1. Polly

Bildverarbeitung und Arbeiten mit Matrizen in MATLAB

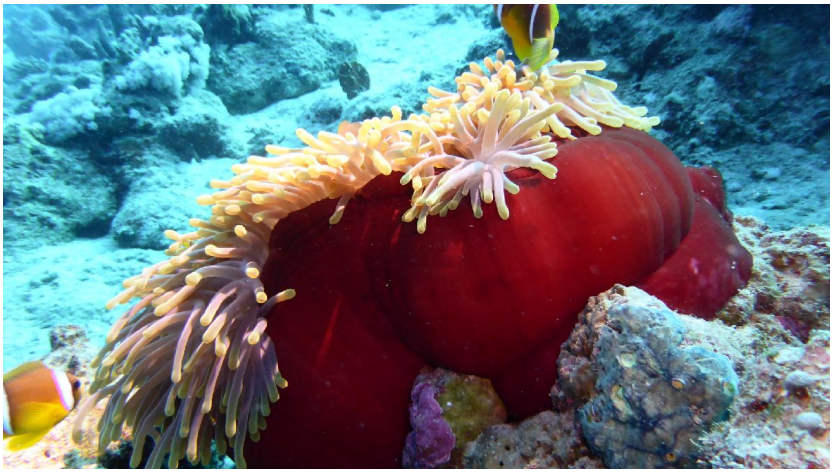


Bild: Aufgabe1.png

a) Lesen Sie das Bild ***Aufgabe1.png*** in MATLAB ein und lassen es sich anzeigen. Jeder Bildpixel hat dabei eine Wortbreite von 16 Bit (2 Bytes), also das Format uint16. Trennen Sie das untere Byte (8 LSBs) ab und stellen Sie das im Lower Byte versteckte Bild korrekt dar. Beachten Sie dabei, dass jede geradzahlige Zeile des versteckten Bildes zusätzlich horizontal gespiegelt ist. Dieser Aufgabenteil ist in einem MATLAB Skript zu lösen.

b) Ersetzen Sie den grünen Hintergrund des erhaltenen versteckten Bildes durch ein Hintergrundbild Ihrer Wahl, ohne dass die anderen Farben, die Helligkeit oder die Sättigung geändert werden. Dieser Teil ist im HSV Farbraum durchzuführen.

Überblick

In dieser Aufgabe wird ein Bild verarbeitet, um versteckte Informationen darin zu extrahieren und den Hintergrund des Bildes zu ändern. Die Aufgabe ist in zwei Teile unterteilt:

- **Teil (a)**: Extrahieren des versteckten Bildes aus den unteren 8 Bits (LSB) des Originalbildes und Korrektur durch horizontales Spiegeln der geradzahligen Zeilen.
- **Teil (b)**: Ersetzen des grünen Hintergrunds des versteckten Bildes durch ein anderes Hintergrundbild.

Implementierung

Teil (a): Bild einlesen und verstecktes Bild extrahieren

- Das Bild `Aufgabe1.png` wird mit der `imread()`-Funktion eingelesen.
- Das Bildformat wird überprüft, um sicherzustellen, dass es im `uint16`-Format vorliegt.
- Die unteren 8 Bits (LSB) des Bildes werden mit der Funktion `bitand()` extrahiert.
- Die geradzahligen Zeilen des extrahierten Bildes werden horizontal gespiegelt, um das Bild korrekt darzustellen.
- **Ergebnis**: Das versteckte Bild wird extrahiert, die geradzahligen Zeilen werden gespiegelt und das Bild wird korrekt dargestellt.

Teil (b): Ersetzen des grünen Hintergrunds

- Das versteckte Bild wird in den HSV-Farbraum umgewandelt, um den grünen Hintergrund leichter isolieren zu können.
- Ein neues Hintergrundbild `hintergrund.jpg` wird eingelesen und auf die Größe des versteckten Bildes skaliert.
- Die grünen Pixel im versteckten Bild werden durch die Pixel des neuen Hintergrundbildes ersetzt.
- Das modifizierte Bild wird schließlich wieder in den RGB-Farbraum konvertiert und angezeigt.
- **Ergebnis**: Der grüne Hintergrund des versteckten Bildes wird durch ein neues Hintergrundbild ersetzt und das fertige Bild wird angezeigt.

Grafiken

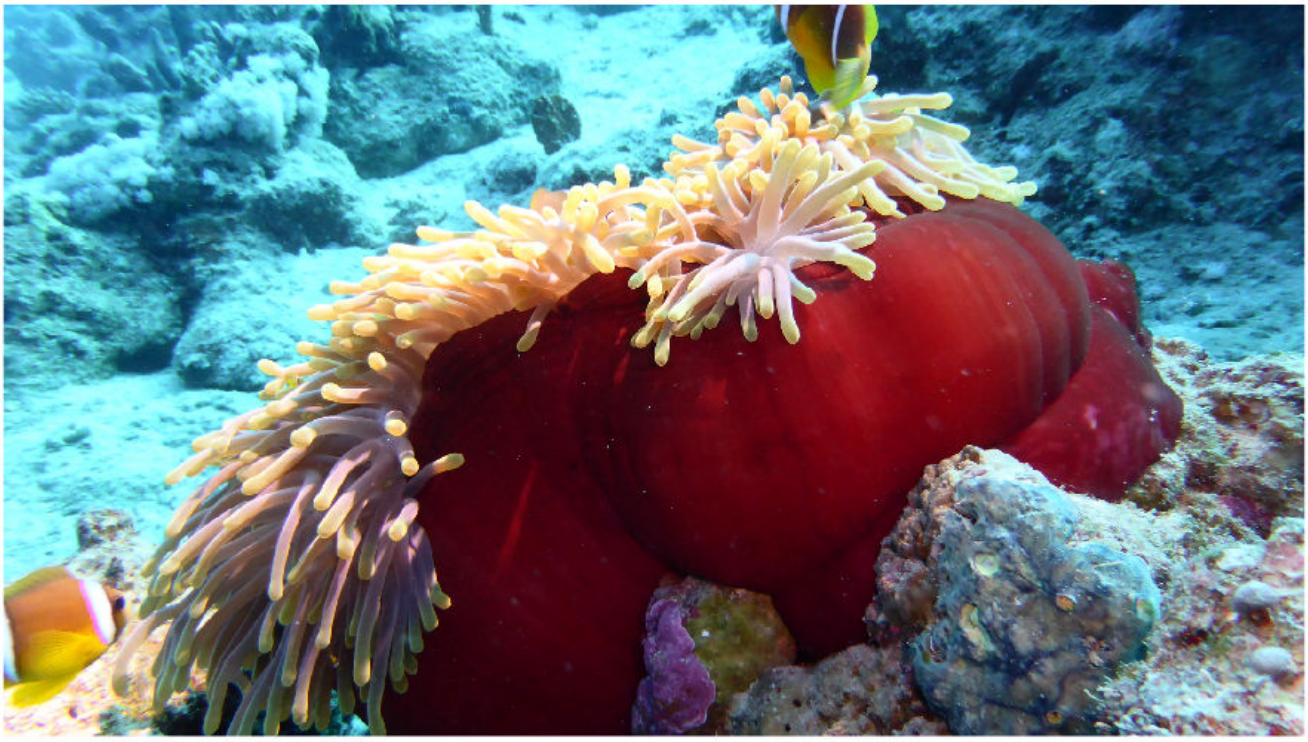
Extrahiertes verstecktes Bild nach dem Spiegeln der geraden Zeilen (Teil a):

- Nach dem Extrahieren der LSBs und dem Spiegeln der geraden Zeilen wird das versteckte Bild dargestellt.
- Dies zeigt das verborgene Bild im Originalbild an, nachdem es korrekt rekonstruiert wurde.

Bild mit ersetztem Hintergrund (Teil b):

- Nachdem der grüne Hintergrund im versteckten Bild erkannt und durch ein neues Hintergrundbild ersetzt wurde, wird das Ergebnis dargestellt.
- Diese Grafik zeigt das modifizierte Bild mit dem neuen Hintergrund.

```
% a) Bild einlesen und anzeigen
bild = imread('Aufgabe1.png'); % Liest das Bild "Aufgabe1.png" ein
imshow(bild); % Zeigt das Bild an
```



```
if isa(bild, 'uint8')
    disp('Das Bild ist vom Datentyp uint8.'); % Überprüft, ob der Datentyp uint8 ist
else
    disp('Das Bild ist vom Datentyp uint16.'); % Überprüft, ob der Datentyp uint16
ist
end
```

