РЕШЕНИЕ СИСТЕМ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Метод ЯКОБИ

все что нужно знать о нем — это только то что это итерационный метод решения систем уравнений, да и по сути все…

теперь перейдем к алгоритму:

1) имеем вот такую систему:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases},$$

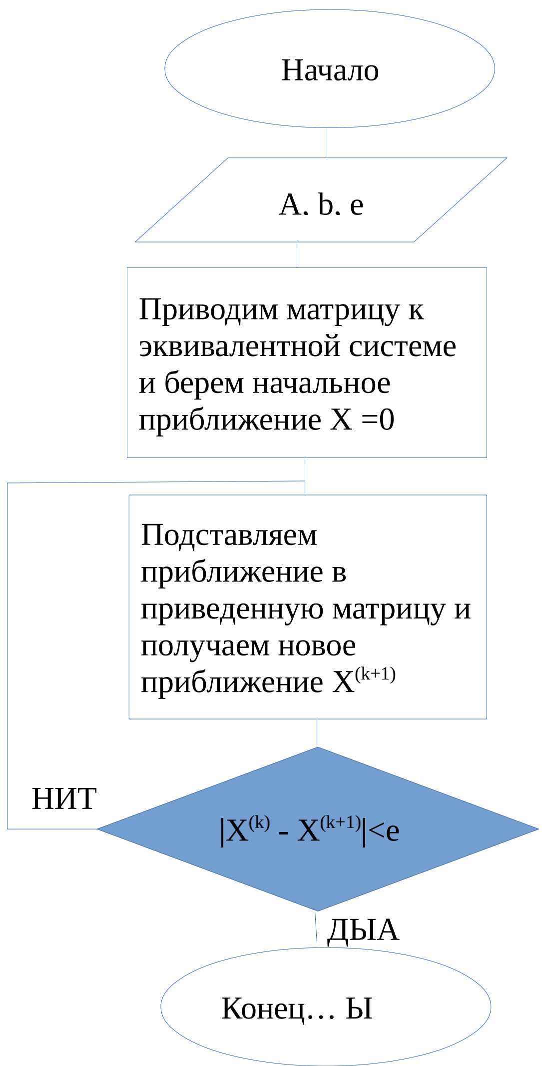
2) приводим ее к эквивалентному виду:

$$\begin{cases} x_1 = \dfrac{1}{a_{11}}(b_1 - a_{12}x_2 - \cdots - a_{1n}x_n) \\ x_2 = \dfrac{1}{a_{22}}(b_2 - a_{21}x_1 - \cdots - a_{2n}x_n) \\ \vdots \\ x_n = \dfrac{1}{a_{nn}}(b_n - a_{n1}x_1 - \cdots - a_{nn-1}x_{n-1}) \end{cases}.$$

3) теперь берем начальное приближение, принимая что все корни $X_n^{(0)} = 0$ и подставляя их в уравнение которое получили во 2 действии мы получаем следующее приближение — $X_n^{(1)}$, и так делаем до тех пор пока не выполнится условие $|X^{(k)} – X^{(k-1)}| < e$

схема и код:

```
function ans = Yacobi(A,b,e)

  it = 0;
  check = 10;
  n = size(A);

  X=zeros(n(1),1);

  while(check>e)
    it=it+1;
    %disp(it);
    X_old = X;
      for i = 1:1:n(1)
        sum = 0;
        for j = 1:1:n(2)
          if(i ~= j)
            sum = sum + A(i,j)*X_old(j);
          end
        end
        X(i) = (b(i) - sum) / A(i,i);
     end

      check  = abs(X(1)-X_old(1));
      for i = 1:1:n(1)
        if(abs(X(i)-X_old(i)) < check)
          check = (abs(X(i)-X_old(i)));
        end
      end
  end

  printf('coll iterations - %d\n',it);
  disp(X);

endfunction

```



Метод Зейделя

Все в точности так же как и я в Якоби, только подставляют не только предыдущее приближение но и уже новое.

То есть это выглядит так: мы находим приближение $X^{(k+1)}$ у 5 корня из 10 и вместо того чтобы в формулу для $X_5^{(k+1)}$ подставить только корни из перечня $X^{(k)}$, тут мы еще и подставляем уже найденное новое приближение, выглядит это так, с 1 по 4 иксы мы подставляем уже найденные $X^{(k+1)}$, а с 6 по 10 уже из $X^{(k)}$, это позволяет уменьшить кол-во итераций и время расчетов ну и точность конечно же

Код:

```
function ans = Zeidel(A,b,e)

  it = 0;
  check = 10;
  n = size(A);

  X=zeros(n(1),1);

  while(check>e)
    it=it+1;
    %disp(it);
    X_old = X;
        for i = 1:1:n(1)
            sum = 0;
            for j = 1:1:n(2)
                if(i ~= j)
                    sum = sum + A(i,j)*X(j);
                end
            end
            X(i) = (b(i) - sum) / A(i,i);
         end

        check  = abs(X(1)-X_old(1));
        for i = 1:1:n(1)
            if(abs(X(i)-X_old(i)) < check)
                check = (abs(X(i)-X_old(i)));
            end
        end
  end

  printf('coll iterations - %d\n',it);
  disp(X);

endfunction
```

Метод Гаусса

мне кажется особо ничего рассказывать тут не надо, просто приложу код:

```
function [ t ] = gauss( a,b )

  n=size(a);
  ab = [a b];

  if rank(a) ~= rank (ab)
    disp('error');
  end

  if rank(a)< n(1)
    disp(' inf of ansv');
  end

  x=zeros(n(1),1);

  for k=1:1:n(1)-1
    if a(k,k)==0
        l=k;
        while(1)%2.2.3
            l=l+1;
            if ((a(l,k)==0) && (l==n(1)));
                disp('The degenerate matrix')
            end
            if (a(l,k)~=0)
                break;
            end
        end%while
        c=b(k);
        b(k)=b(l);
        b(l)=c
        for j=1:1:n(1)
            c=a(k,j);
            a(k,j)=a(l,j);
            a(l,j)=c;
        end
        fprintf('%i & %i were exchange \n',k, l );
    end

    for i=k+1:1:n(1)
        m=a(i,k)/a(k,k);
        for j=1:1:n(1)
            a(i,j) = a(i,j) - m*a(k,j);
        end
        b(i)=b(i)-m*b(k);
    end

  end
  if (a(n(1),n(1))==0) %proverka 3.3
    disp('The degenerate matrix');
  end

  x(n(1))= b(n(1))/ a(n(1),n(1));
  for i=n(1)-1:-1:1
          s=0;

          for j=i+1:1:n(1)
            s=s+a(i,j)*x(j);
          end
          x(i)=(b(i)-s)/a(i,i);
  end

  t=x;
endfunction
```