Introduction

This is a collection of hints I refer to when I want to remember how I have done some task in NGSPICE in the past, but don't remember which simulation it was in. I hope that you, the reader, find some of this helpful. **This document is FREE to redistribute not for profit.**

Orcad Netlist

To ignore a component in the net listing, add a property NETLIST_IGNORE and set it to TRUE.

Pulse

Name	Parameter	Default Value	Units
V1	Initial value	-	V, A
V2	Pulsed value	-	V, A
TD	Delay time	0.0	sec
TR	Rise time	TSTEP	sec
TF	Fall time	TSTEP	sec
PW	Pulse width	TSTOP	sec
PER	Period	TSTOP	sec
PHASE	Phase	0.0	degrees

Example:

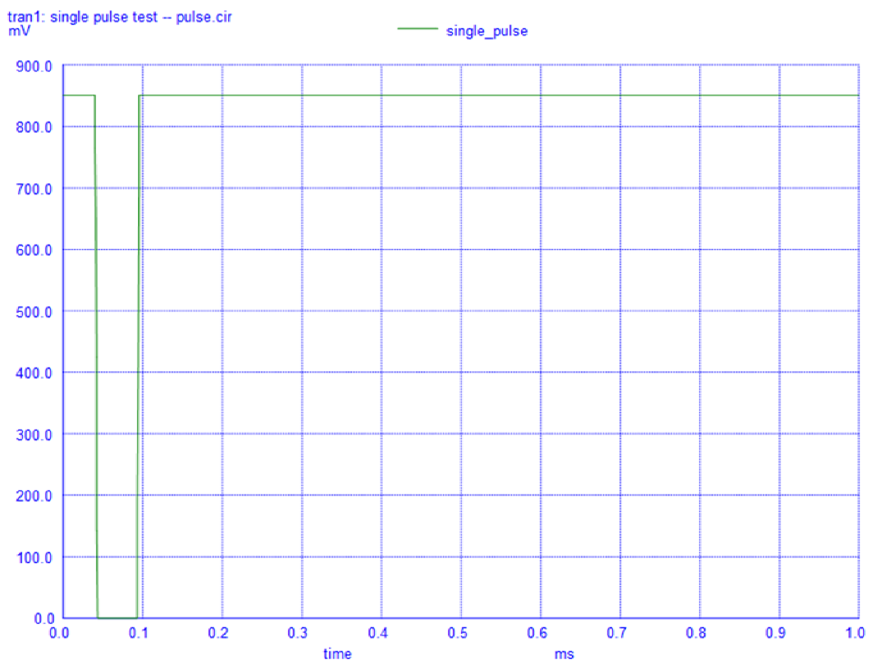
```
V1 1 0 PULSE (1 0.5 0 1u 1u 1m 2m)
```

A single pulse, without phase offset, is described by the following table:

Time	Value
0	V1
TD	V1
TD+TR	V2
TD+TR+PW	V2
TD+TR+PW+TF	V1
TSTOP	V1

and a practical example of it is here:

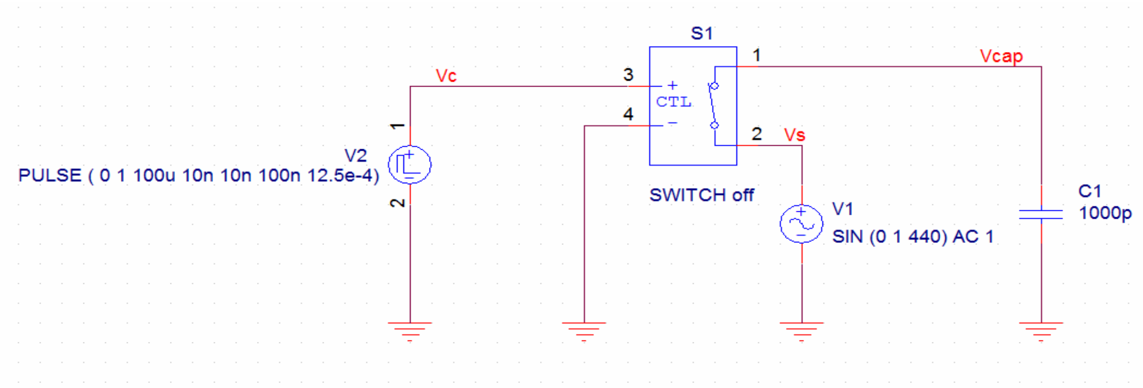
```
V1 single_pulse 0 pulse (0.85 0 40u 3u 3u 50u 100m)
```



The .control options to make this plot are (svg format):

```
set hcopydevtype = svg
set svg_stropts = ( yellow Arial Arial )

set color0 = white
set color1 = blue
set color2 = green

hardcopy pulse.svg single_pulse
```

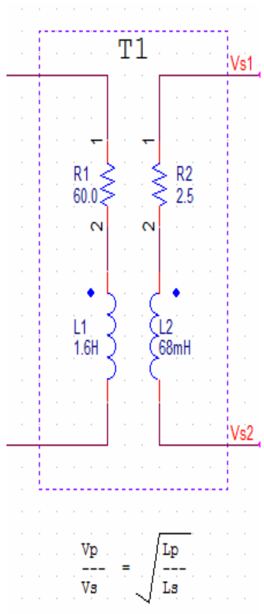
Switch



Netlist:

```
S1 VCAP VS VC 0 SWITCH off
.model SWITCH SW Vt=0.5 Ron=0.001 Roff=1G
```