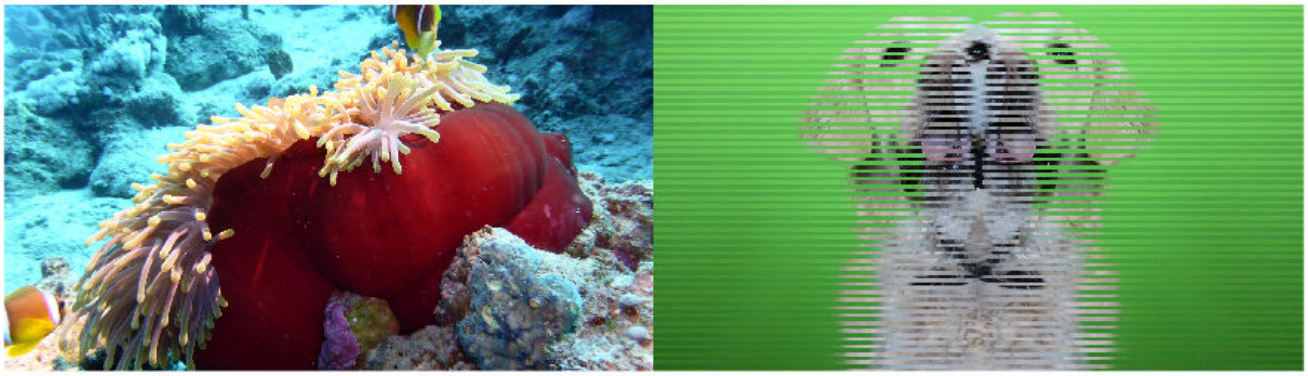
```
Das Bild ist vom Datentyp uint16.
```

```
% Extrahieren des unteren Bytes (8 LSBs)
versteckterBild = uint8(bitand(bild, 255)); % Extrahiert die 8 LSBs, um das
versteckte Bild zu erhalten
figure;
imshow(versteckterBild); % Zeigt das versteckte Bild an
```

```
if isa(versteckterBild, 'uint8')
    disp('Das Bild ist vom Datentyp uint8.');
else
    disp('Das Bild ist vom Datentyp uint16.');
end
```

```
Das Bild ist vom Datentyp uint8.
```

```
figure;
imshowpair(bild, versteckterBild, 'montage'); % Zeigt Original und verstecktes Bild
nebeneinander
```



```
[zeilen, spalten, ~] = size(verstecktesBild); % Bestimmt die Größe des versteckten
Bildes
disp(zeilen)
```

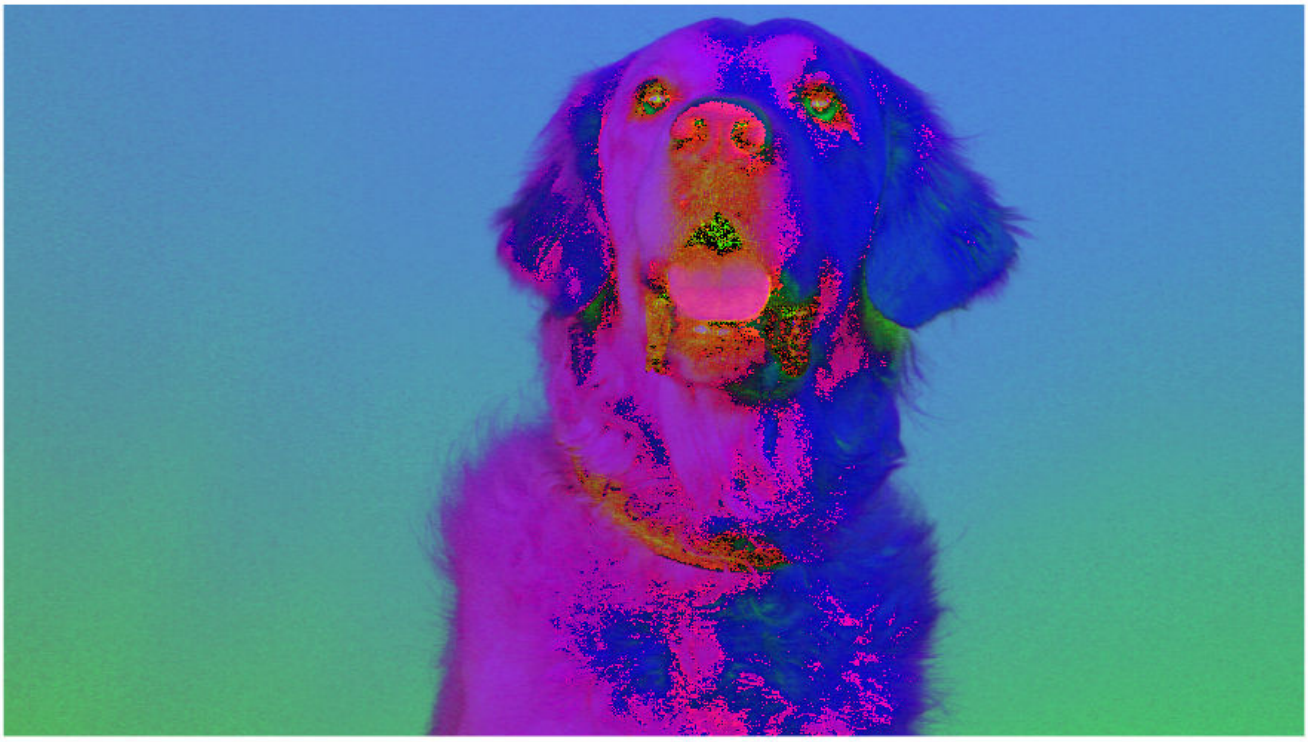
```
       1080
```

```
disp(spalten)
```

```
       1920
```

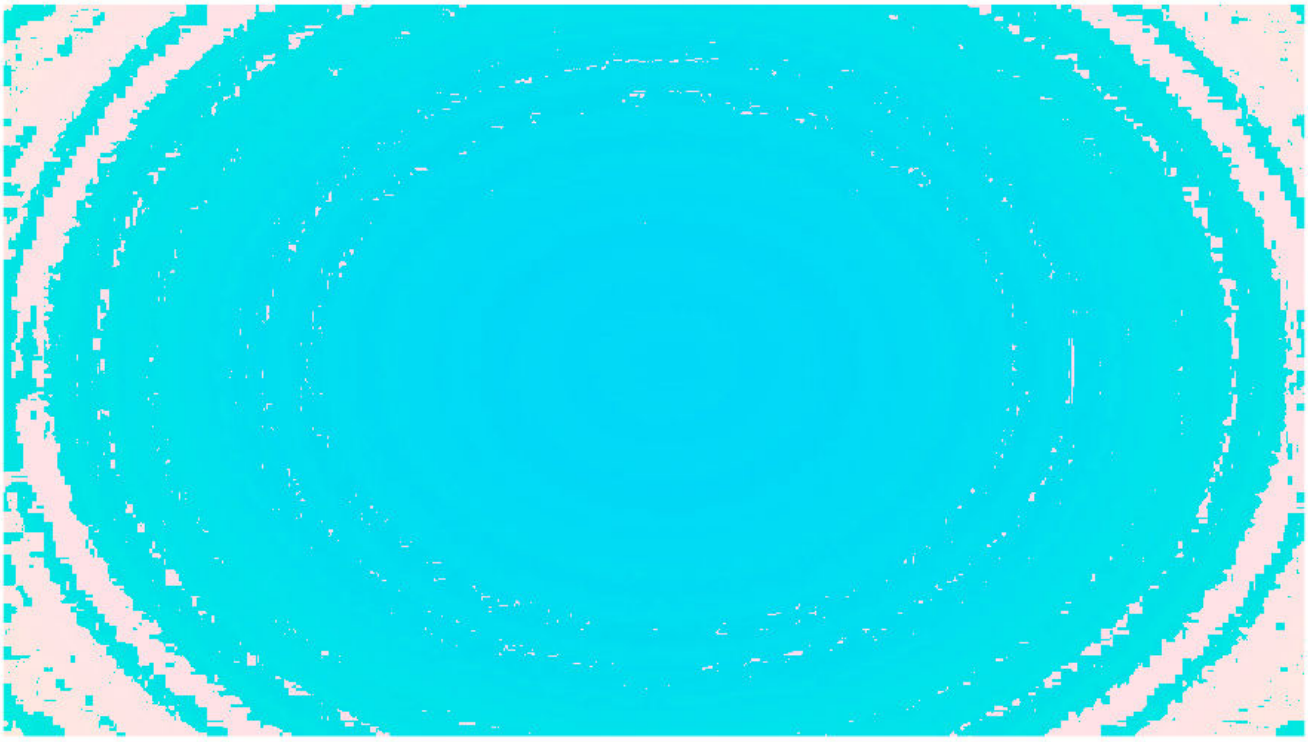
```
% Spiegeln der geradzahligen Zeilen
for zeile = 1:zeilen
    if mod(zeile, 2) == 0 % Prüft, ob die Zeile eine gerade Zahl ist
        verstecktesBild(zeile, :, :) = fliplr(verstecktesBild(zeile, :, :)); %
Spiegelt die Zeile horizontal
    end
end
imshow(verstecktesBild); % Zeigt das korrigierte versteckte Bild an
```

```
% b) Hintergrund ersetzen im HSV-Farbraum
verstecktesBild = rgb2hsv(verstecktesBild); % Konvertiert das versteckte Bild in
den HSV-Farbraum
figure;
imshow(verstecktesBild);
```

