Uniwersytet Bielsko-Bialski

Rozkład w trybie dynamicznym

Matematyka Konkretna

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Autor:

Bartosz Bieniek 058085

1. Cel ćwiczenia

Celem jest poznać metodę DMD Snapshot-Sequence, czyli algorytm wyznaczania liniowego modelu x(k+1) ≈ A x(k) wyłącznie z danych pomiarowych. Obliczyć macierz transformacji A dla zadanych macierzy X i X′ oraz sprawdzić dokładność aproksymacji X′ ≈ A X. Przećwiczyć narzędzia: SVD, pseudoodwrotność, redukcja rzędu, Python NumPy Pandas Jupyter. Zrozumieć interpretację spektrogramu, czyli wartości własnych Λ i trybów Φ, oraz ich związek z dynamiką układu.
Zadania umieścić na Github.

2. Przebieg ćwiczenia

1. Import bibliotek
Zaimportowano niezbędne biblioteki do przetwarzania i wizualizacji danych. Wczytano moduły numpy oraz pandas do operacji numerycznych i manipulacji danymi tabelarycznymi, a także matplotlib.pyplot do tworzenia wykresów.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Rys. 1. Import wymaganych bibliotek

2. Wczytywanie danych wejściowych
Wczytano dane wejściowe z dwóch plików CSV: War1_X.csv oraz War1_Xprime.csv, stosując separator ; i brak nagłówków. Zastosowano funkcję to_float, która konwertowała liczby zapisane z przecinkiem dziesiętnym na format zmiennoprzecinkowy. Następnie przekształcono dane do postaci macierzy NumPy, odrzucając odpowiednio pierwszą kolumnę w X_full oraz ostatnią kolumnę w Xprime_full, aby uzyskać końcowe macierze X i Xprime reprezentujące odpowiednio obserwacje i ich przekształcenia.

```
def to_float(x):
    return float(str(x).replace(',', '.'))
X_full      = pd.read_csv('War1_X.csv',      sep=';', header=None).applymap(to_float)
Xprime_full = pd.read_csv('War1_Xprime.csv', sep=';', header=None).applymap(to_float)

X      = X_full.iloc[:, 1:].to_numpy()
Xprime = Xprime_full.iloc[:, :-1].to_numpy()
```

Rys. 2. Wczytywanie danych wejściowych

3. Implementacja funkcji DMD

Zaimplementowano funkcję dmd_ss, realizującą algorytm Dynamic Mode Decomposition w trybie Snapshot Sequence (DMD-SS) z możliwością redukcji rangi do wartości r. Na wejściu przyjęto macierze X oraz Xprime, zawierające odpowiednio kolejne migawki układu (od stanu 1 do m-1 oraz od 2 do m). Przeprowadzono dekompozycję SVD macierzy X, a następnie zredukowano jej składowe do rzędu r. Na ich podstawie skonstruowano zredukowaną macierz dynamiki A_tilde i obliczono jej wartości własne oraz wektory własne. Wyznaczono macierz trybów DMD Phi, odpowiadającą strukturze dynamicznej układu. Obliczono również wektor amplitud początkowych b na podstawie pierwszego snapshotu. Wynikiem działania funkcji były: macierz trybów Phi, macierz przekątniowa wartości własnych Lambda oraz wektor amplitud b.

```python
def dmd_ss(X, Xprime, r):
    """
    Dynamic Mode Decomposition – Snapshot Sequence (truncation rank = r)
    X      : n × m-1   (snapshots 1 .. m-1)
    Xprime : n × m-1   (snapshots 2 .. m)
    returns Φ (n×r), Λ (r diag), b (r×1)
    """
    U, s, Vt = np.linalg.svd(X, full_matrices=False)   # SVD
    Ur   = U[:, :r]
    Sr   = np.diag(s[:r])
    Vtr  = Vt[:r, :]

    A_tilde = Ur.T @ Xprime @ Vtr.T @ np.linalg.inv(Sr)
    eigvals, W = np.linalg.eig(A_tilde)
    Lambda = np.diag(eigvals)
    Phi = Xprime @ Vtr.T @ np.linalg.inv(Sr) @ W       # tryby DMD

    # amplitudy początkowe (dla 1-go snapshotu)
    alpha1 = Sr @ Vtr[:, 0]
    b = np.linalg.solve(W @ Lambda, alpha1)

    return Phi, Lambda, b
```

Rys. 3. Wczytywanie danych wejściowych

4. Wizualizacja danych
Ustalono wartość rangi obcięcia na r=5 i wywołano funkcję dmd_ss w celu wyznaczenia trybów DMD (Phi), wartości własnych (Lambda) oraz amplitud początkowych (b). Wypisano zaokrąglone do sześciu miejsc dziesiętnych wartości własne oraz amplitudy, co umożliwiło ocenę dynamiki układu.

Wygenerowano wykres rozrzutu przedstawiający widmo DMD, obrazujące rozkład wartości własnych w płaszczyźnie zespolonej. Pozwoliło to na wizualną analizę stabilności i oscylacyjności trybów dynamicznych.

