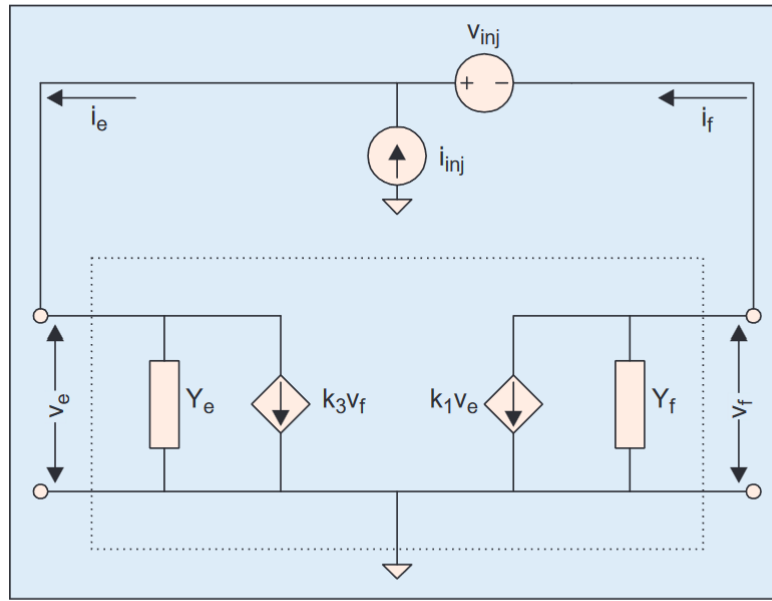


1. Theory behind closed loop stability analysis

It all started with the below paper published by Michael Tian in 2001:

<https://kenkundert.com/docs/cd2001-01.pdf>

While I'm not 100% sure of all the details in this paper, the essence of it is show in Fig.7 below:



7. Double-injection technique based on the bilateral return loop model.

The paper introduces a double injection technique consisting of a voltage probe (vprb) and current probe (iprb) placed in the loop. To perform the technique we run 2 ac analysis. In the first, we set the ac value of vprb to 1 and the ac value of iprb to 0. In the second, the ac values of the 2 sources are reversed. Justin gives a nice explanation of this at the below link:

<http://education.ingenazure.com/ac-stability-analysis-ngspice/>

The loop gain characteristics are then quantified via the below equation (which is the main contribution of the paper):

$$T = \frac{2(AD - BC) - A + D}{2(BC - AD) + A - D + 1}. \quad (30)$$

So what do A, B, C and D mean in (30)?

As already mentioned, the technique performs 2 ac analysis:

ac1: $|vprb| = 1$, $|iprb| = 0$

ac2: $|vprb| = 0$, $|iprb| = 1$

The parameters of (30) are thus defined as:

A = current through iprb at ac2

B = current through iprb at ac1

C = voltage at vprb at ac2

D = voltage at vprb at ac1

As per Fig.1, the currents flow into the -ve terminal of iprb which in spice means they are -ve. Therefore, for elegance, which will become apparent later, we define:

$$A^* = -A$$

$$B^* = -B$$

Now we are ready to break down the equation and make practical use of it. To do this, we refer to Frank Widermans excellent link below:

<https://groups.io/g/LTspice/message/4141>

In this link he shows the following:

$$T = \frac{2(AD - BC) - A + D}{2(BC - AD) + A - D + 1} \dots (1)$$

$$= \frac{1}{\frac{2(BC - AD) + A - D + 1}{2(AD - BC) - A + D}} \dots (2)$$

$$= \frac{1}{-1 + \frac{1}{2(AD - BC) - A + D}} \dots (3)$$

$$= \frac{1}{\left[\frac{1}{2(AD - BC) - A + D} \right] - 1} \dots (4)$$

Substuting in A* and B*, after some re-arranging, gives:

$$T = \frac{1}{\left[\frac{1}{2(B^*C - DA^*) + D + A^*} \right] - 1} \dots (5)$$

In Justins example he makes the following definitions:

A*: let ip21 = ac2.i(vprobe1)

B*: let ip11 = ac1.i(vprobe1)

C: let vprb2 = ac2.probe

D: let vprb1 = ac1.probe

This allows him to define T as:

$$\text{let } av = 1/(1/(2*(ip11*vprb2-vprb1*ip21)+vprb1+ip21)-1)$$

Placing the substitutions for A*, B*, C, D into (5) give the above expression exactly.

