Universität Bremen

Universität Bremen
Fachbereich Physik, Elektro- und Informationstechnik
Institut für Theoretische Elektrotechnik und Mikroelektronik (ITEM)

Lineare Systeme

Programmieraufgaben in Python

von

Prof. Dr.-Ing. Steffen Paul

25.07.2022

ITEM.me

Inhaltsverzeichnis

Vorwort

In der Vorlesung *Lineare Systeme* werden verschiedene Methoden der Signal- und Systembeschreibung vorgestellt. Die behandelten Signale beschränken sich auf bestimmte Testsignale, wie etwa den Diracimpuls oder die Sprungfunktion.

Mit diesen Kenntnissen lassen sich dann Systeme zur Signalverarbeitung modellieren und berechnen, um daraus später dann z.B. eine Schaltung zu bauen. Mit solchen Systemen lassen sich z.B. Sprach- oder Videosignale verarbeiten. Das kann natürlich nicht mehr mit Papier und Bleistift geschehen, sondern nur mit Hilfe von Computerprogrammen. Damit lässt sich die Verarbeitung der Signale im Detail untersuchen bzw. umgekehrt kann geprüft werden, ob die entworfenen Signalverarbeitungssysteme die geforderten Eigenschaften besitzen.

In der Signal- und Systemtheorie werden abgetastete Signale als Vektoren geschrieben. Zeitkontinuierliche Signale lassen sich durch Abtastung unter Berücksichtigung des Abtasttheorems ebenfalls als Vektoren speichern.

Die Computermodellierung von Signalen und Systemen erfolgt mit den verschiedensten Programmiersprachen, z.B. C oder C++. Das C-Programm muss vor der Ausführung compiliert werden und die Programmierung erfordert Übung und Erfahrung. Interaktives Arbeiten und Experimentieren mit Signalen ist mit C nicht möglich. In der numerischen linearen Algebra wird mit Matrizen und Vektoren gerechnet. Verfahren zur Matrixinversion, Matrixzerlegung, Eigenwertberechnung u.ä. wurden in den Bibliotheken EISPACK (Matrix Eigensystem Package) und LINPACK (Linear Equation Package) in der Programmiersprache FORTRAN zusammengetragen.

Cleve Mohler hat Ende der 70-iger Jahre für seine Studenten in Kursen zur numerischen linearen Algebra einen interaktiven Matrixrechner (MATLAB = Matrix Laboratory) entwickelt, um diese Pakete zu nutzen. Das war der Grundstein für das Programmsystem MATLAB. Daraus ist im Laufe der Jahrzehnte eine mächtige und umfangreiche

Programmumgebung für interaktives Arbeiten mit Matrizen und Vektoren entstanden. Matlab zählt zu den sog. Skriptsprachen, also Interpreterprogrammiersprachen.

Wie oben schon erwähnt, nutzt die Signalverarbeitung ebenfalls Matrizen und Vektoren und so ist es nicht überraschend, dass MATLAB dort breit eingesetzt wird. Gleiches gilt für die Regelungstechnik. MATLAB wird kommerziell vertrieben und ist nicht frei verfügbar.

In der Softwareentwicklung ist der Open-Source-Gedanke weit verbreitet, d.h. entwickelte Software wird kostenfrei oder zu sehr geringen Kosten zur Verfügung gestellt. So entstanden parallel zu MATLAB andere, kostenfreie Systeme, wie etwa GNU OCTAVE mit ähnlichen Eigenschaften. Sie sind vom Syntax zu MATLAB weitgehend kompatibel, haben aber keine sehr große Verbreitung gefunden.

In den vergangenen Jahren hat Python als neue Skriptsprache besondere Popularität erlangt, da sie einfach zu erlernen und open source verfügbar ist. Durch zahlreiche Zusatzpakete ist es auch für das Rechnen mit Vektoren sehr gut geeignet.

Daher soll dieses Paket genutzt werden, um die Kenntnisse aus Vorlesung und Übung zu vertiefen. **Ziel ist das praktische Experimentieren mit Signalen udn Systemen am Computer.** Insbesondere die Pakete `numpy`, `matplotlib` und `scipy` werden im weiteren verwendet. Das erlernte Wissen kann dann leicht auf andere Skriptsprachen übertragen werden.

Die Programmieraufgaben sind so gestellt, dass die eigenständig bearbeitet werden können. Voraussetzung ist die Installation von Python auf einem Computer mit Windows, MacOS oder Linux. Die Software ist z.B. als Gesamtpaket erhältlich unter `https://www.anaconda.com/products/individual`.

