Temat: Wyznaczanie wartości i wektorów własnych macierzy symetrycznej metodą potęgową z redukcją Hotellinga			
Wykonał: Marcin Fabrykowski	Wydział: FiIS	Kierunek Inf. Stos.	Grupa: grupa 3

1. Wstęp
   Metoda potęgową z wykorzystaniem redukcji Hotellinga jest skuteczna tylko dla macierzy symetrycznych. Aby wyznaczyć wartości własne należy wykorzystać następujący algorytm:

   (a) Ustalamy numer poszukiwanej wartości własnej $k = 1, 2, \ldots, n$

   (b) Ustalamy wektor startowy $x_0 = [1, 2, \ldots, n]$

   (c) Następnie iteracyjnie wykonujemy kolejne przybliżenia λ:

   i. $x_{i+1} = W_k x_i$

   ii. $\lambda_i = \frac{x_{i+1}^T}{x_i^T x_i}$

   iii. $x_{i+1} = \frac{x_{i+1}}{\|x_{i+1}\|_2}$

   iv. $x_i = x_{i+1}$

   po wykonaniu powyższego, wykonujemy redukcję macierzy W: $W_{k+1} = W_k - \lambda_k x_k x_k^T$

   Wartość λ_k reprezentuje k-tą wartość własną macierzy.

2. Wykonanie
   Wyznaczamy wektory własne macierzy A, gdzie $A_{ij} = A_{ji} = \sqrt{i+j}$ przy użycia biblioteki Numerical Reciples, a następnie porównujemy je z wartościami otrzymanymi metodą Hotellinga.
   Zadanie to realizuje poniższy program:

Listing 1: main.cpp

```
#include <iostream>
#include <cmath>
#include <nrutil.h>
#include <nrutil.c>
```

```
#include <tred2.c>
#include <tqli.c>
#include <pythag.c>
#include <fstream>
void Print(float** A,int n);
void Print(float *A, int n);
void Copy(float* a, float* b, int n);
void Copy(float** A, float** B, int n);
float* Mnoz(float**A, float *v, int n);
float Mnoz(float* v, float *w, int n);
float** Mnoz(float** A, float d, int n);
float* Dziel(float* v,float d, int n);
float** Mnoz2(float* v, float* w, int n);
float** Minus(float** A, float** B, int n);
using namespace std;
int main()
{
        int n=7;
        float **A=matrix(1,n,1,n);
        float **W=matrix(1,n,1,n);
        float *d=vector(1,n);
        float *e=vector(1,n);
        int i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=i;j++)
                {
                        A[i][j]=A[j][i]=sqrt(i+j);
                        W[i][j]=W[j][i]=A[i][j];
                };
        };

//      cout<<"A: "<<endl;Print(A,n);
        tred2(A,n,d,e);
//      cout<<"A: "<<endl;Print(A,n);
//      cout<<"d: "<<endl;Print(d,n);
//      cout<<"e: "<<endl;Print(e,n);

        float** Z=matrix(1,n,1,n);
        tqli(d,e,n,Z);
//      cout<<"Z: "<<endl;Print(Z,n);

        cout<<"d: "<<endl;Print(d,n);

        cout<<"**********************"<<endl;

        float *x=vector(1,n);
        float l;
        float *x1;
```

```
        int k;
        for(k=1;k<=n;k++)
        {
                cout<<"K: "<<k<<endl;
                for(i=1;i<=n;i++)
                        x[i]=1;
                for(i=0;i<8;i++)
                {
//                      cout<<"i: "<<i<<endl;
//                      cout<<"a"<<endl;
                        x1=Mnoz(W,x,n);
//                      cout<<"b"<<endl;
                        l=Mnoz(x1,x,n)/Mnoz(x,x,n);
//                      cout<<"c"<<endl;
                        x1=Dziel(x1,sqrt(Mnoz(x1,x1,n)),n);
//                      cout<<"d"<<endl;
                        Copy(x1,x,n);
//                      cout<<"e"<<endl;
                        cout<<"iter "<<i<<": l="<<l<<endl;
                };
//              cout<<l<<" ";
//              float** W_tmp=Mnoz2(x,x,n);
//              float** W_tmp2=Mnoz(W_tmp,l,n);
                Copy(Minus(W,Mnoz(Mnoz2(x,x,n),l,n),n),W,n);
        }
        cout<<endl;
};

void Print(float** A, int n)
{
        int i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        cout<<A[i][j]<<" ";
                };
                cout<<endl;
        };

};
void Print(float* A, int n)
{
        int i;
        for(i=1;i<=n;i++)
        {
                cout<<A[i]<<" ";
        }
        cout<<endl;
```

```
};
void Copy(float* a, float* b, int n)
{
        int i;
        for(i=0;i<=n;i++)
        {
                b[i]=a[i];
        };
};
float* Mnoz(float**A, float *v, int n)
{
        int i,j;
        float *w=vector(1,n);
        for(i=1;i<=n;i++)
        {
                w[i]=0;
                for(j=1;j<=n;j++)
                {
                        w[i]+=A[i][j]*v[j];
                };

        };
        return w;
};
float Mnoz(float* v, float *w, int n)
{
        float x=0;
        int i;
        for(i=1;i<=n;i++)
        {
                x+=v[i]*w[i];
        };
        return x;
};
float* Dziel(float* v,float d, int n)
{
        float* w=vector(1,n);
        int i;
        for(i=1;i<=n;i++)
        {
                w[i]=v[i]/d;
        };
        return w;
};
float** Mnoz2(float* v, float* w, int n)
{
        float** A=matrix(1,n,1,n);
        int i,j;
        for(i=1;i<=n;i++)
```

```
        {
                for(j=1;j<=n;j++)
                {
                        A[i][j]=v[i]*w[j];
                };
        };
        return A;
};
float** Minus(float** A, float** B, int n)
{
        float **C=matrix(1,n,1,n);
        int i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        C[i][j]=A[i][j]-B[i][j];
                };
        };
        return C;
};
float** Mnoz(float** A, float d, int n)
{
        float** B=matrix(1,n,1,n);
        int i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        B[i][j]=A[i][j]*d;
                };
        };
        return B;
};
void Copy(float** A, float** B, int n)
{
        int i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        B[i][j]=A[i][j];
                };
        };
};
```

