



# Vrije Universiteit Brussel

Faculteit der Toegepaste Wetenschappen  
Departement ELEC  
Pleinlaan 2, 1050 Brussel, Belgium

## A Spice circuit can be synthesised with a specified set of S-parameters

Eli Steenput

Promotoren:  
Prof. Dr. ir. Y. Rolain  
Prof. Dr. ir. J. Schoukens

Bijstelling horend tot het proefschrift  
ingediend tot het behalen van de  
academische graad van doctor in de  
Toegepaste Wetenschappen

Oktober 1999

# A Spice circuit can be synthesised with a specified set of S-parameters

Eli Steenput  
Dept ELEC  
Vrije Universiteit Brussel  
Pleinlaan 2  
1050 Brussel  
Belgium  
esteenpu@vub.ac.be

**ABSTRACT:** Circuit parameter models under rational form, obtained through an identification process for Linear Time Invariant systems, are not readily usable in a Spice simulation. This paper proposes a method to obtain a Spice circuit, built from standard Spice components, that exhibits the same behaviour as the estimated model. The applicability of the method is demonstrated on several models identified from real measurements.

## 1. The problem

Current powerful model parameter identification tools offer serious competition to traditional white-box physical modelling solutions for LTI systems at RF frequencies [1]. One sometimes stated disadvantage of an estimated black-box model with regard to a white-box model is that an equivalent circuit for use in a simulation program (such as Spice) is not readily available. For example, ICAP/4 SpiceNet from Intusoft offers “function blocks”, components whose behaviour in the Laplace domain can be described by a polynomial, but the maximum order of the polynomial is limited to 2.

This paper proposes a method to synthesise a Spice circuit with a prescribed set of S-parameters in the Laplace domain. The resulting circuit must have the same S-parameters as the model. The circuit can be generated automatically, and uses only standard Spice components.

The synthesis of a two-port network with all 4 S-parameters specified (or any other 4 two-port parameters) is hardly mentioned in network synthesis literature. Most synthesis methods derived from the basic Cauer synthesis are concerned with the synthesis of a transfer function, rather than a particular two-port parameter matrix, and mostly realise only 2, sometimes 3 of the two-port parameters. The S-parameter models are not necessarily reciprocal or even stable, and generally cannot be implemented as passive networks.

Also, the realisability constraints placed on traditional synthesis methods do not apply here, since the Spice network does not have to be physically realised.

## 2. A proposed solution

A synthesis directly from the S-parameters is impossible. We choose to transform the S-parameters into Z-parameters. The corresponding Z matrix is derived from the S parameters using the following formulae [2]:

$$Z_{11} = Z_0 \frac{(1 + s_{11})(1 - s_{22}) + s_{12}s_{21}}{(1 - s_{11})(1 - s_{22}) - s_{12}s_{21}} \quad Z_{12} = Z_0 \frac{2s_{12}}{(1 - s_{11})(1 - s_{22}) - s_{12}s_{21}}$$

$$Z_{21} = Z_0 \frac{2s_{21}}{(1 - s_{11})(1 - s_{22}) - s_{12}s_{21}} \quad Z_{22} = Z_0 \frac{(1 - s_{11})(1 + s_{22}) + s_{12}s_{21}}{(1 - s_{11})(1 - s_{22}) - s_{12}s_{21}}$$

The next step consists of synthesising a network characterised by the Z-matrix:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

The basic principle of the proposed circuit is to synthesise  $Z_{11}$  and  $Z_{22}$  as impedances, and to synthesise  $Z_{12}I_2$  and  $Z_{21}I_1$  as voltage-controlled voltage sources (see Figure 1).

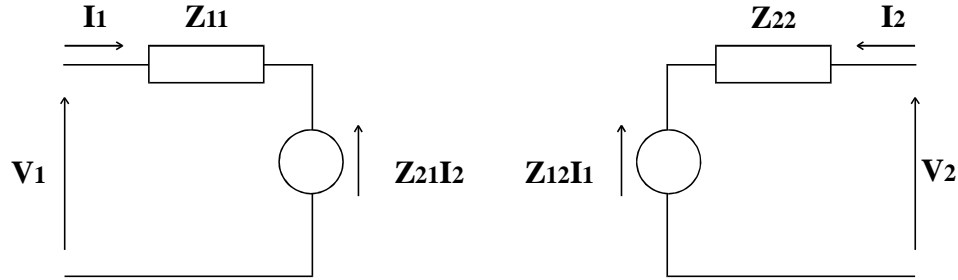


Figure 1 Circuit principle

The controlling voltages for  $Z_{12}I_2$  and  $Z_{21}I_1$  are obtained from separate circuits implementing  $Z_{12}$  and  $Z_{21}$  as impedance with a current-controlled current source. By using the ideal dependent sources available in Spice, each parameter can be implemented independently. The proposed Spice circuit structure is depicted in Figure 2. The components marked **F** are the current-controlled current sources, and the components marked **E** are the voltage-controlled voltage sources.