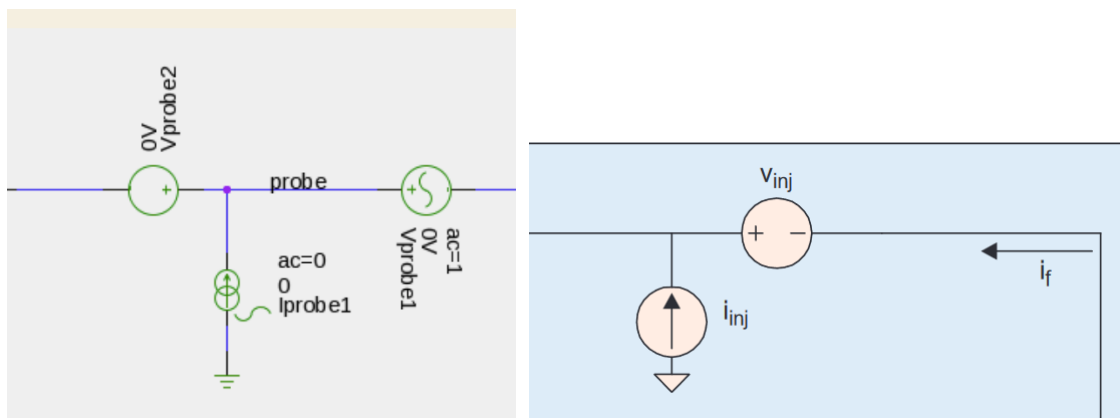
Personally I don't bother re-defining things and just create the expression as follows:

$$\text{let } av = \{1/(1/(2*(ac1.i(v.xstb.Vi)*ac2.v(xstb.x)-ac1.v(xstb.x)*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))-1)\}$$

This expression can be taken at face value for now as it will make more sense when we discuss how to implement this technique.

2. Implementing closed loop stability analysis in ngspice / xschem

Below compares Justin's implementation of the technique with what was published. You will see they only differ by the buffer (Vprobe2) placed by Justin. Unsure exactly why this is but this is the version we are going to implement.



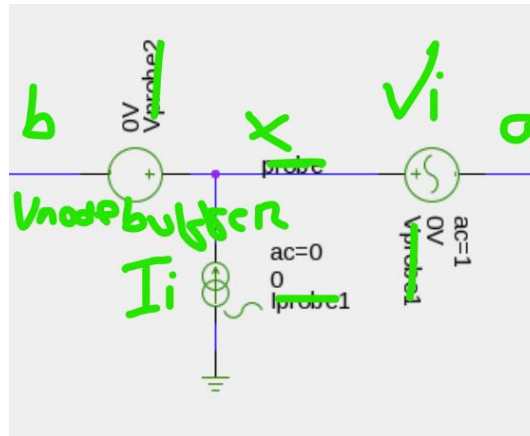
Instead of having to place down voltage / current sources every time we want to perform a closed loop stability measurement, it is much more efficient to place the above in a subckt and instantiate that instead. This is exactly what is done in Cadence with their iprobe symbol, shown.



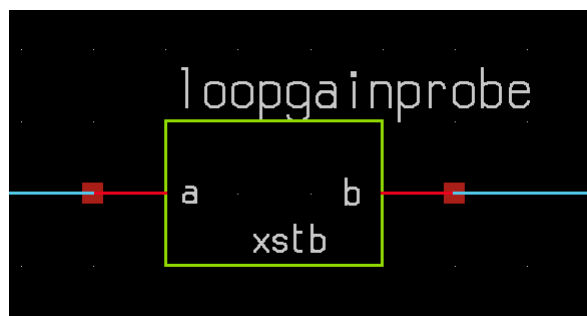
To create the subckt we are going to follow Roberts naming convention in the below post:

[ngspice / Feature Requests / #34 Request for Stability Analysis \(sourceforge.net\)](https://ngspice.org/Feature-Requests/#34-Request-for-Stability-Analysis)

This means renaming Justins nets / probes as follows:



