‘AMP’ - Admittance Matrix Program

AMP ver 8.D.
Marek Kikowski
December.2014

Disclaimer:

AMP is a free program and you can use it at your own risk.

0. Revisions

Date	Rev	Description
31.XII.2014	1	AMP_8D – documentation incomplete
01.I.2015	1	AMP_8D – documentation still incomplete
05.I.2015	1	AMP_8D – documentation less but still incomplete
10.I.2015	1	AMP_8D – documentation complete

Table of Contents

1. Introduction

Program ‘**AMP**’ performs a numeric analysis of linear analog circuits in a frequency domain.
In computations **Y-matrix** techniques are used, yet **s-parameters** of multi-port sub-circuits can be easily incorporated into circuit description to facilitate **RF** circuits analysis.
S-parameters are network description mainly obtained by measurements made by **VNA** - **V**ector **N**etwork **A**nalyzer.
The purpose of **'AMP'** development was to combine easiness of calculations using **Y-matrix** techniques with accuracy of **RF** modeling provided by **s-parameters**.

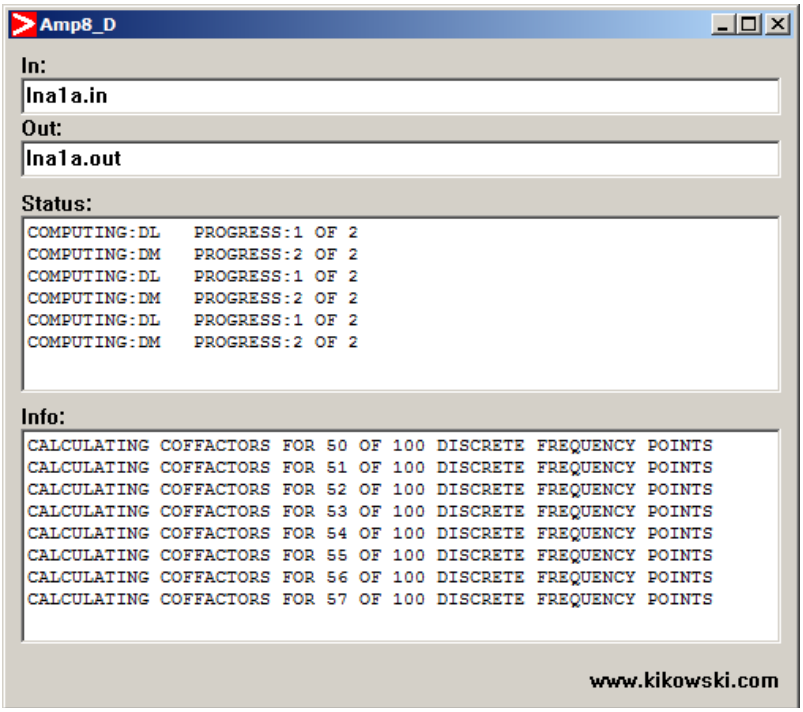


Fig.1.1.1. Amp window.

AMP is capable of computing transfer functions, their sensitivities or admittance matrix co-factors.

AMP process circuit description file and produces results of analysis in a text format. Input format is similar to **Spice**. Schematics entry might be provided by any schematics editor with options to customize **Spice** netlist generation. In all presented examples schematics entry is provided be **Rimu** [1]. **Rimu** configuration to produce circuit description in **AMP** format is described in additional document [4].

AMP was designed to provide flexible interface for external scripting tools, and routinely many tasks are performed as post-processing of it's results. Therefore **AMP** can create data files in a format suitable for post-processing.
In presented examples **Opus Spice-Nutmeg**[2] and **Scilab**[3] are used as a post-processing scripting tools.
$RAW command can be used to generate data in **Nutmeg** format.
If a script with project name and **.nut** extension is found in a working folder, **Spice-Nutmeg** will be executed with a **<project>.nut** script as an input parameter.
$MCAD commands generates additional data for **Scilab** or any other **matlab like** tool.
If a script with with project name and **.sce** extension is found in a working folder, **Scilab** will be invoked with a **<project>.sce** script as an input parameter.
Value modification command (**$LET**) together with include command (**$INC)** gives **AMP** ability to operate in the feedback loop. An external controlling script may modify input data for next run, thus facilitating **optimization** or '**Monte-Carlo**' analysis.
Options for calculations flow are shown in a figure **1.1.2** below.

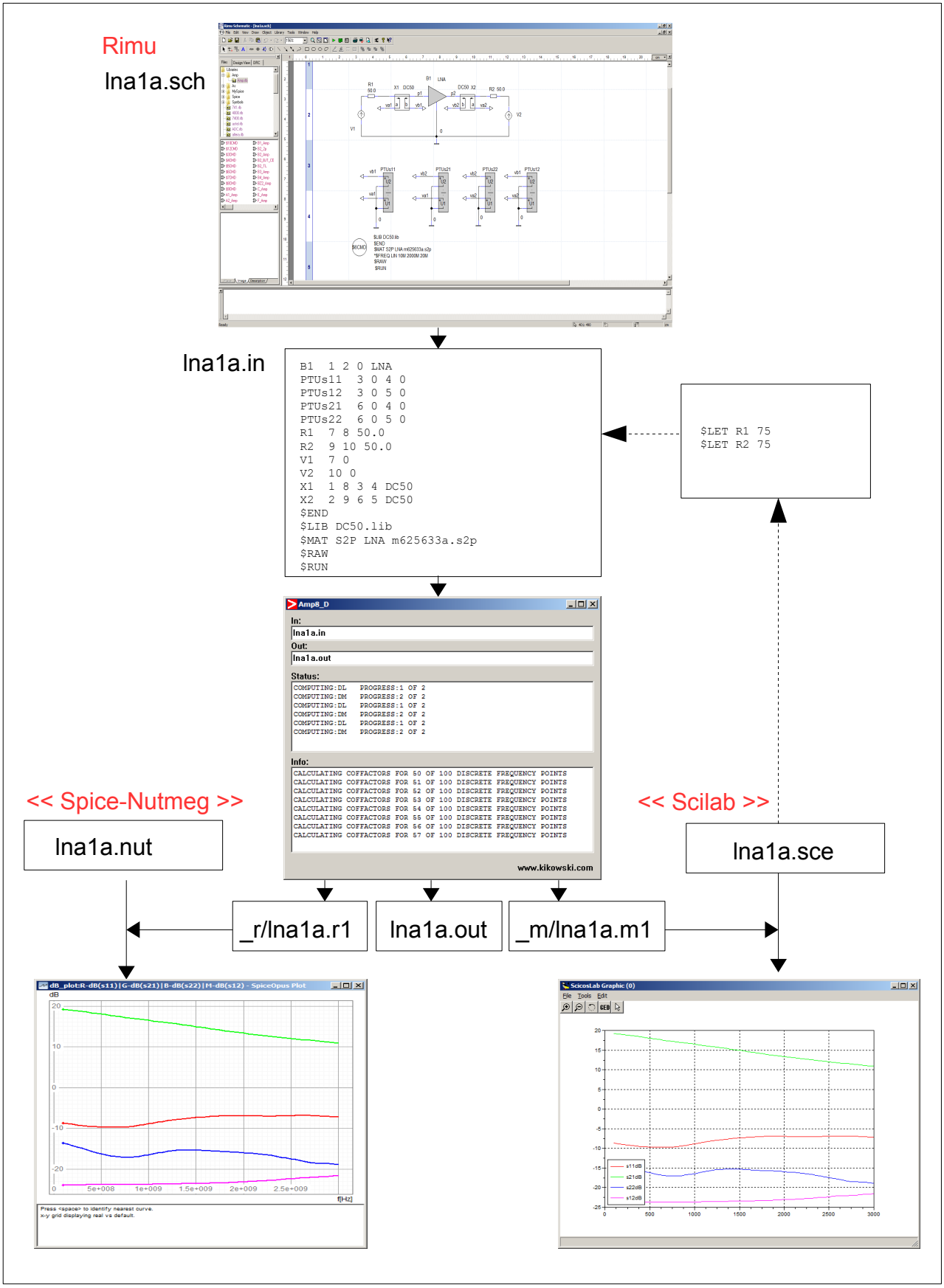


Fig.1.1.2. AMP analysis flow with <**RIMU**> and scripting tools <**NUTMEG**>,<**SCILAB**>.

'Spice-Nutmeg' and **'Scilab'** scripts are optional but have to be prepared individually as part of circuit analysis.

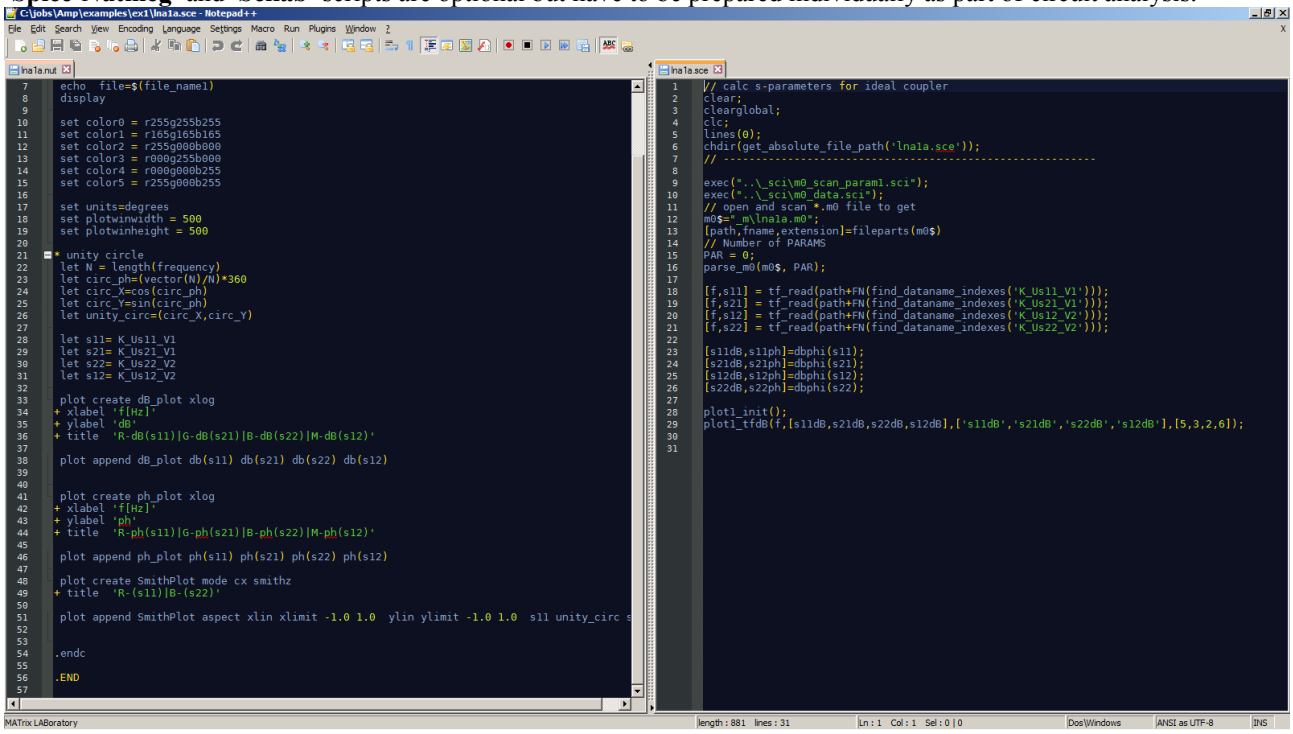


Fig.1.1.3. Typical scripts for interpreters.

It is rather unusual, that both types of scripts are used simultaneously.
Simple tasks are better served by **Spice-Nutmeg** scripts, more complicated tasks often require more powerful **Scilab** scripts.
With proper configuration <**Rimu**, **Amp.ini** file> , results of analysis should be available after pressing '**Run Simulator**' icon in **Rimu** (figure 1.1.4)

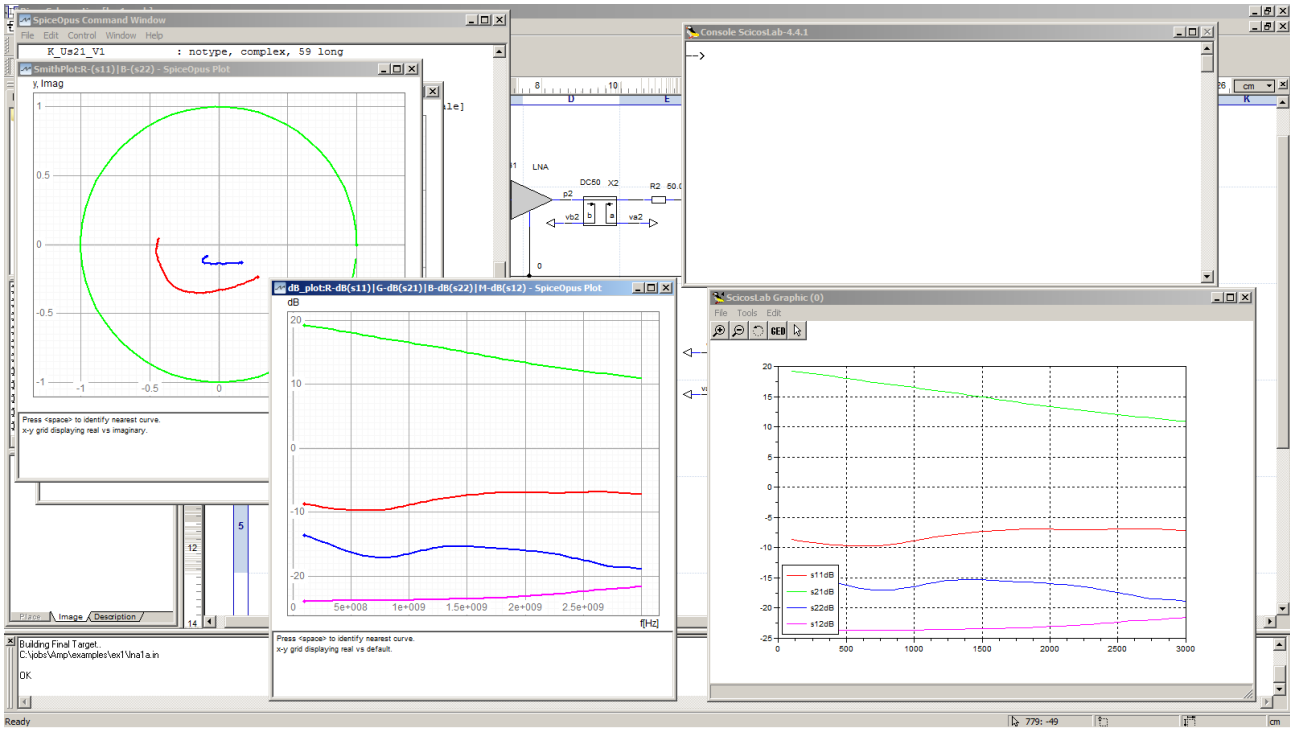


Fig.1.1.4. Plots generated by **Nutmeg** and **Scilab** scripts invoked by **AMP**.

1.1. AMP configuration file.

AMP configuration file (**amp.ini** in exe folder) has to be customized to match target system.

```
IN       .in
OUT      .out
LIBPATH '\jobs\Amp\lib'
NUTMEG   .nut
SPICE    '\programs\SpiceOpus\bin\spiceopus.exe'
MCADEXT .sce
MCADEXE 'cscilex.exe -nouserstartup -f'
TICK    300
GUI     1
MCADDIR '_m'
RAWDIR  '_r'
```

IN - default input file extension,
OUT - default output file extension,
LIBPATH - absolute library search path,
NUTMEG - default extension for **Nutmeg** scripts,
SPICE - absolute path to **OPUS Spice**
MCADEXT - default extension for **Scilab** (or any other matlab-like) scripts,
MCADEXE - absolute path to **Scilab** (actually, this is **Scicoslab** in this case),
TICK - miliseconds per internal clock tick – change it to **1** when running **AMP** in optimization loop,
GUI - boolean flag, if 1 GUI is displayed - change it to **0** when running **AMP** in optimization loop,
MCADDIR - default name of subfolder for files genereated by **$MCAD** command
RAWDIR - default name of subfolder for files genereated by **$RAW** command

1.2. Circuit description.

Circuit description is similar to the standard **Spice** syntax, yet with some notable exceptions.
Circuit specification requires sequential upward node numbering from 0 for the ground. **Rimu** can easily be adopted to automate this task. **AMP** analyzes circuit models, which are build using the following components:

Sources:

V	voltage source
I	current source

Probes:

PU	voltage probe,
PI	current probe,
PZ	impedance probe
PTU	transfer function probe TF = U(a,b)/U(c,d)
PTI	transfer function probe TF = I(a,b)/I(c,d)
PTN	transfer function probe TF = I(a,b)/U(c,d)
PTM	transfer function probe TF = U(a,b)/I(c,d)

Parts:

A	ideal operational amplifier
B	multiport sub-network defined by a matrix file
G	voltage controlled current source
F	current controlled current source
E	voltage controlled voltage source
H	current controlled voltage source
Y	admittance
C	capacitor
L	coil
K	mutual inductance coupling between two coils
R	resistor
Z	impedance
T	lossless transmission line
X	sub-circuit defined by list of other parts

Parts and sources are identified by the first letter.
First letter case is irrelevant.
Probes use two PU, PI, PZ or three letters for identifications PTU, PTI, PTM, PTN.

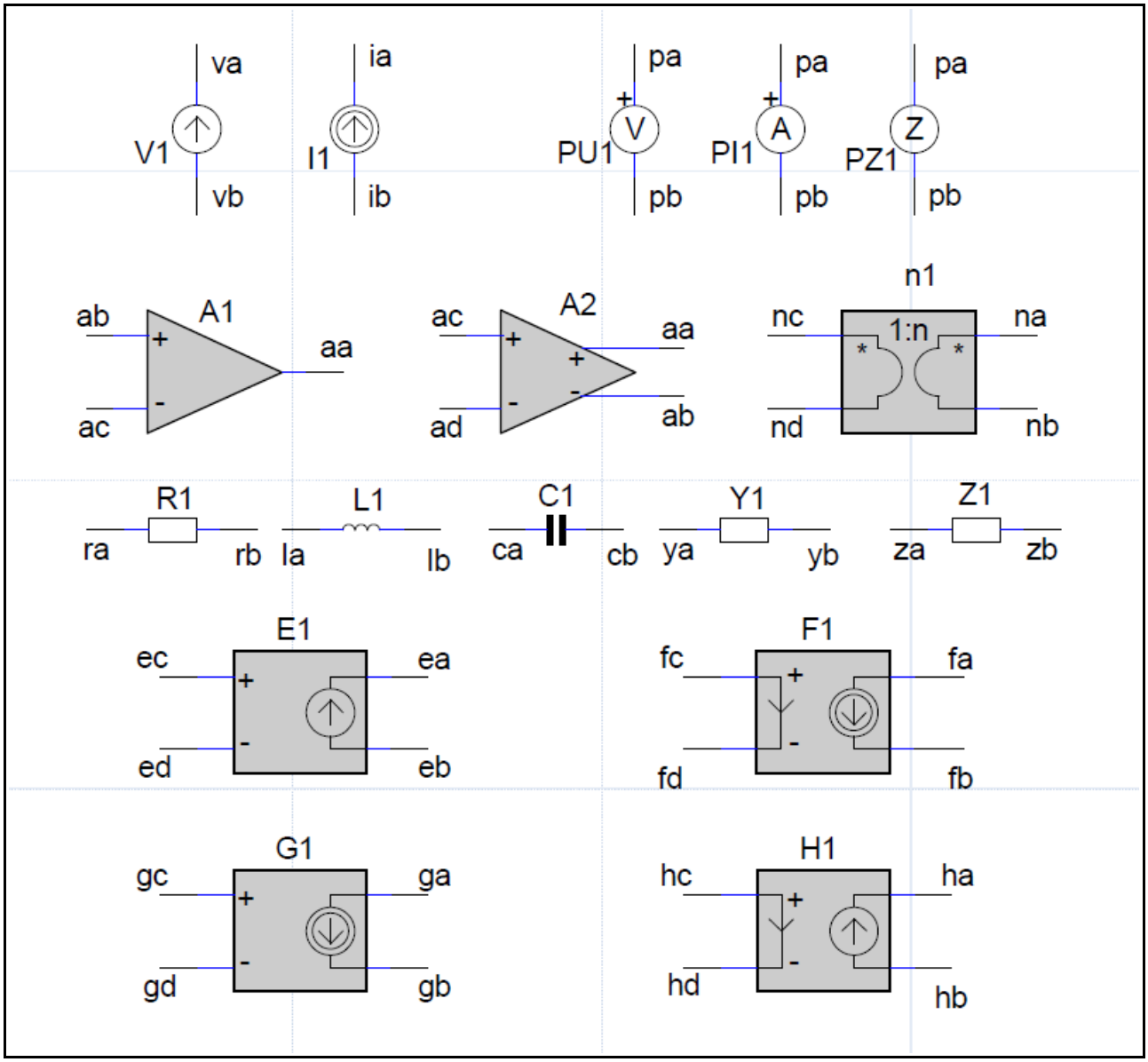


Fig.1.2.1. Sources, basic probes and parts in **Rimu** schematics

1.2.1. Specification of sources and probes.

Sources:
V - Voltage Source

```
Vxxx va vb
```

I - Current Source

```
Ixxx ia ib
```

WARNING:
For current source positive current flows from **ib** to **ia**, i.e. **ia** node is a source (not a sink like in **SPICE**). **Thus, this convention is different than in SPICE**.
Sources have no value, because together with probes they determine the form of calculated transfer function **TF.**

P – Probes (voltage, current, impedance)

```
PUxx pa pb
```
U(pa,pb)=V(pa)-V(pb) | with a,b open I(pa,pb)=0

```
PIxx pa pb
```
I(pa,pb), positive current is flowing from pa to pb, with pa,pb shorted

```
PZxx pa pb
```
U(pa,pb)/I(pa,pb) - impedance probe - requires no source

Basic voltage and current probes **PU**, **PI** determine the nominator of transfer function. This means, that for each individual source (**Vx**,**Ix**) **AMP** calculates transfer function **TF** with nominator determined by a type of the probe. Thus, for each voltage source **Vx,** transfer function is **TF = U(pa,pb)/Vx** or **TF = I(pa,pb)/Vx**, and for each current source **Ix,** transfer function is **TF = U(pa,pb)/Ix** or **TF = I(pa,pb)/Ix** depending on the type of probe and source. Transfer functions **TF** are calculated according to superposition theorem, i.e. all irrelevant sources are in off state (voltage sources nodes are short-circuit **Vy=0**, current sources nodes are open **Iy=0**).

PT – Transfer function probes (voltage, current, impedance)

```
PTU pa pb pc pd
```
transfer function probe TF = U(a,b)/U(c,d)

```
PTI  pa pb pc pd
```
transfer function probe TF = I(a,b)/I(c,d)

```
PTN pa pb pc pd
```
transfer function probe TF = I(a,b)/U(c,d)

```
PTM pa pb pc pd
```
transfer function probe TF = U(a,b)/I(c,d)

1.2.2. Basic parts specification.

A – Amp (ideal):

```
Axxx aa ab ac
Axxx aa ab ac ad
```

E – VCVS:

```
Exxx ea eb ec ed  Evalue
```

F – CCCS:

```
Fxxx fa fb fc fd  Fvalue
```
control nodes fc,fd are short-circuit, positive current flow is from fc to fd

G - VCCS:

```
Gxxx ga gb gc gd  Gvalue
```

H – CCVS:

```
Hxxx ha hb hc hd  Hvalue
```
control nodes hc,hd are short-circuit, positive current flow is from hc to hd

N – Transformer (ideal) :

```
Nxxx na nb nc nd  Nvalue
```
U(na,nb)=Nvalue*U(nc,nd), i(nc,nd)= - Nvalue*i(na,nb) (current sinking into the a is sourced by c)

R – Resistor

```
Rxxx ra rb  Rvalue
```

C – Capacitor:

```
Cxxx ca cb  Cvalue
```

L – Inductor:

```
Lxxx la lb  Lvalue
```

K – Inductive coupling between coils LX LY – nodes (la) of coupled coils have positive magnetic polarity:

```
Kxy LX LY  Kvalue
```

T - Transmssion line, lossless with characteristics impedance Zo and delay TD

```
Txxx ta tb tc Zo TD
```

Y - Admitance

```
Yname ya yb  RE(Yvalue) IM(Yvalue)
```

Z - Impedance

```
Zname za zb  RE(Zvalue) IM(Zvalue)
```

1.2.3. Multiport network specification.

B – Block – multiport network:

```
Bxxx nb1 nb2 [nb3 nb4.....] nb0  MatrixName
$MAT MatrixType   Matrixname   MatrixFileName
```

Multiport network specification consist of two parts. First part is a line with **B** type component declaration together with nodes and matrix name definition. First node **nb1** corresponds to port1, second node **nb2** to port2, and so on. Last node nb0 is common node. Thus, 1 port device requires declaration with 2 nodes, 2 port – 3 nodes, and so on.
Second part is line with **$MAT** command, which for a given matrix name defines matrix type and a file name.
Two matrix types are supported; **SxP** s-parameters in **Touchstone** format or **YRI - Y-matrix** in real, imaginary format.
For **Touchstone** type **SxP**, number of ports is indicated by **x** in the middle of the type. Currently **AMP** accepts Touchstone files for network with up to 9 ports [**S1P** … **S9P**] in all **S** formats (**Y**,**Z** are not supported).
YRI file type should include for each frequency [in **Hz**] and all matrix elements [in **S=Ω^(-1)**] row-wise in a single line separated with spaces like in: < f Re{y11} Im{y11} Re{y12} Im{y12} Re{y21} Im{y21} Re{y22} Im{y22} >

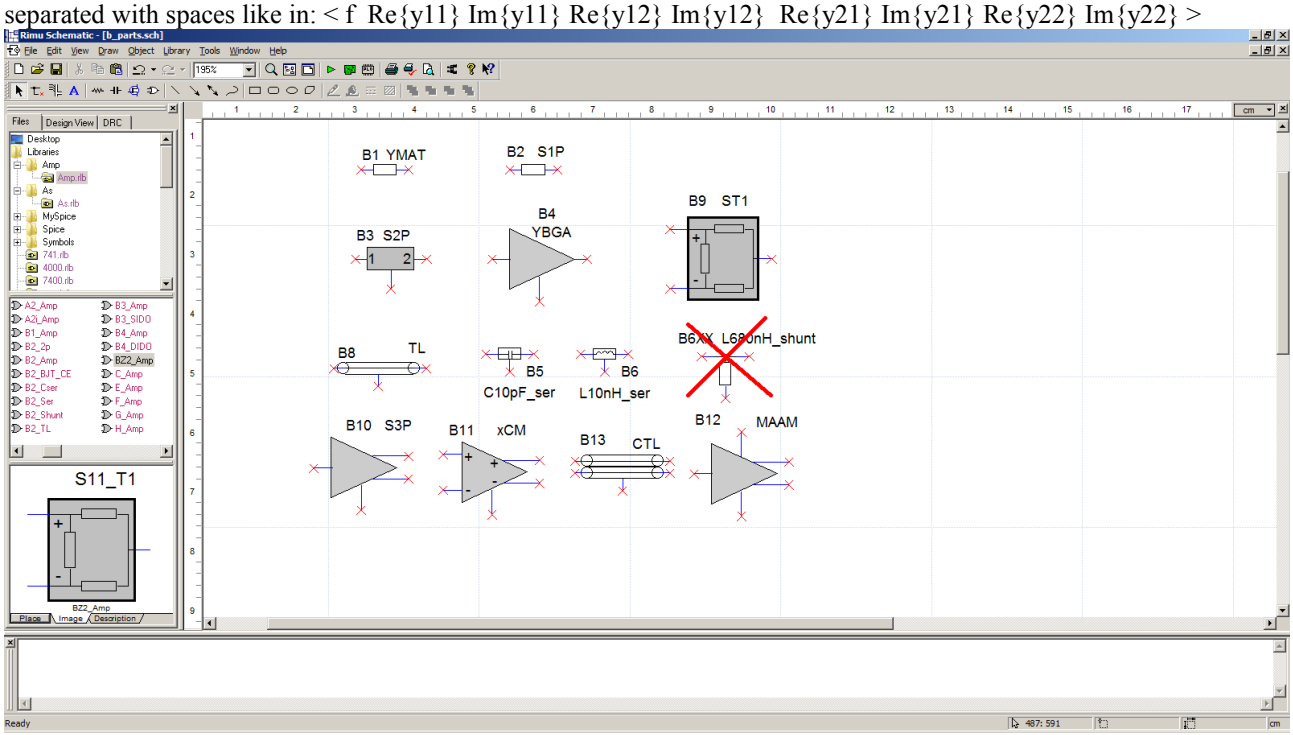


Fig.1.2.1. Example multiport **B-parts**.

Example 1:

```
B2  1 2 0 AH2
$MAT S2P AH2 ah2.s2p
```

Example 2:

```
B1  4 5 0 YBGA
B3  2 3 0 YBGA
$MAT YRI YBGA ybga_1.yri
```

Internally, **s-parameters** are converted into **Y-matrix** format and linearly interpolated for frequencies specified for analysis. This works well for most practical cases, but in one notable case it always fails. This case is, 2 port **RLC** part characterization in shunt configuration. **Y-matrix** does not exist for shunt configuration and **AMP** will report error during **s-parameters** conversion. Since **RLC** parts are in essence one port devices, workaround is relatively easy – it is a conversion to **S1P** parameters with the help of external script (see stability analysis example).
Second exception is **AMP** don't understand noise data embedded in Touchstone file. Therefore, noise data can't be combined with s-parameters in a single file. In case of noise analysis, noise data has to be moved to a separate file (see noise analysis example).

1.2.4. Macro-models - sub-circuits specification

X – Macromodel

```
      X xa xb xc Xmacro
      $LIB FileNameWithXmacroDefinition.lib

Example:
      X1  2  1  2  3  4   DC50
      $LIB DC50.lib
```

<DC50.lib> is a file name with macromodel definition:

```
# DC50 Po Pi Vb Va
A   1  2  Po Pi
Rb  1  Pi         50.0
Ra  2  Po         50.0
Ab  Vb 1  Vb
Aa  Va 2  Va
##
```

Macromodels is a sub-circuit description. Macro-models can be instantianted as X parts. The last parameter in X part declaration is a name of macro model.
The file with macro-model definition has to be declared with **$LIB** command.
The definition starts with single hash # sign, and ends with a line with two hashes - ##.

```
# Xmacro Node1 Node2 Node3 ...
<MACROMODEL BODY>
##
```

The first parameter in macro definition is a name of macro-model (**DC50** in the example).
Next parameters are symbolic **external nodes** (**Po Pi Vb Va** in the example).
Internal nodes has to be numbered sequentially upward starting from 1, 0 is the ground node.

The macro model definition can instantiate all other parts.

Macro-models are hierarchical i.e. a macro-model definition can instantiate other macro-model.

1.2.5. Extended part specification.

Models of RLC parts – models parameters are defined in **$MOD** command

```
Cxxx ca cb CValue  MOD CModelName
Lxxx la lb LValue  MOD LModelName
Rxxx ra rb RValue  MOD RModelName
```

Example:

```
C1 1 2 10p MOD M0402C100p
L1 2 3 10n MOD M0402L10n
R1 3 0 100 MOD M0402R100
```

Parameters – parts declared as parameters have value assigned by **$PAR** command during sweep analysis

```
Cxxx ca cb PAR CVarName
Lxxx la lb PAR LVarName
Rxxx ra rb PAR RVarName
Yname ya yb PAR YVarName
Zname za zb PAR ZVarName
Gxxx ga gb gc gd PAR GvarName
```

Example:

```
C1 1 2 PAR C1_par
L1 2 3 PAR L1_par
R1 3 0 PAR R1_par
```

Variable – variable parts with value assigned by **$VAR** command (mostly used internaly)

```
Cxxx ca cb VAR CVarName
Lxxx la lb VAR LVarName
Rxxx ra rb VAR RVarName
Yname ya yb VAR zref
Zname za zb VAR zref
Gxxx ga gb gc gd VAR GvarName
```

Other specifications

```
Cname ca cb TUN CTunName
Lname la lb TUN LTunName
Rname ra rb TUN RTunName
Gname ga gb gc gd TUN GTunName
Gname ga gb gc gd GEN
Rname ra rb GEN
```

1.3. Commands.

$LIB – load library file command

```
$LIB MyMacroModel.lib
$LIB MyNewLib/WithGreatModels/Inside.lib
```

$MOD – model declaration - to be used with parts specified as MOD

```
$MOD ModelType ModelName ModelParametr1 ModelParametr2
$MOD FQ ModelName ResonanceFrequency EquivalentQualityFactor
Example:
      $MOD FQ M0402C100p 1.0G 3.3
```

$MAT – matrix declaration, can implicitly define frequencies

```
$MAT MatrixType MatrixName MatrixFileName
Examples:
      $MAT S2P AH2 ah2.s2p
      $MAT YRI YBGA ybga_1.yri
```

$INC - include file command

```
$INC FileCommandsOnly.ext
```

$FREQ – define frequencies, can only be skipped only when $MAT is present

```
$FREQ LIST  F1 F2 F3...
$FREQ LIN   Fstart Fstop Fstep
$FREQ LOG   Fstart Fstop
$FREQ FILE  FileName.ext
```

$SENS – execute sensitivity analysis

```
$SENS
```

$RUN – run a single task, each task requires implicit ($MAT) or explicit ($FRQ) frequency declaration

```
$RUN
```

$MCAD – create text file in Matlab friendly format, VAL option to creates files with part values

```
$MCAD
$MCAD VAL
```

$RAW – create text file in Nutmeg format

```
$RAW
```

$LET – assign new value to exisiting part

```
$LET PartName NewValue
```

$TN – assign name to the task

```
$TN TaskName
```

$PAR – assign values by parametric sweep

```
$PAR parname LIST P1 P2 P3
$PAR parname CLIST Re{P1} Im{P1} Re{P2} Im{P2} Re{P3} Im{P3} ...
$PAR parname LIN Pstart Pstop Pstep
$PAR parname LOG Pstart Pstop
$PAR parname FILE  FileName.ext
$PAR parname CFILE ComplexFileName.ext
```

$FOR – printout format (default format is MP RI)

```
$FOR dB
$FOR MP
$FOR RI
$FOR dB MP
$FOR dB RI
$FOR MP RI
$FOR dB MP RI
```

$TRACE – printout internal data

```
$TRACE
```

$TRACE – define transfer function by ratio of coffactors

```
$TF TFName = DL(LCoffactor_spec)/DM(MCoffactor_spec)
```

Examples:

```
$TF  KU1=DL(1+0)(2+0)/DM(1+0)(1+0)
$TF  KU1=DL(1+0)(2+0),(3+0)(3+0)/DM(1+0)(1+0),(3+0)(3+0)
$TF  Z1[V1/I1] = DL(1+0)(1+0)/DM(0+0)(0+0)
```

$VAR – define values of variables

```
$VAR varname LIST V1 V2 V3 ...
$VAR varname CLIST Re{V1} Im{V1} Re{V2} Im{V2} Re{V3} Im{V3} ...
$VAR varname LIN Pstart Pstop Pstep
$VAR varname LOG Pstart Pstop
$VAR varname FILE  FileName.ext
$VAR varname CFILE FileName.ext
```

1.4. Circuit analysis.

Sources together with probes determine the form of calculated transfer function **TF.**
For each individual source (**Vx**,**Ix**) **AMP** calculates transfer function **TF** with nominator determined by a type of the probe.
Thus, for each voltage source **Vx,** transfer function is **TF = U(pa,pb)/Vx** or **TF = I(pa,pb)/Vx**,
and for each current source **Ix,** transfer function is **TF = U(pa,pb)/Ix** or **TF = I(pa,pb)/Ix** depending on the type of probe and source.
Transfer functions **TF** are calculated according to superposition theorem, i.e. all irrelevant sources (**Vy**,**Iy**) are in off state (voltage sources nodes are short-circuit **Vy=0**, current sources nodes are open **Iy=0**).

For file with simple circuit including one voltage source **V1** and one voltage probe **PU2 <ex0a.in>**:

```
* Simple voltage divider
R1    1   2         50
R2    2   0         50
V1    1   0
PU2   2   0
$END
$FREQ LIST   0
$RUN
```

AMP calculates transfer function K_U2_V1 = U2/V1 **<ex0a.out>**

```
* Simple voltage divider
R1    1   2         50
R2    2   0         50
V1    1   0
PU2   2   0
$END
$FREQ LIST   0
$RUN
*****************************      AMP 8_D      *****************************
DATE:1/1/2015
TIME:20:28:08
*****************************      AMP 8_D      *****************************
RESULTS OF TASK:1.0.1 NO_NAME
TF=K_U2_V1   FREQ= 0.0000000E+0000        OMEGA= 0.0000000E+0000
     |TF|=       0.50000000      deg(TF)=          0.00000
   Re(TF)= 5.0000000E-0001       Im(TF)= 0.0000000E+0000

TIME:20:28:08
*****************************      AMP 8_D      *****************************
```

When two commands $RAW and $MCAD VAL with $TN task name are added to the input file **<ex0b.in>:**

```
* Simple voltage divider
R1    1   2         50
R2    2   0         50
V1    1   0
PU2   2   0
$END
$RAW; $MCAD; $FREQ LIST  0; $TN TASK{KU=U2/V1}; $RUN
```

Output file looks virtually the same **<ex0b.out>:**

```
* Simple voltage divider
R1    1   2         50
R2    2   0         50
V1    1   0
PU2   2   0
$END
$MCAD VAL
$RAW
$FREQ LIST 0
$TN TASK{KU=U2/V1}
$RUN
******************************      AMP 8_D      ******************************
DATE:1/1/2015
TIME:20:57:50
******************************      AMP 8_D      ******************************
RESULTS OF TASK:1.0.1 TASK{KU=U2/V1}
TF=K_U2_V1   FREQ= 0.0000000E+0000         OMEGA= 0.0000000E+0000
      |TF|=       0.50000000      deg(TF)=          0.00000
   Re(TF)= 5.0000000E-0001       Im(TF)= 0.0000000E+0000

TIME:20:57:50
******************************      AMP 8_D      ******************************
```

Yet in subfolders **_m/** and **_r/** few new are created.
<_m/ex0b.m0> is a directory file with references to data file :

```
TASK:1.0.1 TASK{KU=U2/V1}
FILE: ex0b.m1
DATANAME: K_U2_V1
DATA: [1..1,1..3][F,RE(TF),IM(TF)]
EOF
```

<_m/ex0b.m1> is data file with TF : {FREQUENCY, RE(TF), IM(TF)

```
 0.00000000000000E+0000    5.00000000000000E-0001    0.00000000000000E+0000
```

<_m/ex0b.v0> is a directory file with references to data file <_m/ex0b.v1>

```
TASK:1.0.1 TASK{KU=U2/V1}
FILE: ex0b.v1
EOF
```

<_m/ex0b.v1>

```
R1 =  5.00000000000000E+0001;
R2 =  5.00000000000000E+0001;
```

<_r/ex0b.r1> is a **Nutmeg** file:

```
Title: Amp AC analysis results for Tasks = (1.0.* ):TASK{KU=U2/V1}
Date:1/1/2015 at 20:57:50
Plotname: AC Analysis
Flags: complex
Sckt. naming: from bottom to top
No. Variables: 2
No. Points: 1
Variables:
      0     frequency   frequency
      1     K_U2_V1     notype
Values:
0      0.00000000000000E+0000, 0.00000000000000E+0000
       5.00000000000000E-0001, 0.00000000000000E+0000
```

When $SENS command is added to the input file **<ex0c.in>:**

```
* Simple voltage divider
R1    1   2         50
R2    2   0         50
V1    1   0
PU2   2   0
$END
$SENS; $RAW; $MCAD; $FREQ LIST  0; $TN TASK{KU=U2/V1}; $RUN
```

AMP calculates transfer function K_U2_V1 = U2/V1 with sensitivities **<ex0c.out>**

```
* Simple voltage divider
R1    1   2         50
R2    2   0         50
V1    1   0
PU2   2   0
$END
$SENS
$MCAD VAL
$RAW
$FREQ LIST 0
$TN TASK{KU=U2/V1}
$RUN

*****************************      AMP 8_D      *****************************
DATE:1/1/2015
TIME:21:16:53
*****************************      AMP 8_D      *****************************
RESULTS OF TASK:1.0.1 TASK{KU=U2/V1}
TF=K_U2_V1   FREQ= 0.0000000E+0000        OMEGA= 0.0000000E+0000
      |TF|=       0.50000000      deg(TF)=          0.00000
   Re(TF)= 5.0000000E-0001       Im(TF)= 0.0000000E+0000
*****************************      AMP 8_D      *****************************
TF SENS OF [R1]
F= 0.0000000E+0000      RE[S]=-5.0000000E-0001  IM[S]= 0.0000000E+0000
TF SENS OF [R2]
F= 0.0000000E+0000      RE[S]= 5.0000000E-0001  IM[S]=-0.0000000E+0000
*****************************      AMP 8_D      *****************************

TIME:21:16:53
*****************************      AMP 8_D      *****************************
```

<_m/ex0c.m0> is a directory file with references to data files :

```
TASK:1.0.1 TASK{KU=U2/V1}
FILE: ex0c.m1
DATANAME: K_U2_V1
DATA: [1..1,1..3][F,RE(TF),IM(TF)]
EOF
TASK:1.0.1 TASK{KU=U2/V1}
FILE: ex0c.m2
DATANAME: S_K_U2_V1_R1
DATA: [1..1,1..3][F,RE(S(TF)),IM(S(TF))]
EOF
TASK:1.0.1 TASK{KU=U2/V1}
FILE: ex0c.m3
DATANAME: S_K_U2_V1_R2
DATA: [1..1,1..3][F,RE(S(TF)),IM(S(TF))]
EOF
```

<_m/ex0b.m1> is data file with TF : {FREQUENCY, RE(TF), IM(TF)

```
 0.00000000000000E+0000   5.00000000000000E-0001   0.00000000000000E+0000
```

<_m/ex0b.m2> is data file with TF : {FREQUENCY, RE(S(TF)), IM(S(TF))

```
 0.00000000000000E+0000  -5.00000000000000E-0001   0.00000000000000E+0000
```

<_m/ex0b.m3> is data file with TF : {FREQUENCY, RE(S(TF)), IM(S(TF))

```
 0.00000000000000E+0000   5.00000000000000E-0001  -0.00000000000000E+0000
```

<_r/ex0c.r1> is a **Nutmeg** file:

```
Title: Amp AC analysis results for Tasks = (1.0.* ):TASK{KU=U2/V1}
Date:1/1/2015 at 21:16:53
Plotname: AC Analysis
Flags: complex
Sckt. naming: from bottom to top
No. Variables: 4
No. Points: 1
Variables:
      0     frequency   frequency
      1     K_U2_V1     notype
      2     S_K_U2_V1_R1      notype
      3     S_K_U2_V1_R2      notype
Values:
0      0.00000000000000E+0000, 0.00000000000000E+0000
       5.00000000000000E-0001, 0.00000000000000E+0000
      -5.00000000000000E-0001, 0.00000000000000E+0000
       5.00000000000000E-0001,-0.00000000000000E+0000
```

2. Examples.

Example1 - Lna

Summary: s-parameters, ideal directional coupler, verification of calculated against actual s-parameters.