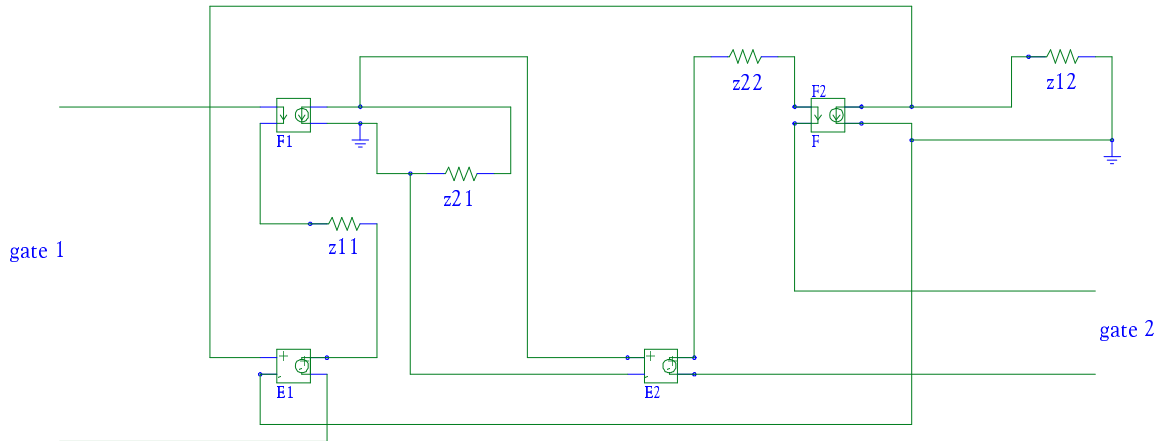


Figure 2 Proposed circuit structure

The corresponding Spice subcircuit is:

```
.SUBCKT ZMODEL 1 9 10 2
** Use as X??? gate1_node1 gate1_node2 gate2_node1 gate2_node2 ZMODEL **
F1 4 0 VF1 1
VF1 9 3 0V
F2 6 0 VF2 1
VF2 5 2 0V
X12 6 0 Z12
E2 7 10 0 4 1
X22 7 5 Z22
E1 8 1 6 0 1
X11 3 8 Z11
X21 0 4 Z21
.ENDS ZMODEL
```

With this approach, each Z-parameter is realised independently as a one-port. A similar approach using the Y-parameters or other parameters works as well, and will be preferable if the denominator of the Z-parameters obtained from the model is small.

The Y-parameters are derived from the S-parameters using the following formulae:

$$Y_{11} = Y_0 \frac{(1 - s_{11})(1 + s_{22}) + s_{12}s_{21}}{(1 + s_{11})(1 + s_{22}) - s_{12}s_{21}} \quad Y_{12} = Y_0 \frac{-2s_{12}}{(1 + s_{11})(1 + s_{22}) - s_{12}s_{21}}$$

$$Y_{21} = Y_0 \frac{-2s_{21}}{(1 + s_{11})(1 + s_{22}) - s_{12}s_{21}} \quad Y_{22} = Y_0 \frac{(1 + s_{11})(1 - s_{22}) + s_{12}s_{21}}{(1 + s_{11})(1 + s_{22}) - s_{12}s_{21}}$$

Figure 3 shows the network characterised by the Y-matrix:

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

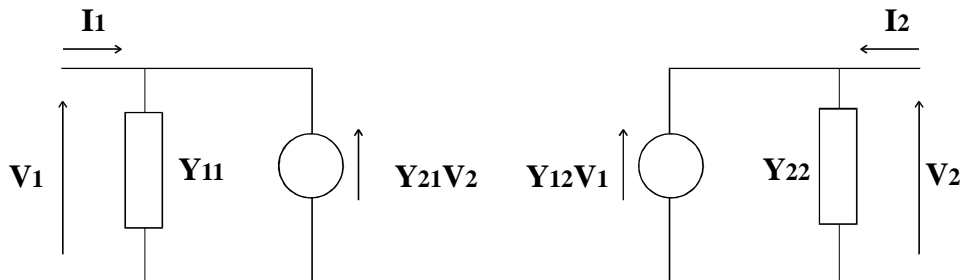


Figure 3 Alternative circuit principle

The corresponding Spice subcircuit is:

```
.SUBCKT YMODEL 1 7 2 8
** Use as X??? gate1_node1 gate1_node2 gate2_node1 gate2_node2 YMODEL **
X11 1 7 Y11
X22 2 8 Y22
F1 1 7 VF1 1
VF1 3 4 0V
E2 3 0 2 8 1
X21 0 4 Y21
E1 5 0 1 7 1
F2 2 8 VF2 1
VF2 5 6 0V
X12 6 0 Y12
.ENDS YMODEL
```

To implement the circuit from Figure 2, each Z-parameter is expanded in partial fractions, which for our models can be of several possible forms:

1. A constant  $k$
2. An infinity pole  $ks$
3. A real pole fraction  $\frac{r}{s-p}$
4. A pair of fractions with conjugate complex poles  $\frac{a+jb}{s-(\sigma+j\omega)} + \frac{a-jb}{s-(\sigma-j\omega)}$

Other forms may occur in rare cases.  $Z_{12}$  and  $Z_{21}$  in particular are not really impedance functions, so they possibly might, for example, have multiple poles at infinity. This possibility is not elaborated here. If required, such fractions could be implemented to at least order 5 by FDNR sections for example (or higher order using nested FDNR sections) [3].

### 3. Implementation

Each Z-parameter is implemented as a series of impedances, with each fraction or pair of fractions implemented as one impedance. Note that resistors, inductors and capacitors in the impedances can have negative values (since the purpose is Spice simulation, realisability is irrelevant).

- A constant  $k$  is implemented as a resistor with resistance value  $R = k(\Omega)$
- A pole at infinity  $ks$  is implemented as a series inductance of value  $L = k(H)$
- A real pole fraction  $\frac{r}{s-p}$  is implemented by an RC parallel circuit with the resistor

$$R = -\frac{r}{p}(\Omega) \text{ and the capacitor } C = \frac{1}{r}(F).$$

- A pair of fractions with conjugate complex poles  $\frac{a+jb}{s-(\sigma+j\omega)} + \frac{a-jb}{s-(\sigma-j\omega)}$  is

implemented as an LRC parallel resonance circuit (suggested in [4]). This circuit is depicted in Figure 4.

