8 stycznia 2020

Michał Gozdera
grupa: G1
nr indeksu: 298869

4. Odwrotna metoda potęgowa z normowaniem (przypadek zespolony). Poszukiwanie najmniejszej (co do wartości bezwzględnej) wartości własnej macierzy A. Rozwiązywanie odpowiedniego układu równań metodą eliminacji Gaussa z częściowym wyborem elementu głównego

Projekt nr 4

1 Wstęp

Celem projektu jest poszukiwanie najmniejszej co do wartości bezwzględnej wartości własnej macierzy A o elementach zespolonych. Stosujemy odwrotną metodę potęgową z normowaniem. Do rozwiązania układu równań (potrzebne, aby uniknąć wyliczania macierzy odwrotnej) używamy metody eliminacji Gaussa z częściowym wyborem elementu głównego.

Powyższa metoda wymaga, aby spełniony był szereg dość restrykcyjnych założeń, tzn. nie działa w każdym przypadku. Jeżeli wiemy, że dla macierzy A założenia te są spełnione, odwrotna metoda potęgowa daje dobre efekty przy niskim koszcie obliczeniowym. Liczba iteracji potrzebnych do uzyskania określonej dokładności zależy od doboru przybliżenia początkowego.

2 Opis metody

Opracujemy najpierw algorytm pomocniczy, tj. rozwiązywanie układu równań liniowych. Jak zostało napisane powyżej, stosujemy metodę eliminacji Gaussa z częściowym wyborem (ang. Gauss Elimination with Partial Pivoting - GEPP). Użycie GEPP ma tę przewagę nad eliminacją Gaussa bez częściowego wyboru (GE), że praktycznie zawsze daje się poprawnie wykonać (w GE istnieje ryzyko dzielenia przez 0). Kolejną zaletą jest zmniejszenie błędów zaokrągleń, powstałych podczas obliczania iloczynów.

Niech $M \in \mathbb{C}^{n\times n}$ oraz:

$$M = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix}.$$

Iterujemy dla i od 1 do $n-1$ (każda iteracja dotyczy kolumny). W każdym kroku poszukujemy w i-tej kolumnie maksymalnego co do wartości bezwzględnej elementu spośród znajdujących się w wierszach od i-tego do n-tego. Niech k - numer wiersza z maksymalnym dla danej iteracji elementem. Dokonujemy zamiany wierszy i-tego i k-tego (w celu zachowania oznaczeń przyjmujemy teraz, że wiersz i-ty zawiera elementy k-tego i odwrotnie). Dalej wykonujemy "zwykłą" eliminację Gaussa, to znaczy korzystamy z wzoru:

$$r_j = r_j - \frac{a_{j,i}}{a_{i,i}} r_i$$

gdzie r_j, r_i - wiersze j-ty i i-ty, $j = i, i+1, \dots, n$.

Wybierając największy co do modułu element w kolumnie, niwelujemy ryzyko równości $a_{i,i} = 0$. Również błędy zaokrągleń ulegają zmniejszeniu - licznik iloczynu (czyli $a_{j,i}$) jest na pewno mniejszy lub równy mianownikowi ($a_{i,i}$).

Dysponując algorytmem GEPP możemy zaimplementować algorytm odwrotnej metody potęgowej z normowaniem.

Niech dana macierz $A \in \mathbb{C}^{n\times n}$. Chcemy znaleźć $\lambda_1 \in \rho(A)$ takie, że:

$$\forall(\lambda \in \rho(A),\ \lambda \neq \lambda_1) \quad |\lambda_1| < |\lambda|.$$

Za pomocą metody potęgowej możemy dla macierzy A znaleźć jej największą wartość własną. Ponieważ:

$$(det(A) \neq 0\ \wedge\ \lambda \in \rho(A^{-1})) \implies \frac{1}{\lambda} \in \rho(A),$$

więc znajdując największą wartość własną dla macierzy A^{-1} otrzymamy odwrotność najmniejszej wartości własnej macierzy A. Teoretycznie można w tym celu zastosować metodę potęgową dla macierzy A^{-1}, ale praktycznie jest to nieefektywne z powodu konieczności obliczania macierzy odwrotnej. Mnożąc zatem obustronnie z lewej strony równanie iteracji w metodzie potęgowej dla A^{-1} przez A otrzymujemy wzór odwrotnej metody potęgowej i zamiast wyznaczania macierzy odwrotnej rozwiązujemy układ równań (metodą GEPP).

