

lab3

May 5, 2025

Rachunek Macierzowy

Projekt nr 2

Maciej Pięta, Hubert Kabziński

0.1 1. Opis zadania

Celem naszego zadania było przygotowanie programu, który umożliwi obliczenie różnych norm macierzowych, w szczególności normy $\|M\|_1$, $\|M\|_2$, $\|M\|_\infty$ oraz odpowiadających im współczynników uwarunkowania. Program został zaimplementowany w języku Python z wykorzystaniem bibliotek NumPy i SciPy. Obliczenia przeprowadzane są dla zadanej macierzy kwadratowej M oraz macierzy losowych.

```
[1]: import numpy as np
import matplotlib.pyplot as plt

from numpy.linalg import norm, svd

from scipy.optimize import minimize
```

0.2 2. Obliczanie normy $\|M\|_1$

Norma $\|M\|_1$ jest zdefiniowana jako maksymalna suma wartości bezwzględnych elementów w kolumnach macierzy:

$$\|A\|_1 = \max_{j=1, \dots, n} \left(\sum_{i=1}^n |a_{ij}| \right)$$

Oznacza to, że dla każdej kolumny obliczamy sumę modułów jej elementów, a następnie wybieramy największą z tych sum.

```
[2]: def norm_1(M):
    return np.max(np.sum(np.abs(M), axis=0))
```

0.3 3. Obliczanie normy $\|M\|_2$

Norma $\|M\|_2$, znana również jako norma spektralna, jest zdefiniowana jako największa wartość osobliwa macierzy M . Formalnie:

$$\|A\|_2 = |\lambda_1|$$

Gdzie $|\lambda_1|$ to największa liczba singularna macierzy AA , uzyskana przez rozkład SVD (Singular Value Decomposition).

```
[3]: def norm_2(M):
    singular_values = svd(M, compute_uv=False)
    return np.max(singular_values)
```

0.4 4. Obliczanie normy $\|M\|_\infty$

Norma $\|M\|_\infty$ jest zdefiniowana jako maksymalna suma wartości bezwzględnych elementów w wierszach macierzy:

$$\|A\|_\infty = \max_{i=1,\dots,n} \left(\sum_{j=1}^n |a_{ij}| \right)$$

Innymi słowy, dla każdej z n wierszy sumujemy wartości bezwzględne elementów, a następnie wybieramy największą z tych sum.

```
[4]: def norm_inf(M):
    return np.max(np.sum(np.abs(M), axis=1))
```

0.5 5. Obliczanie normy $\|M\|_p$

Normy macierzowe $\|M\|_p$ dla $p = 1, 2, \infty$ mają proste, klasyczne definicje analityczne. Jednak dla dowolnego innego $p > 0$, wyznaczenie tej normy nie jest trywialne i wymaga podejścia numerycznego.

Norma macierzy M w sensie p -normy jest zdefiniowana jako:

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p=1} \|Ax\|_p$$

Oznacza to, że szukamy takiego wektora x , którego p -norma wynosi 1 i który maksymalizuje p -normę iloczynu Mx

```
[5]: def norm_p(M, p, maxiter=1e6):
    n = M.shape[1]

    def objective(x):
        x = x.reshape(-1)
        Ax = A@x
        return -np.linalg.norm(Ax, ord=p)

    constraints = {
        'type' : 'eq',
        'fun' : lambda x: np.linalg.norm(x, ord=p) - 1,
    }
```

```

x0 = np.ones(n)/np.sqrt(n)
result = minimize(objective, x0, constraints=[constraints], method=
↳ 'SLSQP', options={'maxiter': maxiter})

return -result.fun

```

Funkcja `norm_p` realizuje obliczanie normy macierzy M w sensie p -normy poprzez rozwiązanie problemu optymalizacji. Najpierw określany jest wymiar wejściowy macierzy, co pozwala przygotować wektor startowy x_0 , w którym wszystkie współrzędne są równe i znormalizowane do długości bliskiej 1.