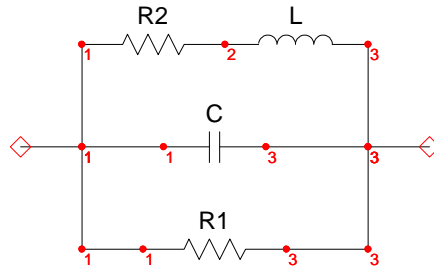


Figure 4 Complex pole pair realisation

It is relatively easy to derive that:

$$C = \frac{1}{2a}(F)$$

$$R_1 = \frac{-2a^2}{a\sigma - b\omega}(\Omega)$$

$$L = \frac{2aR_1}{R_1(\omega^2 + \sigma^2) + 2(a\sigma + b\omega)} (H) \quad R_2 = -L\left(\sigma + \frac{b\omega}{a}\right) (\Omega)$$

When using the Y-parameters, each fraction or pair of fractions can be implemented as a parallel admittance.

- A constant  $k$  is implemented as a resistor with resistance value  $R = \frac{1}{k} (\Omega)$
- A pole at infinity  $ks$  is implemented as a capacitance of value  $C = k(F)$
- A real pole fraction  $\frac{r}{s-p}$  is implemented by an RL series circuit with the inductor  $L = \frac{1}{r} (H)$  and the resistor  $R = -\frac{p}{r} (\Omega)$ .
- A pair of fractions with conjugate complex poles is implemented as an LRC series resonance circuit as depicted in Figure 5:

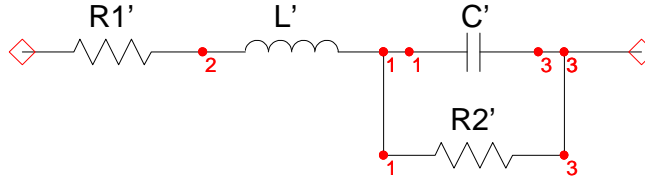


Figure 5 Complex pole pair realisation

where the component values are related to those from Figure 4 as:

$$C' = L \quad L' = C \quad R_1' = \frac{1}{R_1} \quad R_2' = \frac{1}{R_2}$$

## 4. Results

### 4.1 Example I

An InP High Electron Mobility Transistor (HEMT) is modelled in the frequency band from 0.5 to 50 GHz [1], resulting in the following black-box model ( $Z_0=50\Omega$ , normalised frequency):

Table 1 S-parameter model

	Numerator	Denominator
<b>S11</b>	$1.1131e-5s - 1.9031e-3$	$-4.5801e-11s^3 - 4.4194e-8s^2 - 2.6258e-5s - 2.1534e-3$
<b>S21</b>	$-1.4811e-5s + 8.4396e-3$	$-4.5801e-11s^3 - 4.4194e-8s^2 - 2.6258e-5s - 2.1534e-3$
<b>S12</b>	$-4.3647e-6s - 3.0148e-5$	$-4.5801e-11s^3 - 4.4194e-8s^2 - 2.6258e-5s - 2.1534e-3$
<b>S22</b>	$9.9696e-14s^4 - 2.4936e-11s^3 + 2.3827e-8s^2 + 2.0324e-6s - 1.3252e-3$	$-4.5801e-11s^3 - 4.4194e-8s^2 - 2.6258e-5s - 2.1534e-3$

The equivalent circuit components generated by the proposed method are:

```
.SUBCKT Z11 1 9
R1 1 3 1.038883E+000
C1 1 3 -2.056596E-004
R2 1 2 -1.008801E+000
L1 2 3 -1.785593E-004
R3 3 5 1.275936E+001
C3 3 5 -5.168624E-004
R4 3 4 1.612022E-002
L3 4 5 -2.607205E-003
R5 5 6 1.870652E+001
C5 5 6 2.069614E-004
R6 6 7 -1.151128E+000
C6 6 7 -8.899751E-003
R7 7 8 3.034964E+002
C7 7 8 4.813018E-004
R8 8 9 5.000000E+001
.ENDS Z11

.SUBCKT Z12 1 8
R1 1 3 6.058467E-001
C1 1 3 9.590067E-004
R2 1 2 -3.764646E-001
L1 2 3 5.006936E-004
R3 3 5 -5.905646E+000
C3 3 5 -2.587640E-004
R4 3 4 2.462732E+000
L3 4 5 -3.032193E-003
R5 5 6 1.105415E+001
C5 5 6 3.502330E-004
R6 6 7 -3.517611E-001
C6 6 7 -2.912417E-002
R7 7 8 1.275610E-001
C7 7 8 1.145126E+000
.ENDS Z12

.SUBCKT Z21 1 8
R1 1 3 2.990291E-001
C1 1 3 6.418553E-004
R2 1 2 -2.887257E-001
L1 2 3 6.808142E-005
R3 3 5 -1.425776E+000
C3 3 5 -2.088016E-004
R4 3 4 1.341406E+000
L3 4 5 -3.814222E-004
R5 5 6 1.235686E+002
C5 5 6 3.133100E-005
R6 6 7 -8.783457E+000
C6 6 7 -1.166369E-003
R7 7 8 -4.065026E+003
C7 7 8 -3.593417E-005
.ENDS Z21

.SUBCKT Z22 1 9
R1 1 3 3.848777E-001
C1 1 3 2.258750E-003
R2 1 2 -8.041276E-002
L1 2 3 4.441639E-004
R3 3 5 -1.807365E+001
C3 3 5 3.715975E-005
R4 3 4 1.314561E+001
L3 4 5 9.875465E-003
R5 5 6 7.301976E+001
C5 5 6 5.302027E-005
R6 6 7 -2.684045E+000
C6 6 7 -3.816907E-003
R7 7 8 -1.708550E+000
C7 7 8 -8.549552E-002
R8 8 9 -5.000000E+001
.ENDS Z22
```

Figure 6 shows the S-parameters calculated from the voltages and currents simulated by Spice.

