Spektralmesstechnik

1. Diskrete Fourier Transformation

Mit der diskreten Fourier Transformation (DFT)

$$F(f') = \frac{1}{N} \sum_{i=0}^{N-1} z(i) \cdot e^{-j2p f' i} \tag{1.1}$$

lässt sich in einem digitalen Rechenwerk aus einer abgetasteten (diskreten) Zeitfunktion, das zugehörige Spektrum bestimmen. Im Gegensatz zur herkömmlichen Fourier Transformation muss das Zeitsignal für die DFT diskret vorliegen. Die DFT liefert dann ein diskretes, nur auf einem Frequenzraster definiertes, mit f_A periodisches Spektrum.

Mit der Abtastfrequenz f_A und der Anzahl an Abtastwerten N ergibt sich der Abstand der Spektrallinien zu

$$\Delta f = \frac{f_A}{N} \tag{1.2}$$

Dabei wird das Spektrum bezogen auf die normierte Frequenz f_A' im Bereich

$$-\frac{f_A'}{2} \le f < \frac{f_A'}{2} \tag{1.3}$$

berechnet.

Anmerkung: Die DFT liefert die Spektralwerte in der Reihenfolge $f_A'(0); \rightarrow; 0{,}5 - \frac{f_A'}{N}\ ;\ -0{,}5; \rightarrow; -\frac{f_A'}{N}$. Für die gewohnte nach Gleichung (1.3) beschriebene Darstellung muss der Ergebnisvektor der DFT erst umsortiert werden. In Matlab steht hierfür der Befehl `fftshift` zur Verfügung.

Aufgrund der endlichen Beobachtungsdauer, kann mit der DFT auf einem Digitalrechner immer nur die Kurzzeit- Fourier- Transformierte berechnet werden. Dabei wird Signal mit einer Fensterfunktion gewichtet. Die Auswirkungen dieser Fensterung werden in Kapitel 1.1 untersucht.

1.1. Berechnung des Spektrums mit Matlab

Im Folgendem soll das Spektrum zweier sinusförmigen Signale mit Matlab berechnet werden. Die Simulationsparameter sind Tabelle 1-1 zu entnehmen.

Abtastfrequenz f_A	1 Hz
Frequenz x_1	0,1 Hz
Frequenz x_2	0,12 Hz
Fensterbreite N	10
Δf	0,1 Hz

Tabelle 1-1

1.1.1. Matlab-Code (E2A.m)

```
x1 = sin(2 * pi * [0:9] * 0.1);
x2 = sin(2 * pi * [0:9] * 0.12);

figure(1);
plot(x1);
grid;
hold on;
plot(x2, '-.b');
hold off

fAchse = -0.5:1/length(x1):0.5-1/10;
figure(2);
stem(fAchse, abs(fftshift(fft(x1)/length(x1))));
grid; title('f = 0.1');
figure(3)
stem(fAchse, abs(fftshift(fft(x2)/length(x2))));
grid; title('f = 0.15');
```