Example2 - Active splitter

Summary: s-parameters test network, sweep of s-parameter for a model, sweep of part values..

Example3 - Stability analysis.

Summary: stability factors, stability circles, frequency models.

Example4 - Coupled transmission lines

Summary: differential line, coupled line modeling, even-odd modes, decoupling transformations.

Example5 - NF for cascade of LNA and tuner.

Summary: NF - noise figure, noise modeling of active device, noise model of cascade.

Example6 - RF passive filter optimization.

Summary: LC filter optimization , pattern search (Hooke-Jeeves algorithm), preferred numbers, Monte Carlo, sensitivities.

2.1. LNA.

Summary: s-parameters, ideal directional coupler, verification of calculated against actual s-parameters.

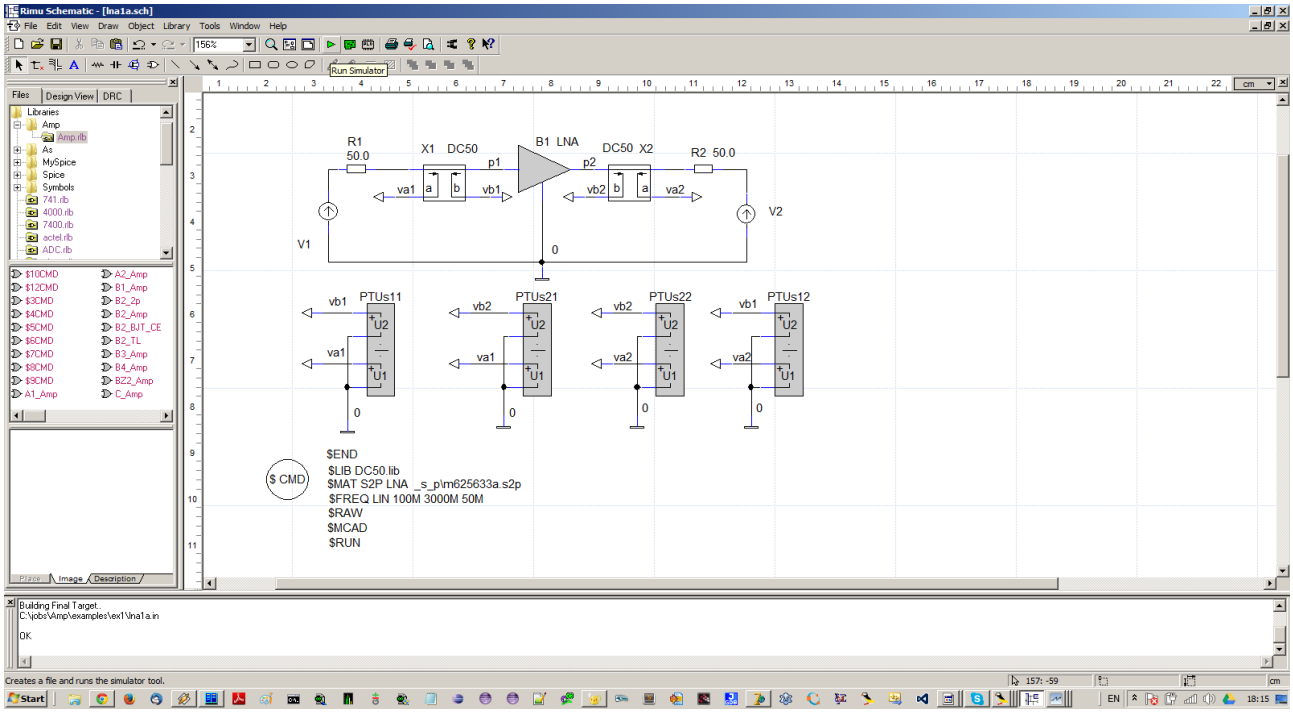


Fig.2.1.1. lna1.sch - schematics.

The lna1.sch circuit consists of two ideal directional couplers and LNA in 'characterization' configuration.

After pressing **'Run Simulator'** – circuit description file **lna1.in** is created and **AMP** is executed. **AMP** in turn invokes **spice-opus nutmeg** and **scilab** script to post process and display results.

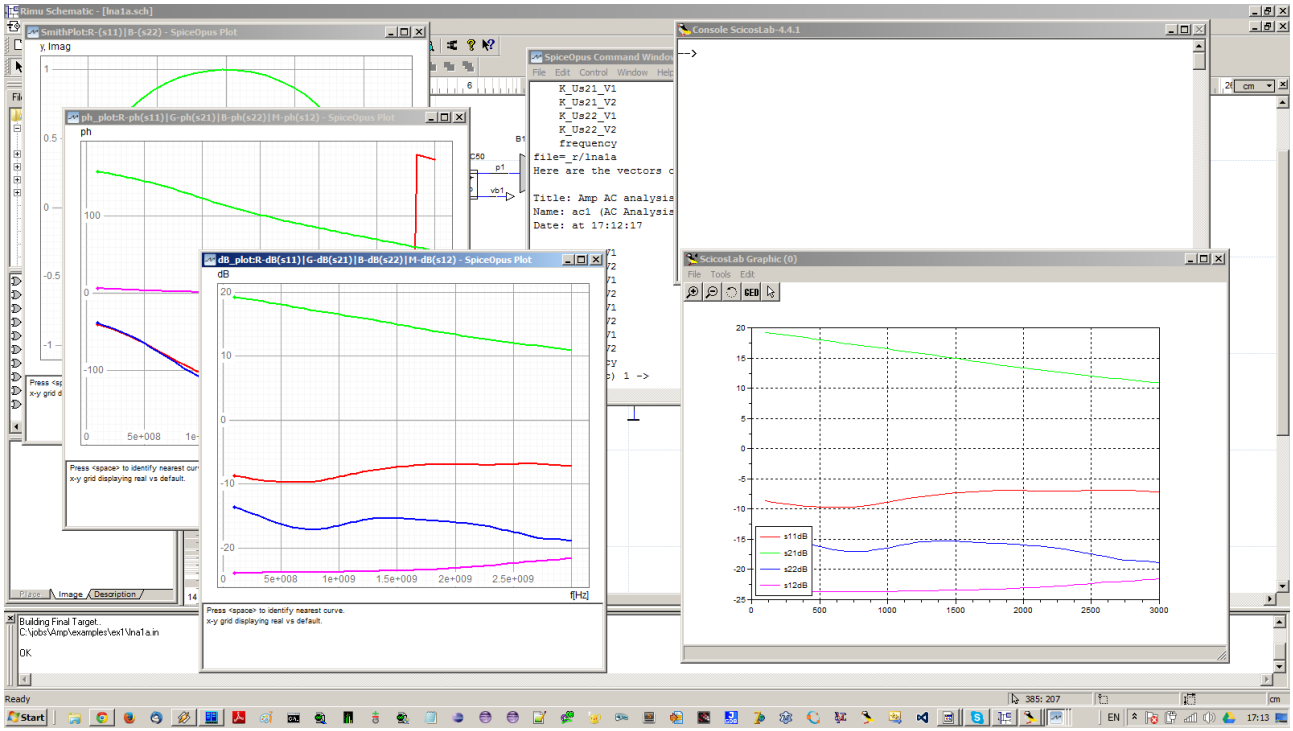


Fig.2.1.2. Results of lna1.sch analysis.

There are eight s-parameters models for different bias current, ranging
from: m625633a.s2p – for bias =10mA
to: m625633h.s2p – for bias =70mA.
By changing reference to the s-parameter model, new results can be easily calculated as shown below in figure 2.1.3.

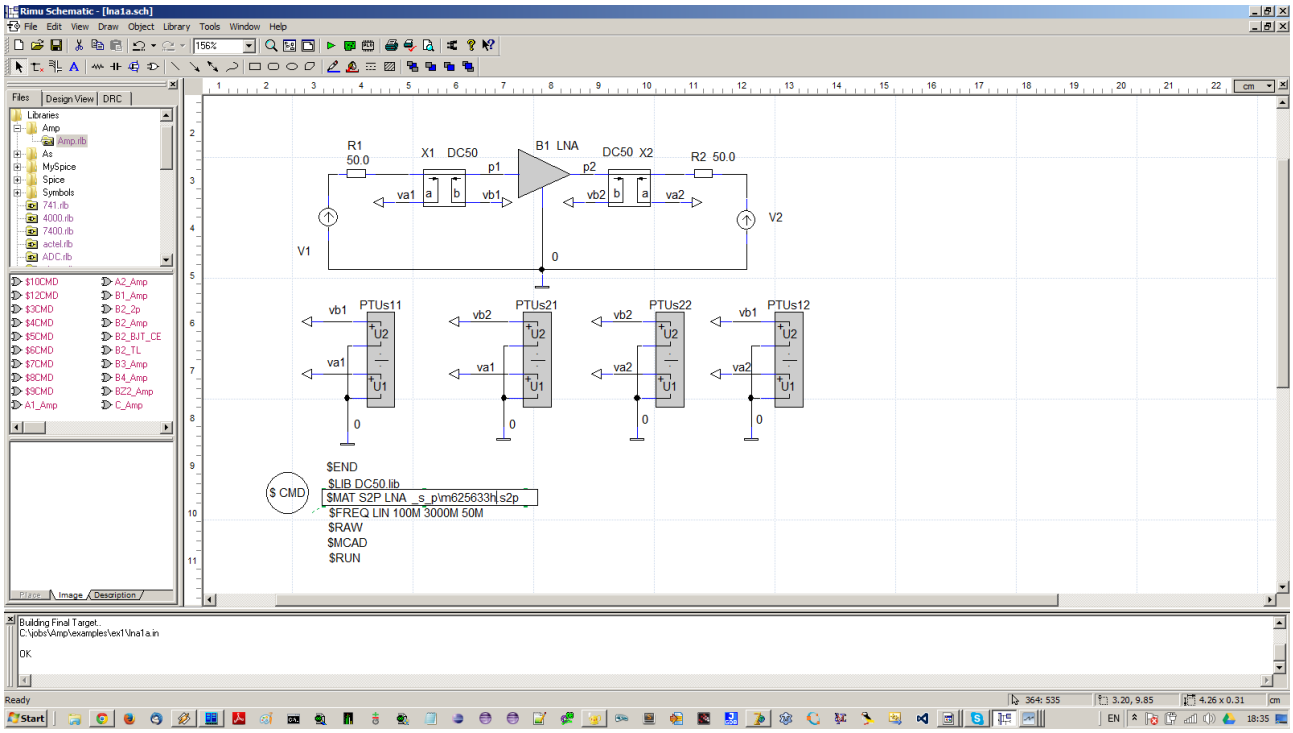


Fig2.1.3. Lna1a.sch – with s-parameters for bias = 70mA.

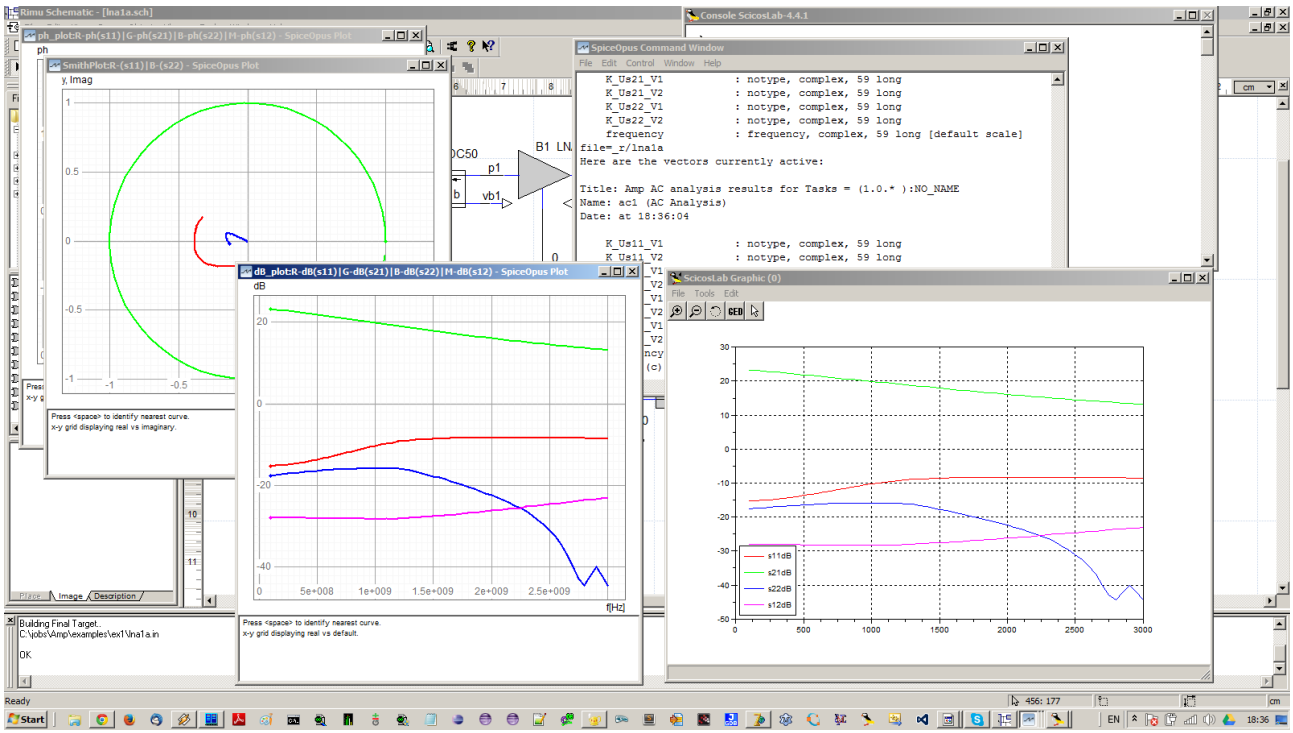


Fig2.1.4. Lna1a.sch – with s-parameters for bias = 70mA.

To verify results of analysis **scilab ex1.sce** script is used. It is located in **sce/** folder. Scripts reads s-parameters data and compares them with data files produced by **AMP**.

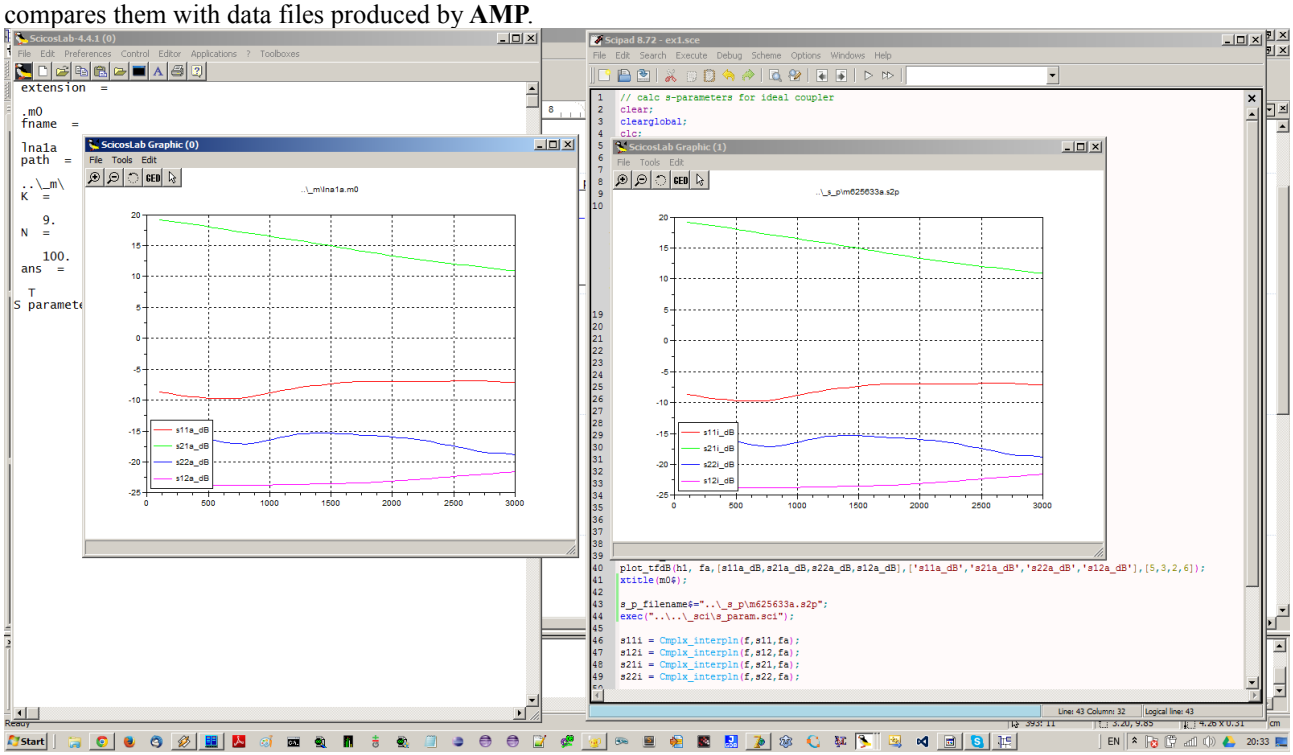


Fig2.1.5. Calculated versus actual for m625633a.s2p model (bias = 10mA).

Data are aligned, as expected.

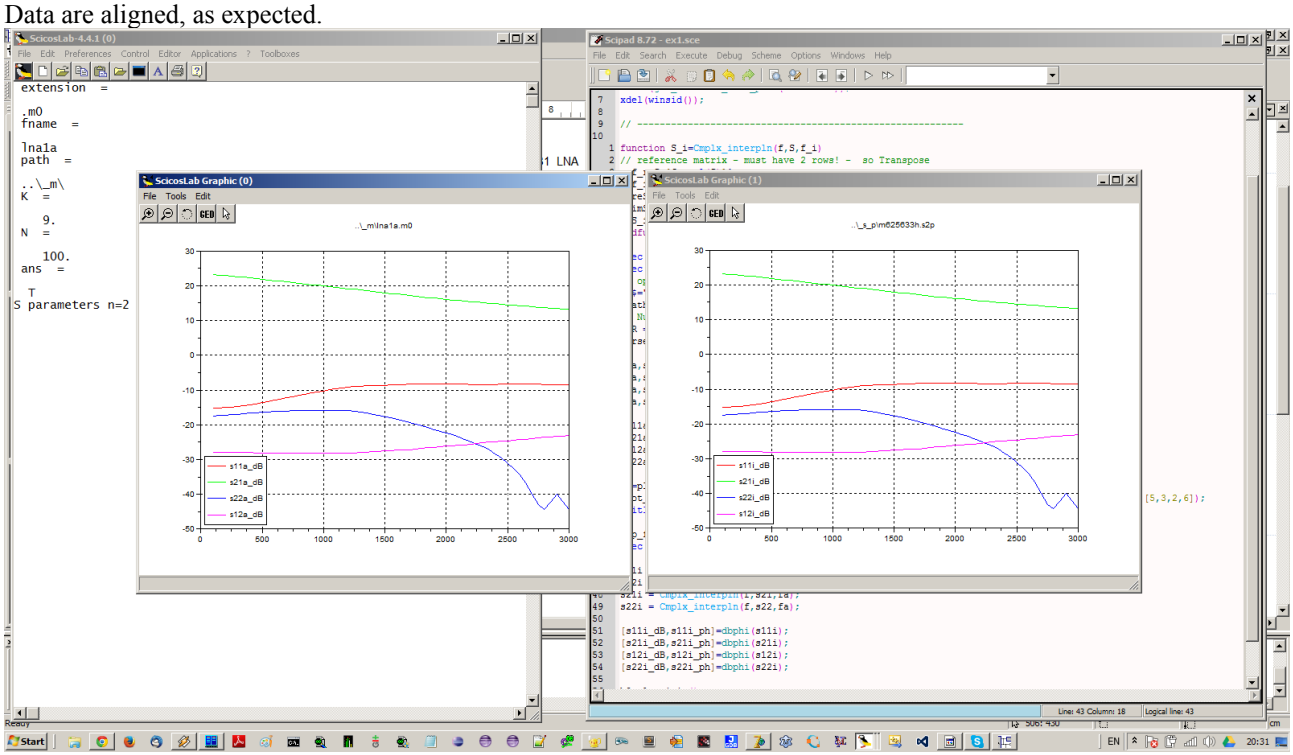


Fig2.1.6. Calculated versus actual for m625633h.s2p model (bias = 70mA).

2.2. Active splitter.

Summary: s-parameters test network, sweep of s-parameter for a model, sweep of part values.

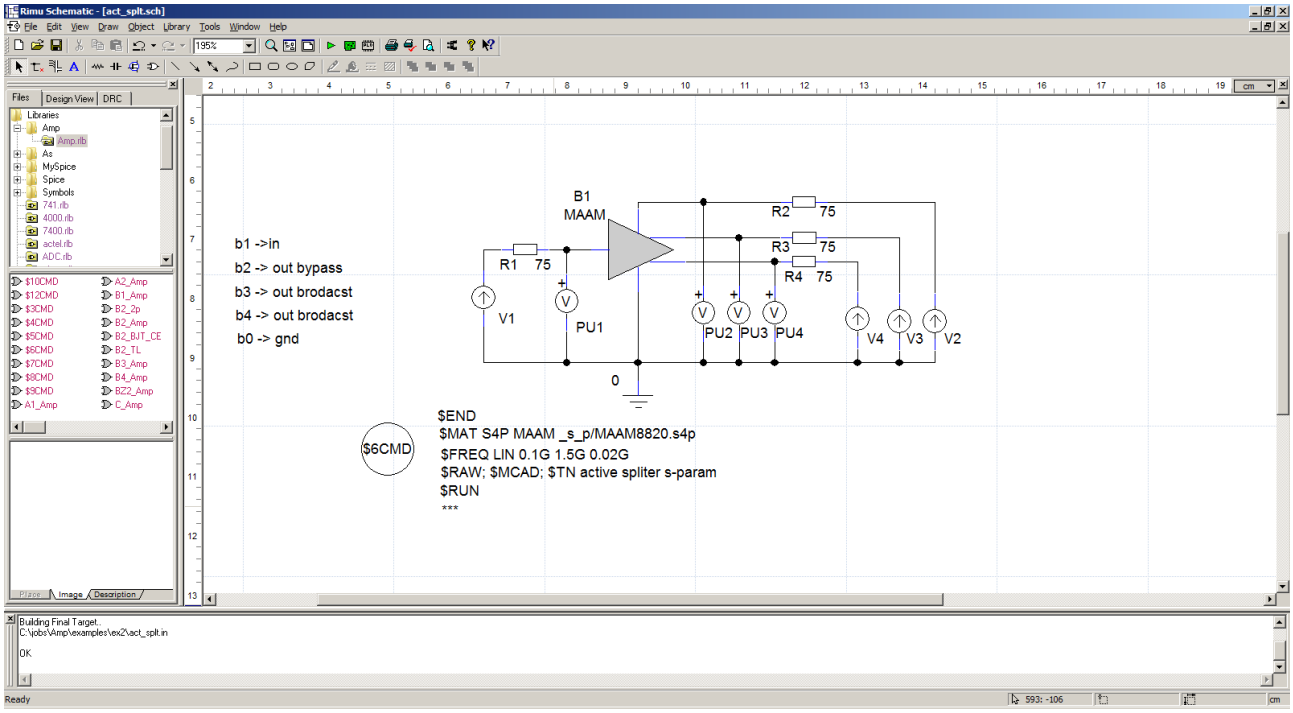


Fig2.2.1. Active splitter in 4 port characterization configuration <<act_split.sch>>**.**

Alternate method of calculating s-parameters is used in the DUT (device under test) test network. DUT is an active splitter with 1 input and 3 outputs. DUT is driven from multiple sources with internal impedance equal with reference impedance **Ro=75** for the ports**.** DUT's s-parameters are derived from voltage gains.

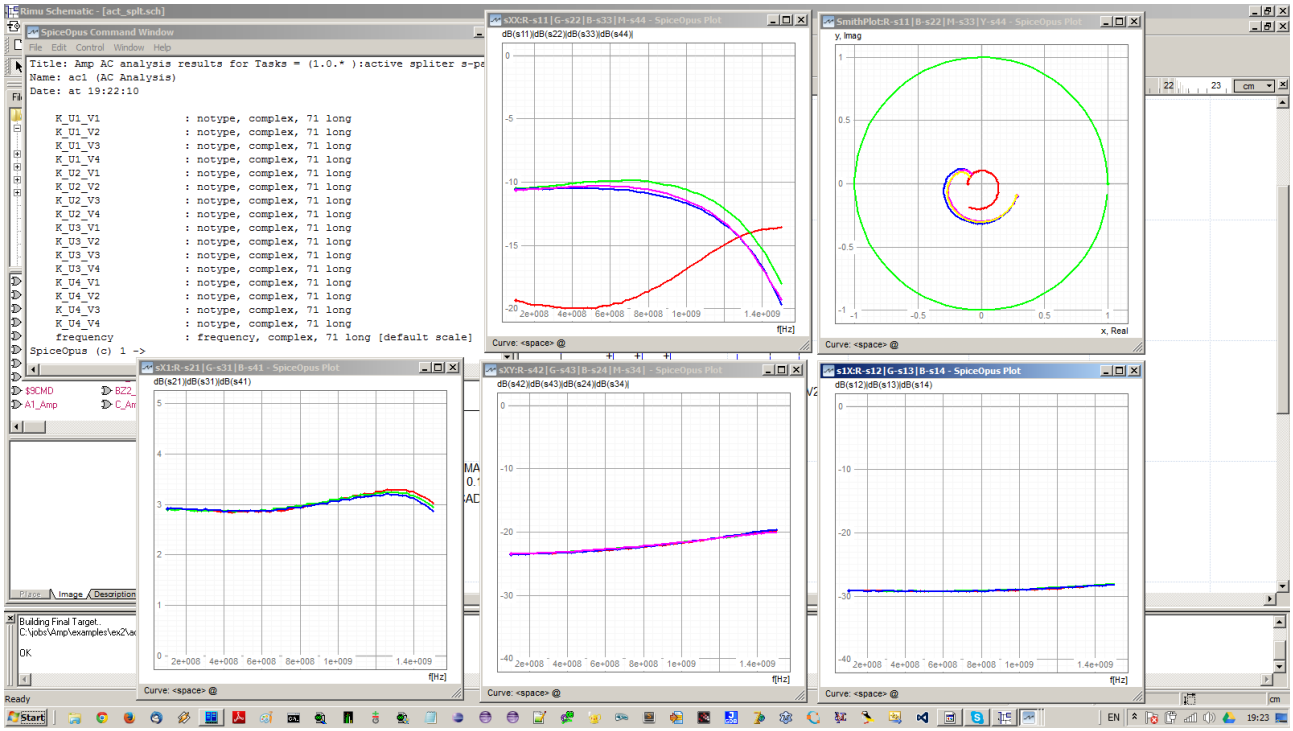


Fig2.2.2. Few of active splitter s-parameters calculated and plotted by <<act_split.nut>> script**.**

Calculated and actual s-parameters might be compared by using **scilab** script.

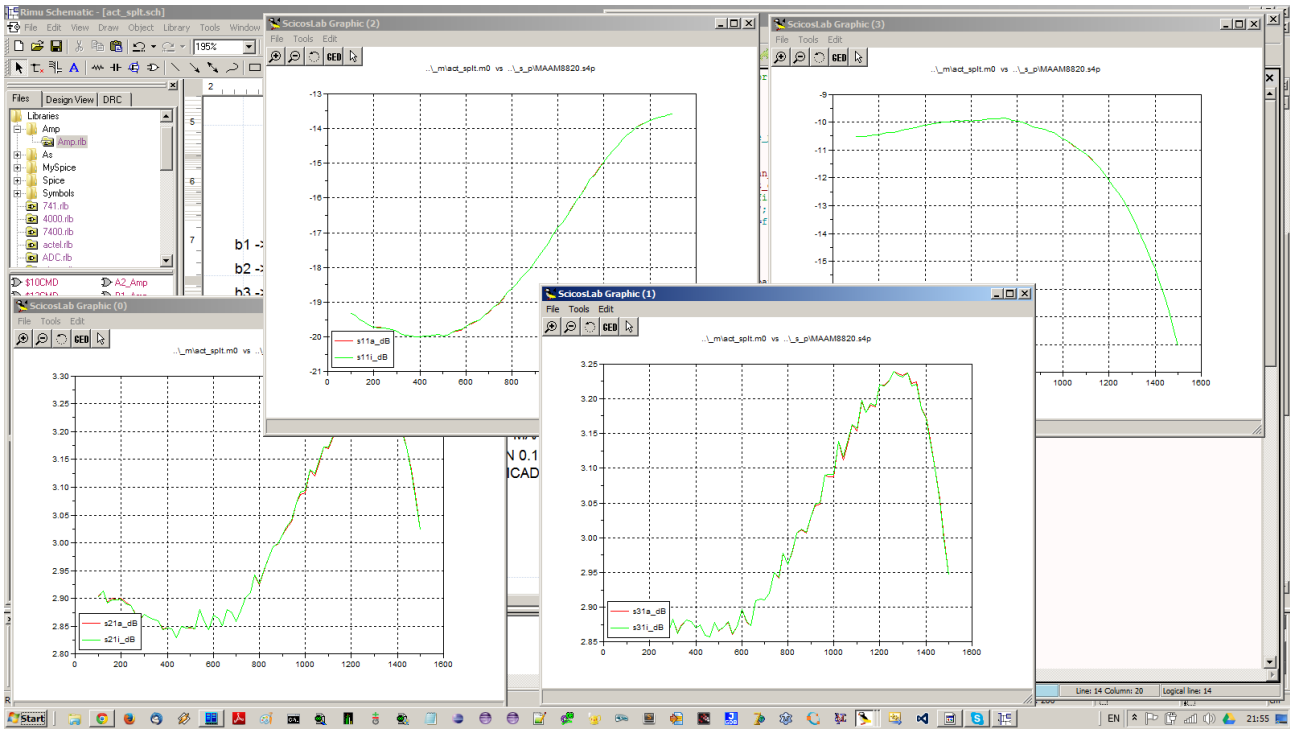


Fig2.2.3. Active splitter s-parameters compared with actual by scilab <<sce/ex2.sce>> script **.**

In order to compare two s-parameters of two different devices, two runs of **AMP** are required. In the next example, simplified characterization circuit is used, which allows user to calculate the sub-set of s-parameters, namely s11, s21, s31, s41. In the first run active splitter model is defined by MAAM8820.s4p file, in the second run by MAAM8822.s4p.

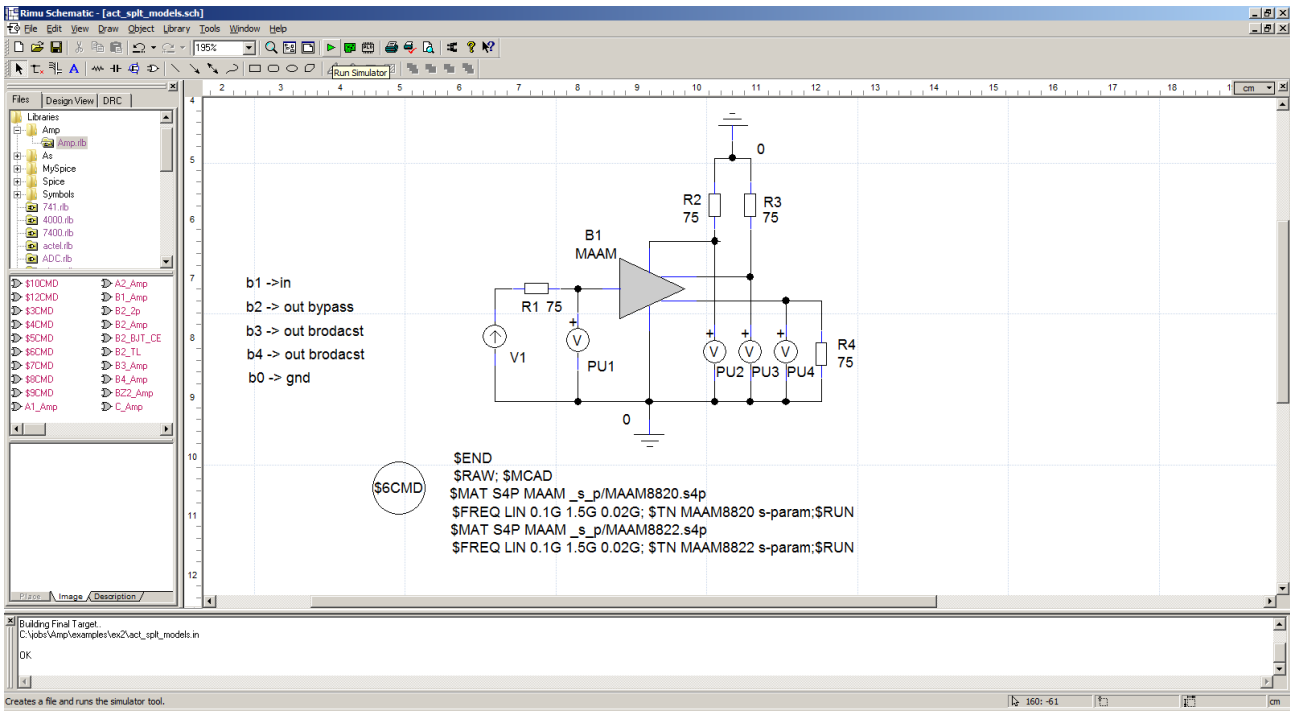


Fig2.2.4. Configuration for comparing s-parameters of two devices <<act_split_models.sch>> .

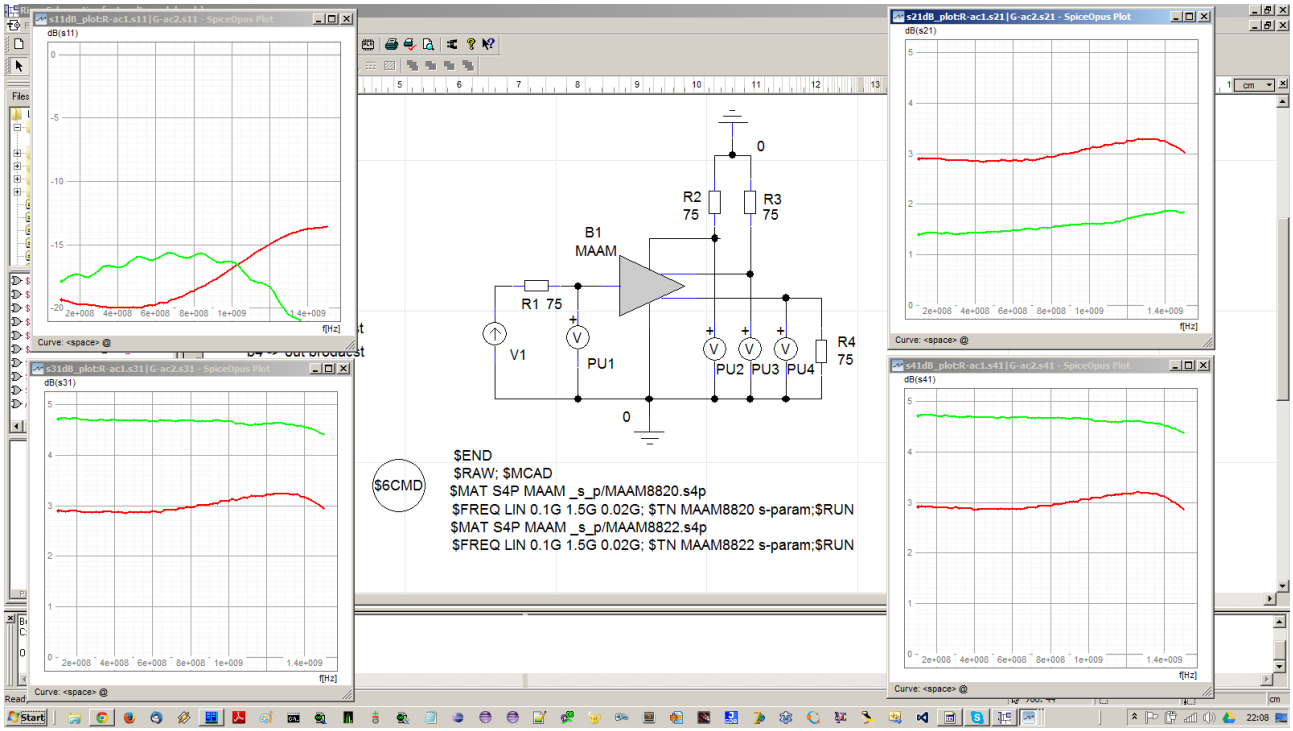


Fig2.2.5 Some of active splitter s-parameters compared and plotted by <<act_split_models.nut>> script**.**

Sometimes, simple sweep of parameters is required. Next example shows how this might be done.
Two parameters are swept, values of capacitor **C1** and coil **L1**. When **$PAR** command is used, all lists are stepped simultaneously, therefore same number of lists elements is required. The first run is performed for **C1**=100nF, **L1**=0.1pH, second for **C1**=56pF, **L1**=1.0nH, third for **C1**=56pF, **L1**=3.3nH, and so on.
S-parameters are calculated for ports identified by voltage probe nodes. Since there are 2 probes and 1 source, only s11 and s41 are calculated.