Coupled Inductor



.option RSHUNT=1Meg --- helps with the grounds

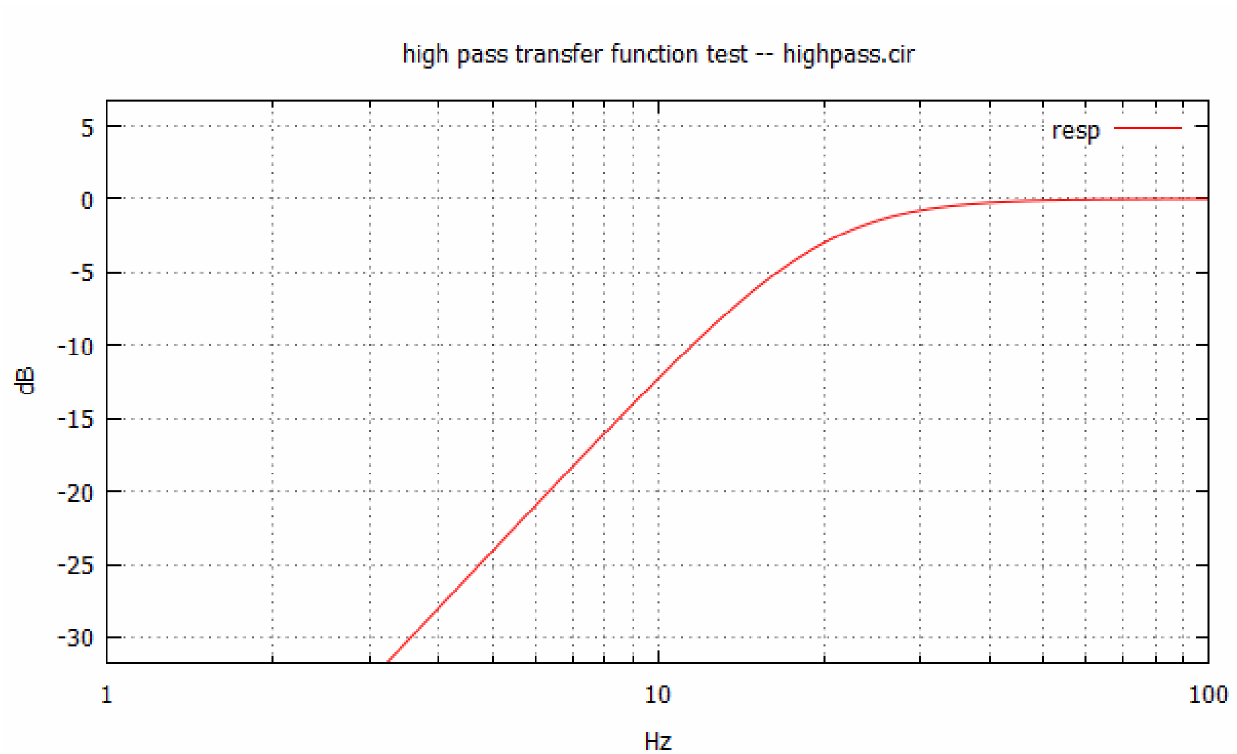
K12 L1 L2 0.99 --- coupling coefficient

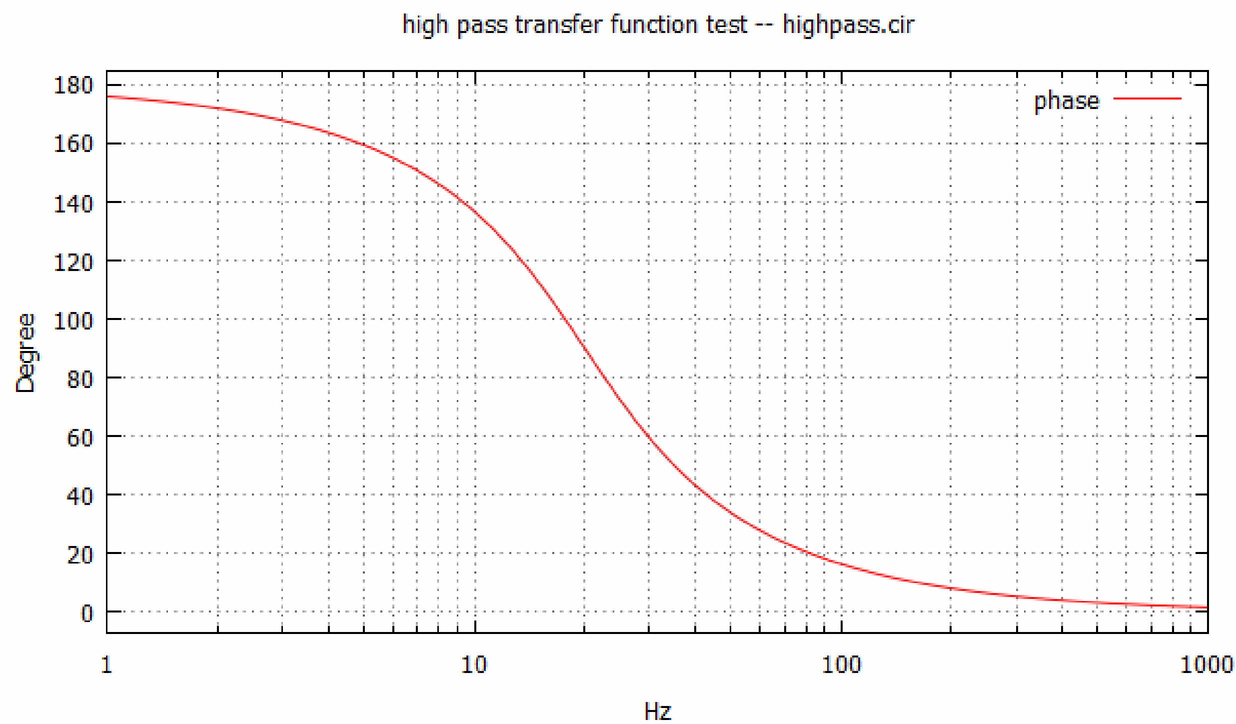
XSPICE

Transfer function example of high pass filter parameterized.

```
.param pi = 4.0 * atan (1.0)
.param alpha = sqrt(2.0)

.param frequency = 20.0
.param w0  = 2.0 * pi * frequency


.model filter s_xfer (gain=1
+ num_coeff=[1.0 0.0 0.0]
+ den_coeff=[1.0 {w0*alpha} w0*w0]
+ int_ic=[0 0] denormalized_freq = 1.0)

a1 1 2 filter
```

and the response:





