Qspice - Command Reference Guide by KSKelvin

KSKelvin Kelvin Leung

Created on 8-4-2023
Last update on 2-2-2025

Comment
(Shortcut “;”)

Comment

Qspice : Comment - 3 Type.qsch | Comment - Chinese.qsch

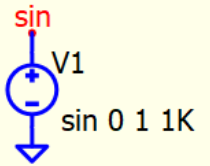
3 Type of Comment

Type #1 : double slash //

```
//Type 1:
//Text Comment with double slash
```

Type #2 : Shortcut semicolon ";"

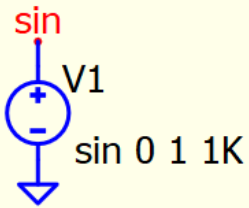
```
Type 2:
Text Comment
  * Right Click Text Box > This is a text comment
  * OR Shortcut ";"
```



Type #3 : semicolon in directive

```
.tran 5/1K ;Type 3 : Text Comment with semicolon in directive
.plot V(sin)
```

Comment with Chinese Character




```
//繁體中文範例
//简体中文范例
```

繁體中文範例 (Traditional Chinese)
简体中文范例 (Simplified Chinese)

```
.tran 5/1K ;瞬態響應
.plot V(sin) ;繪劃波型
```

Comment – Common Unicode in Electronics

Qspice : Comment - Unicode Common Character for Qspice Text.qsch

https://en.wikipedia.org/wiki/List_of_Unicode_characters

https://www.compart.com/en/unicode/block

Unicode Block "Superscripts and Subscripts" U+2070 to U+209F

Superscripts : ