

Rozwiązanie zadania N4

Krzysztof Waniak

Dla układu równań $Ax = b$, gdzie:

$$A = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix}$$

Znaleźć rozwiązanie za pomocą metody gradientów sprzężonych.

Kod programu:

```
#include<stdio.h>
#include<math.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("%9.5f",a)
#define wypisz2(a) printf("%i",a)
#define karetka printf("\n")
#define karetka2 printf("\n\n")
#define space printf(" ")
#define ZERO      0
#define MACH_EPS  2e-16
#define SIZE      25
#define REAL      double

typedef REAL MAT[SIZE][SIZE];
typedef REAL VEC[SIZE];
int pl = 0;

int metoda_gr(int n, MAT a, VEC y, VEC x)
{
    VEC d, g, AmalD;
    REAL alpha, beta, dividend, divisor, pomoc, pomoc2, abstand, xnorm;
    int k, i, j;

    if (n < 2)
        return 1;

    for (i = n - 1; i >= 0; i--)
        if (a[i] == NULL)
            return 1;

    for (i = n - 1; i >= 0; i--)
        x[i] = ZERO;

    for (i = n - 1; i >= 0; i--)
        pomoc = y[i],
        d[i] = pomoc,
        g[i] = -pomoc;

    for (k = n; k > 0; k--)
    {
```

```

dividend = ZERO;
divisor  = ZERO;

for (i = n - 1; i >= 0; i--)
{
    dividend += d[i] * g[i];
    for (j = 0, pomoc = ZERO; j < i; j++)
        pomoc += a[j][i] * d[j];

    for (j = i; j < n; j++)
        pomoc += a[i][j] * d[j];

    AmalD[i] = pomoc;
    divisor += d[i] * pomoc;
}

if (divisor == ZERO)
{
    return 0;
}

alpha = -dividend / divisor;
xnorm  = ZERO;
abstand = ZERO;

for (i = n - 1; i >= 0; i--)
{
    pomoc = x[i];
    xnorm += pomoc*pomoc;
    pomoc2 = alpha * d[i];
    Abstand += pomoc2*pomoc2;
    x[i] = pomoc + pomoc2;
}

if (abstand < MACH_EPS * xnorm)
{
    return 0;
}

for (i = n - 1; i >= 0; i--)
    g[i] += alpha * AmalD[i];

dividend = ZERO;

for (i = n - 1; i >= 0; i--)
    dividend += g[i] * AmalD[i];

beta = dividend / divisor;

for (i = n - 1; i >= 0; i--)
    d[i] = -g[i] + beta * d[i];

++pl;
}
return 0;
}

int main(void)
{
    MAT  a;
    VEC  y, x;
    int  n=3, i, zabez;

```

```
/*int j;*/

/* macierz */
a[0][0]=4.0;
a[0][1]=-1.0;
a[0][2]=0.0;
a[1][0]= -1.0;
a[1][1]=4.0;
a[1][2]=-1.0;
a[2][0]= 0.0;
a[2][1]=-1.0;
a[2][2]= 4.0;

y[0]=2.0; y[1]=6.0; y[2]=2.0;

/*
for (i = 0; i < n; i++)
{
for (j = 0; j < i; j++)
printf(" %9.5f", a[j][i]);
for (j = i; j < n; j++)
printf(" %9.5f", a[i][j]);
printf(" %9.5f\n", y[i]);
}
*/
zabez = metoda_gr(n, a, y, x);

if(zabez)
{
printf("\n Error\n\n");
return 4 + zabez;
}

karetka;
wyp(Wynik:);
for (i = 0; i < n; i++)
{
wypisz(x[i]);
}
karetka2;

wyp2(Ilosc iteracji po ktorych znaleziono wynik =);
space;
wypisz2(p1);
karetka2;

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

Wynik działania programu zapisany do pliku wynik.txt:

```
Wynik:
1.00000 2.00000 1.00000
Ilosc iteracji po ktorych znaleziono wynik = 2
```