Czego wynikiem jest:

```
d:
```

```
1.33909e-07 -7.69934e-07 -6.39993e-06 -0.000335143 -0.0133168 -0.71234 19.7862
**********************
K: 1
iter 0: l=19.4524
iter 1: l=19.7857
iter 2: l=19.7862
iter 3: l=19.7862
iter 4: l=19.7862
iter 5: l=19.7862
iter 6: l=19.7862
iter 7: l=19.7862
K: 2
iter 0: l=-0.0115419
iter 1: l=-0.712339
iter 2: l=-0.71234
iter 3: l=-0.71234
iter 4: l=-0.71234
iter 5: l=-0.71234
iter 6: l=-0.71234
iter 7: l=-0.71234
K: 3
iter 0: l=-2.29747e-06
iter 1: l=-0.0133155
iter 2: l=-0.0133172
iter 3: l=-0.0133172
iter 4: l=-0.0133172
iter 5: l=-0.0133172
iter 6: l=-0.0133172
iter 7: l=-0.0133172
K: 4
iter 0: l=-1.14124e-06
iter 1: l=-3.45381e-05
iter 2: l=-0.000335135
iter 3: l=-0.000335211
iter 4: l=-0.000335211
iter 5: l=-0.000335211
iter 6: l=-0.000335211
iter 7: l=-0.000335211
K: 5
iter 0: l=-1.14067e-06
iter 1: l=-1.49248e-06
iter 2: l=-3.12061e-06
iter 3: l=-6.25824e-06
iter 4: l=-6.68492e-06
iter 5: l=-6.70552e-06
iter 6: l=-6.70643e-06
iter 7: l=-6.70647e-06
K: 6
iter 0: l=-1.13532e-06
```

```
iter 1: l=-1.3838e-06
iter 2: l=-1.40774e-06
iter 3: l=-1.40998e-06
iter 4: l=-1.41018e-06
iter 5: l=-1.4102e-06
iter 6: l=-1.4102e-06
iter 7: l=-1.4102e-06
K: 7
iter 0: l=5.72041e-08
iter 1: l=4.26412e-07
iter 2: l=4.27796e-07
iter 3: l=4.27928e-07
iter 4: l=4.27942e-07
iter 5: l=4.27943e-07
iter 6: l=4.27943e-07
iter 7: l=4.27943e-07
```

3. Wnioski
   Metoda potęgowa jest nieco wolniejsza niż ta z Numerical Reciples, jednakże ta druga nie radzi sobie z większymi macierzami. Przestaje działać przy $n >= 180$. Natomiast metoda potęgowa daje sobie radę nawet z rozmiarami $n = 500$, co czyni ją bardzo przydatną przy większych macierzach.