Scripting for the plots:

```
.control
   destroy all

   ac dec 1000 1 1000
   let resp = db (v(2)/ v(1))
   settype decibel resp

   plot resp xlimit 6 100 ylimit -30 0
   meas AC three_db_point WHEN resp=(-3.0)
   meas AC six_db_point WHEN resp=(-6.0)
   meas AC cutoff FIND resp AT=20.0

*
* Do the phase response.
*
   let phase = 180.0/PI * cph (v(2) / v(1))
   settype phase phase
   plot phase title "High Pass Phase Response"
   meas AC ninety_degrees WHEN phase = 90
.endc
```

The NGSPICE output:

```
ngspice 27 -> highpass.cir

Circuit: high pass transfer function test -- highpass.cir

Reducing trtol to 1 for xspice 'A' devices
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000

Warning: v1: has no value, DC 0 assumed

No. of Data Rows : 3001
three_db_point      =  2.002380e+01
six_db_point        =  1.522080e+01
cutoff              =  -3.010302e+00
ninety_degrees      =  2.000000e+01
```

Measurements

```
*
* Measure the period
*
meas TRAN period TRIG V(5) VAL=200m RISE=5 TARG V(5) VAL=200m RISE=6
let frequency = 1.0/period
print frequency
*
* Measure the duty cycle
*
meas TRAN period TRIG V(5) VAL=200m RISE=5 TARG V(5) VAL=200m RISE=6
meas TRAN pulse_width TRIG V(5) VAL=200M RISE=5 TARG V(5) VAL=200m FALL=5
let duty_cycle = 100.0* pulse_width/period
print duty_cycle
*
* Measure -3dB point, 0 dB max gain
*
meas AC three_db_point WHEN resp=(-3.0)
```

If the gain is greater than zero, the only way I have seen how to do this is to use these lines:

```
let measurement_point = vecmax (resp) - 3.0
meas AC lower_3dB WHEN resp = measurement_point RISE=1
meas AC upper_3dB WHEN resp = measurement_point FALL=1
```

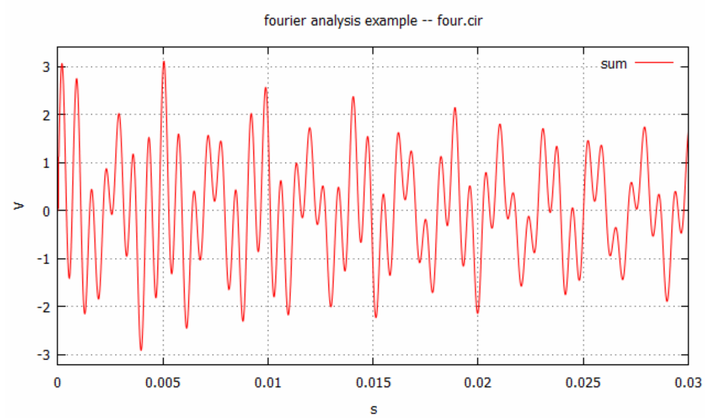
Put FALL=1 if there is more than one falling edge. (I saw this once.)

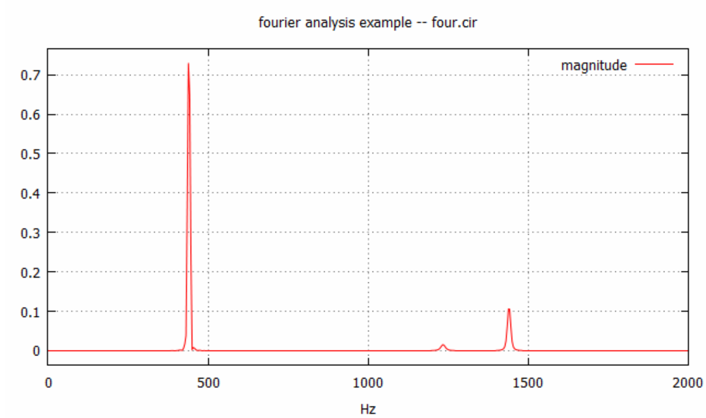
```
*
* Measure rise time
*
meas TRAN range PP v(5)
let ten=0.1*range
let ninety=0.9*range
meas TRAN rise_time TRIG v(5) VAL=ten RISE=5 TARG v(5) VAL=ninety RISE=5
*
* Another rise/fall time measurement where PP voltage is trouble.
meas TRAN rise_time TRIG vout VAL=-100m RISE=3 TARG vout VAL= 100n RISE=3
meas TRAN fall_time TRIG vout VAL= 100m FALL=3 TARG vout VAL=-100n FALL=3
*
* Measure Q of a bandpass filter
*
   let pk = vecmax (resp)
   meas AC peak_frequency WHEN resp = pk
   meas AC lower_3dB WHEN resp = -3.0 RISE=1
   meas AC upper_3dB WHEN resp = -3.0 FALL=1
   let bw = upper_3dB-lower_3dB
   let q = peak_frequency/(upper_3dB-lower_3dB)
   print q
*
* Find the response at a specific frequency
*
meas AC cutoff FIND resp AT=20.0
```

```
*
* Find the time a peak occurs of a waveform after a transient simulation
*
meas TRAN  peak_time MAX V(4)
*
* Find when the output is a given value as a function of X
*
meas DC temp_240 WHEN vout=240m
*
*Measure the time between two edges
*
meas TRAN tdiff  TRIG vout1   RISE=3 VAL=1.0  TARG vout2   RISE=3 VAL=1.0
*
* Measure the RMS value of a waveform
*
meas TRAN rms_voltage RMS vout
```