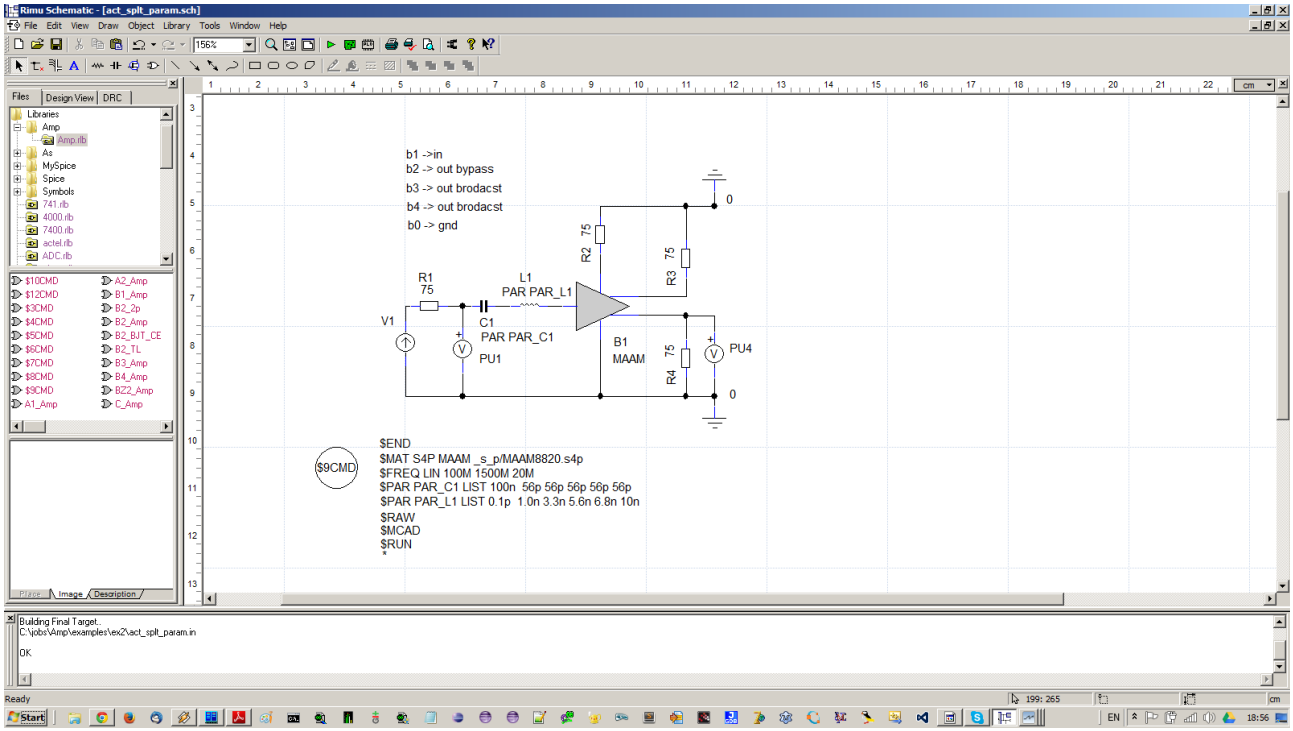


Fig2.2.6. Active splitter with two parameters sweeping <<act_split_models.sch>>**.**

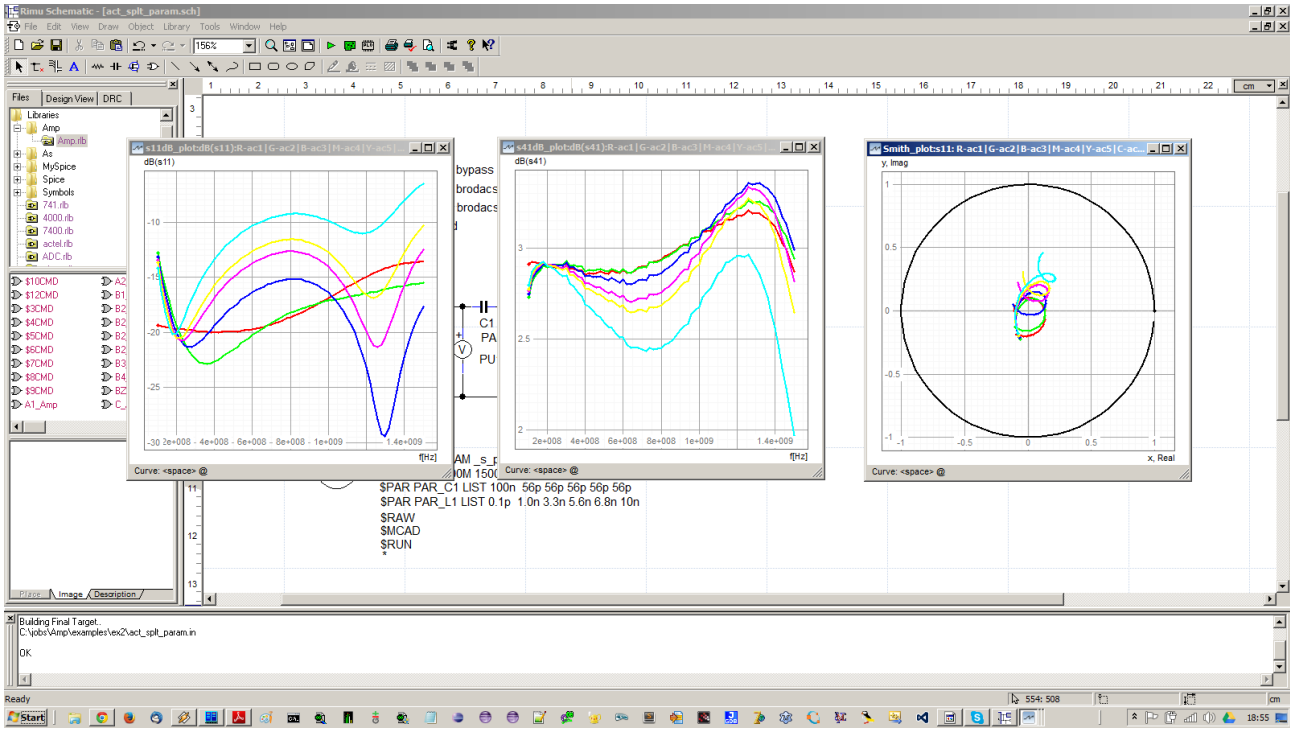


Fig2.2.7 s11,s41 with parameters sweep plotted by <<act_split_params.nut>> script**.**

AMP produces separate raw data file for each run. Each analysis is identified by **ac** prefix in 'nutmeg' script. So, first run for **C1**=100nF, **L1**=0.1pH is identified as **ac1**, second for **C1**=56pF, **L1**=1.0nH as **ac2**, third for **C1**=56pF, **L1**=3.3nH as **ac3**, an so on. Colors are declared inside '**nutmeg**' script. First run (**ac1**) is plotted as red line, second (**ac2**) as green, third (**ac3**) as blue, fourth (**ac4**) as magenta, fifth (**ac5**) as yellow, sixth (**ac6**) as cyan.

2.3. Stability analysis.

Summary: stability factors, stability circles, frequency models.

Model of single transistor gain stage is checked to be unconditionally stable. The transistor is linear 'hybrid pi' model with parameters calculated for Ic=30mA. The analysis includes Rollett's stability factors (**K**,**B1**) and stability circles.

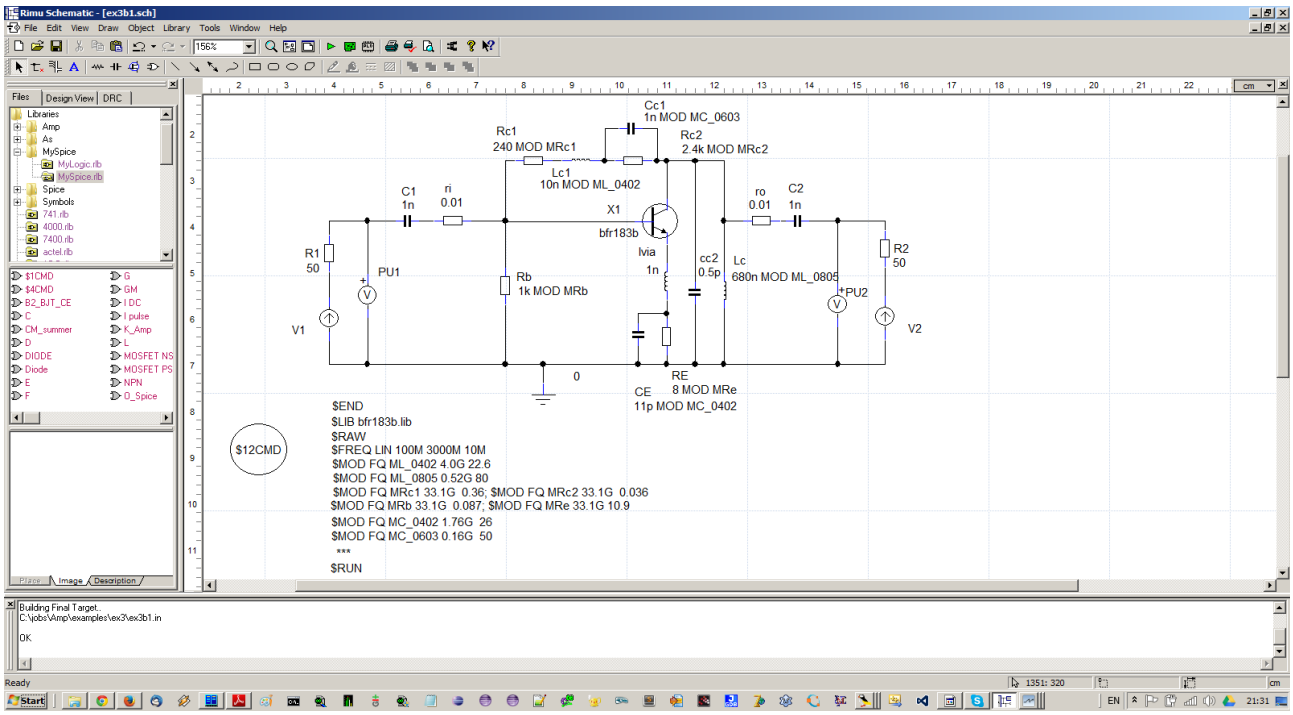


Fig2.3.1. Gain stage in 4 port characterization configuration <<ex3b1.sch>>.

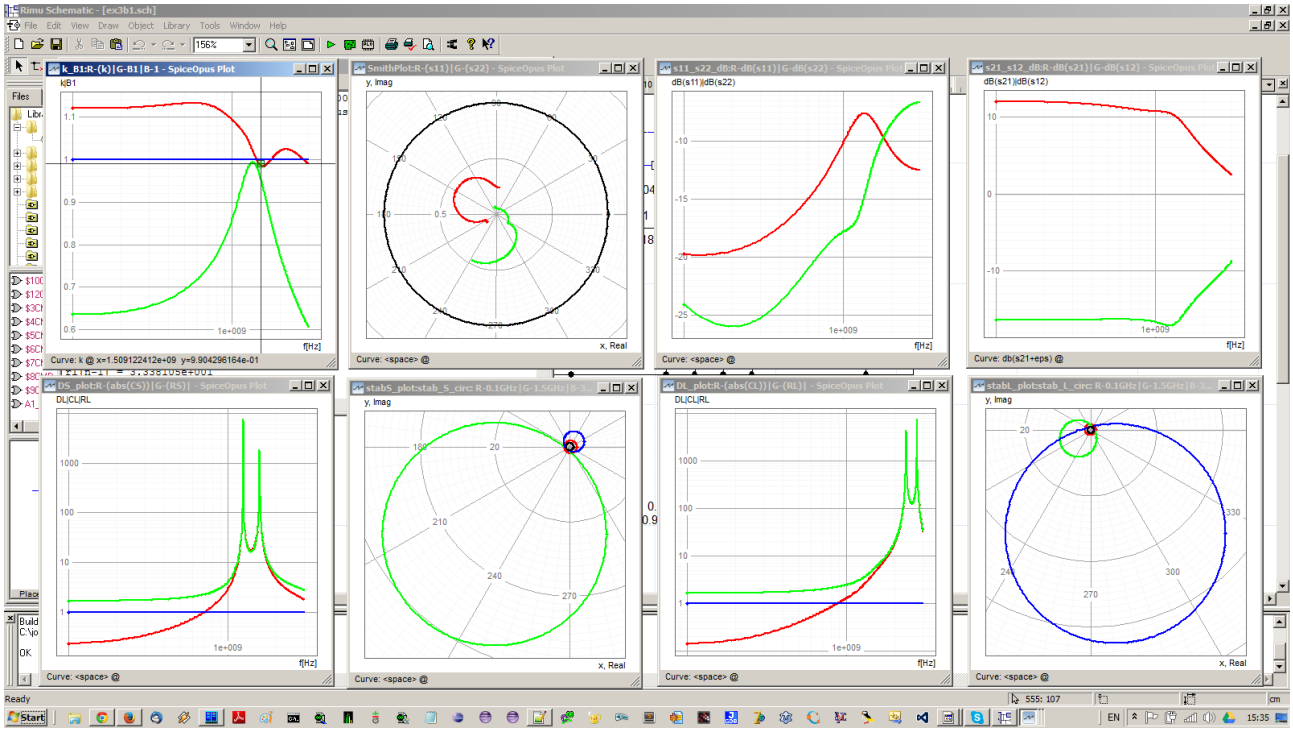


Fig2.3.2. Stability analysis for gain stage <<ex3b1.nut>>.

For clarity only 3 stability circles, for frequency **0.1GHz**, **1.5GHz**, **3.0GHz** are plotted, yet the variation of stability circle radius (**RS,RL**) and distance of the center (**CS,CL**) from the origin of smith chart is shown in graphs adjacent to stability circles plots.
Rollet's stability criteria indicate, that amplifier might be potentially unstable in the range between **1.4GHz** and **1.7GHz** and above **2.6GHz**. The nature of the problem is better understood by inspection of load and source stability circles.

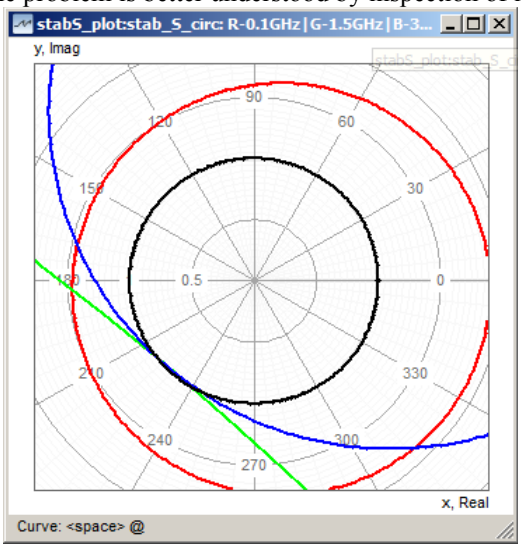


Fig2.3.3. Source stability circles for ex3b1 model.

Indeed, for and $\Gamma_S \approx |1|e(j240^o)$ at **1.5GHz** (green circle) and at **3.0GHz** (blue circle) , output reflection coefficient is greater than unity $|\Gamma_{out}| > 1$.

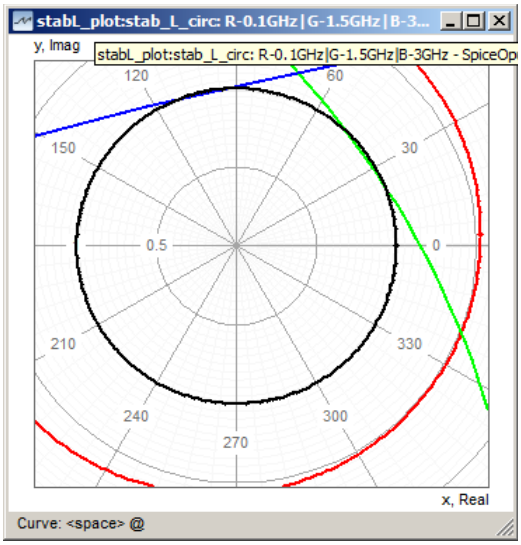


Fig2.3.4. Stability load circles for ex3b1 model.

Moreover, for $\Gamma_L \approx |1|e(+j45^o)$ at **1.5GHz** (green circle), and $\Gamma_L \approx |1|e(+j110^o)$ at **3.0GHz** (blue circle) input reflection coefficient is greater than unity $|\Gamma_{inp}| > 1$.

In order to avoid effects of unfriendly terminations, serial resistors **ri=10 Ohm**, **ro=10 Ohm** were added to input and output port.

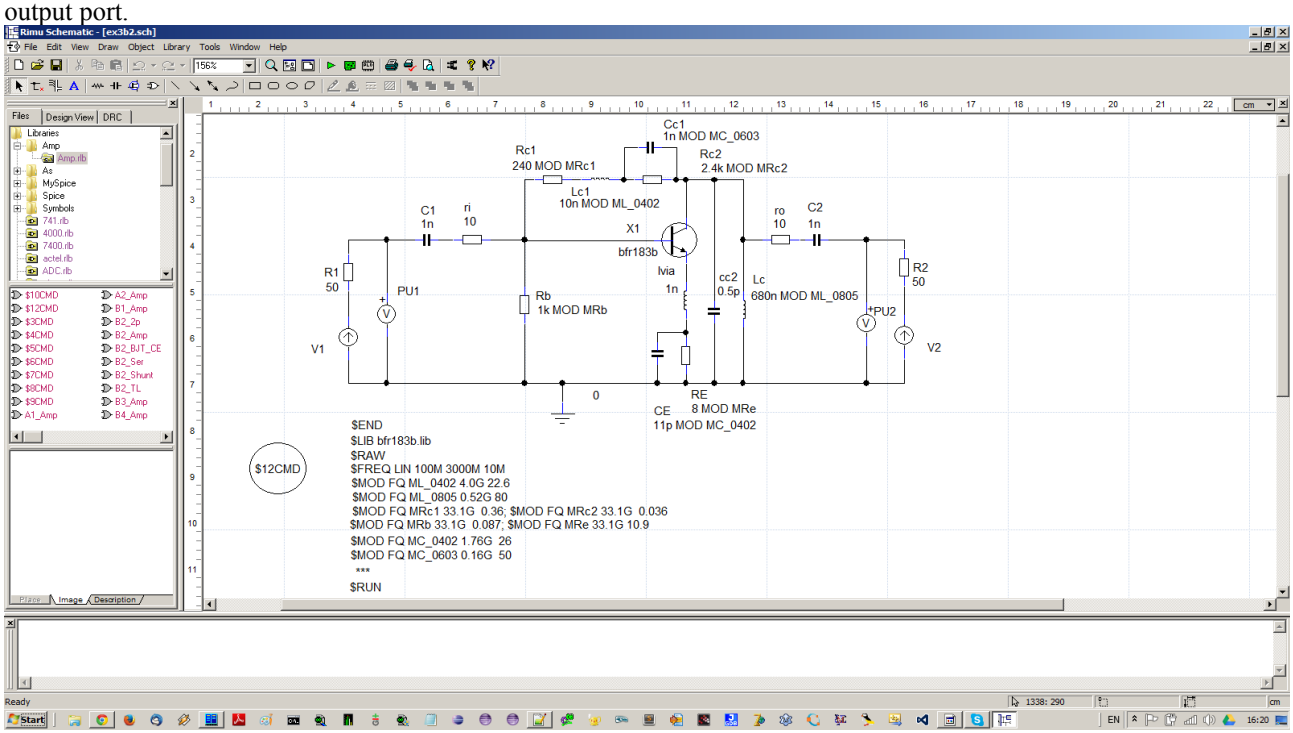


Fig2.3.5. Gain stage with 10Ohm serial resistors <<ex3b2.sch>>.

With added **ri,ro** resistors, stability factors and stability circles are in safe regions, indicating unconditional stability.

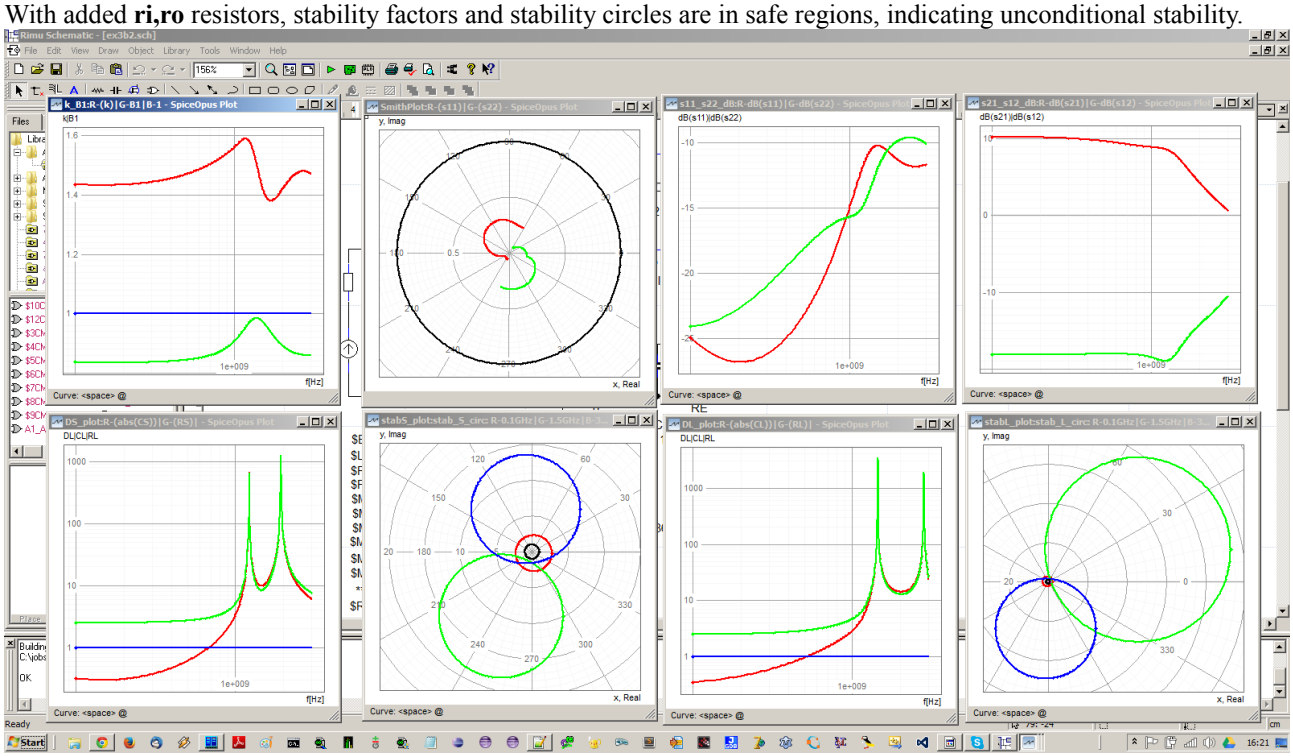


Fig2.3.6. Stability analysis for gain stage with serial resistors <<ex3b2.nut>>.

The solution seems effective, till the point the same analysis is applied to the enhanced model, with transistor and coils modeled by s-parameters.

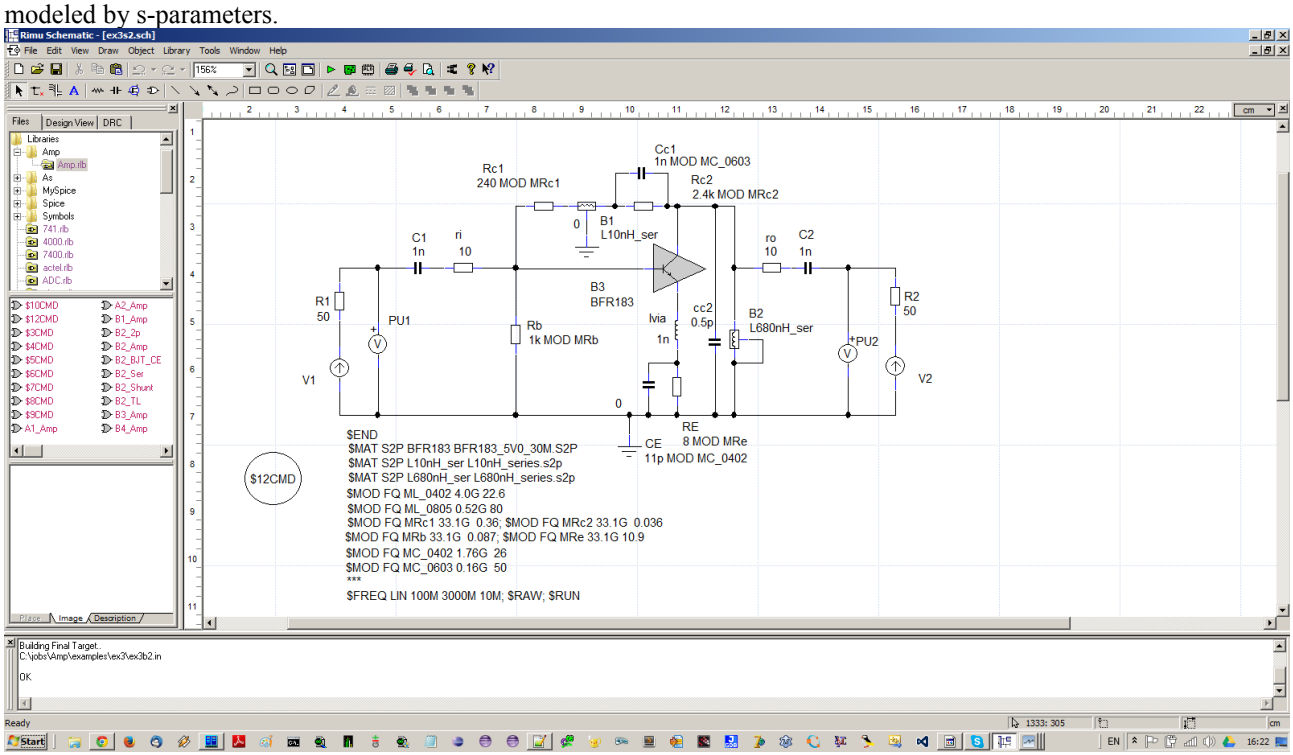


Fig2.3.7. Enhanced gain stage model with 10Ohm serial resistors <<ex3s2.sch>>.

The potential problem around **1.5GHz** is revealed.

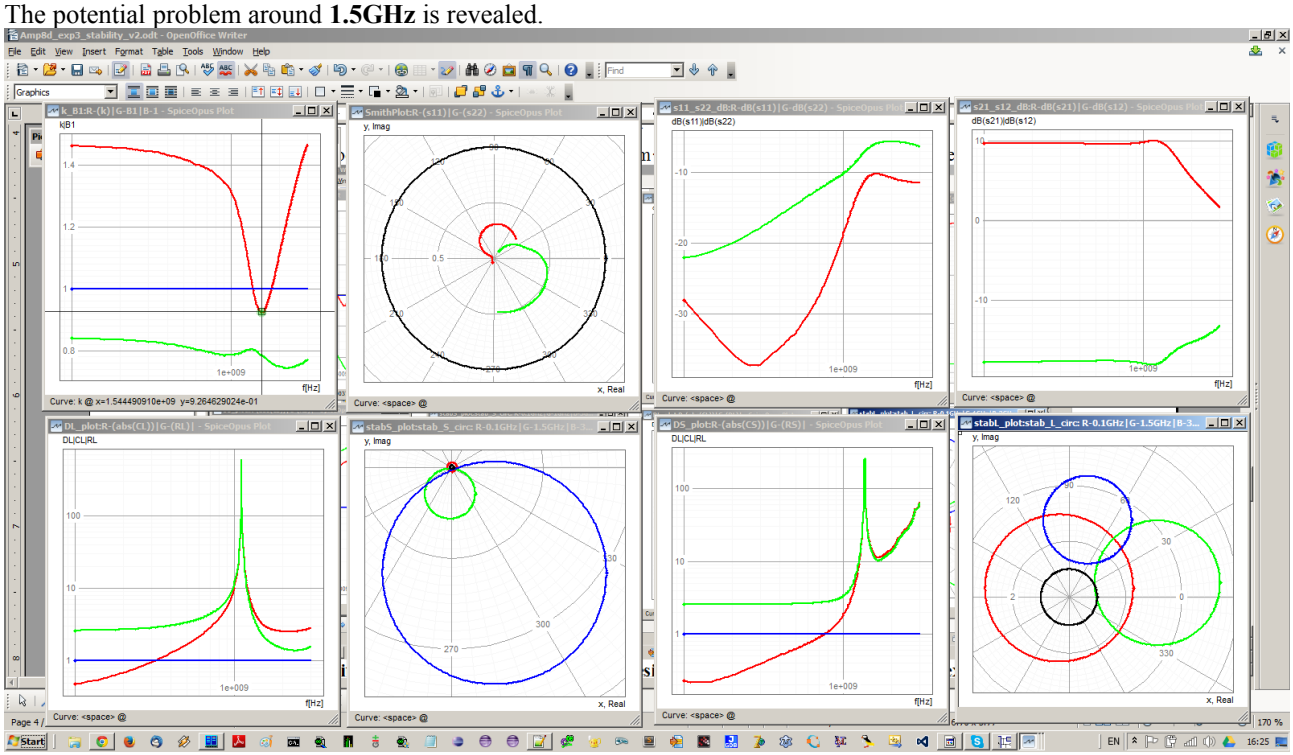


Fig2.3.8. Stability analysis for gain stage with serial resistors and transistor s-parameters <<ex3s2.nut>>.

Enhanced model indicates, that in order to assure unconditional stability, values of serial resistors have to be increased even further.

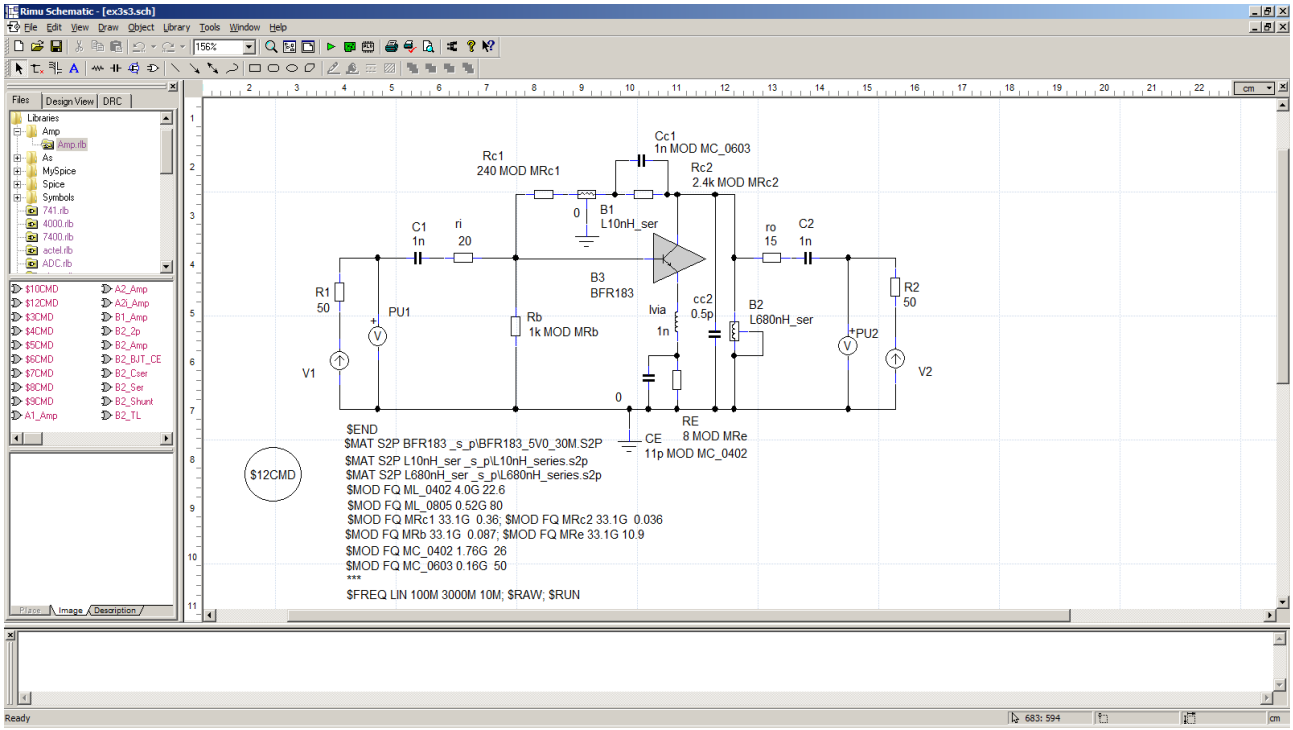


Fig2.3.9. Gain stage with increased value of input/output serial resistors <<ex3s3.sch>>.

The cost of providing unconditional stability is lower gain and higher output return loss.

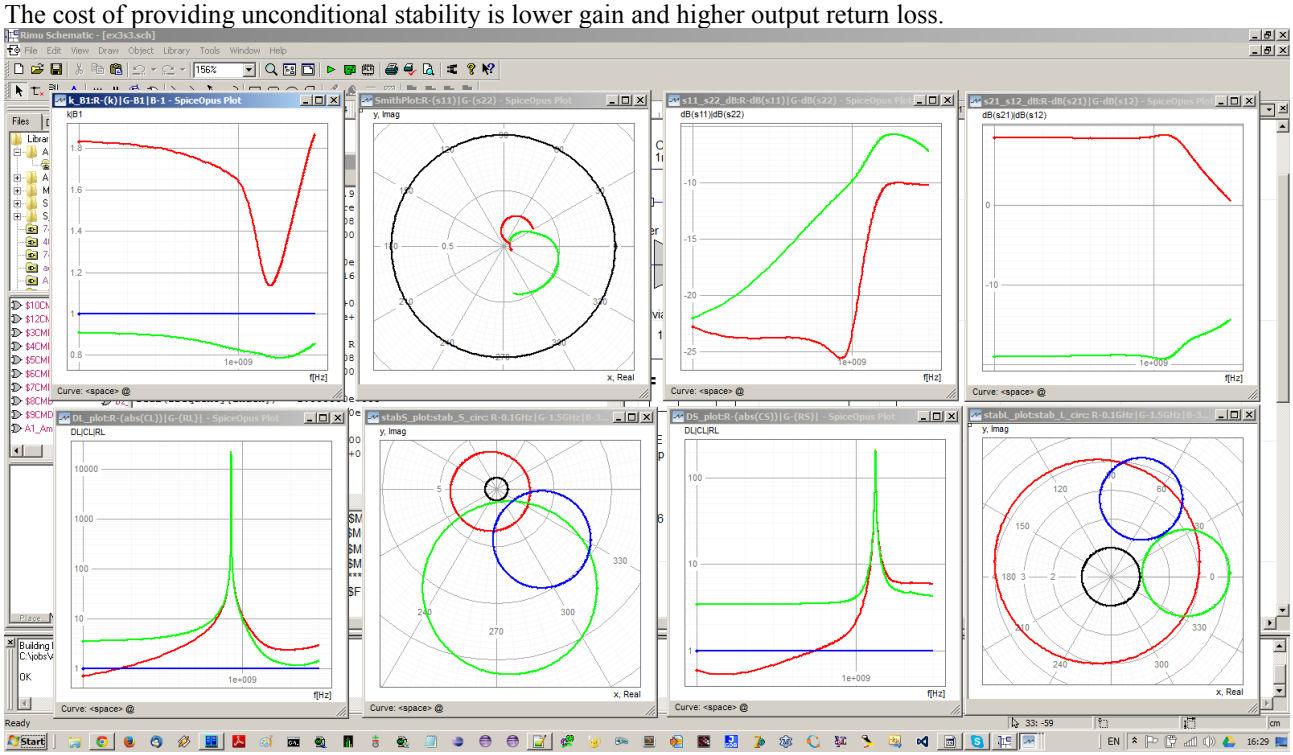


Fig2.3.10. Stability analysis for gain stage with increased value of **ri,ro** resistors <<ex3s3.nut>>.

2.4. Coupled transmission lines

Summary: differential line, coupled line modeling, even-odd modes, decoupling transformations.

The decoupling transformation technique is used to decompose coupled lines into two uncoupled lines.

To estimate parameters of differential transmission line, we use mdtlc.exe - a free 2D impedance calculator, which in turn uses atlc. – free arbitrary transmission line calculator.

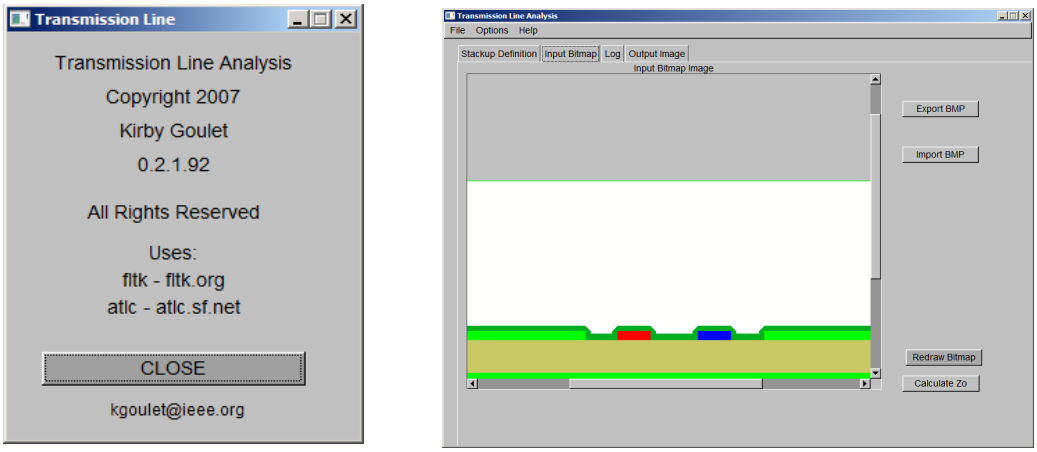


Fig.2.4.1. 2D layout of differential microstrip.

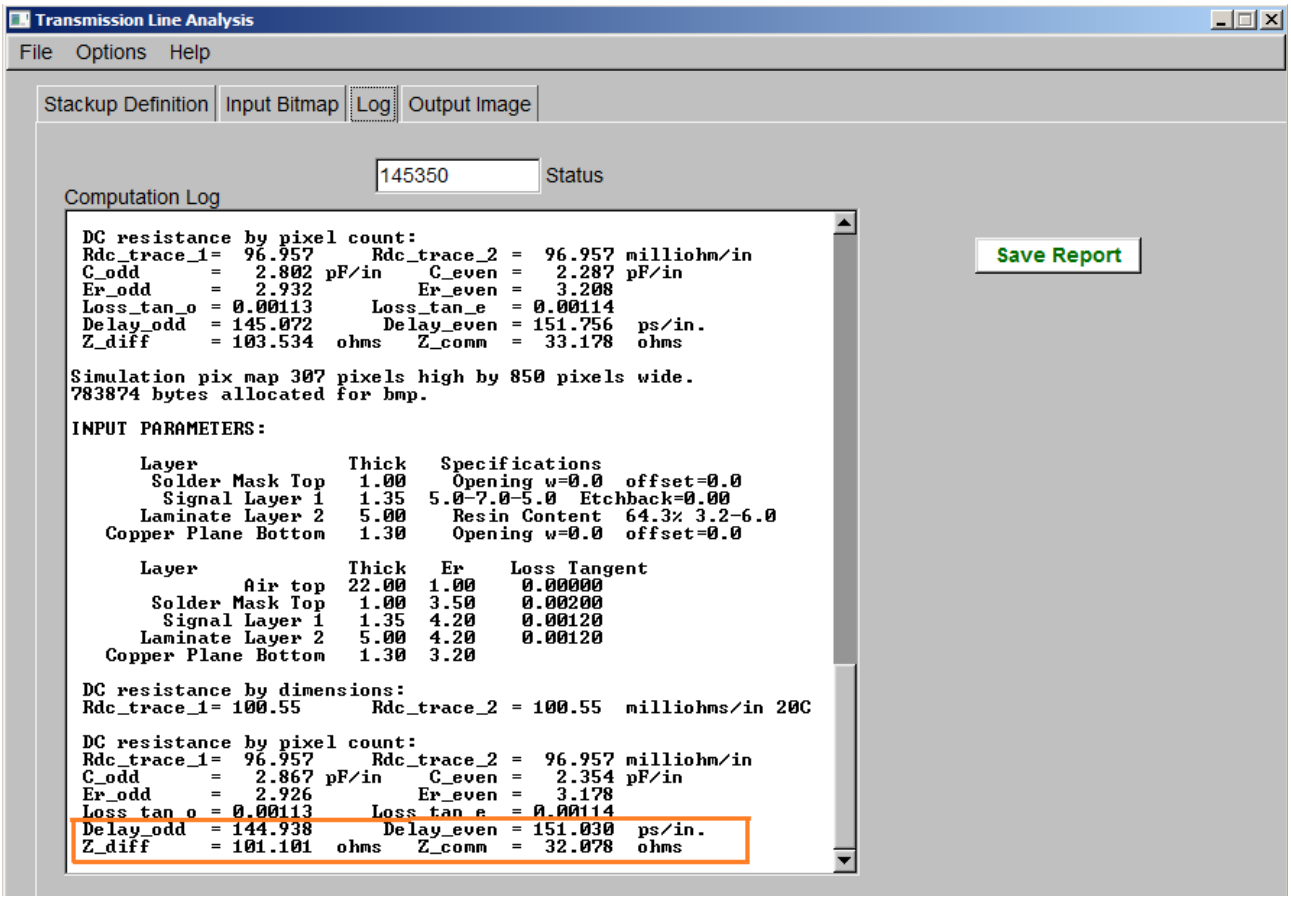


Fig. 2.4.2. Results of impedance analysis of differential microstrip by mdtlc.exe.

In principle, the lossless symmetric differential line might be characterized by four parameters: differential characteristics impedance **Z_diff** and delay **Delay_odd** and common characteristics impedance **Z_comm** and delay **Delay_even**.
The internal schematics of coupled lines macro model is shown in figures 4.2.3, 4.2.4. Coupled lines are decomposed into two uncoupled lines: **T1** - for even and **T2**- for odd mode and two **even-odd** mode converters **XS1**, **XS2**.

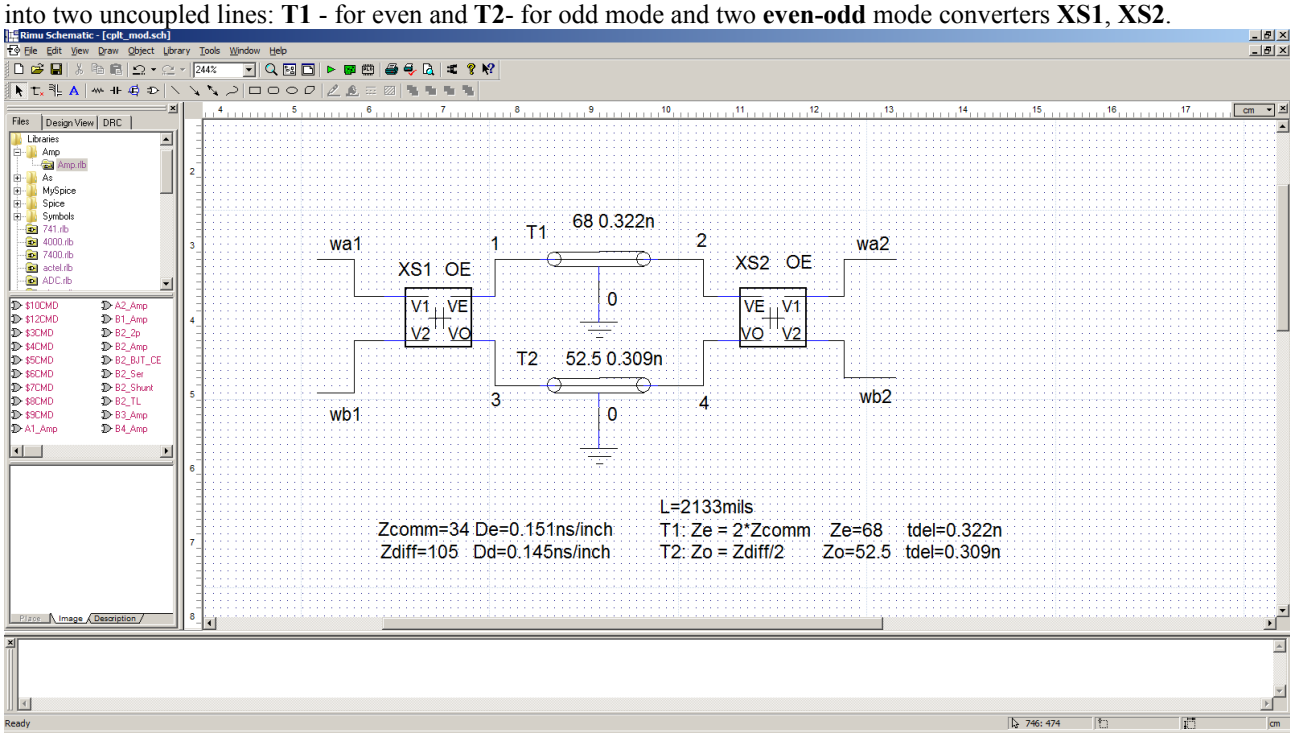


Fig2.4.3. Macro model of coupled transmission lines.

