Homework 3: Análise de um circuito eléctrico

Fis. Computacional 2022-23 (P4)

Fernando Barão

31 de maio de 2023

Resolução de um sistema de equações lineares: determinação das correntes eléctricas de um circuito

Sistemas de equações lineares podem-se encontrar na resolução de problemas de estática de forças, na determinação dos pontos de equilíbrio de sistemas oscilantes ou por exemplo na determinação das correntes eléctricas nos diferentes ramos de um circuito eléctrico. Em qualquer dos casos, o problema pode ser equacionado através da seguinte equação matricial, $\mathbf{A} \cdot \mathbf{X} = \mathbf{b}$, onde:

> **A**, matriz de coeficientes $(n \times n)$ cuja dimensão n depende do número de incógnitas do problema
>
> **X**, vector solução (n)
>
> **b**, vector de constantes do problema (n)

O circuito que se segue possui oito resistências alimentadas por dois potenciais eléctricos (baterias), -10 V e $+10$ V.

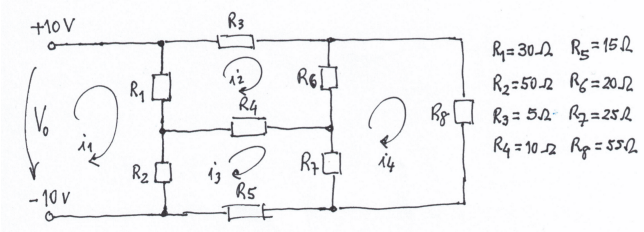

Figura 1: Circuito eléctrico

1. Estabeleça as equações do circuito usando as leis de Kirchoff, das malhas $\sum_i V_i = 0$ e dos nós $\sum_j I_j = 0$.

   solução

   A aplicação da lei das malhas, $\sum_k V_k = 0$, a cada uma das malhas do circuito, e do conhecimento da lei de Ohm, $V = Ri$, permite-nos obter o seguinte sistema de equações:

   $$
   \begin{aligned}
   &-V_0 + R_1(i_1 - i_2) + R_2(i_1 - i_3) = 0 \\
   &R_1(i_2 - i_1) + R_3 i_2 + R_6(i_2 - i_4) + R_4(i_2 - i_3) = 0 \\
   &R_2(i_3 - i_1) + R_4(i_3 - i_2) + R_7(i_3 - i_4) + R_5 i_3 = 0 \\
   &R_6(i_4 - i_2) + R_8 i_4 + R_7(i_4 - i_3) = 0
   \end{aligned}
   $$

2. Escreva as equações em termos matriciais, identificando a matriz **A** e os vectores **X** e **b**.

3. Para resolução do sistema de equações, implemente as seguintes classes (vá criando os métodos à medida que necessita):

- implemente a classe **FCmatrixAlgo**, onde os métodos (algoritmos) de redução das matrizes sejam construídos usando a biblioteca Eigen

```
#include <Eigen/Core>

class FCmatrixAlgo {
  public:
  FCmatrixAlgo() = default;
  ~FCmatrixAlgo() = default;

  /*
  Implements Gauss elimination
  */
  static void GaussElimination(
              Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic>&,
↪  // matrix coeffs
              Eigen::Matrix<double,Eigen::Dynamic,1>&        //
↪  vector of constants
              ); //no pivoting
  static void GaussEliminationPivot(
              Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic>&,
↪  // matrix coeff
              Eigen::Matrix<double,Eigen::Dynamic,1>&,       //
↪  vector of constants
              Eigen::Matrix<double,Eigen::Dynamic,1>&        // row
↪  order indexing
                ); //make pivoting

  /*
  Implements LU decomposition (Doolitle)
  */
  static void LUdecomposition(
              Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic>&,
↪  // matrix coeff
          Eigen::Matrix<int,Eigen::Dynamic,1>&, // row order indexing
          bool bpivot=false // activate pivoting
             );

}
```

- implemente a classe **EqSolver** onde se procede à resolução do sistema de equações

```
#include <Eigen/Dense>

class EqSolver {
  public:
  // constructors and destructor
  EqSolver();
  EqSolver(
           const
           ↪  Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic>&,
           ↪  // matrix coeffs
           const Eigen::Matrix<double,Eigen::Dynamic,1>& // vector of
           ↪  constants
      );
  ~EqSolver() = default;

  // output (optional)
  friend ostream& operator<<(ostream&, const EqSolver&);

  // solvers
  const Eigen::Matrix<double,Eigen::Dynamic,1>& GaussSolver(bool
   ↪  pivot=false);
  const Eigen::Matrix<double,Eigen::Dynamic,1>& LUSolver(bool
   ↪  pivot=false);
  void IterativeJacobiSolver(
       Eigen::Matrix<double,Eigen::Dynamic,1>&, // starting solution
       int& itmax,  //nb of max iterations
       double tol=1.E-3); // tolerance on convergence
  void IterativeGaussSeidelSolver(
       Eigen::Matrix<double,Eigen::Dynamic,1>&,
       int& itmax,
       double tol=1.E-3);

  private:
  Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic> M; //
 ↪  coefficients matrix
  Eigen::Matrix<double,Eigen::Dynamic,1> b; // constants vector
};
```

4. Resolva o sistema utilizando os diferentes métodos: Gauss, LU e iterativo.

> Exemplo sumário do programa principal **tEqSolver.C** onde se procede à resolução do sistema de equações.

```
#include "FCmatrixAlgo.h"
#include "EqSolver.h"
#include <iostream>

int main() {

  // define matrix equations: A.X=b

  (..) // create matrix A and vector of constants b

  // solution with Gauss (no pivoting)

  EqSolver S(A,b);
  auto X = S.GaussSolver();

  std::cout << "Gauss solution: \n" << X << std::endl;

}
```