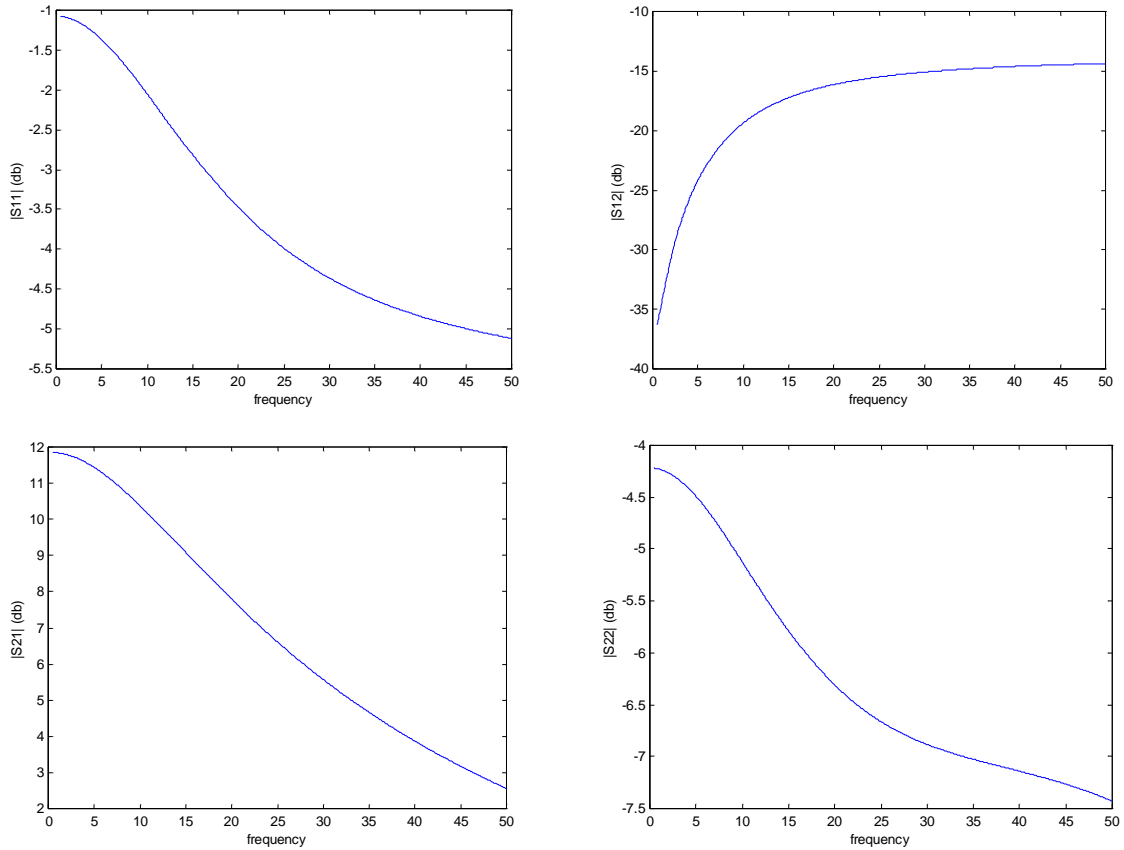


Figure 6 S-parameters of the model

The S-parameters calculated from Spice were compared to the model's S-parameters. Figure 7 shows the difference between  $S_{11}$  calculated by Spice and  $S_{11}$  calculated from the model in Table 1. Similar errors are obtained for the other parameters. The error is consistent with the 8 significant digits precision of the Spice version used.

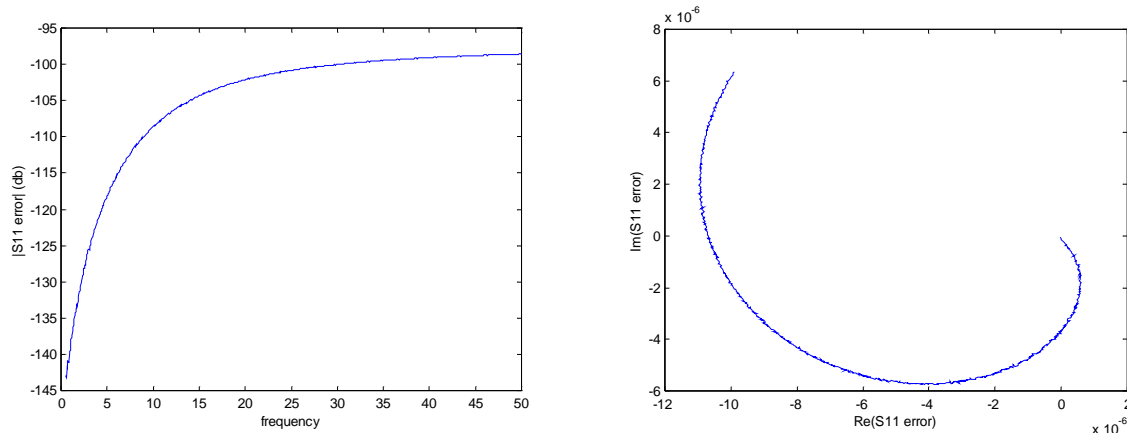


Figure 7 Difference between model and simulation for  $S_{11}$  (Z-model)

