«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет електроніки

Кафедра мікроелектроніки

Лабораторна робота №9

Варіант №21

Виконав: студент групи ДП-82
Мнацаканов Антон
Перевірив: Домбругов М.Р.

Київ-2020

Мета роботи: вивчення алгоритмів та налагодження програм для розв’язання систем лінійних алгебраїчних рівнянь ітераційними методами Якобі та Гауса-Зейделя.

Що зробити: з’ясувати факт збіжності чи розбіжності ітераційних процесів Якобі та Гауса-Зейделя. У випадку збіжності знайти розв’язок СЛАР та перевірити його, підставляючи в СЛАР отримані розв’язки і обраховуючи нев’язки. Визначити порядок збіжності ітераційного процесу.

Визначити порядок збіжності ітераційного процесу.

Фрагмент коду на С:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int n; // порядок матрици
double** slar;// СЛАР
double* r;
double* result;
double determinant;
double i;
double eps = 0.00000001;

int read(const char* filename)
{
 FILE *myfile = fopen (filename, "r");
 if(!myfile)
 {
    printf("ERROR \n");
    return 0;
 }
  fscanf(myfile,"%d", &n);
  slar=malloc(n*sizeof(double*));
  for(int i=0; i<n; ++i)
  {
   slar[i]=malloc(n*sizeof(double));
  }
  r=malloc(n*sizeof(double));
  result=malloc(n*sizeof(double));

  for(int i=0; i<n; ++i)
  {
     for(int j=0; j<n; ++j)
     {
        fscanf(myfile,"%lf", &slar[i][j]);
     }
  }
  for(int j=0; j<n; ++j)
```

```
    {
       fscanf(myfile,"%lf", &r[j]);
    }
    fclose(myfile);
return 1;
}
```

```
int scan()
{
  printf("n=");
  scanf("%d", &n);

  slar=malloc(n*sizeof(double*));
  for(int i=0; i<n; ++i)
  {
   slar[i]=malloc(n*sizeof(double));
  }
  r=malloc(n*sizeof(double));
  result=malloc(n*sizeof(double));

  for(int i=0; i<n; ++i)
  {
     for(int j=0; j<n; ++j)
     {
        printf("СЛАР[%i][%i]=", i+1, j+1);
        scanf("%lf", &slar[i][j]);
     }
  }
  for(int j=0; j<n; ++j)
  {
     printf("ПР[%i]=", j+1);
     scanf("%lf", &r[j]);
  }
  return 1;
}
```

```
int print()
{
  for(int i=0; i<n; ++i)
  {
     for(int j=0; j<n; ++j)
     {
        printf("%e ", slar[i][j]);
     }
     printf(" | %e\n", r[i]);
  }
  printf("\n");
return 1;
```

```
}

int print_result()
{
  for(int i =0; i<n; ++i)
  {
    printf("x[%i]=%e\n",i, result[i]);
  }
return 1;
}
```

```
int copy_vector(double* dest, double* source)
{
  for(int i=0; i<n; ++i)
  {
    dest[i]=source[i];
  }
  return 0;
}
```

```
double summ(int i)
{
  double res=0;
  for (int j=0; j<n; ++j)
  {
    if(j != i)
    {
      res += slar[i][j]*result[j];
    }

  }
  return res;

}
```

```
int jacobi()
{
  double* xnov;
  double delta_max;
  double delta;
  xnov=malloc(n*sizeof(double));
  copy_vector(result,r);
  int number = 0;
    do
    {

      delta_max=0;
```

```
    for(int i=0; i<n; ++i)
    {
       xnov[i]=(r[i]-summ(i))/slar[i][i];
       delta = fabs(xnov[i]-result[i]);
       if(delta>delta_max)
       {
        delta_max=delta;
       }
    }

    copy_vector(result, xnov);

    printf("%i\n",number++);
    print_result();