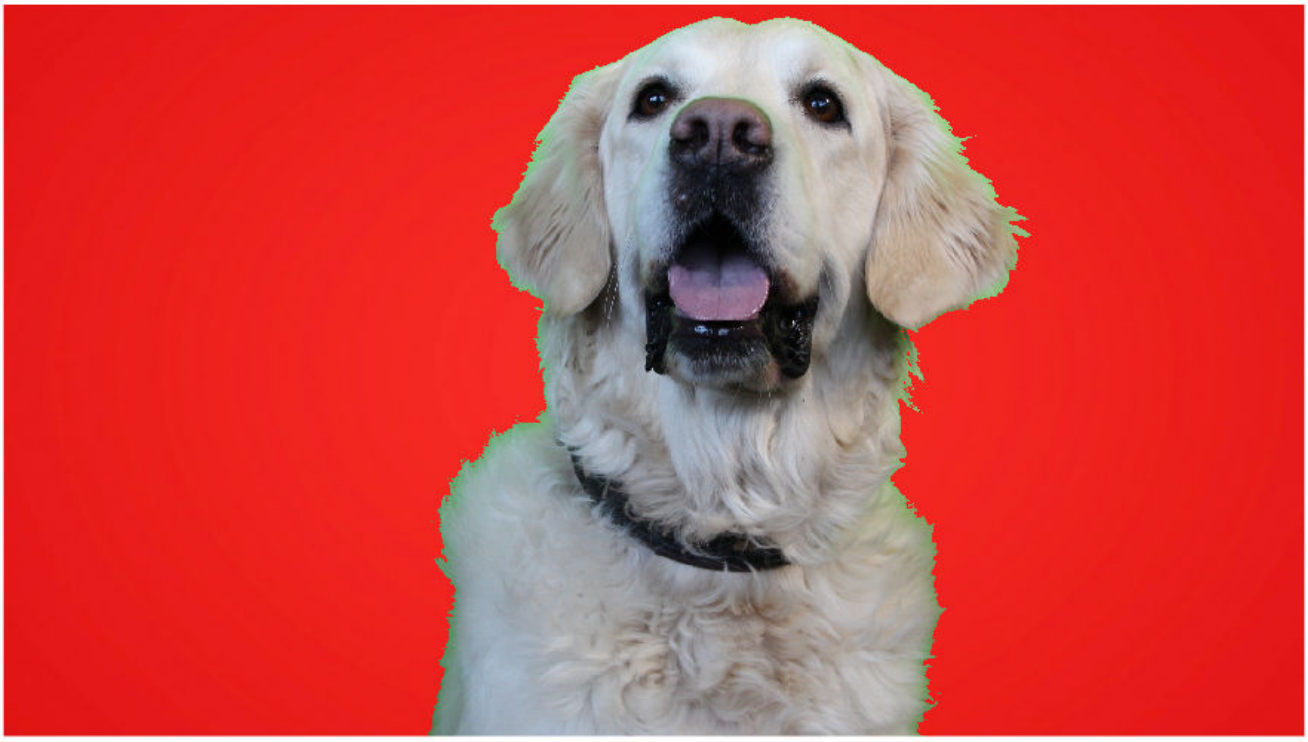


```
hintergrund = imread('hintergrund.jpg'); % Liest das Hintergrundbild ein
imshow(hintergrund);
```

```
hintergrund = imresize(hintergrund, [zeilen, spalten]); % Passt das Hintergrundbild
an die Größe des versteckten Bildes an
hintergrund = rgb2hsv(hintergrund); % Konvertiert das Hintergrundbild in den HSV-
Farbraum
figure;
imshow(hintergrund);
```



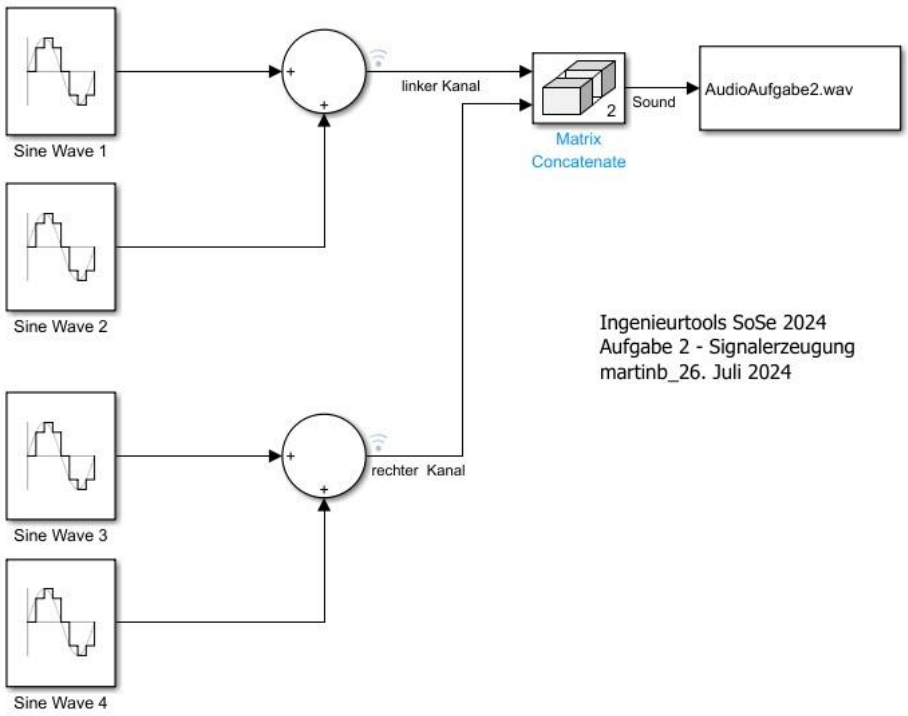
```
% Ersetzen des grünen Hintergrunds durch das neue Hintergrundbild
for zeile = 1:zeilen
    for spalte = 1:spalten
        if verstecktesBild(zeile, spalte, 1) >= 0.25 && verstecktesBild(zeile,
spalte, 1) <= 0.4 && verstecktesBild(zeile, spalte, 2) >= 0.4 &&
verstecktesBild(zeile, spalte, 3) >= 0.2
            verstecktesBild(zeile, spalte, :) = hintergrund(zeile, spalte, :); %
Ersetzt den Pixel, wenn er grün ist
        end
    end
end

verstecktesBild = hsv2rgb(verstecktesBild); % Konvertiert das Bild zurück in den
RGB-Farbraum
imshow(verstecktesBild); % Zeigt das Endergebnis an
```



2. Stereo

Audiosignalverarbeitung und Spektralanalyse in MATLAB, logarithmische und semilogarithmische Plots, FFT

Die Datei ***Aufgabe2.wav*** enthält eine Audiodatei und wurde gemäß nachfolgendem Simulink Blockdiagramm folgendermaßen erzeugt: Das Audiosignal hat zwei unterschiedliche Kanäle (linker/rechter Kanal) und jeder Kanal besteht aus zwei monochromatischen Signalen (Sinusschwingungen) unterschiedlicher Frequenzen.



,

Bild: Audiosignal Erzeugung

a) Lesen Sie die Audiodatei ein, finden Sie die Abtastrate des Audiofiles heraus und hören Sie sich das Audiosignal mit der korrekten Abtastrate an.

b) Stellen Sie das Signal im Zeitbereich mit passender Achsenbeschriftung (auf der x-Achse die korrekte Zeit t darstellen) und Titel dar.

c) Stellen sie einen kurzen Ausschnitt des Zeitsignals über einen Zeitabschnitt von 25 ms dar.

d) Stellen Sie das Signal im Frequenzbereich dar (Achsenbeschriftung, Titel, geeignete halblogarithmische Skalierung des Betragsspektrums). Stellen Sie dabei nur die positiven Frequenzanteile dar. Benutzen Sie dazu den Befehl *fft*. Führen Sie die Fast Fourier Transformation (FFT) getrennt für den linken und rechten Kanal aus. Lesen Sie aus den Betragsspektren des rechten und linken Kanals die Frequenzen der Signale aus. Aus welchen Frequenzen besteht das Audio-Signal?

Die Aufgabe ist mittels eines MATLAB Skriptes zu lösen.

Überblick

In dieser Aufgabe wird ein Stereo-Audiosignal in vier Schritten verarbeitet:

- **Teil (a)**: Einlesen und Abspielen der Audiodatei.
- **Teil (b)**: Darstellung des Signals im Zeitbereich.
- **Teil (c)**: Darstellung eines 25-ms-Ausschnitts des Signals.
- **Teil (d)**: Analyse des Signals im Frequenzbereich mittels FFT.

Implementierung:

Teil (a): Audiodatei einlesen und abspielen

- Wir lesen die Audiodatei `Aufgabe2.wav` mit der `audioread()`-Funktion ein.
- Das Audiosignal und die Abtastrate werden in den Variablen `yAchs` (Audiodaten) und `fs` (Abtastrate) gespeichert.
- Das eingelesene Audiosignal wird mit der `soundsc()`-Funktion abgespielt, unter Berücksichtigung der richtigen Abtastrate.
- **Ergebnis**: Das Audiosignal wird korrekt abgespielt, und die Abtastrate ist bekannt.

Teil (b): Darstellung des Signals im Zeitbereich

- Zunächst berechnen wir die Abtastperiode `T` durch den Kehrwert der Abtastrate.
- Anschließend erstellen wir einen Zeitvektor `t`, der die Dauer des Audiosignals in Sekunden abbildet.
- Das Audiosignal wird über der Zeit geplottet, wobei die x-Achse die Zeit und die y-Achse die Amplitude des Signals darstellt.
- **Ergebnis**: Das Audiosignal wird im Zeitbereich dargestellt, und wir können die Schwankungen des Signals über die Zeit analysieren.

Teil (c): Darstellung eines 25-ms-Ausschnitts

- Ein kurzer Ausschnitt von 25 Millisekunden wird betrachtet.
- Dazu berechnen wir, wie viele Abtastwerte 25 ms entsprechen. Diese werden aus dem Audiosignal extrahiert.
- Der Ausschnitt wird dann im Zeitbereich dargestellt, um die Details des Signals in einem kleinen Zeitfenster zu analysieren.
- **Ergebnis**: Der 25-ms-Ausschnitt des Signals wird im Zeitbereich dargestellt. Dadurch erhält man einen genauen Einblick in das Verhalten des Signals über einen kurzen Zeitraum.