Bremen, Sommer 2022 Steffen Paul

Übungsdurchführung

Die Pythonübung dient zur Vertiefung des Verständnisses des Vorlesungsstoffes.

Das erfordert von Ihrer Seite Mitarbeit bei der Bearbeitung der Übungsaufgaben.

Angaben zum erforderlichen Umfang für erfolgreiches Absolvieren der Übung werden in der Vorlesung gemacht.

Aufgabenblatt 1 dient zur Installation und zum Kennenlernen der Python-Umgebung. Die Aufgabenblätter 2 und 3 werden als betreute Übung durchgeführt. Aufgabenzettel $4-7$ sind während des Semesters abzugeben.

Bitte beachten Sie für die Abgabe folgende Hinweise:

- Abgabe der Lösung als Python-Skript mit Dateiendung `.py` (keine Jupyter Notebooks) sowie als `.pdf`-Datei mit allen erstellen Plots und Konsolenausgaben.
- Dateibezeichnung:
  <Matrikelnummer>_<Studiengang>_Blatt_<Nummer>.py/pdf (Studiengang: ET, WI, SY)
- Die Dateien bitte in Stud.IP den bereitgestellten Ordner hochladen. Deadline der jeweiligen Abgaben wird über Stud.IP verkündet.
- Zur Prüfung der Korrektheit lassen wir die Skripte laufen. Anhand der Programmausgabe wird die Korrektheit und Vollständigkeit der Lösung bewertet.
- Erläuterungen zur Ihren Rechnungen können Sie als Kommentar einfügen.

Literatur

Informationen zu Python und den für die Übung wichtigen Paketen finden Sie z.B. unter:

- Hans-Bernhard Woyand: Python für Ingenieure und Naturwissenschaftler. Hanser (SUUB online).
- Christoph Schäfer: Schnellstart Python. Springer (SUUB online)
- `https://www.python.org/`
- `https://numpy.org/`
- `https://www.scipy.org/`
- `https://matplotlib.org/`
- `https://docs.scipy.org/doc/scipy/reference/index.html`

Darüber hinaus gibt es noch zahlreiche weitere Bücher in der SUUB und Online-Quellen, z.B. `https://docs.scipy.org/doc/scipy/reference/index.html`.

AUFGABENBLATT 1

Installation und erste Schritte

Am einfachsten gelingt die Installation, wenn eine komplette Distribution von Python installiert wird. Dann ist zwar mehr auf der Festplatte, als benötigt wird, dafür sollte es aber keine Installationsprobleme geben und regelmäßige Updates sind meist auch dabei.

Eine verbreitete Distribution wird von Anaconda bereitgestellt `https://www.anaconda.com/products/individual`. Mit dabei ist hier die integrierte Entwicklungsumgebung Spyder aber auch eine Browserschnittstelle für Jupyter-Notebooks. Mit Spyder arbeitet man ähnlich wie in MATLAB. Jupyter entspricht etwa der Arbeitsweise mit Mathematica-Notebooks.

Installieren Sie Python mit Hilfe des Paketes Anaconda.

Mit der Installation von Anaconda sind auch alle Python-Pakete für Signalverarbeitung und Systemtheorie installiert. Dies sind:

- `numpy` für numerisches Rechnen, lineare Algebra (`https://numpy.org/`),
- `scipy` u.a. für Signalverarbeitung (`https://www.scipy.org/`),
- `matplotlib` für graphische Ausgabe (`https://matplotlib.org/`),
- `sympy` für symbolisches Rechnen (`https://www.sympy.org/`).

Zur Einarbeitung in Python eignet ist das Buch

- Hans-Bernhard Woyand: Python für Ingenieure und Naturwissenschaftler, Hanser Verlag zu empfehlen. Es ist in der SuUB als ebook verfügbar.

Weiterführende Literatur sind z.B.