The Y-matrix model yields the following circuit:

```
.SUBCKT Y11 1 2
R1 1 3 1.983465E+004
L1 3 2 -1.782615E+001
R2 1 4 -4.149081E+001
L2 4 2 -5.752911E-002
R3 1 5 7.034542E+001
L3 5 6 1.568342E-001
R4 6 2 5.695539E+002
C3 2 6 2.555916E-005
R5 1 7 -8.778829E-001
L5 7 8 9.042542E-002
R6 8 2 3.031010E+002
C5 2 8 7.528750E-005
R7 1 9 2.650915E+004
L7 9 2 2.775743E+002
R8 1 2 5.000000E+001
.ENDS Y11

.SUBCKT Y12 1 2
R1 1 3 -1.149827E+003
L1 3 2 1.033393E+000
R2 1 4 7.573437E+002
L2 4 2 1.050095E+000
R3 1 5 -1.636140E+001
L3 5 6 6.189356E-001
R4 6 2 3.354458E+002
C3 2 6 5.483379E-006
R5 1 7 3.457749E+001
L5 7 8 -2.827458E-001
R6 8 2 -2.993495E+002
C5 2 8 -2.135848E-005
R7 1 9 8.780577E+004
L7 9 2 9.194041E+002
.ENDS Y12

.SUBCKT Y21 1 2
R1 1 3 -6.988370E+002
L1 3 2 6.280711E-001
R2 1 4 1.234839E+002
L2 4 2 1.712167E-001
R3 1 5 2.267715E+001
L3 5 6 8.609496E-002
R4 6 2 7.238834E+001
C3 2 6 5.442357E-005
R5 1 7 -1.512259E+001
L5 7 8 -5.242319E-002
R6 8 2 4.530855E+001
C5 2 8 -8.677098E-005
R7 1 9 3.445667E+003
L7 9 2 3.607918E+001
.ENDS Y21

.SUBCKT Y22 1 2
R1 1 3 4.051203E+001
L1 3 2 -3.640969E-002
R2 1 4 -2.253987E+003
L2 4 2 -3.125267E+000
R3 1 5 -1.910753E+002
L3 5 6 5.247290E-001
R4 6 2 3.579410E+002
C3 2 6 3.169794E-006
R5 1 7 2.018997E+001
L5 7 8 1.334256E-001
R6 8 2 -1.869199E+002
C5 2 8 4.564487E-005
R7 1 9 1.141302E+004
L7 9 2 1.195044E+002
R8 1 2 -5.000000E+001
.ENDS Y22
```

Figure 8 shows the difference between  $S_{11}$  calculated by Spice and  $S_{11}$  calculated from the model in Table 1. The Y-model circuit appears to be more accurate than the Z-model for higher frequencies.

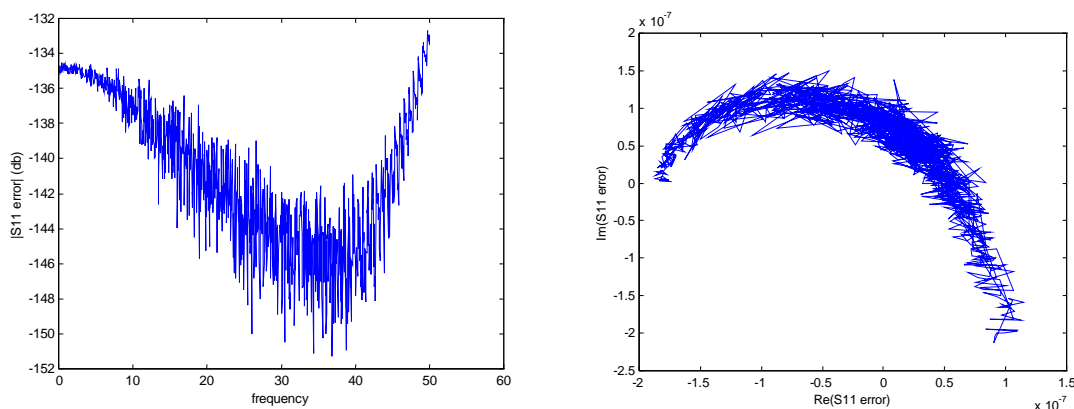


Figure 8 Difference between model and simulation for  $S_{11}$  (Y-model)

At this time the proposed method has only been tested on a limited number of examples. For these examples, transient analysis is numerically stable when the model describes a stable system, but the number of examples is too small to make general statements on the numerical stability of the generated circuits. If black-box models are to be used as circuits in a network simulation, modellers will have to make sure their identification methods produce stable models. The InP HEMT S-parameter model, for example, is unstable, which manifests itself by the occurrence of poles with positive real parts for the Z-parameters.

## 4.2 Example II

The second example was a 24 order model of a HF filter. This model resulted in very inaccurate results. The Z- and Y-matrices obtained were very poorly conditioned. From this example, we learned that if the modelled network is well-matched ( $s_{12}$  and  $s_{21}$  close to unity) and  $s_{11}$  and  $s_{22}$  are small, the denominator becomes nearly zero for both Y- and Z-parameters. The proposed method appears to be unsuitable for modelling such transmission line-like devices. It also seems advisable to check the condition number of the Z or Y matrix before proceeding.

## 4.3 Example III

This example synthesises the Z-parameter model of a synchronous machine modelled by J. Verbeeck [5]. Synchronous machine parameters are traditionally identified with an equivalent LR ladder network. The model's Z-parameters are shown in Figure 9 (note that this is a reciprocal network).