1.1.2. Matlab-Plots

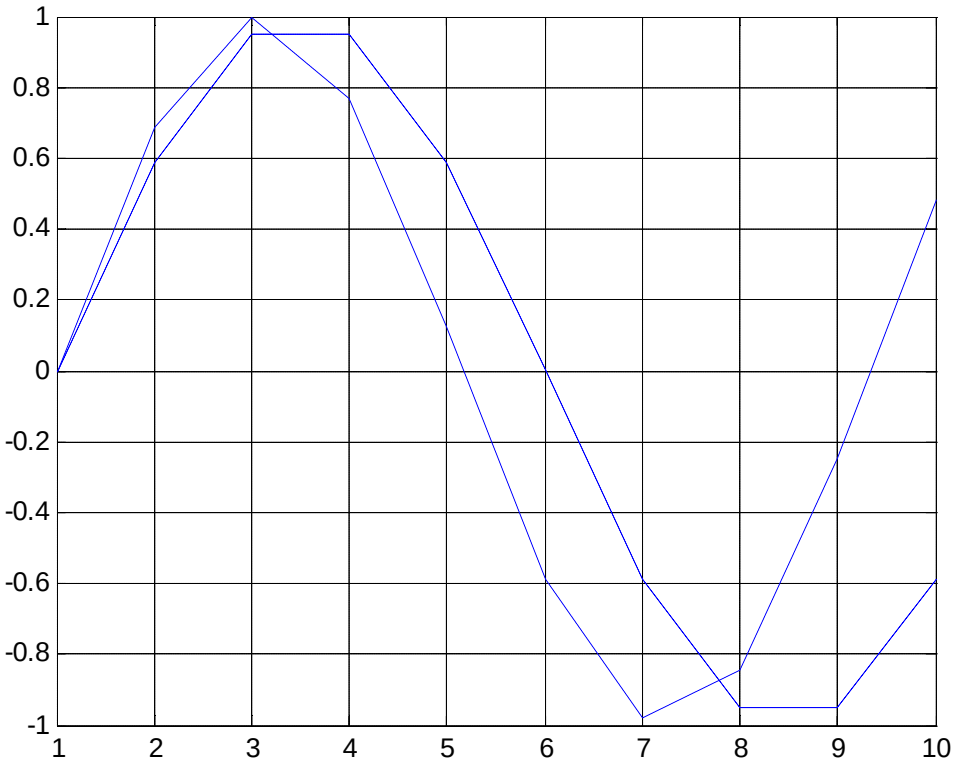


Abbildung 1-1

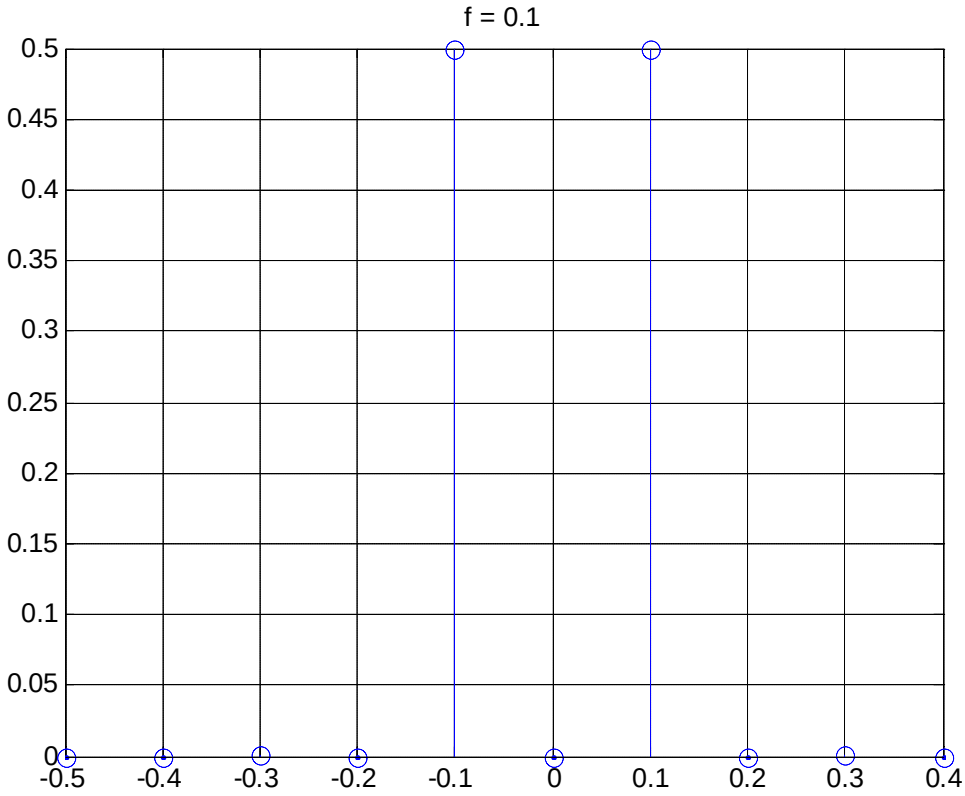


Abbildung 1-2

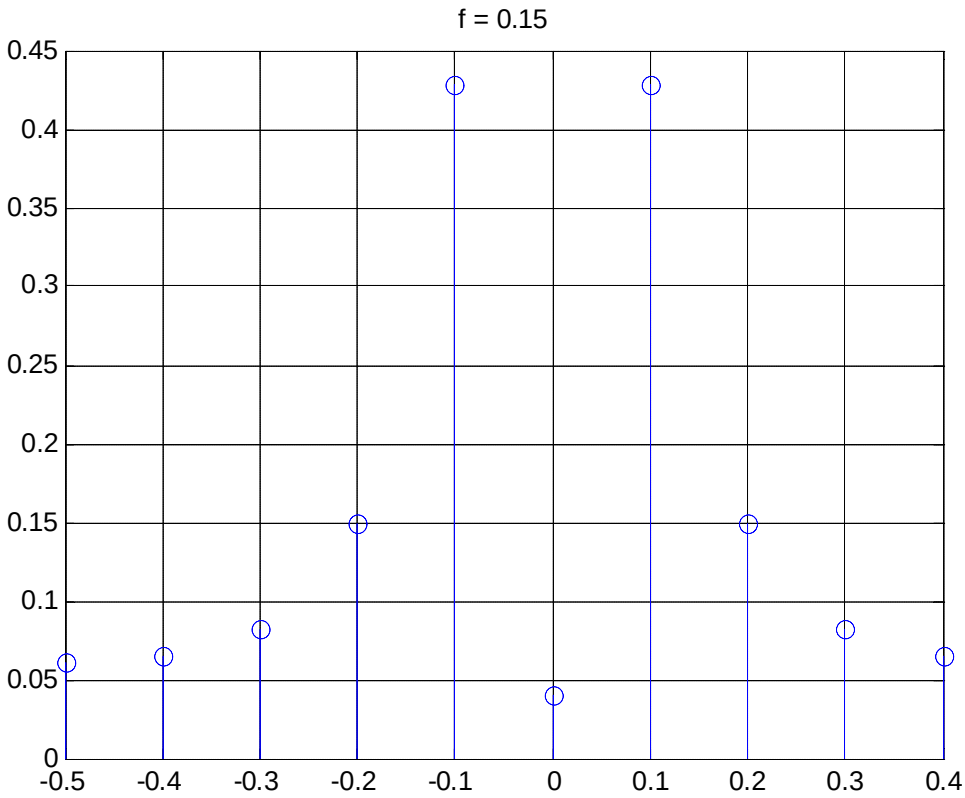


Abbildung 1-3

Die Frequenz des Signals x_2 liegt nicht im Frequenzraster der DFT. Das zugehörige Spektrum wird im Gegensatz zum Spektrum des Signals x_1, das nur Spektralanteile bei -0,1 Hz und 0,1 Hz aufweist, über alle Spektrallinien verteilt.

1.2. Interpolation durch Anfügen von Nullen

Um eine höhere Frequenzauflösung zu erhalten, muss die DFT über eine größere Anzahl von Abtastwerten errechnet werden. Liegen diese Abtastwert nicht vor, kann durch anfügen von Nullen die Frequenzauflösung erhöht werden.
Dieses Verfahren wird auch Zeropadding genannt.

$$z'(i) = \begin{cases} z(i) & ,0 \le n \le N-1 \\ 0 & ,N \le n \le N+M-1 \end{cases} \tag{1.4}$$

$$F(f') = \frac{1}{N} \sum_{i=0}^{N+M-1} z'(i) \cdot e^{-j2p\frac{f'i}{N+M}} = \frac{1}{N} \sum_{i=0}^{N-1} z(i) \cdot e^{-j2p\frac{f'i}{N+M}}$$

Zu beachten ist dabei, dass durch das Anfügen von Nullen kein weitere Informationsgehalt gewonnen wird. Lediglich die Einhüllende wird nun in einem feineren Raster abgetastet. Die Beschreibung der Einhüllenden erfolgt in Kapitel 1.3

1.2.1. Matlab-Code (E2a.m)

Fortetzung des Codes aus 1.1.1:

```
x3 = x2;
x3(40) = 0; % Anfuegen von Nullen

fAchse = -0.5:1/length(x3):0.5-1/length(x3);

figure(4);
stem(fAchse, abs(fftshift(fft(x3))/length(x2)));
grid
```