- `https://www.python.org/`
- Robert Johansson, Numerical Python, APress

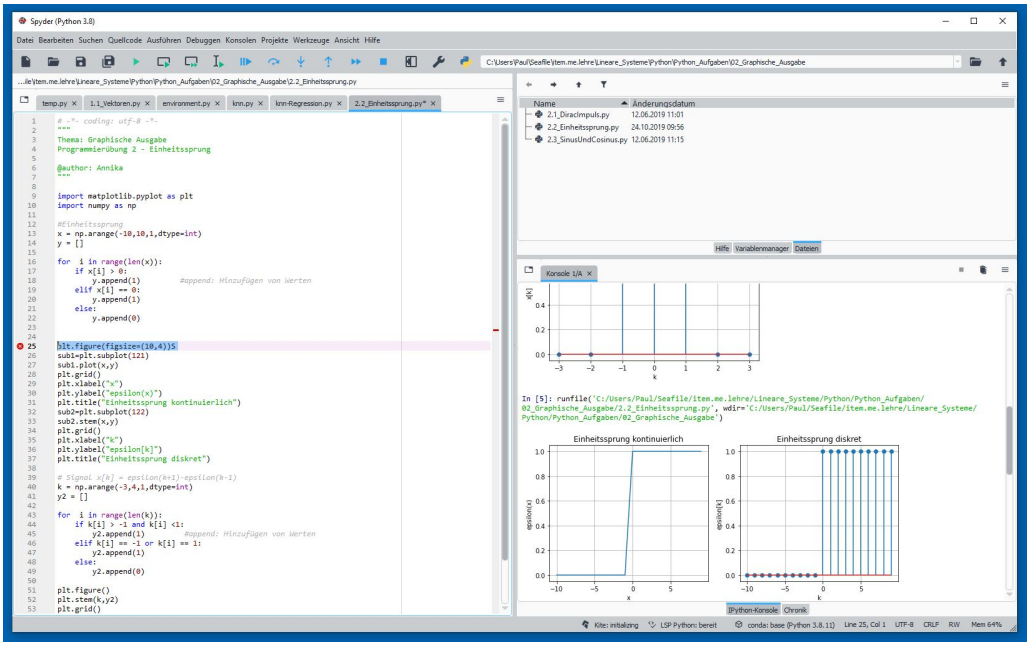

Abbildung 1.1: Programmierumgebung Spyder.

Zahlreiche weitere Bücher, auch als ebook finden Sie in der Uni-Bibliothek.

Eine Auflistung von wichtigen vordefinierten Funktionen für die Systemtheorie finden Sie unter: `https://docs.scipy.org/doc/scipy/reference/signal.html`.

Für das Erstellen von Python-Programmen eignet sich die Programmierumgebung Spyder (Bild 1.1). Damit kann man ein Programm entwickeln, debuggen und ausführen. Die Benutzung ist weitgehend selbsterklärend.

Alternativ kann mit JupyterLab oder Jupyter Notebook auch ein sog. Notebook genutzt werden, mit dem Text zwischen den Rechnungen (Bild 1.2). Die Dateien werden in einem Webbrowser geöffnet.

Jupyter Notebook (Bild 1.2) und JupyterLab (Bild 1.3) unterscheiden sich darin, ob sie mehrere Dateien in einem Tab (JupyterLab) oder für jede Datei ein eigenen Tab öffnen.

Dateinamen von Pythonprogramme enden auf `.py`. Dateinamen von JupyterLab- und Jupyter Notebook-Dateien enden auf `.ipynb`.

Für die Übungen reicht die Umgebung Spyder, da ohnehin nur `.py`-Dateien abgegeben werden sollen.

Python besteht aus einer großen Zahl von Zusatzpaketen, die verschiedene Funktionen bereitstellen. Bevor man diese Funktionen nutzen kann, muss das jeweilige Modul geladen

