FORMELSAMMLUNG LINEARE ALGEBRA

Basierend auf Übungen FH Vorarlberg und dem LA-Kurs von Klaus Rheinberger
[pages.labs.fhv.at/~kr/lehre/la/]

1. VEKTOREN UND MATRIZEN

- Vektoroperationen:
  - Addition: $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_1+b_1 \\ a_2+b_2 \end{bmatrix}$
  - Skalarprodukt: $a \cdot b = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$
  - Kreuzprod. ($\mathbb{R}^3$): $a \times b = [a_2 b_3 - a_3 b_2,\ a_3 b_1 - a_1 b_3,\ a_1 b_2 - a_2 b_1]$

- Matrizen:
  - Multiplikation: $(AB)_{ij} = \Sigma\, A_{ik} \cdot B_{kj}$
  - Transponierte: $(A^T)_{ij} = A_{ji}$
  - Spur: $tr(A) = \Sigma\, A_{ii}$ (Diagonalsumme)

2. LINEARE GLEICHUNGSSYSTEME (LGS)

- Lösbarkeit:

$Rang(A) = Rang(A\|b) = n$	Eindeutige Lösung
$Rang(A) = Rang(A\|b) < n$	Unendlich viele Lösungen
$Rang(A) < Rang(A\|b)$	Keine Lösung

- Gauß-Elimination:
  - Ziel: Zeilenstufenform (REF)
  - Pivot-Elemente: Erste Nicht-Null-Elemente pro Zeile

3. DETERMINANTE & INVERSE

- 2×2-Matrix:
  $\det(A) = ad - bc$ $\quad A^{-1} = 1/\det(A) \cdot \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

- 3×3-Matrix (Sarrus):
  $\det(A) = aei + bfg + cdh - ceg - bdi - afh$

4. EIGENWERTE & EIGENVEKTOREN

- Char. Gleichung: $\det(A - \lambda I) = 0$
- Eigenvektoren: $(A - \lambda I)v = 0$
- Diagonalisierung: $A = PDP^{-1}$ (D = Diagonalmatrix)

5. ORTHOGONALITÄT & PROJEKTION

- Projektion von b auf a:
  $proj_a b = (a \cdot b \,/\, a \cdot a) \cdot a$

- Orthogonale Matrix:
  $Q^T Q = I$ (Spalten orthonormal)

6. LINEARE REGRESSION

- Methode der kleinsten Quadrate:
  $x = (A^T A)^{-1} A^T b$

- Exponentieller Fit:
  $y = ce^{at} \rightarrow \ln(y) = \ln(c) + at$

7. PYTHON-BEFEHLE

`np.linalg.solve(A,b)`	Löst Ax = b
`np.linalg.inv(A)`	Inverse von A
`np.linalg.eig(A)`	Eigenwerte/-vektoren
`np.polyfit(x,y,deg)`	Polynomfit (Grad deg)

8. ANWENDUNGEN (AUS ÜBUNGEN)

- Ill-Conditioned Systems:
  Kleine Änderungen in b → große Änderungen in x

- Adjazenzmatrizen:
  $C_{ij} = 1$ (wenn Kante i→j existiert)

- Leontief-Modell:
  $p = (I - A)^{-1} d$ (Produktionsplanung)

NUMPY BEFEHLE FÜR LINEARE ALGEBRA

1. MATRIXOPERATIONEN

```
np.array([[1,2],[3,4]])          # Matrix erstellen
A.T                              # Transponierte
np.linalg.inv(A)                 # Inverse (für quadratische Matrizen)
np.linalg.det(A)                 # Determinante
np.trace(A)                      # Spur (Summe der Diagonalelemente)
np.eye(n)                        # Einheitsmatrix (n×n)
np.diag([a,b,c])                 # Diagonalmatrix erstellen

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
2. LINEARE GLEICHUNGSSYSTEME
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

np.linalg.solve(A, b)            # Löse Ax = b (für quadratische A)
np.linalg.lstsq(A, b, rcond=None)# Least-Squares-Lösung (für nicht-quadratische
A)
np.linalg.matrix_rank(A)         # Rang der Matrix
np.linalg.pinv(A)                # Pseudoinverse (Moore-Penrose-Inverse)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
3. EIGENWERTE & ZERLEGUNGEN
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

np.linalg.eig(A)                 # Eigenwerte und -vektoren
np.linalg.eigvals(A)             # Nur Eigenwerte
np.linalg.qr(A)                  # QR-Zerlegung
np.linalg.svd(A)                 # Singulärwertzerlegung (SVD)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
4. VEKTOROPERATIONEN
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

np.dot(a, b)                     # Skalarprodukt (a·b)
np.cross(a, b)                   # Kreuzprodukt (nur ℝ³)
np.linalg.norm(a)                # Norm (Länge) des Vektors
np.angle(v)                      # Winkel komplexer Vektoren
np.outer(a, b)                   # Äußeres Produkt (a⊗b)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
5. SPEZIELLE MATRIZEN
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

np.zeros((m,n))                  # Nullmatrix
np.ones((m,n))                   # Einsen-Matrix
np.random.rand(m,n)              # Zufallsmatrix (Gleichverteilung)
np.random.randn(m,n)             # Zufallsmatrix (Normalverteilung)
np.triu(A)                       # Oberes Dreieck
np.tril(A)                       # Unteres Dreieck

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
6. PRAKTISCHE ANWENDUNGEN (AUS ÜBUNGEN)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

# Adjazenzmatrix-Analyse (Übung 4.5)
C @ C                            # Matrixpotenz (C² für 1 Umstieg)
np.linalg.matrix_power(C, k)     # Allgemeine Matrixpotenz

# Dynamische Systeme (Übung 7.1)
np.linalg.matrix_power(M, 50)    # Langzeitverhalten nach 50 Schritten

# Lineare Regression (Übung 8.1)
np.polyfit(t, np.log(y), 1)      # Exponentieller Fit (y = ce^{αt})
np.polyval(coeffs, t_new)        # Auswertung des Polynoms

================================================================================
```