1. Métodos Iterativos para sistemas de ecuaciones lineales

1.1. Jacoby

```
Lista generarSiguienteAproxJacoby(Matriz &A, Lista &B, Lista &X){
    int n = X.size();
    Lista res(n);
    for(int i = 0; i < n; i++){
        res[i] = (B[i] - sumatoria(A,B,X,i)) / A[i][i];
    }
    return res;
}

void Jacoby(Matriz &A, Lista &B, Lista &X, int n, Presicion presicion, string fi){
    string file = fi + ".csv";
    ofstream archivo(file);
    int i = 0;
    archivo<<"\"X\",";
    for(int j = 0; j < X.size(); j++){
        archivo<<"\"X"<<to_string(j)<<"\"";
        if(j != X.size()-1) archivo<<",";
        else archivo<<endl;
    }
    Lista X_anterior;
    do{
        i++;
        X_anterior = X;
        X = generarSiguienteAproxJacoby(A,B,X);
        archivo<<"\"i"<<to_string(i)<<"\""<<endl;
        archivo<<"\"X\",";
        escribirLista(X,archivo);
    }while(_ErrorAbsoluto(X,X_anterior,presicion) and i < n);
    archivo<<"\"Resultado actual\",";
    escribirLista(X,archivo);
    archivo.close();
}
```

1.2. Gauss-Seidel

```
void generarSiguienteAproxGaus(Matriz &A, Lista &B, Lista &X){
    int n = X.size();
    for(int i = 0; i < n; i++){
        X[i] = (B[i] - sumatoria(A,B,X,i)) / A[i][i];
    }
}

void Gaus(Matriz &A, Lista &B, Lista &X, int n, Presicion presicion, string fi){
    string file = fi + ".csv";
    ofstream archivo(file);
    int i = 0;
    archivo<<"\"X\",";
    for(int j = 0; j < X.size(); j++){
        archivo<<"\"X"<<to_string(j)<<"\"";
        if(j != X.size()-1) archivo<<",";
        else archivo<<endl;
    }
    Lista X_anterior = X;
    do{
        i++;
        X_anterior = X;
        generarSiguienteAproxGaus(A,B,X);
        archivo<<"\"i"<<to_string(i)<<"\""<<endl;
        archivo<<"\"X\",";
        escribirLista(X,archivo);
    }while(_ErrorAbsoluto(X,X_anterior,presicion) and i < n);
    archivo<<"\"Resultado actual\",";
    escribirLista(X,archivo);
    archivo.close();
}
```

2. Métodos Directos para sistemas de ecuaciones lineales

2.1. Gauss con pivoteo parcial

```
Matriz Gauss(Matriz A, Lista &B){
    A = matrizAumentada(A,B);
    int pibot = 0;
    while(pibot != B.size() - 1){
        int mayor = buscarMayor(pibot,B.size()-1,pibot,A);
        if (mayor != pibot) swap(A[pibot],A[mayor]);
        for(int i = pibot+1; i < B.size(); i++){
            Num d = A[i][pibot]/A[pibot][pibot];
            for(int j = pibot; j < A[i].size(); j++){
                A[i][j] = A[i][j] - d*A[pibot][j];
            }
        }
        pibot++;
    }
    return A;
}
```

2.2. Descomposición PLU

```
tuple<Matriz,Matriz,Matriz> PLU(Matriz A){
    Matriz P = identidad(A.size());
    Matriz L = zeros(A.size());
    Matriz U = A;
    int pibot = 0;
    while(pibot != U.size() - 1){
        int mayor = buscarMayor(pibot,U.size()-1,pibot,U);
        if (mayor != pibot){
            swap(U[pibot],U[mayor]);
            swap(P[pibot],P[mayor]);
            swap(L[pibot],L[mayor]);
        }
        for(int i = pibot+1; i < U.size(); i++){
            Num d = U[i][pibot]/U[pibot][pibot];
            L[i][pibot] = d;
            for(int j = pibot; j < U[i].size(); j++){
                U[i][j] = U[i][j] - d*U[pibot][j];
            }
        }
        pibot++;
    }
    L = L + identidad(L.size());
    return make_tuple(P,L,U);
}
```

