lab3_jarzecki

June 1, 2025

Analiza współczynnika uwarunkowania macierzy
Łukasz Jarzęcki 331697

[345]:
```
import numpy as np
from scipy.io import loadmat
```

1. **Funkcje pomocnicze**

[346]:
```
def load_A_b_from_mat_file(filename):
    data = loadmat(filename)  # ścieżka do Twojego pliku .mat
    A = data['A']
    b = data['b']
    return A,b.flatten()
```

[347]:
```
def is_symmetric(A, tol=1e-8):
    return np.allclose(A, A.T, atol=tol)
```

2. **Poprawianie współczynnika uwarunkowania**

Poprzez skalowanie

[348]:
```
C,b = load_A_b_from_mat_file("pink.mat") #pink - macierz źle uwarunkowana
```

[349]:
```
print(f"przed skalowaniem: {np.linalg.cond(C)}")

S = np.diag(1/np.sqrt(np.diag(C)))
A_scaled = S*C*S

print(f"po skalowaniu: {np.linalg.cond(A_scaled)}")
```

```
przed skalowaniem: 1844.6339859461466
po skalowaniu: 1.0000000000000007
```

3. **Wyznaczanie współczynnika uwarunkowania metodą iteracyjną**

Implementacja metody

[350]:
```
def calc_max_sing_val(C):
    stop = 1e-6  #
    A = C.T @ C
```

```
    x = np.ones(A.shape[0])
    x /= np.linalg.norm(x)

    lambda_old = 0
    lambda_new = x.T @ A @ x

    while abs(lambda_new - lambda_old) / (abs(lambda_old) + 1e-10) > stop:
        lambda_old = lambda_new
        x = A @ x
        x /= np.linalg.norm(x)
        lambda_new = x.T @ A @ x

    return np.sqrt(lambda_new)
```

[351]:
```
def calc_min_sing_val(A, lam):
    stop = 1e-6
    B = A - np.eye(A.shape[0])*lam

    x = np.ones(B.shape[0])
    x /= np.linalg.norm(x)

    lambda_old = 0
    lambda_new = x.T @ B @ x

    while abs(lambda_new - lambda_old) / (abs(lambda_old) + 1e-10) > stop:
        lambda_old = lambda_new
        x = B @ x
        x /= np.linalg.norm(x)
        lambda_new = x.T @ B @ x

    return lambda_new
```

[352]:
```
def calc_cond(C):
    C_sym = C.T@C

    lambda_max = calc_max_sing_val(C_sym)
    lambda_min = calc_min_sing_val(C_sym,lambda_max) + lambda_max


    return np.sqrt(lambda_max/lambda_min)
```

[353]:
```
def get_result(A):
    result_impl = calc_cond(A)
    result_ready = np.linalg.cond(A)
    diff = abs(result_ready - result_impl) / result_ready * 100

    return result_impl, result_ready, diff
```

Wyznaczenie współczynnika i błędu dla macierzy C z podręcznika

[354]:
```
C = np.array([
    [4,2,-5,2],
    [1,5,3,9],
    [2,2,5,-7],
    [1,4,-1,1]
])

result_impl, result_ready, diff = get_result(C)


print(f"Współczynnik obliczony zaimplementowaną funkcją: {result_impl}")
print(f"Współczynnik obliczony wbudowaną funkcją w numpy: {result_ready}")
print(f"różnica: {diff}%")
```

```
Współczynnik obliczony zaimplementowaną funkcją: 6.431575026765086
Współczynnik obliczony wbudowaną funkcją w numpy: 6.431698388238537
różnica: 0.0019180232965693217%
```

Jak możemy zauważyć, zaimplementowana metoda daje taki sam wynik różniący się o ok. 0,002%