Niech $x^{(0)} \in \mathbb{C}^n$ - pewne przybliżenie początkowe wektora własnego odpowiadającego λ_1. W k-tej iteracji wykonujemy:

$$Ay^{(k+1)} = x^{(k)} \tag{1}$$

$$x^{(k+1)} = \frac{y^{(k+1)}}{\| y^{(k+1)} \|_2}$$

$$\lambda^{(k)} = \frac{< y^{(k+1)}, x^{(k)} >}{\| x^{(k)} \|_2^2}$$

Normę można dobrać dowolnie, ale w praktyce korzystnie wybrać normę drugą, wtedy: $\| x^{(k)} \|_2 = 1$ dla $k > 0$. Wyznaczenie $y^{(k+1)}$ w (1) polega na rozwiązaniu układu równań. Ostatecznie szukana najmniejsza wartość własna macierzy A to $\frac{1}{\lambda^{(k)}}$.

Normowanie stosuje się m. in. po to by zapobiec powstawaniu nadmiaru i niedomiaru (jeśli $|\lambda_1| < 1$ to $|\lambda_1|^k \xrightarrow{k\to\infty} 0$, a jeśli $|\lambda_1| > 1$ to $|\lambda_1|^k \xrightarrow{k\to\infty} \infty$).

Odwrotną metodę potęgową możemy jednak stosować tylko, gdy spełnione są określone założenia:

1. Macierz A jest nieosobliwa, tzn. $det(A) \neq 0$. W przeciwnym przypadku nie moglibyśmy znaleźć macierzy odwrotnej i odwrotna metoda potęgowa nie miałaby sensu.

2. Macierz A posiada najmniejszą co do modułu wartość własną λ_1 tj.:

   $$|\lambda_1| < |\lambda_2| \leqslant \ldots \leqslant |\lambda_n|$$

   gdzie $\lambda_1, \lambda_2, \ldots, \lambda_n \in \rho(A)$. Równoważny warunek: A^{-1} posiada dominującą wartość własną.

3. Macierz A jest diagonalizowalna (istnieje baza $v_1, v_2, \ldots, v_n$ złożona z wektorów własnych tej macierzy; $Av_i = \lambda_i v_i$, $i = 1, 2, \ldots, n$).

4. Przybliżenie początkowe $x^{(0)} \in \mathbb{C}^n$ posiada niezerową składową w kierunku wektora własnego odpowiadającego wartości własnej λ_1, tj.:

   $$x^{(0)} = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$$

   gdzie $\alpha_1 \neq 0$, $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{C}$. Równoważnie: $x^{(0)}$ posiada niezerową składową w kierunku wektora własnego odpowiadającego dominującej wartości własnej macierzy A.

3 Opis programu obliczeniowego

Program składa się z trzech funkcji:

- `[x] = GEPP(A, b)` - zwraca wektor rozwiązań `x` układu równań $Ax = b$, obliczony metodą eliminacji Gaussa z częściowym wyborem.

- `[w, it_count] = inverse_power_method(A, x0, eps)` - zwraca wartość najmniejszej wartości własnej `w` dla macierzy `A` obliczoną za pomocy odwrotnej metody potęgowej; `it_count` to liczba wykonanych iteracji, potrzebna do uzyskania przybliżenia z dokładnością `eps`; `x0` to przybliżenie początkowe wektora własnego odpowiadającego poszukiwanej wartości własnej.

- `[TAB] = test(n, k, eps)` - funkcja używana do testowania algorytmu. Generuje `k` macierzy rozmiaru `n` $\times$ `n` o elementach zespolonych. Argument wejściowy `eps` określa dokładność z jaką obliczamy wartość własną. Funkcja zwraca macierz `TAB` rozmiaru `k` $\times$ 4, której wiersze odpowiadają wygenerowanym macierzom testowym, a znaczenie poszczególnych kolumn to: pierwsza kolumna - informacja czy spełnione są założenia dla metody potęgowej ("should be correct" - gdy algorytm powinien zwrócić dobrą wartość, "assumptions are not met" - gdy nie są spełnione założenia tzn. wektory własne nie tworzą bazy $\mathbb{C}^n$ lub A nie posiada jednej minimalnej wartości własnej oraz "det(A) = 0" - gdy macierz A jest osobliwa); druga kolumna - najmniejsza wartość własna macierzy A obliczona przy pomocy wbudowanych funkcji środowiska *Matlab*; trzecia kolumna - najmniejsza wartość własna obliczona przy pomocy funkcji `inverse_power_method`; czwarta kolumna - liczba iteracji. Wartość przybliżenia początkowego wybierana jest losowo, ale zgodnie z założeniami.