Teil (d): Darstellung des Signals im Frequenzbereich mittels FFT

- Wir wenden die Fast Fourier Transformation (FFT) auf den linken und rechten Kanal des Audiosignals getrennt an.
- Zunächst berechnen wir die nächsthöhere Potenz von 2 für die FFT und erstellen einen Frequenzvektor.
- Das Frequenzspektrum für beide Kanäle (links und rechts) wird dann geplottet, wobei nur die positiven Frequenzanteile dargestellt werden.
- **Ergebnis**: Die Frequenzspektren der beiden Kanäle werden dargestellt. Dies ermöglicht eine Analyse der Frequenzkomponenten des Audiosignals.

Grafiken

Der Code erzeugt verschiedene Grafiken, die das Audiosignal in unterschiedlichen Verarbeitungsstufen zeigen:

Zeitbereichsdarstellung des gesamten Signals (Teil b):

- Das gesamte Audiosignal wird im Zeitbereich dargestellt.
- **x-Achse**: Zeit in Sekunden.

- **y-Achse**: Amplitude des Signals.
- Diese Grafik bietet einen Überblick über den Verlauf des Signals über die gesamte Dauer.

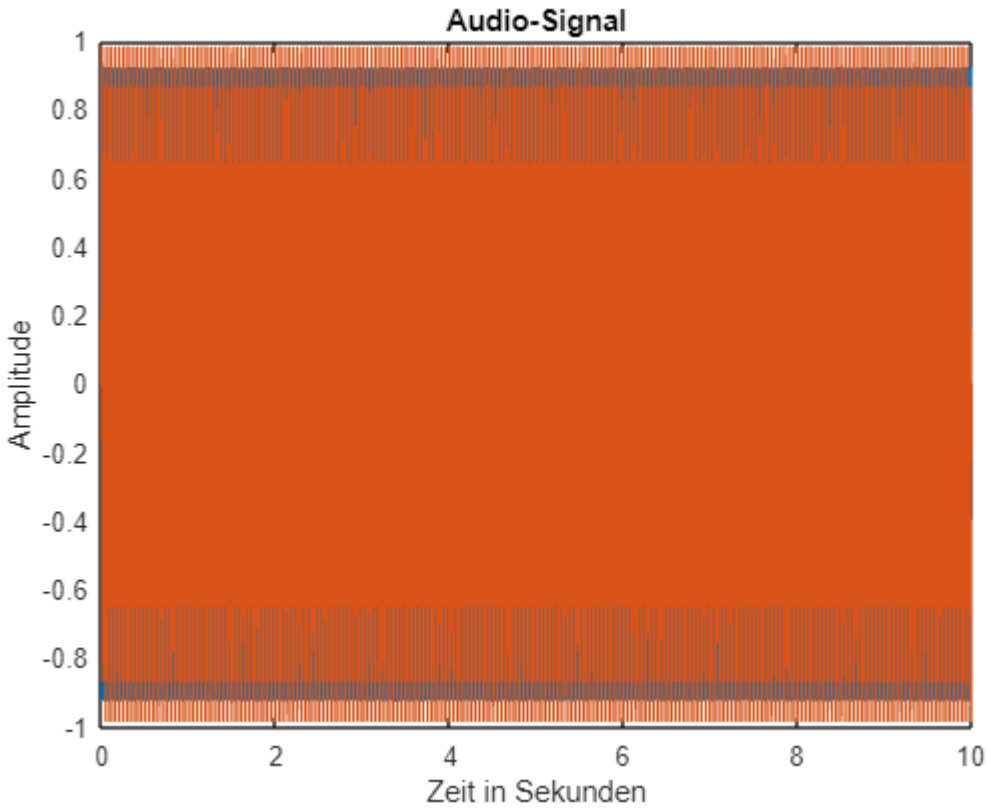
25-ms-Ausschnitt des Signals (Teil c):

- Ein kurzer Abschnitt des Signals, der 25 Millisekunden umfasst, wird im Zeitbereich dargestellt.
- **x-Achse**: Zeit in Sekunden.
- **y-Achse**: Amplitude des Signals.
- Diese Grafik hilft, das Verhalten des Signals in einem sehr kleinen Zeitfenster detailliert zu betrachten.

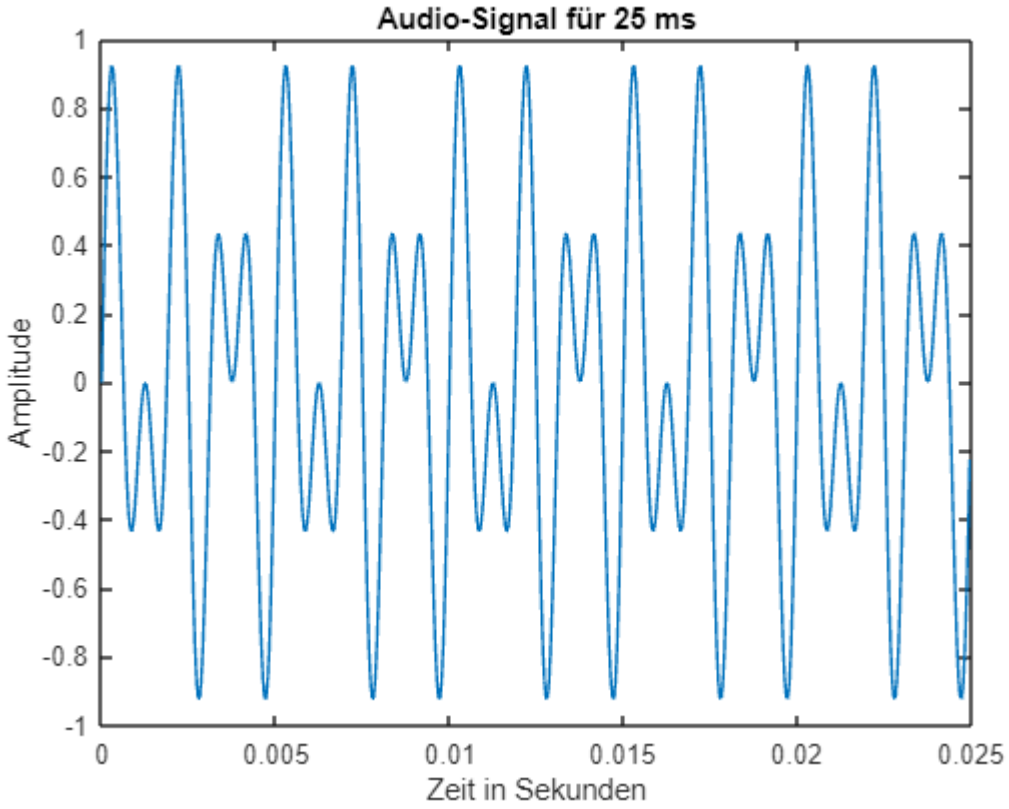
Frequenzspektrum des linken und rechten Kanals (Teil d):

- Das Frequenzspektrum wird separat für den linken und den rechten Kanal des Audiosignals dargestellt.
- **x-Achse**: Frequenz in Hertz.
- **y-Achse**: Amplitude des Signals.
- In diesen Grafiken werden die positiven Frequenzanteile des Audiosignals sichtbar, sodass man erkennen kann, aus welchen Frequenzen das Signal zusammengesetzt ist.