1.2.2. Matlab-Plots

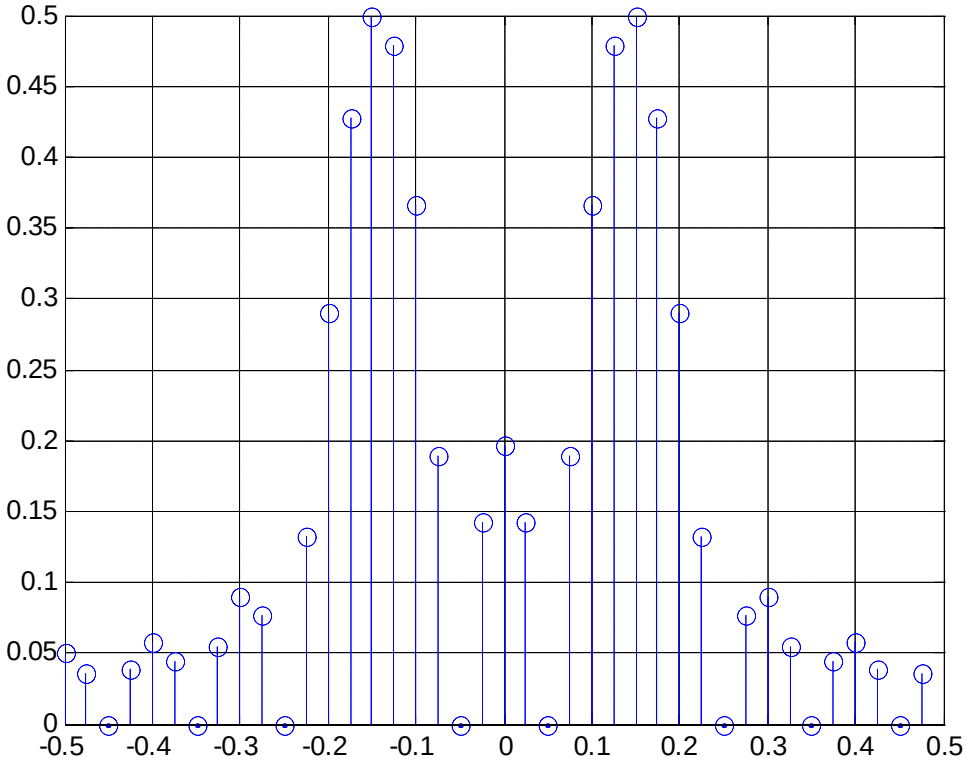


Abbildung 1-4

1.3. Leckeffekt und Lattenzauneffekt

In den Kapiteln 1.1 und 1.2 wurde bereits erwähnt, dass das Spektrum eines sinusförmigen Signals, dessen Frequenz nicht auf dem DFT- Frequenzraster liegt, über den gesamten errechneten spektralen Bereich „verschmiert“ wird. Diesen Effekt nennt man Leckeffekt.
Wie bereits erwähnt wurde, berechnet man mit der DFT in der Praxis die Kurzzeit-DFT. Das hat zur Folge, dass das zu analysierende Signal mit einer Fensterfunktion $w(i)$ bewertet wird.

$$F(f') = \frac{1}{N}\sum_{i=0}^{N-1} w(i)\cdot z(i)\cdot e^{-j2p\,f'i} \tag{1.5}$$

Im einfachsten Fall ist dies ein Rechteckfenster, dessen spektraler Verlauf sich zu

$$F_D(f') = \frac{1}{N}\frac{\sin\left(p\left[f' - f_0'\right]N\right)}{\sin\left(p\left[f' - f_0'\right]\right)} e^{-j\cdot p\cdot\left(f'-f_0'\right)\cdot(N-1)} \tag{1.6}$$

ergibt. Die Fensterfunktion wird im Zeitbereich mit der Signalfolge multipliziert. Insofern ist im Spektralbereich die Fensterfunktion mit dem Signal zu falten. Werden, so wie hier, sinusförmige Signale analisiert, ist die Faltung nach dem Verschiebungssatz der Fouriertransformation besonders einfach, da die Fensterfunktion nur an die Stelle der Signalfrequenzen verschoben wird.

1.3.1. Analyse

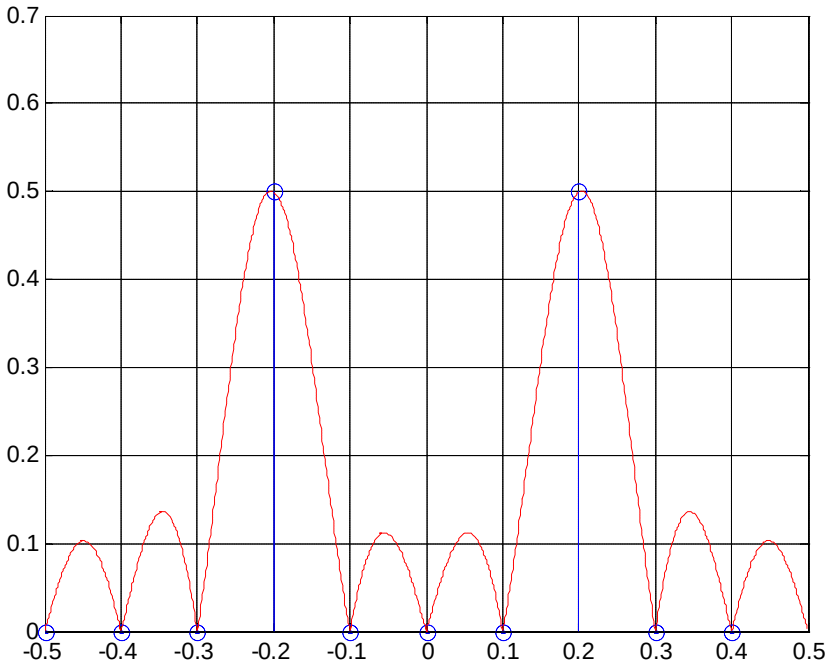


Abbildung 1-5

In Abbildung 1-5 ist sowohl das diskrete Spektrum eines sinusförmigen Signals mit f= 0,2 Hz, als auch die Einhüllende zu erkennen. Die Einhüllende wird im Maximum der Hauptkeule und ansonsten in ihren Nullstellen abgetastet.

Für eine Frequenz, die kein ganzes Vielfaches des Frequenzrasters nach (1.2) sind, ergibt sich ein Spektrum und die dazugehörige Einhüllende nach Abbildung 1-6.