The internal schematic of **even-odd** mode converter is shown in figure 4.2.4.

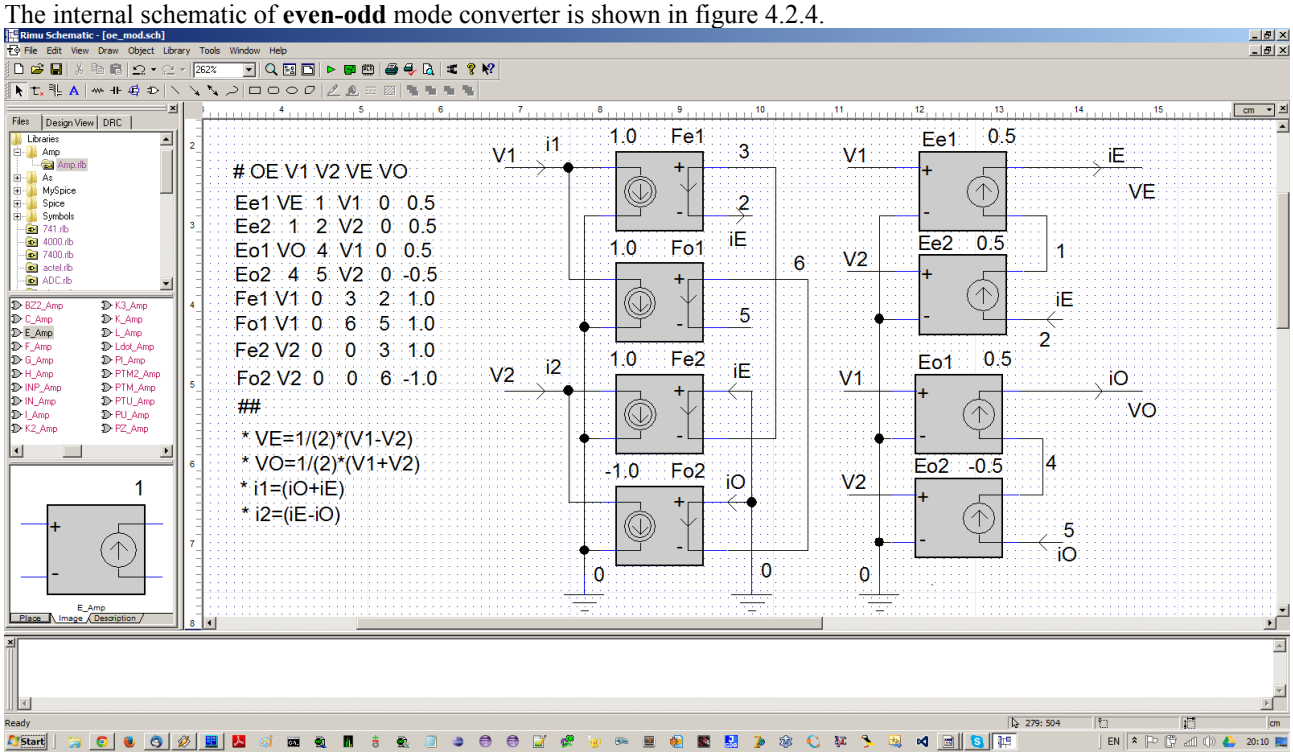


Fig2.4.4. Macro model of even-odd mode converter.

As shown in figure 2.4.5 coupled lines model is in test circuit to verify s-parameters for differential and common mode. It is worth nothing, that for the test circuit reference impedance is 100 Ohm for differential signal and 25 Ohm for common mode signals.

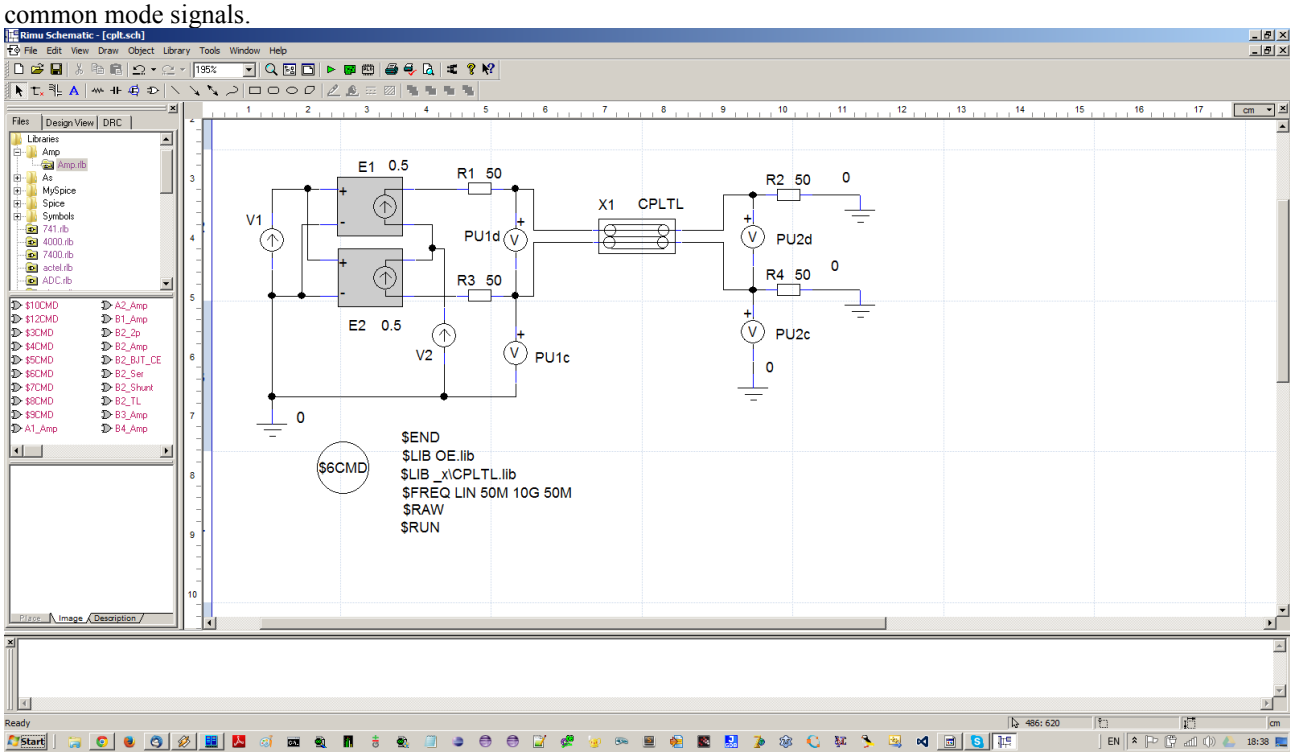


Fig.2.4.5. Characterization of coupled transmission lines.

Nutmeg script calculates s11, s21 for differential and common signals and displays them together with theoretical values. As seen in figure 2.4.6, only one color of plot is visible, meaning that, values calculated for the model match theoretical.

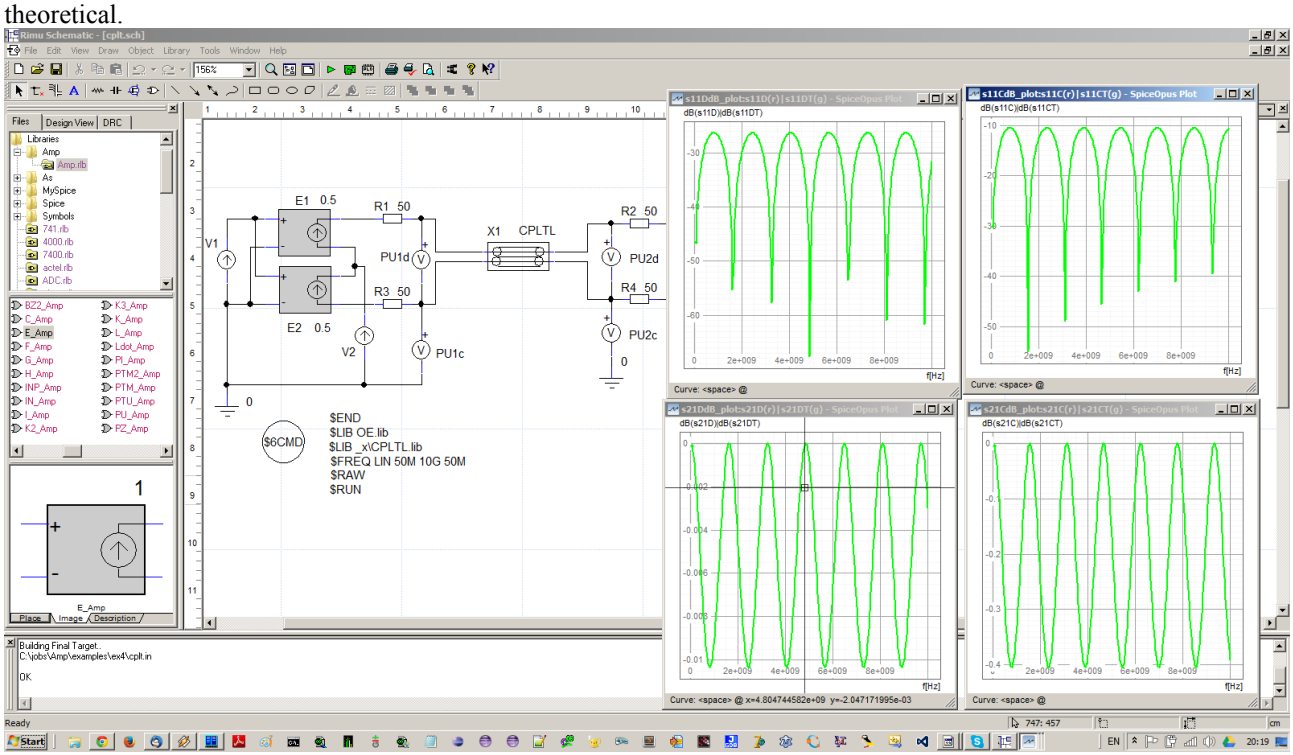


Fig.2.4.6. Results of analysis for s11,s21 for common s11C,s21C and differential s11D,s21D signals.

2.5. NF for cascade of LNA and tuner.

Summary: noise modeling, noise model of cascade, NF - noise figure, impact of source impedance on NF.

The noise model of RF front-end shown in figure 2.5.1. consists of **tuner**, **LNA** and transmission line **X1** plus some input coupling components . Tuner input impedance characterized by 2 port s-parameter set (**B1**) and voltage $\mathbf{V_{nt}}$ and current $\mathbf{I_{nt}}$ noise sources. LNA is characterized by 3 port s-parameter set (**B2**) and voltage $\mathbf{V_{na}}$ and current $\mathbf{I_{na}}$ noise sources. Transmission line model is the same as in the example 2.4.

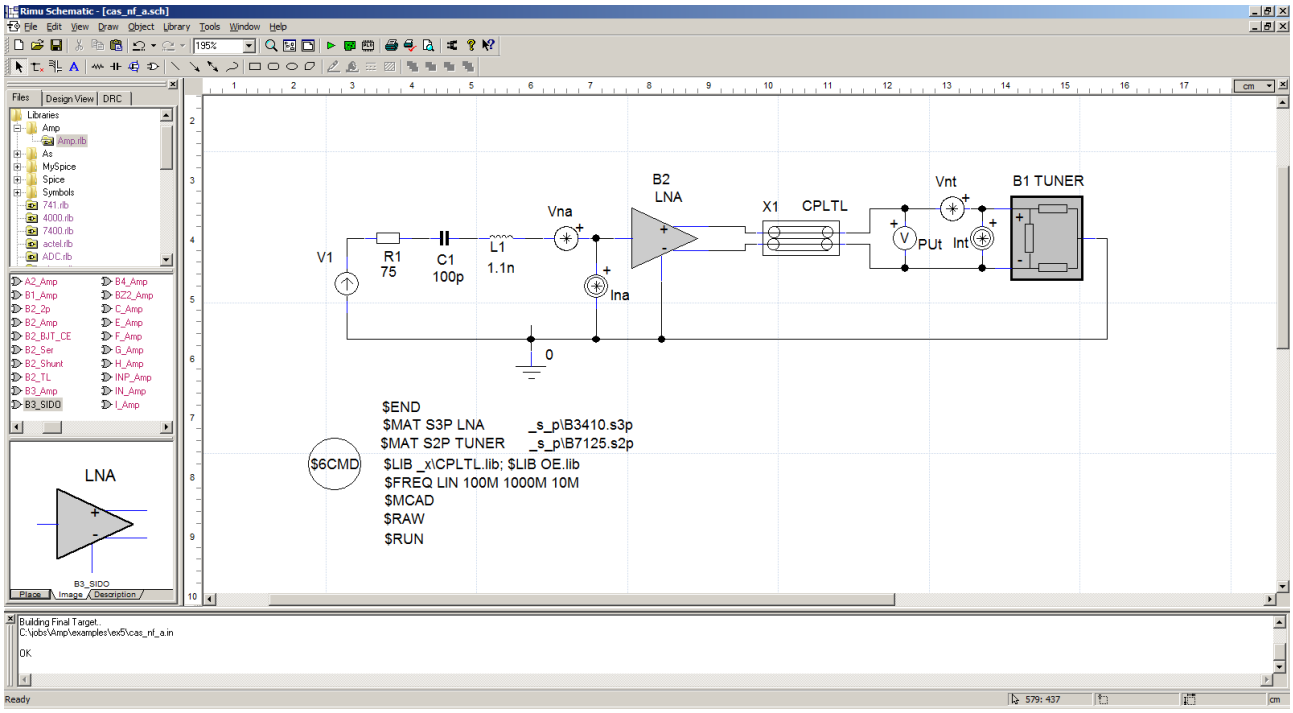


Fig.2.5.1. Noise model of LNA and tuner cascade.

To calculate the noise figure **NF** of the cascade, the ratio of total noise to noise due to the source resistance has to be evaluated at selected reference port. In the example differential input of the tuner **B1** is selected as reference port for noise analysis and **RMS** value of voltage noise is 'measured' by voltage probe $\mathbf{PU_t}$.
Noise sources can be referenced to tuner port as equivalent **Thevenin's** (voltage) or **Norton's** (current) sources. For this reason, **NF** does not depend on loading impedance of reference ports, and formally tuner impedance is redundant in this analysis. Yet, since the result of **NF** analysis does not depend on reference port load impedance, open, short or any load is equivalent as long methodology is consistent. It simply seems more natural to perform the analysis with reference port loaded as in actual circuit, rather than perform calculations for open or short circuit.
Noise characterization embedded in **'Touchstone'** file is represented by 3 parameters $\mathbf{NF_{min}}$, $\mathbf{\Gamma_{opt}}$, $\mathbf{R_n}$ for each frequency. $\mathbf{NF_{min}}$ is minimum noise figure, achievable for optimal complex source reflection coefficient ($\mathrm{mag}(\mathbf{\Gamma_{opt}})$, $\arg(\mathbf{\Gamma_{opt}})$).
Equivalent noise resistance $\mathbf{R_n}$ represents the voltage noise noise contribution.
Native **'Touchstone'** noise characterization set is not convenient for circuit analysis and **AMP does not accept noise data embedded in s-parameters files**.
Therefore, to perform noise analysis, noise data has to be moved to separate files (in the example files have ***.noi** extension) and processed along with transfer functions results by dedicated **scilab** script.
From circuit analysis perspective, noise characterization parameters should directly describe noise sources – as in $\mathbf{S_{vv}=4*kTR1}$ for voltage power density of source resistance **R1**. Accordingly, active devices might be fully characterized by power noise density $\mathbf{S_{vv}}$ for equivalent voltage noise, power noise density $\mathbf{S_{ii}}$ for equivalent current noise density, and complex correlation admittance **Yc=Gc+jBc** between voltage and current sources.
The procedure of converting **'Touchstone'** noise parameters to noise source characterization and calculation of power noise density at voltage probe due to lna, **tuner**, **lna**, and source resistance **R1** is performed by **scilab** script.
Both for **LNA** and **tuner** scilab script reads $\mathbf{NF_{min}}$, $\mathbf{\Gamma_{opt}}$, $\mathbf{R_n}$ noise parameters from **'*.noi'** files and converts them into noise source parameters $\mathbf{S_{vv}}$, $\mathbf{S_{ii}}$, **Yc**.

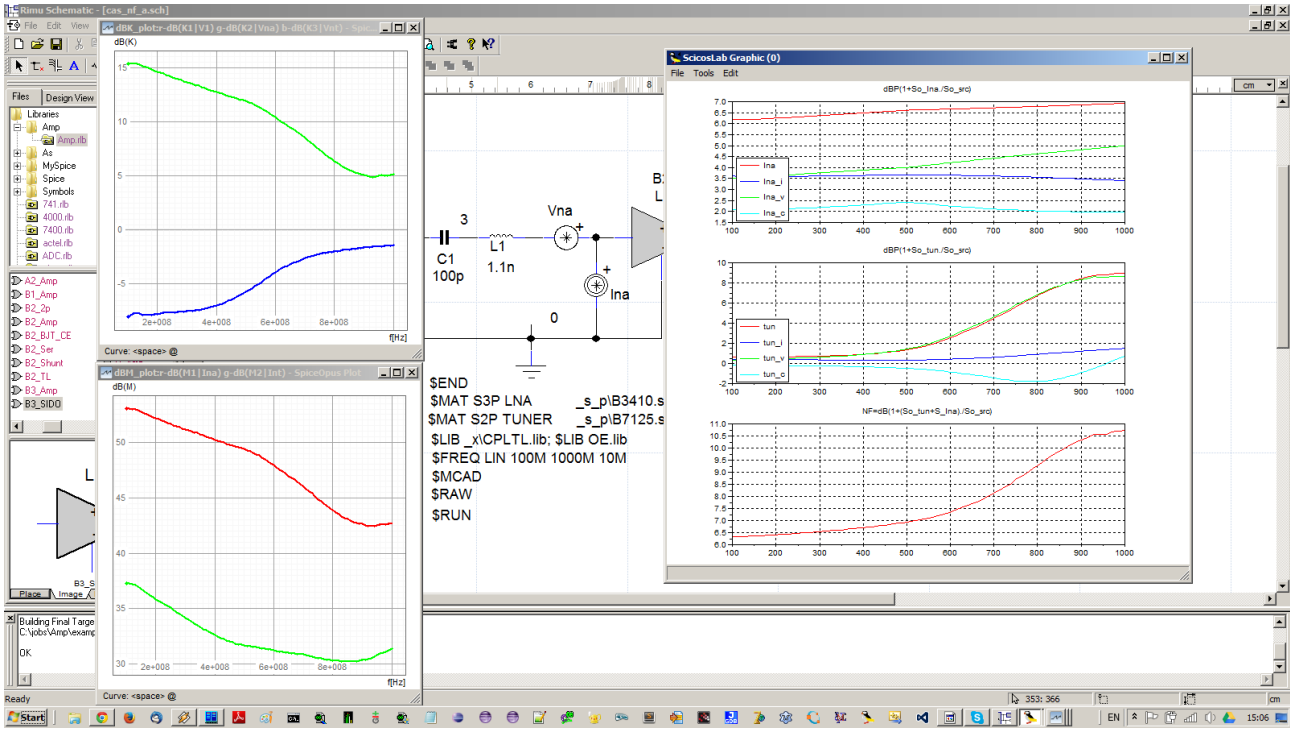


Fig.2.5.2. Results of noise analysis for model in figure 2.5.1.

AMP calculates all required transfer function from noise sources to voltage probe $\mathbf{PU_t}$ at noise reference port. Namely voltage-voltage transfer functions $\mathbf{K1=U_t/V_1}$, $\mathbf{K2=U_t/V_{na}}$, $\mathbf{K3=U_t/V_{nt}}$, and current-voltage transfer functions $\mathbf{M1=U_t/I_{na}}$, $\mathbf{M2=U_t/I_{nt}}$. Values of transfer functions are plotted by **nutmeg** script. **Scilab** script reads **K1,K2,K3** and **M1,M2** and using noise source parameters ($\mathbf{S_{vv}}$, $\mathbf{S_{ii}}$, **Yc**) calculates voltage noise density for voltage probe $\mathbf{PU_t}$.

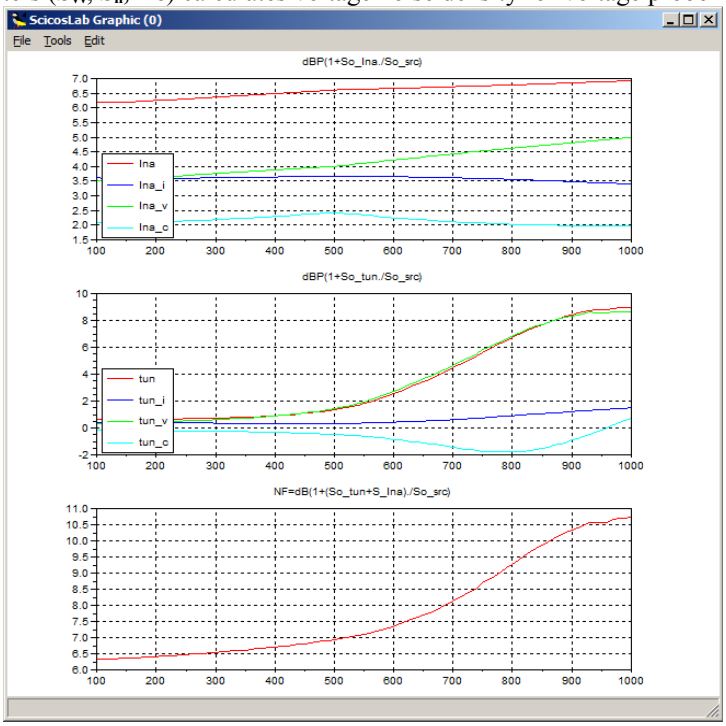


Fig.2.5.3. Noise figure NF analysis.

First graph shows **LNA** contribution, second **tuner** contribution to total **NF** [dB] shown in third graph. Red plot represent accumulated **NF**[dB] value due to current noise(blue), voltage noise (green) and correlation between the them (cyan).

In order to better understand the impact of transmission line on degradation of noise parameters in upper band the model without transmission line was analyzed.

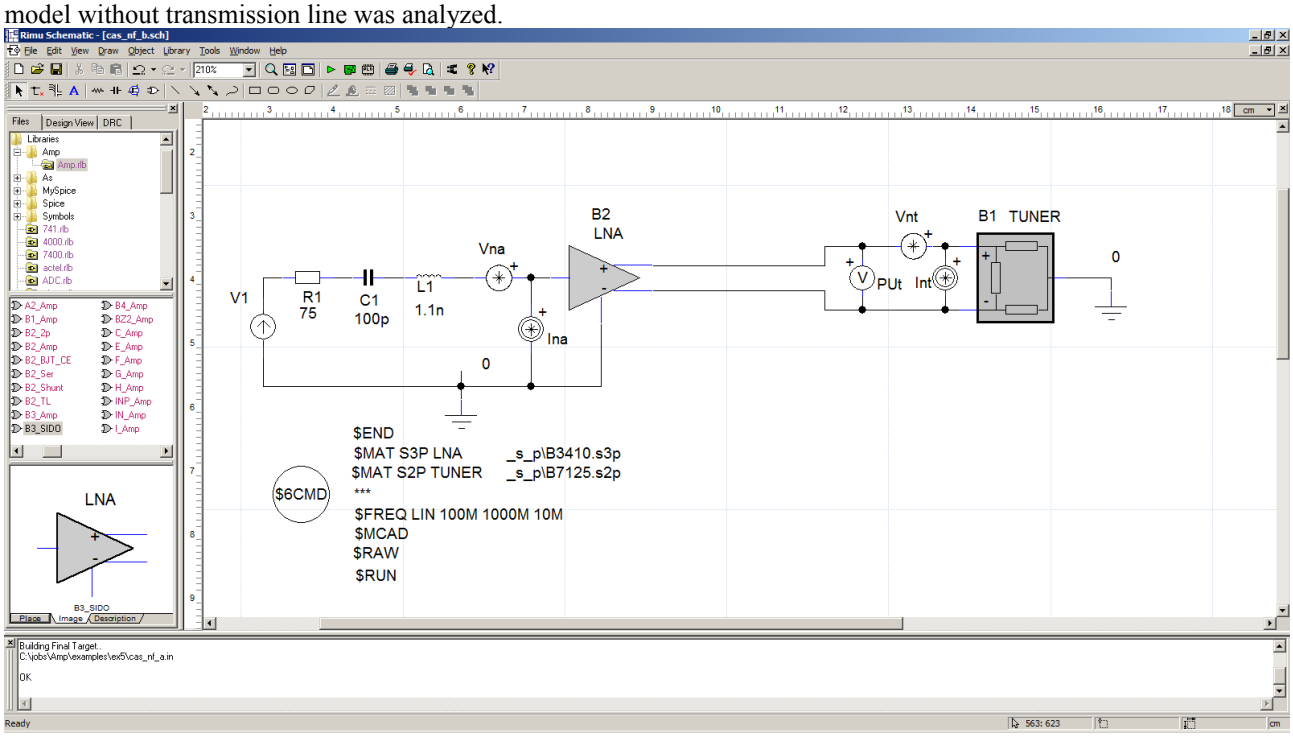


Fig.2.5.4. Noise model of LNA and tuner cascade without transmission line.

Without impedance transformation effect of transmission line, from noise perspective, tuner is much better matched with LNA . Comparing with previous case, the maximum noise figure is below 9dB and the difference due to transmission line is almost 2dB in the upper band.

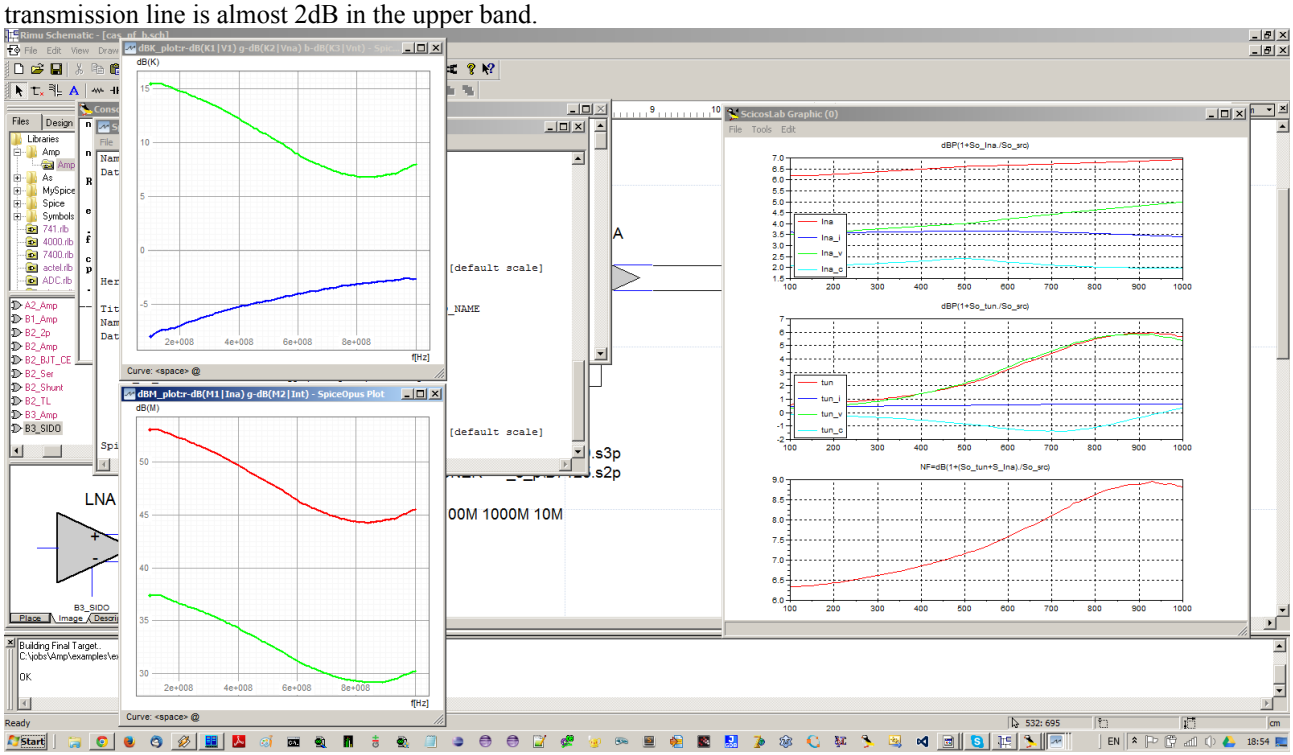


Fig.2.5.5. Results for model of LNA and tuner cascade without transmission line.

Since the effect of transmission line was considered to be significant, the actual s-parameters of the line for differential mode were measured and put into the noise model as **B3**. Measurement technique allowed only for 2 port characterization in differential mode, therefore two ideal baluns **n1**,**n2** provide conversion between single ended s-parameters of the line and differential source and load.

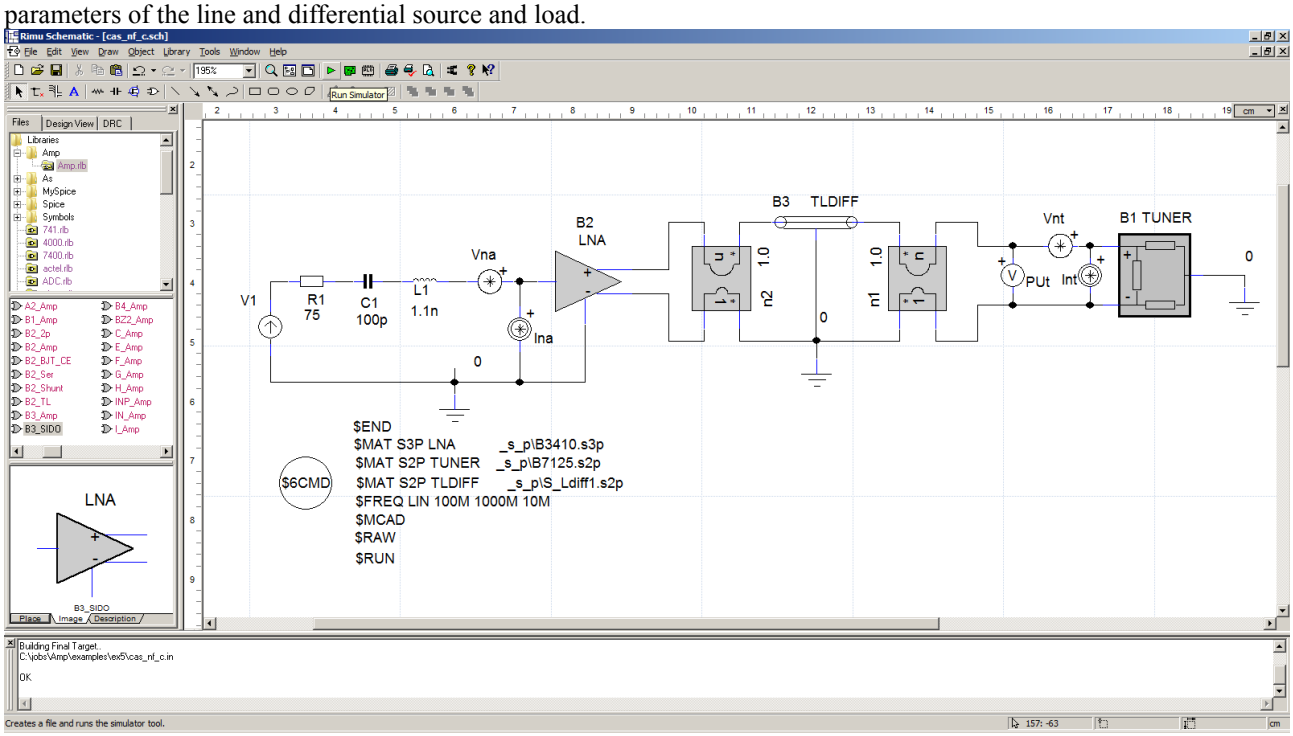


Fig.2.5.6. LNA and tuner cascade with measured differential s-parameters of the transmission line.

With actual parameters of transmission line, the maximum NF is 10dB and degradation due to relatively long connection between tuner and LNA is about 1dB.

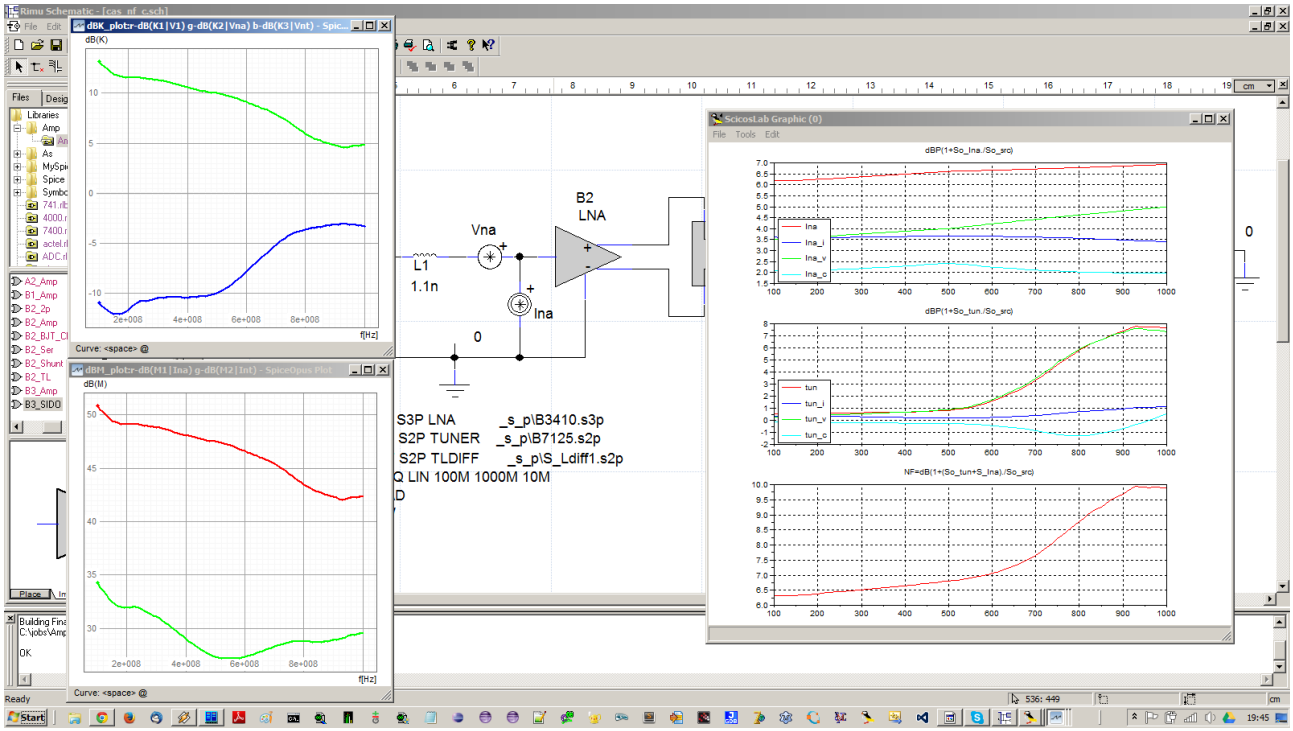


Fig.2.5.7. Results for LNA and tuner cascade with measured differential s-parameters of the transmission line.

2.6. RF passive filter optimization.

Summary: LC filter optimization , pattern search (Hooke-Jeeves algorithm), preferred numbers, Monte Carlo, sensitivities.

Let's assume, that the target for the passive filter transfer function is:

- low pass band 900MHz, with pass band ripple < 1dB,
- stop-band 2400MHz, with attenuation > 80dB
- input return loss < 6dB

The design starts with 7th order 1GHz, 1dB ripples low pass Chebyshev approximation and reference impedance of 75 Ohms. The ideal filter is shown in figure 2.6.1.

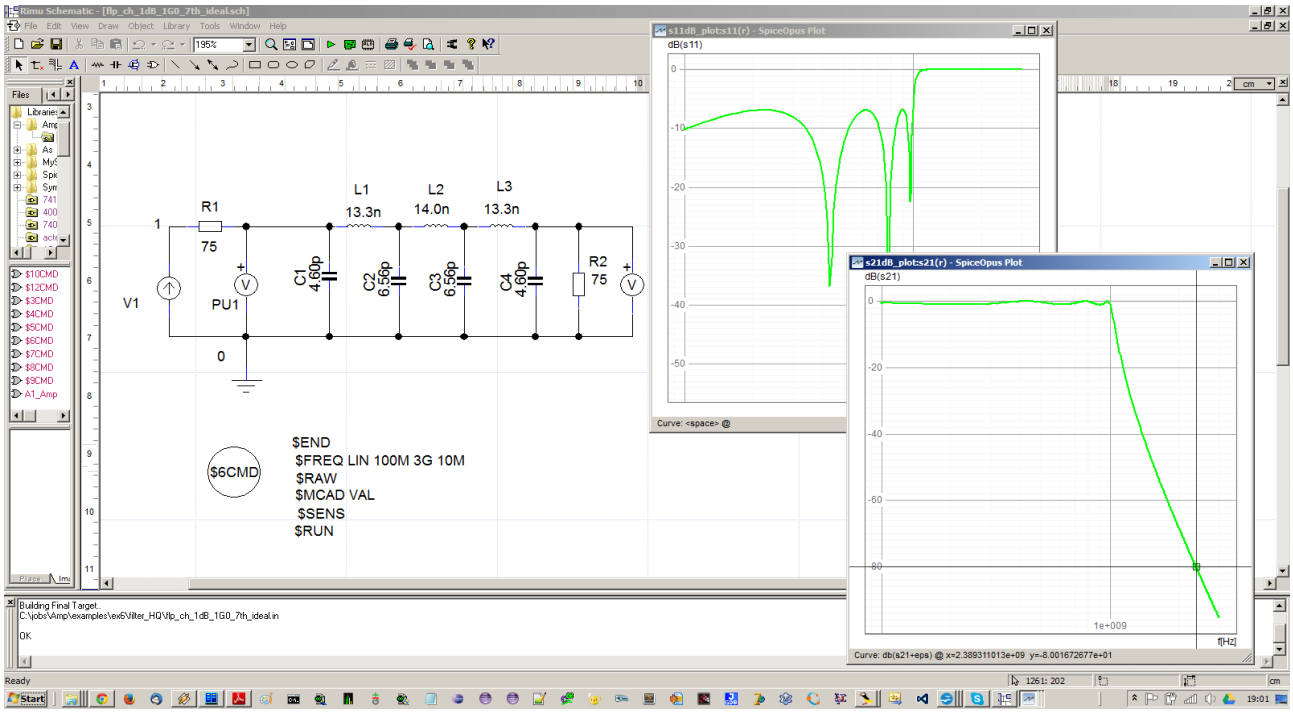


Fig.2.6.1. Ideal filter characteristics.

However, there is a problem with this mathematically elegant solution. The filter build with real discrete parts will hardly perform as expected. The basic reason being, that for this frequency range, discrete coil's and capacitor's characteristics diverge significantly from an ideal parts.

The models of actual discrete part may vary significantly in the level of accuracy and complexity.
The simplest model used by **AMP** is **FQ** model, next in hierarchy are different levels of **X** macro-models, and most complex and most accurate are **s-parameters** models.
FQ model replaces each **FQ MOD** type part with equivalent **RLC** model. Two parameters of **FQ** model are **F**-resonant frequency, and **Q**-equivalent quality factor.
For a given **F,Q** values, **AMP** calculates values of two parasitic for each device. For coil **L**, it is a parasitic capacitance and resistance and for capacitance **C**, equivalent series resistance and inductance. However, **FQ** model does not introduce frequency dependent component values, and as such, can be accurate in a limited frequency band.

The first thing to do with a ideal filter, is to make a feasibility study, with parts modeled as **FQ** models with realistic parameters to approximate frequency response similar of the actual devices used in the design.
This step is illustrated in figures 2.6.2, 2.6.3, for **low-Q**, and **high-Q** coils, respectively.
Base values of parts has not been changed, yet ideal parts are replaced by **FQ** models.

In figures 2.6.2, 2.6.3 green plot represents ideal circuit response, blue plot circuit response of the circuit with FQ models. With **low-Q** coils there is a drop at the end of pass-band - S21 value is **-16dB at 900MHz**.