Następnie definiowana jest funkcja celu (`objective`), która zwraca przeciwną wartość normy $\|Mx\|_p$, ponieważ używany algorytm optymalizacyjny minimalizuje funkcję, a naszym celem jest jej maksymalizacja.

Wprowadzamy ograniczenie równościowe, które wymusza, aby norma wektora xx wynosiła dokładnie 1. Cały problem rozwiązywany jest za pomocą funkcji `minimize` z pakietu `scipy.optimize`, z wykorzystaniem metody SLSQP (Sequential Least Squares Programming).

SLSQP to metoda optymalizacji ograniczonej, która umożliwia rozwiązywanie problemów z ograniczeniami zarówno równościowymi, jak i nierównościowymi przy użyciu technik programowania kwadratowego.

Jeżeli optymalizacja zakończy się sukcesem, wynik funkcji stanowi największą wartość $\|Mx\|_p$, czyli poszukiwaną normę macierzy M .

0.6 6. Obliczanie współczynnika uwarunkowania $\|M\|_p$

Współczynnik uwarunkowania macierzy M w normie p , oznaczany jako $cond_p(M)$, jest miarą tego, jak bardzo wynik rozwiązania układu równań liniowych może się zmienić w odpowiedzi na małe zmiany w danych wejściowych. Współczynnik uwarunkowania oblicza się jako iloczyn normy macierzy M oraz normy jej odwrotności:

$$cond_p(M) = \|M\|_p * \|M\|_p^{-1}$$

```

[6]: def condition_number_p(M, p, approximate=True):
    if M.shape[0] != M.shape[1]:
        raise ValueError("Condition number is only defined for square matrices.
↳ ")

    M_inv = np.linalg.inv(M)

    if p not in [1, 2, np.inf] or approximate:
        norm_M = norm_p(M, p)
    elif p == 1:
        norm_M = norm_1(M)
    elif p == 2:
        norm_M = norm_2(M)
    elif p == np.inf:

```

```

    norm_M = norm_inf(M)

    norm_M_inv = norm_p(M_inv, p)

    return norm_M * norm_M_inv

```

0.7 7. Wyniki i sprawdzenie poprawności i dokładności implementacji obliczania normy $\|M\|_p$

W celu sprawdzenia poprawności i dokładności implementacji funkcji `norm_p`, przeprowadziliśmy eksperymenty na małej macierzy oraz na większych macierzach losowych porównując wyniki z wartościami wyznaczanymi bezpośrednio za pomocą znanych wzorów dla norm macierzowych dla $p = 1, 2, \infty$.

0.7.1 Przykładowe wyniki dla macierzy $A = \begin{pmatrix} 4 & 9 & 2 \\ 3 & 5 & 7 \\ 8 & 1 & 6 \end{pmatrix}$

```

[7]: A = np.array([[4,9,2],[3,5,7],[8,1,6]])
      print("Matrix A:\n", A)

```

```

Matrix A:
[[4 9 2]
 [3 5 7]
 [8 1 6]]

```

0.7.2 Norma $p=1$

```

[8]: p=1
      print(f"P-norm (p={p}): {norm_1(A)}")
      print(f"P-norm (p={p}) (using approximation): {norm_p(A, p)}")
      print(f"Condition number (p={p}): {condition_number_p(A, p, approximate=False):.4f}")
      print(f"Condition number (p={p}) (using approximation): {condition_number_p(A, p, approximate=True):.4f}")

```

```

P-norm (p=1): 15
P-norm (p=1) (using approximation): 14.999999999999996
Condition number (p=1): 225.0000
Condition number (p=1) (using approximation): 225.0000

```

0.7.3 Norma $p=2$

```

[9]: p=2
      print(f"P-norm (p={p}): {norm_2(A)}")
      print(f"P-norm (p={p}) (using approximation): {norm_p(A, p)}")
      print(f"Condition number (p={p}): {condition_number_p(A, p, approximate=False):.4f}")

