Wydział WFiIS	Imię i nazwisko 1. Ihnatsi Yermakovich		Rok II	Grupa 03	
METODY **NUMERYCZNE**	Temat Problem własny, Metoda potęgowa, Ilorazy Rayleighe'a				Nr ćwiczenia 05
Data wykonania 9.05.2022	Data oddania 14.05.2022	Zwrot do popr.	Data oddania	Data zaliczenia	OCENA

Problem własny, Metoda potęgowa, Ilorazy Rayleighe'a

Ćwiczenie nr 05

Ihnatsi Yermakovich

1 Wprowadzenie

Często przy tworzeniu modeli matematycznych wykorzystywanych do symulacji zjawisk fizycznych czy zachowania się układu, zachodzi potrzeba rozwiązania tzw. problemu własnego. Standardowy problem własny ma postać:

$$\mathbf{A}x = \lambda\mathbf{x} \tag{1}$$

gdzie:

$\mathbf{A}$ - dana macierz kwadratowa.

$\mathbf{x}$ - szukany wektor (tzw. wektor własny).

λ - szukana liczba (tzw. wartość własna).

W celu rozwiązania problemu własnego poznaliśmy rozkład QR. Pozwala on na znajdowanie wszystkich wektorów i wartości własnych. Natomiast w tym sprawozdaniu skupimy się na Metodzie potęgowej oraz Ilorazach Rayleighe'a.

Metoda potęgowa w wersji podstawowej służy do wyznaczania maksymalnej co do modułu wartości własnej i odpowiadającego jej wektora własnego. Możemy zmodyfikować metodę tak, aby przy jej pomocy wyznaczyć wartość własną najbliższą dowolnej liczbie. Metoda potęgowa w wersji podstawowej polega na wykonaniu ciągu mnożeń przyjętego wektora startowego przez macierz, której dominującej wartości własnej i odpowiadającego wektora własnego poszukujemy. Procedura taka jest równoznaczna pomnożeniu wektora startowego przez macierz wejściową podniesioną do potęgi równej liczbie iteracji. Wektor będący iloczynem zmierza wtedy do wektora odpowiadającego największej wartości własnej, samą wartość własną możemy natomiast obliczyć np. z wyrażenia zwanego ilorazem Rayleigha. Zapisać to możemy następująco:

$$r^{(k)} = \frac{\phi\left(\mathbf{x}^{(k+1)}\right)}{\phi\left(\mathbf{x}^{(k)}\right)} \tag{2}$$

gdzie:

k - numer iteracji.

ϕ - dowolny niezerowy funkcjonał.

Funkcjonał ϕ może być zdefiniowany np. jako pobranie pierwszego elementu wektora lub jako suma wszystkich elementów wektora, pierwszych n elementów wektora itd. ϕ wybiera się w zależności od problemu, który rozwiązujemy. Natomiast opisany proces wyznaczania szukanego wektora własnego możemy zdefiniować następująco:

$$\mathbf{x}^{(k+1)} = \mathbf{A}\mathbf{x}^{(k)} \tag{3}$$

Ze względu na ograniczenia numeryczne po każdej iteracji się normalizuje wektor $\mathbf{x}$. Wyniki dołączone do tego sprawozdania pokazują zastosowania 3 podstawowych norm: $\|x\|_\infty$, $\|x\|_1$, $\|x\|_2$.

2 Implementacja i wyniki

W celu ułatwienia sprawdzania poprawności zebranych wyników najpierw zaimplementowany został rozkład QR:

```
1  gsl_vector *wart_wlasne = gsl_vector_alloc(N);
2  gsl_matrix *wekt_wlasne = gsl_matrix_alloc(N, N);
3
4  // QR
5  {
6      gsl_eigen_symmv_workspace *wsp = gsl_eigen_symmv_alloc(N);
7      gsl_eigen_symmv(A, wart_wlasne, wekt_wlasne, wsp);
8      gsl_eigen_symmv_free(wsp);
9      gsl_eigen_symmv_sort(wart_wlasne, wekt_wlasne, GSL_EIGEN_SORT_ABS_DESC);
10 }
```

Listing 1: Znajdowanie wektorów i wartości własnych zap pomocą rozkładu QR

W linijce 9 elementy wektora sortujemy co to wartości bezwzględnej w celu ułatwienia pobierania prawidłowych danych w dalszej części programu przy Metodzie potęgowej.