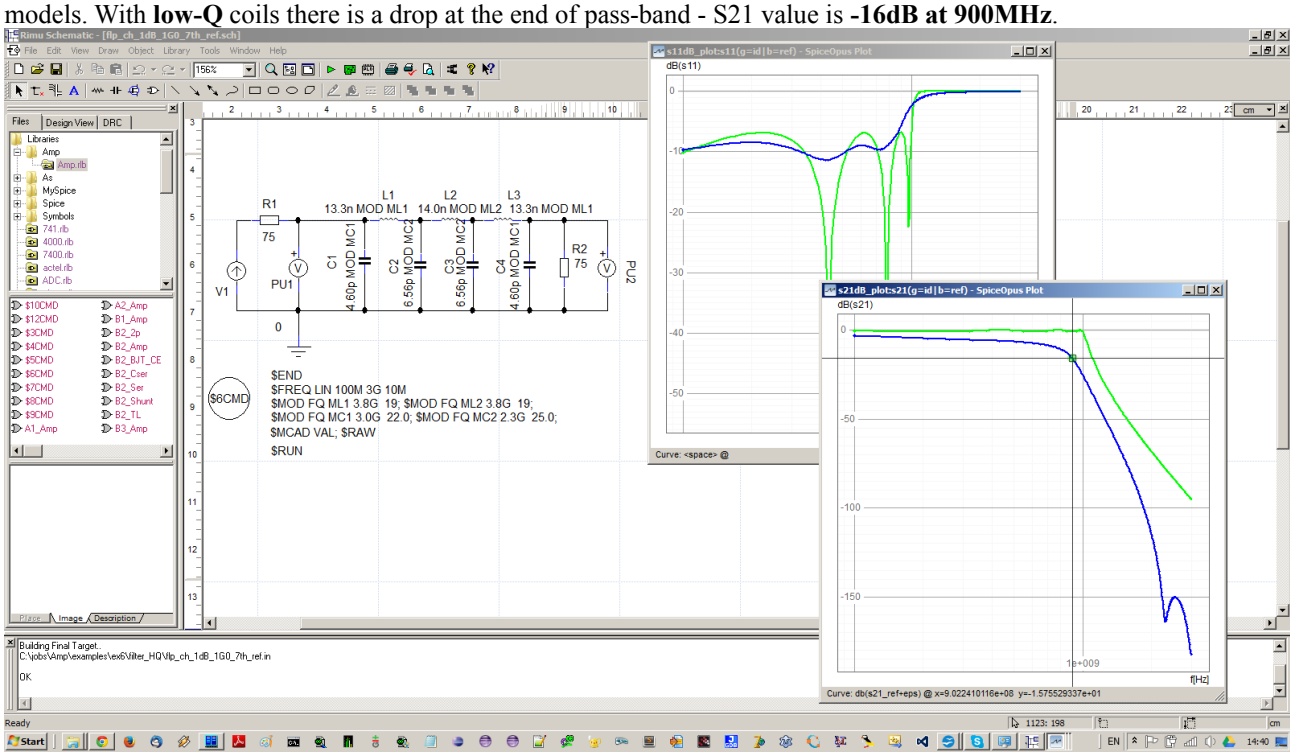


Fig.2.6.2. Filter characteristics for FQ models – low Q coils.

For **high-Q** coils there is also drop at the end of pass-band, S21 value is **-9.5dB at 900MHz**, slightly better but still quite depressing. Clearly, with so significant errors, other approach to filter design is needed.

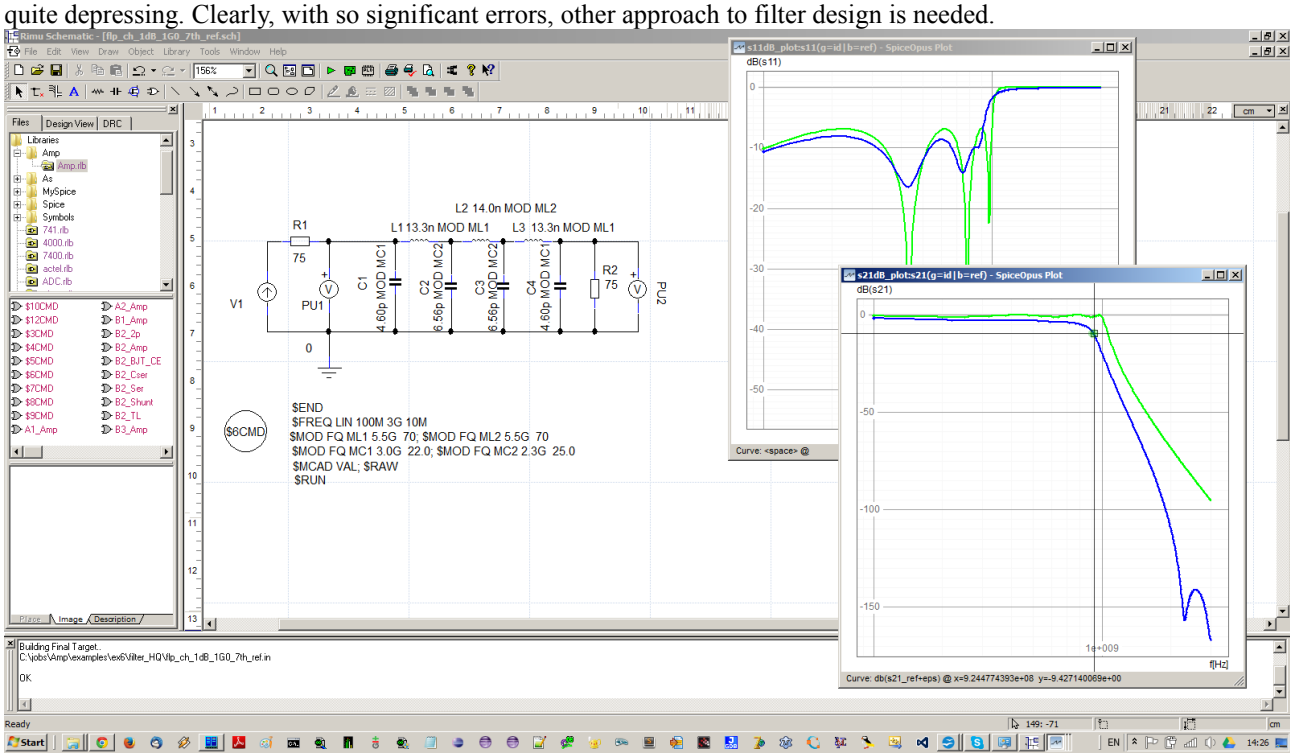


Fig.2.6.3. Filter characteristics for FQ models – high Q coils.

In **AMP** any declared part values can be modified by **$LET** command. With **$LET** command it is relatively easy to implement optimization procedure written as a script for external interpreter - a **scilab** in this case.

In case of optimization runs, schematic editor is used only for preparation of input file, as **AMP** is invoked by running script. In order to prepare the circuit for a optimization runs, the standard description has to be modified.
Required changes are removal of model declarations, addition of **$INC** command and usage **$MCAD VAL** command.

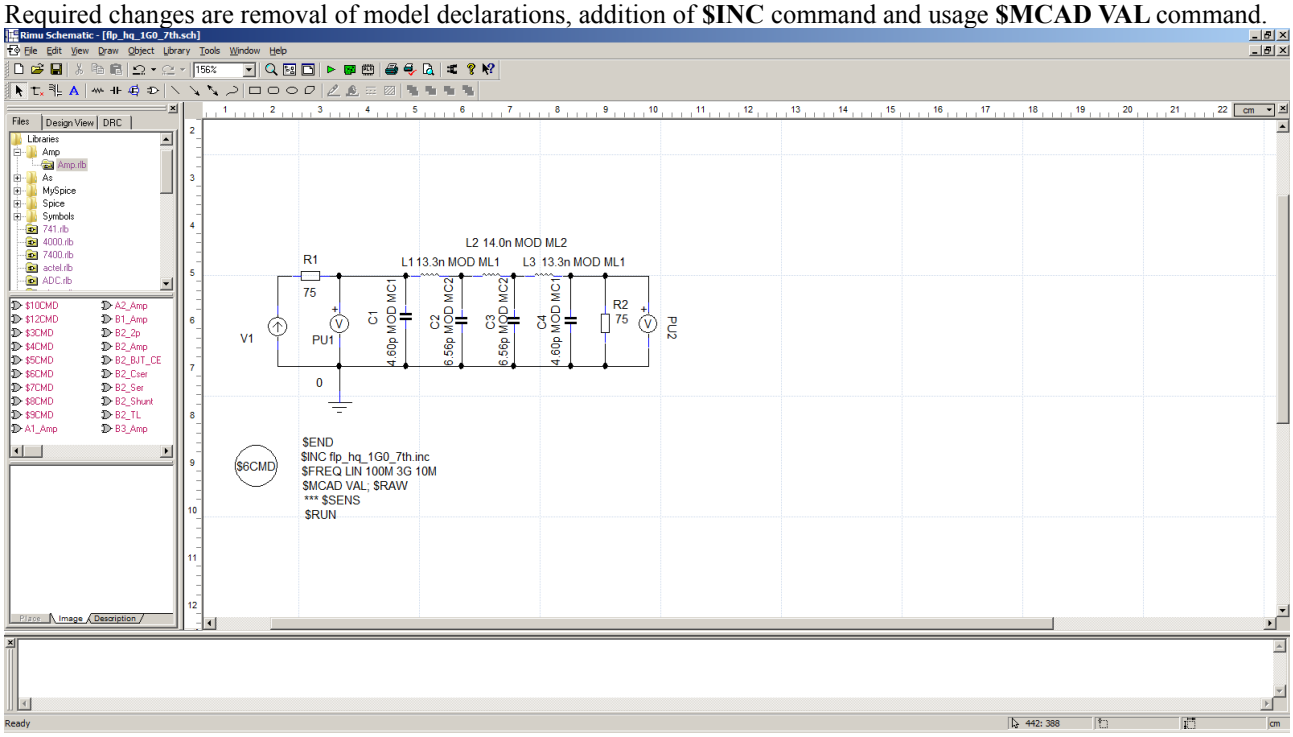


Fig.2.6.4. Circuit description for optimization runs.

In principle, **scilab script** does not read filter circuit description. Script invokes **AMP**, reads files with results of analysis and generates new version of a file included by **$INC** command for a next run.
To get started, scripts invokes **AMP** with ideal filter circuit. Part values read from ideal filter are used to create **FQ** models for initial optimization run. Other scenarios are possible for initialization of optimization, yet the one with ideal filter is actually used in the examples. **$LET** commands changes declared part values. **FQ** models are updated according to models build into the **script**.

```
$MOD FQ MC1 3.0375E+009 20.8242
$LET C1 4.6E-012
$MOD FQ MC2 2.34E+009 22.1586
$LET C2 6.56E-012
$MOD FQ MC3 2.34E+009 22.1586
$LET C3 6.56E-012
$MOD FQ MC4 3.0375E+009 20.8242
$LET C4 4.6E-012
$MOD FQ ML1 5.78333E+009 72.129
$LET L1 1.33E-008
$MOD FQ ML2 5.66667E+009 71.3991
$LET L2 1.4E-008
$MOD FQ ML3 5.78333E+009 72.129
$LET L3 1.33E-008
```

The format of data generated by **script** is exactly the same for each run. File generated in the final run can be used as a result of optimization without the need of modifying circuit description.

```
$MOD FQ MC1 4.15629E+009 27.016
$LET C1 2.74371E-012
$MOD FQ MC2 3.13218E+009 20.6253
$LET C2 4.34752E-012
$MOD FQ MC3 3.13218E+009 20.6253
$LET C3 4.34752E-012
$MOD FQ MC4 4.15629E+009 27.016
$LET C4 2.74371E-012
$MOD FQ ML1 7.13664E+009 106.828
$LET L1 9.75405E-009
$MOD FQ ML2 7.91643E+009 112.526
$LET L2 8.35042E-009
$MOD FQ ML3 7.13664E+009 106.828
$LET L3 9.75405E-009
```

Example optimization session is shown in figure 2.6.5. The session is started and controlled by **scilab**. While script is running, **AMP** is invoked multiple times, and in order to to save time, it is better to configure **AMP** without graphical interface (**GUI = 0)** and with display delays set to minimum (**TICK = 1)**.

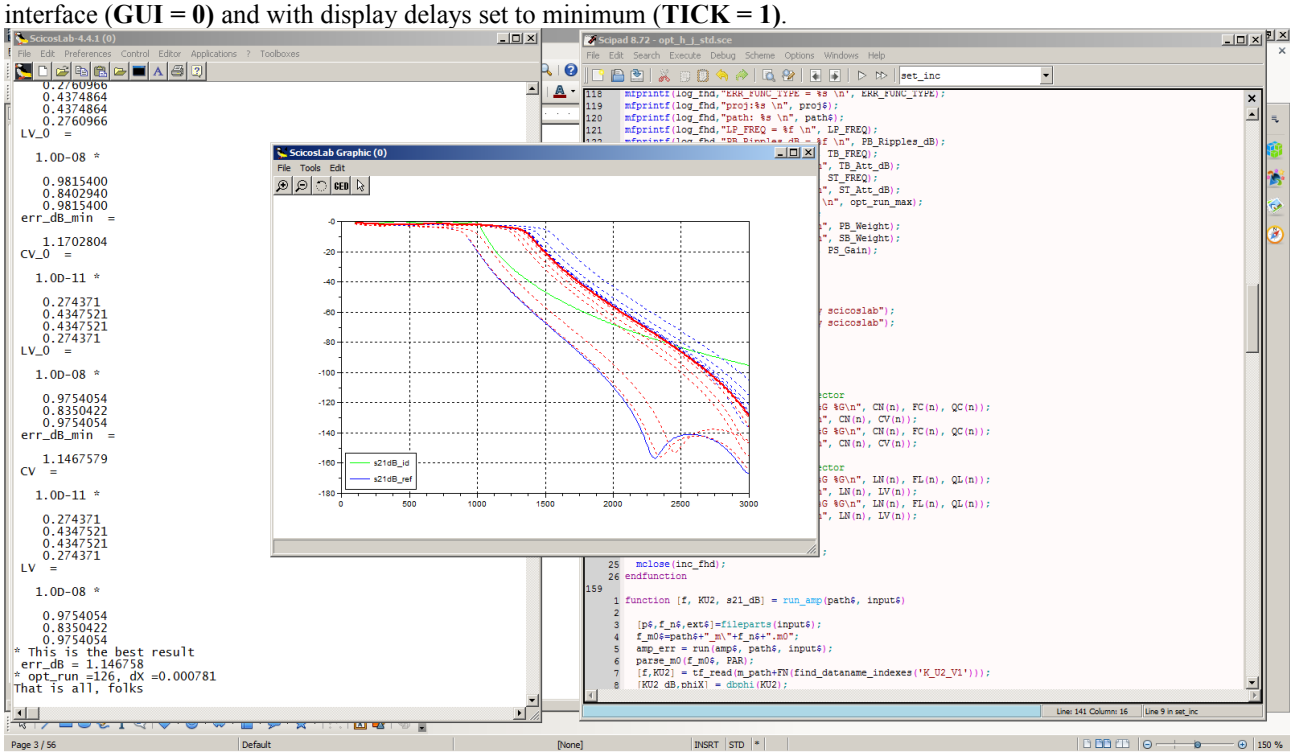


Fig.2.6.5. Example pattern search optimization session in **scilab**.

Optimization runs can have many flavors and as an inherently nonlinear processes may give different results depending on problem formulation or initial conditions. In principle optimization runs attempt to minimize selected error function **[err(X)]** by searching of optimal values **[min = err (X_{opt})]** for all passive components **X=(C1,C2,C3,C4, L1,L2,L3)**. Due to the nature of problem, parameters of corresponding **FQ** model **have to be updated along with the new value of component** during search procedure**.**

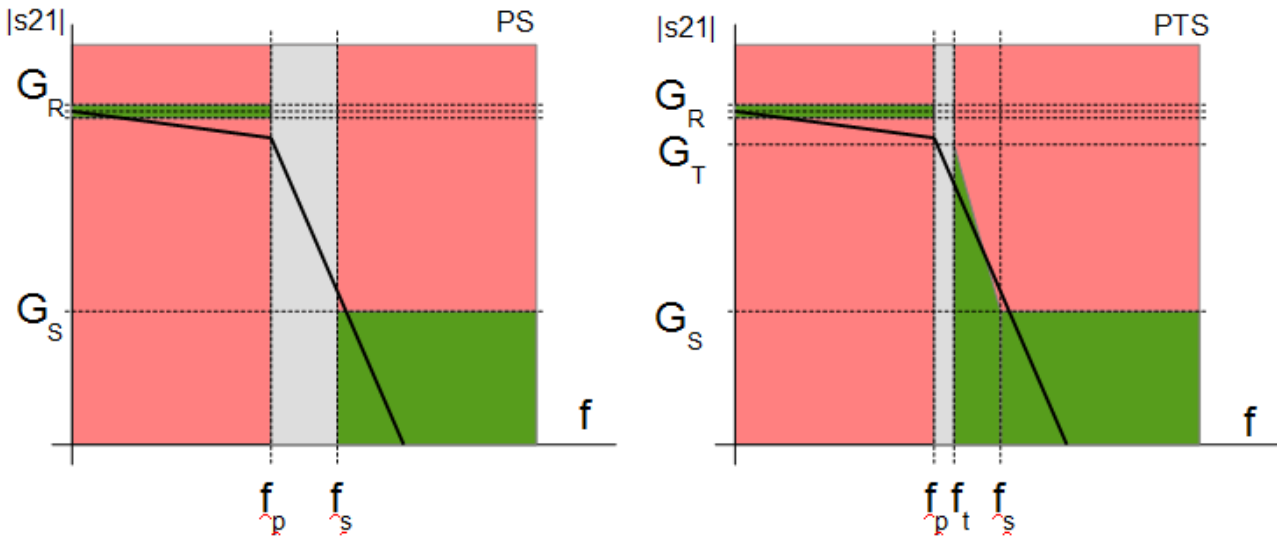


Fig.2.6.6. Types of error functions used in filter optimization.

Search procedure in optimization algorithm may generate virtually any value for a component (**X= (C1,C2,C3, C4, L1,L2,L3)** space values is continouse), therefore parameters of **FQ** models have to estimated by interpolation. For a given value of the component, two discrete **FQ** models are found in a component database, for the closest smaler and

larger values. **FQ** parameters are calculated by linear interpolation. The definition of a used **FQ** database is a configuration parameter for optimization script.
Two types of error function shown in figure 2.6.6 were used. **PS** function (left) calculates error only for pass(**P**) and stop(**S**) band. **PTS** function (right) calculates also error in transition (**T**) band. Green areas are considered to be target function values. If for a given frequency, function value is in the green area, error function is zero. If function value is in the red area, error is calculated as a distance to green area. Errors are calculated and expressed as decibels [**dB**]. Gray color marks don't care area, where errors are not calculated. The pass band is defined by ripples $\mathbf{G_R}$**[dB]** and frequency $\mathbf{f_p}$. Stop band by attenuation $\mathbf{G_S}$**[dB]** and frequency $\mathbf{f_s}$. Transition band by attenuation $\mathbf{G_T}$**[dB]** at frequency $\mathbf{f_T}$.
For a given discrete frequency set defined by **$FREQ** command, error function can be expressed as sum of errors (**SUM**) or maximum value of errors (**MAX**). These, together with two types of error functions (**PT**, **PTS)** give possible error functions: **PS-SUM**, **PS-MAX**, **PTS-SUM**, **PTS-MAX**, used in optimization procedure.
The optimization algorithm is based on **'pattern search' (a.k.a. Hooke-Jeeves)** approach.
The first modification is that search step represents relative change. Starting step value of **dX=0.20** means, that initial search space is determined by **20%** relative component varation of all components in **X** vector.
Second modification is, that along with standard exploration **STD**, alternate aproach using sensitivities may be used in a exploration steps. **SENS** modification minimizes number of **AMP** calls but makes **AMP** calculations more elaborate and less accurate since sensitivies analysis takes into account only the change of base value, while **FQ** model parameters remain unchanged. But, since sensitivities are used only for exploratory steps, only the quality of pattern search estimation might be affected. The pattern search steps, are performed with simultanous change of componet values and **FQ** model updates, regardless of the method used in exploration.
Basic optimizer configuration is determined by:

error function : **PS-SUM**, **PS-MAX**, **PTS-SUM**, **PTS-MAX**
exploration method : **STD**, **SENS**
FQ models database: **HQ** – high Q coils, **LQ**- low Q coils.
Steps configuration: **dX_max**, **dX_min**, **N_STEP** maximum number of search steps (exploratory steps are not counted)

The optimizer stops when either:

- solution with zero error is found **err** $(\mathbf{X_{opt}}) = \mathbf{0}$,
- or relative step is lower then minimum value **dX<dX_min**,
- or maximum number of steps is exceeded **N > N_STEP**.

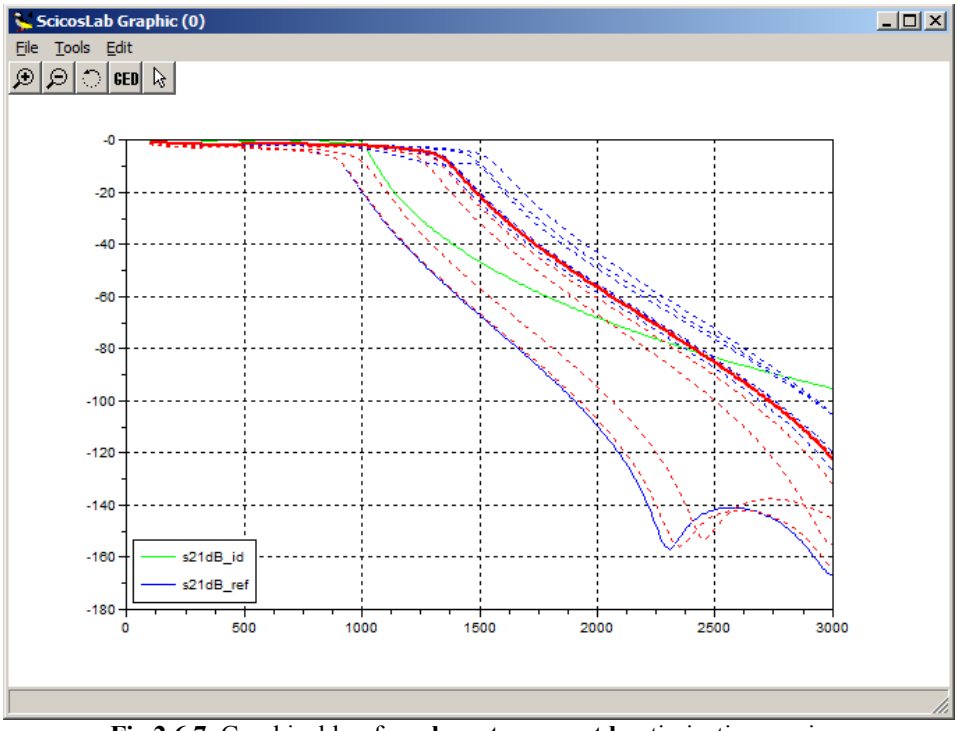


Fig.2.6.7. Graphical log from **hq_pts_sum_std** optimization session.

The script creates graphical and textual log.
Graphical log is shown in figure 2.6.7. The solid green plot is for the ideal filter, the solid blue plot for a reference filter, each red dashed marks a succesful, each dashed blue marks an unsuccesfull searching attempt, thick red plot marks the solution.
Begininnig and ending piece of a textual log is shown below.

```
log:_log_pts_sum\flp_ch_1dB_1G0_7th_1.log
ERR_FUNC_TYPE = PTS_SUM
FQ_MOD = HIGH_Q
ref:flp_ch_1dB_1G0_7th_ref.in
proj:flp_ch_1dB_1G0_7th.in
path: \jobs\Amp\ex\ex6\flp_hq_pts_sum_std\
LP_FREQ = 1000000000.000000
PB_Ripples_dB = 1.000000
TB_FREQ = 1500000000.000000
TB_Att_dB = -40.000000
ST_FREQ = 2400000000.000000
ST_Att_dB = -80.000000
opt_run_max = 150
dX = 0.100000
PB_Weight = 1.000000
SB_Weight = 1.000000
PS_Gain = 1.000000
* ideal run
err_dB = 18.010174 pass_err_dB =0.056018 tran_err_dB=17.954156 stop_err_dB=0.000000
* reference run
err_dB = 325.242172 pass_err_dB =325.242172 tran_err_dB=0.000000 stop_err_dB=0.000000
* initial run
* set by scicoslab
$MOD FQ MC1 3.0375E+009 20.8242
$LET C1 4.6E-012
$MOD FQ MC2 2.34E+009 22.1586
$LET C2 6.56E-012
$MOD FQ MC3 2.34E+009 22.1586
$LET C3 6.56E-012
$MOD FQ MC4 3.0375E+009 20.8242
$LET C4 4.6E-012
$MOD FQ ML1 5.78333E+009 72.129
$LET L1 1.33E-008
$MOD FQ ML2 5.66667E+009 71.3991
$LET L2 1.4E-008
$MOD FQ ML3 5.78333E+009 72.129
$LET L3 1.33E-008
******
err_dB = 325.938888 pass_err_dB =325.938888 tran_err_dB=0.000000 stop_err_dB=0.000000
...
...
...
$MOD FQ MC1 4.16914E+009 27.1639
$LET C1 2.73086E-012
$MOD FQ MC2 3.23163E+009 20.4163
$LET C2 4.08233E-012
$MOD FQ MC3 3.23163E+009 20.4163
$LET C3 4.08233E-012
$MOD FQ MC4 4.16914E+009 27.1639
$LET C4 2.73086E-012
$MOD FQ ML1 6.94204E+009 103.955
$LET L1 1.01159E-008
$MOD FQ ML2 7.26759E+009 107.785
$LET L2 9.51833E-009
$MOD FQ ML3 6.94204E+009 103.955
$LET L3 1.01159E-008
******
err_dB = 69.846449 pass_err_dB =61.162841 tran_err_dB=8.133144 stop_err_dB=0.550463
* This is the best result
 err_dB = 69.846449
* opt_run =129, dX =0.000781
That is all, folks
```

The results obtained by different optimizer configuration are in a table 2.6.1 and in a figure 2.6.8.

Only two runs gave exactly equal results (**hq_pts_max_std**, **hq_pts_max_sens**), the reason being, that there were only four successful search steps in both cases.
The two best designs (**hq_pt_sum_std**, **hq_pts_sum_std**) are also practically identical, which is quite remarkable as both were found in more than 100 steps.

	hq_ps_max_std	hq_ps_max_sens	hq_ps_sum_std	hq_ps_sum_sens
C1	2.7437E-012	2.6756E-012	2.7139E-012	2.7265E-012
C2	4.3475E-012	4.2395E-012	4.0826E-012	3.8882E-012
C3	4.3475E-012	4.2395E-012	4.0826E-012	3.8882E-012
C4	2.7437E-012	2.6800E-012	2.7139E-012	2.7265E-012
L1	9.7541E-009	1.0320E-008	1.0148E-008	1.1479E-008
L2	8.3504E-009	9.0001E-009	9.5189E-009	9.8858E-009
L3	9.7541E-009	1.0320E-008	1.0148E-008	1.1479E-008
	hq_pts_max_std	hq_pts_max_sens	hq_pts_sum_std	hq_ps_sum_sens
C1	3.3120E-012	3.3120E-012	2.7309E-012	2.7094E-012
C2	4.7232E-012	4.7232E-012	4.0823E-012	3.8639E-012
C3	4.7232E-012	4.7232E-012	4.0823E-012	3.8639E-012
C4	3.3120E-012	3.3120E-012	2.7309E-012	2.7094E-012
L1	9.5760E-009	9.5760E-009	1.0116E-008	1.1550E-008
L2	1.0080E-008	1.0080E-008	9.5183E-009	9.9476E-009
L3	9.5760E-009	9.5760E-009	1.0116E-008	1.1550E-008

Tab.2.6.1. Results of optimization session.

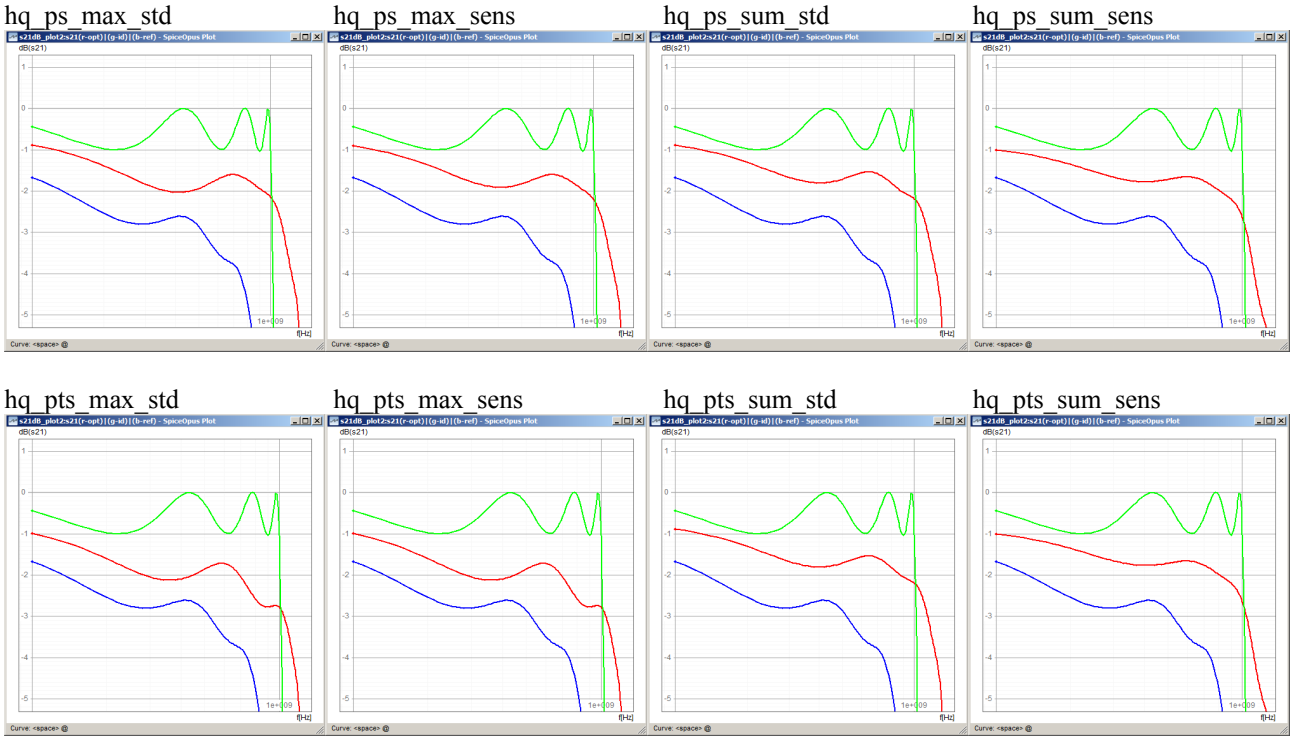


Fig.2.6.8. Comparison of results of different optimization session from table 2.6.1.

The major problem with the design is lack of flatness in upper pass band. The upper pass band is zoomed in figure 2.6.8, where green plot is ideal filter response, blue is reference filter response (the design shown in figure 2.6.3), red plot is the result of optimization.
From pass band flatness perspective, the best design is **hq_pts_sum_std**, and this design is qualified to preferred number optimization.

The preferred number optimization tries to minimize error function while replacing arbitrary part values with values from a standard set. The standard set can be either predefined **E6**, or **E12** or **E24** or any other.
The entry point in a process is the design selected as the best result of pattern search optimization stage - **hq_pts_sum_std** in this case. For each part value, two discrete numbers from standard set are found, the closest smaller or equal and closest larger number. Thus during preferred optimization, part value can be replaced by either lower or higher discrete number. The first combination checked is that of closest values. All other combinations are checked and the one, which provides smallest error, is the result of preferred number optimization and marked as a thick red line.

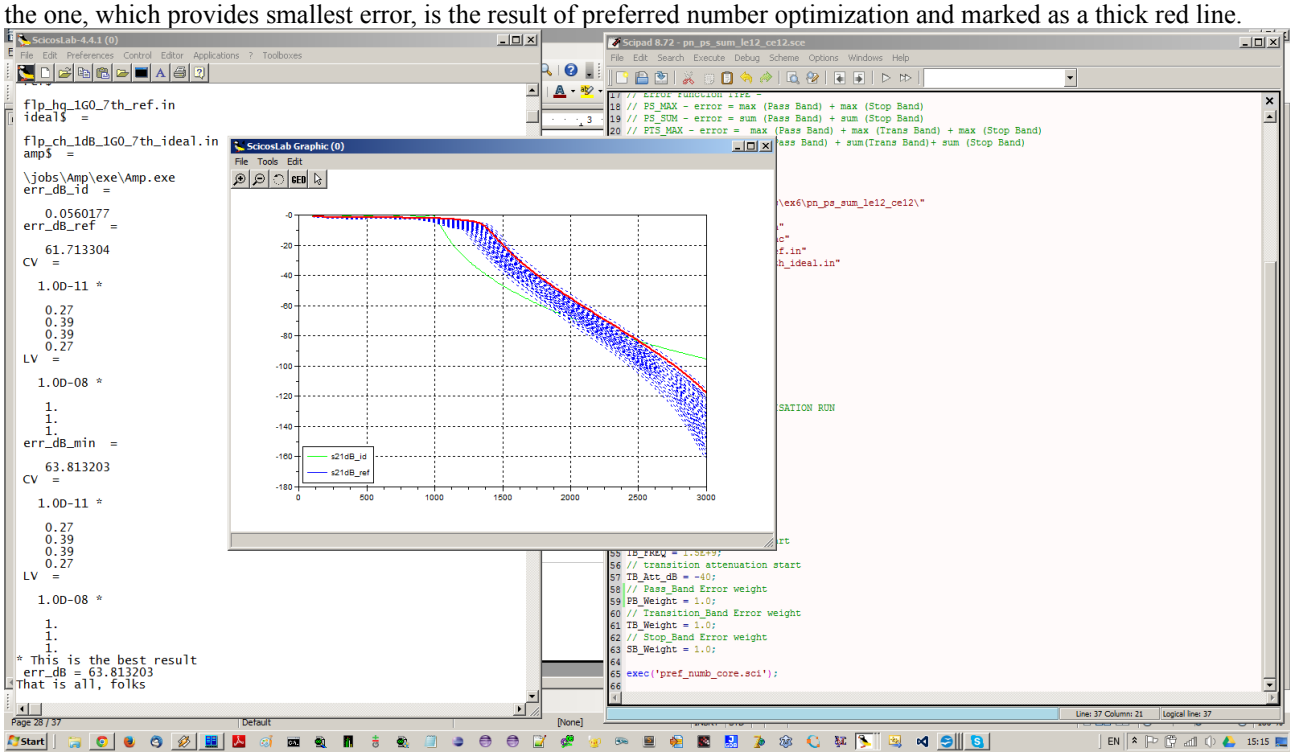


Fig.2.6.9. Example preferred number optimization session in **scilab**.

	hq_ps_max	hq_ps_sum	hq_pts_max	hq_pts_sum	closest
C1	2.70E-012	2.70E-012	3.30E-012	2.70E-012	2.70E-012
C2	4.70E-012	3.90E-012	4.70E-012	3.90E-012	3.90E-012
C3	4.70E-012	3.90E-012	4.70E-012	3.90E-012	3.90E-012
C4	3.30E-012	2.70E-012	3.30E-012	2.70E-012	2.70E-012
L1	1.00E-008	1.00E-008	1.00E-008	1.00E-008	1.00E-008
L2	1.00E-008	1.00E-008	1.00E-008	1.00E-008	1.00E-008
L3	1.00E-008	1.00E-008	1.00E-008	1.00E-008	1.00E-008

Tab.2.6.2. Results of preferred number optimization sessions.

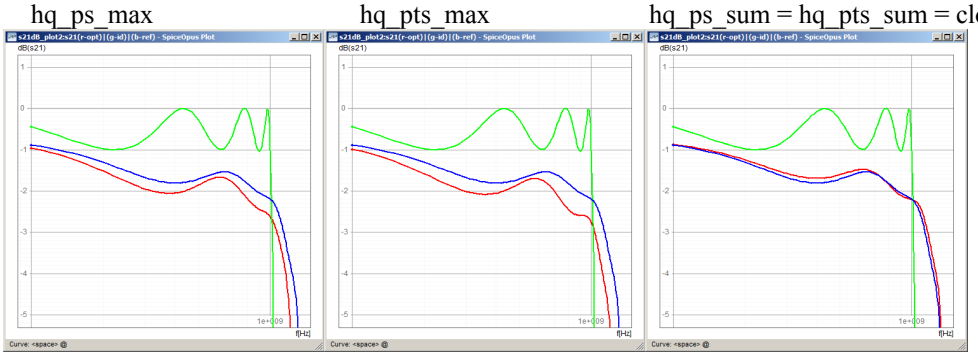


Fig.2.6.10. Comparison of results of preferred number optimization sessions from table 2.6.2.

The preferred number optimization should provide smallest possible deviation from the pattern-search result, and the solution obtained for 2 error functions (**hq_ps_sum**, **hq_pts_sum**) meets this criteria. As it happens, it is also **closest values** solution. The results are shown in table 2.6.2 and in figure 2.6.10, where green plot is the ideal filter response, blue plot is a result of pattern-search optimization, red plot is the result of preferred number optimization.

The next step in filter design is sensitivity analysis. There are two approaches supported by **AMP**. First option is Monte-Carlo analysis implemented as a **scilab** script. Second option is usage of **$SENS** command and post processing of results.

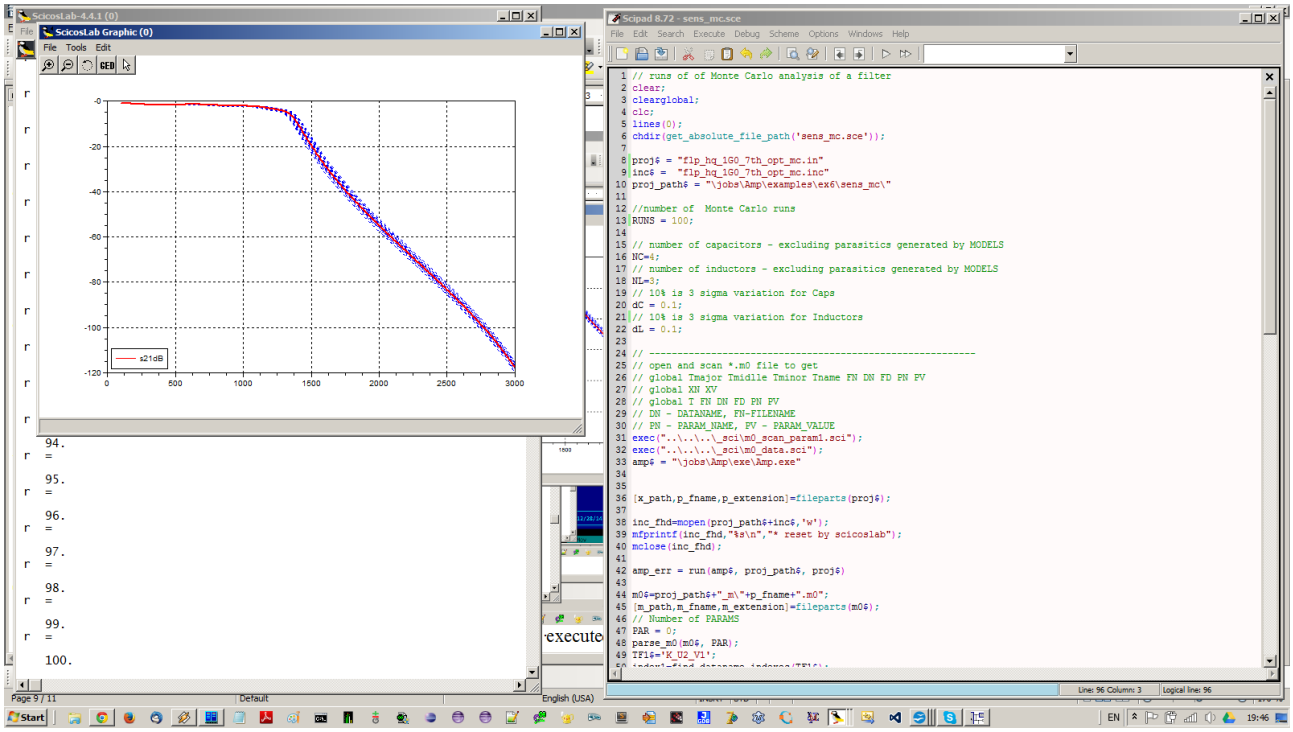


Fig.2.6.11. Monte-Carlo analysis executed by **scilab**.

In Monte-Carlo analysis **AMP** is invoked multiple times, values of parts are generated by random number generators and by means of generation include files with **$LET** commands create a circuit description with random spread of parameters. Therefore, circuit description should include nominal part values and parameters of **FQ** models, as shown below.

```
C1   0 1 2.7p MOD MC1
C2   0 2 3.9p MOD MC2
C3   0 3 3.9p MOD MC2
C4   0 4 2.7p MOD MC1
L1   1 2 10n MOD ML1
L2   2 3 10n MOD ML1
L3   3 4 10n MOD ML1
PU1   1 0
PU2   4 0
R1   5 1 75
R2   0 4 75
V1   5 0
* Commands
$END
$INC flp_hq_1G0_7th_opt_mc.inc
$FREQ LIN 100M 3G 10M
$MOD FQ ML1 7.0G  106
$MOD FQ MC1 4.2G  27.0; $MOD FQ MC2 3.3G  22.0
$MCAD VAL
$RUN
```

Scilab script is configured with number of required runs and value of 3σ spread for capacitor and inductors. The presented results in figures 2.6.12, 2.6.13 are for 100 runs and 10% 3σ spread for both capacitors and coils. The first run is a nominal values run, with empty included file, and response for this run is marked as thick red line, all others with randomly generated part values are represented as blue doted plots. Random number generators are not correlated.

