

C++ code 2.5.12: Constructor of **ImpedanceMap**

```
2 ImpedanceMap::ImpedanceMap(double R, double V)
3     // TODO: (2-5.g) initializer list.
4     // START
5     : R_(R),
6       V_(V)
7 // END
8 {
9     // TODO: (2-5.g) Build the matrix and the r.h.s. vector by either
10    using
11    // the topology or by hardcoding the matrix and vector. Also, do
12    // the setup phase for the solver here or find an even smarter way
13    // for an efficient implementation
14    // START
15
16    // Build the topology with source voltage V_.
17    Topology top(V_);
18
19    // Automatically build A0 filling from topology
20    // A0 is the matrix  $A_R$  (i.e. with  $R_x = R$ )
21    Eigen::MatrixXd A0 = Eigen::MatrixXd::Zero(nnodes_, nnodes_);
22    for (resistor& res : top.T) {
23        // Shift indices down by 1 (indices in Figure start from 1)
24        std::size_t i = res.first - 1;
25        std::size_t j = res.second - 1;
26        // Fill diagonal: each resistance contributes += R
27        // into the diagonal of both nodes
28        A0(i, i) += 1;
29        A0(j, j) += 1;
30        // Fill the off diagonal (negative part in  $\Delta V_{ij}$ ,
31        // the matrix is kept symmetric
32        A0(i, j) -= 1;
33        A0(j, i) -= 1;
34    }
35
36    // Fill in the rest (source and ground), i.e.
37    // components with  $\Delta V_{ij}$  with  $j > 15$ .
38    // Each node  $i$  connected to ground or source contributes
39    // to the r.h.s  $b$  with  $R \cdot V$ 
40    // ( $R$  is the resistance between node  $i$  and ground/source
41    // node,  $V$  is voltage at sink or source)
42    // and to its own diagonal with  $R$ 
43    Eigen::VectorXd b = Eigen::VectorXd::Zero(nnodes_);
44    for (voltage& volt : top.S) {
45        // Shift index down by 1 and get voltage in  $V_2$ 
46        int i = std::get<0>(volt) - 1;
47        double source_voltage = std::get<2>(volt);
48        // Add voltage to r.h.s. (resistance assumed to be  $R$ )
49        b(i) += source_voltage;
50        // Add resistance to matrix diagonal: contribution of source
51        // current
52        // scaled by  $R$ .
53        A0(i, i) += 1;
54    }
55
56    // Following the "smart" approach for the SMW formula;
57    // we don't even have to store the LU-decomposition!
58    Eigen::PartialPivLU<Eigen::MatrixXd> lu(A0);
```