4. **Analiza wpływu własności macierzy na współczynnik uwarunkowania**

Macierze z plików

[355]:
```
blue,b = load_A_b_from_mat_file("blue.mat") #dobrze
orange,b = load_A_b_from_mat_file("orange.mat") #dobrze
pink,b = load_A_b_from_mat_file("pink.mat") #zle

result_impl, result_ready, diff = get_result(blue)
print(f"Właściwości macierzy z blue.mat: kształt: {blue.shape}, symetryczność:␣
 ↪{is_symmetric(blue)}, błąd metody: {diff}, współczynnik wyliczony:␣
 ↪{result_impl}, współczynnik gotowy: {result_ready}")

result_impl, result_ready, diff = get_result(orange)
print(f"Właściwości macierzy z orange.mat: kształt: {orange.shape},␣
 ↪symetryczność: {is_symmetric(orange)}, błąd metody: {diff}, współczynnik␣
 ↪wyliczony: {result_impl}, współczynnik gotowy: {result_ready}")

result_impl, result_ready, diff = get_result(pink)
print(f"Właściwości macierzy z pink.mat: kształt: {pink.shape}, symetryczność:␣
 ↪{is_symmetric(pink)}, błąd metody: {diff}, współczynnik wyliczony:␣
 ↪{result_impl}, współczynnik gotowy: {result_ready}")
```

```
Właściwości macierzy z blue.mat: kształt: (100, 100), symetryczność: False, błąd
metody: 95.66196091084367, współczynnik wyliczony: 41.307139045529865,
współczynnik gotowy: 952.2076264546396
Właściwości macierzy z orange.mat: kształt: (100, 100), symetryczność: False,
błąd metody: 95.6619608797047, współczynnik wyliczony: 41.30713901693265,
```

```
współczynnik gotowy: 952.2076189603557
Właściwości macierzy z pink.mat: kształt: (100, 100), symetryczność: False, błąd
metody: 97.75137218105002, współczynnik wyliczony: 41.47895296579108,
współczynnik gotowy: 1844.6339859461466
```

Macierze wygenerowane

[356]:
```python
matrices = {
    "A (4x4, niesymetryczna)": np.array([
        [4, 2, -5, 2],
        [1, 5, 3, 9],
        [2, 2, 5, -7],
        [1, 4, -1, 1]
    ]),
    "B (4x4, symetryczna)": np.array([
        [2, -1, 0, 0],
        [-1, 2, -1, 0],
        [0, -1, 2, -1],
        [0, 0, -1, 2]
    ]),
    "C (10x10, losowa)": np.random.rand(10, 10),
    "D (100x100, losowa)": np.random.rand(100, 100)
}

for name, M in matrices.items():

    result_impl, result_ready, diff = get_result(M)

    print(f"\n{name}")
    print(f"współczynnik wyliczony:      {result_impl:.6e}")
    print(f"współczynnik gotowy: {result_ready:.6e}")
    print(f"błąd metody:  {diff:.6f}%")
```

```
A (4x4, niesymetryczna)
współczynnik wyliczony:      6.431575e+00
współczynnik gotowy: 6.431698e+00
błąd metody:  0.001918%

B (4x4, symetryczna)
współczynnik wyliczony:      6.854102e+00
współczynnik gotowy: 9.472136e+00
błąd metody:  27.639320%

C (10x10, losowa)
współczynnik wyliczony:      4.571885e+01
współczynnik gotowy: 2.728141e+02
błąd metody:  83.241756%
```

```
D (100x100, losowa)
współczynnik wyliczony:      4.550645e+01
współczynnik gotowy: 1.883439e+03
błąd metody:  97.583864%
```

5. **Wnioski**

Na podstawie powyższych wyników można wyciągnąć następujące wnioski:
- dla metod numerycznych im większa macierz, tym większy błąd metody - może być to spowodowane kumulowaniem sie błędów dla dużej liczby iteracji
- symetryczność i niesymetryczność macierzy nie jest najważniejszym czynnikiem wpływającym na współczynnik uwarunkowania

Kiedy macierz jest źle uwarunkowana:
- kiedy wartości własne są małe w porównaniu z największą wartością własną
- kiedy macierzy blisko jest do bycia macierzą osobliwą

Poprawianie współczynnika uwarunkowania:
- skalowanie macierzy
- regularizacja - dodanie małej wartości na przekątnej
- prekondycjonowanie
- wybór odpowiedniej metody numerycznej