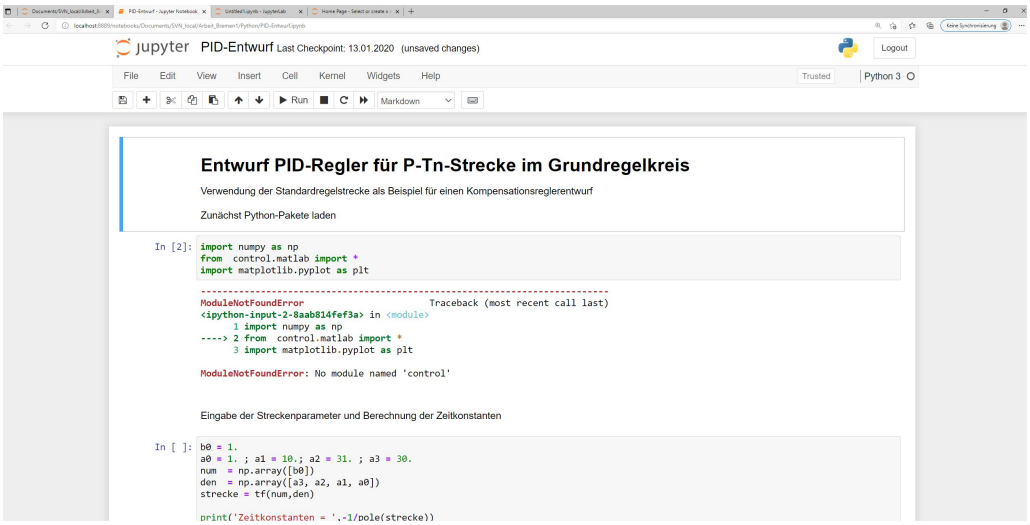

Abbildung 1.2: Programmierumgebung Jupyternotebook.

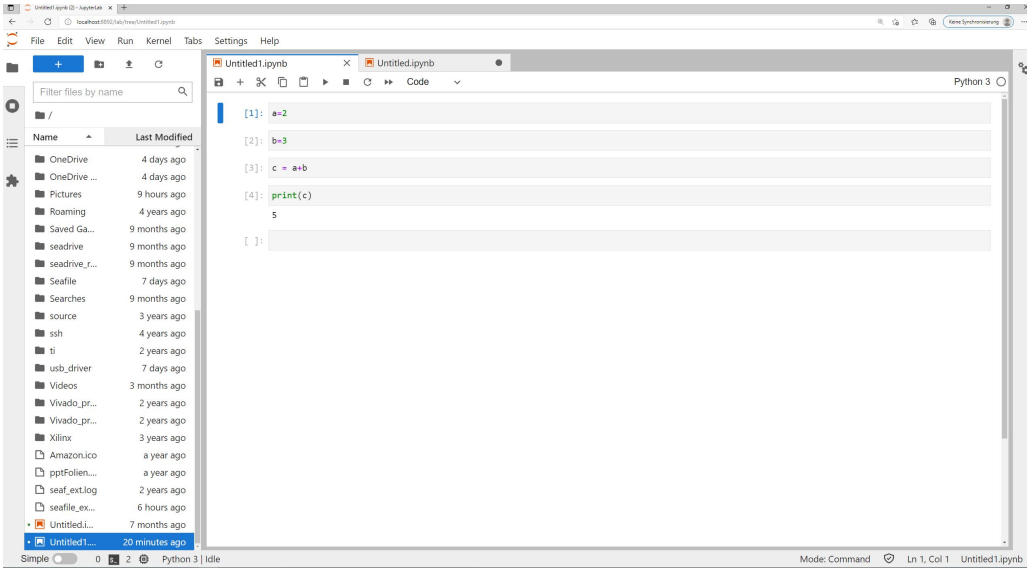

Abbildung 1.3: Programmierumgebung JupyterLab.

werden. Wir wollen das an einem Beispiel anschauen, bei dem die Determinante einer Matrix berechnet werden soll.

Da die Modulnamen Programme sehr unübersichtlich machen, ist es gängige Praxis einen Kurznamen einzuführen:

```
import numpy as np
from numpy.linalg import det
A = np.array([[1., 2.],[3., 4.]])
print det(A)
```

In Zeile 1 wird das Modul `numpy` geladen und mit der Kurzbezeichnung np versehen. Aus dem Paket `numpy` wird in Zeile 2 die Determinantenfunktion aus dem Unterpaket linalg geladen. Dadurch vermeidet man den umständlichen Aufruf `numpy.linalg.det(A)`.

Sie können diese Beispiel gleich als Test für die Funktion Ihrer Installation nutzen.

Zur Dokumentation von Programmcode sind Kommentare unverzichtbar. Sie können in folgender Form eingegeben werden:

```
"""
This is a comment
written in
more than just one line
"""
```

oder

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

oder bei einzeiligen Kommentaren:

```
#This is a comment
```

AUFGABENBLATT 2

Umgang mit Vektoren und Matrizen

2.1 Programmierübung - Vektoren

Nutzen Sie Funktionen aus dem Paket `numpy`.

a) Erstellen Sie den Zeilenvektor $\boldsymbol{a}$

   i) durch Verwendung der Funktion `arange()`

   ii) durch Verwendung der Funktion `array()`

   iii) durch Verwendung der Funktion `linspace()`.