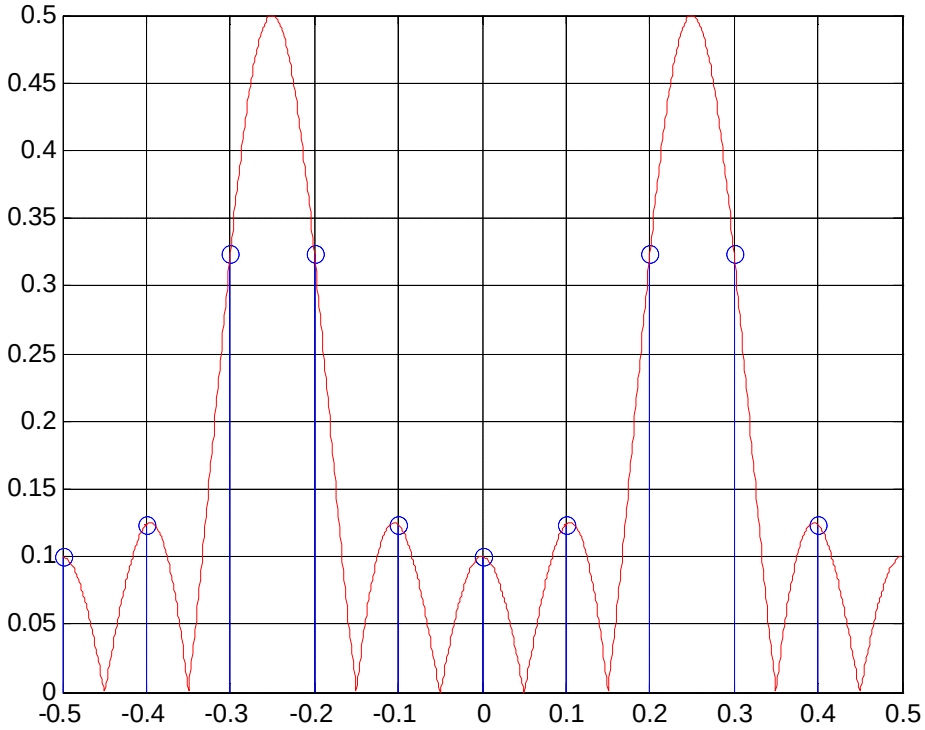


Abbildung 1-6

Die Maxima der Hauptkeulen und die Nullstellen liegen jeweils gerade genau zwischen zwei diskreten Spektrallinien. Das Signalspektrum „leckt“ durch den gesamten Analysebereich.

Aus Abbildung 1-7 ist das Spektrum zweier Additiv überlagerten sinusförmigen Schwingungen mit den Frequenzen $f_0 = 0{,}1$ Hz und $f_1 = 0{,}3$ Hz zu entnehmen. Die DFT liefert jeweils in den Maxima der Hauptkeulen und in den Nullstellen eine Spektrallinie und zeigt so das erwartete und richtige Ergebnis.

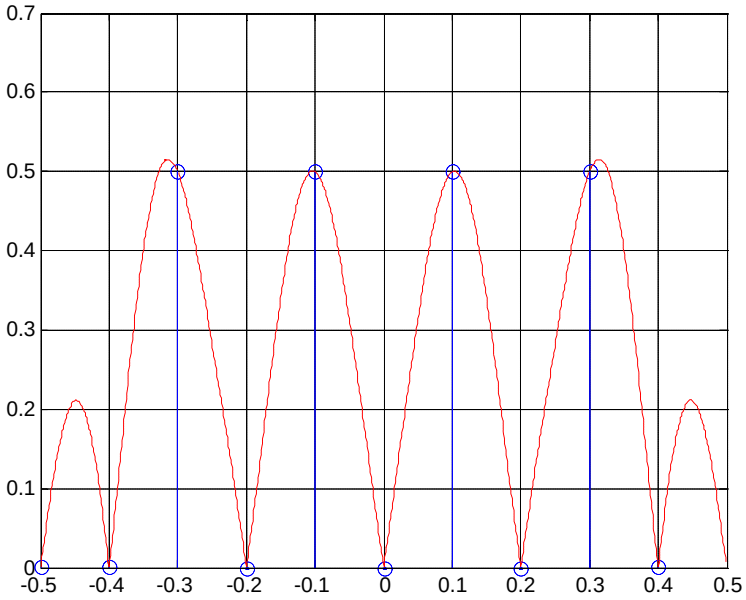


Abbildung 1-7

Wird nun eine der beiden Sinusschwingungen auf eine Frequenz gelegt die nicht in dem Frequenzraster der DFT liegt, verfälscht die Hüllkurve dieser Frequenz nicht nur ihren eigenen Spektralanteil, sondern auch die Spektrallinie der Frequenz die in einem ganzzahleigen Verhältnis zum Frequenzintervall steht.

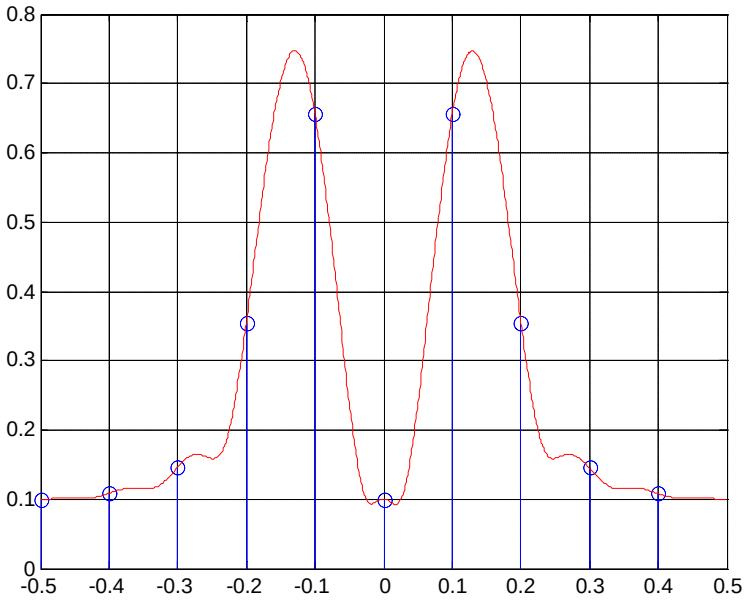


Abbildung 1-8 Spektrum zweier Sinusschwingungen mit $f_0 = 0{,}1$ Hz und $f_1 = 0{,}15$ Hz

