Мiнiстерство освiти i науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського» Iнститут прикладного системного аналiзу

**Лабораторна робота № 3**

з курсу «Чисельнi методи 1»

з теми «Ітераційні методи розв’язання СЛАР»

Варіант № 16

Виконав студент 2 курсу групи КА-91

Панченко Єгор Станіславович

перевiрила старший викладач

Хоменко Ольга Володимирiвна

Київ-2021

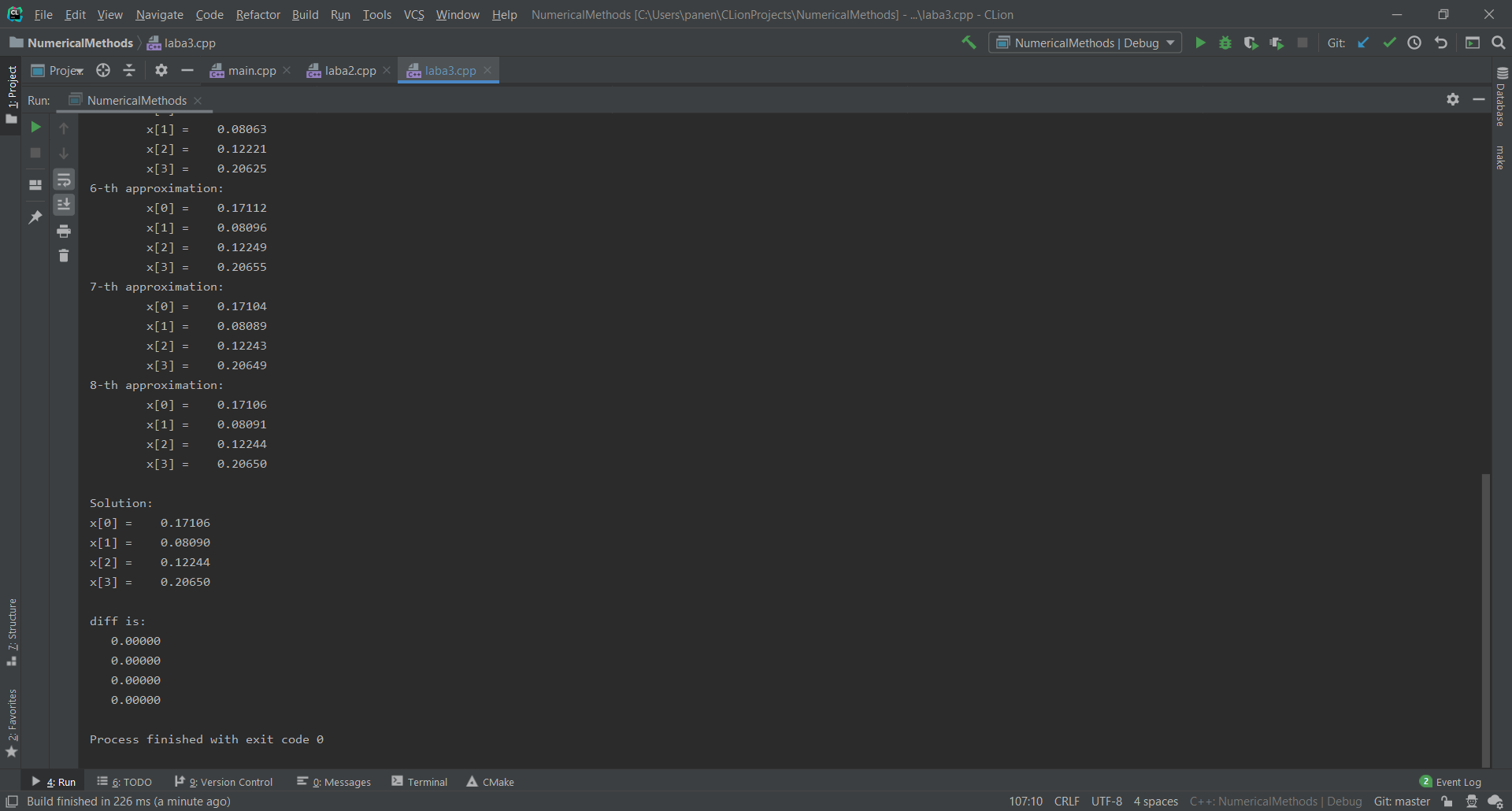
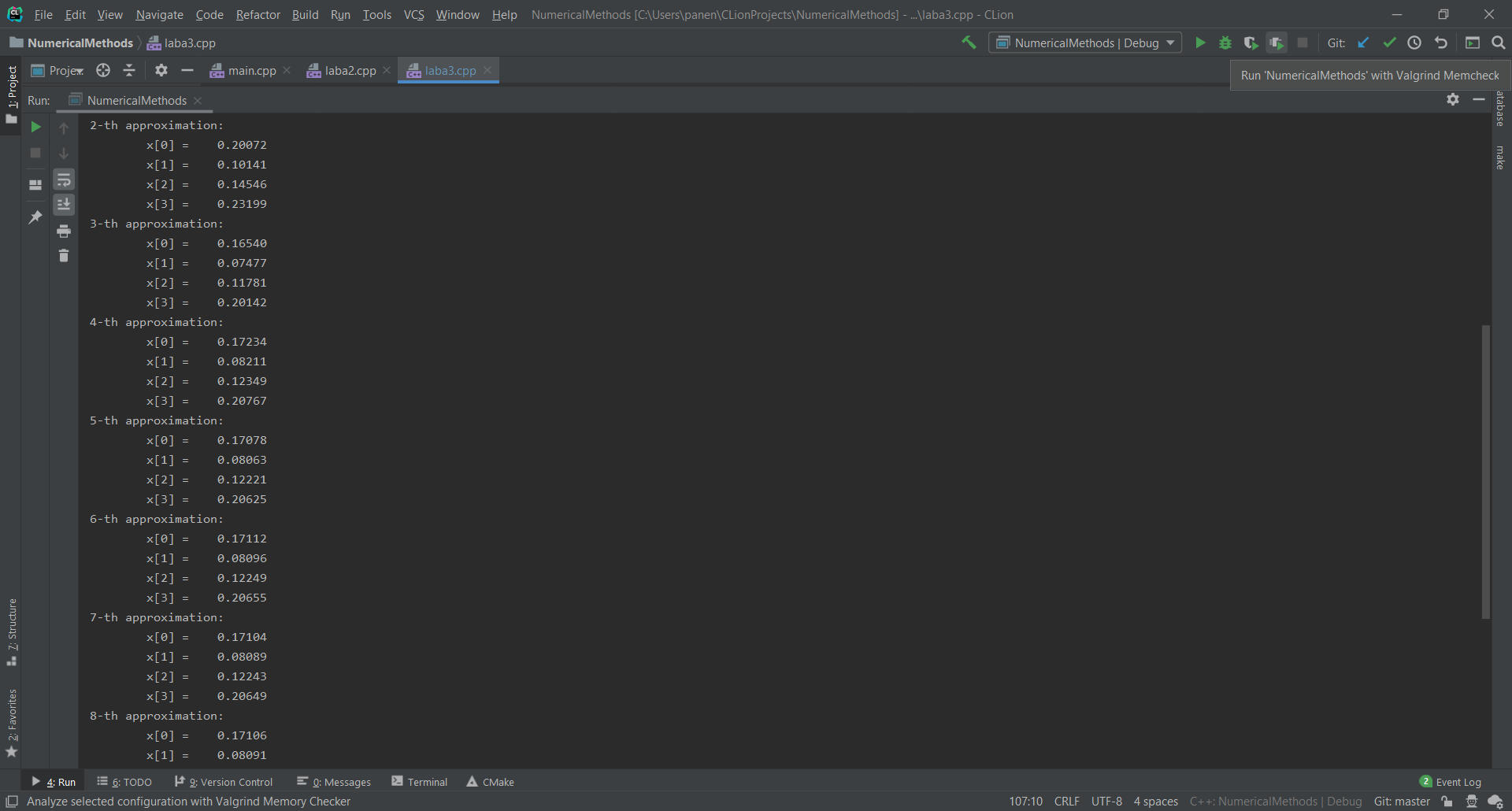
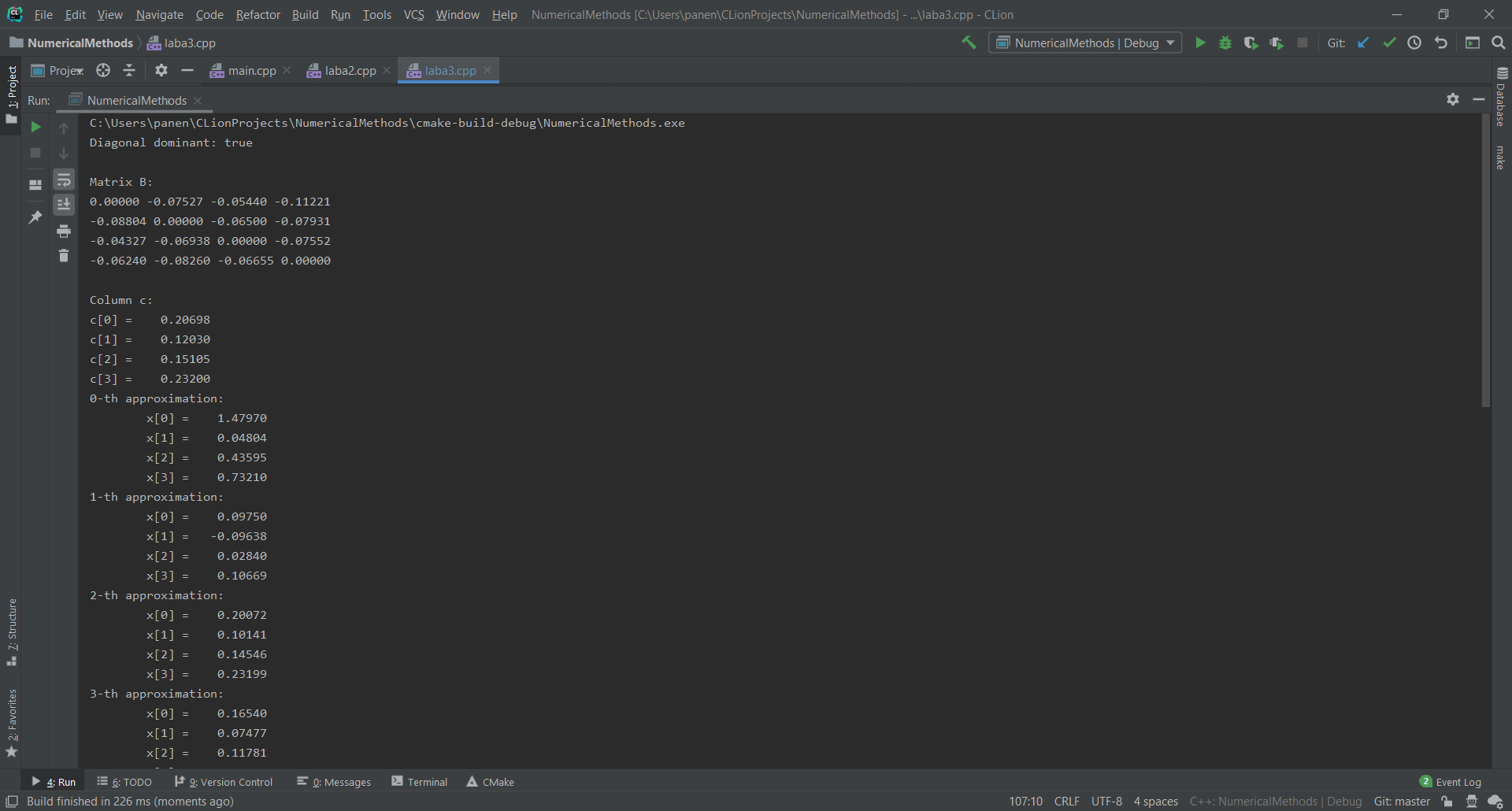
**Завдання: розв’язати СЛАР методом Якобі.**

