Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfungsvorbereitungskurs Lineare Algebra

Python Crashkurs

Janick Matter

HS23

1 Vorwort

In Diesem Dokument findet Ihr die wichtigsten Python Befehle, welche im Skript von Professor Auer (HS23) erwähnt wurden in kurzen Code Snippets, sowie einige grössere Beispiele mit Lücken, zum selber Lösen. Die Lösungen zu den Beispielen sind ebenfalls in den entsprechenden Kapiteln. Ich kann allerdings weder für Vollständigkeit noch für Richtigkeit garantieren, und möchte euch ermuntern die Liste mit eigenen Code Beispielen zu erweitern.
Für die Code Snippets und Beispiele habe ich die Umgebung Visual Studio Code Expert (V. 1.85.1) verwendet, mit Python 3.11.2.

Für gespottete Typos oder Anregungen bin ich immer sehr dankbar. Ich wünsche euch alles Gute für eure Prüfungen **;)**
jamatter@student.ethz.ch

2 Überarbeitungen

- 30.12.2023: Initial version
- -

Contents

3 Python Code: Example Snippets

3.0.1 `np.linalg.solve(A, b)`

Löst die lineare Matrixgleichung $Ax = b$ für x, wobei A in diesem Beispiel eine quadratische Matrix und b ein Spaltenvektor ist.

```
import numpy as np

# Beispielmatrizen
A = np.array([[2, 1], [1, 3]])
b = np.array([4, 7])

# Loese das lineare System Ax = b mit np.linalg.solve()
x = np.linalg.solve(A, b)

print("Loesung x:", x)
```

Listing 1: Beispiel mit `np.linalg.solve(A, b)`

3.0.2 `np.linalg.cond()`

Berechnet die Konditionszahl einer Matrix, die die Empfindlichkeit der Lösung gegenüber Ungenauigkeiten in der Matrix misst.

```
import numpy as np

# Beispielmatrix
A = np.array([[2, 1], [1, 3]])

# Berechne die Konditionszahl der Matrix A mit np.linalg.
    cond()
condition_number = np.linalg.cond(A)

print("Konditionszahl von A:", condition_number)
```

Listing 2: Beispiel mit `np.linalg.cond()`

3.0.3 `np.array([])`

Erstellt ein NumPy-Array mit den angegebenen Elementen.

```
import numpy as np

# Erstelle ein 1D-Array mit np.array()
arr_1d = np.array([1, 2, 3])

print("1D-Array:", arr_1d)
```

Listing 3: Beispiel mit `np.array([])`

3.0.4 `np.array([], [])`

Erstellt ein 2D-NumPy-Array mit den angegebenen Elementen.

```
import numpy as np

# Erstelle ein 2D-Array mit np.array()
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])

print("2D-Array:")
print(arr_2d)
```

Listing 4: Beispiel mit `np.array([], [])`

3.0.5 `np.dot()`

Berechnet das Skalarprodukt zweier Arrays (Spaltenvektoren).

```
import numpy as np

# Beispielarrays
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# Berechne das Skalarprodukt mit np.dot()
dot_product = np.dot(arr1, arr2)

print("Skalarprodukt:", dot_product)
```

Listing 5: Beispiel mit `np.dot()`

3.0.6 `np.linalg.norm()`

Berechnet die euklidische Norm (Betrag) eines Vektors oder die Frobenius-Norm einer Matrix.

```
import numpy as np

# Beispielvektor
vec = np.array([3, 4])

# Berechne die euklidische Norm mit np.linalg.norm()
norm_result = np.linalg.norm(vec)

print("Euklidische Norm:", norm_result)
```

Listing 6: Beispiel mit `np.linalg.norm()`

3.0.7 `np.cross()`

Berechnet das Kreuzprodukt zweier Vektoren.

```
import numpy as np

# Beispielvektoren
vec1 = np.array([1, 0, 0])
vec2 = np.array([0, 1, 0])

# Berechne das Kreuzprodukt mit np.cross()
cross_product = np.cross(vec1, vec2)

print("Kreuzprodukt:", cross_product)
```