1.3.2. Fensterfunktion und Zeropadding

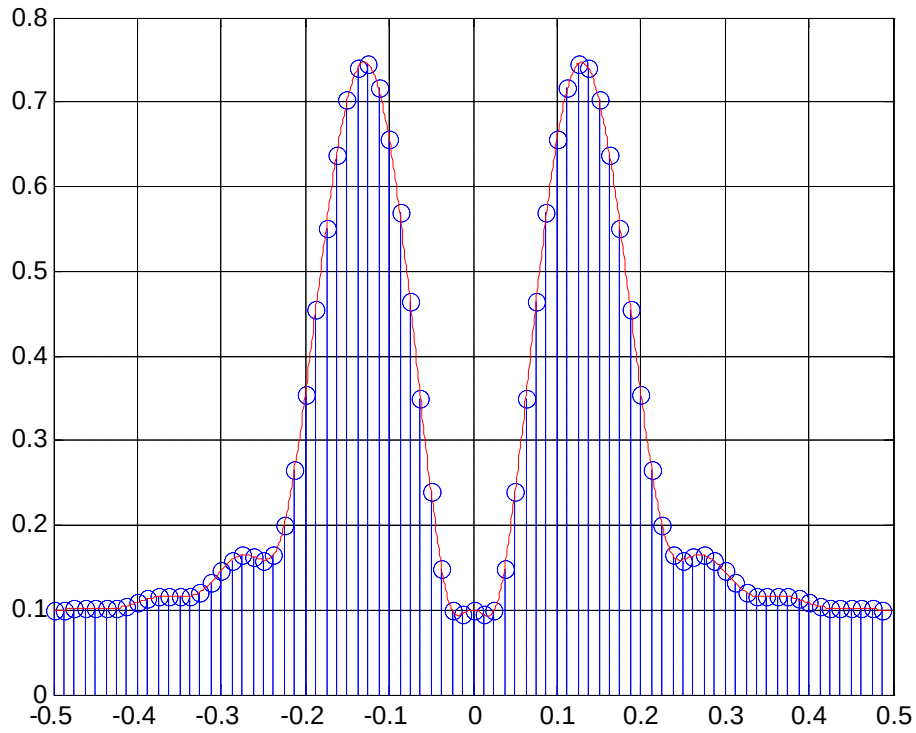


Abbildung 1-9 Spektrum zweier Sinusschwingungen mit $f_0 = 0{,}1$ Hz und $f_1 = 0{,}15$ Hz mit Zeropadding

In Abbildung 1-9 wird noch einmal, wie bereits in Kapitel 1.2 besprochen, die spektrale Auflösung durch Zeropadding verbessert. Durch das Anfügen von Nullen wird die **gefensterte** Funktion besser beschrieben. Man erhält also mit Zunahme der angefügten Nullen mehr Abtastwerte der Einhüllenden und das Spektrum zeigt immer deutlicher den Verlauf der Faltung zwischen Signal und Fensterfunktion.

1.3.3. Verminderung des Leckeffekts durch geeignete Fenster

Eine Verbesserung des Leckeffekts kann mit der Gewichtung durch andere Fensterfunktionen herbeigeführt werden. In Abbildung 1-10 wurde das Spektrum einmal mit einem Rechteckfenster und einmal mit einem Hanning-Fenster gewichtet.

Durch die Gewichtung mit dem Hanning-Fenster fallen die Nebenzipfel viel schneller ab als bei dem Rechteckfenster.

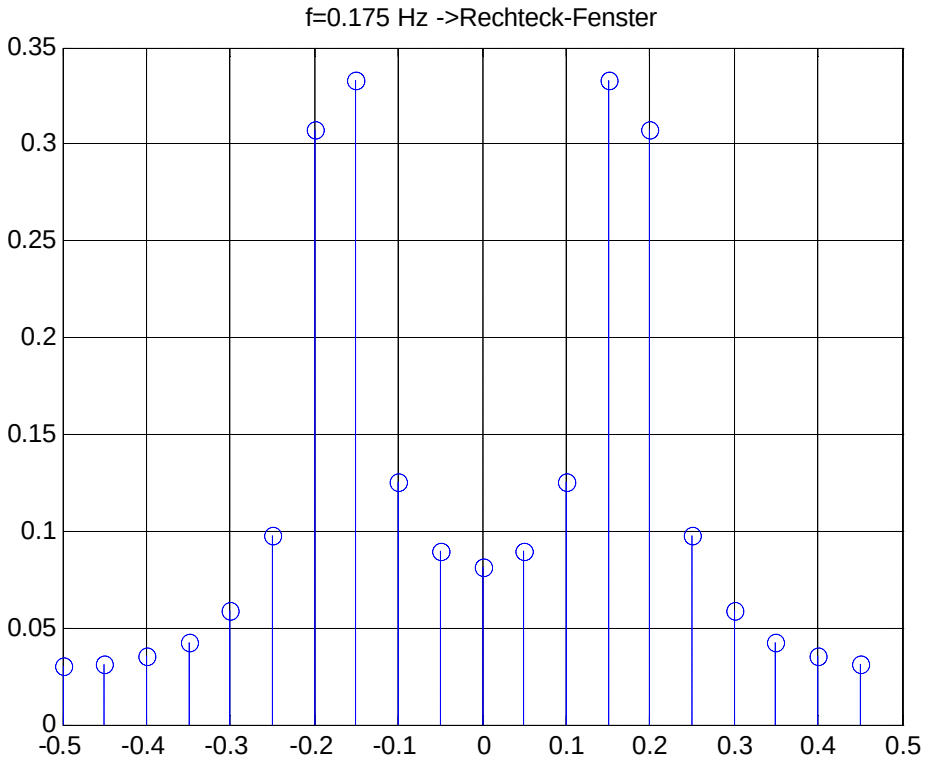
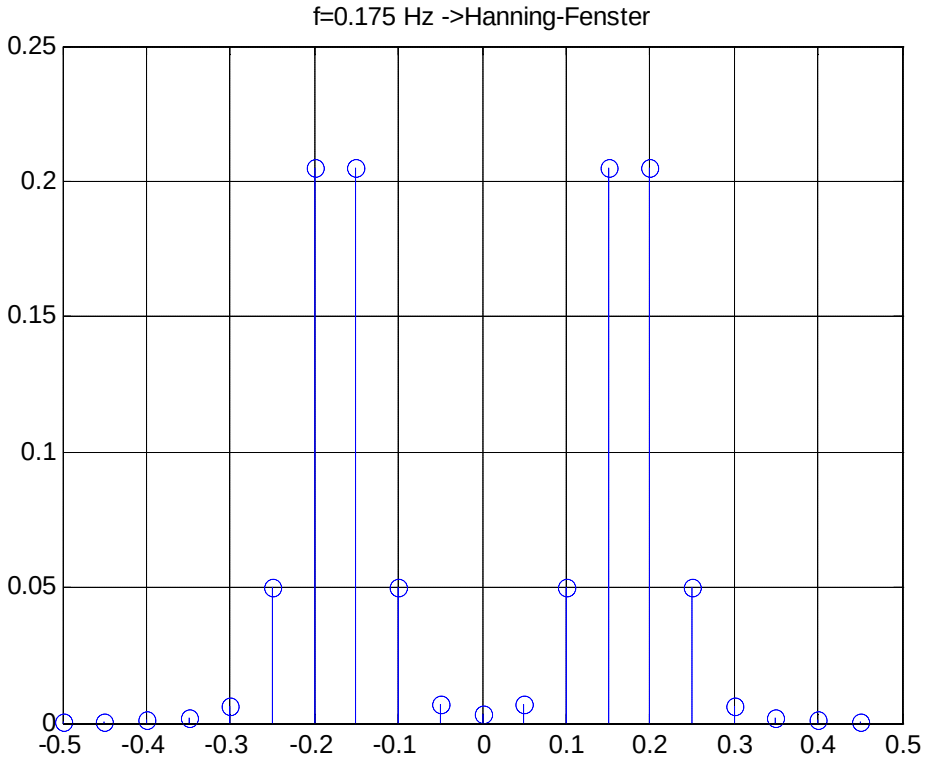