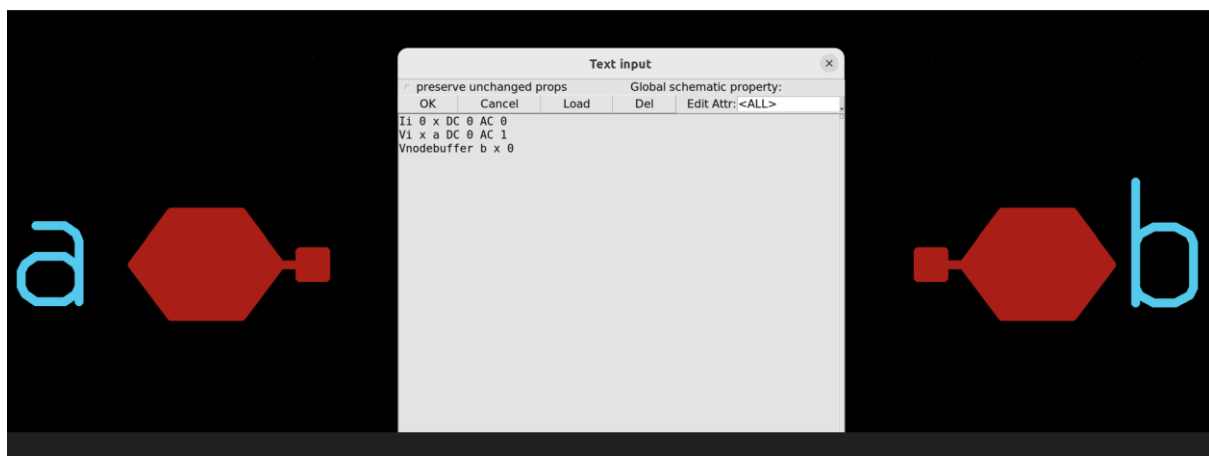
So we simply create an empty schematic in xschem with 2 pins (a and b) which we then make into a symbol, as shown below:



Note: See below link for how to create symbols from schematics in xschem:

https://xschem.sourceforge.io/stefan/xschem_man/creating_symbols.html

Next, descend into the symbol, press 'q' and enter the below text which defines the subckt:



Now, everytime you want to perform closed loop stability analysis, insert this probe and name it "xstb".

At this point it is worth revisiting the below expression:

```
let av = {1/(1/(2*(ac1.i(v.xstb.Vi)*ac2.v(xstb.x)-
ac1.v(xstb.x)*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))-1)}
```

Now it can be understood as follows:

A*: ac2.i(v.xstb.Vi) ... probe current through Vi in xstb from the 2nd ac analysis

B*: ac1.i(v.xstb.Vi) ... probe current through Vi in xstb from the 1st ac analysis

C: ac2.v(xstb.x) ... probe voltage at node x in xstb from the 2nd ac analysis

D: ac1.v(xstb.x) ... probe voltage at node x in xstb from the 1st ac analysis

By substituting the above into (5) (shown below for reference), you will arrive at my expression in ngspice:

$$T = \frac{1}{\left[\frac{1}{2(B^*C - DA^*) + D + A^*} \right] - 1} \dots (5)$$

So in ngspice you will place this expression but only after both ac analyses have been done. The first ac analysis is where we set the ac value of v.xstb.Vi to 1 and the ac value of i.xstb.li to 0, as shown below (note we save the results to raw file tb_OTA_1stage_ac1.raw).

**** 2. AC ANALYSIS ****

```
alter i.xstb.li acmag=0
alter v.xstb.Vi acmag=1
```

```
ac dec 10 1 100G
remzerovec
write tb_OTA_1stage_ac1.raw
```

The 2nd ac analysis is where we set the ac value of v.xstb.Vi to 0 and the ac value of i.xstb.li to 1, as shown below. It is after we have performed this 2nd ac analysis that we can use our Tian equation after which we write the results to raw file tb_OTA_1stage_ac2.raw.