```

```
print(f"Condition number (p={p}) (using approximation): {condition_number_p(A, p, approximate=True):.4f}")
```

P-norm (p=2): 15.000000000000002
P-norm (p=2) (using approximation): 15.000000000000004
Condition number (p=2): 225.0000
Condition number (p=2) (using approximation): 225.0000

0.7.4 Norma $p=\text{inf}$

```
[10]: p=np.inf
print(f"P-norm (p={p}): {norm_inf(A)}")
print(f"P-norm (p={p}) (using approximation): {norm_p(A, p)}")
print(f"Condition number (p={p}): {condition_number_p(A, p, approximate=False):.4f}")
print(f"Condition number (p={p}) (using approximation): {condition_number_p(A, p, approximate=True):.4f}")
```

P-norm (p=inf): 15
P-norm (p=inf) (using approximation): 14.999996434057024
Condition number (p=inf): 224.9999
Condition number (p=inf) (using approximation): 224.9999

0.7.5 Norma $p=3$

```
[11]: p=3
print(f"P-norm (p={p}) (using approximation): {norm_p(A, p)}")
print(f"Condition number (p={p}) (using approximation): {condition_number_p(A, p, approximate=True):.4f}")
```

P-norm (p=3) (using approximation): 14.999999981014176
Condition number (p=3) (using approximation): 225.0000

0.8 Różnice wyniku aproksymacji od dokładnej wartości

```
[12]: def plot_norm_comparison(sizes, approx_norms, exact_norms, p):
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    plt.loglog(sizes, approx_norms, 'o-', label='Approximate')
    plt.loglog(sizes, exact_norms, 's--', label='Exact')
    plt.xlabel('Matrix size (n x n)')
    plt.ylabel(f'Matrix norm (p={p})')
    plt.title(f'Norm Comparison (p={p})')
    plt.legend()
    plt.grid(True, which="both", ls="--")

    plt.subplot(1, 2, 2)
```

```

abs_error = [abs(a - e) for a, e in zip(approx_norms, exact_norms)]
plt.plot(sizes, abs_error, 'x-m', label='Absolute Error')
plt.xlabel('Matrix size (n x n)')
plt.ylabel('Absolute Error')
plt.title('Difference between Approx and Exact Norms')
plt.yscale('log')
plt.grid(True, which="both", ls="--")
plt.legend()

plt.tight_layout()
plt.show()

```

0.8.1 $p = 1$

```

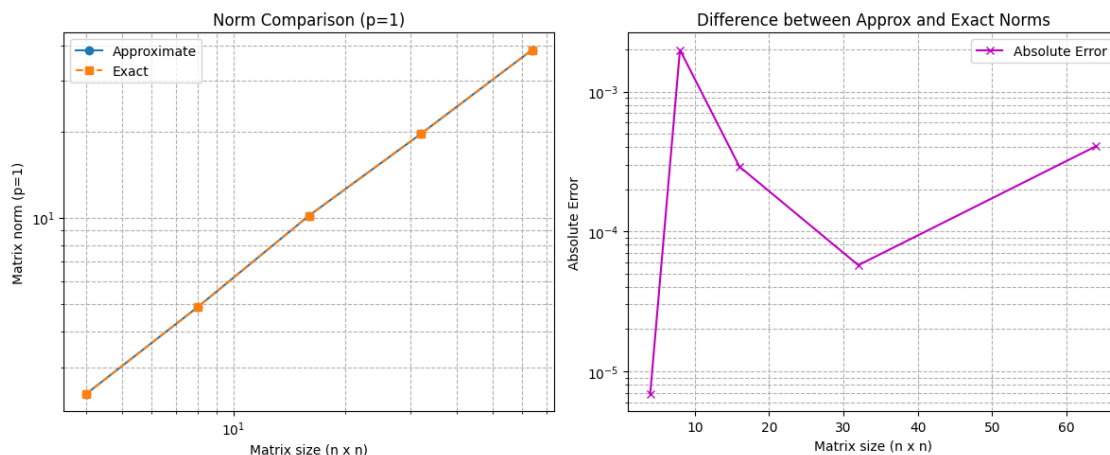
[13]: sizes = [2**i for i in range(2,7)]

p = 1
approx_norms = []
exact_norms = []

for size in sizes:
    A = np.random.rand(size, size)
    approx_norm = norm_p(A, p)
    exact_norm = norm_1(A)

    approx_norms.append(approx_norm)
    exact_norms.append(exact_norm)
plot_norm_comparison(sizes, approx_norms, exact_norms, p)

