Rozwiązanie zadania N4

Krzysztof Waniak

Program rozwiązuje układ równań podany w zadaniu N4 korzystając z metody LU.

$$Au = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -2 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} 0 \\ -2h^2 \\ -2h^2 \\ -2h^2 \\ \vdots \\ -2h^2 \\ -2h^2 \\ 1 \end{bmatrix} = b_1$$

dla N = 100 oraz h = 0.01.

Kod programu:

```
#include<stdio.h>
#include<ctype.h>          /* zawiera F_OK itp.    */
#include<unistd.h>        /* zawiera funkcje access(), usleep()  */
#define wyp(a) printf(#a "\n")
#define wyp2(a) printf(#a)
#define wypisz(a) printf("%lf",a)
#define karetka printf("\n")
#define karetka2 printf("\n\n")
#define space printf(" ")
#define N 100

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

void lu(double X[N+1][1], double A[N+1][N+1], double b[N+1][1])
{
    int p1, p2 ,p3;
    double Z[N+1][1];
    double L[N+1][N+1];
    double U[N+1][N+1];
    double suma_pomoc_1 = 0;
    double suma_pomoc_2 = 0;

    /* wypelniam macierz zerami jedynkami do LU */
    for(p1 = 0; p1<N+1; p1++)
    {
        for(p2 = 0; p2<N+1; p2++)
        {
            if(p1==p2) L[p1][p2] = 1;
            else L[p1][p2] = 0;
            U[p1][p2] = 0;
```

```
        }
    }



    for(p1 = 0; p1<N+1; p1++)
    {
        for(p2 = p1; p2<N+1; p2++)
            {
            for(p3 = 0; p3 <= p1-1; p3++)
            {
                suma_pomoc_1 += L[p1][p3]*U[p3][p2];
                suma_pomoc_2 += L[p2][p3]*U[p3][p1];
            }
            U[p1][p2] = A[p1][p2] - suma_pomoc_1;

            if(p2>= p1+1)
            {
                L[p2][p1] = (A[p2][p1] - suma_pomoc_2)/U[p1][p1];
            }
            suma_pomoc_1 = 0;
            suma_pomoc_2 = 0;
        }
    }

    suma_pomoc_1 = 0;

    for(p1 = 0; p1 < N+1; p1++)
    {
        for(p2 = 0; p2 < p1+1; p2++)
        {
            if( p1 != p2)
            {
                suma_pomoc_1 += L[p1][p2]*Z[p2][0];
            }
        }
        Z[p1][0] =  (b[p1][0] - suma_pomoc_1)/L[p1][p1];
        suma_pomoc_1 = 0;
    }

    suma_pomoc_2 = 0;

    for(p1 = N+1; p1 >= 0; p1--)
    {
        for(p2 = p1; p2 <N+1; p2++)
        {
            if( p1 != p2)
            {
                suma_pomoc_2 += U[p1][p2]*X[p2][0];
            }
        }
        X[p1][0] =  (Z[p1][0] - suma_pomoc_2)/U[p1][p1];
        suma_pomoc_2 = 0;
    }

}


int main(void)
{
    FILE *fwynik;
```

```
    char plik_b[30];
    int d, p1, p2;
    double h = 0.01;
    double X[N+1][1];
    double A[N+1][N+1];
    double b[N+1][1];

    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;

    /* tworzenie zadanej macierzy */
    for(p1 = 0; p1<N+1; p1++)
    {
        for(p2 = 0; p2<N+1; p2++)
        {
            if((p1 == 0 && p2 == 0)||((p1==N) && (p2 == N)))
            {
                A[p1][p2] = 1;
            }
            else
            {
                if(p1 == p2-1)
                {
                    A[p1][p2] = 1;
                }
                else if( p1 == p2 )
                {
                    A[p1][p2] = -2;
                }
                else if(p1 == p2+1)
                {
                    A[p1][p2] = 1;
                }
                else A[p1][p2] = 0;
            }
        }
    }
    A[0][1] = 0;
    A[100][99] = 0;


    /* tworzenie wektora wyrazow wolnych z zadania*/
    for(p1 = 0; p1<N+1; p1++)
    {
        if(p1 == 0)
        {
            b[p1][0] = 0;
        }
        else if( p1 == N)
        {
            b[p1][0] = 1;
        }
```

```
        else
        {
            b[p1][0] =  -2*h*h;
        }
    }

    lu(X, A, b);

    fwynik = fopen(plik_b, "w");
    for(p1 = 0; p1 < N+1; p1++)
    {
        fprintf(fwynik,"x%i = %f \n", p1+1 ,X[p1][0]);
        printf("x%i = %f", p1+1, X[p1][0]);
        karetka;
    }
    fclose(fwynik);

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

    return 0;

}
```

Wynik działania programu zapisany do pliku wynik.txt:

```
Podaj nazwe pliku wyjsciowego:
wynik.txt

x1  = 0.000000
x2  = 0.019900
x3  = 0.039600
x4  = 0.059100
x5  = 0.078400
x6  = 0.097500
x7  = 0.116400
x8  = 0.135100
x9  = 0.153600
x10 = 0.171900
x11 = 0.190000
x12 = 0.207900
x13 = 0.225600
x14 = 0.243100
x15 = 0.260400
x16 = 0.277500
x17 = 0.294400
x18 = 0.311100
x19 = 0.327600
x20 = 0.343900
x21 = 0.360000
x22 = 0.375900
x23 = 0.391600
x24 = 0.407100
x25 = 0.422400
x26 = 0.437500
x27 = 0.452400
x28 = 0.467100
x29 = 0.481600
x30 = 0.495900
x31 = 0.510000
x32 = 0.523900
x33 = 0.537600
x34 = 0.551100
x35 = 0.564400
x36 = 0.577500
x37 = 0.590400
x38 = 0.603100
x39 = 0.615600
x40 = 0.627900
x41 = 0.640000
x42 = 0.651900
x43 = 0.663600
x44 = 0.675100
x45 = 0.686400
x46 = 0.697500
x47 = 0.708400
x48 = 0.719100
x49 = 0.729600
x50 = 0.739900
x51 = 0.750000
x52 = 0.759900
x53 = 0.769600
x54 = 0.779100
x55 = 0.788400
x56 = 0.797500
x57 = 0.806400
x58 = 0.815100
x59 = 0.823600
x60 = 0.831900
x61 = 0.840000
x62 = 0.847900
x63 = 0.855600
x64 = 0.863100
x65 = 0.870400
x66 = 0.877500
x67 = 0.884400
x68 = 0.891100
x69 = 0.897600
x70 = 0.903900
x71 = 0.910000
x72 = 0.915900
x73 = 0.921600
x74 = 0.927100
x75 = 0.932400
x76 = 0.937500
x77 = 0.942400
x78 = 0.947100
x79 = 0.951600
x80 = 0.955900
x81 = 0.960000
x82 = 0.963900
x83 = 0.967600
x84 = 0.971100
x85 = 0.974400
x86 = 0.977500
x87 = 0.980400
x88 = 0.983100
x89 = 0.985600
x90 = 0.987900
x91 = 0.990000
x92 = 0.991900
x93 = 0.993600
x94 = 0.995100
x95 = 0.996400
x96 = 0.997500
x97 = 0.998400
x98 = 0.999100
x99  = 0.999600
x100 = 0.999900
x101 = 1.000000


Wynik obliczony i zapisany jako: wynik.txt
```