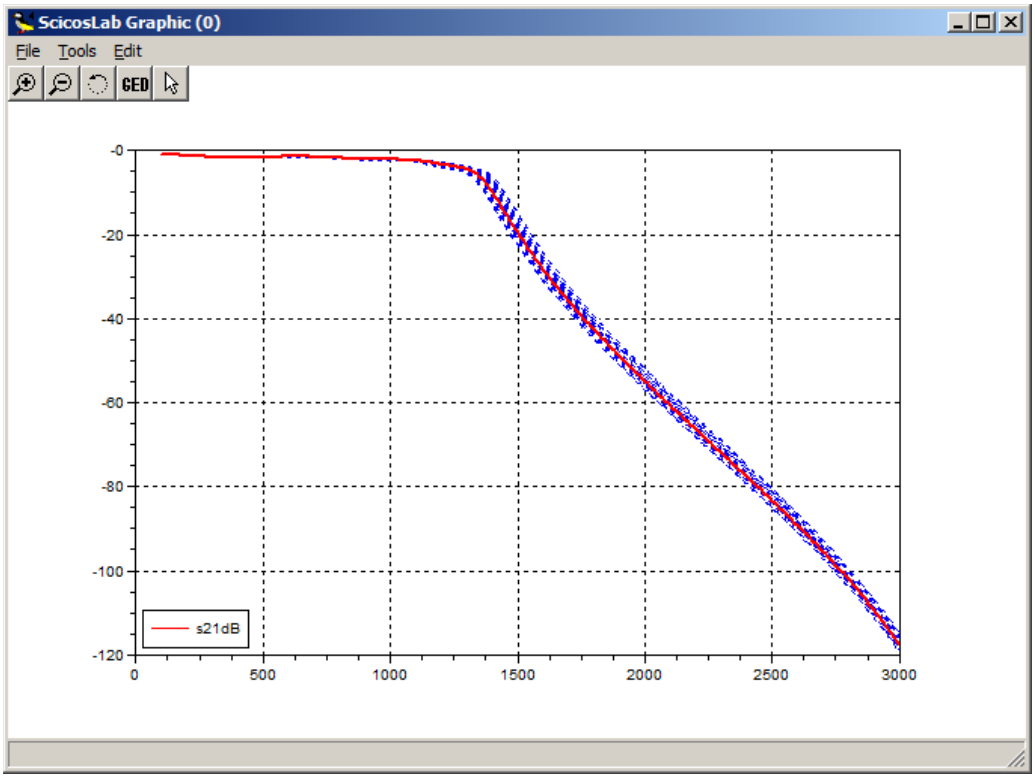


Fig.2.6.12. Monte Carlo results for C_3σ =10%, L_3σ=10%, 100 runs.

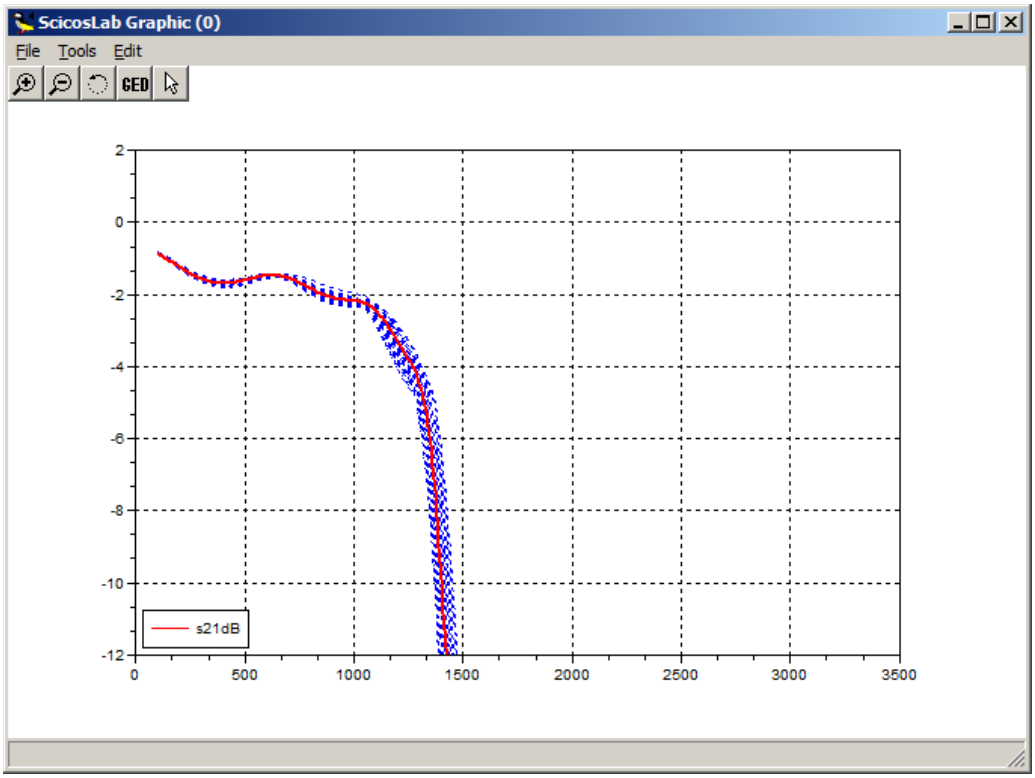


Fig.2.6.13. Zoomed pass band of the graph in figure 2.6.12.

AMP calculates sensitivities of transfer functions when **$SENS** command is used in circuit. This type of analysis may be launched directly from schematics editor, yet **scilab** script has to be used as a post processing tool.

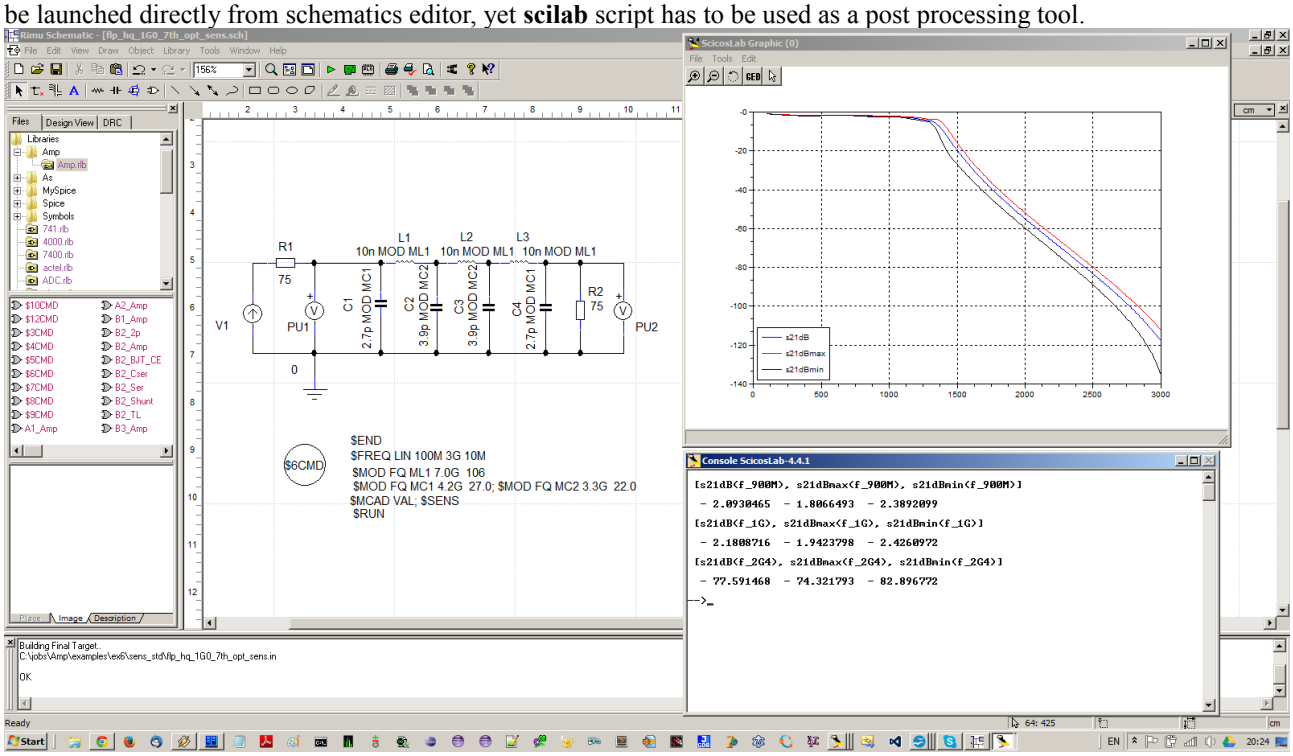


Fig.2.6.14. Sensitivity analysis of the filter.

Similarly, like in Monte-Carlo analysis part values are considered to be random variables, with nominal values corresponding with mean value. Spread of variable is assumed to be Gaussian and determined by 3σ. With sensitivities one can calculate the deviation of transfer function δTF due to the change of any parameter δX. Thus, assuming that spread of part values are known to be random variables with know spread σ_x, it is relatively easy to the calculate estimation the spread of transfer functions $\sigma_{TF}=S(\sigma_x)$ as long the part variations can be considered to be small $\sigma_x<<1$. For uncorrelated spread of all L,C parts defined by $3\sigma_x = 10\%$, $3\sigma_{TF}$ - deviation of transfer function is calculated.

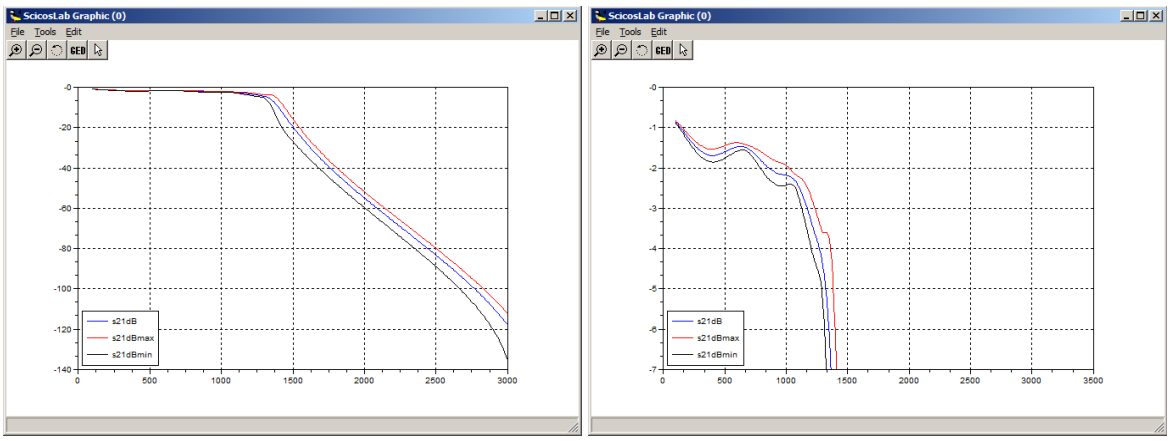


Fig.2.6.15. Results of sensitivity analysis of the filter.

S21 is nominal value of the magnitude of a transfer function, S21max = $S21(1+3\sigma_{TF})$ corresponds to nominal value with $3\sigma_{TF}$ deviation added, S21dB = $S21(1-3\sigma_{TF})$ corresponds to nominal value with $3\sigma_{TF}$ deviation subtracted.

For 900MHz: S21dB= - 2.09, S21dBmax = -1.81, S21dBmin = -2.39
For 1000MHz: S21dB= - 2.18, S21dBmax = -1.94, S21dBmin = -2.42
For 2400MHz: S21dB= - 77.6, S21dBmax = -74.3, S21dBmin = -82.9

With all their flexibility **FN** models are not very accurate. Therefore, in the last stage of analysis **FN** models are replaced by s-parameters models of **L,C** parts.

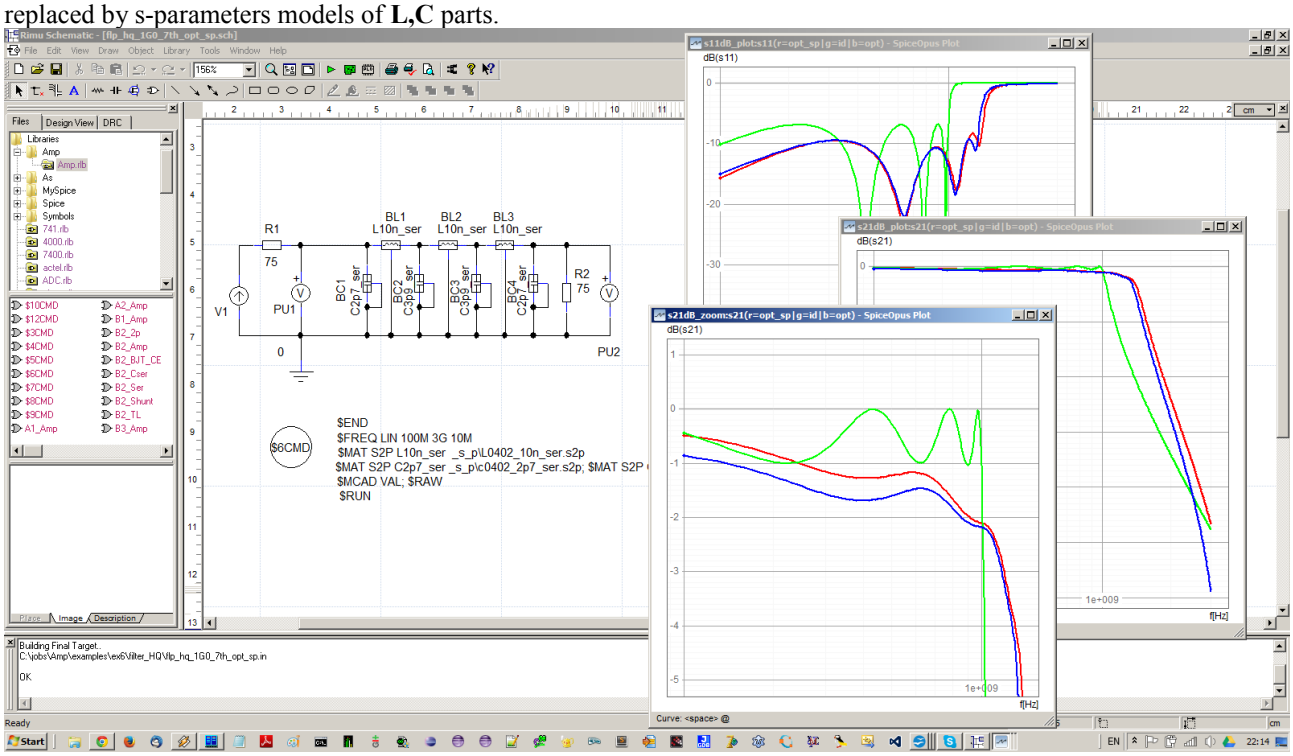


Fig.2.6.16. Comparison of results of analysis for **FN** (blue) models and s-parameters (red) 0402 models.

S-parameters models give less attenuation in pass-band, but less in stop-band.

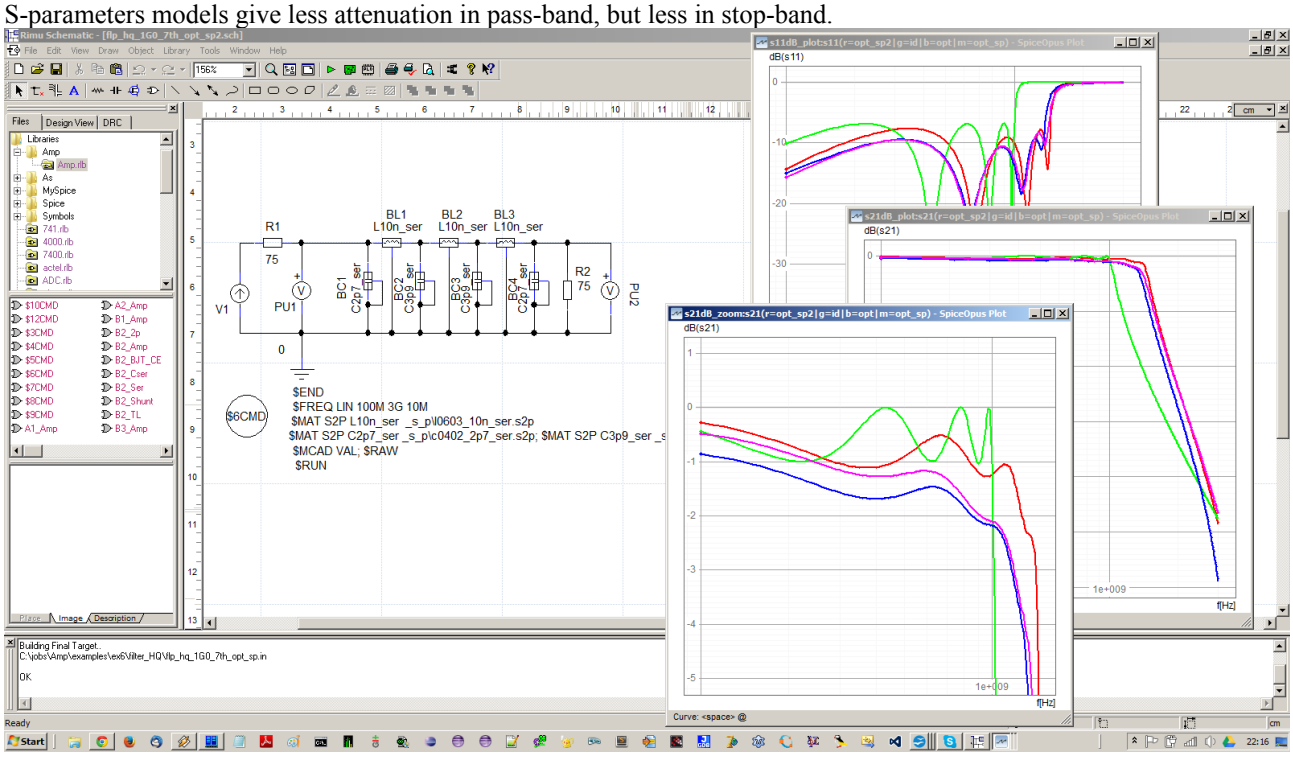


Fig.2.6.17. Comparison of results of analysis with s-parameters for 0603 coils(red) vs 0402 (magenta).

If 0402 coils are replaced with 0603 coils the flattness in pass-band is even better, but stop band attenuation is still about 10dB worse than predicted by **FN** models. If this is a problem, next design iteration is required, which means return with the design to pattern-search optimization stage.

3. Examples explained.

Example1 - Lna

Summary: ideal directional coupler

Example2 - Active splitter

Summary: voltage gains and s-parameters test network.

Example3 - Stability analysis.

Summary: stability factors, stability circles, frequency models.

Example4 - Coupled transmission lines

Summary: even-odd modes, coupled line modeling, decoupling transformations.

Example5 - NF for cascade of LNA and tuner.

Summary: noise modeling, NF, minimal NF, Touchstone noise data, noise analysis.

Example6 - RF passive filter optimization.

Summary: folder structure, sensitivities.

3.1. LNA example explained.

Summary: ideal directional coupler.

3.1.1. Ideal directional coupler.

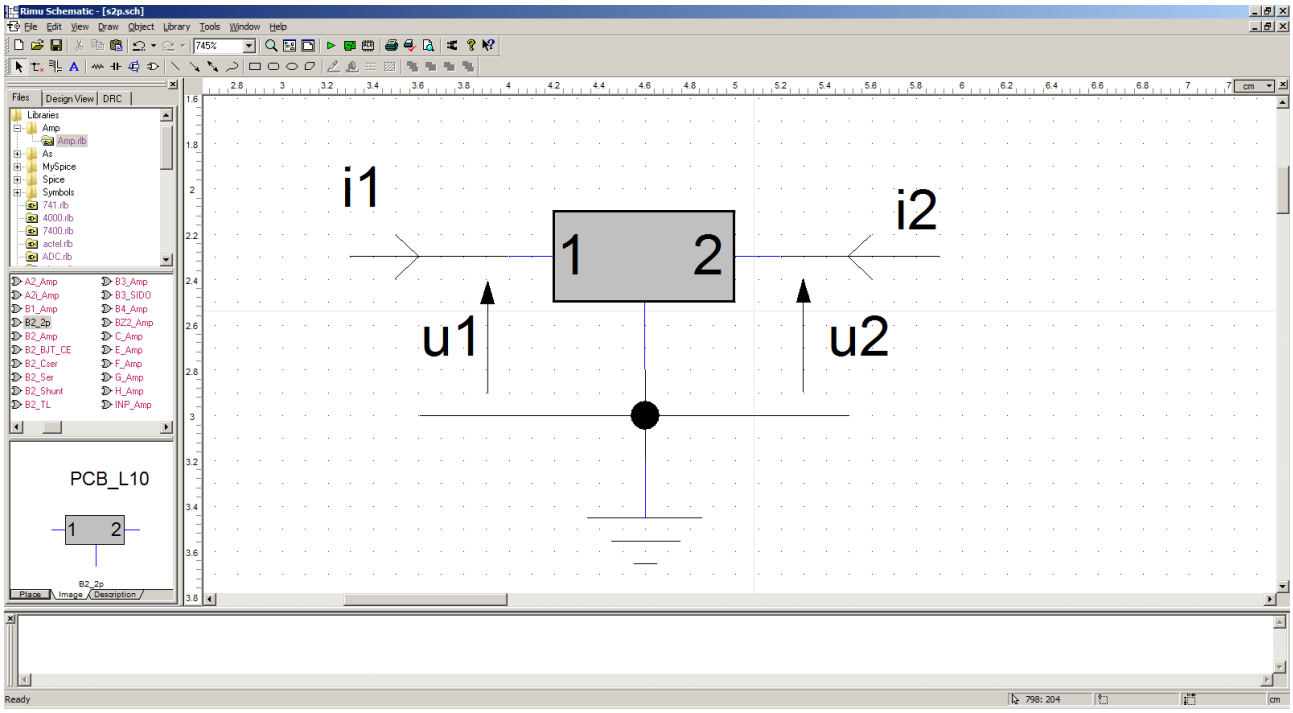


Fig.3.1.1. Two port network.

Definition of s-parameters for two port network is a matrix

$$\begin{bmatrix} b1 \\ b2 \end{bmatrix} = \begin{bmatrix} s11 & s12 \\ s21 & s22 \end{bmatrix} \begin{bmatrix} a1 \\ a2 \end{bmatrix} \qquad [3.1.1]$$

where, 'power waves' **a**, **b** can be represented by port's voltages and currents as:

$$b1 = \frac{u1 - i1R0}{2\sqrt{R0}} \quad a1 = \frac{u1 + i1R0}{2\sqrt{R0}}$$
$$b2 = \frac{u2 - i2R0}{2\sqrt{R0}} \quad a2 = \frac{u2 + i2R0}{2\sqrt{R0}} \qquad [3.1.2]$$

The 'power waves' definition in [3.1.2] assumes that reference impedance for all ports is **R0**, i.e. it is the same and real.

Ideal coupler is 'made' with 3 ideal amplifiers and 2 resistors of value equal with reference impedance **R0.**

Internal schematics is shown in figure 3.1.2.

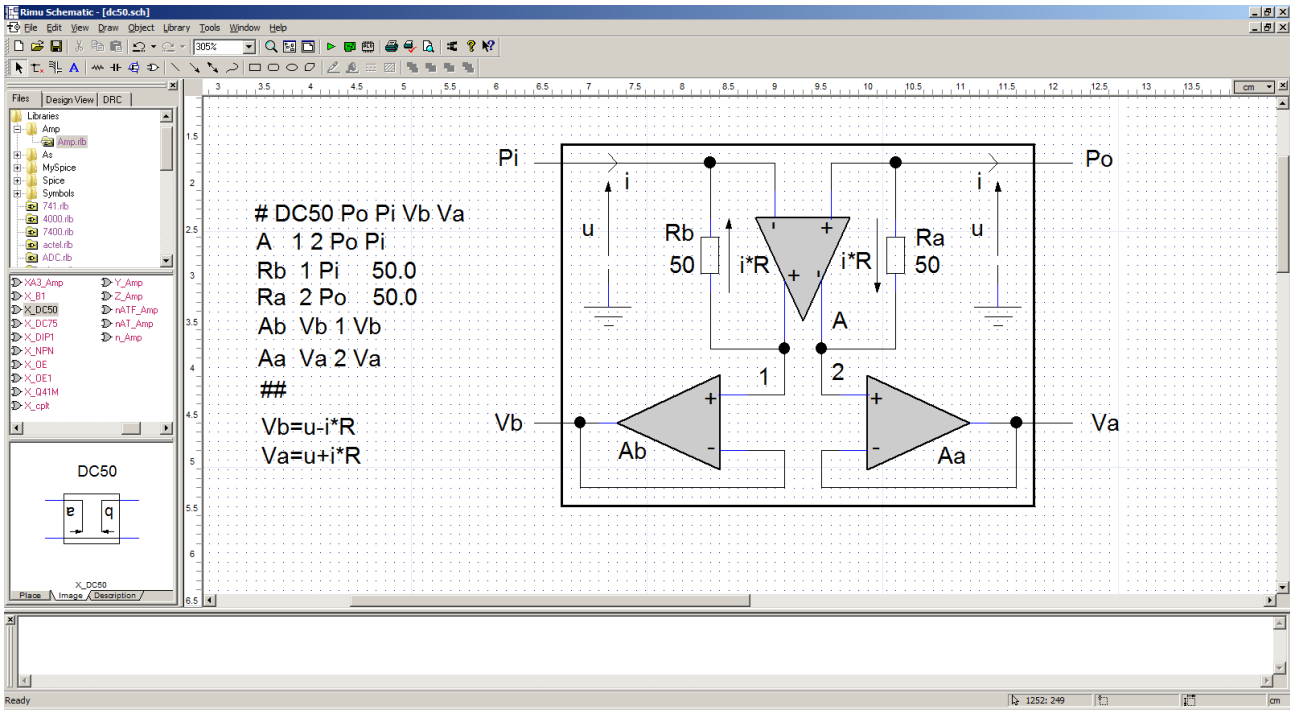


Fig.3.1.2. Ideal coupler internal schematics.

Ports **Pi**, **Po** corresponds to input power port and transmitted power port respectively. From network perspective, voltage and input current at **Pi** port is equal with voltage and output current of **Po** port. Thus for a signal path there is equivalent short-circuit between **Po**, **Pi ports with no impact on probed network**. However, current and voltage are sensed internally and voltage signals at output ports **Va**, **Vb are proportional to 'incident power wave' a**

$$Va = u + i\,R0 \qquad [3.1.3]$$

and '**reflected power wave**' **b**

$$Vb = u - i\,R0 \qquad [3.1.4]$$

Scaling factor (1/(2sqrt(R0)) is not important, as all s-parameters of the network can be expressed as a ratios of voltage signals produced by ideal coupler.

$$s11 = \frac{b1}{a1}_{a2=0} = \frac{u1 - i1R0}{u1 + i1R0} \qquad s21 = \frac{b2}{a1}_{a2=0} = \frac{u2 - i2R0}{u1 + i1R0}$$
$$s12 = \frac{b1}{a2}_{a1=0} = \frac{u1 - i1R0}{u2 + i2R0} \qquad s22 = \frac{b2}{a2}_{a1=0} = \frac{u2 - i2R0}{u2 + i2R0} \qquad [3.1.5]$$

The incident power port at port is zero (**a = 0**) when port is terminated by a reference impedance **R0** (i.e. **u=-i R0**). Therefore, matched load and source have to be used in a test network if s-parameters calculation are to be straightforward.

3.2. Active splitter example explained.

Summary: voltage gains and s-parameters test network.

3.2.1 Voltage gains and s-parameters in a test network

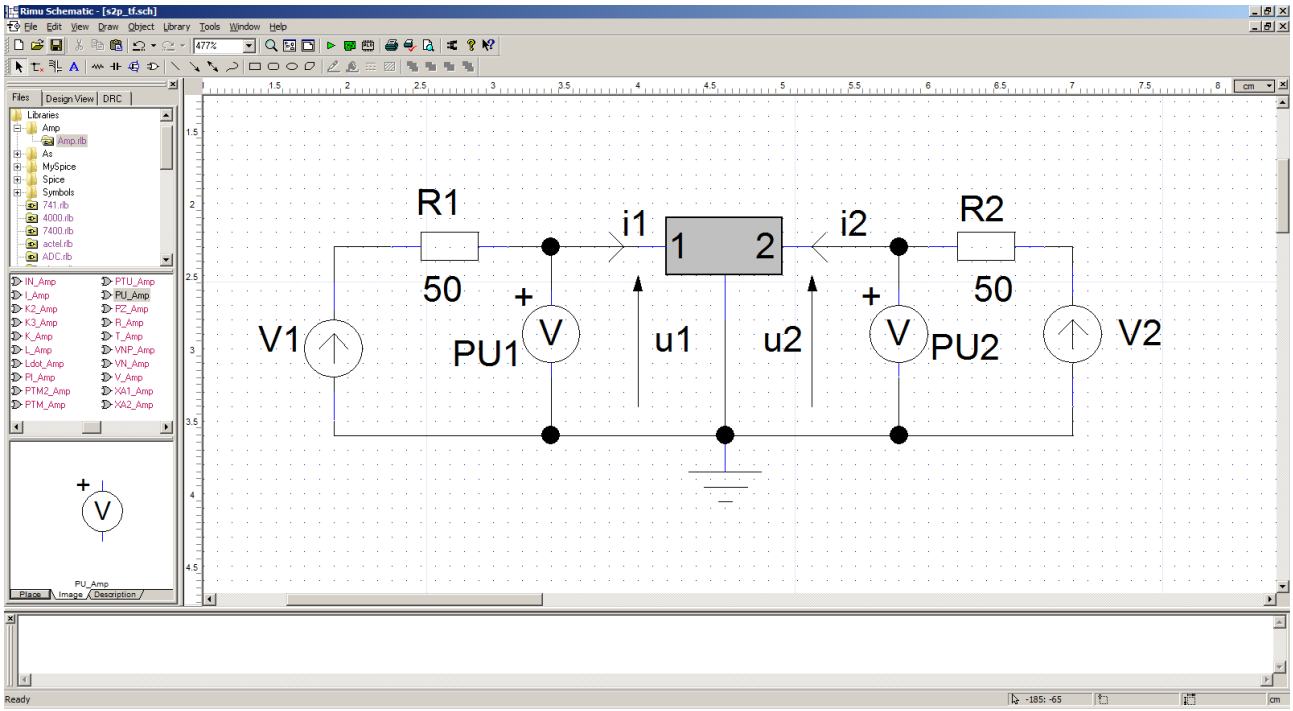


Fig.3.2.1. S-parameters test network with voltage probes.

When in the test network source R1 and load R2 impedance are the same and equal with R0 reference impedance R1=R2=R0, s-parameters can be expressed by voltage gains.

$$s11=\frac{u1-i1R0}{u1+i1R0}_{V2=0}=\frac{u1-i1R0}{u1+i1R1}=\frac{u1-\frac{V1-u1}{R1}R0}{V1}=2\frac{u1}{V1}-1=2\mathrm{K}_{u1/V1}-1 \qquad [3.2.1]$$

$$s21=\frac{u2-i2R0}{u1+i1R0}_{V2=0}=\frac{u2-\frac{-u2}{R2}R0}{V1}=2\frac{u2}{V1}=2\mathrm{K}_{u2/V1} \qquad [3.2.2]$$

$$s22=\frac{u2-i2R0}{u2+i2R0}_{V1=0}=\frac{u2-i2R0}{u2+i2R2}=\frac{u2-\frac{V2-u2}{R2}R0}{V2}=2\frac{u2}{V2}-1=2\mathrm{K}_{u2/V2}-1 \qquad [3.2.3]$$

$$s12=\frac{u1-i1R0}{u2+i2R0}_{V1=0}=\frac{u1-\frac{-u1}{R1}R0}{V2}=2\frac{u1}{V2}=2\mathrm{K}_{u1/V2} \qquad [3.2.4]$$

3.3. Stability analysis example explained.

Summary: stability factors, stability circles, frequency models.

3.3.1. Stability factors.

Two port device is said to be stable when [5]:

$$|\Gamma_{inp}|=\left|s_{11}+\frac{s_{12}s_{21}\Gamma_L}{1-s_{22}\Gamma_L}\right|_{|\Gamma_L|<1}<1 \quad |\Gamma_{out}|=\left|s_{22}+\frac{s_{12}s_{21}\Gamma_S}{1-s_{11}\Gamma_S}\right|_{|\Gamma_S|<1}<1 \qquad [3.3.1.1]$$

Equivalent conditions can be expressed by **K**-factor and $\mathbf{B_1}$-factor:

$$K=\frac{1-|s_{11}|^2-|s_{22}|^2+|\Delta|^2}{2|s_{12}s_{21}|}>1 \quad B_1=1+|s_{11}|^2-|s_{22}|^2-|\Delta|^2>0 \qquad [3.3.1.2]$$

K-factor and $\mathbf{B_1}$-factor are most popular stability indicators:

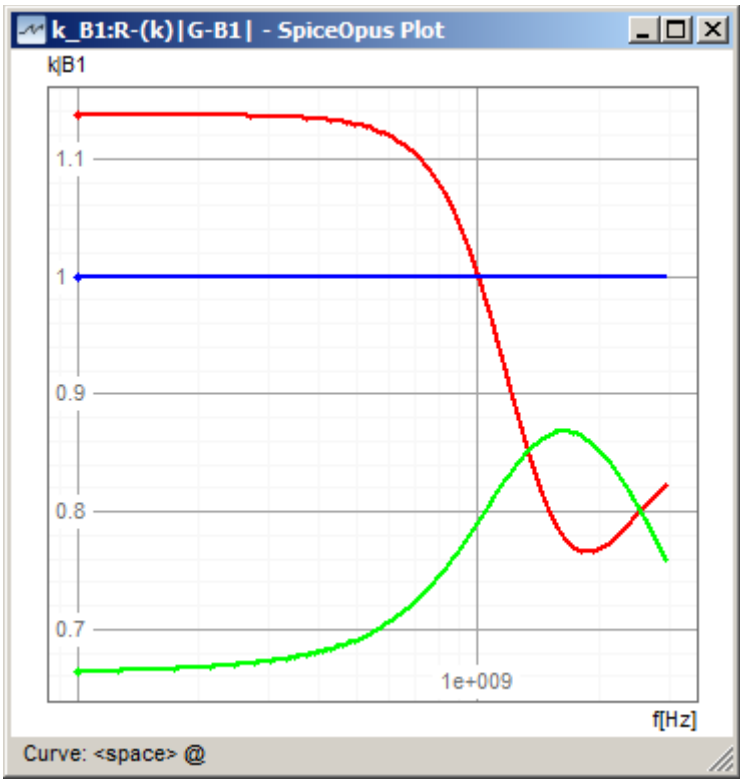


Fig. 3.3.1.1. K-factor red plot, B1-green plot for circuit potentially unstable above 1GHz.

3.3.2. Stability circles.

Load stability circle [5] is defined by equation:

$$\left|\Gamma_{inp}(\Gamma_L)\right| = \left| s_{11} + \frac{s_{12}s_{21}\Gamma_L}{1 - s_{22}\Gamma_L} \right| = 1 \qquad [3.3.2.1]$$

Solution of equation [3.3.2.1] is a circle with center C_L and radius R_L:

$$C_L = \frac{s_{11}\Delta^* + s_{22}^*}{|\Delta|^2 - |s_{22}|^2} \quad R_L = \left| \frac{s_{21}s_{12}}{|\Delta|^2 - |s_{22}|^2} \right| \qquad [3.3.2.2]$$

Source stability circle is defined by equation:

$$\left|\Gamma_{out}(\Gamma_S)\right| = \left| s_{22} + \frac{s_{12}s_{21}\Gamma_S}{1 - s_{11}\Gamma_S} \right| = 1 \qquad [3.3.2.3]$$

Solution of equation [3.3.2.3] is a circle with center C_S and radius R_S:

$$C_S = \frac{s_{22}\Delta^* + s_{11}^*}{|\Delta|^2 - |s_{11}|^2} \quad R_S = \left| \frac{s_{21}s_{12}}{|\Delta|^2 - |s_{11}|^2} \right| \qquad [3.3.2.4]$$

Load and source stability circles define the borderline between 'stable area' and 'unstable area'. For unconditional stability unity circle should be in 'stable area'.

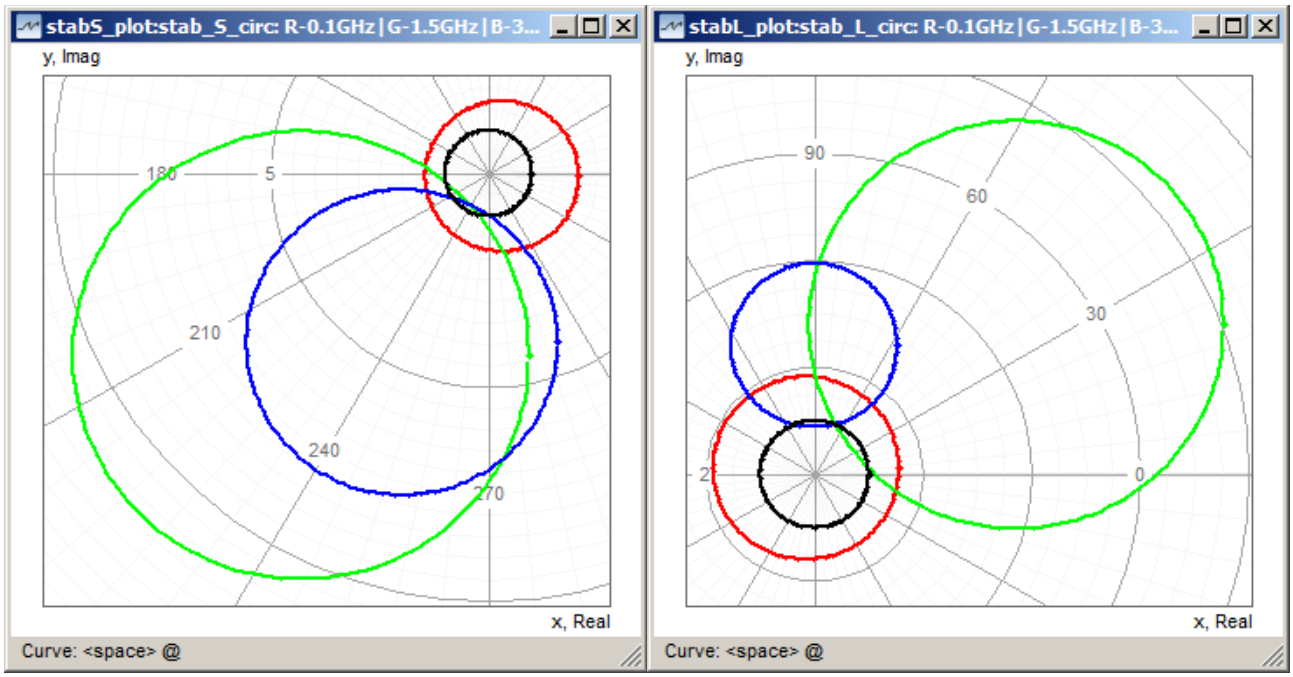


Fig. 3.3.2.1. Stability circles for f={0.1GHz(red), 1.5GHz(green), 3.0GHz(blue)}. Unity circle is black.

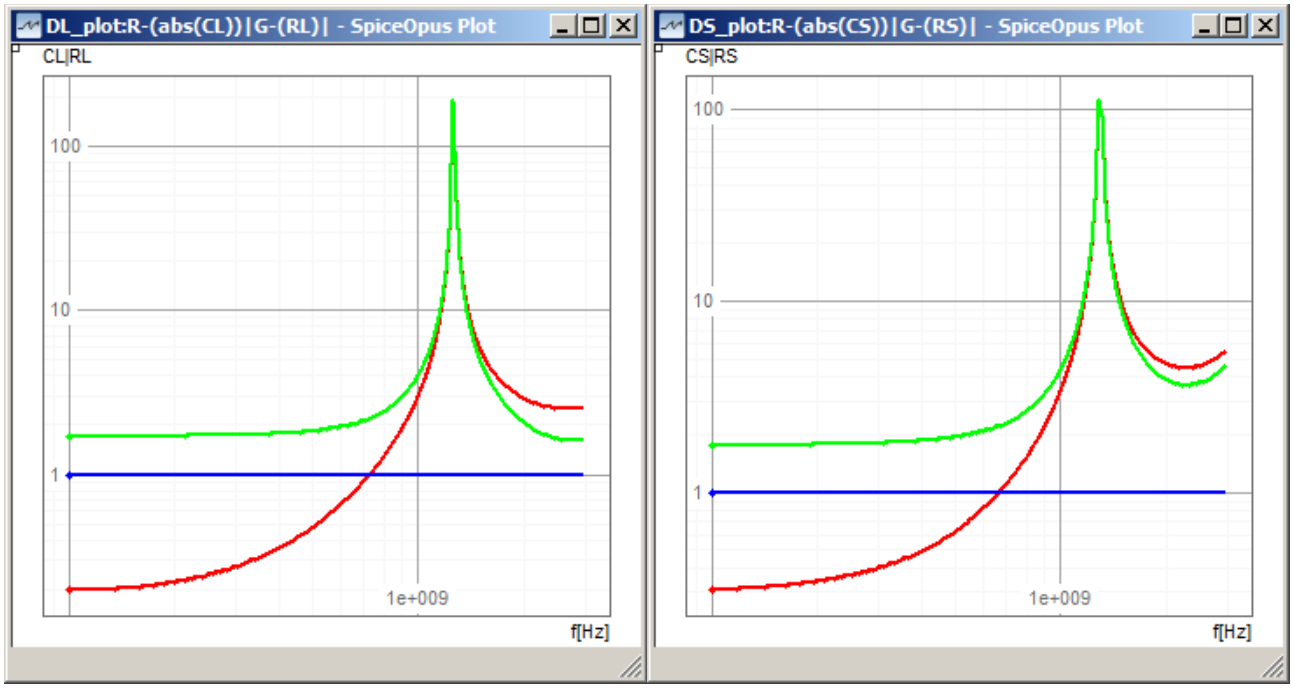


Fig. 3.3.2.2. Distance to the stability circles center and stability circles radius vs frequency.