4 Opis eksperymentów

Aby przetestować algorytm, generujemy losowe macierze kwadratowe o elementach zespolonych, korzystając z funkcji `test`. Porównujemy wartości obliczone odwrotną metodą potęgową i funkcjami środowiska *Matlab*. Zwracamy również uwagę na pierwszą kolumnę macierzy `TAB` - zapisane w niej informacje pozwolą na analizę "przydatności" odwrotnej metody potęgowej.

5 Przykłady obliczeniowe

Poniżej przedstawiam kilka przykładowych wywołań funkcji `test` wraz z fragmentami tabeli `TAB`:

1. `test(50, 10, 0.000001)`

Tabela 1: Tabela wyników dla 10 losowych macierzy rozmiaru 50×50

czy spełnione założenia?	λ dokładna	λ przybliżona	liczba iteracji
should be correct	-0.1556 - 0.2174i	-0.1556 - 0.2174i	24
should be correct	0.0885 - 0.4218i	0.0885 - 0.4218i	150
should be correct	0.1599 + 0.2599i	0.1599 + 0.2599i	25
should be correct	0.0387 + 0.0770i	0.0387 + 0.0770i	12
should be correct	-0.0679 + 0.2491i	-0.0679 + 0.2491i	54
should be correct	-0.2356 - 0.2582i	-0.2356 - 0.2582i	82
should be correct	0.3812 - 0.1054i	0.3812 - 0.1054i	519
should be correct	0.3508 - 0.1628i	0.3508 - 0.1628i	39
should be correct	0.3983 - 0.2049i	0.3983 - 0.2049i	63
should be correct	0.0110 - 0.1826i	0.0110 - 0.1826i	21

2. `test(8, 1000, 0.000001)`

Tabela 2: Fragment wyników dla 1000 losowych macierzy rozmiaru 8×8

czy spełnione założenia?	λ dokładna	λ przybliżona	liczba iteracji
should be correct	-0.1556 - 0.2174i	-0.1556 - 0.2174i	24
should be correct	-0.2543 + 0.3240i	-0.2543 + 0.3240i	33
should be correct	-0.4058 - 0.0703i	-0.4058 - 0.0703i	66
should be correct	0.2581 - 0.3834i	0.2581 - 0.3834i	42
should be correct	-0.1876 - 0.5205i	-0.1876 - 0.5205i	131
should be correct	0.1160 + 0.0930i	0.1160 + 0.0930i	14
should be correct	0.1035 + 0.0802i	0.1035 + 0.0802i	13
should be correct	0.0395 - 0.0970i	0.0395 - 0.0970i	30
should be correct	0.3537 + 0.0835i	0.3537 + 0.0835i	55
should be correct	-0.0751 + 0.0172i	-0.0751 + 0.0172i	11
should be correct	-0.0877 + 0.0209i	-0.0877 + 0.0209i	15
should be correct	-0.0555 - 0.2114i	-0.0555 - 0.2114i	58
should be correct	-0.2642 + 0.4184i	-0.2642 + 0.4184i	200
should be correct	0.0903 + 0.1169i	0.0903 + 0.1169i	29
should be correct	0.0278 + 0.3929i	0.0278 + 0.3929i	63
should be correct	-0.2307 - 0.2341i	-0.2307 - 0.2341i	40
should be correct	0.0061 - 0.5290i	0.0061 - 0.5290i	70
should be correct	-0.4769 + 0.0140i	-0.4769 + 0.0140i	49
should be correct	0.4600 + 0.4816i	0.4600 + 0.4816i	183
should be correct	0.1259 - 0.1242i	0.1259 - 0.1242i	14
should be correct	0.3708 - 0.2376i	0.3708 - 0.2376i	53

3. `test(4, 1, 0.000001)` - wywołana dla macierzy o 3-krotnej wartości własnej o najmniejszej wartości:

$$A = \begin{pmatrix} 4 & 0 & 2 & 3 \\ 0 & 4 & 2 & 1 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

Tabela 3: Wyniki dla powyższej macierzy

czy spełnione założenia?	λ dokładna	λ przybliżona	liczba iteracji
assumptions are not met	0×0 double	0×0 double	0×0 double

4. `test(4, 1, 0.000001)` - wywołana dla macierzy osobliwej:

$$A = \begin{pmatrix} 4 & 9 & 2 & 3 \\ 8 & 18 & 4 & 6 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

Tabela 4: Wyniki dla powyższej macierzy

czy spełnione założenia?	λ dokładna	λ przybliżona	liczba iteracji
det(A) = 0	0×0 double	0×0 double	0×0 double

W powyższych przykładach limit iteracji ustawiony został na 100 000.

