Instituto Tecnológico de Cancún

METODOS NUMERICOS

CONEJO EROSA
JESUS GUSTAVO

UNIDAD 6 ACTIVIDADES INTEGRADORAS

UNIDAD 1 CODIGO EN JAVA METODO ITERATIVO

```
#include<iostream>
#include<cmath>
#include<iomanip>//biblioteca para poder utilizar la funcion setprecision(int)
using namespace std;
/*
M-->matriz aumentada de sistema lineal
n-->numero de ecuaciones
*/
/*
*modificar y luego saber si la matriz es daiagonalmente dominante
*si la matriz es diagonalmente dominante se asegura la convergencia del método
*/
bool convergencia(double M[100][100],int n){
    double may;//variable para almacenar el mayor de la columna k
    int ind;//indice del mayor-->indice de may
                                                    bool band=true;
    double aux;
    double acum;

    for(int k=0;k<n;k++){//recorrer columnas de la matriz reducida
        may=abs(M[k][k]);//se inicializa may con el primer elemento de la columna ind=k;
        //recorrer la columna k para buscar el indice del mayor
        for(int l=k+1;l<n;l++){
            if(may<abs(M[l][k])){
                may=abs(M[l][k]);
                ind=l;
            }

        }


        //cambiar filas
        if(k!=ind){//asegurar que no se trata de la misma fila
            for(int i=0;i<n+1;i++){
                aux=M[k][i];
                M[k][i]=M[ind][i];
                M[ind][i]=aux;
            }

        }
```

```
        //verificar convergencia
        acum=0;
        for(int j=0;j<n;j++){
            if(k!=j){
                acum=acum+abs(M[k][j]);
            }
        }
        if(acum>abs(M[k][k])){//no se trata de una matriz
diagonalmente dominante band=false;
            break;//termina el primer ciclo for
        }


}
    return band;
}


/*
M-->matriz aumentada modificada en la funcion convergencia(matriz
diagonalmente dominante)
V-->vector de la solución(inicializada con los punto iniciales de las
variables)
n-->numero de ecuaciones
tol-->tolerancia para encontrar la solucion
*/
int jacobi(double M[100][100],double V[100],int n,double tol){
    double error=0;
    double acum=0;
    double VA[100];//vector solucion de la iteracion anterior
    int iter=0;//número de iteraciones
    do{
        iter++;

        //recorrer diagonal de matriz disminuida
        for(int k=0;k<n;k++){
            acum=M[k][n];
            //recorrer la fila k de la matriz disminuida
            for(int j=0;j<n;j++){
                if(k!=j){
                    acum=acum-VA[j]*M[k][j];
                }

            }
            V[k]=acum/M[k][k];

        }
```

```
        //error
        acum=0;
        for(int i=0;i<n;i++){
            error=V[i]-VA[i];
            acum=acum+pow(error,2);
        }
        error=sqrt(acum);

        //preparando VA para la proxima iteracion
        for(int i=0;i<n;i++){
            VA[i]=V[i];
        }
    }while(error>tol);
    return iter;
} /*
//funcion que muestra todas las iteraciones
void jacobi(double M[100][100],double V[100],int n,double tol){
double error=0;
double acum=0;
double VA[100];//vector solucion de la iteracion anterior
int iter=0;//número de iteraciones
cout<<iter<<"\t";
for(int i=0;i<n;i++){
V[i]=0;
VA[i]=0;
cout<<VA[i]<<"\t";
}cout<<endl;
do{
iter++;
for(int k=0;k<n;k++){//recorrer diagonal de matriz disminuida
acum=M[k][n];
for(int j=0;j<n;j++){//recorrer la fila k
if(k!=j){
acum=acum-VA[j]*M[k][j];
}

}
V[k]=acum/M[k][k];

}

//error
acum=0;
for(int i=0;i<n;i++){
error=V[i]-VA[i];
acum=acum+pow(error,2);
```

```
}
error=sqrt(acum);
//preparando VA para la proxima iteracion
cout<<iter<<"\t";
for(int i=0;i<n;i++){
cout<<V[i]<<"\t";//mostrar vector solucion
VA[i]=V[i];
}
//mostrar error
cout<<error<<endl;
}while(error>tol);

}

*/
int main (int argc, char *argv[]) {
    double M[100][100];
    int n;
    bool band;
    double V[100];
    double tol;
    cout<<"ingrese el numero de ecuaciones:";
    cin>>n;
    cout<<"ingrese elementos de la matriz aumentada:"<<endl;
    for(int i=0;i<n;i++){
        cout<<"fila "<<i+1<<":"<<endl;
        for(int j=0;j<n+1;j++){
            cout<<"\tcolumna "<<j+1<<":"<<endl;
            cin>>M[i][j];
        }
    }
    cout<<"matriz aumentada:"<<endl;
    for(int i=0;i<n;i++){
        for(int j=0;j<n+1;j++){
            cout<<M[i][j]<<"\t";
        }cout<<endl;
    }
    band=convergencia(M,n);
    if(band){

        cout<<"se garantiza la convergencia"<<endl;
        cout<<"matriz diagonalmente dominante:"<<endl;
        for(int i=0;i<n;i++){
            for(int j=0;j<n+1;j++){
                cout<<M[i][j]<<"\t";
            }cout<<endl;
        }
```

```
        if(band){
            cout<<"ingrese                tolerancia:                ";
            cin>>tol;


            cout<<"-->iteraciones:          "<<jacobi(M,V,n,tol)<<endl;
            cout<<"la                solucion                es:"<<endl;
            for(int i=0;i<n;i++){
                cout<<"-->X"<<i+1<<"="<<setprecision(30)<<V[i]<<endl;
            }
        }



    }
    else
        cout<<"no  se  garantiza  convergencia-->la  matriz  no  es
diagonalmente                                        dominante"<<endl;



    return 0; }
```