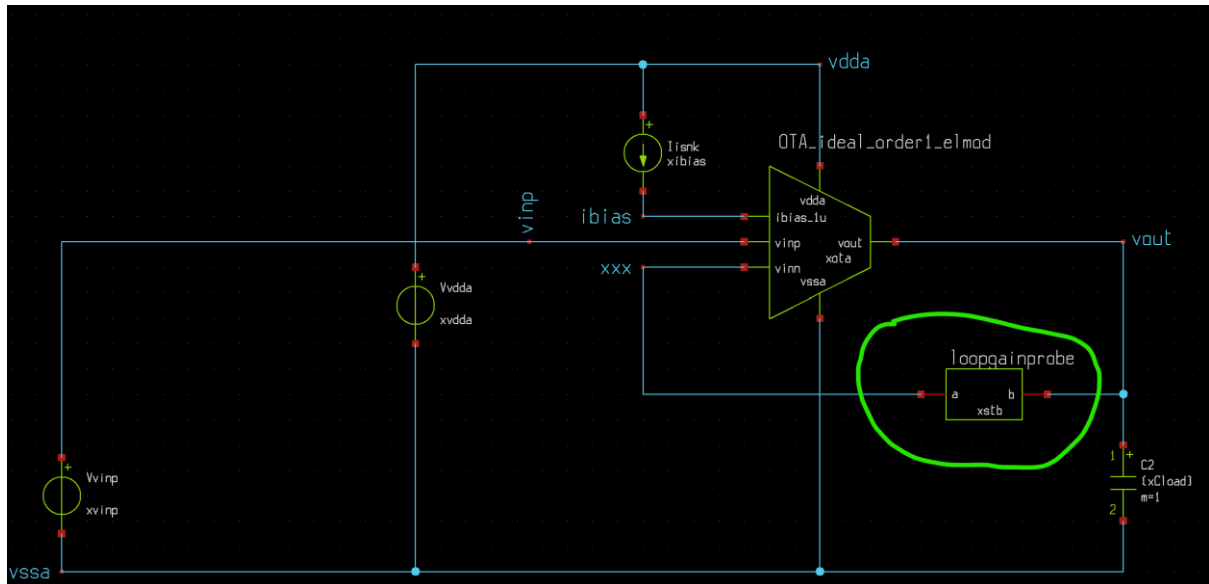
```
alter i.xstb.li acmag=1
alter v.xstb.Vi acmag=0
```

```
ac dec 10 1 10G
```

```
let av = {1/(1/(2*(ac1.i(v.xstb.Vi)*ac2.v(xstb.x)-
ac1.v(xstb.x)*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))-1)}
```

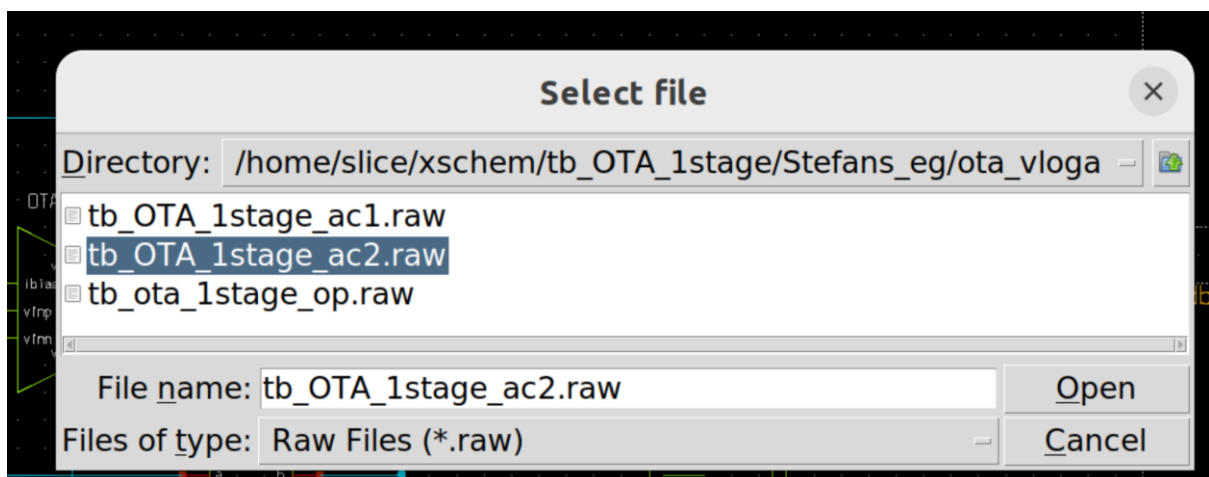
```
remzerovec
write tb_OTA_1stage_ac2.raw
```

Below exemplifies the use of the probe in a circuit with the resulting code shown in the Appendix. Note: There is a lot going on in this code so I have highlighted in bold the sections relevant to this closed loop stability analysis.



3. Viewing results from the closed loop stability analysis in xschem

As per section 2, the closed loop characteristics are stored as variable “av” saved in file tb_OTA_1stage_ac2.raw. To view this data in xschem, open the waveviewer, create a plot and open tb_OTA_1stage_ac2.raw as per below:

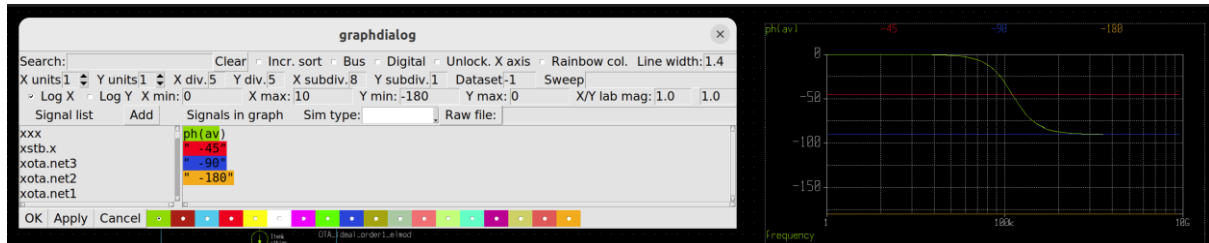


Note: For information on xschems waveviewer, the reader is referred to the following link:

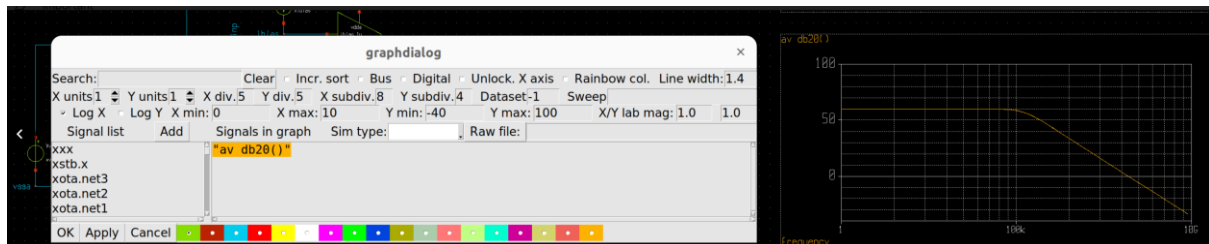
https://xschem.sourceforge.io/stefan/xschem_man/graphs.html

In the waveviewer, double click and select “av”. In the below we wanted to plot the closed loop phase response and so inputted ph(av).

Note: To get horizontal cursors we inputted “<y-evel>”. E.g. for a horizontale cursor placed at -45deg, input “-45”. To place a vertical cursor press ‘a’. To place a 2nd vertical cursor, and make a differential measurement with the first, press ‘b’. Then to remove the cursors, simply place ‘a’ and ‘b’ again.



In the below we wanted to plot the closed loop magnitude response and so inputted “av db20()”.



4. Creating expressions from the closed loop stability analysis in xschem / ngspice

Ngspice is very good at dealing with measures. Below shows example measures related with determining the various characteristics of a loops response.

**** 3. MEASURES ****

```
let n45_rads = -45*(pi/180)
meas AC Av_0 FIND vdb(av) AT=10
echo --
meas AC BW WHEN vp(av)=n45_rads CROSS=1
echo --
meas AC Av_BW FIND vdb(av) WHEN vp(av)=n45_rads CROSS=1
echo --
meas AC ULGF WHEN vdb(av)=0
echo --
meas AC ULGF_phi_rads FIND vp(av) WHEN vdb(av)=0 CROSS=1
let ULGF_phi_deg = ULGF_phi_rads*(180/pi)
print ULGF_phi_deg
```

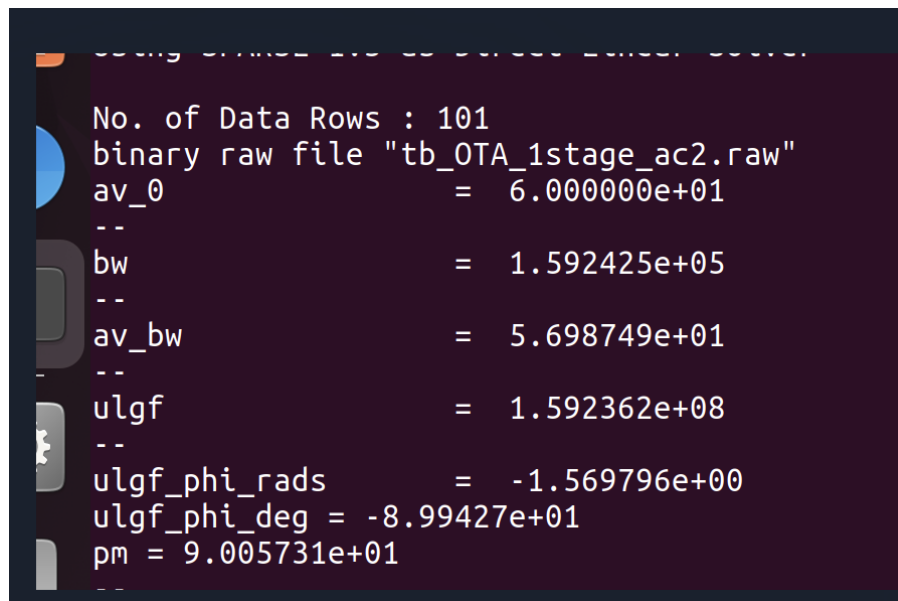
```

let PM = 180+ULGF_phi_deg
print PM
echo --
* putting in -170 since this is a 2nd-order system so doesnt ever achieve -180
let n180_rads = -170*(pi/180)
meas AC GM FIND vdb(av) WHEN vp(av)=n180_rads CROSS=1

```

Anything behind this and you can search various measures in the forum and (or) post to the forum itself.