1. **Текст програм**

#include <vector>  
  
using namespace std;  
  
namespace Laba3 {  
  
 const vector<vector<double> > A = {{2.923, 0.220, 0.159, 0.328},  
 {0.363, 4.123, 0.268, 0.327},  
 {0.169, 0.271, 3.906, 0.295},  
 {0.241, 0.319, 0.257, 3.862}};  
 const vector<double> B = {0.605, 0.496, 0.590, 0.896};  
 double eps = 1e-5;  
}  
  
using namespace Laba3;  
  
bool DiagonalDominant(vector< vector < double > > A) {  
 for (int i = 0; i < A.size(); i++) {  
 for (int j = 0; j < A.size(); j++) {  
 if (i == j) {  
 continue;  
 }  
 if (A[i][j] >= A[i][i]) {  
 return false;  
 }  
 }  
 }  
  
 return true;  
}  
  
void PrintMat(vector < vector < double > > A) {  
 for (int i = 0; i < A.size(); i++) {  
 for (auto j : A[i]) {  
 printf("%4.5lf ", j);  
 }  
 printf("\n");  
 }  
}  
  
void PrintRow(vector < double > B) {  
 for (auto i : B) {  
 printf("%4.5lf ", i);  
 }  
 printf("\n");  
}  
  
void SolveByYacob() {  
 bool dom = DiagonalDominant(Laba3::A);  
 printf("Diagonal dominant: %s\n", dom ? "true" : "false");  
 auto A\_it = Laba3::A;  
 auto B\_it = Laba3::B;  
 for (int i = 0; i < A\_it.size(); i++) {  
 for (int j = 0; j < A\_it.size(); j++) {  
 if (i == j) {  
 A\_it[i][j] = 0;  
 continue;  
 }  
 A\_it[i][j] = -Laba3::A[i][j] / Laba3::A[i][i];  
 }  
 B\_it[i] /= Laba3::A[i][i];  
 }  
 printf("\nMatrix B:\n");  
 PrintMat(A\_it);  
 printf("\nColumn c:\n");  
 for (int i = 0; i < B\_it.size(); i++) {  
 printf("c[%d] = %10.5lf\n", i, B\_it[i]);  
 //PrintRow(B\_it);  
 }  
  
 vector < vector < double > > x(0);  
 bool need\_random = true;  
 if (need\_random) {  
 srand(time(0));  
 vector < double > r(B\_it.size());  
 for (auto &i : r) {  
 i = (double)rand() / rand();  
 }  
 x.push\_back(r);  
 } else {  
 x.push\_back(B\_it);  
 }  
  
 for (;;) {  
 auto x\_it = Multy(A\_it, x.back());  
 for (int i = 0; i < x\_it.size(); i++) {  
 x\_it[i] += B\_it[i];  
 }  
  
  
 double maxdiff = 0;  
 for (int i = 0; i < x\_it.size(); i++) {  
 maxdiff = max(maxdiff, abs(x\_it[i] - x.back()[i]));  
 }  
  
 x.push\_back(x\_it);  
  
 if (maxdiff <= Laba3::eps) {  
 break;  
 }  
 }  
 printf("\nSolution:\n");  
 for (int i = 0; i < x.back().size(); i++) {  
 printf("x[%d] = %10.5lf\n", i, x.back()[i]);  
 //PrintRow(x.back());  
 }  
  
 auto diff = Substract(Laba3::B, Multy(Laba3::A, x.back()));  
  
 printf("\ndiff is:\n");  
 for (auto j : diff) {  
 printf("%10.5lf\n", j);  
 }  
}

1. **Результат роботи програм**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № ітерації |  |  |  |  |  |
| 0 | 1.47970 | 0.04804 | 0.43595 | 0.73210 | - |
| 1 | 0.09750 | -0.09638 | 0.02840 | 0.10669 | 1.382 |
| 2 | 0.20072 | 0.10141 | 0.14546 | 0.23199 | .19779 |
| 3 | 0.16540 | 0.07477 | 0.11781 | 0.20142 | .03532 |
| 4 | 0.17234 | 0.08211 | 0.12349 | 0.20767 | .00734 |
| 5 | 0.17078 | 0.08063 | 0.12221 | 0.20625 | .00156 |
| 6 | 0.17112 | 0.08096 | 0.12249 | 0.20655 | .00034 |
| 7 | 0.17104 | 0.08089 | 0.12243 | 0.20649 | .00008 |
| 8 | 0.17106 | 0.08091 | 0.12244 | 0.20650 | .00002 |
| допоміжні | 1.3822 | .144 | .408 | .625 |  |
| обчислення | .10322 | .19779 | .11706 | .1253 |  |
|  | .03532 | .02664 | .02765 | .03057 |  |
|  | .00694 | .00734 | .00568 | .00625 |  |
|  | .00156 | .00148 | .00128 | .00142 |  |
|  | .00034 | .00033 | .00028 | .0003 |  |
|  | .00008 | .00007 | .00006 | .00006 |  |
|  | .00002 | .00002 | .00001 | .00001 |  |



1. **Висновок**

У ході виконання лабораторної роботи було програмно реалізовано метод Якобі для будь-якої СЛАР розмірності , для якої виконується умова діагональної переваги. Перед початком роботи самого методу перевіряється умова діагональної переваги. Програма виводить шуканий вектор-розв’язок. Також вираховується похибка – наскільки відрізняється добуток початкової матриці і знайденого вектора від початкового вектора . За норму брався абсолютний поелементний максимум з різниці двох векторів. Програма виводить послідовність наближень. Початкове наближення можна задати двома способами – вручну або випадковими числами. В програмі реалізований саме випадковий варіант. При декількох запусках розв’язки не змінилися. Кількість наближень коливається в межах 10.