```



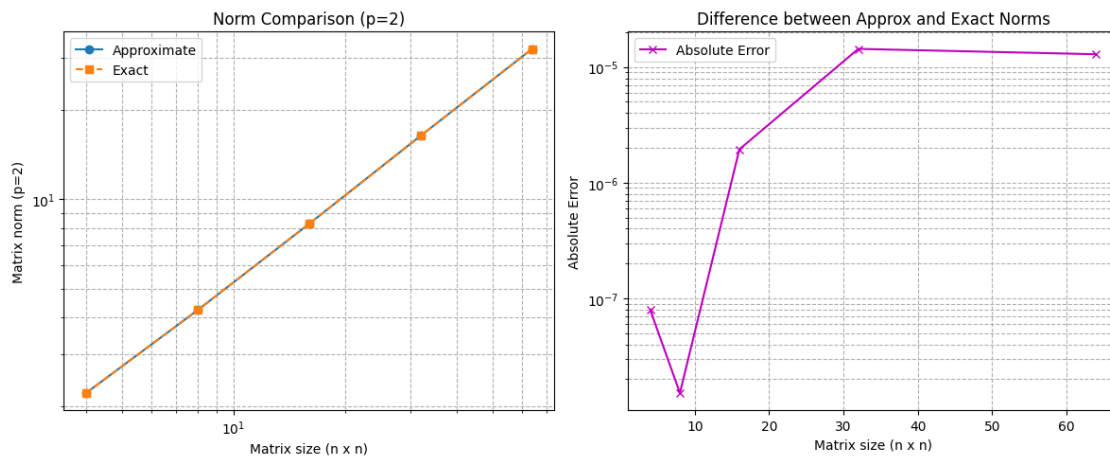
0.8.2 $p = 2$

```
[14]: p = 2
approx_norms = []
exact_norms = []

for size in sizes:
    A = np.random.rand(size, size)
    approx_norm = norm_p(A, p)
    exact_norm = norm_2(A)

    approx_norms.append(approx_norm)
    exact_norms.append(exact_norm)

plot_norm_comparison(sizes, approx_norms, exact_norms, p)
```



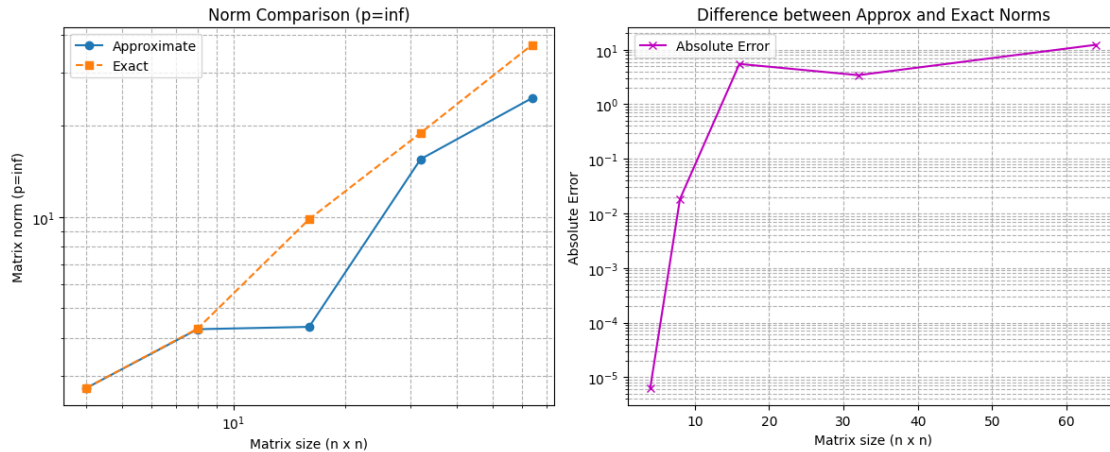
0.8.3 $p = \infty$

```
[15]: p = np.inf
approx_norms = []
exact_norms = []

for size in sizes:
    A = np.random.rand(size, size)
    approx_norm = norm_p(A, p)
    exact_norm = norm_inf(A)

    approx_norms.append(approx_norm)
    exact_norms.append(exact_norm)

plot_norm_comparison(sizes, approx_norms, exact_norms, p)
```