FFT

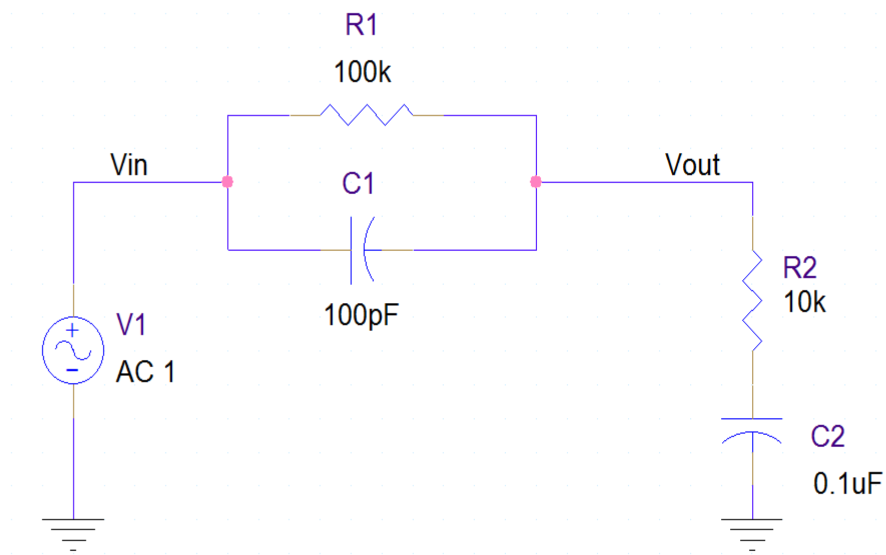
```
Fourier Analysis Example --
four.cir

V1 1 0 SIN (0 1 440 0)
V2 2 0 SIN (0 1.7 1440 0 30)
V3 3 0 SIN (0 0.9 1234 0 55)

a2 [1 2 3] 4 sum1
.model sum1 summer
+  (in_gain=[1.0 1.0 1.0]
+    out_gain = 1.0)

.control
destroy all

tran 1e-6 200e-3
let sum = v(4)
gnuplot tran  sum xlimit 0 30m

linearize sum
fft sum
let magnitude = mag(sum)
gnuplot mag magnitude xlimit 0
2k

.endc

.end
```



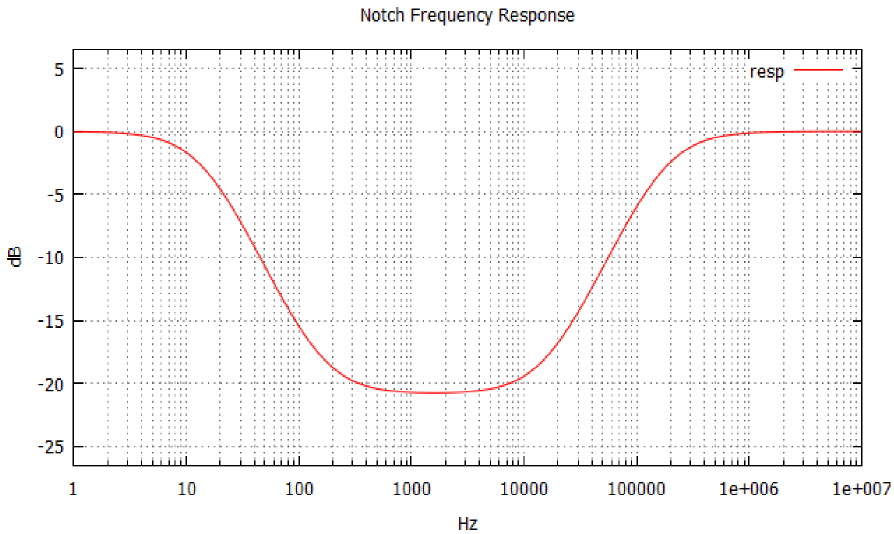


Pole-Zero Analysis and Transfer Function

Here is a simple circuit to analyze:



This is lag-lead compensator from D'Azzo and Houpis, Linear Feedback and Conrol Systems, Synthesis and Design, McGraw Hill, New York.



The frequency response of this network is given below:

The pole-zero analysis yields:

```
pole(1) = -1.10091e+06,0.000000e+00
pole(2) = -9.08340e+01,0.000000e+00
zero(1) = -1.00000e+05,0.000000e+00
zero(2) = -1.00000e+03,0.000000e+00
```

And the transfer function report is:

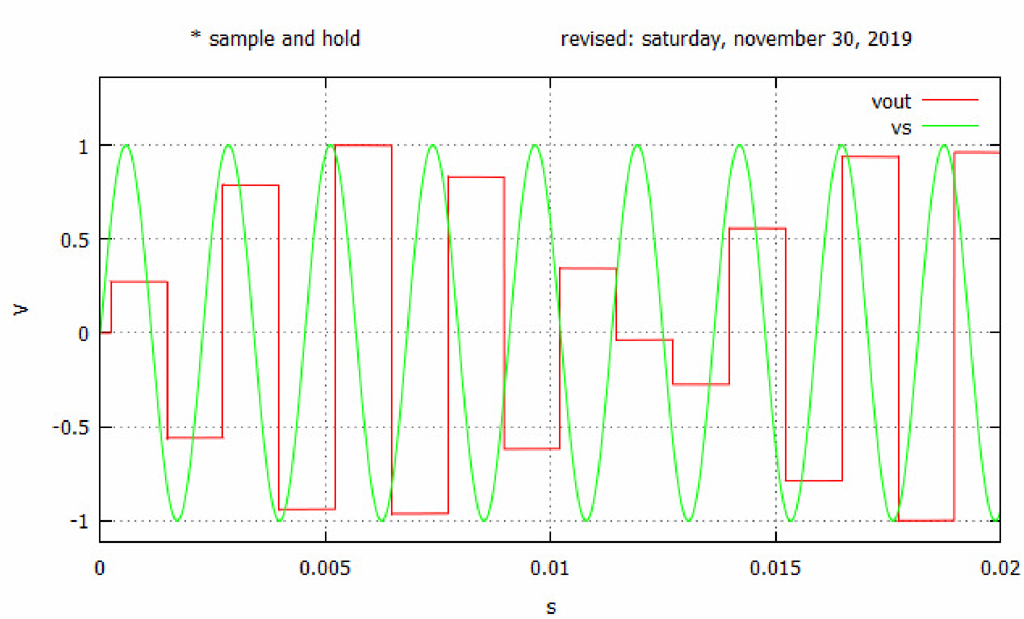
```
transfer_function = 1.000000e+00
output_impedance_at_v(vout,0) = 1.000000e+05
v1#input_impedance = 1.000000e+20
```

And the commands for the control section to produce this output are:

```
pz Vin 0 Vout 0 vol pz
print all
tf v(vout, 0) V1                ; Notice V1 is a source, not a node!
print all
```

Fourier

See the switch circuit above.




```
ngspice 66 -> fourier 440 vout
Fourier analysis for vout:
  No. Harmonics: 10, THD: 44.2091 %, Gridsize: 200, Interpolation Degree: 1

Harmonic Frequency   Magnitude   Phase       Norm. Mag   Norm. Phase
-------- ---------   ---------   -----       ---------   -----------
 0        0          -0.11667    0           0           0
 1        440        1.23285     171.923     1           0
 2        880        0.192827    -106.2      0.156408    -278.12
 3        1320       0.370736    155.709     0.300716    -16.214
 4        1760       0.18348     -122.4      0.148826    -294.32
 5        2200       0.176647    139.508     0.143284    -32.415
 6        2640       0.168498    -138.6      0.136674    -310.52
 7        3080       0.0810902   123.311     0.0657748   -48.613
 8        3520       0.148732    -154.8      0.120641    -326.72
 9        3960       0.0217612   107.127     0.0176512   -64.796
```

Transient Simulation Trick

To alter the resistance during transient simulation, you may use a resistor like:

```
Rs 1 0 R = 'TIME > 3m ? 2k : 1k'
```

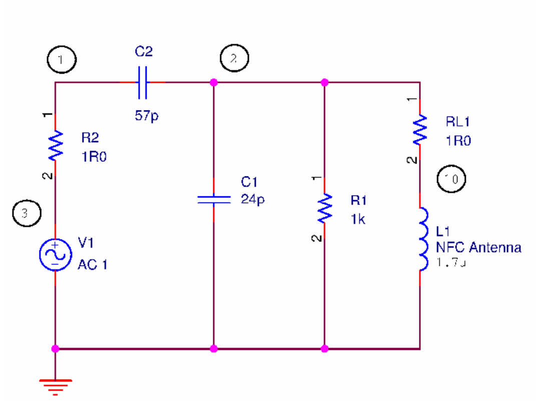
Alter a sinusoidal source

```
alter @v1[sin] [ 0 $&newamp $&newfreq ]
```

[Alter Sinusoidal Source Reference]

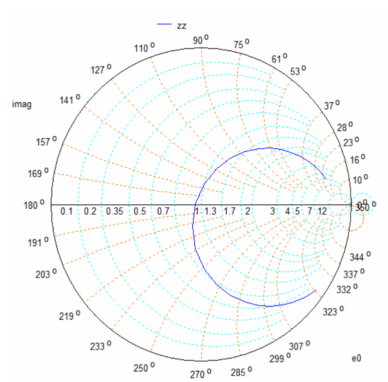
Smith Chart

The following circuit is analyzed:



The netlist:

```
.TITLE  NFC Matching Network -- c.cir

R1   1 0 1000
L1  10 0 1.7u
RL1 10 1 1

C1 1 0 24p
C2 1 2 57p
R2 2 3 1

V1 3 0 sin (0, 1, 13.56E6) AC 1

.control

set color1 = black      ; set up the display colors
set color0 = white

ac dec 100 10E6 20E6
let z1=-v(3)/v1#branch  ; calculated the driving point impedance
settype impedance z1
plot z1

let zz=z1/50            ; normalize for the chart
settype impedance zz

plot zz smith

.endc
.end
```



The plot shows the impedance as seen by the input generator as a function of frequency. It can be seen at a frequency the impedance is 50 ohms, normalized on the chart.

Here are regions of the Smith Chart to give you a hint on how best to proceed:

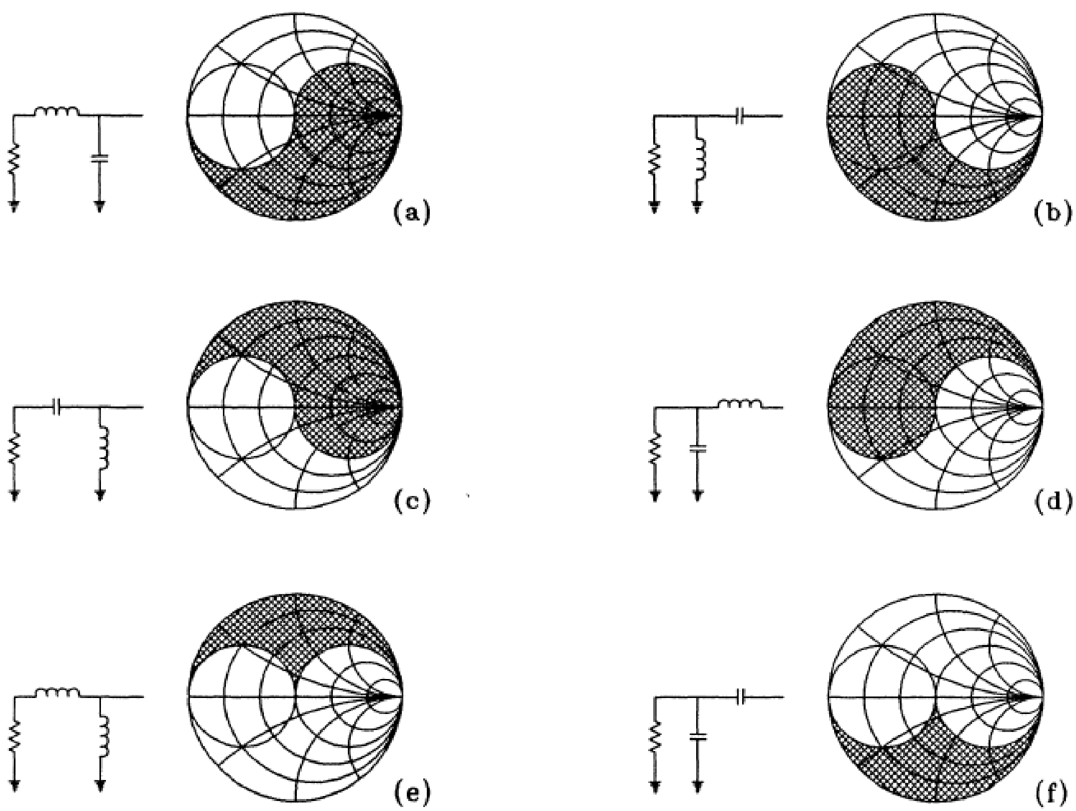


Figure 3.6 Matching regions for two-element networks.

From Medley, <u>Microwave and RF Circuits: Analysis, Synthesis and Design,</u> Artech House, Boston, London. Used with permission.

Gummel Plots

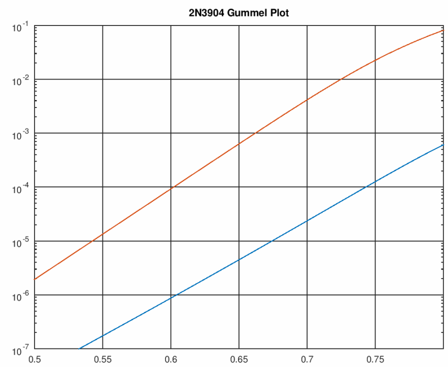
```
.TITLE NPN Gummel plot -- gummel.cir
*
* Creates Gummel data set from BJT model
*
.include "c:\SpiceModels\BJT\2N2369A.lib"
.include "c:\SpiceModels\BJT\bjt.lib"

V1 2 0 0
V2 3 0 12

Q1 3 2 0 2N3904

.control
set wr_singlescale

dc v1 0.35 1 0.01
let ic = -v2#branch
let ib = -v1#branch
let vbe = v(2)

wrdata gummel.txt  vbe ic ib

.endc
.end
```

Data output sample:

```
 6.90000000e-01  6.90000000e-01  2.86630295e-03  1.68741671e-05
 7.00000000e-01  7.00000000e-01  4.14159168e-03  2.35835543e-05
 7.10000000e-01  7.10000000e-01  5.94164201e-03  3.29793695e-05
 7.20000000e-01  7.20000000e-01  8.44540163e-03  4.61284749e-05
 7.30000000e-01  7.30000000e-01  1.18662520e-02  6.45018413e-05
 7.40000000e-01  7.40000000e-01  1.64441373e-02  9.01048612e-05
 7.50000000e-01  7.50000000e-01  2.24319010e-02  1.25627338e-04
 7.60000000e-01  7.60000000e-01  3.00775557e-02  1.74594907e-04
 7.70000000e-01  7.70000000e-01  3.96042577e-02  2.41483436e-04
```

Octave reading the table:

```
m=dlmread ('gummel.txt');

ic  = m(:, 3);
ib  = m(:, 4);
vbe = m(:, 2);
```

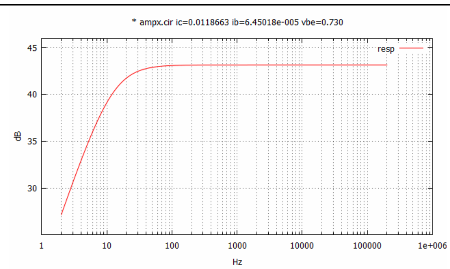


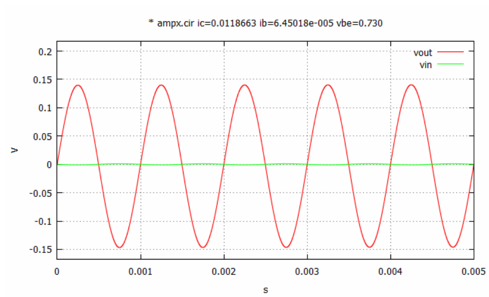
Results from various devices:

```
BC337   Maximum gain: 53.2 at Ic=0.00118846, Ib=2.58526e-006,
Vbe=0.600
2N2369A Maximum gain: 37.6 at Ic=0.859622,   Ib=0.0112886,
Vbe=0.980
2N3904  Maximum gain: 45.3 at Ic=0.0118663,  Ib=6.45018e-005,
Vbe=0.730
2N2222A Maximum gain: 46.2 at Ic=0.026325,   Ib=0.000129467,
Vbe=0.730
2N5089  Maximum gain: 56.4 at Ic=0.00108467, Ib=1.64301e-006,
Vbe=0.620
MPSA18  Maximum gain: 63.3 at Ic=0.0258665,  Ib=1.76724e-005,
Vbe=0.710
```