3.3.3. Small signal transistor model.

OPUS Spice was used to determine point of operation and small signal model of the transistor.

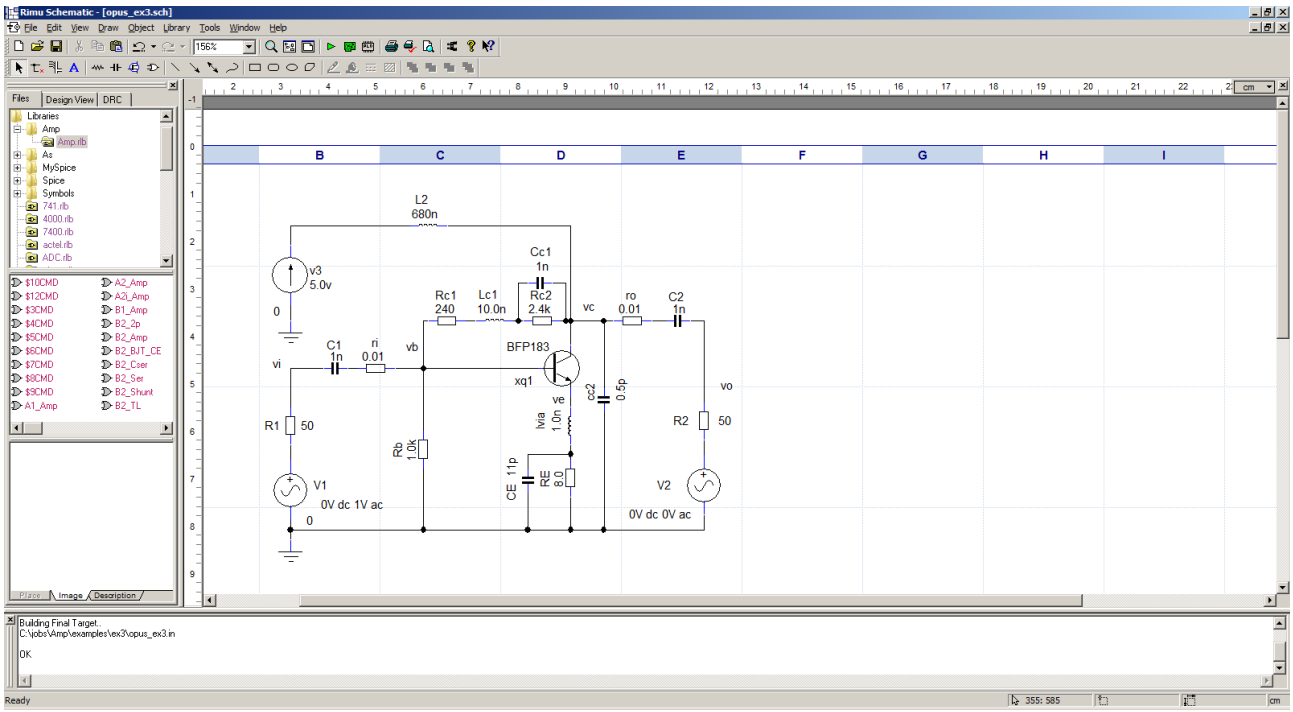


Fig. 3.3.3.1. Spice model of the gain block.

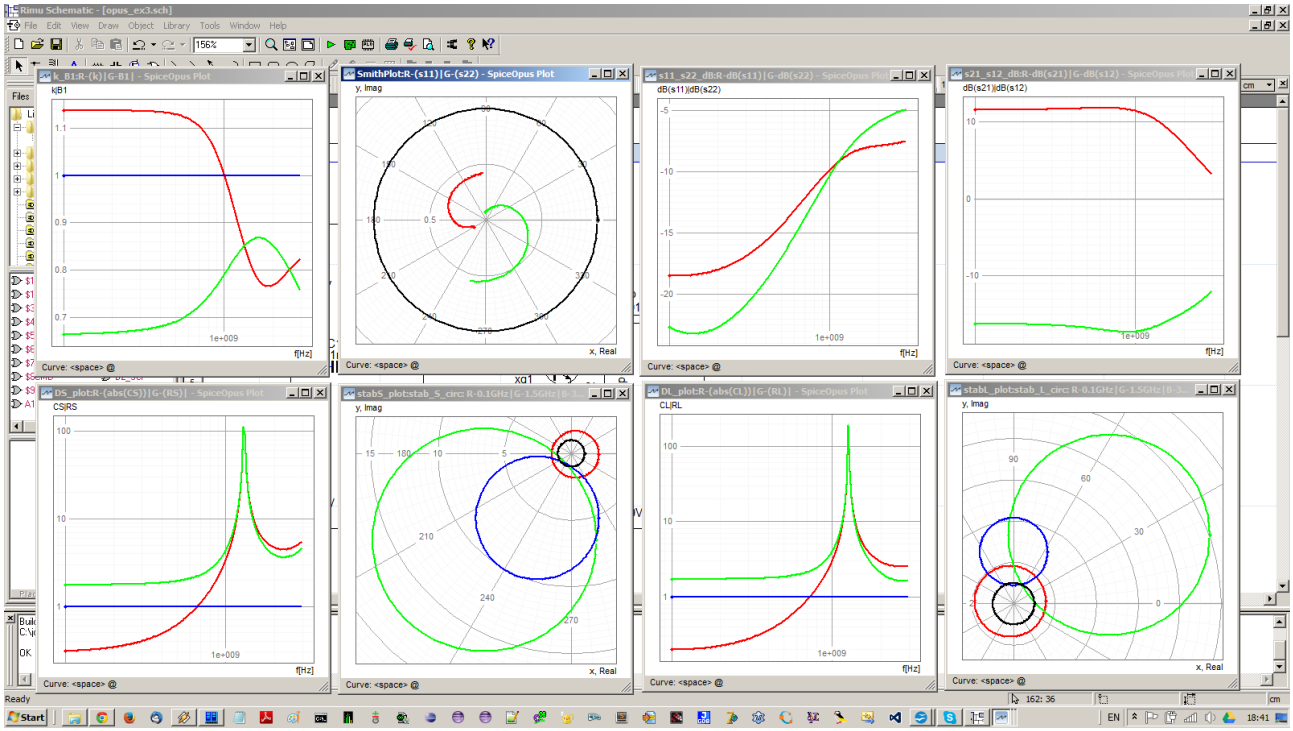


Fig. 3.3.3.2. Results of analysis of the gain block in Spice.

SpiceOpus Command Window

File Edit Control Window Help

```
Warning: BJT model m_bfp183:xq1, parameter fc limited to 0.9999
v(vc) = 5.000000e+000
v(vb) = 1.095786e+000
v(ve) = 2.403004e-001
 BJT: Bipolar Junction Transistor
 device       q1:xq1
 model     m_bfp183:
              m          1
             ic     0.0297
             ib   0.000383
             ie      -0.03
            vbe      0.848
            vbc      -3.81
             gm      0.938
            gpi     0.0148
            gmu   1.68e-013
             gx      0.345
             go   0.000543
            cpi   2.34e-011
            cmu   9.8e-014
            cbx   2.3e-014
           ccbs          0
```

Fig. 3.3.3.2. Parameters of the transistor in the point of operation.

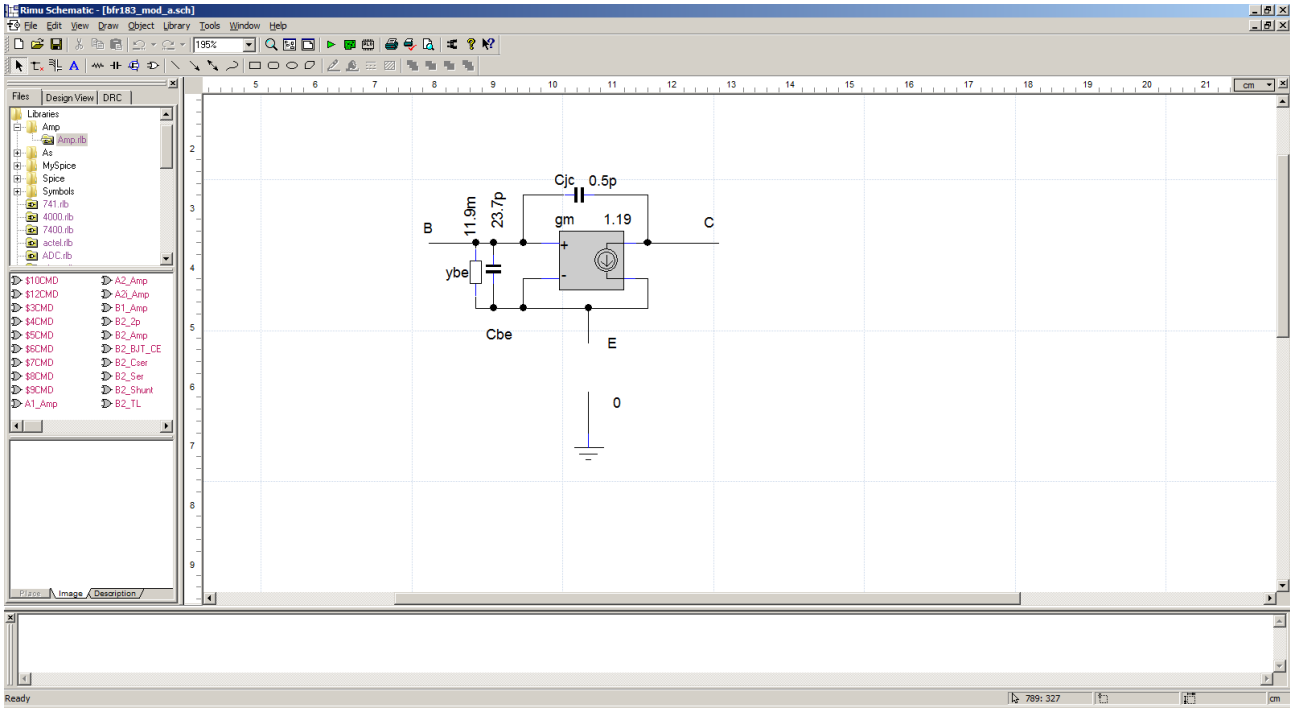


Fig. 3.3.3.3. Hybrid-pi small signal model of the transistor <_bfp_pi_a.mod> <_bfp183a.lib>.

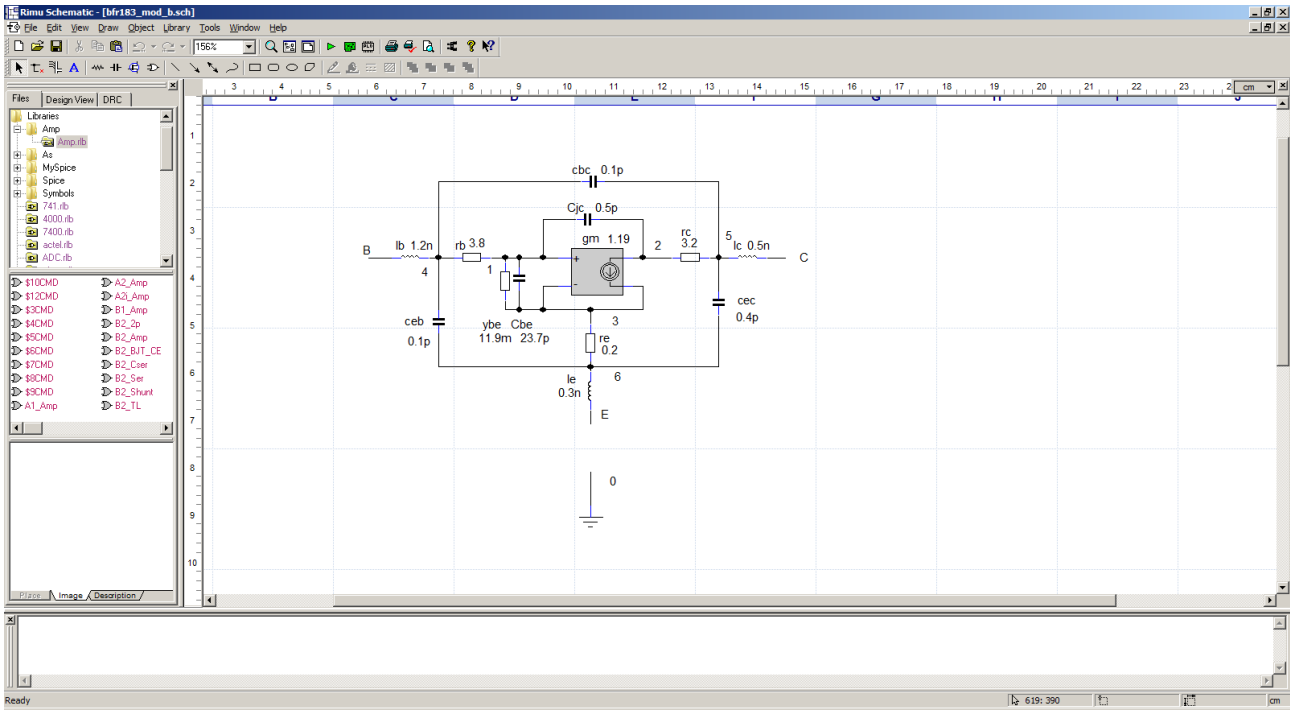


Fig. 3.3.3.4. Hybrid-pi small signal model with package parasitic components <_bfp_pi_a.mod> <_bfp183b.lib>.

Parameters of the model are calculated by <ex3_mod.sce> script. Parasitic package components are extracted from Spice model.

3.3.4. Capacitor FQ model.

FQ model of a capactior is medium accurancy equivalent series RLC model.

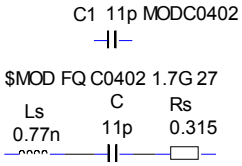


Fig. 3.3.4.1. Equivalent **FQ** model of capacitor C=11p with **F**=1.7GHz, **Q**=27

Two model parameters are: **F** – rezonant frequency and **Q** – equivalent quality factor as in:

$$Z(s)=\frac{s^2LC+sRC+1}{sC}=\frac{1}{sC}\frac{s^2+s\frac{R}{L}+\frac{1}{LC}}{\frac{1}{LC}}=\frac{1}{sC}\frac{s^2+s\frac{\Omega}{Q}+\Omega^2}{\Omega^2}\qquad \Omega^2=\frac{1}{LC}\qquad Q=\frac{\Omega L}{R} \qquad [3.3.4.1]$$

For a given **C, F,Q**, parasitic **R,L** are calculated according to equations:

$$\Omega=2\pi F\qquad L=\frac{1}{\Omega^2 C}\qquad R=\frac{\Omega L}{Q} \qquad [3.3.4.2]$$

Parameters of **FQ** model have to be estimated by analyzing frequency response of impedance of a given capacitor (C= 11p).

Fig. 3.3.4.2 Impedance frequency response of C = 11p [0402, NP0 – Kemet Spice].

From the plot one can estimate that a resonont frequency is **F=1.7GHz** and eqivalent series resitance is **R=0.3** (blue plot - relatively flat up from 100MHz up to 3GHz). For F,R Cmodel_v1.sce script calculates series inductance L=0.77nH and equivalent quality factor **Q = 27**.

3.3.5. Coil FQ model.

FQ model of a resistor is medium accurancy equivalent parallel RLC model.

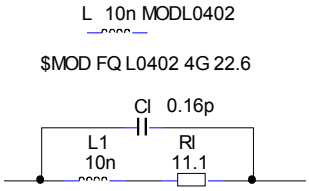


Fig. 3.3.5.1. Equivalent **FQ** model of L=10n coil with **F**=4GHz, **Q**=22.6

Two model parameters are: **F** – rezonant frequency, **Q** – equivalent quality factor as in:

$$Z(s)=\frac{sL+R}{s^2LC+sRC+1}=\frac{sL+R}{LC\left(s^2+s\frac{R}{L}+\frac{1}{LC}\right)}=\frac{(sL+R)\Omega^2}{s^2+s\frac{\Omega}{Q}+\Omega^2}\quad \Omega^2=\frac{1}{LC}\quad Q=\frac{\Omega L}{R} \qquad [3.3.5.1]$$

For a given **L, F,Q**, parasitic **R,C** are calculated according to equations:

$$\Omega=2\pi F\quad C=\frac{1}{\Omega^2 L}\quad R=\frac{\Omega L}{Q} \qquad [3.3.5.2]$$

Parameters of **FQ** model have to be estimated by analyzing specification and frequency response of impedance of a given coil (L= 10n).

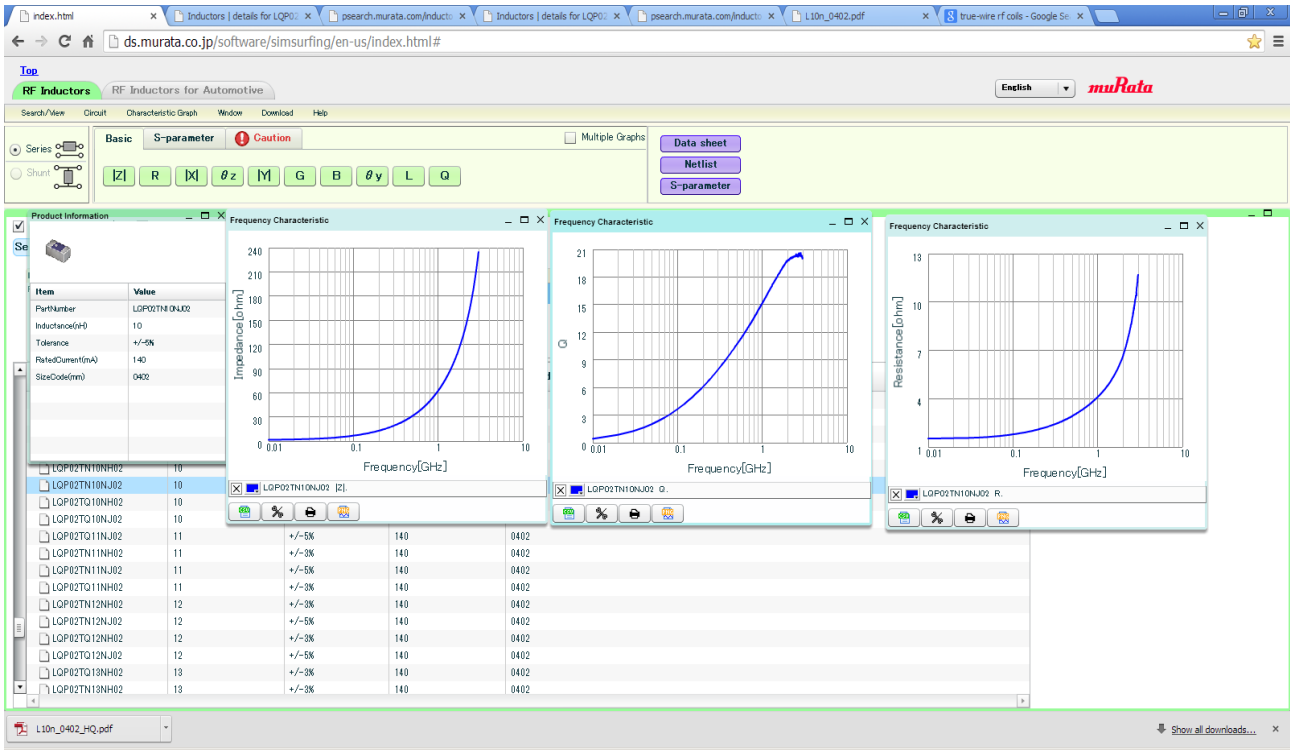
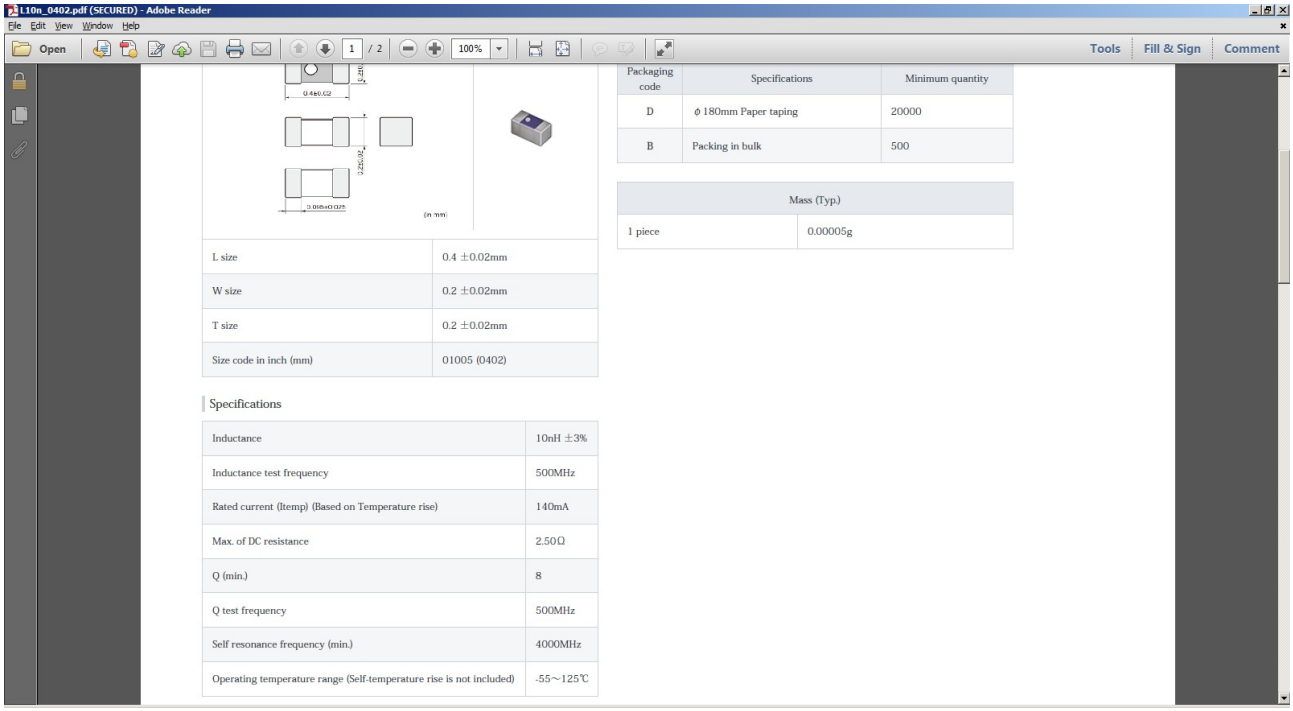


Fig. 3.3.5.2 Impedance frequency response of L = 10n [0402, TN-Murata]. Resonant Frequency 4GHz is above scale.

Coil data sheet specifies resonant frequency **F=4 GHz** and quality factor **q=8** measured at f**q=500Mhz**.



Packaging code	Specifications	Minimum quantity
D	ϕ 180mm Paper taping	20000
B	Packing in bulk	500

Mass (Typ.)	
1 piece	0.00005g

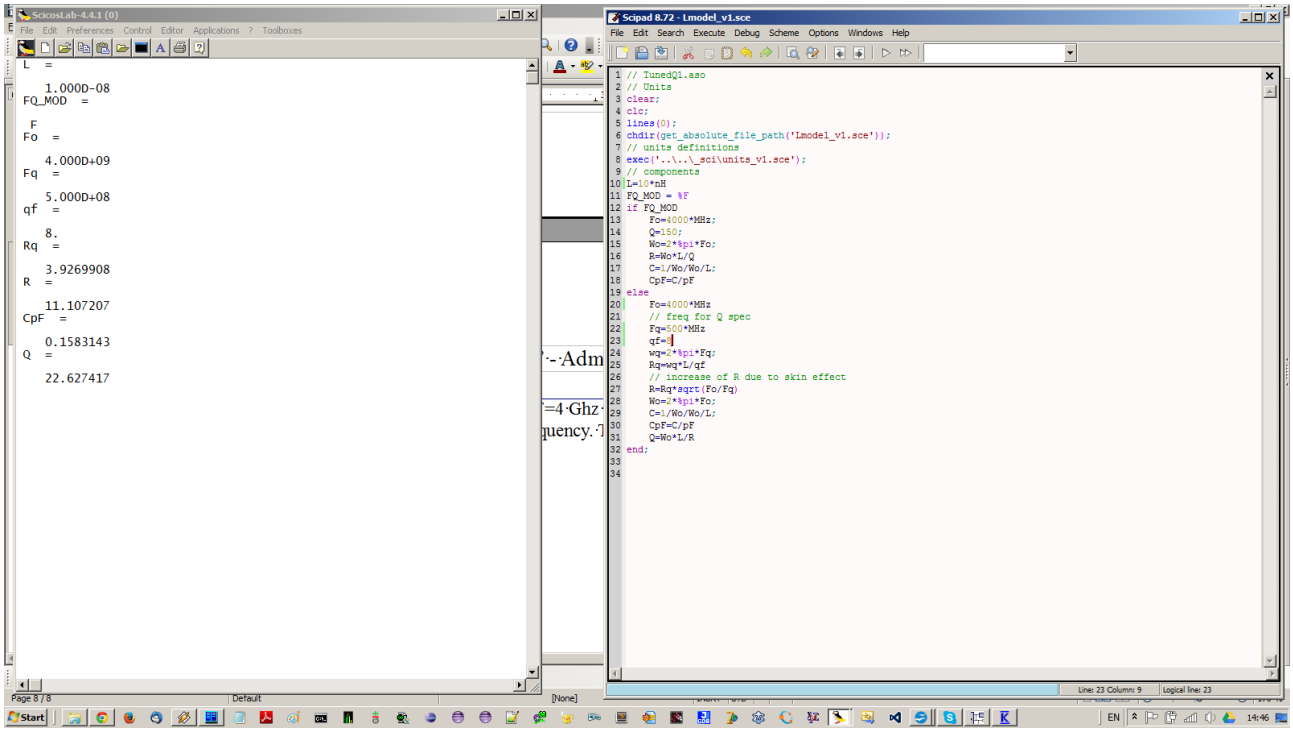
L size	0.4 ±0.02mm
W size	0.2 ±0.02mm
T size	0.2 ±0.02mm
Size code in inch (mm)	01005 (0402)

Specifications

Inductance	10nH ±3%
Inductance test frequency	500MHz
Rated current (Itemp) (Based on Temperature rise)	140mA
Max. of DC resistance	2.50Ω
Q (min.)	8
Q test frequency	500MHz
Self resonance frequency (min.)	4000MHz
Operating temperature range (Self-temperature rise is not included)	-55~125℃

Fig. 3.3.5.3. Coil specification.

FQ model requires extrapolation of quality factor at resonant frequency. This is done by scilab script <Lmodel_v1.sce>, which uses square root frequency approximation to account for skin effect.



```
 L  =

   1.000D-08
FQ_MOD  =

 F
Fo  =

   4.000D+09
Fq  =

   5.000D+08
qf  =

   8.
Rq  =

   3.9269908
R  =

   11.107207
CpF  =

   0.1583143
Q  =

   22.627417
```

```
// TunedQ1.sce
// Units
clear;
clc;
lines(0);
chdir(get_absolute_file_path('Lmodel_v1.sce'));
// units definitions
exec('..\..\_sci\units_v1.sce');
// components
L=10*nH
FQ_MOD = %F
if FQ_MOD
    Fo=4000*MHz;
    Q=150;
    Wo=2*%pi*Fo;
    R=Wo*L/Q
    C=1/Wo/Wo/L;
    CpF=C/pF
else
    Fo=4000*MHz
    // freq for Q spec
    Fq=500*MHz
    qf=8
    wq=2*%pi*Fq;
    Rq=wq*L/qf
    // increase of R due to skin effect
    R=Rq*sqrt(Fo/Fq)
    Wo=2*%pi*Fo;
    C=1/Wo/Wo/L;
    CpF=C/pF
    Q=Wo*L/R
end;
```

Fig. 3.3.5.4 Calculations of FQ model parameters for coil <Lmodel_v1.sce>.

3.3.6. Resistor FQ model.

FQ model of a resistor is medium accurancy equivalent parallel RLC model.

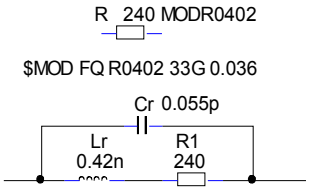


Fig. 3.3.6.1. Equivalent **FQ** model of resistor R=240 with **F**=33GHz, **Q**=0.36

Two model parameters are: **F** – rezonant frequency, **Q** – equivalent quality factor as in:

$$Z(s)=\frac{sL+R}{s^2LC+sRC+1}=\frac{sL+R}{LC\left(s^2+s\frac{R}{L}+\frac{1}{LC}\right)}=\frac{(sL+R)\Omega^2}{s^2+s\frac{\Omega}{Q}+\Omega^2}\quad \Omega^2=\frac{1}{LC}\quad Q=\frac{\Omega L}{R} \qquad [3.3.6.1]$$

For a given **R, F,Q**, parasitic **L,C** are calculated according to equations:

$$\Omega=2\pi F\quad L=\frac{RQ}{\Omega}\quad C=\frac{1}{\Omega^2 L} \qquad [3.3.6.2]$$

Parameters of **FQ** model have to be estimated by analyzing frequency response of impedance of a given resistor. As indicated in literature[6], resistor parasitics are size dependent.

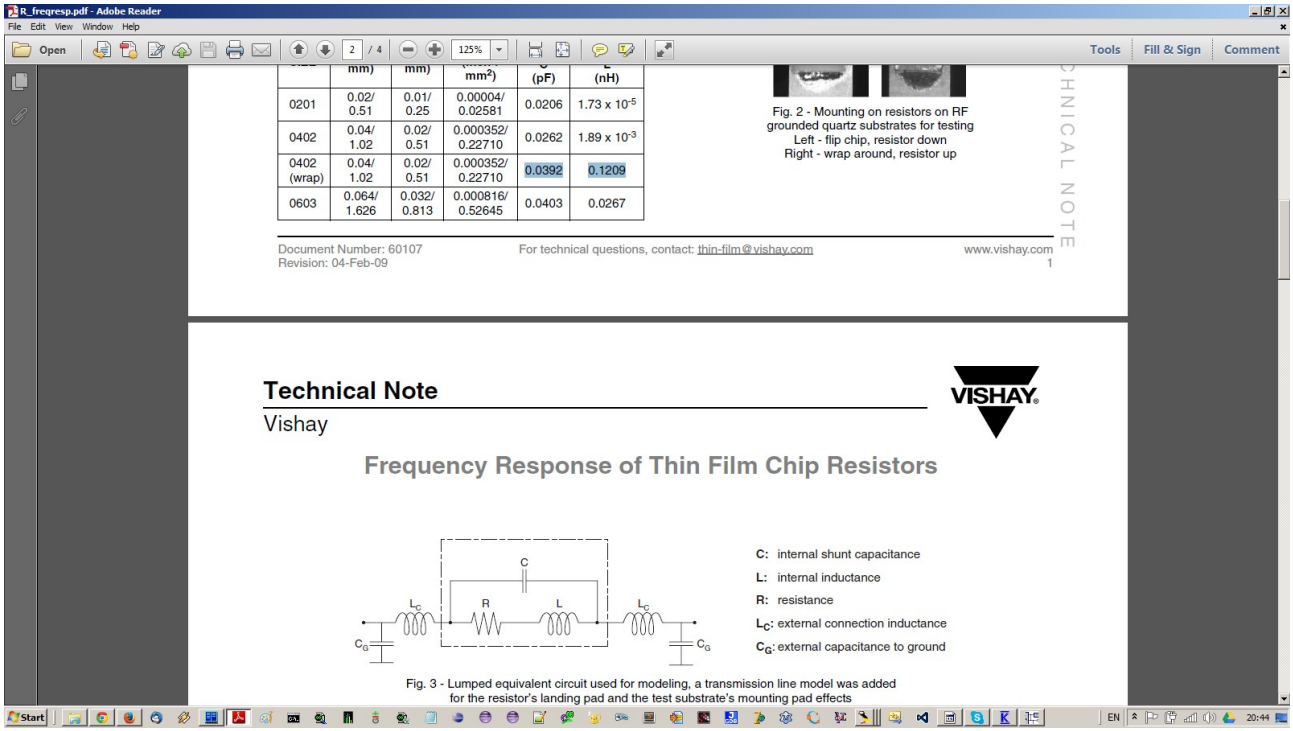


Fig. 3.3.6.2. Impedance frequency response of resitors – Vishay technical note .

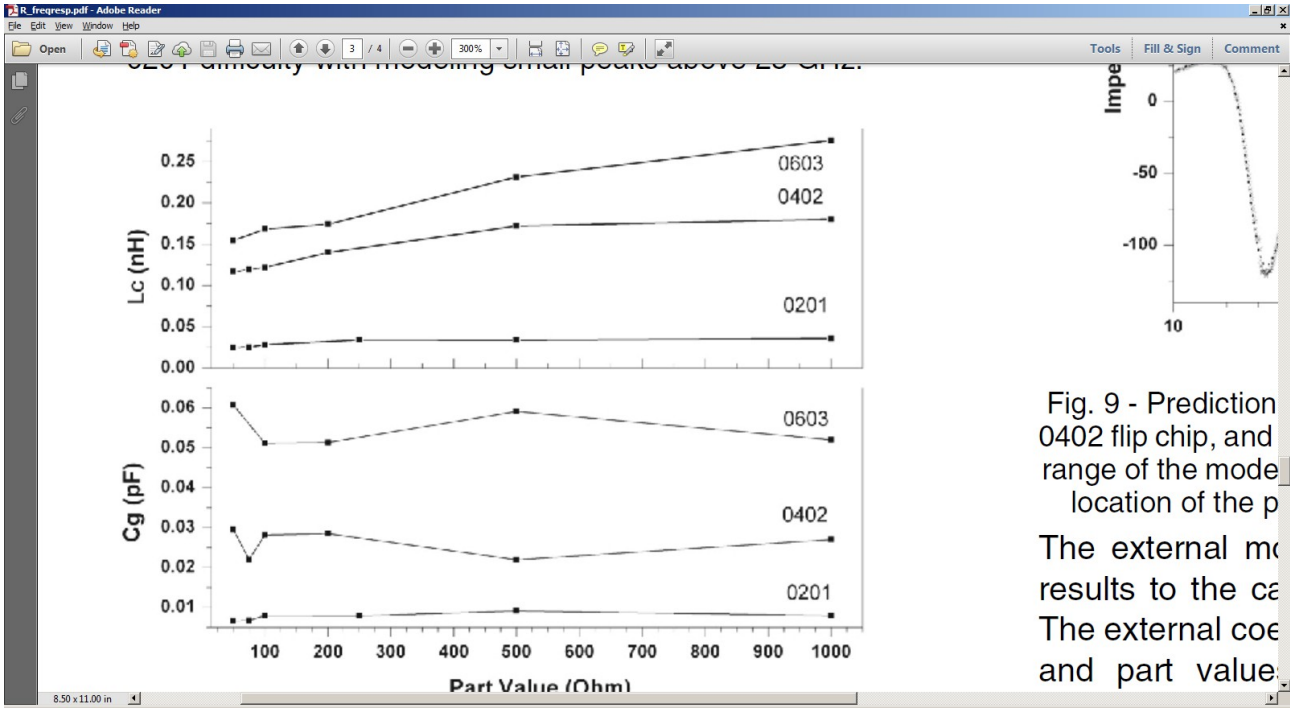


Fig. 3.3.6.3. Impedance frequency response of resitors continued – Vishay technical note .

Using data presented in Vishay techincal note, it was assumed to use equivalent series inductance **L+Lc+Lc=0.12+0.30 = 0.42nH** and equivalent parallel capacitance for 2 port device **C+Cg/2= 0.04+0.015 = 0.055pF**. For assumed parsitc components **FQ** parameters are calculated by a script Rmodel_v1.sce

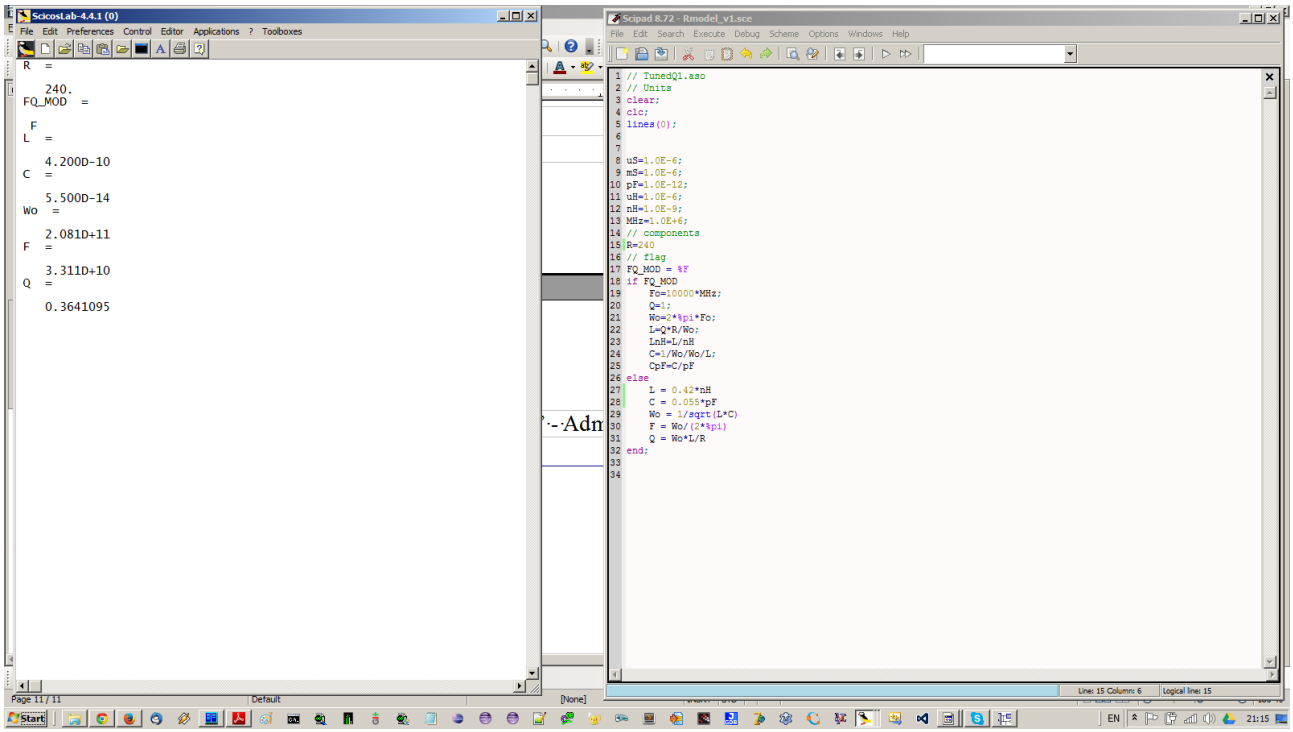


Fig. 3.3.6.4. Calculations of FQ model parameters for resistor <Rmodel_v1.sce>.