Listing 7: Beispiel mit `np.cross()`

3.0.8 `np.zeros()`

Erstellt ein mit Nullen gefülltes Array (Matrix).

```
import numpy as np

# Erstelle ein 2x3-Array mit Nullen mit np.zeros()
zeros_array = np.zeros((2, 3))

print("Nullen-Array:")
print(zeros_array)
```

Listing 8: Beispiel mit `np.zeros()`

3.0.9 `np.eye()`

Erstellt eine Einheitsmatrix.

```
import numpy as np

# Erstelle eine 3x3 Einheitsmatrix mit np.eye()
identity_matrix = np.eye(3)

print("Einheitsmatrix:")
print(identity_matrix)
```

Listing 9: Beispiel mit `np.eye()`

3.0.10 `np.linalg.inv(A)`

Berechnet die Inverse einer quadratischen Matrix A.

```
import numpy as np
```

```
# Beispielmatrix
A = np.array([[2, 1], [1, 3]])

# Berechne die Inverse von A mit np.linalg.inv()
inverse_matrix = np.linalg.inv(A)

print("Inverse von A:")
print(inverse_matrix)
```

Listing 10: Beispiel mit `np.linalg.inv()`

3.0.11 `np.diag()`

Extrahiert die Diagonalelemente einer Matrix oder konstruiert eine diagonale Matrix mit den gegebenen Werten.

```
import numpy as np

# Beispielmatrix
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Extrahiere die Diagonalelemente mit np.diag()
diagonal_elements = np.diag(matrix)

print("Diagonalelemente:")
print(diagonal_elements)
```

Listing 11: Beispiel mit `np.diag()`

3.0.12 `np.trace()`

Berechnet die Spur (Summe der Diagonalelemente) einer Matrix.

```
import numpy as np

# Beispielmatrix
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Berechne die Spur mit np.trace()
trace_result = np.trace(matrix)

print("Spur der Matrix:", trace_result)
```

Listing 12: Beispiel mit `np.trace()`

3.0.13 `np.linalg.det(A)`

Berechnet die Determinante einer quadratischen Matrix A.

```python
import numpy as np

# Beispielmatrix
A = np.array([[2, 1], [1, 3]])

# Berechne die Determinante von A mit np.linalg.det()
determinant_result = np.linalg.det(A)

print("Determinante von A:", determinant_result)
```

Listing 13: Beispiel mit `np.linalg.det()`

3.0.14 `np.poly(A)`

Berechnet die Koeffizienten des charakteristischen Polynoms einer Matrix A.

```python
import numpy as np

# Beispielmatrix
A = np.array([[2, 1], [1, 3]])

# Berechne die Koeffizienten des charakteristischen Polynoms
    mit np.poly()
poly_coefficients = np.poly(A)

print("Koeffizienten des charakteristischen Polynoms:",
   poly_coefficients)
```

Listing 14: Beispiel mit `np.poly()`

3.0.15 `EW, EV = np.eig()`

Berechnet die Eigenwerte (`EW`) und Eigenvektoren (`EV`) einer Matrix.

```python
import numpy as np

# Beispielmatrix
A = np.array([[2, 1], [1, 3]])

# Berechne Eigenwerte und Eigenvektoren mit np.eig()
eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenwerte:")
print(eigenvalues)
print("Eigenvektoren:")
print(eigenvectors)
```

Listing 15: Beispiel mit `np.eig()`

3.0.16 `sc.linalg.expm()`

Berechnet die Matrix-Exponentialfunktion für eine quadratische Matrix.

```
import numpy as np
import scipy.linalg as sc

# Beispielmatrix
A = np.array([[1, 2], [3, 4]])

# Berechne die Matrix-Exponentialfunktion mit sc.linalg.expm
    ()
matrix_exponential = sc.expm(A)

print("Matrix-Exponentialfunktion:")
print(matrix_exponential)
```