Below shows the ngspice output for these measures:



```

No. of Data Rows : 101
binary raw file "tb_OTA_1stage_ac2.raw"
av_0 = 6.000000e+01
--
bw = 1.592425e+05
--
av_bw = 5.698749e+01
--
ulgf = 1.592362e+08
--
ulgf_phi_rads = -1.569796e+00
ulgf_phi_deg = -8.99427e+01
pm = 9.005731e+01
--

```

There are many more expressions which can be done in ngspice that are documented in the manual and discussed in various posts of the forum.

Note you can create expressions in xschem. But this expression editor is intended to create expressions which returns a single waveform and not a single waveform number.

5. Making the stb probe available for global usage in xschem

To avoid having to copy the loopgainprobe schematic and symbol to the directory of every tb you are using it in, it is more efficient to place it in the global directory found at:

```
/usr/local/share/xschem/xschem_library/devices
```

To do this, first you need to change ownership of this dir to yourself to allow you to copy files into it. This is done using the "chown" command (short for change ownership) as follows:

```
sudo chown -R <username> /usr/local/share/xschem/xschem_library/devices
```


To find your username (if you don't already know it) just type whoami into a terminal. For me it is slice making the below command:

`sudo chown -R slice /usr/local/share/xschem/xschem_library/devices`

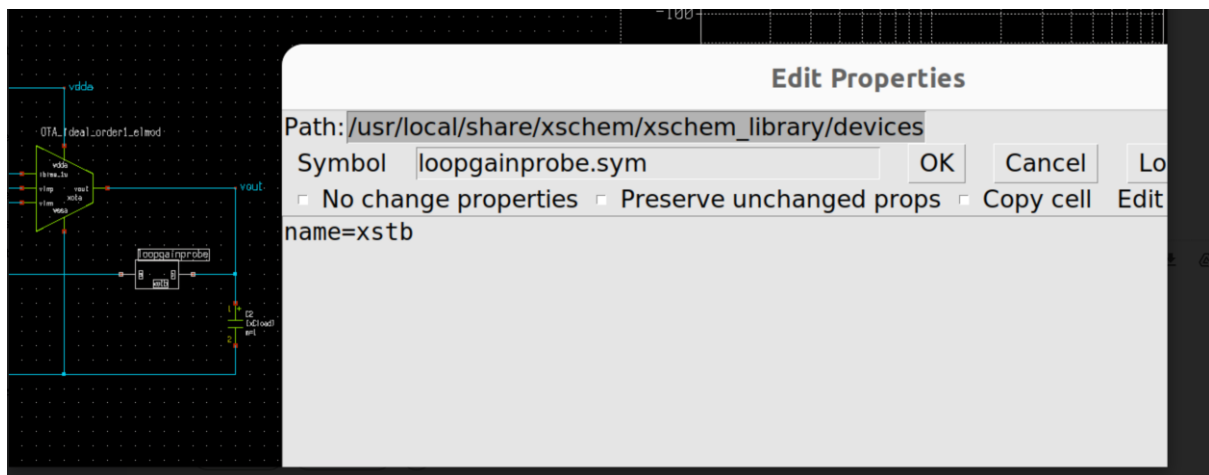
Now simply copy loopgainprobe.sch and loopgainprobe.sym into the dir as normal. Result will be as follows:

```

slice@slice-Inspiron-16-Plus-7620: /usr/local/share/xschem/xschem_library/devices$ p
/usr/local/share/xschem/xschem_library/devices
slice@slice-Inspiron-16-Plus-7620: /usr/local/share/xschem/xschem_library/devices$ l
adc_bridge.sym      code_shown.sym      device_param_probe.sym  k.sym
anmeter.sym         conn_10x2.sym        diode.sym               lab_generic.sym
arch_declarations.sym  conn_14x1.sym        generic_pin.sym         lab_pin.sym
architecture.sym      conn_3x1.sym         gnd.sym                 lab_show.sym
asrc.sym             conn_4x1.sym         ind.sym                 lab_wire.sym
assign.sym            conn_6x1.sym         intuitive_interface_cheatsheet.sch  launcher.sym
a11butes.sym          conn_8x1.sym         intuitive_interface_cheatsheet.sym  led.sym
b_e.sym              connecto.sym         loopgainprobe.sch       loopgainprobe.sym
bus_connect.sym       connecto.sym         netlist_at_end.sym      netlist_not_shown_at_end.sym
bus_tap.sym           crystal_2.sym         isource_arith.sym       netlist_not_shown.sym
capa_2.sym            crystal.sym           isource_pwl.sym         netlist_options.sym
cccs.sym              dac_bridge.sym        isource.sym             netlist.sym
ccvs.sym              delay_line.sym        jumper.sym              ngspice_analog_delay.sym
ngspice-Inspiron-16-Plus-7620: /usr/local/share/xschem/xschem_library/devices$ l "loop"
loopgainprobe.sch  loopgainprobe.sym