b) Wandeln Sie den Zeilenvektor $\boldsymbol{a}$ in den Spaltenvektor $\boldsymbol{b}$

   i) durch Verwendung der Funktionen `reshape` und `shape`

   ii) durch Verwendung der Funktion `newaxis`

   um.

c) Erstellen Sie einen Zufallsvektor $\boldsymbol{c}$ des Datentyps `float` und der Länge 10.

d) Erstellen Sie einen Zufallsvektor $\boldsymbol{d}$ des Datentyps `int`. Die Werte des Vektors liegen im Bereich [0,100] und seine Länge beträgt 10.

$$\boldsymbol{a} = \begin{pmatrix} 1 & 3 & 5 & 7 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \end{pmatrix}$$

2.2 Programmierübung - Matrizen

a) Erstellen Sie die Matrix $\boldsymbol{A}$,

   i) durch Verwendung der Funktion `array()`

   ii) durch Verwendung der Funktion `matrix()`

b) Geben Sie die Anzahl der Zeilen und Spalten der Matrix $\boldsymbol{A}$ aus.

c) Geben Sie die zweite Zeile der Matrix $\boldsymbol{A}$ aus.

d) Geben Sie den Wert der 2. Spalte und der 3. Zeile aus.

e) Geben Sie den Teilbereich der ersten beiden Zeilen bis einschließlich der 3. Spalte aus.

f) Erstellen Sie die Matrix $\boldsymbol{B}$. (Hinweis: Verwenden Sie die Funktion `zeros()`.)

g) Erstellen Sie die Matrix $\boldsymbol{C}$. (Hinweis: Verwenden Sie die Funktion `ones()`.)

h) Erstellen Sie die Einheitsmatrix $\boldsymbol{D}$. (Hinweis: Verwenden Sie die Funktion `identity()`.)

i) Erstellen Sie eine 4 x 5 Zufallsmatrix $\boldsymbol{E}$. Die Werte der Matrix liegen im Bereich [10,70].

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{C} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad \boldsymbol{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Hinweis: Alle Funktionen stammen aus dem Paket numpy.

2.3 Programmierübung - Rechenoperationen

a) Berechnen Sie das Skalarprodukt $\boldsymbol{a} \cdot \boldsymbol{b}$.

b) Berechnen Sie das Kreuz- bzw. Vektorprodukt $\boldsymbol{a} * \boldsymbol{b}$.

c) Multiplizieren Sie die Matrizen $\boldsymbol{A} \cdot \boldsymbol{B}$,

i) indem die Matrizen als `array()` definiert werden

ii) indem die Matrizen als `matrix()` definiert werden.

d) Berechnen Sie das Matrix-Vektorprodukt $\boldsymbol{A} \cdot \boldsymbol{a}$.

e) Berechnen Sie die inverse Matrix $\boldsymbol{A}^{-1}$. Hinweis: Verwenden Sie das Paket `numpy.linalg`.

$$\boldsymbol{a} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}, \qquad \boldsymbol{A} = \begin{pmatrix} 1 & 5 & 5 \\ 1 & 4 & 4 \\ 1 & 5 & 4 \end{pmatrix}, \qquad \boldsymbol{B} = \begin{pmatrix} 1 & 3 & 4 \\ 2 & 8 & 6 \\ 1 & 3 & 1 \end{pmatrix}$$

AUFGABENBLATT 3

Graphische Ausgabe, Zeitdiskrete Signale

3.1 Programmierübung - Plot-Funktionen und Dirac-Impuls

Stellen Sie den kontinuierlichen und zeitdiskreten Dirac-Impuls graphisch dar. Es soll der Bereich von $[-10, 10]$ abgebildet werden. Achten Sie auf Achsenbeschriftung und Titel. Erstellen Sie hierfür ihre eigenen Plot-Funktionen für zeitdiskrete (`stem`) und kontinuierliche Plots (`plot`). Diese Funktionen können in den weiteren Aufgaben wiederverwendet werden. Speichern Sie die Plots als pdf-Datei.