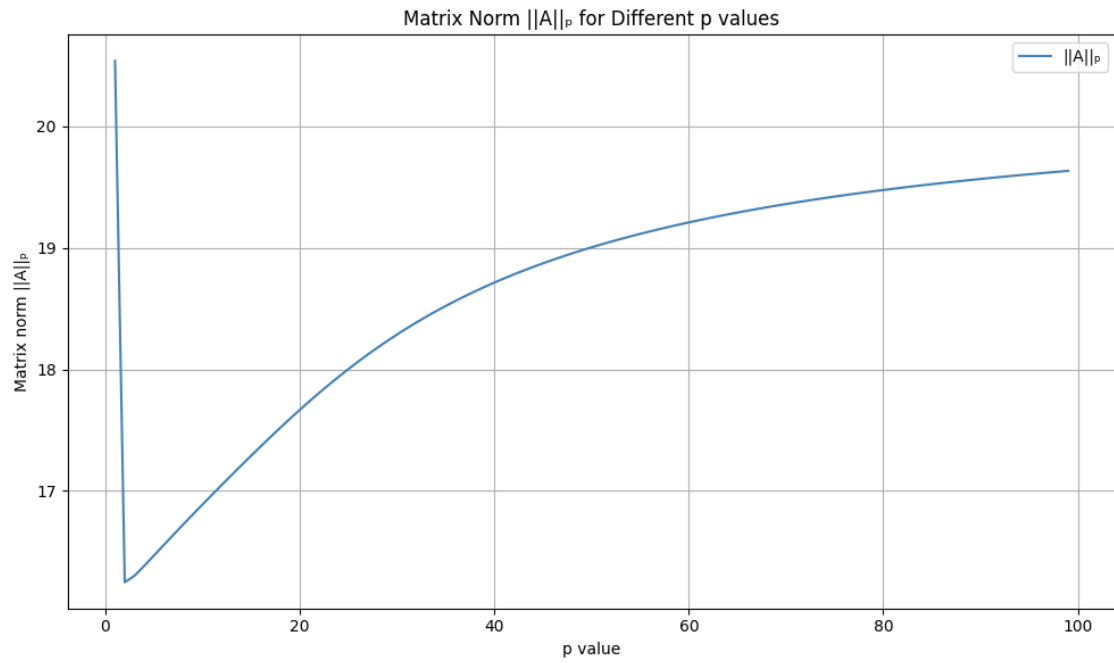


0.9 Wartości kolejnych p-norm dla macierzy losowej 32x32

```
[16]: A = np.random.rand(32,32)
ps = [i for i in range(1, 100)]

norms = []
for p in ps:
    norm = norm_p(A, p)
    norms.append(norm)

plt.figure(figsize=(10,6))
plt.plot(ps, norms, label='||A|| ', color='steelblue')
plt.xlabel('p value')
plt.ylabel('Matrix norm ||A|| ')
plt.title('Matrix Norm ||A|| for different p values')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

[]: