SPRAWOZDANIE NUM6

JAKUB KRĘCISZ

Treść zadania

Zadana jest macierz:

$$M = \begin{pmatrix} 3 & 6 & 6 & 9 \\ 1 & 4 & 0 & 9 \\ 0 & 0.2 & 6 & 12 \\ 0 & 0 & 0.1 & 6 \end{pmatrix}$$

(a) Stosując algorytm QR znajdź wszystkie wartości własne macierzy **M**.
(b) Stosując metodę potęgową znajdź największą co do modułu wartość własną macierzy **B** z zadania 9 oraz odpowiadający jej wektor własny.

Wyniki sprawdź używając wybranego pakietu algebry komputerowej.

Dla ambitnych: w pkt. (a) wyznacz również wektory własne.

Omówienie zadania

Naszym zadaniem jest wyznaczenie wartości własnych oraz odpowiadających im wektorów własnych poprzez użycie dwóch odmiennych metod. Pierwszą metodą, którą użyjemy na podanej macierzy M będzie algorytm QR. Natomiast drugą metodą, którą użyjemy na podanej macierzy B, jest metoda potęgowa.

Zajmijmy się na początek algorytmem QR. Zasada działania algorytmu QR dla danej macierzy, w naszym przypadku macierzy M to faktoryzacja QR a następnie policzenie wyniku iloczynu tych czynników wziętych w odwrotnej kolejności. Na dodatek procedurę możemy iterować:

$$A_1 = M$$

$$A_1 = Q_1 R_1$$

$$A_2 = R_1 Q_1 = Q_2 R_2$$

$$A_3 = R_2 Q_2 = Q_3 R_3$$

Co możemy przedstawić wzorem:

$$A_n = R_{n-1} Q_{n-1} = Q_n R_n$$

Widzimy, że wymnożenie czynników faktoryzacji QR w odwrotnej kolejności stanowi ortogonalną transformację podobieństwa naszej macierzy A. Przy każdej następnej iteracji podanej procedury macierz A_n zbiega do macierzy trójkątnej górnej, w której na diagonali stoją kolejne wartości własne.

Zajmijmy się teraz naszą drugą metodą, czyli metodą potęgową. Do tej metody potrzebujemy, by nasza macierz była macierzą symetryczną. Dzięki temu wiemy, że naszą macierz możemy zdiagonalizować, ma rzeczywiste wartości własne a unormowane wektory własne tworzą bazę ortonormalną. Przyjmijmy, że naszą bazą będzie:

$$\{e_i\}_{i=1}^{N}$$

Przy czym:

$$Ae_i = \lambda_i e_i$$

Przyjmijmy również, że wartości własne są malejąco uporządkowane, więc λ_1 będzie największą wartością własną.

Jeśli weźmiemy wektor y o takim samym wymiarze jak nasza macierz (przyjmijmy nasz wymiar jako N) możemy go rozłożyć na sumę:

$$y = \sum_{i=1}^{N} \beta_i e_i$$

A zatem możemy użyć wzoru:

$$A^k y = \sum_{i=1}^{N} \beta_i \lambda_i^k e_i$$

Widzimy, że dla dostatecznie dużych k wyraz zawierający $\lambda^k{}_1$ będzie się znacznie dominował pozostałe współczynniki w naszej sumie. Powoduje to, że dla dostatecznie dużych k, suma będzie dążyć do wektora własnego λ_1, którym jest wektor własny do największej wartości własnej. Dzięki temu wiemy, że iteracja:

$$Ay_k = z_k$$

$$y_{k+1} = \frac{z_k}{||z_k||}$$

Zbiega do unormowanego wektora własnego. Odpowiadającą temu wektorowi największą wartość własną możemy obliczyć poprzez wyliczenie:

$$\lambda_{MAX} = ||z_k||$$

Jak dobrze wiemy, metody iteracyjne w zasadzie mogą iść w nieskończoność. Dlatego też, musimy ustalić jakieś limity i warunki, kiedy program musi się zakończyć:

1. Dla algorytmu QR, warunkiem zakończenia programu jest sprawdzenie przy każdej iteracji wartości poddiagonalnych, jeżeli wszystkie wartości będą mniejsze od naszej ustalonej precyzji, wtedy algorytm zakończy działanie i zwróci rezultat.
2. Dla metody potęgowej użyjemy normy euklidesowej, będziemy porównywać normę z aktualnej iteracji z tą z poprzedniej, jeśli różnica będzie mniejsza niż nasza ustalona precyzja, wtedy program zakończy działanie i wypisze rezultat.
3. Zdefiniowałem również stałą w config.py w której definiujemy maksymalną ilość iteracji dla obu metod, żeby program nie działał w nieskończoność.

Uruchomienie programu

Do uruchomienia programu wykorzystamy Makefile:

Aby uruchomić nasz program, wystarczy użyć polecenia:

```
make run
```

Wyniki

1. Pierwszym podpunktem było wyliczenie wszystkich wartości własnych metodą QR:

   $\lambda_1 = 1.0530438315655364$

   $\lambda_2 = 4.815806590682049$

   $\lambda_3 = 5.900157268333131$

   $\lambda_4 = 7.230992309419286$

2. Drugim podpunktem było wyznaczenie największej co do modułu wartości własnej i odpowiadający jej wektor własny, metodą potęgową:

   $\lambda_{MAX} = 10.015982848255213$

   $y = \begin{bmatrix} 0.5582969 \\ 0.77620837 \\ 0.28678781 \\ 0.0596481 \end{bmatrix}$

Wnioski

Jeżeli chodzi o wyniki to obie metody działają, a wyniki porównane z wbudowaną biblioteką numPy się zgadzają. Wyniki przy dobrze określonej precyzji (w naszym przypadku 10^{-16}) mogą wyznaczyć o wiele bardziej precyzyjne wyniki wartości własnych i wektorów własnych.

Następną rzeczą, którą mogę porównać to ilość iteracji jaką potrzebowała każda z metod. Oczywiście nie jest to miarodajne i nie odzwierciedla, która z metod jest lepsza, ponieważ każda z nich wykonuje całkiem co innego. Ale jeżeli zależałoby nam jedynie na jednej wartości własnej i to tej największej a nie wszystkich, metoda potęgowa wykonuje zadanie w znacznie mniejszej ilości iteracji a co za tym idzie w krótszym czasie.