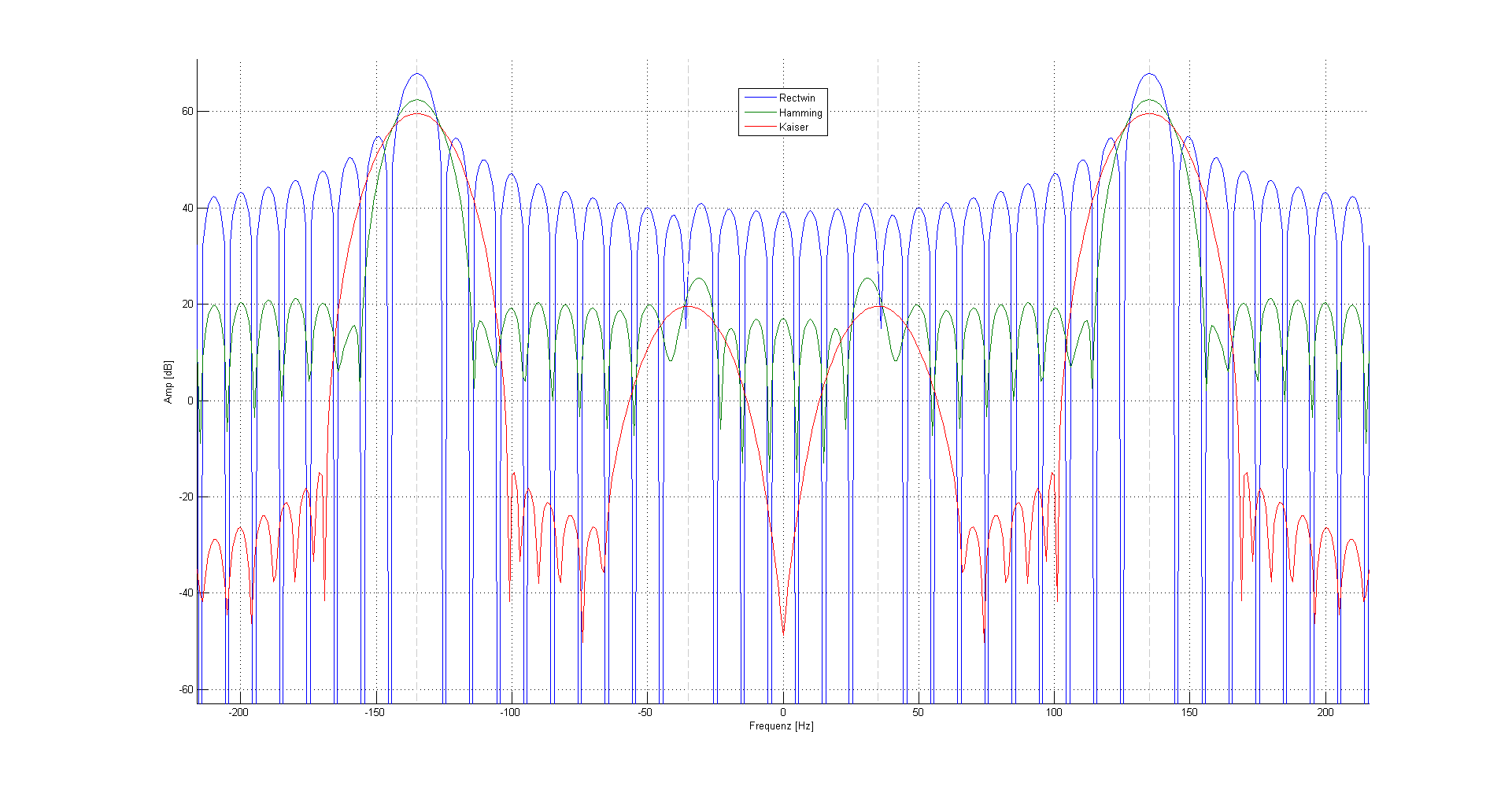
Interpretation: Das Ergebnis ist im Grunde genau gleich, der einzige Unterschied ist die „Form“ des Rauschens. Das Rauschen kommt hier (da eigentlich ein reines Sinussignal transformiert wurde) von der „Quantisierung“ eines Datenwerten (also Fliesskommazahl-Genauigkeit) von Matlab. Eigentlich sollte das Rauschen auf -∞ sein. Das Rauschen sieht bei den beiden Versionen anders aus, da die zu Grunde liegenden Rechnungsmethoden nicht genau gleich sind und deshalb an anderen Stellen gerundet wird.

# Aufgabe 2

Matlab Code für 2 Signale mit **40dB** Amplitudendifferenz und den Fenstern *Rechteck*, *Hamming* und *Kaiser* :

|  |
| --- |
| clear all;  close all;  grid on;    N = 50; % "Zeitvektor" mit 50 Samples  n = 0:N-1;  NFFT = 10\*N;    f0 = 1; % Grundfrequenz zum Normieren    x1 = sin(2\*pi\*0.07\*f0\*n); % Harmonisches Signal mit 0.07Hz  x2 = 100\*sin(2\*pi\*0.27\*f0\*n); % Harmonisches Signal mit 0.27Hz und 40dB grösser als das andere    x = x1 + x2;    % Mit Rechteckfenster  xrect = x.\*rectwin(N)';  xhann = x.\* hamming(N)';  xkais = x.\* kaiser(N, 10)';    % Signal in den Spektralbereich transformieren  X1 = fftshift(fft(xrect, NFFT));  X2 = fftshift(fft(xhann, NFFT));  X3 = fftshift(fft(xkais, NFFT));    % Signale darstellen  hold all;  plot([-NFFT/2:NFFT/2-1], 20\*log10(abs(X1)));  plot([-NFFT/2:NFFT/2-1], 20\*log10(abs(X2)));  plot([-NFFT/2:NFFT/2-1], 20\*log10(abs(X3)));      h = line([0.07\*f0\*NFFT 0.07\*f0\*NFFT], ylim);  h = [h line([-0.07\*f0\*NFFT -0.07\*f0\*NFFT], ylim)];  h = [h line([0.27\*f0\*NFFT 0.27\*f0\*NFFT], ylim)];  h = [h line([-0.27\*f0\*NFFT -0.27\*f0\*NFFT], ylim)];  set(h, 'LineStyle', '--', 'Color', [0.8 0.8 0.8]);    legend('Rectwin', 'Hamming', 'Kaiser');  xlabel('Frequenz [Hz]');  ylabel('Amp [dB]'); |

Ergebnis:



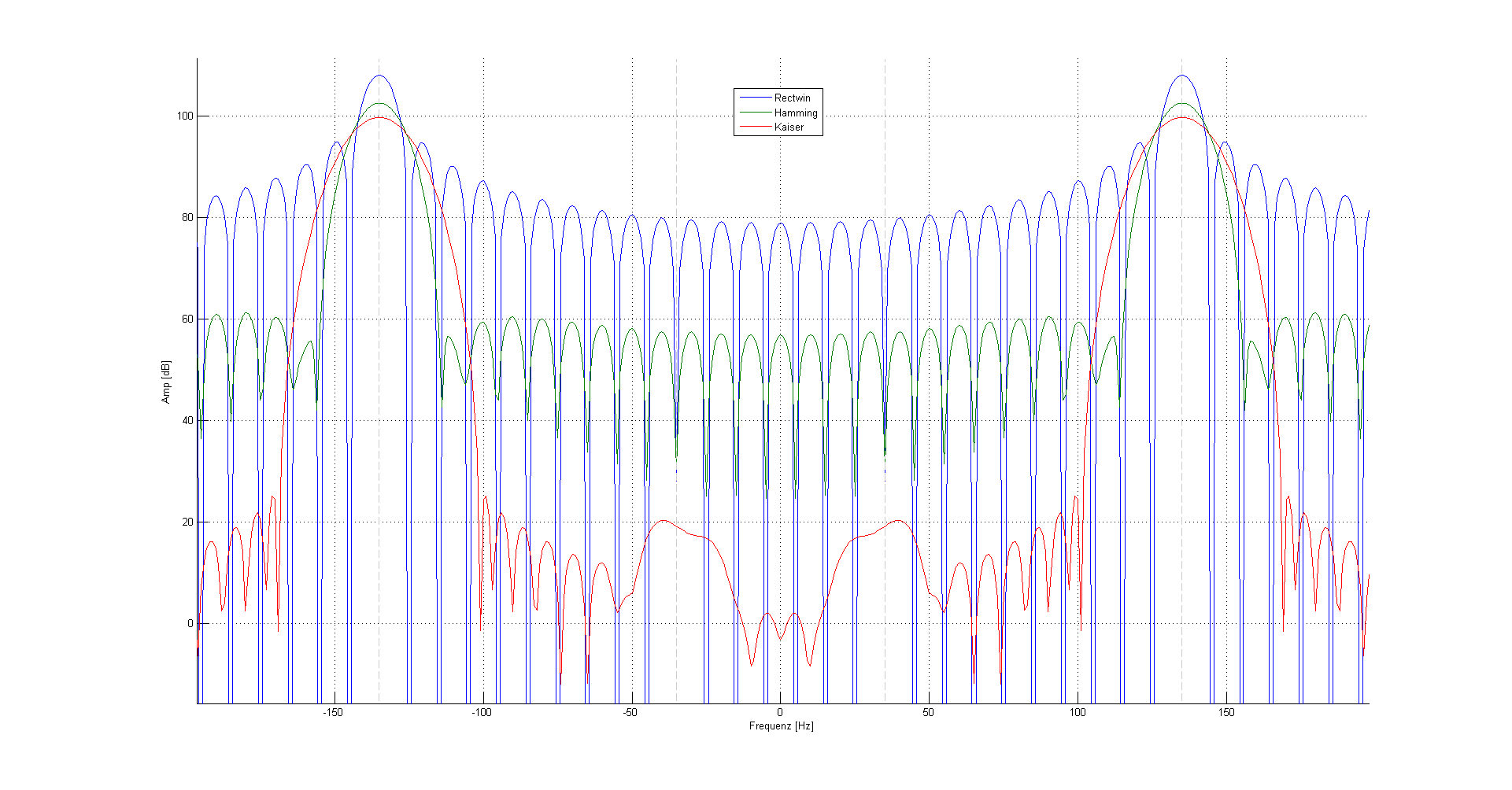
Interpretation:

Die verschiedenen Fenster haben unterschiedliche Charakteristiken. Die Nebenlappen sind beim Rechteckfenster am höchsten, das Hammingfenster ist schon besser und das Kaiserfenster (mit Parameter 10!) ist am besten. Die tiefe Frequenz bei 0.07 wird nur mit dem Kaiserfenster richtig hervorgehoben.

Matlab Code für 2 Signale mit **80dB** Amplitudendifferenz und den Fenstern *Rechteck*, *Hamming* und *Kaiser* :

|  |
| --- |
| *Die Zeile*  x2 = 100\*sin(2\*pi\*0.27\*f0\*n); % Harmonisches Signal mit 0.27Hz und 40dB grösser als das andere  *wurde geändert zu:*  x2 = 10000\*sin(2\*pi\*0.27\*f0\*n); % Harmonisches Signal mit 0.27Hz und 80dB grösser als das andere |

Ergebnis:

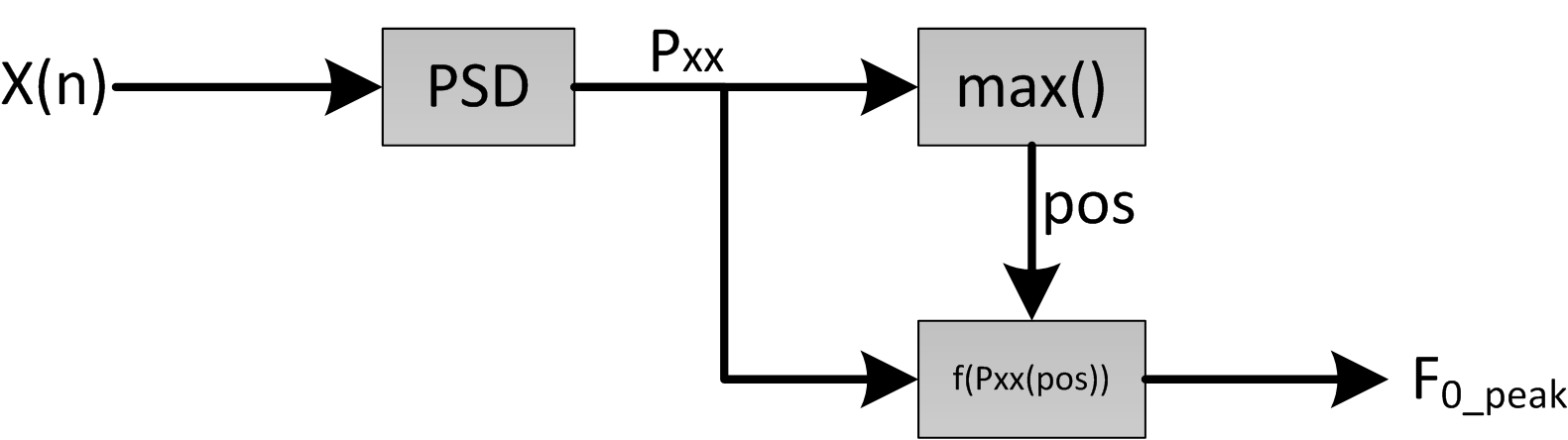


Interpretation:

Grundsätzlich sieht es gleich aus, aber die tiefe Frequenz bei 0.07 wird nun auch mit dem Kaiserfenster nur noch verschwommen sichtbar (so wie das Signal vorher mit dem Hammingfenster).

# Aufgabe 3

Um die Frequenz des stochastischen Signals zu bestimmen, gehen wir nach folgendem Ablauf des Algorithmus vor.

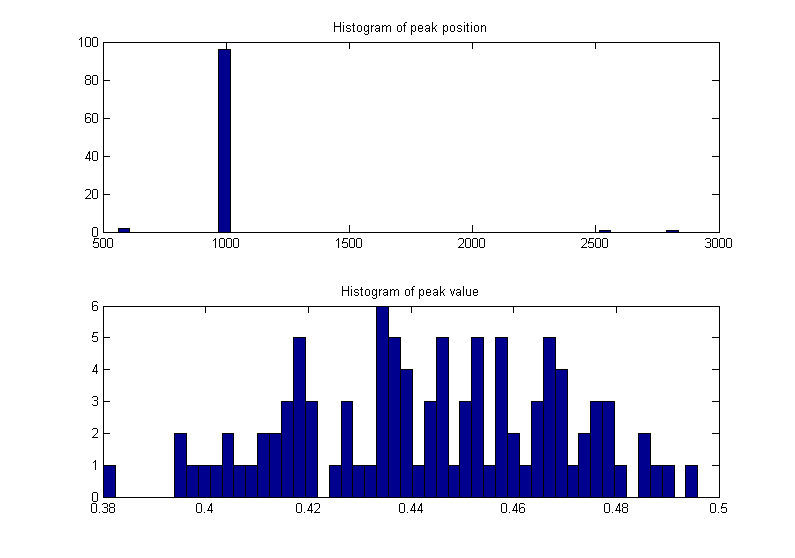


Das Signal wird mittels pwelch (PSD: power spectral density) Funktion (welche später parametrisiert wird) im Leistungsdichtespektrum dargestellt. Danach wird der Maximalwert gesucht und die Frequenz an dieser Stelle berechnet.

Matlab Code für Peak Detektion:

|  |
| --- |
| clear all;  close all;    fa = 8000;  dt = 1/fa;    x = load('HarmNoise01.mat');  x = x.x;  windowlength = 512;  overlap = ceil(windowlength/1.5);  window = kaiser(windowlength, 10);    tic  peakvalue = zeros(100,1);  peakpos = zeros(100,1);  for k = 1:100  [Pxx, f] = pwelch(x(:, k), window, overlap, fa);  [peakvalue(k), peakpos(k)] = max(Pxx);  end  toc    % Ausgabe geschätzte Frequenz  mean(peakpos)    figure  subplot(2,1,1); hist(peakpos, 50); title('Histogram of peak position');  subplot(2,1,2); hist(peakvalue, 50); title('Histogram of peak value'); |

Ausgabe :



Diskussion :

Damit der Algorithmus optimal funktioniert, müssen die Parameter richtig evaluiert werden. Dabei können folgende Werte eingestellt werden:

* Window: Art des Fensters, wir haben ein Kaiserfenster mit Beta = 10 gewählt
* Windowlength: Wir haben etwa einen Hundertstel der Signallänge genommen, nämlich 512.
* Overlap: Wie viel das Fenster überlappt. Da haben wir 2/3 der Fensterbreite (341) gewählt.

Mit der Analyse sieht man, dass die Schätzung in 96% der Fälle dasselbe Resultat ergibt, woraus man schliessen kann, dass es stimmt.

Die geschätzte Frequenz liegt bei 1025.4 Hz.

Um die Ausgabe zu berechnen brauchte Matlab etwa 16sec.

Wir können sagen, dass der Algorithmus gut funktioniert und wir mit der Methode der Leistungsdichteschätzung viel bessere Resultate erzeugen, als mittels fft.