```

From now on insert the loopgainprobe.sym from this dir as per below:



6. Example

All files necessary to run the above closed loop stability analysis have been checked into the below location in github:

https://github.com/SLICESemiconductor/OpenSourceTool_Examples/tree/main/Running_closed_loop_stability_analysis_in_ngspice_through_xschem

7. Appendix

**** sch_path:** /home/slice/xschem/tb_OTA_1stage/Stefans_eg/ota_vloga/tb_OTA_1stage.sch

**** .subckt** tb_OTA_1stage

Vvssa vssa GND 0

Vvdda vdda vssa xvdda

Vvinp vinp vssa xvinp

xota vdda vout vfb vinp ibias vssa OTA_ideal_order1_elmod

C2 vout vssa {xCload} m=1

xstb vfb vout loopgainprobe

```
lisnk vdda ibias xibias
```

```
**** begin user architecture code
```

```
* Models
```

```
* Note: I name my nmos models as nmos and pmos ones as pmosx (just messing a bit)
```

```
.model nmos nmos level=54 version=4.8.2
```

```
.model pmosx pmos level=54 version=4.8.2
```

```
* Parameters
```

```
.param xvdda = 1.8
```

```
.param xvinp = 1
```

```
.param xibias = 2u
```

```
.param xCload = 0p
```

```
.param xgm_dp = 1e-3
```

```
.param xRin = 1/xgm_dp
```

```
.param xCout = 1p
```

```
.param xRout = 1e6
```

```
.model OTA_vcv5 OTA_vcv5
```

```
* vlogA instantiation
```

```
** 1. DCOP analysis **
```

```
** must save the below for DCOP analysis to be back annotated onto the schematic
```

```
.option savecurrents
```

```
.save
```

```
+ @m.xota.m1[vgs]
```

```
+ @m.xota.m1[vth]
```

```
+ @m.xota.m1[gds]
```

```
+ @m.xota.m1[gm]
```

```
+ @m.xota.m1[id]
```

```
+ @m.xota2.m1[vgs]
```

```
+ @m.xota2.m1[vth]
```

```
+ @m.xota2.m1[gds]
```

```
+ @m.xota2.m1[gm]
```

```
+ @m.xota2.m1[id]
```

```
+ @m.xota.m2[vgs]
```

```
+ @m.xota.m2[vth]
```

```
+ @m.xota.m2[gds]
```

```
+ @m.xota.m2[gm]
```

- + @m.xota.m2[id]
- + @m.xota2.m2[vgs]
- + @m.xota2.m2[vth]
- + @m.xota2.m2[gds]
- + @m.xota2.m2[gm]
- + @m.xota2.m2[id]

- + @m.xota.m3[vgs]
- + @m.xota.m3[vth]
- + @m.xota.m3[gds]
- + @m.xota.m3[gm]
- + @m.xota.m3[id]
- + @m.xota2.m3[vgs]
- + @m.xota2.m3[vth]
- + @m.xota2.m3[gds]
- + @m.xota2.m3[gm]
- + @m.xota2.m3[id]

- + @m.xota.m4[vgs]
- + @m.xota.m4[vth]
- + @m.xota.m4[gds]
- + @m.xota.m4[gm]
- + @m.xota.m4[id]
- + @m.xota2.m4[vgs]
- + @m.xota2.m4[vth]
- + @m.xota2.m4[gds]
- + @m.xota2.m4[gm]
- + @m.xota2.m4[id]

- + @m.xota.m5[vgs]
- + @m.xota.m5[vth]
- + @m.xota.m5[gds]
- + @m.xota.m5[gm]
- + @m.xota.m5[id]
- + @m.xota2.m5[vgs]
- + @m.xota2.m5[vth]
- + @m.xota2.m5[gds]
- + @m.xota2.m5[gm]
- + @m.xota2.m5[id]