Abbildung 1-10



1.3.4. Matlab-Code

Der Matlab-Code wurde jeweils für die einzelnen Plots angepasst.

```
%close all;
clear all;
f0 = 0.15;
f1 = 0.1;
x1 = exp(-j*2*pi*[0:9]*f0)/2;
x2 = exp(j*2*pi*[0:9]*f0)/2;
x3 = x1 + x2 + cos(2*pi*[0:9]*f1);
x3(80) = 0;
X1 = fftshift(fft(x1))/length(x1);
X2 = fftshift(fft(x2))/length(x2);
X3 = fftshift(fft(x3))/length(x2);
N=length(x1);

fAchse = -0.5:1/length(x1):0.5-1/length(x1);

figure(1); stem(fAchse, abs(X1));
figure(2); stem(fAchse, abs(X2));
fAchse = -0.5:1/length(x3):0.5-1/length(x3);
figure(3); stem(fAchse, abs(X3));

fAchseSinc = -0.5:1/1000:0.5-1/1000;

y1 = exp(-j*pi*(fAchseSinc+f0)*(N-1)) .* (sin((f0+fAchseSinc)*N*pi)./
                                           sin((f0+fAchseSinc)*pi))/(N);
y2 = exp(-j*pi*(fAchseSinc-f0)*(N-1)).*(sin((-f0+fAchseSinc)*N*pi)./
                                          sin((-f0+fAchseSinc)*pi))/(N);

figure(1); hold on; plot(fAchseSinc, abs(y1)/2 ,'--r');hold off ; grid;
figure(2); hold on; plot(fAchseSinc, abs(y2)/2 ,'--r'); hold off; grid;
title('exp(j*2*pi*[0:9]*f0)')

y3 = y2 + y1;
f0 = f1;

y1 = exp(-j*pi*(fAchseSinc+f0)*(N1)).*(sin((f0+fAchseSinc)*N*pi)./
                                      sin((f0+fAchseSinc)*pi))/(N);

y2 = exp(-j*pi*(fAchseSinc-f0)*(N-1)).*(sin((-f0+fAchseSinc)*N*pi)./
                                          sin((-f0+fAchseSinc)*pi))/(N);

y3 = y3 + y2 +y1;

figure(3); hold on; plot(fAchseSinc, abs(y3)/2 ,'--r'); hold off;grid
```

2. Zoom-FFT

Eine Verbesserung der spektralen Auflösung der DFT lässt sich bei gegebener Abtastfrequenz nur durch Erhöhung der Anzahl der Abtastwerte erreichen. Das lässt sich aus dem Umstand erklären, dass Gleichung (1.2) nicht nur den Abstand zwischen zwei Frequenzen angibt, sondern auch die kleinste von Null verschiedene Frequenz. Um eine Frequenz aber bestimmen zu können, müssen Abtastwerte mindestens einer Periode vorliegen.
Interessiert nun eine hohe spektrale Auflösung in einem Teilintervall des Spektrums führt die nötige Erhöhung der Anzahl an Abtastwerten unter Umständen zu einer nicht tolerierbaren Zunahme des Rechenaufwandes. Von Interesse sind also Algorithmen, die einen bestimmten Ausschnitt des Gesamtspektrums berechnen können.

2.1. Überlagerungsprinzip

Beim Überlagerungsprinzip wird das interreisende Teilspektrum in die Nulllage gemischt und anschließend tiefpassgefiltert. Bei diesem gefilterten Signal kann nun eine Abtastratenreduktion durchgeführt werden um dann die DFT mit weniger Werten und somit geringem Aufwand zu berechnen.
Zu dem Überlagerungsprinzip wurden keine Simulationen mit Matlab durchgeführt.

2.2. Subtransformation

Mit Hilfe der Subtransformation lässt sich ebenfalls ein Teilspektrum des Gesamtspektrums berechnen. Gegenüber der herkömmlichen DFT bzw. FFT und auch gegenüber dem Überlagerungsprinzip hat dieses Verfahren den Vorteil, weniger komplexe Multiplikationen (DFT) und keine steilflankeigen Filter (Überlagerungsprinzip) zu benötigen.

Zu Berechnung der DFT nach der Subtransformationsmethode wird die DFT-Summe in zwei Summen aufgeteilt. Dabei kann die innere Summe (mit dem Index n) mit dem Standart-FFT- Algorithmus berechnet werden. Die Ergebnisse der FFT werden dann mit einer Phasenkorrektur multipliziert und anschließend addiert. Es werden also N Frequenzwerte im Abstand $\Delta f = \frac{f_A}{N \cdot K}$ berechnet. Dabei kann die Lage des Teilspektrums mit l eingestellt werden.

$$\underline{F}\left(m + l \cdot N\right) = \frac{1}{K} \sum_{i=0}^{K-1} \left\{ \frac{1}{N} \sum_{n=0}^{N-1} z\left(Kn + i\right) e^{-j2p\frac{mn}{N}} \right\} \cdot e^{-j2p\frac{m+l \cdot N}{M}} \qquad (2.1)$$

Um ein analytisches Signal auf der negativen Frequenzachse zu berechnen, muss die Lage des Teilspektrums in den negativen Bereich der Frequenzachse der FFT geschoben werden. Die Anmerkung aus Kapitel 1 ist zu beachten.

In der inneren Summe (Laufindex n) wird eine Abtastratenreduktion ohne vorherige Bandbegrenzung durchgeführt. Das Abtasttheorem wird somit für jede einzelne FFT-Berechnung der inneren Schleife verletzt und es kommt zu Alaising.