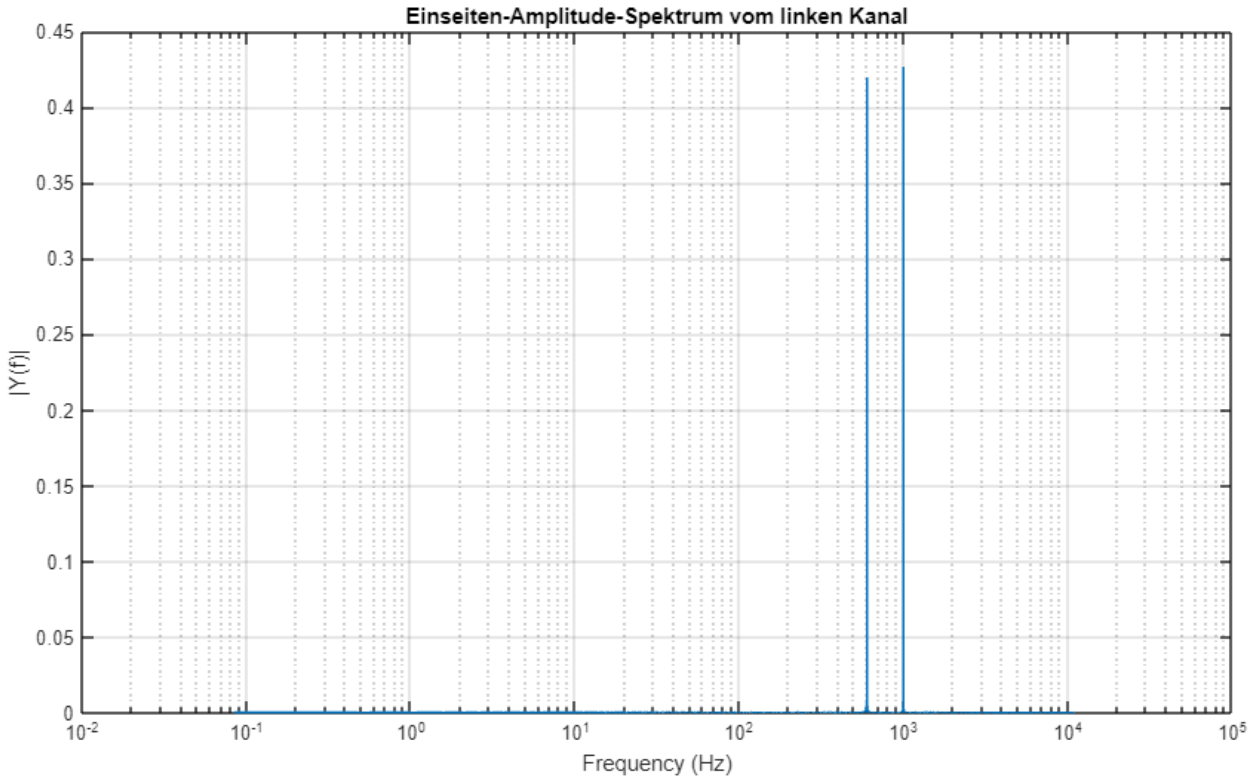
```
% a) Audio einlesen und abspielen
clear all; % Löscht alle bestehenden Variablen aus dem Speicher
[yAchs, fs] = audioread('Aufgabe2.wav'); % Liest die Audiodatei ein; yAchs enthält
Daten, fs ist die Abtastrate
soundsc(yAchs, fs) % Spielt das Audiosignal mit der richtigen Abtastrate ab

% b) Darstellung des Signals im Zeitbereich
T = 1/fs; % Berechnet die Abtastperiode (Zeit zwischen zwei Abtastpunkten)
t = (0:length(yAchs)-1) * T; % Erstellt einen Zeitvektor für die Dauer des
Audiosignals
figure; % Erstellt ein neues Grafikfenster

plot(t, yAchs); % Plottet das Audiosignal über der Zeit
xlabel('Zeit in Sekunden'); % Beschriftung der x-Achse
ylabel('Amplitude'); % Beschriftung der y-Achse
title('Audio-Signal'); % Titel des Plots
```



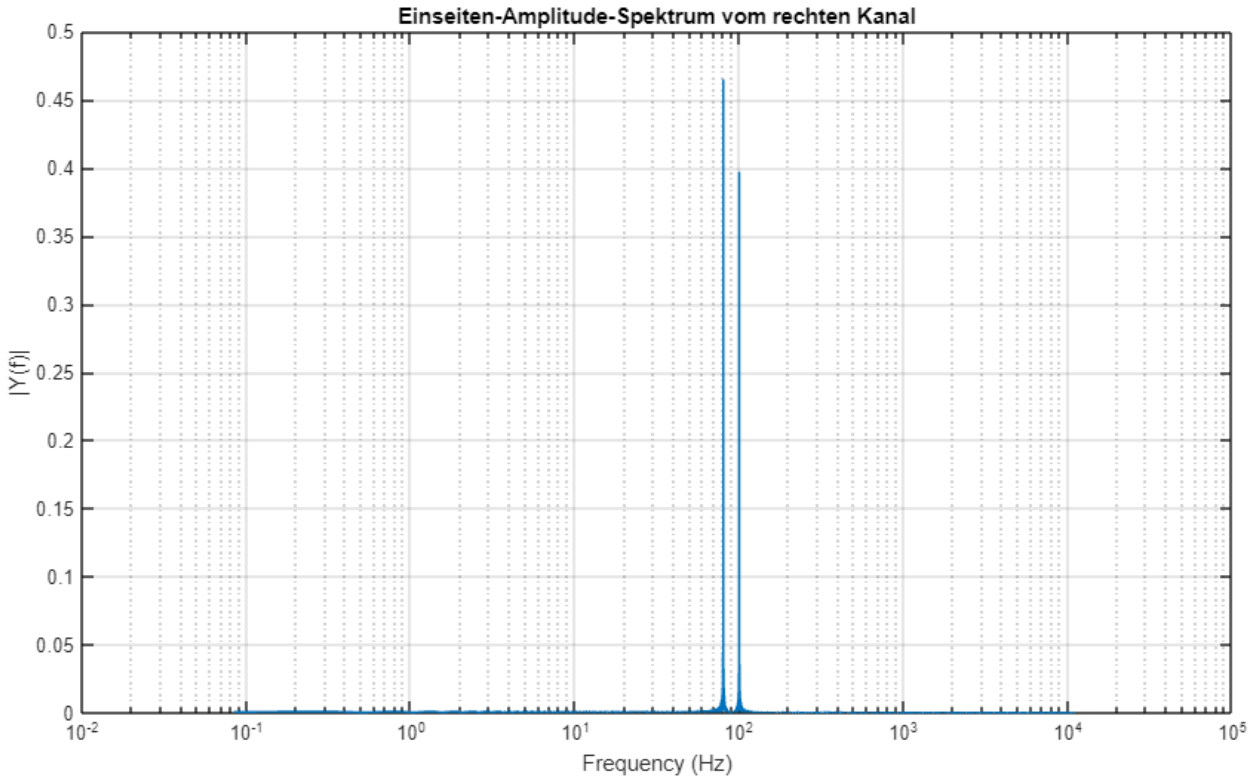

```
% c) Darstellung eines 25 ms Ausschnitts des Signals
time = 0.025; % Definiert die Länge des Ausschnitts in Sekunden
L = time * fs; % Berechnet die Anzahl der Abtastwerte für 25 ms
zeitsignals_ausschnitt = yAchs(1:L); % Extrahiert die ersten L Abtastwerte
t = (0:length(zeitsignals_ausschnitt) - 1) * T; % Erstellt den Zeitvektor für den
Ausschnitt

figure; % Erstellt ein neues Grafikfenster
plot(t, zeitsignals_ausschnitt); % Plottet den 25 ms Ausschnitt
xlabel('Zeit in Sekunden'); % Beschriftung der x-Achse
ylabel('Amplitude'); % Beschriftung der y-Achse
title('Audio-Signal für 25 ms'); % Titel des Plots
```




```
% d) Darstellung des Signals im Frequenzbereich
linkenKanal = yAchs(:, 1); % Extrahiert den linken Kanal des Audiosignals
rechtenKanal = yAchs(:, 2); % Extrahiert den rechten Kanal des Audiosignals

% Anzahl der Abtastwerte
L = length(yAchs);  % Bestimmt die Gesamtlänge des Signals

% Bestimmt die nächsthöhere Potenz von 2 für die FFT
nfft = 2^nextpow2(L);

% Frequenzvektor für die positiven Frequenzen
f = fs / 2 * linspace(0, 1, nfft / 2 + 1);

% Berechnet das Frequenzspektrum beider Kanäle
YL = fft(linkenKanal, nfft) / L; % FFT für den linken Kanal, normalisiert
YR = fft(rechtenKanal, nfft) / L; % FFT für den rechten Kanal, normalisiert

figure(1);
semilogx(f, 2 * abs(YL(1:nfft / 2 + 1))); % Plottet das Amplitudenspektrum des
linken Kanals
title('Einseiten-Amplitude-Spektrum vom linken Kanal'); % Titel des Plots
xlabel('Frequency (Hz)'); % Beschriftung der x-Achse
ylabel('|Y(f)|'); % Beschriftung der y-Achse
grid on; % Gitternetz anzeigen
```



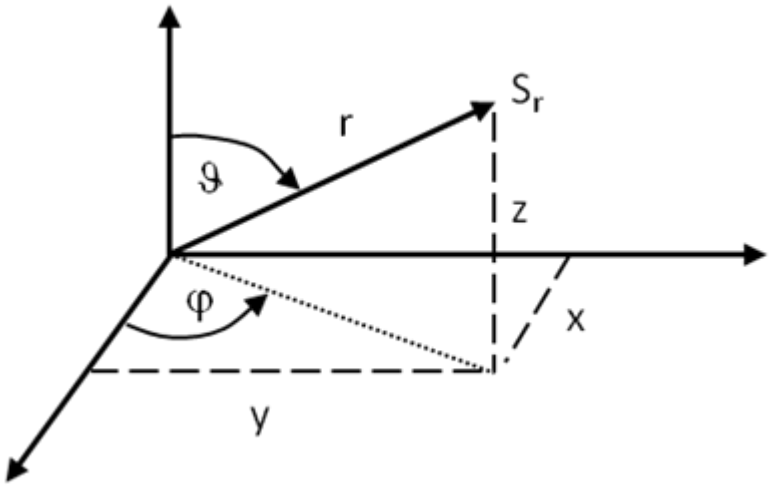

```
figure(1);
semilogx(f, 2 * abs(YR(1:nfft / 2 + 1))); % Plottet das Amplitudenspektrum des
rechten Kanals
title('Einseiten-Amplitude-Spektrum vom rechten Kanal'); % Titel des Plots
xlabel('Frequency (Hz)'); % Beschriftung der x-Achse
ylabel('|Y(f)|'); % Beschriftung der y-Achse
grid on; % Gitternetz anzeigen
```



3. Double Cone

3D Darstellung in MATLAB

Kugel- statt kartesischen Koordinaten sind für rotationssymmetrische Volumenkörper von Vorteil. Hierbei ist jeder Punkt im Raum durch den Winkel J (Elevation), den Winkel j (Azimut) und den Radius r beschreibbar.