An Octave script processed the data and generated a common emitter netlist.

```
* ampx.cir Ic=0.0118663 Ib=6.45018e-005
Vbe=0.730
V12 12 0 9V
R1  12 2 10000
R2   2 0 3900
Rc  12 5 470
Re   3 0 160
V3   5 4 0V
V1   0 1 SIN  (0 0.001 1000)  AC 1
Q1   4 2 3 2N3904
                C1   1 2 0.00027
C2   3 0 0.0047
C3   4 6 8.2e-005
RL   6 0 10k
.include "c:\SpiceModels\BJT\bjt.lib"
.control
tran 1e-7 5e-3
let Vin = V(1)
let Vout = V(6)
plot Vin Vout
ac dec 1000 2 200000
let resp = db (v(6) / v(1))
settype decibel resp
plot resp
.endc
.end
```





Parameterized Netlist for Device Selection

Netlist	Analysis

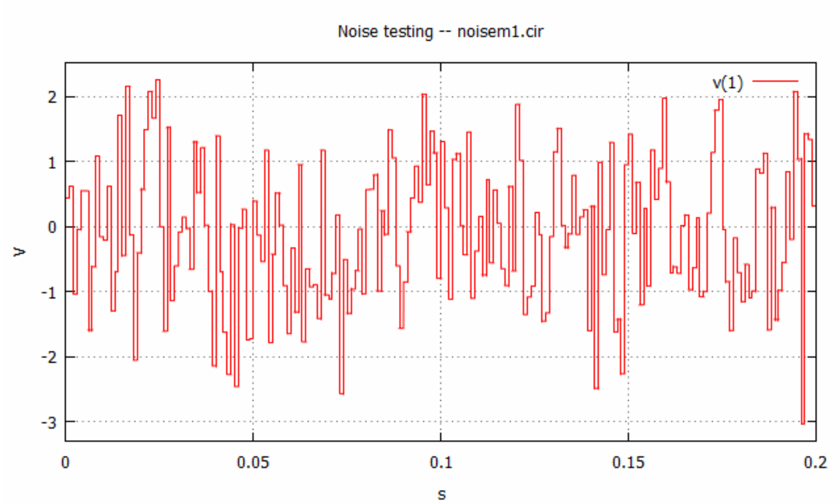
Parmaterized netlist trial -- p.cir
 	
* Device list:
*------------ 	
* 1: 2n3904
* 2: 2n3906
* 3: MPS18
*
.param device=3 ; device selector
 	
V1 2 0 0.7
V2 5 0 5
R1 5 1 10k
 	
.if (device = 1)
   Q1 1 2 0 2N3904
.elseif (device = 2)
   Q1 1 2 0 2N3906
.else
   Q1 1 2 0 MPSA18
.endif
 	
.include c:\SpiceModels\BJT\bjt.lib
 	
.control
 	
op
print all
.endc
 	
.end
	ngspice 1 -> p.cir
	
	Circuit: parmaterized netlist trial -- p.cir
	
	Doing analysis at TEMP = 27.000000 and TNOM = 27.000000
	
	No. of Data Rows : 1
	v(1) = 6.329698e-03
	v(2) = 7.000000e-01
	v(5) = 5.000000e+00
	v1#branch = -1.48829e-03
	v2#branch = -4.99367e-04

Test Selector Mechanism

```
* This is the test selector.  Given tests may be turned on or off.
*
set transient = 1
set frequency_response = 1
set input_impedance = 0
set operating_point = 01
set reverse_voltage = 0
set power_dissipation = 1
set power_gain = 0
set capacitor_voltage = 0
set pulse_test = 1


if ($operating_point)
echo operating point
op
print all
end
```

Random voltage waweform

```
.TITLE Noise testing -- noisem1.cir

VR1 1 0 dc 0 trrandom (2 1e-3 0 1)  ; Gaussian
v2  4 1 2.5

.control
tran 1e-6 200m
plot v(1)
.endc

.end
```



The peak time is found measured as above:

```
meas TRAN peak_time MAX V(1)
peak_time            =  2.260727e+000 at=  2.500000e-002
```

Using a File Source to Drive an Analog Design

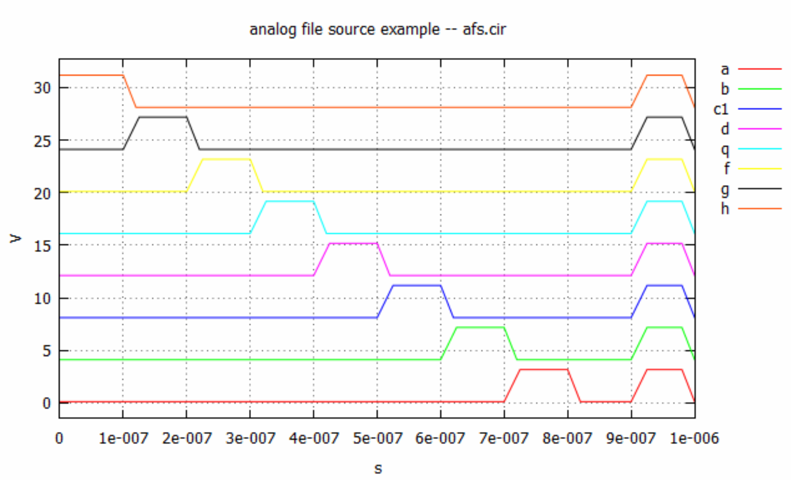
This reads a digital waveform file and converts the digital signal via the bridge to analog. This is just a different way to generate an analog source rather than using the XSPICE filesource reading voltages.

```
.subckt filesource 1 2 3 4 5 6 7 8

A1 [11 12 13 14 15 16 17 18] input_vector

abridge1 [11 12 13 14 15 16 17 18] [1 2 3 4 5 6 7 8] dac1

.model input_vector d_source (input_file = "simple.txt")

.model dac1 dac_bridge(out_low = 0.12 out_high = 3.18 out_undef = 2.2
+ input_load = 5.0e-12 t_rise = 50e-9
+ t_fall = 20e-9)

.ends
```

