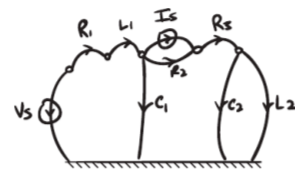


**imp.examat Impedance modeling example in Matlab**

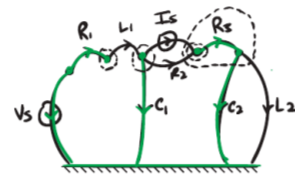
Example imp.examat-1

re: Impedance modeling with Matlab

1 Consider the linear graph of an electronic system, below. Use impedance methods to derive the transfer functions from inputs  $V_S$  and  $I_S$  to outputs  $v_{C_1}$  and  $i_{R_1}$ .



2 The normal tree is shown, below, along with contours to be used for continuity equations.



3 We switch over to Matlab for the remainder of the solution.

Let's define the required symbolic variables.

```
syms vS IS ...
vC1 iC1 vC2 iC2 vL1 iL1 vL2 iL2 ...
vR1 iR1 vR2 iR2 vR3 iR3 ...
sC1 sC2 sL1 sL2 sR1 sR2 sR3 ...
C1 C2 L1 L2 R1 R2 R3
```

We also specify the unknown variables (two for each passive element), output variables, and input variables.

```
unknowns = [ ...
vC1 iC1 vC2 iC2 vL1 iL1 vL2 iL2 ...
vR1 iR1 vR2 iR2 vR3 iR3 ...
];
out_i = [3,10]; % output indices
in = [18,19]; % input variables
```

Now let's define our elemental, continuity, and compatibility equations.

```
elemental = [ ...
vC1 == sC1*iC1,...
vC2 == sC2*iC2,...
vL1 == iL1*sL1,...
vL2 == iL2*sL2,...
vR1 == iR1*sR1,...
vR2 == iR2*sR2,...
vR3 == iR3*sR3 ...
];
continuity = [ ...
iC1 == iI1 - iR2,...
iR1 == iI1,...
iC2 == iS + iR2,...
iR3 == iS + iR2 ...
];
compatibility = [
vL1 == -vR1 + vS + vC1,...
vL2 == vC2,...
vR2 == vC1 - vC2 - vR3...
];
```

These form a linear system of  $2 \times 7 = 14$  unknowns and 14 equations. Such systems can be defined in matrix form as  $M \cdot \text{unknowns} = b$ , where  $M$  are the coefficients of the unknowns, unknowns is the vector of unknowns, and  $b$  is the vector of terms that include the inputs. Matlab has the function `equationsToMatrix` for specifying the matrix form from a list of equations.

```
[M,b] = equationsToMatrix(...
[elemental,continuity,compatibility],... % eq's
unknowns ... % unknown variables
);
disp('first 10 columns of M:') % to fit on screen
disp(M(1:10))
disp('b transposed:') % for pretty
disp(b.')
```

```
first 10 columns of M:
[ 1, -sC1, 0, 0, 0, 0, 0, 0, 0, 0]
[ 0, 0, 1, -sC2, 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 1, -sL1, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 1, -sL2, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0, 0, 1, -sR1]
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[ 0, 1, 0, 0, 0, 0, -1, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, -1, 0, 0, 0]
[ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[ 1, 0, 0, 0, 1, 0, 0, 0, 0, 1]
[ 0, 0, -1, 0, 0, 0, 0, 1, 0, 0]
[ -1, 0, 1, 0, 0, 0, 0, 0, 0, 0]
```

$Ax = b$

Furthermore, Matlab has the function `linsolve` for solving for the unknowns.

```
sol_s = linsolve(M,b);
```

This solution `sol_s` includes impedances. We would like to substitute for the actual impedance values, defined as follows in struct form.

```
impedances.sC1 = 1/(C1*s);
impedances.sC2 = 1/(C2*s);
impedances.sL1 = L1*s;
impedances.sL2 = L2*s;
impedances.sR1 = R1;
impedances.sR2 = R2;
impedances.sR3 = R3;
```

Now we can substitute impedances with subs, which gives us a solution in terms of  $s$ .

```
sol = simplify(...
subs(...
sol_s,...
fieldnames(impedances),...
struct2cell(impedances)...
)...
);
[n,d]=numden(sol);
sol_nd = collect(n,d,s);
```

Finally, we can compute the transfer function matrix  $H(s)$  by using the solutions for our outputs and substituting 1 for the input of interest and 0 for the others (this code generalizes to more than two inputs).

```
for input_i = 1:length(in) % each input
for output_i = 1:length(out_i) % each output
output_index = out_i(output_i); % id of o/p var
input_var = in(input_i); % input variable
other_inputs = setdiff(in,[input_var]); % sneaky
num = sol_nd(output_index,i); % w/all 's'
den = sol_nd(output_index,2); % w/all 's'
num_tf = subs(... % eliminate other inputs
num,...
[input_var,other_inputs],...
[1,zeros(size(other_inputs))],...
);
den_tf = subs(... % eliminate other inputs
den,...
[input_var,other_inputs],...
[1,zeros(size(other_inputs))],...
);
H(input_i,output_i) = ... % collect s and divide
collect(num_tf,s)/collect(den_tf,s);
end
end
pretty(H(2,1)) % display H_21
```

```
2
(C1 L1 R2 s + C1 R1 R2 s + R2)/((C1 C2 L1 R2 + C1
C2 L1 R3)
3
s + (C1 L1 + C2 L1 + C1 C2 R1 R2 + C1 C2 R1 R3)
2
s + (C1 R1 + C2 R1 + C2 R2 + C2 R3) s + 1
```