Мiнiстерство освiти i науки України

Національний технічний університет України

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

**Лабораторна робота № 4**

з курсу «Чисельнi методи 1»

з теми «Методи розв’язання нелінійних систем»

Варіант № 16

Виконав студент 2 курсу групи КА-91

Панченко Єгор Станіславович

перевiрила старший викладач

Хоменко Ольга Володимирiвна

Київ-2021

**Завдання: розв’язати систему методом простих ітерацій. Також розв’язати систему методом Ньютона.**

1. **Текст програм**

#include <vector>  
  
using namespace std;  
  
//x = 0.8 - cos(y + 0.5) = f(y)  
//y = -0.8 + 0.5 \* sin(x) = g(x)  
  
namespace Laba4 {  
  
 double eps = 1e-5;  
  
 double f(double y) {  
 return (0.8 - cos(y + 0.5));  
 }  
  
 double g(double x) {  
 return (-0.8 + 0.5 \* sin(x));  
 }  
  
 //sin(x+y)-1.4x = 0  
 //x^2+y^2 - 1 = 0  
 pair<double, double> Subtract(pair<double, double> a, pair<double, double> b) {  
 return make\_pair(a.first - b.first, a.second - b.second);  
 }  
  
};  
  
  
void SolveBySimpleIterationMethod() {  
 vector < pair<double,double> > x;  
 if (bool Correct = true; Correct == true) {  
 x.push\_back(make\_pair(-0.13, -0.86));  
 } else {  
 x.push\_back(make\_pair(10, 100));  
 }  
  
 for (;;) {  
 auto nextappr = make\_pair(Laba4::f(x.back().second), Laba4::g(x.back().first));  
 if (max(abs(nextappr.first - x.back().first), abs(nextappr.second - x.back().second)) < Laba4::eps) {  
 x.push\_back(nextappr);  
 break;  
 }  
 x.push\_back(nextappr);  
 }  
  
 for (int i = 0; i < x.size(); i++) {  
 printf("%d-th approximation:\n", i);  
 printf("\tx, y = %10.5lf, %10.5lf\n", x[i].first, x[i].second);  
 if (i > 0) {  
 printf("\tdiff = %10.5lf\n", max(abs(x[i].first - x[i - 1].first), abs(x[i].second - x[i - 1].second)));  
 }  
 }  
  
 printf("System translates into\n");  
 printf("\t%10.5lf = %10.5lf\n", x.back().first, Laba4::f(x.back().second));  
 printf("\t%10.5lf = %10.5lf\n", x.back().second, Laba4::g(x.back().first));  
}  
  
  
//Find inverse to 2 by 2 matrix  
vector < vector < double > > CalculateInverseAt(pair <double, double> p) {  
 double x = p.first;  
 double y = p.second;  
 vector < vector < double > > mat(2, vector < double > (2));  
 mat[0][0] = 2 \* y;  
 mat[0][1] = -cos(x + y);  
 mat[1][0] = -2 \* x;  
 mat[1][1] = cos(x + y) - 1.4;  
 double det = mat[0][0] \* mat[1][1] - mat[0][1] \* mat[1][0];  
 for (auto &i : mat) {  
 for (auto &j : i) {  
 j /= det;  
 }  
 }  
 return mat;  
}  
  
pair<double, double> F(pair<double, double> x) {  
 return make\_pair(sin(x.first + x.second) - 1.4 \* x.first, pow(x.first, 2) + pow(x.second, 2) - 1);  
}  
  
pair < double, double > Mult(vector < vector < double > > invW, pair<double, double> F) {  
 return make\_pair(invW[0][0] \* F.first + invW[0][1] \* F.second,  
 invW[1][0] \* F.first + invW[1][1] \* F.second);  
}  
  
void SolveByNewtonMethod() {  
 vector < pair<double, double> > x;  
 if (bool Correct = true; Correct == true) {  
 x.push\_back(make\_pair(0.7, 0.7));  
 } else {  
 x.push\_back(make\_pair(-5, 10));  
 }  
  
 for (;;) {  
 auto nextappr = Laba4::Subtract(x.back(), Mult(CalculateInverseAt(\*x.begin()), F(x.back())));  
 double diff = max(abs(nextappr.first - x.back().first), abs(nextappr.second - x.back().second));  
 if (diff < Laba4::eps) {  
 x.push\_back(nextappr);  
 break;  
 }  
 x.push\_back(nextappr);  
 }  
  
 for (int i = 0; i < x.size(); i++) {  
 printf("%d-th approximation:\n", i);  
 printf("\tx, y = %10.5lf, %10.5lf\n", x[i].first, x[i].second);  
 if (i > 0) {  
 printf("\tdiff = %10.5lf\n", max(abs(x[i].first - x[i - 1].first), abs(x[i].second - x[i - 1].second)));  
 }  
 }  
  
 printf("System translates into\n");  
 printf("\t%10.5lf = %10.5lf\n", sin(x.back().first + x.back().second) - 1.4 \* x.back().first, 0.0);  
 printf("\t%10.5lf = %10.5lf\n", pow(x.back().first, 2) + pow(x.back().second, 2) - 1, 0.0);  
  
}

1. **Результат роботи програм для методу простих ітерацій**

По-перше, доведемо збіжність. Матриця Якобі матиме вигляд . Оскільки сума абсолютних значень другого рядка не перевищує , а отже і 1. Отже, збіжність до розв’язку є.

|  |  |  |  |
| --- | --- | --- | --- |
| № ітерації |  |  |  |
| 0 | -0.13 | -0.86 | - |
| 1 | -0.1359 | -0.86482 | 0.00590 |
| 2 | -0.13419 | -0.86774 | 0.00292 |
| 3 | -0.13314 | -0.86689 | 0.00105 |
| 4 | -0.13345 | -0.86637 | 0.00052 |
| 5 | -0.13363 | -0.86653 | 0.00019 |
| 6 | -0.13358 | -0.86662 | 0.00009 |
| 7 | -0.13355 | -0.86659 | 0.00003 |
| 8 | -0.13355 | -0.86657 | 0.00002 |
| 9 | -0.13356 | -0.86658 | 0.00001 |

1. **Результати роботи програми для методу Ньютона**

|  |  |  |  |
| --- | --- | --- | --- |
| № ітерації |  |  |  |
| 0 | 0.7 | 0.7 | - |
| 1 | 0.70563 | 0.70866 | 0.00866 |
| 2 | 0.70555 | 0.70866 | 0.00008 |
| 3 | 0.70555 | 0.70866 | 0.00000 |

1. **Вивід програми**
2. **Висновок**

У ході виконання лабораторної роботи було програмно реалізовано метод простих ітерацій та спрощений метод Ньютона для розв’язку заданих систем. Для методу простих ітерацій було перевірено умови збіжності. Було зроблено висновок, що послідовність ітерацій збігається до розв’язку системи. Це можна перевірити емпіричним шляхом, якщо задавати різні початкові наближення. Для спрощеного методу Ньютона глобальної збіжності як такої немає. Це можна перевірити, задавши дуже далекі від розв’язку початкові наближення.