- + @m.xota.m6[vgs]

```

+ @m.xota.m6[vth]
+ @m.xota.m6[gds]
+ @m.xota.m6[gm]
+ @m.xota.m6[id]
+ @m.xota2.m6vgs]
+ @m.xota2.m6[vth]
+ @m.xota2.m6[gds]
+ @m.xota2.m6[gm]
+ @m.xota2.m6[id]

```

```

.control
pre_osdi OTA_vcv.sosdi
save all

```

```

op
write tb_ota_1stage_op.raw

```

**** 2. AC ANALYSIS ****

```

alter i.xstb.Ii acmag=0
alter v.xstb.Vi acmag=1

```

```

ac dec 10 1 100G
remzerovec
write tb_OTA_1stage_ac1.raw

```

```

alter i.xstb.Ii acmag=1
alter v.xstb.Vi acmag=0

```

```

ac dec 10 1 10G

```

*** use this line if you want the phase response to start at 180deg**

```

*let av = {1/(1-1/(2*(ac1.i(v.xstb.Vi)*ac2.v(xstb.x)-
ac1.v(xstb.x)*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))))}

```

*** use this line if you want the phase response to start at 0deg (more conventional and directly corresponds to Franks derivation)**

```

let av = {1/(1/(2*(ac1.i(v.xstb.Vi)*ac2.v(xstb.x)-
ac1.v(xstb.x)*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))-1)}

```

```

remzerovec

```

```
write tb_OTA_1stage_ac2.raw
```

```
*ngspice plots
```

```
*plot vdb(av) xlog
```

```
*plot {(180/pi)*vp(av)} xlog
```

```
*plot 180*cph(av)/pi
```

```
** 3. MEASURES **
```

```
let n45_rads = -45*(pi/180)
```

```
meas AC Av_0 FIND vdb(av) AT=10
```

```
echo --
```

```
meas AC BW WHEN vp(av)=n45_rads CROSS=1
```

```
echo --
```

```
meas AC Av_BW FIND vdb(av) WHEN vp(av)=n45_rads CROSS=1
```

```
echo --
```

```
meas AC ULGF WHEN vdb(av)=0
```

```
echo --
```

```
meas AC ULGF_phi_rads FIND vp(av) WHEN vdb(av)=0 CROSS=1
```

```
let ULGF_phi_deg = ULGF_phi_rads*(180/pi)
```

```
print ULGF_phi_deg
```

```
let PM = 180+ULGF_phi_deg
```

```
print PM
```

```
echo --
```

```
* putting in -170 since this is a 2nd-order system so doesnt ever achieve -180
```

```
let n180_rads = -170*(pi/180)
```

```
meas AC GM FIND vdb(av) WHEN vp(av)=n180_rads CROSS=1
```

```
setplot
```

```
.endc
```

```
**** end user architecture code
```

```
** .ends
```

```
* expanding symbol: loopgainprobe.sym # of pins=2
```

```
** sym_path: /usr/local/share/xschem/xschem_library/devices/loopgainprobe.sym
```

```
** sch_path: /usr/local/share/xschem/xschem_library/devices/loopgainprobe.sch
```

```
.subckt loopgainprobe a b
```

```
*.iopin b
```

```

*.iopin a
**** begin user architecture code
li 0 x DC 0 AC 0
Vi x a DC 0 AC 1
Vnodebuffer b x 0
**** end user architecture code
.ends

```

```

* expanding symbol: OTA_ideal_order1_elmod.sym # of pins=6
** sym_path: /home/slice/xschem/tb_OTA_1stage/Stefans_eg/ota_vloga/OTA.sym
** sch_path:
/home/slice/xschem/tb_OTA_1stage/Stefans_eg/ota_vloga/OTA_ideal_order1_elmod.sch
.subckt OTA_ideal_order1_elmod vdda vout vinn vinp ibias_1u vssa
*.ipin vssa
*.ipin vinp
*.ipin ibias_1u
*.ipin vinn
*.ipin vdda
*.opin vout
E1 net1 vssa vinp vinn 1
R1 net2 vssa {xRin} m=1
F1 vdda net3 Vidp 1
Vidp net1 net2 0
R2 vout vssa {xRout} m=1
C2 vout vssa {xCout} m=1
Viout net3 vout 0
R3 ibias_1u vssa 1MEG m=1
.ends

.GLOBAL GND
.end

```