6 Analiza wyników

Powyższe wywołania funkcji `test` dowodzą poprawności algorytmu. Dla ustalonych jak powyżej parametrów dokładności i limitu iteracji niemal zawsze uzyskujemy dokładny wynik.

Przykład nr 1 pokazuje wyniki dla macierzy o dużych rozmiarach.

W przykładzie nr 2 widać, że dla losowej macierzy o elementach zespolonych prawdopodobieństwo, że nie będzie spełniała założeń odwrotnej metody potęgowej jest bardzo małe (każda z 1000 losowych macierzy spełniała założenia). Liczba iteracji rzadko przekracza kilkaset, co biorąc pod uwagę zastosowaną wartość *eps* jest dobrym wynikiem.

W przykładach 3 i 4 jawnie wpisana została macierz, która nie spełnia założeń - dopiero wówczas uzyskaliśmy w pierwszej kolumnie macierzy `TAB` wynik inny niż "should be correct". Mimo tego, w praktycznych zastosowaniach możemy mieć do czynienia ze specyficznymi macierzami, dlatego zawsze należy upewniać się, czy spełniają one założenia odwrotnej metody potęgowej. Sprawdzenie założeń nie jest zadaniem trywialnym, w szczególności dobranie odpowiedniego przybliżenia początkowego

(posiadającego niezerowa składową w kierunku wektora własnego odpowiadającego poszukiwanej wartości własnej) może okazać się dużym problemem. Jeżeli znajdziemy taki wektor, liczba iteracji zależeć będzie od tego, jak "bliski" wektorowi własnemu jest to wektor. Przykładowo dla dokładności $eps = 0.0001$ i macierzy

$$A = \begin{pmatrix} 1 & 2 & 3 & 5 & 4 & 2 \\ 1 & 3 & 4 & 2 & 1 & 5 \\ 5 & 3 & 1 & 3 & 5 & 1 \\ 4 & 7 & 8 & 8 & 9 & 1 \\ 1 & 1 & 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 & 5 & 1 \end{pmatrix},$$

której najmniejszą wartością własną jest $\lambda_1 = 1.8430$, a odpowiadającym jej wektorem własnym jest:

$$v_0 = \begin{pmatrix} -0.2469 \\ -0.2676 \\ -0.0212 \\ -0.5708 \\ 0.7032 \\ 0.2159 \end{pmatrix},$$

metoda `inverse_power_method` z v_0 jako wartością początkową wektora własnego wykonuje się w dwóch iteracjach (co oczywiste - pierwsze przybliżenie jest od razu wektorem własnym odpowiadającym poszukiwanej wartości własnej), a dla przybliżenia początkowego:

$$v_1 = 40v_0 + \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 500000 \\ 322 \end{pmatrix},$$

będącego (rożnym od v_0) wektorem o niezerowej składowej w kierunku wektora własnego odpowiadającego λ_1 potrzeba 39 iteracji. Wnioskiem tego, dobranie odpowiedniego przybliżenia początkowego wpływa na liczbę iteracji algorytmu, a tym samym jego szybkość.

Przeanalizujemy teraz w jakim stopniu liczba iteracji zależy od dokładności, a w jakim od przybliżenia początkowego. Oczywistym jest, że gdy dla danej, konkretnej macierzy i ustalonego przybliżenia wykonujemy algorytm z większą dokładnością (mniejsze *eps*), liczba iteracji będzie większa. Ale już średnia liczba iteracji dla losowych macierzy tego samego rozmiaru i losowego przybliżenia nie spełnia tej zależności. Widać to w poniższej tabeli:

Tabela 5: Średnia liczba iteracji algorytmu dla 30 losowych macierzy 10×10 i dla zadanej dokładności

dokładność - eps	średnia liczba iteracji
0.0000000100000	134.0666666666667
0.0000001000000	58.8666666666667
0.0000010000000	49.7333333333333
0.0000100000000	521.2999999999999
0.0001000000000	138.8000000000000
0.0010000000000	54.9333333333333

Musi istnieć zatem czynnik inny niż zadana dokładność, który wpływa na liczbę iteracji. Jest to oczywiście wybór przybliżenia początkowego, co dodatkowo potwierdza wyniki eksperymentu z macierzą A (zdefiniowaną powyżej).

Podsumowując, w zdecydowanej większości przypadków odwrotna metoda potęgowa jest dobrym sposobem wyznaczania najmniejszej wartości własnej macierzy, jeżeli przyjmiemy, że zadana macierz spełnia wszystkie założenia konieczne dla zbieżności.