Here is the input file:

```
*        a  b  c1  d  q  f  g  h
      0 0s 0s 0s 0s 0s 0s 0s 1s
 100e-9 0s 0s 0s 0s 0s 0s 1s 0s
 200e-9 0s 0s 0s 0s 0s 1s 0s 0s
 300e-9 0s 0s 0s 0s 1s 0s 0s 0s
 400e-9 0s 0s 0s 1s 0s 0s 0s 0s
 500e-9 0s 0s 1s 0s 0s 0s 0s 0s
 600e-9 0s 1s 0s 0s 0s 0s 0s 0s
 700e-9 1s 0s 0s 0s 0s 0s 0s 0s
 800e-9 0s 0s 0s 0s 0s 0s 0s 0s
 900e-9 1s 1s 1s 1s 1s 1s 1s 1s
 980e-9 0s 0s 0s 0s 0s 0s 0s 0s
```

Here are the generated waveforms:



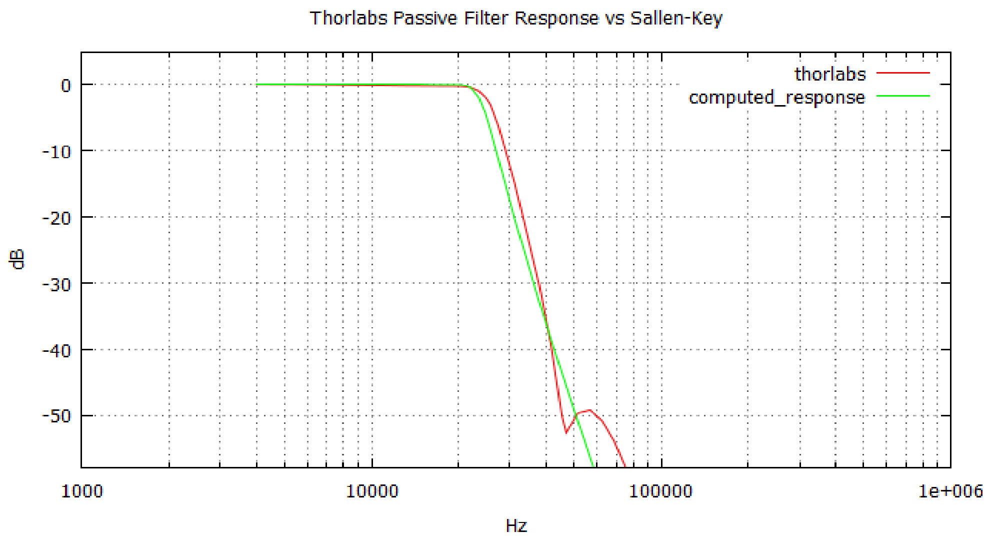
Vector Composition Example and a Reference Design

This demonstrates how to compare a reference design to your own design.

```
.TITLE Thorlabs Passive Filter Response vs Sallen-Key

V1 1 0 ac 1
V5 5 0 5
V6 6 0 -5

X1 1 10 5 6 filterlab3

.control
destroy all
*
* The reference design is here: https:;www.thorlabs.com/thorproduct.cfm?partnumber=EF122#ad-
image-0
*
*
* This is the reference desgin frequency response, in kHz
*
compose frequency_list values 4.00 5.42 7.36 8.91 10.79 13.18 14.76 16.24 18.03 19.84 20.01 20.19
21.27 22.02 23.41 24.66
+ 25.76 27.38 31.19 37.78 38.11 41.57 41.94 45.35 46.96 51.68 56.87 62.58 68.87 75.79 83.40 91.78
101.00 111.15 122.32 128.87

let frequency_list = frequency_list * 1000.0 ; scale the response to Hz
settype frequency frequency_list             ; correct the units
*
* This is a list of frequencies where the reference data was taken.
*
compose reference_list values (0.00) (-0.03) (-0.09) (-0.14) (-0.18) (-0.23) (-0.24) (-0.25) (-
0.26) (-0.28)(-0.28) (-0.29) (-0.37) (-0.50) (-0.99) (-1.91) (-3.12 )(-6.25) (-14.87 )
+ (-29.98 )(-30.77 )(-39.25) (-40.14 )(-50.03) (-52.57) (-49.58) (-49.13) (-50.85) (-53.87) (-
58.15 )(-66.20) (-69.13) (-64.11) (-60.65) (-58.90 )(-58.29)
settype decibel reference_list               ; correct the units.
*
* Analyze the design.  Do this every 1 Hz to line up with the reference design data.
*
ac lin 150000 1 150000
let resp = db (v(10)/v(1))
settype decibel resp
*
* Create working vectors for the error computation and sampling the results of the analysis
*
let  error = vector (length(frequency_list))
let sampled_resp = vector (length(frequency_list))
settype decibel sampled_resp


let index=0
*
* Loop through the reference design and compare the response at each
*   frequency of the reference design to the Sallen-Key.
*
while index < length (frequency_list-1)
   let frequ = frequency_list[index]      ; get a frequency from the list
   let result = resp [frequ]              ; get a computed response at that frequency
   let sampled_resp[index] = result       ; save it off

   let ref_value = reference_list[index]  ; get the reference value

   let err =  ref_value - result          ; compute the error
   settype decibel err                    ; correct the units
   let error[index] = err                 ; save the error in the list
   let index = index + 1
end

settype decibel error
*
* Assign the vectors to more meaningful names for the plots.
*
   let Sallen_Key = resp
   let Thorlabs = reference_list
   let computed_response = sampled_resp
*
```

```
* Plot the results
*
plot ylimit -55 2 xlog Sallen_key
plot ylimit -55 2 xlog Thorlabs computed_response vs frequency_list
plot xlog error vs frequency_list ylimit -10 5

.endc

.include filterlab3.lib          ; include the SPICE netlist from the Microchip Filterlab tool
.end
```

This showed how the Sallen-key implementation compared to the purchased Thor Labs filter.



===

Some treasures are to be found here:

http:;www.idea2ic.com/NGSPICE_TEMPLATES/NGSPICE%20TEMPLATES.html