Listing 16: Beispiel mit `sc.linalg.expm()`

3.0.17 `U, S, V = sc.linalg.svd(A)`

Berechnet die Singulärwertzerlegung (SVD) einer Matrix A, wobei U die orthogonale Matrix der linken Singulärvektoren, S eine Diagonalmatrix mit den Singulärwerten und V die orthogonale Matrix der rechten Singulärvektoren ist.

```
import numpy as np
import scipy.linalg as sc

# Beispielmatrix
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Berechne die Singulaerwertzerlegung mit sc.linalg.svd()
U, S, V = sc.svd(A)

print("Matrix U der linken Singulaervektoren:")
print(U)
print("Diagonalmatrix S der Singulaerwerte:")
print(np.diag(S))
print("Matrix V der rechten Singulaervektoren:")
print(V)
```

Listing 17: Beispiel mit `sc.linalg.svd()`

Die Singulärwertzerlegung (SVD) von A besteht aus den Matrizen U, S und V, wobei U die linken Singulärvektoren enthält, S eine Diagonalmatrix mit den Singulärwerten ist, und V die rechten Singulärvektoren enthält. In diesem Beispiel können wir die orthogonale Matrix U, die Diagonalmatrix S und die orthogonale Matrix V ausgeben und analysieren.

3.0.18 `np.linalg.norm()`

Berechnet die euklidische Norm (Betrag) eines Vektors oder die Frobenius-Norm einer Matrix.

```
import numpy as np

# Beispielvektor und Matrix
vec = np.array([1, 2, 3])
matrix = np.array([[1, 2], [3, 4]])

# Berechne die euklidische Norm des Vektors und die
    Frobenius-Norm der Matrix
norm_vector = np.linalg.norm(vec)
norm_matrix = np.linalg.norm(matrix, 'fro')

print("Euklidische Norm des Vektors:", norm_vector)
print("Frobenius-Norm der Matrix:", norm_matrix)
```

Listing 18: Beispiel mit `np.linalg.norm()`

Die euklidische Norm eines Vektors v ist definiert als $\|v\| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$, wobei v_i die Komponenten des Vektors sind. Die Frobenius-Norm einer Matrix A ist definiert als $\|A\|_{\text{Fro}} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$, wobei a_{ij} die Elemente der Matrix sind. In diesem Beispiel berechnen wir die euklidische Norm eines Vektors und die Frobenius-Norm einer Matrix.

3.0.19 `np.linalg.norm(A, 'fro')`

Berechnet die Frobenius-Norm einer Matrix A, die die euklidische Norm der abgeflachten Matrix repräsentiert.

```
import numpy as np

# Beispielmatrix
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Berechne die Frobenius-Norm der Matrix A mit np.linalg.
    norm()
frobenius_norm = np.linalg.norm(A, 'fro')

print("Frobenius-Norm der Matrix A:", frobenius_norm)
```

Listing 19: Beispiel mit `np.linalg.norm()` und `'fro'`

Die Frobenius-Norm einer Matrix A kann auch mit `np.linalg.norm(A, 'fro')` berechnet werden. Diese Norm ist äquivalent zur euklidischen Norm der abgeflachten Matrix A. In diesem Beispiel berechnen wir die Frobenius-Norm einer Matrix.

4 Python Code: Beispiele

In diesem Kapitel fidnet Ihr einige Beispiele mit den Lösungen, die Verschiedene Gebiete des Stoffs behandeln.

4.1 NumPy-Grundlagen (Mit Lücken)

Gegeben sei die Matrix A:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

1. **Erstellen von NumPy-Arrays:** a. Erstelle ein 1D-NumPy-Array arr_1d mit den Werten 1, 2, 3. b. Erstelle ein 2D-NumPy-Array arr_2d mit den Werten:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

2. **Dot-Produkt:** a. Berechne das Dot-Produkt von arr_1d und arr_1d. b. Berechne das Dot-Produkt von arr_2d und seiner Transponierten.

3. **Normen:** a. Berechne die euklidische Norm des Vektors $v = [1, 2, 3]$. b. Berechne die Frobenius-Norm der Matrix A.

