Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра математической кибернетики

Лабораторная работа № 2

по курсу «Численные методы»

Тема: решение нелинейных уравнений и систем нелинейных уравнений.

Студент: Мукин Ю. Д.
Группа: 80-304Б-18
Преподаватель: Гидаспов В.Ю.

Москва, 2021

Задание 1

1) Постановка задачи: Реализовать методы простой итерации и Ньютона решения нелинейных уравнений в виде программ, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения найти положительный корень нелинейного уравнения (начальное приближение определить графически). Проанализировать зависимость погрешности вычислений от количества итераций.

Вариант 12: $3^x - 5x^2 + 1 = 0$

2.1) Теория (метод простой итерации):

1) Исходное уравнение заменяется эквивалентным уравнением с выделенным линейным членом

2) Выбирается начальное приближение и начиная с него строится последовательность . Если – непрерывная ф-ция, а – сходится, то решением уравнения будет

Теорема 2.3. Пусть функция $\varphi(x)$ определена и дифференцируема на отрезке [a,b]. Тогда если выполняются условия:

1) $\varphi(x) \in [a,b] \quad \forall x \in [a,b]$,

2) $\exists q : |\varphi'(x)| \le q < 1 \quad \forall x \in (a,b)$,

то уравнение (2.5) имеет и притом единственный на [a,b] корень $x^{(*)}$;

3) Условие окончания итерационного процесса можно записать так:

$$\frac{q}{|1-q|}|x^{(k+1)} - x^{(k)}| < \varepsilon$$

2.2) Теория (метод Ньютона):

1) Выбирается отрезок [a,b] такой, что f(a)f(b) < 0

2) Выбирается начальное приближение на [a,b] так, чтобы $f(x^0)f''(x^0) < 0$

3) Строится последовательность , которая, согласно теореме, будет сходится корню.

4) Условие окончания итерационного процесса можно записать так: $|x^{(k+1)} - x^{(k)}| < \varepsilon$

3) Локализация корней:

Графически методом было выяснено, что корни уравнения лежать на отрезке [3,5]

3) Результат работы программы:

```
enter precision: 0.01
solution obtained by simple iteration method
k=0 x=3.000000 fi_func(x)=3.444518
```

k=1 x=3.444518 fi_func(x)=3.701038
k=2 x=3.701038 fi_func(x)=3.833887
k=3 x=3.833887 fi_func(x)=3.899006
k=4 x=3.899006 fi_func(x)=3.930084
k=5 x=3.930084 fi_func(x)=3.944727
k=6 x=3.944727 fi_func(x)=3.951587
k=7 x=3.951587 fi_func(x)=3.954790
k=8 x=3.954790 fi_func(x)=3.956285
x=3.956285
solution obtained by Newton method
k=0 x=5.000000 f_func(x)=119.000000 f_derivative(x)=216.962786 f_func(x)/f_derivative(x)=0.548481
k=1 x=4.451519 f_func(x)=34.939077 f_derivative(x)=101.621314 f_func(x)/f_derivative(x)=0.343816
k=2 x=4.107702 f_func(x)=7.808140 f_derivative(x)=59.088112 f_func(x)/f_derivative(x)=0.132144
k=3 x=3.975558 f_func(x)=0.828620 f_derivative(x)=46.874327 f_func(x)/f_derivative(x)=0.017677
k=4 x=3.957881 f_func(x)=0.013212 f_derivative(x)=45.384920 f_func(x)/f_derivative(x)=0.000291
x=3.957590

4) Код программы:

```
double fi_func_v12(double x)
{
   return log(5*pow(x,2)-1)/log(3);
}

double f_func_v12(double x)
{
   return pow(3,x)-5*pow(x,2)+1;
}

double f_derivative_v12(double x)
{
   return pow(3,x)*log(3)-10*x;
}

double simple_iterations(double(*fi_func)(double a),double x0, double epsilon, double q)
{
   double prev_x;
   double x = x0;
   int k=0;
   do
   {
      printf("\nk=%d x=%lf fi_func(x)=%lf", k, x, fi_func(x));
      prev_x = x;
      x = fi_func(x);
      k++;
   }
   while((q/(1-q)) * fabs(x - prev_x) > epsilon && k<1000);
   return x;
}

double newton_method(double(*f_func)(double a),double(*f_derivative)(double a),double x0, double epsilon)
{
   double x = x0;
   double prev_x;
   int k=0;
   do
   {
         printf("\nk=%d  x=%lf  f_func(x)=%lf  f_derivative(x)=%lf  f_func(x)/f_derivative(x)=%lf",  k,  x,  f_func(x),
f_derivative(x), f_func(x)/f_derivative(x));
      prev_x = x;
      x-= f_func(x)/f_derivative(x);
```

```
        k++;
    }
    while(fabs(x-prev_x) >= epsilon && k<500);
    return x;
}

void equation_solution()
{
    double eps;
    printf("\nenter precision: ");
    scanf("%lf",&eps);
    printf("solution obtained by simple iteration method");
    printf("\nx=%lf", simple_iterations(fi_func_v12,3, eps, 0.826));//826));
    printf("\nsolution obtained by Newton method");
    printf("\nx=%lf", newton_method(f_func_v12, f_derivative_v12, 5, eps));
}
```

Задание 2

1) Постановка задачи: Реализовать методы простой итерации и Ньютона решения систем нелинейных уравнений в виде программного кода, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения решить систему нелинейных уравнений (при наличии нескольких решений найти то из них, в котором значения неизвестных являются положительными); начальное приближение определить графически. Проанализировать зависимость погрешности вычислений от количества итераций.