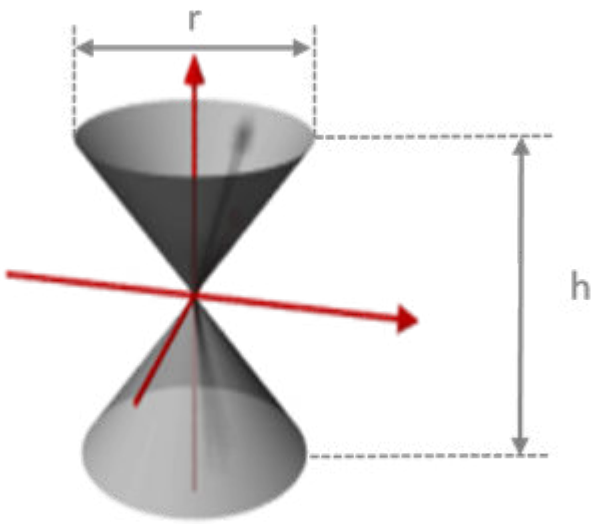


Skizze zu Aufgabe 3: Kugelkoordinaten

Die Beziehung zwischen den Komponenten in Kugel- und kartesischen Koordinaten lautet:

$$S_x = r \cdot \cos(\varphi) \cdot \sin(\vartheta)$$
$$S_y = r \cdot \sin(\varphi) \cdot \sin(\vartheta)$$
$$S_x = r \cdot \cos(\varphi)$$

Erzeugen Sie ein MATLAB Skript derart, mit dem Sie einen Doppelkegel mit dem max. Radius r = 2 und einer Gesamthöhe h = 10 dreidimensional darstellen können.



Doppelkegel

Überblick

In dieser Aufgabe geht es darum, einen Doppelkegel in MATLAB dreidimensional darzustellen. Der Doppelkegel besteht aus zwei symmetrischen Kegeln, die an ihrer Spitze miteinander verbunden sind. Wir erstellen ihn, indem wir Koordinaten für beide Kegelhälften berechnen und dann die 3D-Darstellung erzeugen.

Die Aufgabe umfasst drei Hauptschritte:

- Erstellen der Winkel und Höhen, um die Kegel zu definieren.
- Berechnen der X-, Y- und Z-Koordinaten für beide Kegelhälften.
- Darstellung des Kegels in einer 3D-Grafik.

Implementierung1. Parameter für den Doppelkegel festlegen

- **Maximaler Radius**: Der größte Radius des Kegels, `r = 2`.
- **Höhe des Doppelkegels**: Die Gesamthöhe des Kegels beträgt `h = 10`.

Wir verwenden die Funktion `meshNetzwerk()`, um die benötigten Winkel und Höhen zu berechnen. Diese Funktion erstellt eine Art Gitter, das es uns ermöglicht, die Oberfläche des Kegels darzustellen.

Implementierung :

a). Parameter festlegen

- Der Doppelkegel hat einen **Radius** von 2 (an der breitesten Stelle) und eine **Gesamthöhe** von 10.
- Wir berechnen die Winkel und Höhen, die für die Erstellung der Kegel benötigt werden.(durch die Funktion meshNetzwerk)

Funktoin - meshNetzwerk(h)

- Diese Funktion erstellt zwei Werte: **theta** (Winkel) und **z1** (Höhe). Diese Werte sind notwendig, um den Doppelkegel in Kugelkoordinaten zu beschreiben.
- **theta**: Wird von 0 bis 2*pi in 200 Schritten unterteilt. Dies ermöglicht es, den gesamten Kegel in einem Kreis zu berechnen.
- **z1**: Wird von 0 bis zur halben Höhe des Kegels (5) unterteilt. Dies ist die Höhe für den oberen Kegel.

b) Berechnen der Kegel-Koordinaten

- Für den **oberen Kegel** verwenden wir positive Z-Werte und berechnen den Radius und die X-, Y- und Z-Koordinaten.
- Für den **unteren Kegel** verwenden wir negative Z-Werte, um die Koordinaten der unteren Hälfte zu berechnen.

c) 3D-Darstellung des Doppelkegels - durch die Funktion zeigeDoppelkegel

- Wir verwenden die **surf()**-Funktion, um die Oberflächen der oberen und unteren Kegel darzustellen.
- Achsen werden beschriftet und ein Titel hinzugefügt, um die Grafik abzurunden.

Funktion - zeigeDoppelkegel(X1, X2, Y1, Y2, Z1, Z2)

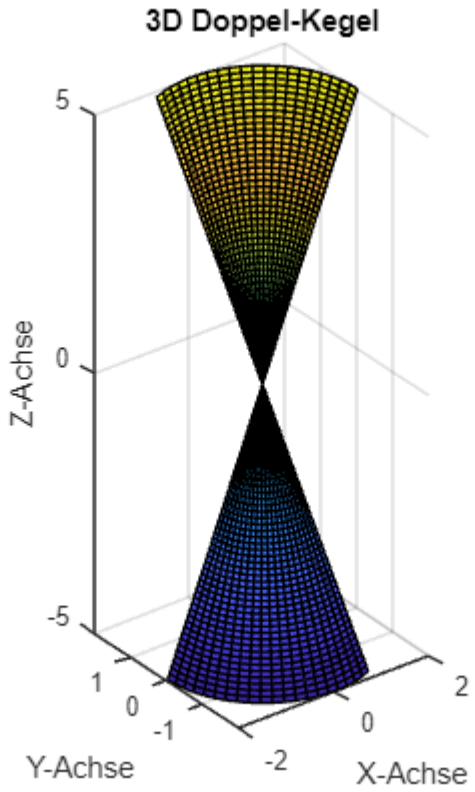
- Diese Funktion erzeugt die 3D-Darstellung des Doppelkegels.
- Sie verwendet **surf()**, um die Oberflächen des oberen und unteren Kegels basierend auf den berechneten Koordinaten darzustellen.
- Die Achsen werden beschriftet und ein Gitternetz sowie ein Titel hinzugefügt, um die Darstellung zu verbessern.

Grafiken :

3D-Darstellung des Doppelkegels:

- Der Doppelkegel wird in 3D gezeigt, indem die X-, Y- und Z-Koordinaten berechnet und verwendet werden, um die Oberfläche des oberen und unteren Kegels darzustellen.

```
% a)
r = 2; % maximale Radius des Kegels
h = 10; % Höhe des Doppelkegels

% meshgrid für die Winkel (theta) und die Höhe (z)
[theta, z1] = meshNetzwerk(h); % die Funktion meshNetzwerk aufrufen, um die Winkel
und Höhenbereiche zu berechnen

% b)
[Theta, Z1] = meshgrid(theta, z1); % meshgrid für den oberen Kegel
[~, Z2] = meshgrid(theta, z1 * -1); % meshgrid für den unteren Kegel

R1 = (r / (h / 2)) * Z1; % Radius des oberen Kegels
R2 = (r / (h / 2)) * Z2; % Radius des unteren Kegels

X1 = R1 .* cos(Theta); % X-Koordinaten für den oberen Kegel
Y1 = R1 .* sin(Theta); % Y-Koordinaten für den oberen Kegel
X2 = R2 .* cos(Theta); % X-Koordinaten für den unteren Kegel
Y2 = R2 .* sin(Theta); % Y-Koordinaten für den unteren Kegel

% c)
% die Funktion zeigeDoppelkegel aufrufen, um den Doppelkegel darzustellen.
zeigeDoppelkegel(X1, X2, Y1, Y2, Z1, Z2)
```



4. Symbolisches Rechnen

Symbolisches Rechnen, Vektoranalysis, Darstellung von skalaren Feldern und Vektorfeldern

Gegeben ist das zweidimensionale Skalarfeld

$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}\,e^{-(x+1)^2-y^2}$$

Überblick

In dieser Aufgabe geht es um symbolisches Rechnen und die Darstellung von Skalar- und Vektorfeldern in MATLAB. Es wird ein zweidimensionales Skalarfeld analysiert und berechnet, gefolgt von der Berechnung des Gradienten, der Divergenz und der Rotation des Feldes. Anschließend werden die Felder grafisch dargestellt. Die Aufgabe besteht aus vier Teilen:

1. 3D-Darstellung des Skalarfeldes.
2. Berechnung des Gradientenfeldes (Vektorfeld).
3. Berechnung von Divergenz und Rotation des Vektorfeldes.
4. Grafische Darstellung des Gradientenfeldes.

Implementierung :

Teil - a) :

3D-Darstellung des Skalarfeldes

- Zunächst definieren wir die **x- und y-Werte** im Bereich von -3 bis +3.
- Wir erstellen ein **Mesh-Grid**, das alle Kombinationen von x- und y-Werten enthält.
- Anschließend berechnen wir das Skalarfeld `f(x, y)` und stellen es in einer 3D-Grafik dar.
- **Ergebnis**: Das Skalarfeld wird in 3D dargestellt, wobei die **x- und y-Werte** auf den Achsen sind und die Z-Achse die Werte des Skalarfeldes zeigt.

Teil - b) :

Berechnung des Gradientenfeldes

- Wir berechnen symbolisch den **Gradienten des Skalarfeldes**. Der Gradient zeigt uns die Richtung der größten Änderung des Skalarfeldes an.
- Dies wird durch die Funktion `gradient()` realisiert.
- **Ergebnis**: Der Gradient des Skalarfeldes wird symbolisch berechnet und liefert ein Vektorfeld, das die Richtungen der Änderungen des Skalarfeldes beschreibt.