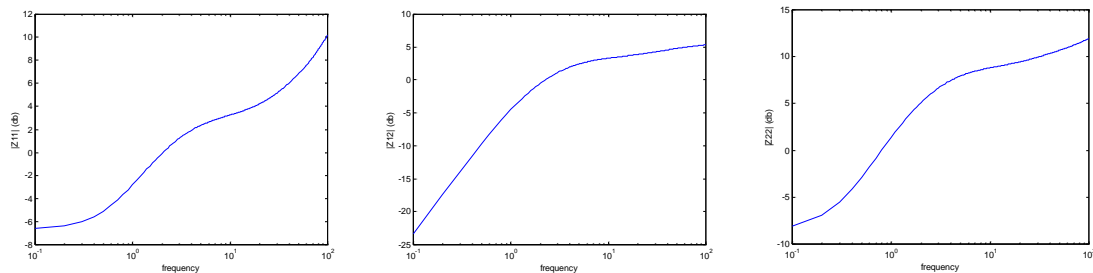


Figure 9 Z-parameters of synchronous machine

The 4/3 order model is then approximated by the following equivalent LR ladder network:

```
.SUBCKT Jef 1 4 12 4
R1 1 2 4.741618814635397e-001
R4 6 4 2.971556084096895e+000
R6 8 4 1.328237101443709e+001
R8 10 4 2.143615057838099e+001
R9 11 12 3.763330408395091e-001
L1 2 3 -3.599159752387534e-002
L2 3 4 1.069567284337460e-001
L3 3 5 6.399584942680099e-002
L4 5 6 1.094262956526974e-003
L5 5 7 -1.151214817679008e-003
L6 7 8 6.668201160125510e-002
L7 7 9 -5.277981128391941e-003
L8 9 10 1.181033726542707e+000
L9 9 11 8.481258744540992e-003
.ENDS Jef
```

The new method proposed here results in the following circuit ( $Z_{21}=Z_{12}$ ):

```
.SUBCKT Z11 1 6
R1 1 2 -1.991195E-001
C1 1 2 -2.030820E-002
R2 2 3 -3.636251E-001
C2 2 3 -1.186608E-001
R3 3 4 -5.478088E-001
C3 3 4 -1.666291E-001
L4 4 5 4.465771E-003
R5 5 6 1.584715E+000
.ENDS Z11

.SUBCKT Z12 1 6
R1 1 2 -3.513564E-001
C1 1 2 -1.150899E-002
R2 2 3 -5.948018E-001
C2 2 3 -7.254188E-002
R3 3 4 -8.675134E-001
C3 3 4 -1.052213E-001
L4 4 5 6.840193E-004
R5 5 6 1.813672E+000
.ENDS Z12

.SUBCKT Z22 1 6
R1 1 2 -6.199862E-001
C1 1 2 -6.522337E-003
R2 2 3 -9.729505E-001
C2 2 3 -4.434762E-002
R3 3 4 -1.373800E+000
C3 3 4 -6.644408E-002
L4 4 5 3.115033E-003
R5 5 6 3.343070E+000
.ENDS Z22
```

Note that, not including the main circuit, this amounts to 32 components, whereas the “traditional” ladder network requires only 14. On the other hand, the traditional ladder network is unable to model any  $Z$ -matrix, while the proposed method has a lot more degrees of freedom. However, the traditional network is thought to partially represent the underlying physical structure of a synchronous machine.

Figure 10 shows the difference between  $Z_{11}$  as calculated from the element values of the LR equivalent network and  $Z_{11}$  calculated from the Spice simulation of the LR network on the left, and between  $Z_{11}$  from the element values of the LR network and  $Z_{11}$  calculated from the Spice simulation of the proposed equivalent circuit on the right. The result is consistent with Spice’s numerical accuracy.

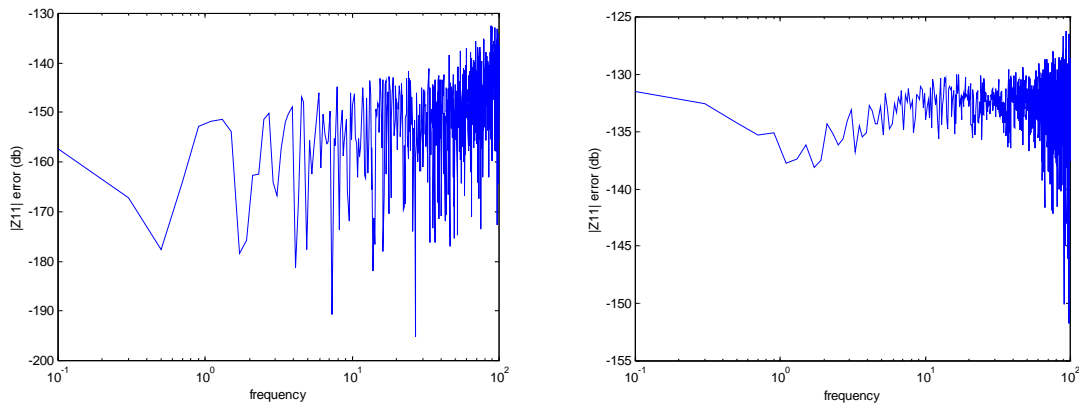


Figure 10  $Z_{11}$  error for traditional LR network (left) and new equivalent (right)

Note that this is a rather simple network. Experience suggests that, for very high order models, partial fraction expansion tends to become numerically inaccurate unless properly scaled after each step. Use of the proposed method for such models will require proper caution against this eventuality.

Since the synchronous machine model is stable, a transient analysis is possible for this example. Figure 11 shows the result of a Spice transient analysis; the output voltage following a 100V, 1s step excitation, is compared for the traditional LR ladder circuit and the proposed circuit structure. Clearly the proposed circuit is equivalent to the network whose  $Z$ -matrix it is meant to synthesise.