   }
   while(delta_max>=eps);

 return 0;
}
```

```
int Gaus_Seidel()
{
 double xnov;
 double delta_max;
 double delta;
 int number = 0;
 //xnov=malloc(n*sizeof(double));
 copy_vector(result,r);
   do
   {
    delta_max=0;
    for(int i=0; i<n; ++i)
    {
       xnov=(r[i]-summ(i))/slar[i][i];
       delta = fabs(xnov-result[i]);
       if(delta>delta_max)
       {
        delta_max=delta;
       }
       result[i]=xnov;
    }
printf("%i\n",number++);
    print_result();
   }
   while(delta_max>=eps);
```

```
  return 0;
}
   if (slar[i][i]==0)
   {
    printf("ERROR (МАТРИЦЯ ВИРОДЖЕНА)\n");
   return 0;
   }
//
_____________________________________________________________________________
    for(int k=i+1; k<n; ++k)
    {
     double p=slar[k][i]/slar[i][i];
       for(int j=i; j<n; ++j)
       {
          slar[k][j]=slar[k][j]-p*slar[i][j];
       }
   r[k]=r[k]-p*r[i];
   print();
   }
  }
  result[n-1]=r[n-1]/slar[n-1][n-1];
  for (int i = n-2; i>=0; --i)
  {
   double s=0;
   for(int j=i+1; j<n; ++j)
   {
    s=s+slar[i][j]*result[j];
   }
   result[i]=(r[i]-s)/slar[i][i];
  }
return 1;
}
//_____________________________________________________________________________
int main(int argc, char* argv[])
{
read(argv[1]);
print();
jacobi();
Gaus_Seidel();
print_result();

return 0;
}
```

```
antonmnacakanov@MacBook-Air exFAT % ./a.out lab9.txt
1.200000e+01 0.000000e+00 3.000000e+00 0.000000e+00  | 1.500000e+01
0.000000e+00 9.000000e+00 0.000000e+00 3.000000e+00  | 1.000000e+01
3.000000e+00 0.000000e+00 3.000000e+00 0.000000e+00  | 5.000000e+00
0.000000e+00 3.000000e+00 0.000000e+00 6.000000e+00  | 0.000000e+00

x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
```

21	В	$a = 3\,,\quad b = 3$

В) $\begin{pmatrix} 4a & 0 & a & 0 \\ 0 & 3a & 0 & b \\ a & 0 & b & 0 \\ 0 & b & 0 & 2b \end{pmatrix} \mathbf{x} = \begin{pmatrix} 15 \\ 10 \\ 5 \\ 0 \end{pmatrix}$

Кількість ітерацій :

для методу Якобі

```
0
x[0]=0.000000e+00
x[1]=1.111111e+00
x[2]=-1.333333e+01
x[3]=-5.000000e+00
1
x[0]=4.583333e+00
x[1]=2.777778e+00
x[2]=1.666667e+00
x[3]=-5.555556e-01
2
x[0]=8.333333e-01
x[1]=1.296296e+00
x[2]=-2.916667e+00
x[3]=-1.388889e+00
3
x[0]=1.979167e+00
x[1]=1.574074e+00
x[2]=8.333333e-01
x[3]=-6.481481e-01
4
x[0]=1.041667e+00
x[1]=1.327160e+00
x[2]=-3.125000e-01
x[3]=-7.870370e-01
5
x[0]=1.328125e+00
x[1]=1.373457e+00
x[2]=6.250000e-01
x[3]=-6.635802e-01
6
x[0]=1.093750e+00
x[1]=1.332305e+00
x[2]=3.385417e-01
x[3]=-6.867284e-01
7
x[0]=1.165365e+00
x[1]=1.340021e+00
x[2]=5.729167e-01
x[3]=-6.661523e-01
8
x[0]=1.106771e+00
x[1]=1.333162e+00
x[2]=5.013021e-01
x[3]=-6.700103e-01
9
x[0]=1.124674e+00
x[1]=1.334448e+00
x[2]=5.598958e-01
x[3]=-6.665809e-01
10
x[0]=1.110026e+00
x[1]=1.333305e+00
x[2]=5.419922e-01
x[3]=-6.672239e-01
11
x[0]=1.114502e+00
x[1]=1.333519e+00
x[2]=5.566406e-01
x[3]=-6.666524e-01
```