Teil - c) :

Berechnung von Divergenz und Rotation

- Nun berechnen wir symbolisch die **Divergenz** und die **Rotation** des Gradientenfeldes.
- **Divergenz**: Gibt an, wie stark sich das Vektorfeld aus einem Punkt heraus ausbreitet.
- **Rotation**: Zeigt, wie stark das Vektorfeld um einen Punkt herumwirbelt.
- **Ergebnis**: Die symbolische Berechnung der Divergenz und Rotation liefert Informationen darüber, wie sich das Vektorfeld verhält: ob es sich ausbreitet (Divergenz) oder wirbelt (Rotation).

Teil - d) :

Grafische Darstellung des Gradientenfeldes

- Zuletzt stellen wir das **Gradientenfeld** grafisch dar. Hier verwenden wir die Funktion **quiver()**, um das Vektorfeld zu zeichnen, das die Richtung und Größe der Veränderungen im Skalarfeld zeigt.
- **Ergebnis**: Das Gradientenfeld wird als Vektorfeld dargestellt, das die Richtung und Stärke der größten Änderung des Skalarfeldes visualisiert.

Grafiken :

Teil - a) : 3D-Darstellung des Skalarfeldes:

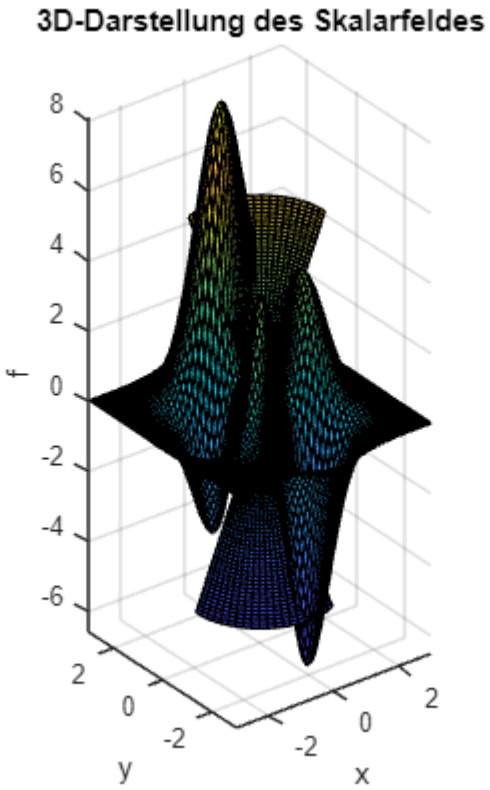
- Diese Grafik zeigt das **Skalarfeld** in einer dreidimensionalen Ansicht. Die X- und Y-Werte werden auf der horizontalen Ebene gezeigt, und das Skalarfeld `f(x, y)` wird auf der Z-Achse dargestellt.
- sie bietet auch einen klaren Überblick über das Verhalten des Skalarfeldes im 3D-Raum.

Teil - b) : Grafische Darstellung des Gradientenfeldes (Vektorfeld):

- Das **Gradientenfeld** wird als Vektorfeld dargestellt. Dabei werden kleine Pfeile verwendet, um die Richtung und Stärke der größten Änderung des Skalarfeldes zu visualisieren.
- Diese Grafik zeigt die Richtung und Stärke der Änderung des Skalarfeldes. Jeder Pfeil zeigt die Richtung der stärksten Änderung an einem bestimmten Punkt im Feld.
- Die Länge der Pfeile repräsentiert die Stärke der Änderung.

a) Stellen Sie das Skalarfeld in einer 3D-Darstellung mit korrekter Achsenbeschriftung für Werte von x und y zwischen -3 und +3 dar.

```
x = -3:0.1:3; % x-Werte von -3 bis +3
y = -3:0.1:3; % y-Werte von -3 bis +3

[X, Y] = meshgrid(x, y); % Erzeugen der Matrizen X und Y für das Mesh-Netzwerk

% Berechnen der Funktion f(x, y) für die Matrix
f = 3*(1 - X).^2 .* exp(-X.^2 - (Y + 1).^2) - 10 * (X/5 - X.^3 - Y.^5) .* exp(-X.^2
- Y.^2) - 1/3 * exp(-(X + 1).^2 - Y.^2);

surf(X, Y, f); % das Skalarfeld in einer 3D-Darstellung

% Achsenbeschriftungen
xlabel('x');
ylabel('y');
zlabel('f');
title('3D-Darstellung des Skalarfeldes'); % Titel des Plots
grid on;
```



b) Berechnen Sie **symbolisch** das Gradientenfeld des angegebenen Skalarfeldes.

```
syms x y f(x,y);

% die funktion des Skalarfeldes
f(x,y) = 3*(1 - x)^2 * exp(-x^2 - (y + 1)^2) - 10 * (x/5 - x^3 - y^5) * exp(-x^2 -
y^2) - 1/3 * exp(-(x + 1)^2 - y^2);

% symbolische Berechnung des Gradientens des Skalarfeldes
gradientenfeld_Vektorfeldes = gradient(f(x,y), [x, y])
```

```
gradientenfeld_Vektorfeldes =
```

$$\begin{pmatrix} \frac{\sigma_4\,(2\,x+2)}{3} + 3\,\sigma_1\,(2\,x-2) + \sigma_2\,(30\,x^2-2) - 6\,x\,\sigma_1\,(x-1)^2 - 2\,x\,\sigma_2\,\sigma_3 \\ \frac{2\,y\,\sigma_4}{3} + 50\,y^4\,\sigma_2 - 3\,\sigma_1\,(2\,y+2)\,(x-1)^2 - 2\,y\,\sigma_2\,\sigma_3 \end{pmatrix}$$

where

$$\sigma = \mathrm{e}^{-(y+1)^2-x^2}$$

$$\sigma_2 = \mathrm{e}^{-x^2-y^2}$$

$$\sigma_3 = 10\,x^3 - 2\,x + 10\,y^5$$

$$\sigma_4 = \mathrm{e}^{-(x+1)^2-y^2}$$

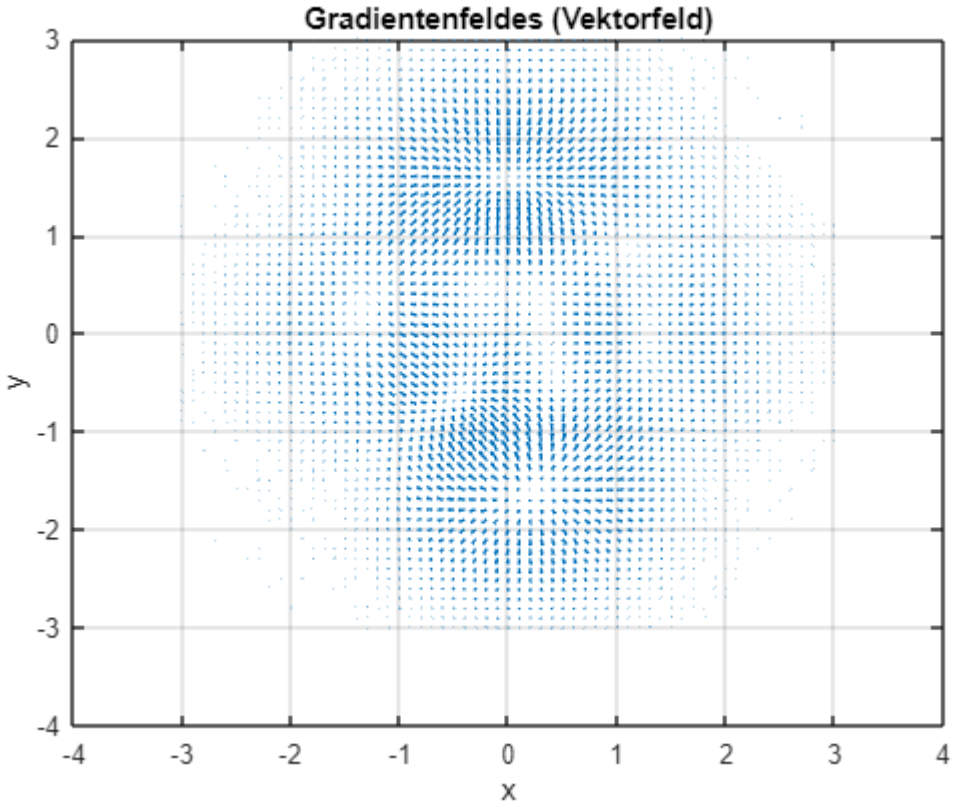
c) Berechnen Sie **symbolisch** die Rotation und die Divergenz des in Aufgabenteil b enthaltenen Vektorfeldes.

```
% Die Divergenz berechnen
divergenz = diff(gradientenfeld_Vektorfeldes(1), x) +
diff(gradientenfeld_Vektorfeldes(2), y);
% Rotation berechnen
rotation = diff(gradientenfeld_Vektorfeldes(2), x) -
diff(gradientenfeld_Vektorfeldes(1), y);
```

d) Stellen Sie das in Aufgabenteil b erhaltene Gradientenfeld (Vektorfeld) graphisch dar.