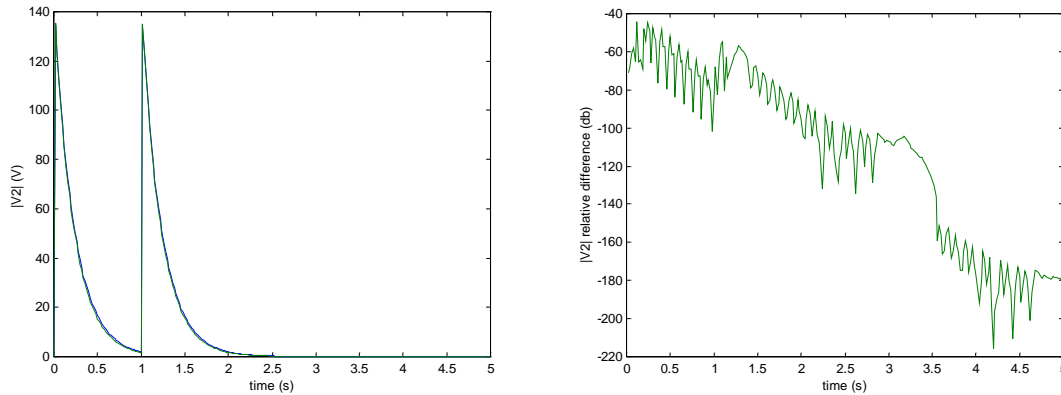


Figure 11 Transient responses and difference between the proposed and LR networks

## 5. Conclusion

A method is proposed to automatically generate a Spice circuit corresponding to a given S-parameter model in the Laplace domain, using only standard Spice components. The method lends itself to the generation of circuits corresponding to other two-port parameters as well. Test results show the method to be accurate, if the Z-matrix (or Y matrix) for the S-parameter model exists. The resulting network is numerically stable if the model is.

Synthesising each 2-port parameter separately is very simple. One possible disadvantage of this approach is the large number of circuit elements needed. In more traditional synthesis methods, components are selected to synthesise several 2-port parameters at once. The proposed approach may result in at most 4 times as many components. The simplicity of the parameter synthesis more than likely outweighs this disadvantage.

## References

- 
- [1] Y. Rolain, G. Vandersteen, D. Schreurs, S. Van den Bosch, "Parametric modelling of Linear Time Invariant S-parameter Devices in the Laplace domain." *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, Brussels, Belgium, June 1996, pp. 1244-1249.
  - [2] Max W. Medley, *Microwave and RF circuits: analysis, synthesis and design*. Artech house, Inc, 1993, pp. 29.
  - [3] Gobind Daryanani, *Principles of Active Network Synthesis and Design*. John Wiley & Sons, 1976, pp. 375-380.
  - [4] Norman Balabanian, *Network Synthesis*. Prentice Hall, 1964, pp. 132-133.
  - [5] J. Verbeeck, R. Pintelon and P. Lataire, "Identification of Synchronous Machine Parameters Using a Multiple Input Multiple Output Approach", Paper n° PE-059-EC-0-04-1998 accepted for publication in *IEEE Trans. on Energy Conversion* and presented at the 1999 IEEE PES Winter Meeting, Jan. 31- Feb. 4, 1999, New York.

# Appendix:

## MatLab m-files to create a Spice subcircuit with given S parameters.

```
function c =ZModel(name,z0,s11num,s12num,s21num,s22num,denom)

% parameters should be given as equally sized polynomial coefficient arrays
% use zp2tf to convert from zero-pole form to polynomial form
% z0 is the reference impedance
% name will be the name of the generated subcircuit

z11teller=z0*(conv(denom+s11num,denom-s22num)+conv(s12num, s21num));
z12teller=z0*2*conv(denom, s12num);
z21teller=z0*2*conv(denom, s21num);
z22teller=z0*(conv(denom-s11num,denom+s22num)+conv(s12num,s21num));
noemer=conv(denom-s11num,denom-s22num)-conv(s12num,s21num);

s='';
s=strcat(s, impedantie('Z11',z11teller,noemer));
s=strcat(s, impedantie('Z12',z12teller,noemer));
s=strcat(s, impedantie('Z21',z21teller,noemer));
s=strcat(s, impedantie('Z22',z22teller,noemer));

main=sprintf('.SUBCKT %s 1 9 10 2 \n ** Use as X??? gate1_node1 gate1_node2 gate2_node1
gate2_node2 %s ** \n F1 4 0 VF1 1 \n VF1 9 3 0V \n F2 6 0 VF2 1 \n VF2 5 2 0V \n X12 6
0 Z12 \n E2 7 10 4 0 -1 \n X22 7 5 Z22 \n E1 8 1 6 0 1 \n X11 3 8 Z11 \n X21 0 4
Z21 \n.ENDS %s \n', name,name,name);

c=strcat(s,main);

function z =impedantie(naam,teller,noemer)

s='';
[r,p,k]=residue(teller,noemer);

n=length(r);
ind1=1;

while ind1<=n
    if (imag(p(ind1)) ~= 0)
        s= strcat(s, adddouble(r,p,ind1));
        ind1=ind1+2;
    end
    if (imag(p(ind1)) == 0)
        s=strcat(s, addsingle(r,p,ind1));
        ind1=ind1+1;
    end
end

n=length(k);
if n>2
    error=sprintf('\n\nERROR! \n impedance %s contains %d infinity poles \n\n', naam, n-1 );
    s=strcat(s,error);
end
ind2=1;
while ind2<=n
    s=strcat(s,addk(k,ind2,ind1));
    ind1=ind1+1;
    ind2=ind2+1 ;
end

t=sprintf('.SUBCKT %s %d %d \n', naam, 1, ind1);
t2=sprintf('.ENDS %s \n \n', naam);
z=strcat( t, s, t2) ;

function s= adddouble(r,p,ind1)
a=real(r(ind1));
b=imag(r(ind1));
q=real(p(ind1));
w=imag(p(ind1));
```

```

condensator= 1/(2*a) ;
R2overL=-(q+b*w/a);
weerstand1=(-1/R2overL)/(2*q*condensator/R2overL+condensator);
spoel=1/(condensator*(w^2+q^2-R2overL/(weerstand1*condensator)));
weerstand2=R2overL*spoel;

s=sprintf('R%d %d %d %E \nC%d %d %d %E \nR%d %d %d %E \nL%d %d %d %E \n', ind1, ind1,
ind1+2, weerstand1, ind1, ind1, ind1+2, condensator, ind1+1, ind1, ind1+1, weerstand2,
ind1, ind1+1, ind1+2,spoel);

function s= addsingle(r,p,ind1)

weerstand= -r(ind1)/p(ind1);
condensator=1/r(ind1);
s=sprintf('R%d %d %d %E \nC%d %d %d %E \n', ind1, ind1, ind1+1, weerstand, ind1, ind1,
ind1+1, condensator);

function s= addk(k,ind2,ind1)

if ind2==length(k)
s=sprintf('R%d %d %d %E \n', ind1, ind1, ind1+1, k(ind2));
else
s=sprintf('L%d %d %d %E \n', ind1, ind1, ind1+1, k(ind2));
end

