Praktikum Digitale Signalverarbeitung

Einführung

Prof. Dr.-Ing. Johann-Markus Batke
SS 2022

Inhaltsverzeichnis

1 Organisatorisches

Im Rahmen dieses Praktikums erarbeiten Sie sich grundlegende Kenntnisse im Bereich der digitalen Signalverarbeitung. Beachten Sie dabei bitte folgende Punkte:

Anwesenheit Für dieses Praktikum besteht (wie für jedes andere Praktikum auch) Anwesenheitspflicht. Im Krankeheitsfall muss ein ärztliches Attest vorgelegt werden.
Fehlen Sie *mehr als einmal* unentschuldigt, wird das Praktikum mit „nicht bestanden“ bewertet (in anderen Worten: ein Fehltermin wird akzeptiert).

Abgaben Die Aufgaben werden individuell bearbeitet. Die bearbeiteten Aufgaben werden im Moodlekurs als Jupyter-Notebook abgegeben. Wählen Sie als Dateiname für Versuch n
`pdsn_vorname_nachname.ipynb`
beispielsweise also `pds0_johann-markus_batke.ipynb`

2 Einleitung

Digitale Signalverarbeitung bedeutet die Verarbeitung von Zahlenfolgen, damit bietet sich die Rechner-gestützte Betrachtung vieler Signalverarbeitungsfragen durch Programmierung förmlich an. Die Programmierung soll im Rahmen dieses Praktikums unter **Python** erfolgen. Als Programmierumgebung wird **Jupyter** empfohlen. Für eine einfache mathematische Umsetzung der betrachteten Aufgaben dient dass Pythonmodul **Numpy** [1, bzw. die aktuelle Variante]. Im Versuch geht es darum, sich mit der Arbeitsumgebung Jupyter und er Verwendung des Moduls Numpy vertraut zu machen. Weitere Informationen zu Jupyter und Numpy sind in den

entsprechenden Arbeitsblättern zu finden.

2.1 Programmstart

Jupyter verwaltet Programmtext in Notebooks. Ein Jupyter-Notebook lässt sich über den Launcher des Pythonpakets Anaconda neu generieren. Alternativ ist die Kommandozeilenangabe

```
jupyter notebook
```

möglich.

2.2 Jupyter cell

Im *Notebook* erscheint das Eingabeprompt in einer Zelle (cell). Dort können direkt Pythonbefehle eingegeben und danach ausgeführt werden.

Weiterhin lassen sich Zellen für die Dokumentation der bearbeiteten Aufgabe verwenden. Die Formatierung erfolgt in Markdown-Syntax, auch einfache LaTeX-Konstrukte sind möglich.

2.3 Konstanten und Variablen

Namen von Konstanten und Variablen in Python können Buchstaben und Ziffern und den Unterstrich enthalten; sie beginnen stets mit einem Buchstaben. Groß- und Kleinschreibung wird unterschieden.

Es existiert in Python eine Reihe vordefinierter Variablen mit Werten wichtiger Konstanten. Im Kontext der Signalverarbeitung sind besonders wichtig die Konstanten e, j und π. Die Ausgabe komplexer Zahlen erfolgt kartesisch.

Beispiele:

- Die Euler'sche Zahl kann wie folgt ausgegeben werden:

```
import numpy as np
e = np.exp(1)
print(e)
```

2.718281828459045

- Die Kreiszahl lautet

```
pi = np.pi
print(pi)
```

3.141592653589793

- Die komplexe Einheit hat den Wert

```
i = 1j
print(i)
```

1j

2.4 Workspace

Alle Variablen der Kommandozeileneingabe werden im Arbeitsspeicher, dem `Workspace`, gehalten. Den Inhalt des `Workspace` kann man mit `whos` anzeigen lassen. Eine beispielhafte Ausgabe kann so aussehen:

```
1 whos
```

```
Variable   Type       Data/Info
------------------------------
e          float64    2.718281828459045
i          complex    1j
np         module     <module 'numpy' from '/us<...>kages/numpy/__init__.py'>
pi         float      3.141592653589793
```

Der Inhalt des `Workspace` kann mit dem Befehl `reset` gelöscht werden.

```
In [10]: reset
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

2.5 Hilfefunktionen und Dokumentation

An der Kommandozeile können Hilfetexte mit dem Befehl `help()` abgerufen werden.

```
1 help(print)
```

```
Help on built-in function print in module builtins:

print(*args, sep=' ', end='\n', file=None, flush=False)
    Prints the values to a stream, or to sys.stdout by default.

    sep
      string inserted between values, default a space.
    end
      string appended after the last value, default a newline.
    file
      a file-like object (stream); defaults to the current sys.stdout.
    flush
      whether to forcibly flush the stream.
```

Der Abruf der Dokumentation zu einer Funktion kann auch per vorangestelltem `?` vor dem fraglichen Schlüsselwort erfolgen. Für etwa den Befehl `np.linspace`

```
1 import numpy as np
2 ?np.linspace
```

zeigt sich dann der Text

```
Signature:
np.linspace(
    start,
    stop,
    num=50,
    endpoint=True,
    retstep=False,
    dtype=None,
    axis=0,
)
Docstring:
Return evenly spaced numbers over a specified interval.