4. **Vektoroperationen:** a. Berechne das Kreuzprodukt von $v1 = [1, 0, 0]$ und $v2 = [0, 1, 0]$.

5. **Erstellen von Null-Matrizen und Einheitsmatrizen:** a. Erstelle eine 2x3-Matrix zeros_matrix gefüllt mit Nullen. b. Erstelle eine 3x3-Einheitsmatrix identity_matrix.

Ergänze den Python-Code, um die geforderten Berechnungen durchzuführen.

```python
import numpy as np

# Gegebene Matrix A
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# 1. Erstellen von NumPy-Arrays
# a. Erstelle ein 1D-NumPy-Array arr_1d
arr_1d = # Dein Code hier

# b. Erstelle ein 2D-NumPy-Array arr_2d
arr_2d = # Dein Code hier

print("1D-Array:")
print(arr_1d)
print("\n2D-Array:")
print(arr_2d)
print("\n")
```

```
% 2. Dot-Produkt
% a. Berechne das Dot-Produkt von arr_1d und arr_1d
dot_product_1d = # Dein Code hier

% b. Berechne das Dot-Produkt von arr_2d und seiner
    Transponierten
dot_product_2d = # Dein Code hier

print("Dot-Produkt von arr_1d:", dot_product_1d)
print("Dot-Produkt von arr_2d und Transponierter:",
    dot_product_2d)
print("\n")

% 3. Normen
% a. Berechne die euklidische Norm des Vektors v = [1, 2, 3]
norm_vector = # Dein Code hier

% b. Berechne die Frobenius-Norm der Matrix A
norm_matrix = # Dein Code hier

print("Euklidische Norm des Vektors v:", norm_vector)
print("Frobenius-Norm der Matrix A:", norm_matrix)
print("\n")

% 4. Vektoroperationen
% a. Berechne das Kreuzprodukt von v1 = [1, 0, 0] und v2 =
    [0, 1, 0]
cross_product = # Dein Code hier

print("Kreuzprodukt von v1 und v2:", cross_product)
print("\n")

% 5. Erstellen von Null-Matrizen und Einheitsmatrizen
% a. Erstelle eine 2x3-Matrix zeros_matrix gefuellt mit
    Nullen
zeros_matrix = # Dein Code hier

% b. Erstelle eine 3x3-Einheitsmatrix identity_matrix
identity_matrix = # Dein Code hier

print("Nullen-Matrix:")
print(zeros_matrix)
print("\nEinheits-Matrix:")
print(identity_matrix)
print("\n")
```

Listing 20: Python-Code MIT Lücken

4.1.1 NumPy-Grundlagen (Lösung)

Lösung des Beispiels 4.1

```
import numpy as np

# Gegebene Matrix A
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# 1. Erstellen von NumPy-Arrays
# a. Erstelle ein 1D-NumPy-Array arr_1d
arr_1d = np.array([1, 2, 3])

# b. Erstelle ein 2D-NumPy-Array arr_2d
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])

print("1D-Array:")
print(arr_1d)
print("\n2D-Array:")
print(arr_2d)
print("\n")

% 2. Dot-Produkt
# a. Berechne das Dot-Produkt von arr_1d und arr_1d
dot_product_1d = np.dot(arr_1d, arr_1d)

# b. Berechne das Dot-Produkt von arr_2d und seiner
    Transponierten
dot_product_2d = np.dot(arr_2d, arr_2d.T)

print("Dot-Produkt von arr_1d:", dot_product_1d)
print("Dot-Produkt von arr_2d und Transponierter:",
    dot_product_2d)
print("\n")

% 3. Normen
# a. Berechne die euklidische Norm des Vektors v = [1, 2, 3]
norm_vector = np.linalg.norm([1, 2, 3])

# b. Berechne die Frobenius-Norm der Matrix A
norm_matrix = np.linalg.norm(A, 'fro')

print("Euklidische Norm des Vektors v:", norm_vector)
print("Frobenius-Norm der Matrix A:", norm_matrix)
print("\n")

% 4. Vektoroperationen
# a. Berechne das Kreuzprodukt von v1 = [1, 0, 0] und v2 =
    [0, 1, 0]
cross_product = np.cross([1, 0, 0], [0, 1, 0])
```

```
print("Kreuzprodukt von v1 und v2:", cross_product)
print("\n")

% 5. Erstellen von Null-Matrizen und Einheitsmatrizen
# a. Erstelle eine 2x3-Matrix zeros_matrix gefuellt mit
    Nullen
zeros_matrix = np.zeros((2, 3))

# b. Erstelle eine 3x3-Einheitsmatrix identity_matrix
identity_matrix = np.eye(3)


print("Nullen-Matrix:")
print(zeros_matrix)
print("\nEinheits-Matrix:")
print(identity_matrix)
print("\n")
```