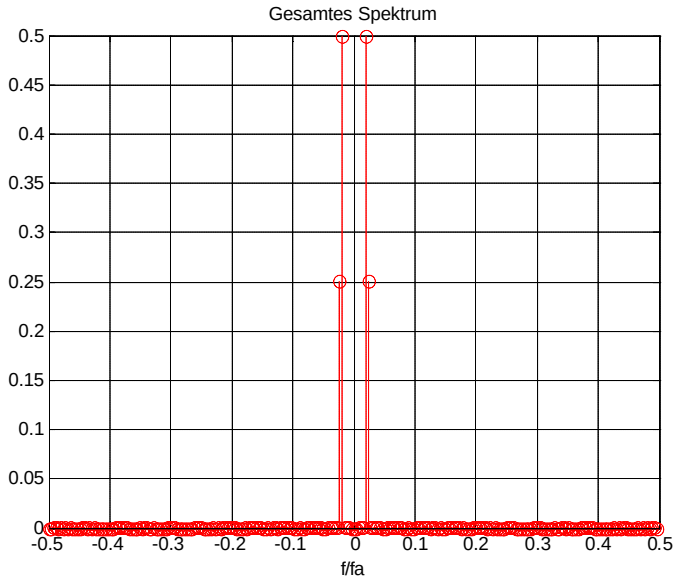
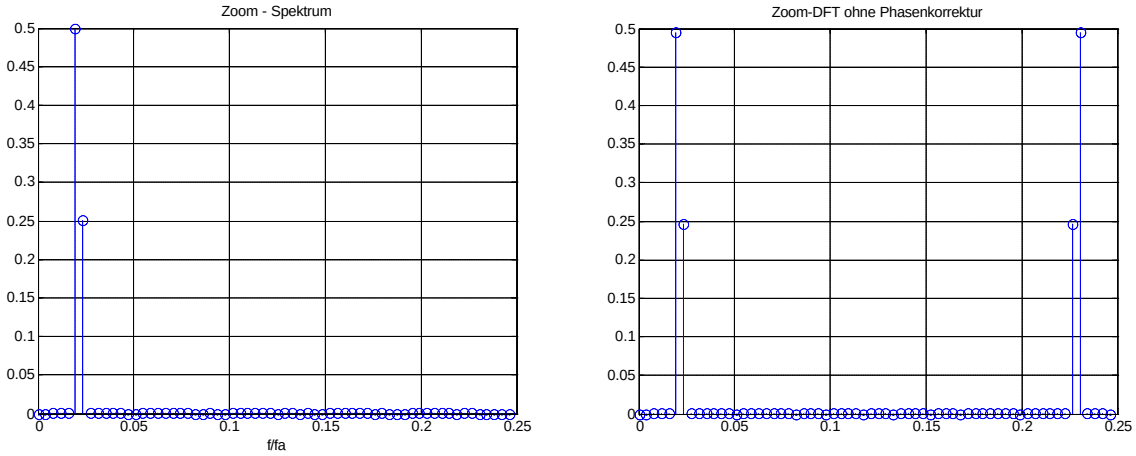


Abbildung 2-1 256-Punkte FFT



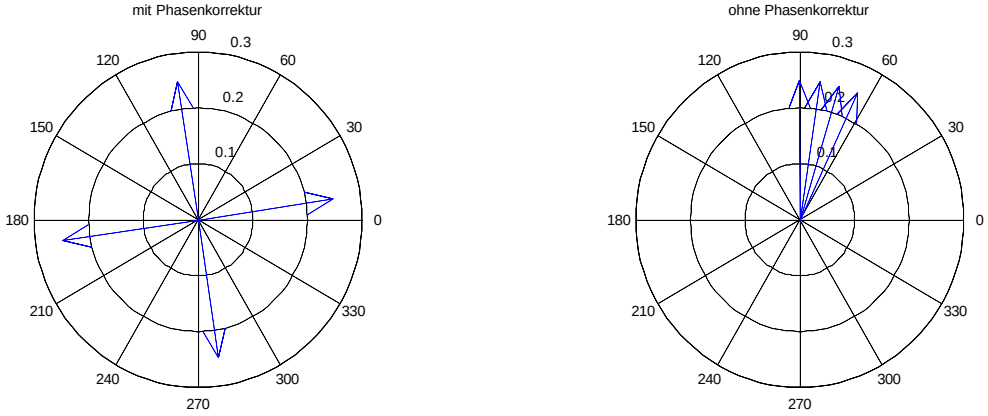


Abbildung 2-2 Oben: Zoom-FFT mit und ohne Phasenkorrektur

Unten: Die zugehörigen Zeigerdiagramme für f = 0,226 Hz (Alaisingfrequenz)

