Rozwiązanie zadania N3

Krzysztof Waniak

Rozwiązać układ równań metodą Shermana-Morrisona:

$$\begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 4 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix} \Leftrightarrow Bx = b \text{, gdzie } B = (A + uv^T)$$

Rozwiązaniem takiego układu jest: $x = B^{-1}b$

W celu obliczenia odwrotności macierzy B korzystamy ze wzoru Shermana-Morrisona, który sam udowadniałem przy tablicy na zajęciach:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u}$$

Przekształcając (po właściwym podstawieniu) wzór doprowadza się go do postaci następującej:

$$B^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{\alpha} \text{, gdzie } \alpha = 1 + v^T A^{-1}u$$

Program korzysta z funkcji bibliotecznych GSL. Dokumentacja wskazuje, że podane funkcje odwracają macierz za pomocą wzoru Shermana-Morrisona.

Kod programu:

```
#include<stdio.h>
#include<ctype.h>          /* zawiera F_OK itp.    */
#include<unistd.h>        /* zawiera funkcje access(), usleep()  */
#include <gsl/gsl_linalg.h>
#define wyp(a) printf(#a "\n")
#define wyp2(a) printf(#a)
#define wypisz(a) printf("%2.6f",a)
#define karetka printf("\n")
#define karetka2 printf("\n\n")
#define space printf(" ")

/* Sprawdzanie, czy wejściowy plik nie istnieje, jesli nie istnieje zwraca
wartosc "TRUE" */
int nieistnieje(const char* nazwa)
{
    return access(nazwa, F_OK);
}

int main (void)
{
    /* rozwiazanie macierzy pasmowej "cyklicznej" za pomoca biblioteki
```

```
GSL*/
    /*
    4, 1, 0, 0, 0, 0, 1,
    1, 4, 1, 0, 0, 0, 0,
    0, 1, 4, 1, 0, 0, 0,
    0, 0, 1, 4, 1, 0, 0,
    0, 0, 0, 1, 4, 1, 0,
    0, 0, 0, 0, 1, 4, 1,
    1, 0, 0, 0, 0, 1, 4
    */

    double b_data[] = { 1, 2, 3, 4, 5, 6, 7 };
    double e_data[] = { 1, 1, 1, 1, 1, 1, 1};
    double f_data[] = { 1, 1, 1, 1, 1, 1, 1};
    double d_data[] = { 4, 4, 4, 4, 4, 4, 4 };
    FILE *fwynik, *fdane;
    char plik_b[30];
    double x2[8];
    int p1;

    for(p1 = 0; p1 <= 7 ; p1++)
    {
            x2[p1]=0.0;
    }

    wyp(Podaj nazwe pliku wyjsciowego:);
    scanf("%s",  &plik_b[0]);
    while(!nieistnieje(plik_b))
    {
        karetka;
        wyp(Taka nazwa pliku juz istnieje);
        wyp(Wprowadz inna nazwe pliku);
        karetka;
        scanf("%s",  &plik_b[0]);
    }
    karetka;

    gsl_vector_view e = gsl_vector_view_array (e_data , 7 ) ;
    gsl_vector_view f = gsl_vector_view_array (f_data , 7 ) ;
    gsl_vector_view b = gsl_vector_view_array (b_data, 7);
    gsl_vector_view d = gsl_vector_view_array (d_data, 7);
    gsl_vector *x = gsl_vector_alloc (7);

    gsl_linalg_solve_cyc_tridiag(&d.vector, &e.vector , &f.vector ,
&b.vector , x) ;

    fwynik = fopen(plik_b, "w");
    wyp(Rozwiazanie:);
    gsl_vector_fprintf(fwynik, x, "%2.6f");
    gsl_vector_free(x);
    fclose(fwynik);

    fdane = fopen(plik_b, "r");
    for(p1 = 0; p1 <= 6 ; p1++)
    {
            fscanf(fdane,"%lf",&x2[p1]);
    }
    karetka;
    fclose(fdane);
    /* estetyczne wypisywanie wyniku */
    fwynik = fopen(plik_b, "w");
```

```
    for(p1 = 0; p1 <= 6; p1++)
    {
        fprintf(fwynik,"x%i = %2.6f \n", p1+1,x2[p1]);
        printf("x%i = %f", p1+1,x2[p1]);
        karetka;
    }

    fclose(fwynik);

    karetka2;
    p1=0;
    wyp2(Wynik obliczony i zapisany jako:);
    space;
    while(plik_b[p1]!='\0')  printf("%c",plik_b[p1++]);
    karetka2;

    /*system("pause");*/
    return 0;
}
```

Rozwiązanie zapisane do pliku wyniki.txt
Wynik działania programu:

```
Podaj nazwe pliku wyjsciowego:
wyniki.txt

Rozwiazanie:

x1 = -0.260163
x2 = 0.447154
x3 = 0.471545
x4 = 0.666667
x5 = 0.861789
x6 = 0.886179
x7 = 1.593496


Wynik obliczony i zapisany jako: wyniki.txt

Aby kontynuować, naciśnij dowolny klawisz . . .
```

Wyniki sprawdzone za pomocą programu Octave; zgadzają się.