2.3. Sustituciones

```
Lista SustitucionRegresiva(Matriz A){ //Con Matriz Aumentada
    int j = 0;
    Lista res(A.size());
    for(int i = A.size() - 1; i >= 0; i--, j++){
        Num sum = 0;
        for(int k = 0; k < j; k++){
            sum += A[i][A[i].size() - 2 - k] * res[A[i].size() -2 -k];
        }
        res[A[i].size() -2-j] = (A[i][A[i].size()-1] - sum) / A[i][A[i].size()-2-j];

    }
    return res;
}

Lista SustitucionProgresiva(Matriz A){ //Con Matriz Aumentada
    int j = 0;
```

```
    Lista res(A.size());
    for(int i = 0; i < A.size(); i++, j++){
        Num sum = 0;
        for(int k = 0; k < j; k++){
            sum += A[i][k] * res[k];
        }
        res[j] = (A[i][A[i].size()-1] - sum) / A[i][j];
    }
    return res;
}
```

3. Métodos Iterativos para sistemas de ecuaciones no lineales

3.1. Punto fijo generalizado

```
Lista _PuntoFijo(ListaFunciones g, Lista r){
    Lista res;
    for(int i = 0; i < g.size(); i++){
        res.push_back(g[i](r));
    }
    return res;
}

void MetodoPuntoFijo(ListaFunciones g, Lista r, string fi, Num n, Num presicion){
    string file = fi + ".csv";
    ofstream archivo(file.c_str());
    int i = 0;
    archivo<<"R"<<i<<endl;
    escribirLista(r,archivo);
    Lista r_anterior;
    do{
        r_anterior = r;
        r = _PuntoFijo(g,r);
        i++;
        archivo<<"R"<<i<<endl;
        escribirLista(r,archivo);
    }while(!ErrorAbsoluto(r_anterior,r,presicion) and i < n);
    archivo<<"Resultado actual"<<endl;
    escribirLista(r,archivo);
    archivo.close();
}
```

3.2. Método de Newton-Raphson

```
Matriz JF(MatrizFunciones jf, Lista r){
    int n = jf.size();
    Matriz res = zeros(n);
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            res[i][j] = jf[i][j](r);
        }
    }
    return res;
}

void Newton(ListaFunciones f, MatrizFunciones jf, Lista r, string fi, Num n, Num presicion){
    string file = fi + ".csv";
    ofstream archivo(file.c_str());
    int i = 0;
    archivo<<"R"<<i<<endl;
    escribirLista(r,archivo);
    Lista r_anterior;
    do{
        r_anterior = r;
        r = r + MetodoPLU(JF(jf,r),F(f,r) * -1);
        i++;
        archivo<<"R"<<i<<endl;
        escribirLista(r,archivo);
    }while(!ErrorAbsoluto(r_anterior,r,presicion) and i < n);
    archivo<<"Resultado actual"<<endl;
```

```
    escribirLista(r,archivo);
    archivo.close();
}
```

3.3. Método de Newton-Raphson aproximado

```
Matriz JF(ListaFunciones f, Lista r, Num h){
    int n = f.size();
    Matriz res = zeros(n);
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            Lista temp = r;
            temp[j] += h;
            res[i][j] = (f[i](temp) - f[i](r)) / h;
        }
    }
    return res;
}

void NewtonAprox(ListaFunciones f, Lista r, string fi, Num n, Num presicion, Num h){
    string file = fi + ".csv";
    ofstream archivo(file.c_str());
    int i = 0;
    archivo<<"R"<<i<<endl;
    escribirLista(r,archivo);
    Lista r_anterior;
    do{
        r_anterior = r;
        r = r + MetodoPLU(JF(f,r,h),F(f,r) * -1);
        i++;
        archivo<<"R"<<i<<endl;
        escribirLista(r,archivo);
    }while(!ErrorAbsoluto(r_anterior,r,presicion) and i < n);
    archivo<<"Resultado actual"<<endl;
    escribirLista(r,archivo);
    archivo.close();
}
```