2.2.1. Plots

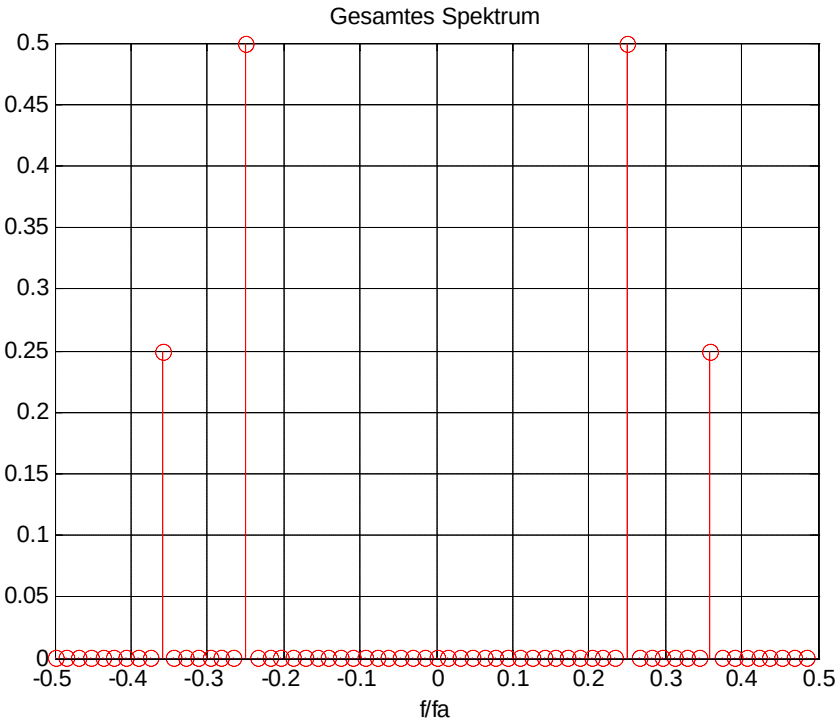


Abbildung 2-3 DFT mit 64 Punkten

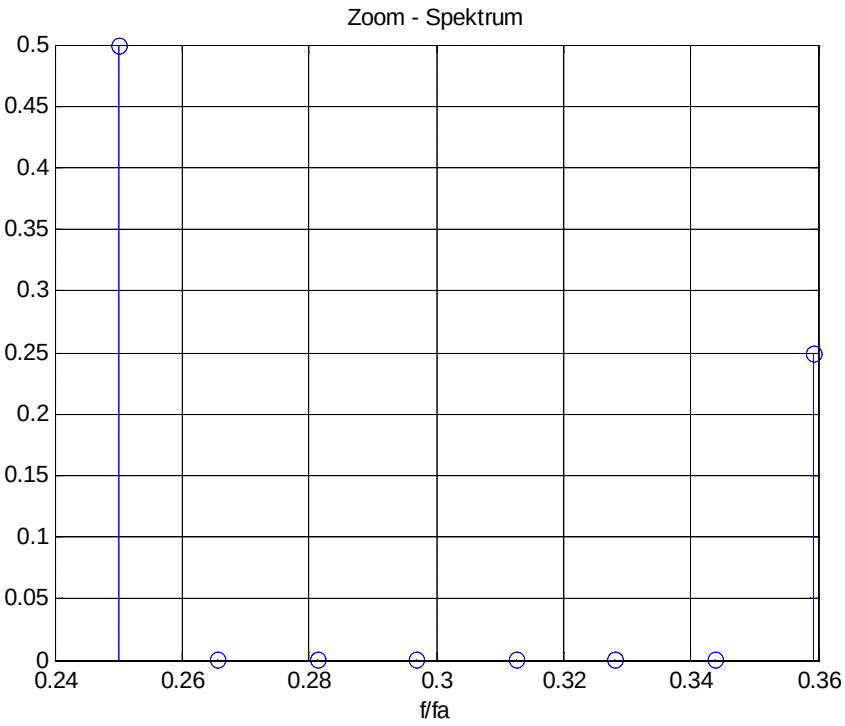


Abbildung 2-4 Zoom-FFT (64 Punkte) mit Zoomfaktor K = 8 und Lage des Fensters l = 2

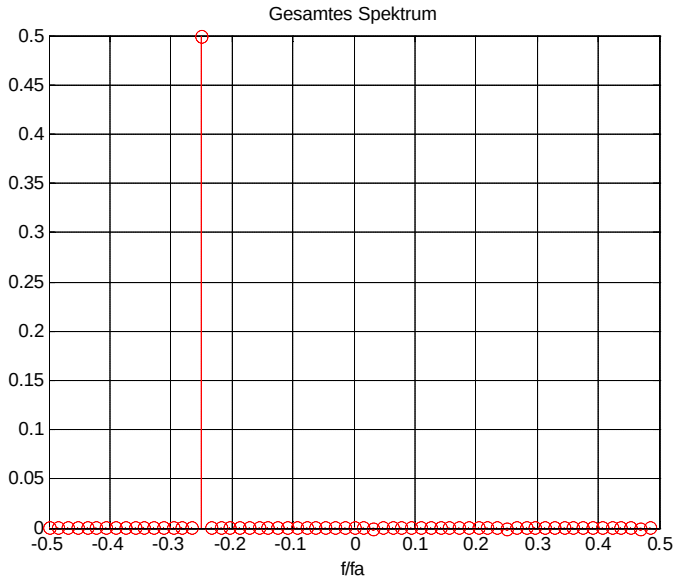


Abbildung 2-5 FFT eines analytischen Signals

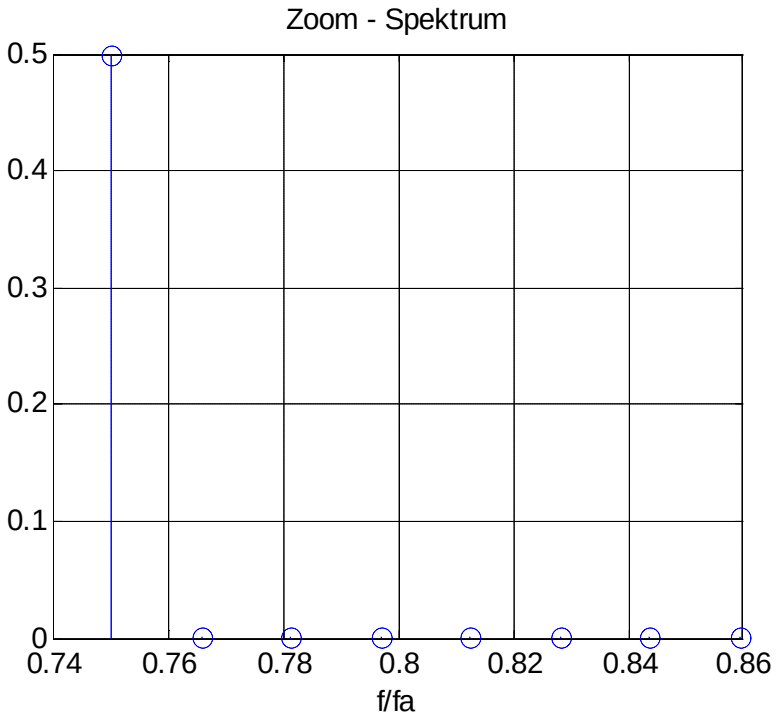


Abbildung 2-6 Zoom - FFT eines analytischen Signals.

Für ein analytisches Signal mit negativer Frequenz muss das Fenster über die positiven Frequenzen hinweg geschoben werden.

2.2.2. Matlab-Code

Nach dem ein Verständnis für die Abläufe dieses Algorithmussees vorhanden ist, kann dieser sehr leicht in Matlab implementiert werden.

```
clear all
close all
f0 = 5/(256);
f02 = 6/(256);
x = sin(2 * pi *[0:256-1] * f0) + 0.5*sin(2 * pi *[0:256-1] * (f02)); %
Test-Signal
%x = exp(-j*2*pi*[0:64-1]*f0)/2;

K = 4; % Zoom-Faktor
begin = 0;
sig = x;
N = length(sig)/K; % Teilintervall

specZoom = zeros(1,N); % Zoom-Spektrum
specAliasing = specZoom;
fAchseZ = begin/K:1/length(sig):begin/K + (N-1)/length(sig);

for m = 1:K % Zoom-Faktor
    sig(m:K:length(sig));
    temp(m,:) = fft(sig(m:K:length(sig)))/N;
    tempPhaseKor(m,:) = temp(m,:) .* exp(-j*2*pi*([0:N-
1]+begin*N)*m/(K*N));
    specZoom = specZoom +  (temp(m,:).* exp(-j*2*pi*([0:N-
1]+begin*N)*m/(K*N)))/K;
    specAliasing = specAliasing + temp(m,:)/K;
end

figure(1)
fAchseZ = begin/K:1/length(sig):begin/K + (N-1)/length(sig);
stem(fAchseZ, abs((specZoom))); grid
title('Zoom - Spektrum')
xlabel('f/fa')

figure(2)
fAchse= -0.5: 1/length(x):0.5-1/length(x);
X = fftshift(fft(x))/length(sig);
stem( fAchse, abs(X), 'r');grid
title('Gesamtes Spektrum')
xlabel('f/fa')

figure(3)
compass((temp(:,59))); title('ohne Phasenkorrektur')
grid
figure(4)
compass((tempPhaseKor(:,59)),'b'); title('mit Phasenkorrektur')
grid

figure(5)
stem(fAchseZ, abs(specAliasing));grid; title('Zoom-DFT ohne
Phasenkorrektur')
```