Listing 21: Python-Code OHNE Lücken

4.2 SVD und Normen (Mit Lücken)

Gegeben sei die Matrix A:

$$A = \begin{bmatrix} 2 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

1. **Singulärwertzerlegung (SVD):** Berechne die Singulärwertzerlegung von A und gib die Matrizen U, S und V aus.
2. **Normberechnungen:** a. Berechne die euklidische Norm des Vektors $v = [1, 2, 3]$. b. Berechne die Frobenius-Norm der Matrix A.
3. **Frobenius-Norm mit `np.linalg.norm()`:** Verwende `np.linalg.norm(A, 'fro')` und berechne die Frobenius-Norm von A.

Ergänze den Python-Code und fülle die Lücken entsprechend aus.

```
import numpy as np
import scipy.linalg as sc

# Gegebene Matrix A
A = np.array([[2, 1, 3], [4, 5, 6], [7, 8, 9]])

# 1. Singulaerwertzerlegung (SVD)
# Ergaenze die fehlenden Befehle, um U, S und V zu berechnen
# _____ (Dein Code)

# Ausgabe der Singulaerwertzerlegung
print("Matrix U der linken Singulaervektoren:")
print(U)
print("Diagonalmatrix S der Singulaerwerte:")
print(np.diag(S))
print("Matrix V der rechten Singulaervektoren:")
print(V)
print("\n")

# 2. Normberechnungen
# a. Euklidische Norm des Vektors v = [1, 2, 3]
# Ergaenze den Befehl fuer die Euklidische Norm
# _____ (Dein Code)

print("Euklidische Norm des Vektors v:", norm_vector)
print("\n")

# b. Frobenius-Norm der Matrix A
# Ergaenze den Befehl fuer die Frobenius-Norm
# _____ (Dein Code)
```

```
print("Frobenius-Norm der Matrix A:", norm_matrix)
print("\n")
```

Listing 22: Python-Code MIT Lücken

4.2.1 SVD und Normen (Lösung)

Gefüllte Lücken des Beispiels 4.2

```
import numpy as np
import scipy.linalg as sc

# Gegebene Matrix A
A = np.array([[2, 1, 3], [4, 5, 6], [7, 8, 9]])

# 1. Singulaerwertzerlegung (SVD)
# Ergaenze die fehlenden Befehle, um U, S und V zu berechnen
U, S, V = sc.svd(A)

# Ausgabe der Singulaerwertzerlegung
print("Matrix U der linken Singulaervektoren:")
print(U)
print("Diagonalmatrix S der Singulaerwerte:")
print(np.diag(S))
print("Matrix V der rechten Singulaervektoren:")
print(V)
print("\n")

# 2. Normberechnungen
# a. Euklidische Norm des Vektors v = [1, 2, 3]
# Ergaenze den Befehl fuer die Euklidische Norm
norm_vector = np.linalg.norm([1, 2, 3])

print("Euklidische Norm des Vektors v:", norm_vector)
print("\n")

# b. Frobenius-Norm der Matrix A
# Ergaenze den Befehl fuer die Frobenius-Norm
norm_matrix = np.linalg.norm(A, 'fro')

print("Frobenius-Norm der Matrix A:", norm_matrix)
print("\n")
```

Listing 23: Python-Code OHNE Lücken