3.4. Coupled lines example explained.

Summary: even, odd modes, coupled line modeling, decoupling transformations

3.4.1. Even, odd modes.

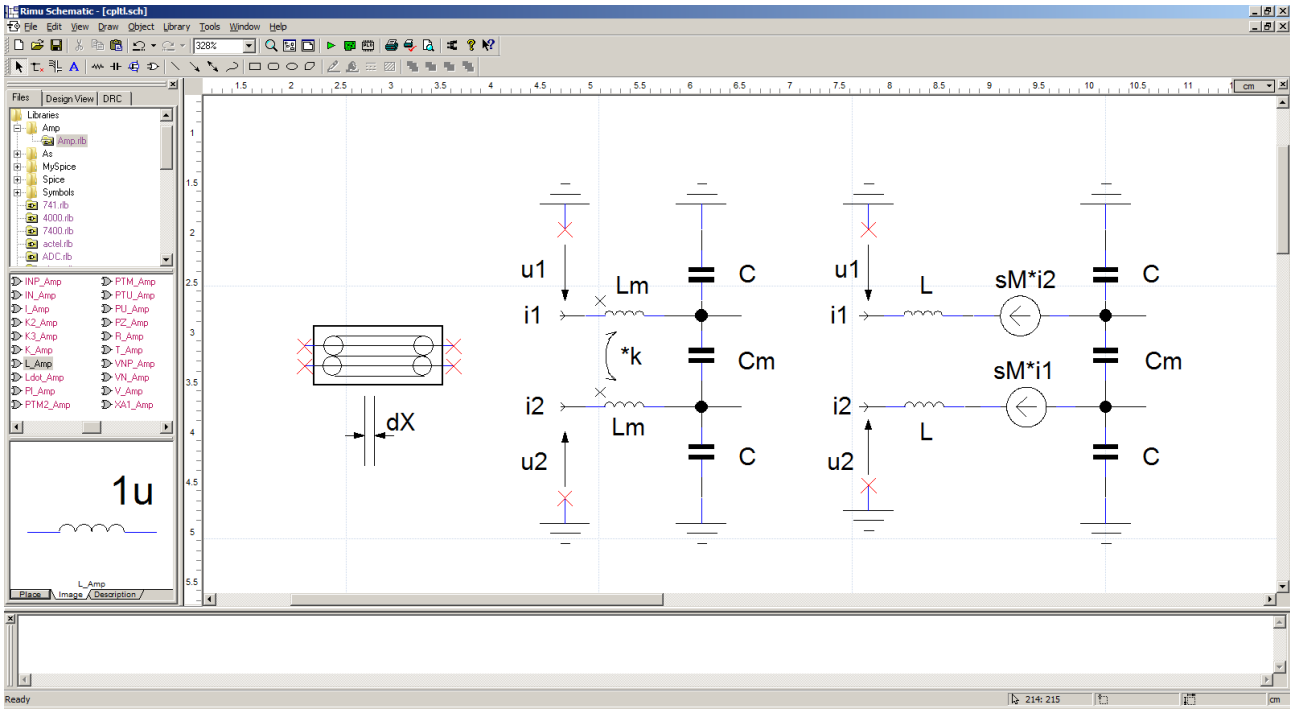


Fig.3.4.1.1. Equivalent lumped model for a segment of lossless symmetric coupled lines.

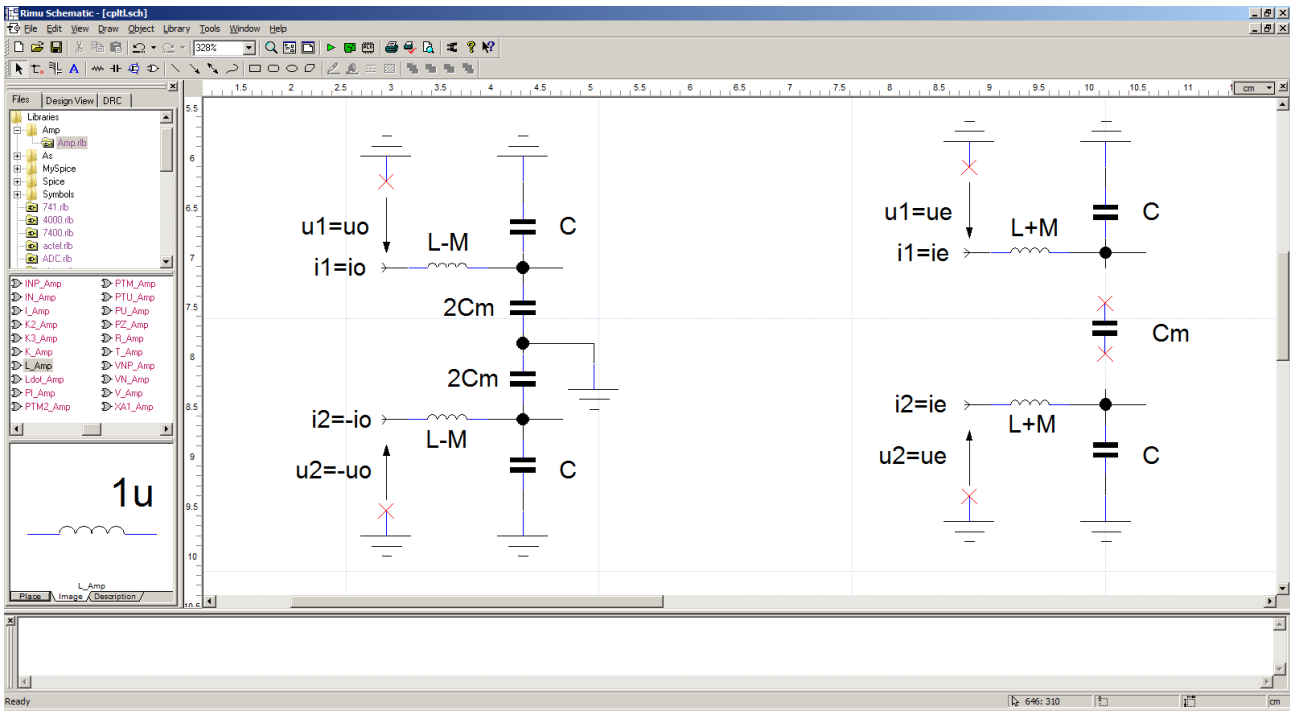


Fig.3.4.1.2. Equivalent lumped models for a coupled lines segment in odd and even mode.

In a odd excitation mode both lines are driven mode with equal magnitude and opposite phase. Current flowing in opposite directions cancels magnetic field (reduced equivalent inductance). Asymmetric voltage increases electric coupling (increased equivalent capacitance).
Due to symmetry the signal flow is like in two uncoupled lines, both with odd characteristic impedance and phase velocity:

$$Z_{odd}=\sqrt{\frac{l-m}{c+2\mathrm{cm}}}\quad \nu_{odd}=\frac{1}{\sqrt{(l-m)(c+2\mathrm{cm})}}\quad Z_{diff}=\frac{2\,uo}{io}=2\,Z_{odd} \qquad [3.4.1.1]$$

In an even excitation mode lines are driven mode with equal magnitude and phase. Current flowing in the same direction increases magnetic field (increased equivalent inductance). Symmetric voltage decreases electric coupling (decreased equivalent capacitance).
Due to symmetry the signal flow is like in two uncoupled lines, both with even characteristic impedance and phase velocity:

$$Z_{even}=\sqrt{\frac{l+m}{c}}\quad \nu_{even}=\frac{1}{\sqrt{(l+m)c}}\quad Z_{comm}=\frac{ue}{2\,ie}=\frac{Z_{even}}{2} \qquad [3.4.1.2]$$

where:

$$L=l\,\Delta x\quad M=m\,\Delta x\quad C=c\,\Delta x\quad Cm=cm\,\Delta x \qquad [3.4.1.3]$$

For pure odd-even excitation not only lines can be represented as uncoupled, but also relation between input currents and 'mode' currents can be easily determined (one uncoupled line determines signal flow when in 'pure' mode).

3.4.2. Coupled line modeling.

Signal flow in a symmetric lossless transmission line segment is defined by set of equations for a voltage drop and a current drop per unit length:

$$-\begin{bmatrix}\Delta u1\\ \Delta u2\end{bmatrix}=-\begin{bmatrix}\frac{\partial u1}{\partial x}\Delta x\\ \frac{\partial u2}{\partial x}\Delta x\end{bmatrix}=\begin{bmatrix}L & M\\ M & L\end{bmatrix}\begin{bmatrix}\frac{d}{dt}i1\\ \frac{d}{dt}i2\end{bmatrix} \qquad [3.4.2.1]$$

$$-\begin{bmatrix}\Delta i1\\ \Delta i2\end{bmatrix}=-\begin{bmatrix}\frac{\partial i1}{\partial x}\Delta x\\ \frac{\partial i2}{\partial x}\Delta x\end{bmatrix}=\begin{bmatrix}C+Cm & -Cm\\ -Cm & C+Cm\end{bmatrix}\begin{bmatrix}\frac{d}{dt}u1\\ \frac{d}{dt}u2\end{bmatrix} \qquad [3.4.2.2]$$

The equation set is dependent (coupled) for variable set [u1,u2,i1,i2].

If new variable set is defined as even, odd voltages and currents:

$$ue=\frac{u1+u2}{2}\quad uo=\frac{u1-u2}{2}$$
$$ie=\frac{i1+i2}{2}\quad io=\frac{i1-i2}{2} \qquad [3.4.2.3]$$

and substituted into transmission line equations

$$u1=ue+uo\quad u2=ue-uo$$
$$i1=ie+io\quad i2=ie-io \qquad [3.4.2.4]$$

$$-\begin{bmatrix}\Delta(ue+uo)\\ \Delta(ue-uo)\end{bmatrix}=-\begin{bmatrix}\frac{\partial}{\partial x}(ue+uo)\Delta x\\ \frac{\partial}{\partial x}(ue-uo)\Delta x\end{bmatrix}=\begin{bmatrix}L & M\\ M & L\end{bmatrix}\begin{bmatrix}\frac{d}{dt}(ie+io)\\ \frac{d}{dt}(ie-io)\end{bmatrix} \qquad [3.4.2.5]$$

$$-\begin{bmatrix}\Delta(ie+io)\\ \Delta(ie-io)\end{bmatrix}=-\begin{bmatrix}\frac{\partial}{\partial x}(ie+io)\Delta x\\ \frac{\partial}{\partial x}(ie-io)\Delta x\end{bmatrix}=\begin{bmatrix}C+Cm & -Cm\\ -Cm & C+Cm\end{bmatrix}\begin{bmatrix}\frac{d}{dt}(ue+uo)\\ \frac{d}{dt}(ue-uo)\end{bmatrix} \qquad [3.4.2.6]$$

Now, line equations set can simplified into independent equation set (for uncoupled odd-even lines):

$$-\begin{bmatrix}\Delta ue\\ \Delta uo\end{bmatrix}=-\begin{bmatrix}\frac{\partial ue}{\partial x}\Delta x\\ \frac{\partial uo}{\partial x}\Delta x\end{bmatrix}=\begin{bmatrix}L+M & 0\\ 0 & L-M\end{bmatrix}\begin{bmatrix}\frac{d}{dt}ie\\ \frac{d}{dt}io\end{bmatrix} \qquad [3.4.2.7]$$

$$-\begin{bmatrix}\Delta ie\\ \Delta io\end{bmatrix}=-\begin{bmatrix}\frac{\partial}{\partial x}(ie)\Delta x\\ \frac{\partial}{\partial x}(io)\Delta x\end{bmatrix}=\begin{bmatrix}C & 0\\ 0 & C+2\mathrm{Cm}\end{bmatrix}\begin{bmatrix}\frac{d}{dt}ue\\ \frac{d}{dt}uo\end{bmatrix} \qquad [3.4.2.8]$$

Therefore, coupled line problem can be solved by converting port variable set **u=[u1,u2] i=[i1,i2]** into mode variable set **um=[uo,ue], im=[ie,io]**, calculating solution in 'mode' domain and converting back into port domain.

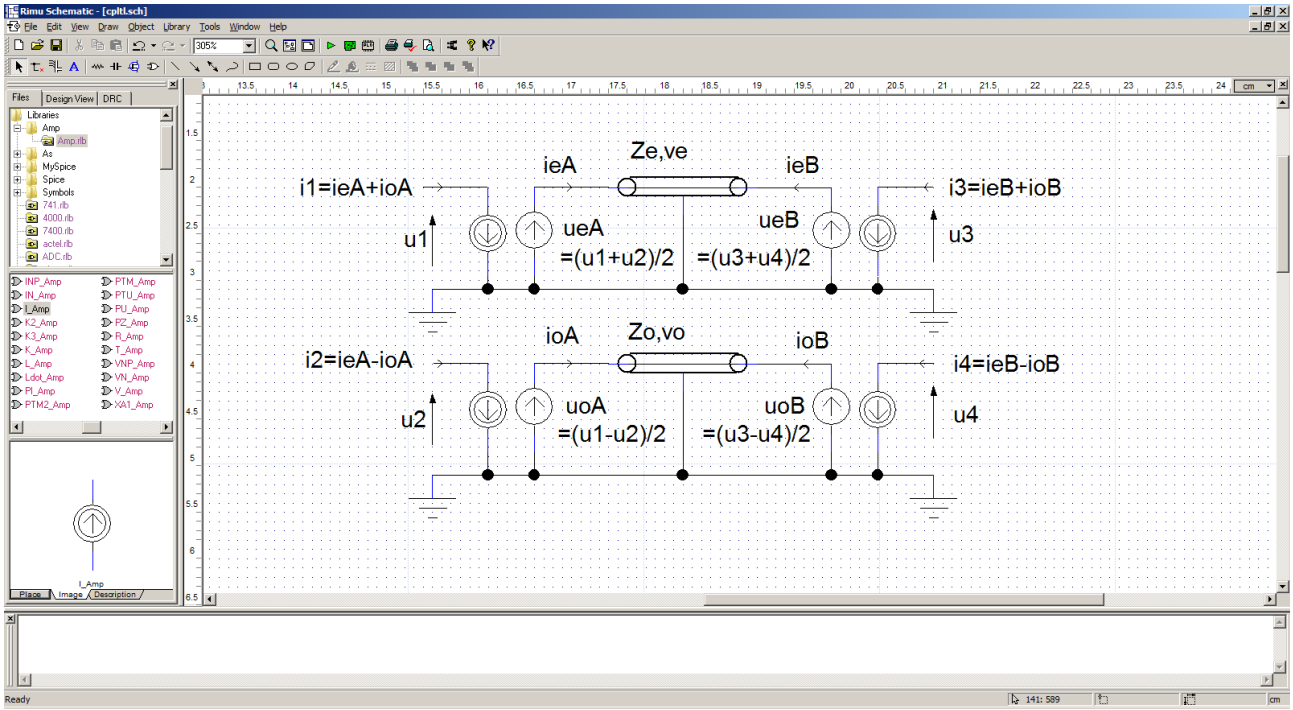


Fig.3.4.2. Model of two symmetrical lossless coupled lines.

3.4.3. Decoupling transformations.

Presented above decoupling model follows physical and network interpretation but is not a unique decomposition.

General solution to multiple multiple lossless lines can be found in a book [7] (together with links to Fortran programs for spice models generation).

The problem is formulated as a search for two mode transformations $\mathbf{T_u}$, $\mathbf{T_i}$ matrixes:

$$[U]=\boldsymbol{T_u}[U_m] \qquad [3.4.3.1]$$

$$[I]=\boldsymbol{T_i}[I_m] \qquad [3.4.3.2]$$

which convert coupled equation set in port domain [**U**,**I**]:

$$-\frac{\partial}{\partial x}[U]=\boldsymbol{L}\frac{d}{dt}[I] \qquad [3.4.3.3]$$

$$-\frac{\partial}{\partial x}[I]=\boldsymbol{C}\frac{d}{dt}[U] \qquad [3.4.3.4]$$

into uncoupled equation set in mode domain [$\mathbf{U_m}$,$\mathbf{I_m}$]:

$$-\frac{\partial}{\partial x}[U_m]=\boldsymbol{L_m}\frac{d}{dt}[I_m] \qquad [3.4.3.5]$$

$$-\frac{\partial}{\partial x}[I_m]=\boldsymbol{C_m}\frac{d}{dt}[U_m] \qquad [3.4.3.6]$$

Therfore, for a given **L**,**C** matrixes, mode transformations $\mathbf{T_u}$, $\mathbf{T_i}$ should make $\mathbf{L_m}$,$\mathbf{C_m}$ diagonal:

$$\boldsymbol{L_m}=\boldsymbol{T_u^{-1}}\,\boldsymbol{L}\,\boldsymbol{T_i}=\begin{bmatrix} l_{m1} & 0 \\ 0 & l_{m2} \end{bmatrix} \qquad [3.4.3.7]$$

$$\boldsymbol{C_m}=\boldsymbol{T_i^{-1}}\,\boldsymbol{C}\,\boldsymbol{T_u}=\begin{bmatrix} c_{m1} & 0 \\ 0 & c_{m2} \end{bmatrix} \qquad [3.4.3.8]$$

3.5. NF of cascade example explained.

Summary: noise modeling, NF, minimal NF, Touchstone noise data, noise analysis

3.5.1. Noise modeling.

Noise Power Spectrum Density PSD for signal x(t) is defined as [8]:

$$S_{xx}(j\omega)=\frac{1}{T}<|X(j\omega)|^2>=\frac{1}{T}<X(j\omega)X(j\omega)^*> \qquad X(j\omega)=F(x(t)) \tag{3.5.1.1}$$

RMS value of x(t) for signal bandwidth BW = f_2-f_1:

$$E[x(t)^2]=\int_{f_1}^{f_2} S_{xx}(j\omega)\,df \tag{3.5.1.2}$$

Cross PSD between correlated signals x(t), y(t) is defined as:

$$S_{xy}(j\omega)=\frac{1}{T}<X(j\omega)Y(j\omega)^*> \qquad Y(j\omega)=F(y(t)) \tag{3.5.1.3}$$

From **AMP** perspective noise sources are either voltage or current sources. Output signal is also voltage or current and as a signal of a noise type should be defined by PSD. Therfore, noise analysis should determine output PSD.

Output PSD for a network with two correlated sources is defined by [8]:

$$S_{yy}(j\omega)=\begin{bmatrix}T_1 T_2\end{bmatrix}\begin{bmatrix}S_{11} & S_{12}\\ S_{21} & S_{22}\end{bmatrix}\begin{bmatrix}T_1^*\\ T_2^*\end{bmatrix} \tag{3.5.1.4}$$

where T1, T2 are transfer functions from source to output (voltage or current).

From network analysis perspective transfer functions vector [**T]** is to be defined. The second input is cross PSD matrix [**S**]. Parameters in cross PSD matrix are source characterization, and are results of signal analysis.

Noise PSD is a real value function , cross PDS is a complex value function of frequency.

$$S_{yy}(j\omega)=\begin{bmatrix}T_1 T_2\end{bmatrix}\begin{bmatrix}S_{11}T_1^*+S_{12}T_2^*\\ S_{21}T_1^*+S_{22}T_2^*\end{bmatrix}=S_{11}|T_1|^2+S_{12}T_1T_2^*+S_{21}T_1^*T_2+S_{22}|T_2|^2 \tag{3.5.1.5}$$

Since cross PSD are conjugated $S_{21} = (S_{12})^*$ (3.5.1.3),

$$S_{12}T_1T_2^*=(S_{21}T_2T_1^*)^* \tag{3.5.1.6}$$

$$S_{yy}(j\omega)=S_{11}|T_1|^2+S_{22}|T_2|^2+2\Re(S_{12}T_1T_2^*)=S_{11}|T_1|^2+S_{22}|T_2|^2+2\Re(S_{21}T_2T_1^*) \tag{3.5.1.7}$$

When sources are uncorrelated ($S_{12} = S_{21} = 0$) formula for output PSD simplifes to:

$$S_{yy}(j\omega)=S_{11}|T_1|^2+S_{22}|T_2|^2 \tag{3.5.1.7}$$

3.5.2. NF.

Noise figure is a ratio of total noise to noise due to source resistance.

$$F=\frac{N}{N_{Rs}} \quad NF=10\log(F) \qquad [3.5.2.1]$$

Let's assume that device noise is represented as input equivalent **v,i** noise model.

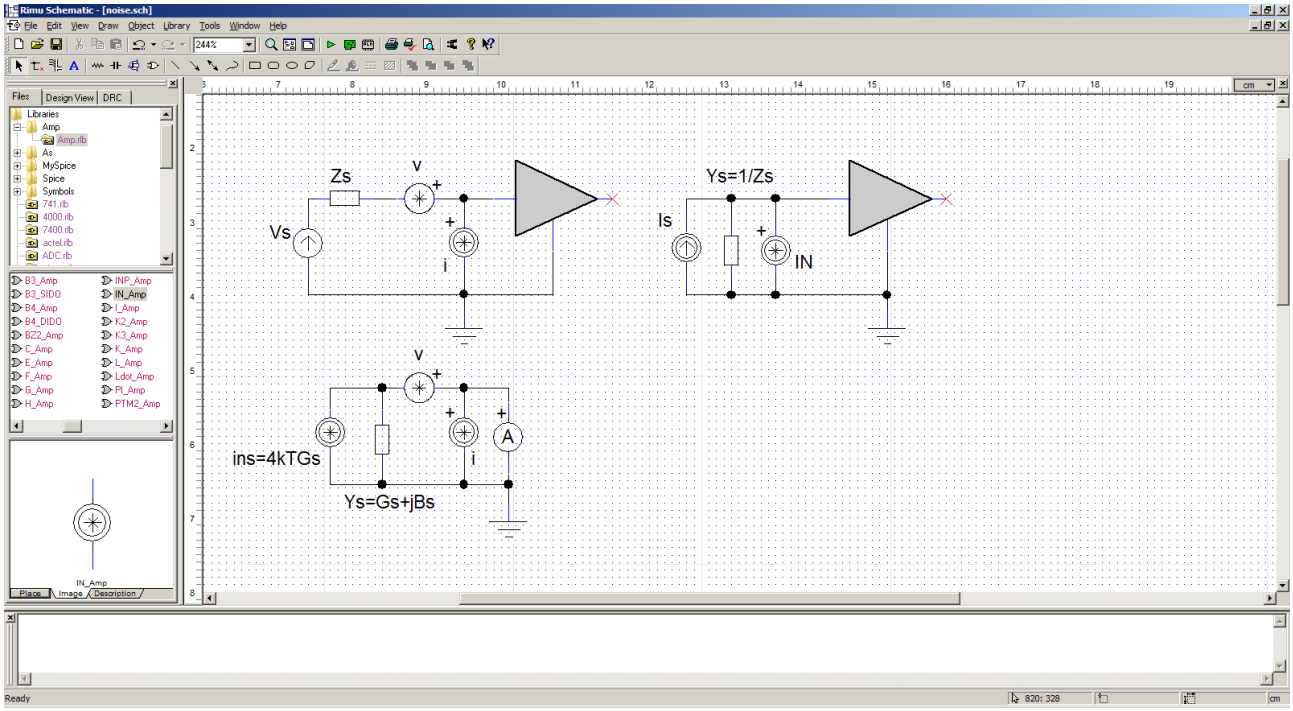


Fig. 3.5.2. Equivalent noise models.

If sources are converted into equivalent Norton $\mathbf{I_N}$ source, then

$$F=\frac{N}{N_{Rs}}=\frac{I_N^2}{i_{ns}^2}=\frac{S_{iiN}\,df}{S_{iis}\,df}=\frac{S_{iiN}}{4\,k\,T\,G_s} \qquad where: G_s=\Re(Y_s);\ Y_s=G_s+j\,B_s \qquad [3.5.2.2]$$

The notable effect of combining equivalent device noise into the Norton source representation is that NF does not depend on source load (i.e. input impedance of active device).

Source noise $\mathbf{i_{ns}}$ and $\mathbf{v_n}$, $\mathbf{i_n}$ are not correlated, therefore PSD for Norton noise $\mathbf{I_N}$ source is:

$$S_{iiN}(j\omega)=\begin{bmatrix}1\ Y_s\ 1\end{bmatrix}\begin{bmatrix}S_{iis} & 0 & 0\\ 0 & S_{vv} & S_{vi}\\ 0 & S_{iv} & S_{ii}\end{bmatrix}\begin{bmatrix}1\\ Y_s^*\\ 1\end{bmatrix} \qquad [3.5.2.3]$$

$$S_{iiN}(j\omega)=S_{iis}+S_{ii}+S_{vv}|Y_s|^2+2\,\Re(S_{iv}Y_s^*)=S_{iis}+S_{ii}+S_{vv}|Y_s|^2+2\,\Re(S_{vi}Y_s) \qquad [3.5.2.4]$$

Since **i** noise source is correlated with **v,** current noise **i** is modeled as sum of independent component $\mathbf{i_n}$ and voltage dependent component $\mathbf{Y_c v_n}$, with complex correlation admittance $\mathbf{Y_c}$.

$$I(j\omega)=I_n(j\omega)+Y_c V(j\omega) \qquad [3.5.2.5]$$

Since, now $\mathbf{i_n}$ component and **v** are ucorrelated, thus :

$$S_{ii}=\frac{1}{T}<I\,I^*>=\frac{1}{T}<(I_n+Y_c*V)(I_n+Y_c*V)^*>=\frac{1}{T}<I_n I_n^*+Y_c Y_c^* V\,V^*>=S_{iin}+|Y_c|^2 S_{vv} \quad [3.5.2.6]$$

$$S_{iv}=\frac{1}{T}<I\,V^*>=\frac{1}{T}<(I_n+Y_c*V)V^*>=\frac{1}{T}Y_c<V^**V>=Y_c S_{vv} \quad [3.5.2.7]$$

$$S_{vi}=\frac{1}{T}<V\,I^*>=\frac{1}{T}<V(I_n+Y_c*V)^*>=\frac{1}{T}Y_c^*<V^**V>=Y_c^* S_{vv} \quad [3.5.2.8]$$

$$S_{iiN}(j\omega)=S_{iis}+S_{iin}+S_{vv}\left(|Y_s|^2+|Y_c|^2+2\,\Re(Y_c Y_s^*)\right) \quad [3.5.2.9]$$

For $\mathbf{Y_s=G_s+j*B_s}$, $\mathbf{Y_c=G_c+j*B_c}$:

$$S_{iiN}(j\omega)=S_{iis}+S_{iin}+S_{vv}\left(G_s^2+B_s^2+G_c^2+B_c^2+2\,G_s G_c+2\,B_s B_c\right) \quad [3.5.2.10]$$

$$S_{iiN}(j\omega)=S_{iis}+S_{iin}+S_{vv}\left((G_s+G_c)^2+(B_s+B_c)^2\right) \quad [3.5.2.11]$$

$$S_{iiN}(j\omega)=S_{iis}+S_{iin}+S_{vv}|Y_c+Y_s|^2 \quad [3.5.2.12]$$

and finally:

$$F=\frac{S_{iiN}}{S_{iis}}=1+\frac{S_{iin}+S_{vv}|Y_c+Y_s|^2}{4\,k\,T\,G_s} \quad [3.5.2.13]$$

3.5.3. Minimal NF.

The F in the form with PSD is rarely seen, other form with narrow band RMS noise is more common:

$$F=\frac{S_{iiN}\,df}{S_{iis}\,df}=1+\frac{i_n^2+v^2|Y_s+Y_c|^2}{4\mathrm{kT}\,G_s\,df} \quad [3.5.3.1]$$

But usually, to make things even more complicated, PSD of $\mathbf{i_n\,v}$ is replaced by equivalent noise resistance and admittance:

voltage noise is related with equivalent noise resistance $\mathbf{R_n}$

$$v_n^2=S_{vv}\,df=4\mathrm{kTR}_n\,df \quad [3.5.3.2]$$

uncorrelated part of current noise with noise admitance $\mathbf{G_n}$

$$i_n^2=S_{iin}\,df=4\mathrm{kTG}_n\,df \quad [3.5.3.3]$$

After substituting $\mathbf{i_n\,v}$ with $\mathbf{R_n,G_n}$ noise, factor can be represented as a function of complex input admittance:

$$F(Y_s)=F(G_s,B_s)=1+\frac{G_n+R_n\left((G_s+G_c)^2+(B_s+B_c)^2\right)}{G_s} \quad [3.5.3.4]$$

Noise factor $\mathbf{F(G_s,B_s)}$ has a minimum $\mathbf{F_{min}(G_{sopt},B_{sopt})}$for optimal source admittance $\mathbf{Y_{sopt}=G_{sopt}+j\,B_{sopt}}$

$$F_{min}=1+\frac{G_n+R_n(G_{sopt}+G_c)^2}{G_{sopt}}\quad G_{sopt}=1+\sqrt{(G_c^2+\frac{G_n}{R_n})};\ B_{sopt}=-B_c \quad [3.5.3.5]$$

3.5.4. Touchstone noise data.

In equations [3.5.3.4], [3.5.3.5] noise sources are represented by 3 parameters; 2 real $\mathbf{G_n}$, $\mathbf{R_n}$, and complex correlation admittance $\mathbf{Y_c}$. They act as a representation of 3 PSD functions: 2 real $\mathbf{S_{vv}}$, $\mathbf{S_{ii}}$, and one complex $\mathbf{S_{vi}} = \mathbf{(S_{iv})^*}$.
From signal analysis perspective PSD functions are natural description of noise source properties, yet not so common in RF domain. Touchstone data are related with formula for minimal noise figure [3.5.3.5].
For each frequency noise source is characterized by:

$\mathbf{NF_{min}}$[dB] – minimum noise figure,
$\mathbf{abs(\Gamma_{opt})}$, $\mathbf{arg(\Gamma_{opt})}$ – magnitude and argument of optimal reflection coefficient needed to achieve $\mathbf{NF_{min}}$,
$\mathbf{R_n/Z_0}$ - normalized noise resistance.

Four parameters Touchstone noise parameters can be converted into $\mathbf{F_{min}}$, $\mathbf{Y_{sopt}} = \mathbf{G_{sopt}} + \mathbf{j}\,\mathbf{B_{sopt}}$, $\mathbf{R_n}$.
Two ($\mathbf{R_n}$, $\mathbf{B_c}$) of four required ($\mathbf{R_n}$, $\mathbf{G_n}$, $\mathbf{Y_c}$, $\mathbf{B_c}$) parameters are directly available:

$$S_{vv} = 4kTR_n \qquad [3.5.4.1]$$

$$B_c = -B_{sopt} \qquad [3.5.4.2]$$

And remaining two ($\mathbf{G_n}$, $\mathbf{G_c}$) require solving nonlinear equation set :

$$\left[\begin{array}{c} F_{min} = 1 + \dfrac{G_n + R_n\left(G_{sopt} + G_c\right)^2}{G_{sopt}} \\ G_{sopt} = 1 + \sqrt{\left(G_c^2 + \dfrac{G_n}{R_n}\right)} \end{array}\right] \qquad [3.5.4.3]$$

for a given $\mathbf{F_{min}}$, $\mathbf{G_{sopt}}$, $\mathbf{R_n}$.

3.5.5. Noise analysis.

Cascade elements have separate s-parameter and noise description. The equivalent noise model of each stage is defined by PSD functions of voltage $\mathbf{S_{vv}}$ and current noise source $\mathbf{S_{ii}}$, and correlation between the two is defined by correlation admittance **Yc**.

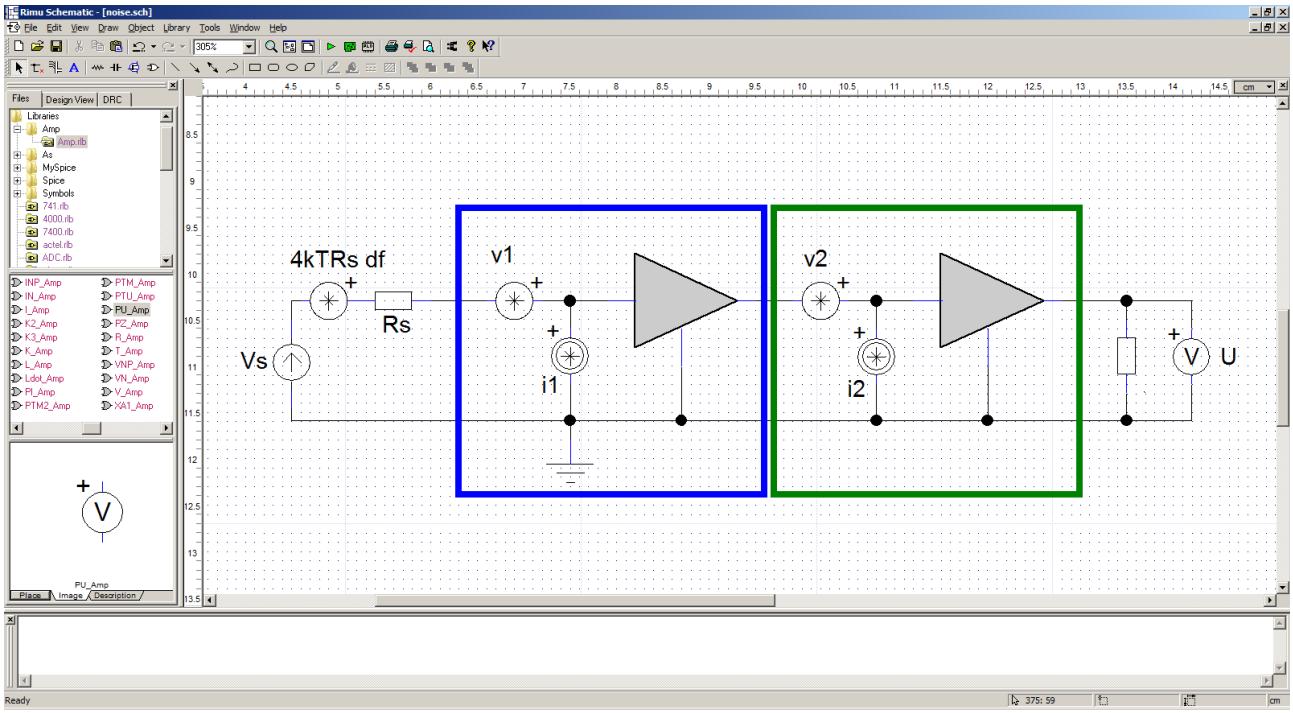


Fig. 3.5.5. Noise model of a cascade.

All passive components are assumed to be noiseless, therefore all noise sources has to defined explicitly as voltage or current sources. Noise is measured at port defined by voltage probe.

There is no correlation between internal and external noise sources. Thus, according to superposition theorem, output PSD $\mathbf{S_{uu}}$ measured by voltage probe, can be represented as a sum of contributions of the source noise and all 'noisy' devices in a network:

$$S_{uu}(j\omega)=S_{uuS}+S_{uu1}+S_{uu2}+... \quad [3.5.5.1]$$

First contribution is due to the source resistance $\mathbf{S_{vvS}=4kTR_s}$:

$$S_{uuS}(j\omega)=\left|(K_s(j\omega))\right|^2 4\mathrm{kTR}_s \quad [3.5.5.2]$$

$\mathbf{K_S}$ is system voltage gain input to output.

Contribution of each active device N=(1,2,3...) is defined as:

$$S_{uuN}(j\omega)=\begin{bmatrix}K_N M_N\end{bmatrix}\begin{bmatrix}S_{vvN} & S_{viN}\\ S_{ivN} & S_{iiN}\end{bmatrix}\begin{bmatrix}K_N^*\\ M_N^*\end{bmatrix} \quad [3.5.5.3]$$

where, transfer functions $\mathbf{K_N, M_N}$ are calculated by **AMP** and noise source parameters ($\mathbf{S_{vvN}, S_{iiN}, Y_{cN}}$) are derived from Touchstone data:

$$S_{uuN}(j\omega)=S_{vvN}\left|K_N\right|^2+S_{iiN}\left|M_N\right|^2+2\,\mathrm{Re}\left(S_{ivN}K_N^*M_N\right) \quad [3.5.5.4]$$

$$S_{iiN}=S_{iinN}+\left|Y_{cN}\right|^2 S_{vvN} \quad [3.5.5.5]$$

$$S_{ivN}=Y_{cN}S_{vvN} \quad [3.5.5.6]$$

$$S_{uuN}(j\omega)=S_{iinN}\left|M_N\right|^2+S_{vvN}\left[\left|K_N\right|^2+\left|M_N\right|^2\left|Y_{cN}\right|^2+2\,\mathrm{Re}\left(Y_{cN}K_N^*M_N\right)\right] \quad [3.5.5.7]$$

Noise factor is calculated as:

$$NF=10\log\left(\frac{S_{uu}(j\omega)}{S_{uuS}(j\omega)}\right)=10\log\left(1+\frac{S_{uu1}+S_{uu2}+...}{S_{uuS}}\right) \quad [3.5.5.8]$$

3.6. RF passive filter optimization example explained.

Summary: folder structure, sensitivities.

3.6.1. Folder structure.

Fig.3.6.1.1. Folder structure of optimization example.

To run optimization use scilab scripts in **sce** folder. Other folders contain AMP source files with input/output files. Design entry and results are in folders:

```
filter_HQ                    - high Q filter desing
filter_LQ                    - low Q filter design
```

Pattern-search optimization are in folders:

```
flp_hq_ps_max_sens           - pattern search optimization MOD = high Q; err = PS_MAX , OPT_SENS
flp_hq_ps_max_std            - pattern search optimization MOD = high Q; err = PS_MAX, OPT_STD
flp_hq_ps_sum_sens           - pattern search optimization MOD = high Q; err = PS_SUM, OPT_SENS
flp_hq_ps_sum_std            - pattern search optimization MOD = high Q; err =PS_SUM, OPT_STD
flp_hq_pts_max_sens          - pattern search optimization MOD = high Q; err =PTS_MAX, OPT_SENS
flp_hq_pts_max_std           - pattern search optimization MOD = high Q; err =PTS_MAX, OPT_STD
flp_hq_pts_sum_sens          - pattern search optimization MOD = high Q; err =PTS_SUM, OPT_SENS
flp_hq_pts_sum_std           - pattern search optimization MOD = high Q; err =PTS_SUM, OPT_STD
flp_lq_ps_max_std            - pattern search optimization MOD = low Q; err =PS_MAX, OPT_STD
```

Preferred numbers optimization are in folders:

```
pn_ps_max_le12_ce12          - preferred numbers optimization L:E12, C:E12, err = PS_MAX
pn_ps_sum_le12_ce12          - preferred numbers optimization L:E12, C:E12, err = PS_SUM
pn_pts_max_le12_ce12         - preferred numbers optimization L:E12, C:E12, err = PTS_MAX
pn_pts_sum_le12_ce12         - preferred numbers optimization L:E12, C:E12, err = PTS_SUM
```

Sensitivity analysis are in folders:

```
sens_mc                            - Monte-Carlo sensitivity analysis
sens_std                     - differential sensitivity analysis
```

Optimization and sensitivity analysis scripts are located in **sce** sub-folders. Each optimization sub-folder includes log files.

Fig.3.6.1.2. Folder structure of scilab sub-folders.

Scilab common functions are in **examples_sci** folder.

Fig.3.6.1.3. Common scilab folder.

3.6.2. Sensitivities.

AMP calculates sensitivity $\mathbf{S_X^T}$ of transfer function **T** due to a change of parameter **X** as:

$$S_X^T = \frac{X}{T}\frac{\partial T}{\partial X} \qquad [3.6.2.1]$$

Therefore, relative change of transfer function **δT** can be expressed as relative change of parameter **δX** times sensitivity $\mathbf{S_X^T}$:

$$\delta T = \frac{\Delta T}{T} = \frac{1}{T}\left(\frac{\partial T}{\partial X}\Delta X\right) = \left(\frac{X}{T}\frac{\partial T}{\partial X}\right)\frac{\Delta X}{X} = S_X^T \delta X \qquad [3.6.2.2]$$

When transfer function **T** is a complex function of real parameter **X** then:

$$\frac{\partial T}{\partial X} = \frac{\partial\left(|T|\exp(j\varphi)\right)}{\partial X} = \frac{\partial(|T|)}{\partial X}\exp(j\varphi) + j|T|\exp(j\varphi)\frac{\partial(\varphi)}{\partial X} \qquad [3.6.2.3]$$

$$\frac{1}{T}\frac{\partial T}{\partial X} = \frac{1}{|T|}\frac{\partial(|T|)}{\partial X} + j\frac{\partial(\varphi)}{\partial X} \qquad [3.6.2.4]$$

Therefore relative change of transfer function module |**T**| can be expressed as relative change of parameter **δX** times real part of the sensitivity $\mathbf{S_X^T}$:

$$\delta|T| = \frac{\Delta|T|}{|T|} = \Re(S_X^T)\delta X \qquad [3.6.2.5]$$

and absolute change of transfer function phase Δφ as relative change of parameter **δX** times imaginary part of the sensitivity $\mathbf{S_X^T}$:

$$\delta\varphi = \Im(S_X^T)\delta X \qquad [3.6.2.6]$$

In exploratory searching, rather than directly calculate new value of transfer function, sensitivities can be used to estimate transfer function change:

$$T(X+\Delta X) = T(X(1+\delta X)) \approx T + \Delta T = T + \frac{\partial T}{\partial X}\Delta X = T + T S_X^T \delta X = T(1 + S_X^T \delta X) \qquad [3.6.2.7]$$

AMP calculates sensitivities of voltage gain $\mathbf{K_u}$. Yet, for a filter in a test circuit:

$$s_{21} = 2K_u \qquad [3.6.2.8]$$

thus:

$$S_X^{s_{21}} = S_X^{K_u} \qquad [3.6.2.9]$$

Exploratory search can be calculated according to equation:

$$|s_{21}(X(1+\delta X))| \approx |s_{21}| + \Delta|s_{21}| = |s_{21}| + |s_{21}|\Re(S_X^{s_{21}})\delta X = |s_{21}|(1 + \Re(S_X^{K_u})\delta X) \qquad [3.6.2.10]$$

where **X** is element of a part value vector **[L1,L2,L3, C1,C2,C3,C4]**.

4. References

[1] – http://www.hutson.co.nz/ - Rimu schematics.

[2] – http://www.spiceopus.si/ - OPUS Spice – AMP supports data in Nutmeg format

[3] - http://www.scicoslab.org/ – used as scripting language (IMHO Scicoslab is older and better fork of Scilab)

[4] -http://www.kikowski.com/web/doc/Rimu/rimu_setup.pdf – Rimu setup for AMP, AS,Spice

[5] R. Gilmore, L. Besser - Practical RF Circuit Design for Modern Wireless Systems: Active Circuits and Systems – ISBN: 1580535224

[6] http://www.vishay.com/docs/60107/freqresp.pdf - Frequency Response of Thin Film Chip Resistors.

[7] C.R. Paul – Analysis of Multiconductor Transmission Lines - ISBN: 978-0-470-13154-1

[8] J. S. Bendat, A. G. Piersol - Random Data: Analysis and Measurement Procedures ISBN: 978-0-470-24877-5

[9] J. A. Dobrowolski - Microwave Network Design Using the Scattering Matrix ISBN: 9781608071296