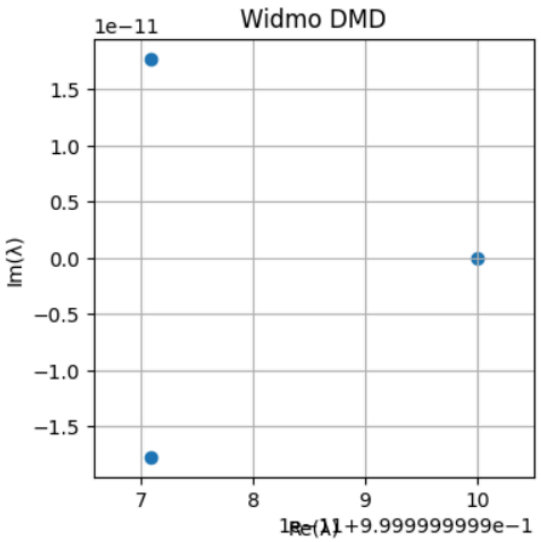
Na koniec obliczono i wyświetlono pełną macierz operatora ewolucji A, jako pseudoodwrotność iloczynu A=X′X+A=X′X+, a następnie przedstawiono ją w postaci tabelarycznej z zaokrągleniem do dwóch miejsc dziesiętnych.

```
r = 5
Phi, Lambda, b = dmd_ss(X, Xprime, r)

print("Wartości własne (λ):")
print(np.round(np.diag(Lambda), 6))

print("\nAmplitudy początkowe (b):")
print(np.round(b, 6))

plt.figure(figsize=(4,4))
plt.scatter(np.real(np.diag(Lambda)), np.imag(np.diag(Lambda)))
plt.xlabel('Re(λ)'); plt.ylabel('Im(λ)')
plt.title('Widmo DMD'); plt.grid(True); plt.axis('equal')
plt.show()

A = Xprime @ np.linalg.pinv(X)
A_df = pd.DataFrame(A).round(2)
print(A_df)
```

Rys. 4. Wczytywanie danych wejściowych

```
Wartości własne (λ):
[1.+0.j 1.+0.j 1.-0.j]

Amplitudy początkowe (b):
[-3.73278720e+24-5.36870912e+08j  1.42048989e+24-2.09388303e+24j
  1.42048989e+24+2.09388303e+24j]
```

Rys. 5. Wartości własne i amplitudy początkowe



Rys. 6. Wykres widma DMD

	0	1	2	3	4	5	6	7	8	9	...	13	\
0	0.50	0.16	0.13	-0.12	-0.08	0.06	0.37	-0.03	0.02	0.09	...	0.05	
1	0.16	0.51	0.05	-0.01	0.10	-0.05	0.09	0.11	0.01	-0.15	...	-0.24	
2	0.13	0.05	0.43	0.03	0.14	0.02	-0.05	0.10	0.09	0.13	...	-0.00	
3	-0.12	-0.01	0.03	0.17	0.04	-0.06	-0.04	0.07	0.07	0.04	...	-0.01	
4	-0.08	0.10	0.14	0.04	0.28	0.04	-0.12	0.10	0.10	0.04	...	0.02	
5	0.06	-0.05	0.02	-0.06	0.04	0.36	0.10	0.02	0.11	-0.02	...	0.04	
6	0.37	0.09	-0.05	-0.04	-0.12	0.10	0.41	-0.01	0.08	0.07	...	0.06	
7	-0.03	0.11	0.10	0.07	0.10	0.02	-0.01	0.41	0.15	0.12	...	-0.05	
8	0.02	0.01	0.09	0.07	0.10	0.11	0.08	0.15	0.39	-0.08	...	-0.03	
9	0.09	-0.15	0.13	0.04	0.04	-0.02	0.07	0.12	-0.08	0.39	...	0.21	
10	0.06	-0.03	-0.04	0.13	-0.17	-0.19	0.09	-0.11	-0.08	-0.04	...	0.09	
11	-0.06	-0.14	0.13	0.11	0.00	0.18	0.00	-0.13	0.20	-0.14	...	0.01	
12	0.00	-0.03	0.08	-0.02	-0.00	0.13	-0.03	0.10	0.02	-0.03	...	0.06	
13	0.05	-0.24	-0.00	-0.01	0.02	0.04	0.06	-0.05	-0.03	0.21	...	0.29	
14	-0.03	0.10	-0.14	0.09	0.13	-0.07	0.07	-0.07	-0.01	0.05	...	0.07	
15	0.03	0.14	0.21	0.08	0.03	0.03	-0.02	-0.07	-0.15	0.05	...	-0.09	
16	-0.11	0.05	-0.09	0.19	0.04	0.06	0.09	0.01	0.03	0.07	...	-0.03	
17	-0.05	0.11	-0.24	-0.00	-0.02	0.19	0.08	0.15	0.02	-0.13	...	0.00	
18	-0.01	0.17	0.09	-0.00	0.14	-0.12	-0.14	0.01	-0.08	-0.08	...	0.03	
19	0.04	-0.07	-0.01	-0.04	0.11	0.06	0.02	-0.05	-0.01	0.08	...	0.20	
20	0.03	0.02	0.02	0.00	0.21	0.01	0.01	-0.11	0.14	-0.04	...	0.11	
21	-0.04	-0.05	0.05	0.16	-0.01	-0.15	0.02	0.28	0.13	0.16	...	0.05	
22	-0.05	-0.03	0.02	-0.02	0.05	0.15	-0.05	0.04	-0.22	0.17	...	0.10	

...									
21	-0.01	-0.10	0.05	0.02	-0.00	-0.05	-0.08	0.39	-0.08
22	0.00	0.18	0.09	0.15	0.05	0.09	-0.11	-0.08	0.37

[23 rows x 23 columns]

Rys. 7. Macierz A

3. Wnioski

Charakterystyka wartości własnych: Wszystkie obliczone wartości własne (λ) miały moduł równy 1 i były zlokalizowane na jednostkowym okręgu w płaszczyźnie zespolonej. Taki rozkład wskazywał na obecność trybów o charakterze oscylacyjno-stałym, bez tendencji do wzrostu ani zaniku amplitudy w czasie, co świadczyło o marginalnej stabilności układu dynamicznego.

Symetria spektralna: Obecność sprzężonych zespolonych wartości własnych sugerowała istnienie par trybów reprezentujących oscylacje o przeciwnych kierunkach rotacji w przestrzeni fazowej.