
# Laboratorium 6 – Układy równań liniowych

Tomasz Belczyk
13.06.2021

Metody Obliczeniowe w Nauce i Technice
Informatyka niestacjonarna 2020/2021
Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

# 1. Treść zadań

Korzystając z przykładu napisz program, który:

1. Jako parametr pobiera rozmiar układu równań n
2. Generuje macierz układu A(nxn) i wektor wyrazów wolnych b(n)
3. Rozwiązuje układ równań
4. Sprawdza poprawność rozwiązania (tj., czy Ax=b)
5. Mierzy czas dekompozycji macierzy - do mierzenia czasu można skorzystać z przykładowego programu dokonującego pomiaru czasu procesora spędzonego w danym fragmencie programu.
6. Mierzy czas rozwiązywania układu równań

• Zadanie domowe: Narysuj wykres zależności czasu dekompozycji i czasu rozwiązywania układu od rozmiaru układu równań. Wykonaj pomiary dla 10 wartości z przedziału od 10 do 1000.

# Podejście do rozwiązania zadań

Zaczynamy od implementacji programu i wczytywania wielkości układu równań

```c
int main(int argc, char** args)
{
    if(argc!=2)
    {
        printf("poprawne uzycie: ./rownania n, gdzie n to rozmiar kwadratowej macierzy\n");
        return 1;
    }
    int n, i, s, j;
    struct rusage t0, t1, t2;
    if((n=atoi(args[1])) <= 0)
    {
        printf("niepoprawny wymiar macierzy!\n");
        return 1;
    }
```

Następnie alokujemy macierze i je wypełniamy losowymi liczbami

```c
    gsl_matrix* a = gsl_matrix_alloc(n, n);
    gsl_matrix* a_kopia = gsl_matrix_alloc(n, n);
    gsl_vector* b = gsl_vector_alloc(n);
    gsl_vector* b_wymnoz = gsl_vector_alloc(n);
    gsl_vector* x = gsl_vector_alloc(n);
    gsl_permutation* p = gsl_permutation_alloc(n);

    srand(time(0));
        for(i=0; i<n; i++) // wypelniam macierz A losowymi liczbami
        for(j=0; j<n; j++)
        {
            gsl_matrix_set(a, i, j, rand_double());
            gsl_matrix_set(a_kopia, i, j, gsl_matrix_get(a, i, j)); // kopia potrzebna do koncowego
                                                                    // sprawdzenia, bo dekompozycja niszczy A
        }
    for(i=0; i<n; i++)
        gsl_vector_set(b, i, rand_double()); // wypelniam wektor b losowymi liczbami
```

Następnie implementujemy dekompozycje, obliczanie i sprawdzanie zgodności. Również implementujemy pomiary czasu:

- dekompozycji (t0-t1)

- obliczania (t1-t2)

```
    getrusage(RUSAGE_SELF, &t0);
    gsl_linalg_LU_decomp(a, p, &s); // dekompozycja LU i rozwiazanie ukladu rownan
    getrusage(RUSAGE_SELF, &t1);
    gsl_linalg_LU_solve(a, p, b, x);
    getrusage(RUSAGE_SELF, &t2);
    printf("Wektor x rozwiazan = \n"); // wypisuje wyliczony wektor x
    gsl_vector_fprintf (stdout, x, "%g");
    printf("\n");
    printf("Czas dekompozycji \n");
    czas(&t0,&t1);
    printf("Czas rozwiazania \n");
    czas(&t1,&t2);

    for(i=0; i<n; i++)
        gsl_vector_set(b_wymnoz, i, sprawdz(i, n, a_kopia, x)); // licze b = A*x

    printf("Zaalokowany wektor b = \n"); // sprawdzam zgodnosc
    gsl_vector_fprintf (stdout, b, "%g");
    printf("\n");

    printf("Wymnozony wektor b = \n");
    gsl_vector_fprintf (stdout, b_wymnoz, "%g");

    gsl_permutation_free(p);
    gsl_vector_free(x);
    gsl_vector_free(b_wymnoz);
    gsl_vector_free(b);
    gsl_matrix_free(a_kopia);
    gsl_matrix_free(a);
    return 0;
}
```

Uruchamiamy program dla n = 5

```
root@3e6b9c1405ca: /mownit/l
root@3e6b9c1405ca:/mownit/lab6# ./rownania 5
Wektor x rozwiazan =
1.32845
0.171026
-3.49042
-0.278128
-0.381699

Czas dekompozycji
total time: 0.000010
Czas rozwiazania
total time: 0.000005
Zaalokowany wektor b =
-0.671529
-1.94826
3.9132
-0.293509
0.728987

Wymnozony wektor b =
-0.671529
-1.94826
3.9132
-0.293509
0.728987
root@3e6b9c1405ca:/mownit/lab6#
```

Analogicznie wynik wygląda dla np. n = 10

```
root@3e6b9c1405ca: /mownit/l
root@3e6b9c1405ca:/mownit/lab6# ./rownania 10
Wektor x rozwiazan =
-1.80293
-0.138854
0.439788
-3.8726
-1.96051
-0.0394436
1.99762
3.60641
1.23741
-0.00313761

Czas dekompozycji
total time: 0.000027
Czas rozwiazania
total time: 0.000010
Zaalokowany wektor b =
2.05493
2.84106
-1.04178
0.212963
0.554431
-0.705177
2.01624
-0.612196
0.683859
-1.02897

Wymnozony wektor b =
2.05493
2.84106
-1.04178
0.212963
0.554431
-0.705177
2.01624
-0.612196
0.683859
-1.02897
root@3e6b9c1405ca:/mownit/lab6#
```

Co możemy zauważyć to bardzo małe czasy dekompozycji jak i rozwiązywania układu równań. Dla tak małych wartości jest to praktycznie natychmiastowy wynik

# Wykresy, tabele, wyniki liczbowe

| | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| dekompozycja | 0.000423 | 0.001485 | 0.004876 | 0.013164 | 0.027455 | 0.040337 | 0.064896 | 0.099678 | 0.148311 | 0.207023 |
| obliczanie | 0.000043 | 0.000048 | 0.000107 | 0.000189 | 0.000327 | 0.000425 | 0.000569 | 0.000776 | 0.00155 | 0.001189 |

Jak możemy zauważyć najbardziej rosnącą jest tutaj dekompozycja w zależności od n