```

```

function c =YModel(name,z0,s11num,s12num,s21num,s22num,denom)

% parameters should be given as equally sized polynomial coefficient arrays
% use zp2tf to convert from zero-pole form to polynomial form
% z0 is the reference impedance
% name will be the name of the generated subcircuit

y11teller=(conv(denom-s11num,denom+s22num)+conv(s12num, s21num));
y12teller=-2*conv(denom, s12num);
y21teller=-2*conv(denom, s21num);
y22teller=(conv(denom+s11num,denom-s22num)+conv(s12num,s21num));
noemer=z0*(conv(denom+s11num,denom+s22num)-conv(s12num,s21num));

s='';
s=strcat(s, admittantie('Y11',y11teller,noemer));
s=strcat(s,admittantie('Y12',y12teller,noemer));
s=strcat(s,admittantie('Y21',y21teller,noemer));
s=strcat(s,admittantie('Y22',y22teller,noemer));

main=sprintf('SUBCKT %s 1 7 2 8 \n ** Use as X??? gate1_node1 gate1_node2 gate2_node1
gate2_node2 %s ** \n X11 1 7 Y11\nX22 2 8 Y22\nF1 1 7 VF1 1\nVF1 3 4 0V\nE2 3 0 2
8 1\nX21 0 4 Y21\nE1 5 0 1 7 1\nF2 2 8 VF2 1\nVF2 5 6 0V\nX12 6 0 Y12 \n.ENDS %s
\n', name,name,name);

c=strcat(s,main);

function z =admittantie(naam,teller,noemer)

s='';
[r,p,k]=residue(teller,noemer);

n=length(r);
ind1=1;

while ind1<=n
    if (imag(p(ind1)) ~= 0)
        s= strcat(s, adddouble(r,p,ind1));
        ind1=ind1+2;
    end
    if (imag(p(ind1)) == 0)
        s=strcat(s, addsingle(r,p,ind1));
        ind1=ind1+1;
    end
end

n=length(k);
if n>2
error=sprintf('\n\nERROR! \n impedance %s contains %d infinity poles \n\n', naam, n-1 );
s=strcat(s,error);
end
ind2=1;
while ind2<=n
    s=strcat(s,addk(k,ind2,ind1));
    ind1=ind1+1;
    ind2=ind2+1 ;
end

t=sprintf('SUBCKT %s %d %d \n', naam, 1, 2);
t2=sprintf('ENDS %s \n \n', naam);
z=strcat( t, s, t2) ;

function s= adddouble(r,p,ind1)
a=real(r(ind1));
b=imag(r(ind1));
q=real(p(ind1));
w=imag(p(ind1));

condensator= 1/(2*a) ;
R2overL=-(q+b*w/a);
weerstand1=(-1/R2overL)/(2*q*condensator/R2overL+condensator);
spoel=1/(condensator*(w^2+q^2-R2overL/(weerstand1*condensator)));
weerstand2=R2overL*spoel;

```

```

Lbis=condensator;
Cbis=spoel;
Rlbis=1/weerstand1;
R2bis=1/weerstand2;

s=sprintf('R%d %d %d %E \nL%d %d %d %E \nR%d %d %d %E \nC%d %d %d %E \n', ind1,1, ind1+2,
Rlbis, ind1, ind1+2, ind1+3, Lbis, ind1+1, ind1+3,2,R2bis, ind1, 2, ind1+3,Cbis);

function s= addsingle(r,p,ind1)

weerstand= -p(ind1)/r(ind1);
spoel=1/r(ind1);
s=sprintf('R%d %d %d %E \nL%d %d %d %E \n', ind1,1, ind1+2, weerstand, ind1, ind1+2, 2,
spoel);

function s= addk(k,ind2,ind1)

s=sprintf('R%d %d %d %E \n', ind1, 1,2,1/k(ind2));

function s= addk(k,ind2,ind1)

if ind2==length(k)
    s=sprintf('R%d %d %d %E \n', ind1, 1,2,1/k(ind2));
else
    s=sprintf('C%d %d %d %E \n', ind1, 1,2, k(ind2));
end

```