Returns `num` evenly spaced samples, calculated over the
interval [`start`, `stop`].

The endpoint of the interval can optionally be excluded.

.. versionchanged:: 1.16.0
    Non-scalar `start` and `stop` are now supported.

Parameters
----------
start : array_like
    The starting value of the sequence.
stop : array_like
    The end value of the sequence, unless `endpoint` is set to False.
    In that case, the sequence consists of all but the last of ``num + 1``
    evenly spaced samples, so that `stop` is excluded.  Note that the step
    size changes when `endpoint` is False.
num : int, optional
    Number of samples to generate. Default is 50. Must be non-negative.
endpoint : bool, optional
    If True, `stop` is the last sample. Otherwise, it is not included.
    Default is True.
retstep : bool, optional
    If True, return (`samples`, `step`), where `step` is the spacing
    between samples.
dtype : dtype, optional
    The type of the output array.  If `dtype` is not given, infer the data
    type from the other input arguments.

    .. versionadded:: 1.9.0

axis : int, optional
```

```
    The axis in the result to store the samples.  Relevant only if start
    or stop are array-like.  By default (0), the samples will be along a
    new axis inserted at the beginning. Use -1 to get an axis at the end.

    .. versionadded:: 1.16.0

Returns
-------
samples : ndarray
    There are `num` equally spaced samples in the closed interval
    ``[start, stop]`` or the half-open interval ``[start, stop)``
    (depending on whether `endpoint` is True or False).
step : float, optional
    Only returned if `retstep` is True

    Size of spacing between samples.


See Also
--------
arange : Similar to `linspace`, but uses a step size (instead of the
         number of samples).
geomspace : Similar to `linspace`, but with numbers spaced evenly on a log
            scale (a geometric progression).
logspace : Similar to `geomspace`, but with the end points specified as
           logarithms.

Examples
--------
>>> np.linspace(2.0, 3.0, num=5)
array([2.  , 2.25, 2.5 , 2.75, 3.  ])
>>> np.linspace(2.0, 3.0, num=5, endpoint=False)
array([2. ,  2.2,  2.4,  2.6,  2.8])
>>> np.linspace(2.0, 3.0, num=5, retstep=True)
(array([2.  ,  2.25,  2.5 ,  2.75,  3.  ]), 0.25)

Graphical illustration:

>>> import matplotlib.pyplot as plt
>>> N = 8
>>> y = np.zeros(N)
>>> x1 = np.linspace(0, 10, N, endpoint=True)
>>> x2 = np.linspace(0, 10, N, endpoint=False)
>>> plt.plot(x1, y, 'o')
[<matplotlib.lines.Line2D object at 0x...>]
```

```
>>> plt.plot(x2, y + 0.5, 'o')
[<matplotlib.lines.Line2D object at 0x...>]
>>> plt.ylim([-0.5, 1])
(-0.5, 1)
>>> plt.show()
File:      /usr/lib/python3/dist-packages/numpy/core/function_base.py
Type:      function
```

Die Anzeige des Quelltextes des Befehls `np.linspace` erfolgt per Voranstellen von `??`:

```
1 ??np.linspace
```

2.6 Vektoren und Matrizen

Vektoren und Matrizen spielen in der Programmierung von Signalverarbeitungsverfahren eine überragende Rolle, da sie direkt zur Darstellung von z.B. Signalabschnitten verwendet werden können. Das Modul `numpy` unterstützt aus diesem Grund eine Reihe von einfachen Eingabemöglichkeiten für Vektoren und Matrizen (die Benutzung ist ähnlich zu Matlab/Octave).

3 Aufgaben

Die nachfolgenden Aufgaben sind durch Pythonprogramme zu lösen. Die Abgabe der Lösung soll als Jupyter-Notebook erfolgen. Dokumentieren Sie Ihre Lösung durch markdown-formatierte Zellen.

a) Welches Ergebnis hat dieser Ausdruck?

```
1   print(1*2**3+4)
```

b) Prüfen Sie den Wert des Ausdrucks $\mathrm{e}^{\mathrm{i}\pi/2}$ in Python!

c) Weisen Sie der Variable z die komplexe Zahl $1 + \mathrm{i}2$ zu!

d) Erzeugen Sie die Matrix

$$\begin{matrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{matrix}$$

unter Verwendung eines Spalten-Vektors mit den Werten `[1 2 3]` und mithilfe des Befehls `np.ones`!

Literatur

[1] NumPy Community, Hrsg. *NumPy Reference. Release 1.15.4*. 2018 (siehe S. 1).