Die Aufgabe ist mit einem MATLAB Skript zu lösen.

```
% Darstellung des Gradientenfeldes
figure

quiver(X, Y, subs(gradientenfeld_Vektorfeldes(1), {x, y}, {X, Y}),
subs(gradientenfeld_Vektorfeldes(2), {x, y}, {X, Y})); % Darstellung des
Vektorfeldes - Gradientenfeldes
title('Gradientenfeldes (Vektorfeld)');
xlabel('x');
ylabel('y');
grid on;
```



5. Signalverarbeitung in Simulink

Simulink, Übertragungsfunktion, Lösen von Differentialgleichungen, Impuls- und Sprungantwort

Der nachfolgend dargestellte RLC Serienschwingkreis ist wie folgt beschalte:. Die Eingangsgröße ist die Spannung U(t), die Ausgangsgröße die Spannung $U_L(t)$über dem Kondensator.

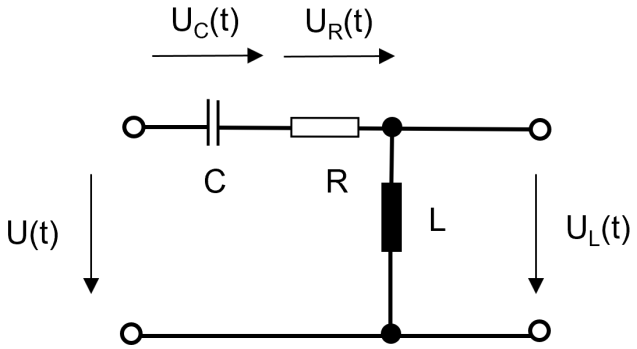


BIld: Serienschwingkreis als Tiefpass 2. Ordnung beschaltet

Es handelt sich um ein elektrisches, schwingungsfähiges System, welches mit einer Differentialgleichung 2. Ordnung beschreibbar ist. Die DGL ist durch die Strom- Spannungsbeziehungen der einzelnen Komponenten

$$U_R(t) = R \cdot I(t)$$

$$U_L(t) = L \cdot \frac{d}{\mathrm{dt}} \cdot I(t)$$

$$U_C(t) = \frac{1}{C} \cdot \int I(t)\,\mathrm{dt} \qquad \text{bzw.} \qquad I(t) = C \cdot \frac{dU_C(t)}{\mathrm{dt}}$$

und der Maschenregel

$$U(t) = U_R(t) + U_L(t) + U_C(t)$$

herleitbar zu:

$$U(t) = \mathrm{LC} \cdot \frac{d^2U_C(t)}{dt^2} + \mathrm{RC} \cdot \frac{dU_C(t)}{\mathrm{dt}} + U_C(t)$$

Die Bauteilewerte des Serienschwingkreises seien:

- R = 5 Ω
- L = 1 μH
- C = 10 nF

a) Erstellen SIe ein **Simulink Modell** zur Berechnung der Spannungen $U_L(t)$und des Stromes $I(t)$ auf. Lösen Sie dazu die Differentialgleichung nach der höchsten Ableitung von $U_C(t)$ auf und implementieren Sie das zugehörige Simulink Modell für obige DGL.

b) Stellen Sie die Spannung $U_L(t)$ und den Strom $I(t)$ für die Zeit von t = 0 bis t = 5 μs für die folgende sprunghafte Änderung der Eingangsspannung $U(t)$ dar:

- zum Zeitpunkt t = 0 ändert sich die Eingangsspannung von 0 auf 1 Volt (Einheitssprung)

Achten Sie auf eine geeignete Schrittweite und Simulationsdauer.

c) Berechnen Sie zusätzlich i**n Simulink** die Spannung $U_C(t)$ mit Hilfe des Blockes 'Transfer Fcn' für die entsprechende Eingangsanregung gemäß Aufgabenteil b.

d) Stellen Sie die Ergebnisse für $U_L(t)$ der Simulink Simulation im Liveskript in einem Plot dar und vergleichen Sie das Ergebnis von Aufgabenteil b) und c).

Überblick

In dieser Aufgabe modellieren und simulieren wir einen elektrischen Serienschwingkreis in Simulink. Wir berechnen die Spannungen und Ströme dieses Systems und vergleichen die Ergebnisse verschiedener Ansätze. Die Aufgabe besteht aus vier Teilen:

1. **Teil (a)**: Erstellung eines Simulink-Modells zur Berechnung von Spannung und Strom im Schwingkreis.
2. **Teil (b)**: Simulation der Spannung und des Stroms bei einer sprunghaften Änderung der Eingangsspannung.

3. **Teil (c)**: Berechnung der Spannung mithilfe eines Transfer-Funktion-Blocks in Simulink.
4. **Teil (d)**: Vergleich der Simulationsergebnisse aus Teil (b) und (c).

Implementierung :

Teil - a):

- In diesem Teil erstellen wir ein **Simulink-Modell** für den RLC-Serienschwingkreis.
- Das Modell löst die Differentialgleichung für das System, indem die Strom- und Spannungsverläufe berechnet werden.**Wichtige Komponenten im Simulink-Modell**:
- **Spannungsquelle**: Liefert die Eingangsspannung.
- **Resistor (R)**, **Inductor (L)** und **Capacitor (C)**: Repräsentieren die Bauteile des RLC-Kreises.
- **Scope**: Zeigt die Spannungs- und Stromverläufe an.
- Das Modell ist in der Datei `aufgabe5_a_b.slx` implementiert und bereit zur Ausführung.

Teil - b):

- In diesem Schritt simulieren wir den Schwingkreis, wobei die Eingangsspannung sprunghaft von 0 auf 1 Volt ansteigt (Einheitssprung).
- Die Spannungen und Ströme werden für einen Zeitraum von **0 bis 5 Sekunden** berechnet und dargestellt.
- Wir verwenden geeignete Schrittweiten, um die Simulation präzise zu halten.
- Das Modell ist in der Datei `aufgabe5_a_b.slx` implementiert und bereit zur Ausführung.
- **Ergebnis:** die **Spannung über dem Kondensator** zeigt typische Schwingungen aufgrund der Eigenschaften des RLC-Kreises. Der **Stromverlauf** zeigt die Reaktion des Stroms auf den Spannungssprung.

Teil - c):

- Hier verwenden wir den **Transfer-Funktion-Block** in Simulink, um die Spannung über dem Kondensator zu berechnen.
- Die Simulation erfolgt erneut für den Zeitraum von **0 bis 5 Sekunden**, unter denselben Bedingungen wie in Teil (b).
- Der Transfer-Funktion-Block modelliert das Systemverhalten basierend auf der Übertragungsfunktion des Schwingkreises.
- Das Modell ist in der Datei `aufgabe5_c.slx` implementiert und bereit zur Ausführung.
- **Ergebnis:** Die Spannung wird ebenfalls über den angegebenen Zeitraum berechnet, basierend auf der Übertragungsfunktion.

Teil - d):

- In diesem Teil vergleichen wir die Spannungsverläufe aus den Teilen (b) und (c).
- Beide Simulationen werden in einem Diagramm zusammengeführt, um zu sehen, wie gut die Ergebnisse übereinstimmen.
- Das Modell ist in der Datei `aufgabe5_d.slx` implementiert und bereit zur Ausführung.

- **Ergebnis**: Beide Spannungsverläufe sollten ähnliche Verläufe zeigen, da sie dasselbe System beschreiben.

Grafiken:

Vergleich der Spannungen aus Teil (b) und (c) durch die Funktion - zeigeSpannungen

- Beide Spannungsverläufe, die in Teil (b) und Teil (c) berechnet wurden, werden in einer **gemeinsamen Grafik** dargestellt.
- Die Darstellung zeigt, wie gut die beiden Methoden (numerische Berechnung und Transfer-Funktion) übereinstimmen.

Funktion - zeigeSpannungen :

Die Funktion `zeigeSpannungen` erstellt den Vergleich der beiden Spannungsverläufe. Sie hat folgende Aufgaben:

1. **Zeitvektor erstellen**: Die Funktion erstellt einen Zeitvektor von 0 bis 5 Sekunden für die x-Achse.
2. **Spannungen plotten**: Zwei Spannungsverläufe (aus Teil b und Teil c) werden im gleichen Diagramm geplottet.

```
zeigeSpannungen(spannung_ul, spannung_uc);
```

```
Unrecognized function or variable 'spannung_ul'.
```

```
function [theta, z1] = meshNetzwerk(hoehe)
    theta = 0:0.1:2; % Vektor von 0 bis 2*pi aufgeteilt in 200 Schritte
    z1 =  0:0.1:hoehe/2;  % Vektor von 0 bis zur Hälfte der Gesamthöhe in 20
Schritte aufgeteilt
end

%  Doppelkegels in 3D darstellen.
function zeigeDoppelkegel(X1, X2, Y1, Y2, Z1, Z2)
    figure
    surf(X1, Y1, Z1); % Oberfläche des oberen Kegels darstellen
    hold on

    surf(X2, Y2, Z2); % Oberfläche des unteren Kegels darstellen

    xlabel('X-Achse')
    ylabel('Y-Achse')
    zlabel('Z-Achse')
    title('3D Doppel-Kegel')
    axis equal
    grid on
end
```

```
function zeigeSpannungen(spannung1, spannung2)

    zeit = linspace(0, 5e-6, length(spannung1));  % zeit vektor

    figure;

    plot(zeit, spannung1, 'DisplayName', 'Spannung UL');

    hold on;

    plot(zeit, spannung2, 'DisplayName', 'Spannung UC');

    title('Spannung UL vs UC');
    xlabel('Zeit in s');
    legend show;
end
```