Hinweis: Verwenden Sie die Funktion `unit_impulse()` aus dem Paket `scipy.signal`, sowie das Paket `matplotlib.pyplot` und die darin enthaltenen Funktionen `plot` und `stem`, sowie `subplot` und `savefig`. Im folgenden ist die Syntax für eine Funktionsdefinition gezeigt:

```
def function_name(arguments):
    block of code
    return value
```

3.2 Programmierübung - Einheitssprung

a) Stellen Sie den Einheitssprung in kontinuierlicher und zeitdiskreter Form graphisch dar. Es soll der Bereich von $[-10, 10]$ abgebildet werden. Achten Sie auf Achsenbe-

schriftung und Titel. (Hinweis: Verwenden Sie zur Berechnung des Einheitssprunges eine for-Schleife und eine if-Abfrage)

b) Stellen Sie die zeitdiskrete Funktion $x[k]$ im Bereich $[-3, 3]$ graphisch dar.

$$x[k] = \epsilon[k+1] - \epsilon[t-1]$$

3.3 Programmierübung - Sinus und Cosinus

a) Stellen Sie die Sinusfunktion $y_1(t)$ mit einer Amplitude von A=2 im Bereich von $[-1, 1]$ in 200 Schritten graphisch dar. Zusätzlich soll die Cosinusfunktion $y_2(t)$ mit halber Amplitude dargestellt werden. Achten Sie auf Achsenbeschriftung, Titel und Legende.

$$y_1(t) = A * sin(2 * \pi * t)$$
$$y_2(t) = \frac{A}{2} * cos(2 * \pi * t)$$

b) Stellen Sie die Sinusfolge $y[k]$ im Bereich $[-10, 10]$ dar. Die Abtastrate beträgt 50. Was passiert, wenn die Abtastrate verringert bzw. erhöht wird?

$$y[k] = sin(k)$$

AUFGABENBLATT 4

Zeitdiskrete Signale - Darstellung und Verhalten im Zeitbereich

4.1 Programmierübung - Diskrete Fouriertransformation (DFT)

Stellen Sie den Zeit- und Spektralbereich der folgenden Funktionen mit Hilfe der diskreten Fouriertransformation dar. (Hinweis: Verwenden Sie die Funktionen `numpy.fft.fft()` und `numpy.fft.fftfreq()`)

a) Impuls-Funktion

$$x[k] = \begin{cases} 0 & \text{für } -50 \leq k \leq -1 \\ 1 & \text{für } k = 0 \\ 0 & \text{für } 1 \leq k \leq 50 \end{cases}$$

b) Rechteck-Funktion

$$x[k] = \begin{cases} 0 & \text{für } -50 \leq k < -20 \\ 1 & \text{für } -20 \leq k \leq 20 \\ 0 & \text{für } 21 \leq k \leq 50 \end{cases}$$

4.2 Programmierübung - Faltung diskreter Signale

Berechnen Sie die Faltungsprodukte $x_1[k] * x_2[k]$, $x_2[k] * x_1[k]$ und $x_2[k] * x_2[k]$. Stellen Sie die beiden Signale, sowie die Ergebnisse der Faltungsprodukte graphisch dar. (Hinweis: Verwenden Sie die Funktion numpy.convolve().)

$$x_1[k_1] = 0.9^{k_1} \text{ mit } 0 \leq k_1 < 20$$
$$x_2[k_2] = 1 \text{ für } 0 \leq k_2 < 5$$

4.3 Programmierübung - Stabilität

a) Gegeben sind folgende Systeme:

$$y[k] + \frac{1}{4}y[k-1] - \frac{1}{4}y[k-2] - \frac{1}{16}y[k-3] = x[k] + 2x[k-1]$$

$$y[k] + y[k-2] + y[k-4] = x[k] + 2x[k-1]$$

Untersuchen Sie die Stabilität durch eine Simulation. Welches System ist stabil?

b) Gegeben ist das System

$$y[k] - \frac{7}{4}y[k-1] - \frac{1}{2}y[k-2] = x[k].$$

Untersuchen Sie die Stabilität mit dem speziellen Eingangssignal

$$x[k] = \{1, -2, 0, 0, \ldots\}$$

Was ist Ihre Schlussfolgerung bzgl. Stabilität des Systems?

AUFGABENBLATT 5

Zeitdiskrete Signale - Darstellung und Verhalten im Bildbereich

5.1 Programmierübung - Partialbruchzerlegung

Gegeben ist die Übertragungsfunktion

$$H(z) = \frac{z}{z^2 + \frac{1}{2}z - \frac{3}{16}}.$$

a) Ermitteln Sie die Partialbruchzerlegung von $H(z)/z$.

b) Stellen Sie Impuls- und Sprungantwort des Systems dar.

c) Bestimmen Sie Nullstellen, Polstellen und Verstärkungsfaktor mit der Funktion `scipy.signal.tf2zpk`.

d) Ermitteln Sie für das System eine Zustandsraumdarstellung mit der Funktion `scipy.signal.tf2ss`.

