Metody numeryczne

Wojciech Chrobak

27 listopada 2017

Zadanie 6

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & 1 \\ -1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 2 & -1 \\ 1 & 0 & 0 & -1 & 2 \end{bmatrix}$$

Aby znaleźć przybliżony wektor własny do wartości własnej $\lambda \approx 0.38197$ należy wykonać iterację opisaną poniżej. Wektorem startowym ustalmy takie y_1, że $||y_1|| = 1$.

$$(A - \tau\mathbb{1})^{-1} y_k = x_k \Longrightarrow (A - \tau\mathbb{1}) x_k = y_k$$

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

Aby obliczyć $(A - \tau\mathbb{1})x_k = y_k$ możemy zastosować algorytm Shermana-Morrisona a więc nasza macierz $A - uv^T$ wygląda tak:

$$A_1 = A - uv^T = \begin{bmatrix} 1 - 0.38197 & -1 & 0 & 0 & 0 \\ -1 & 2 - 0.38197 & 1 & 0 & 0 \\ 0 & 1 & 1 - 0.38197 & 1 & 0 \\ 0 & 0 & 1 & 2 - 0.38197 & -1 \\ 0 & 0 & 0 & -1 & 1 - 0.38197 \end{bmatrix}$$

$$u = v = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

Rozwiązujemy równania $A_1 z = b$ i $A_1 q = u$ po czym obliczamy x ze wzoru:

$$x = z - \frac{v^T z}{1 + v^T q} q$$

GSL

Użyte (ważniejsze) funkcje:

```
gsl_linalg_solve_symm_tridiag(diag, e, b, x)
```

rozwiązuje układ równań $Ax = b$ jeśli macierz A jest macierzą symetryczną, trójdiagonalną.

```
gsl_blas_ddot(x, y, result)
```

oblicza skalar $x^T y$

```
gsl_blas_dnrm2(x)
```

oblicza norme wektora x

Kod

```cpp
#include <iostream>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_linalg.h>

using namespace std;

int main() {
    const int size = 5;

    gsl_vector *diag, *e;
    diag = gsl_vector_alloc(size);
    // wektor na diagonali
    gsl_vector_set_all(diag, 2.0);

    // odejmujemy 1 od pierwszego i ostatniego (Sherman - Morrison)
    gsl_vector_set(diag, 0, 1.0);
    gsl_vector_set(diag, 4, 1.0);
    gsl_vector_set(diag, 2, 1.0);


    // odejmujemy wartosc wlasna na diagonali
    gsl_vector_add_constant(diag, -0.38197);

    // wektor nad i pod diagonala
    e = gsl_vector_alloc(size - 1);
    gsl_vector_set_all(e, 1.0);
    gsl_vector_set(e, 0, -1.0);
    gsl_vector_set(e, 3, -1.0);


    // wektory potrzebne do obliczenia Az = y ze wzoru Shermana - Morrisona
    gsl_vector *u, *v, *x, *q;
    u = gsl_vector_alloc(size);
    gsl_vector_set_all(u, 0);
    gsl_vector_set(u, 0, 1);
    gsl_vector_set(u, 4, 1);

    v = gsl_vector_alloc(size);
    gsl_vector_set_all(v, 0);
    gsl_vector_set(v, 0, 1);
    gsl_vector_set(v, 4, 1);

    x = gsl_vector_alloc(size);
    q = gsl_vector_alloc(size);


    // wektory potrzebne do wyliczenia wektora wlasnego
    gsl_vector *y, *temp;
    y = gsl_vector_alloc(size);
    temp = gsl_vector_alloc(size);

    gsl_vector_set_zero(y);
    // aby || y || = 1
    gsl_vector_set(y, 0, 1);


    // szukanie wektora wlasnego dla 0.38197
    while(true) {
        double x_prev_norm = gsl_blas_dnrm2(x);
        // Sherman - Morrison
        // Ax = y
        gsl_linalg_solve_symm_tridiag(diag, e, y, x);
```

```
63          // Aq = u
64          gsl_linalg_solve_symm_tridiag(diag, e, u, q);
65
66          double data1 = 0;
67          double data2 = 0;
68          double *vx = &data1;
69          double *vq = &data2;
70          //vTx
71          gsl_blas_ddot(v, x, vx);
72          // vTq
73          gsl_blas_ddot(v, q, vq);
74          // 1 + vTq
75          *vq += 1;
76
77          gsl_vector_scale(q, *vx / *vq);
78          gsl_vector_sub(x, q);
79          // obliczony x
80
81          // liczymy kolejny wektor y
82          double norm = gsl_blas_dnrm2(x);
83          gsl_vector_memcpy(temp, x);
84          gsl_vector_scale(temp, 1 / norm);
85          gsl_vector_memcpy(y, temp);
86
87
88          if (abs(norm - x_prev_norm) < 10e-16)
89              break;
90      }
91
92      cout << "Wektor wlasny dla wartosci 0.38197: " << endl;
93      gsl_vector_fprintf(stdout, y, "%f");
94
95      return 0;
96  }
```

Wynik

$\lambda = 0.38197$ dla $\begin{bmatrix} 0.601501 & 0.371748 & -0.000000 & -0.371748 & -0.601501 \end{bmatrix}$