A następnie zaimplementujemy Metodę potęgową odrazu korzystając z Ilorazów Rayleighe'a:

```
gsl_vector *v = gsl_vector_calloc(N);
for (size_t i = 0; i < N; i++)
    v_set(v, i, i + 1);

gsl_vector_scale(v, 1. / dnrmx(v));

int k = 500;
float r = 0;


for (size_t i = 0; i < k; i++)
{
    gsl_vector *u = gsl_vector_calloc(N);
    gsl_blas_dgemv(CblasNoTrans, 1, B, v, 0, u);

#ifndef Rayleigha
    r = (v_get(u, 0) + v_get(u, N - 1)) / (v_get(v, 0) + v_get(v, N - 1)); // can be different
#endif

#ifdef Rayleigha
    double ddot = 0;
    gsl_blas_ddot(u, v, &ddot);
    r = ddot / dnrm2(v) / dnrm2(v);
#endif

    vector_cpvv(u, v);
    gsl_vector_scale(v, 1. / dnrmx(v));

    gsl_vector_free(u);
}
```

Listing 2: Metoda potęgowa i Ilorazów Rayleighe'a

W naszym przypadku funkcjonał ϕ został zdefiniowany jako: $\phi(\mathbf{x}) = \mathbf{x}_1 + \mathbf{x}_N$. Funkcja dnrmx jest aliasem do jednej z trzech wspomnianych norm:

```
#define dnrmx dnrminf

double dnrm1(gsl_vector *v);
double dnrm2(gsl_vector *v);
double dnrminf(gsl_vector *v);
```

Listing 3: Definicja aliasu dnrmx

I ostatecznie zobaczymy przykładowe wyniki dla macierzy 6x6:

```
QR solution:
41.39054 -> [0.38417, 0.24380, 0.59376, 0.17528, 0.34985, 0.53600]
16.47539 -> [0.00282, -0.78828, 0.31926, -0.35386, -0.26083, 0.28883]
7.85948 -> [0.42668, -0.33031, 0.02210, -0.14553, 0.62730, -0.54191]
2.15292 -> [0.38477, -0.09476, 0.29960, 0.55574, -0.53582, -0.39656]
0.50781 -> [0.52136, -0.21361, -0.67473, 0.23271, -0.03086, 0.41496]
0.01386 -> [-0.50049, -0.39428, 0.00388, 0.67814, 0.35778, 0.07846]


Find first vector:

Result found within 0.000010 error in 10 iterations. Exit.

Vector found = [0.38417, 0.24381, 0.59376, 0.17529, 0.34986, 0.53600]
Vector by QR = [0.38417, 0.24380, 0.59376, 0.17528, 0.34985, 0.53600]


My wart wlasna = 41.39054
QR wart wlasna = 41.39054
```

Listing 4: Przykładowy output dla macierzy 6x6 przy zastosowaniu Ilorazów Rayleighe'a i normy $\|x\|_2$

Ciekawym faktem jest to, że dla macierzy 6x6 stosowanie normy $\|x\|_\infty$ daje rozwiązanie na 2 iteracji szybciej, czyli za 8 iteracji. Wszystkie wyniki są w dołączonym archiwum.

3 Wnioski

- Wejściowa macierz musi spełniać określone warunki, aby znajdowanie wartości własnych miało sens. Problem własny ma nietrywialne rozwiązanie $\mathbf{x} \neq \mathbf{0}$ tylko wtedy, gdy: $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$.
- Rozkład QR pozwala na znajdowanie wektorów i wartości własnych.
- Podstawowa wersja metody potęgowej pozwala na efektywne znajdowanie najwięszej co do modułu wartości własnej oraz wektoru własnego odpowiadającego tej wartości własnej.
- Dobór stosowanej normy oraz funkcjonału ϕ jest dokonywany z uwzględnieniem szczegołow problemu który probujemy rozwiązać.