5.2 Programmierübung - Übertragungsfunktion, Betragsnullstellen

Gegeben ist die Übertragungsfunktion

$$H(z) = \frac{z^2 + 1}{z^2 - z + 0.5}.$$

a) Stellen Sie den Frequenzgang nach Betrag und Phase dar.

b) Mit welchem Signalen kann das System angeregt werden, so dass am Ausgang nach dem Einschwingvorgang das Signal 0 erscheint? Weisen Sie Ihr Ergebnis durch eine Simulation nach. Hinweis: Nutzen Sie die Funktion `scipy.signal.lfilter`.

5.3 Programmierübung - Tiefpass-Transformationen

a) Entwerfen Sie einen Tiefpass mit einem FIR-Filter mit folgender Spezifikation: Cut-Off-Frequenz $f_G = 0.2$ und Filtergrad 5. Nutzen Sie dafür die Funktion `scipy.signal.firwin`. Das implementierte Entwurfsverfahren war nicht Teil der Vorlesung, ist für die Aufgabe aber auch nicht relevant.

b) Prüfen Sie, dass es sich tatsächlich um einen Tiefpass handelt.

c) Transformieren Sie das Filter in einen Hochpass. Weisen Sie die Funktion nach.

d) Transformieren Sie das Filter in einen Bandpass. Weisen Sie die Funktion nach.

AUFGABENBLATT 6

Zeitkontinuierliche Signale - Darstellung und Verhalten im Zeitbereich

6.1 Programmierübung - Fouriertransformation

Gegeben ist das Signal $x(t) = \frac{\sin 2t}{\pi t}$.

a) Stellen Sie das Signal $x(t)$ sowie das numerisch ermittelt Spektrum des Signals $|X(j\omega)|$ dar.

b) Das Signal wird jetzt moduliert mit $m(t) = \cos \omega_0 t$. Untersuchen Sie das Spektrum des modulierten Signals für $\omega_0 = 2$ und $\omega_0 = 4$.

6.2 Programmierübung - Numerische Lösung von Differentialgleichungen

Gegeben ist das folgende Anfangswertproblem

$$y'(t) = f(x, y) = 2 \cdot y + x, \qquad y(0) = 1$$

a) Lösen Sie das Anfangswertproblem näherungsweise mit dem expliziten Euler-Verfahren.

$$y_{k+1} = y_k + h \cdot f(x_k, y_k) \qquad \text{mit} \qquad k = 0, 1, 2, ...$$
$$x_k = x_0 + k \cdot h \qquad \text{mit} \qquad k = 0, 1, 2, ...$$

b) Lösen Sie das Anfangswertproblem näherungsweise mit dem impliziten Euler-Verfahren.

$$\begin{aligned} y_{k+1} &= y_k + h \cdot f(x_{k+1}, y_{k+1}) && \text{mit} && k = 0, 1, 2, ... \\ x_k &= x_0 + k \cdot h && \text{mit} && k = 1, 2, ... \end{aligned}$$

c) Lösen Sie das Anfangswertproblem näherungsweise mit der Trapez-Methode.

$$\begin{aligned} y_{k+1} &= y_k + \frac{h}{2} \cdot (f(x_{k+1}, y_{k+1}) + f(x_k, y_k)) && \text{mit} && k = 0, 1, 2, ... \\ x_k &= x_0 + k \cdot h && \text{mit} && k = 1, 2, ... \end{aligned}$$

Berechnen Sie die Lösungen zu der Schrittweite $h = 0,25$ und $N = 4$ Iterationen. Die Lösung liegt im Bereich x = [0,1]. Stellen Sie alle Ergebnisse, sowie die exakte Lösung $y(x) = \frac{5}{4} \cdot \exp(2 \cdot x) - \frac{1}{2} \cdot x - \frac{1}{4}$ graphisch dar.

6.3 Programmierübung - Zustandsmodelle

Gegeben ist die Übertragungsfunktion

$$H(s) = \frac{s - 1}{s^3 + \frac{201}{10}s^2 + \frac{10201}{100}s + \frac{10001}{100}}$$

.

a) Bestimmen ein dazugehöriges Zustandsmodell mit der Funktion `scipy.signal.tf2ss`.

b) Berechnen Sie die Eigenwerte der Systemmatrix numerisch.

c) Bestimmen Sie die Impulsantwort des Systems aus dem Zustandsmodell und stellen Sie diese graphisch dar.

d) Bestimmen Sie die Systemantwort für das Eingangssignal $x(t) = \sin(2\pi t)\sigma(t)$ mit der Funktion `scipy.signal.lsim` und stellen Sie diese graphisch dar.

e) Approximieren Sie das System durch ein System 1. Ordnung, so dass die Impulsantwort für Zeiten $t > 10$ etwa der Impulsantwort des ursprünglichen Systems entspricht. Stellen Sie die Impulsantwort graphisch dar.