Вариант 12: $\begin{cases} x_1 - \cos(x_2) = 3 \\ x_2 - \sin(x_1) = 3 \end{cases}$

2.1) Теория (метод простой итерации):

1) Исходная система заменяется на эквивалентную систему вида

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n) \\ x_2 = \varphi_2(x_1, x_2, \dots, x_n) \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ x_n = \varphi_n(x_1, x_2, \dots, x_n) \end{cases}$$

2) Выбираем начальное приближение $X^0 = (x_1^0, x_2^0 \dots x_n^0)$ и строим последующие приближения по формулам:

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \\ x_2^{(k+1)} = \varphi_2(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ x_n^{(k+1)} = \varphi_n(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \end{cases}$$

3) Условие сходимости итерационного процесса: $\max_{x \in G} \|\varphi'(x)\| < q < 1$

4) Условие окончания итерационного процесса:

$$\frac{q}{|1-q|}\|x^{(k+1)}-x^{(k)}\|<\varepsilon\,;\ \|x^{(k+1)}-x^{(k)}\|=\max_i\|x_i^{(k+1)}-x_i^{(k)}\|$$

2.2) Теория (метод Ньютона):

Итерационная формула метода: $x^{(k+1)}=x^k-J^{-1}(x^x)f(x^k)$

$$\mathbf{J}(\mathbf{x})=\begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Условие окончания итерационного процесса можно записать так: $|x^{(k+1)}-x^{(k)}|<\varepsilon$

3) Локализация корней:

Графически методом было выяснено, что корни уравнения лежать на отрезке [2,4]

3) Результат работы программы:

enter precision: 0.01

10 iterations have been done

solution is obtained by the method of simple iterations

x1=2.207110, x2=3.798477

5 iterations have been done

solution is obtained by Newton method

x1=2.210444, x2=3.802306

4) Код программы:

```
void fi_system_v12(double x1, double x2, double val[])
{
   val[0] = 3 + cos(x2);
   val[1] = 3 + sin(x1);
}

void f_system_v12(double x1, double x2, double val[])
{
   val[0] = x1-cos(x2)-3;
   val[1] = x2-sin(x1)-3;
}

void f_system_dx1_v12(double x1, double x2, double val[])
{
   val[0] = 1;
   val[1] = -cos(x1);
}
```

```
void f_system_dx2_v12(double x1, double x2, double val[])
{
   val[0] = sin(x2);
   val[1] = 1;
}

void simple_iterations_fs(void(*fi)(double x1, double x2, double val[]), double x0[], double epsilon, double q)
{
   double *prevu_x = malloc(2*sizeof(double));
   int k = 0;
   do
   {
      free(prevu_x);
      memcpy(prevu_x, x0, 2*sizeof(double));
      fi(x0[0], x0[1], x0);
      k++;
   }
   while(sqrt(pow(x0[0]-prevu_x[0],2)+pow(x0[1]-prevu_x[1],2))>epsilon && k<150);
   free(prevu_x);
   printf("\n%d iterations have been done",k);
}

double newton_method_fs(void(*f)(double x1, double x2, double val[]),void(*fdx1)(double x1, double x2, double val[]),void(*fdx2)(double x1, double x2, double val[]), double x0[], double epsilon)
{
   double *prevu_x = malloc(2*sizeof(double));
   double *fm = malloc(2*sizeof(double));
   double *fdx1m = malloc(2*sizeof(double));
   double *fdx2m = malloc(2*sizeof(double));
   int k = 0;
   double e;
   do
   {
      free(prevu_x);
      memcpy(prevu_x, x0, 2*sizeof(double));
      f(prevu_x[0], prevu_x[1], fm);
      fdx1(prevu_x[0], prevu_x[1], fdx1m);
      fdx2(prevu_x[0], prevu_x[1], fdx2m);
      x0[0] -= (fm[0]*fdx2m[1]-fm[1]*fdx2m[0])/(fdx1m[0]*fdx2m[1]-fdx1m[1]*fdx2m[0]);
      x0[1] -= (fm[1]*fdx1m[0]-fdx1m[1]*fm[0])/(fdx1m[0]*fdx2m[1]-fdx1m[1]*fdx2m[0]);
      k++;
   }
   while(sqrt(pow(x0[0]-prevu_x[0],2)+pow(x0[1]-prevu_x[1],2))>epsilon && k<150);
   free(prevu_x); free(fm); free(fdx1m); free(fdx2m);
   printf("\n%d iterations have been done",k);
}

void solution_system_of_equations()
{
   double eps;
   printf("\nenter precision: ");
   scanf("%lf",&eps);
   double x[2]={PI,2};
   simple_iterations_fs(fi_system_v12, x, eps, 0.9);
   printf("\nsolution is obtained by the method of simple iterations\nx1=%lf, x2=%lf", x[0], x[1]);//3.802 2.21
   x[0] = 4; x[1] = 2.3464;
   newton_method_fs(f_system_v12, f_system_dx1_v12, f_system_dx2_v12, x, eps);
   printf("\nsolution is obtained by Newton method\nx1=%lf, x2=%lf", x[0], x[1]);//3.802 2.21
}
```