```
12
x[0]=1.110840e+00
x[1]=1.333329e+00
x[2]=5.521647e-01
x[3]=-6.667595e-01
13
x[0]=1.111959e+00
x[1]=1.333364e+00
x[2]=5.558268e-01
x[3]=-6.666643e-01
14
x[0]=1.111043e+00
x[1]=1.333333e+00
x[2]=5.547078e-01
x[3]=-6.666821e-01
15
x[0]=1.111323e+00
x[1]=1.333338e+00
x[2]=5.556234e-01
x[3]=-6.666663e-01
16
x[0]=1.111094e+00
x[1]=1.333333e+00
x[2]=5.553436e-01
x[3]=-6.666692e-01
17
x[0]=1.111164e+00
x[1]=1.333334e+00
x[2]=5.555725e-01
x[3]=-6.666666e-01
18
x[0]=1.111107e+00
x[1]=1.333333e+00
x[2]=5.555026e-01
x[3]=-6.666671e-01
19
x[0]=1.111124e+00
x[1]=1.333333e+00
x[2]=5.555598e-01
x[3]=-6.666667e-01
20
x[0]=1.111110e+00
x[1]=1.333333e+00
x[2]=5.555423e-01
x[3]=-6.666667e-01
21
x[0]=1.111114e+00
x[1]=1.333333e+00
x[2]=5.555566e-01
x[3]=-6.666667e-01
22
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555522e-01
x[3]=-6.666667e-01
23
x[0]=1.111112e+00
x[1]=1.333333e+00
x[2]=5.555558e-01
x[3]=-6.666667e-01
```

```
24
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555547e-01
x[3]=-6.666667e-01
25
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
26
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555553e-01
x[3]=-6.666667e-01
27
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
28
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555555e-01
x[3]=-6.666667e-01
29
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
30
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555555e-01
x[3]=-6.666667e-01
31
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
32
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
```

для методу Гауса-Зейделя

```
0
x[0]=0.000000e+00
x[1]=1.111111e+00
x[2]=1.666667e+00
x[3]=-5.555556e-01
1
x[0]=8.333333e-01
x[1]=1.296296e+00
x[2]=8.333333e-01
x[3]=-6.481481e-01
2
x[0]=1.041667e+00
x[1]=1.327160e+00
x[2]=6.250000e-01
x[3]=-6.635802e-01
3
x[0]=1.093750e+00
x[1]=1.332305e+00
x[2]=5.729167e-01
x[3]=-6.661523e-01
4
x[0]=1.106771e+00
x[1]=1.333162e+00
x[2]=5.598958e-01
x[3]=-6.665809e-01
5
x[0]=1.110026e+00
x[1]=1.333305e+00
x[2]=5.566406e-01
x[3]=-6.666524e-01
6
x[0]=1.110840e+00
x[1]=1.333329e+00
x[2]=5.558268e-01
x[3]=-6.666643e-01
7
x[0]=1.111043e+00
x[1]=1.333333e+00
x[2]=5.556234e-01
x[3]=-6.666663e-01
8
x[0]=1.111094e+00
x[1]=1.333333e+00
x[2]=5.555725e-01
x[3]=-6.666666e-01
9
x[0]=1.111107e+00
x[1]=1.333333e+00
x[2]=5.555598e-01
x[3]=-6.666667e-01
10
x[0]=1.111110e+00
x[1]=1.333333e+00
x[2]=5.555566e-01
x[3]=-6.666667e-01
11
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555558e-01
x[3]=-6.666667e-01
```

```
12
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
13
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
14
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
15
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
x[0]=1.111111e+00
x[1]=1.333333e+00
x[2]=5.555556e-01
x[3]=-6.666667e-01
```

Висновок: програмно створив декілька функцій, а також файл з якого бралися всі початкові значення для розв’язання СЛАР методом Якобі та Гауса-Зейделя також перевірив їх, підставляючи в СЛАР отримані розв’язки. Вивів кількість ітерацій які знадобилися для знаходження розв’язків СЛАР для обох методів.