AUFGABENBLATT 7

Zeitkontinuierliche Signale - Darstellung und Verhalten im Bildbereich

7.1 Programmierübung - Symbolische Berechnungen mit Python

Mit dem Paket `sympy` kann auch symbolisch gerechnet werden. Damit lassen sich z.B. Laplace-Transformationen als geschlossener Ausdruck ermitteln. Gegeben ist die Funktion $y(t) = \exp(-at)$. Berechnen Sie die Laplace-Transformation.

Nutzen Sie die Funktion `sympy.integrals.laplace_transform`. Beim symbolischen Rechnen müssen die symbolischen Variablen festgelegt werden. Das erfolgt mit `s = sympy.symbol('s')`.

Beachten Sie, dass die Exponentialfunktion aus dem Paket `sympy` verwendet werden muss.

7.2 Programmierübung - Analyse eines zeitkontiniuerlichen Systems

Gegeben ist die Übertragungsfunktion[1]

$$H(s) = \frac{1}{s^3 + 2s^2 + 2s + 1}.$$

[1] vgl. Werner: Signale und Systeme - Übungsteil zum Buch, Seite 127,128

a) Bestimmen Sie die Pol- und Nullstellen des Systems. Zeigen Sie die Stabilität anhand des Diagramms.

b) Stellen Sie die Impuls- und Sprungantwort von $H(s)$ graphisch dar.

   i) Verwenden Sie die Funktionen `signal.impulse2()` und `signal.step2()`

   ii) Verwenden Sie den Residuensatz mit Hilfe der Funktion `signal.residue()`

c) Erzeugen Sie ein Bodediagramm mit Amplituden- und Phasengang.

7.3 Programmierübung - Entwurf von Filtern

Entwerfen Sie die folgenden Tiefpassfilter, indem Sie zunächst die Filterordnung bestimmen und anschließend das Filter entwerfen. Stellen Sie den Amplitudengang und das Pol-Nullstellen-Diagramm dar.

- Butterworth (`signal.buttord(), signal.butter()`)
- Tschebyscheff Typ I (`signal.cheb1ord(), signal.cheby1()`)
- Tschebyscheff Typ II (`signal.cheb2ord(), signal.cheby()`)
- Cauer (`signal.ellipord(), signal.ellip()`)

Verwenden Sie das folgende Toleranzschema (siehe Abbildung 7.1):

- Bandpassfrequenz $f_D = 1\,\text{kHz}$
- Cut-off-Frequenz $f_S = 1.4\,\text{kHz}$
- Ripple $R_P = 0.5\,\text{dB}$
- Cut-off-Dämpfung $R_S = 30\,\text{dB}$
- Abtastfrequenz $f_A = 8\,\text{kHz}$

7.4 Programmierübung - Abtastung und Rekonstruktion mit Rechteckfunktion

Ein Sinus $y(t) = \sin(2 \cdot \pi \cdot t)$ mit $t = [-5, 5]$ wird mit einer Abtastfrequenz von $f_a = 8\,\text{Hz}$ abgetastet.

a) Stellen Sie den abgetasteten Sinus graphisch dar.

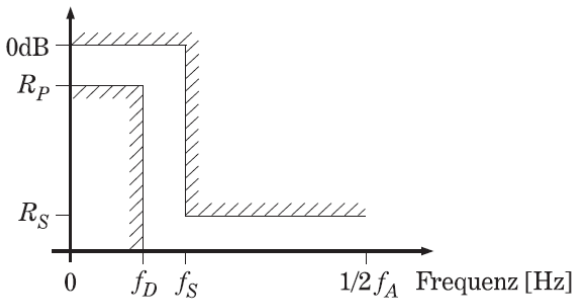

Abbildung 7.1: Toleranzschema

b) Rekonstruieren Sie den Sinus mit Hilfe der Rechteckfunktion. Stellen Sie die Rechteckfunktion und das rekonstruierte Ergebnis graphisch dar.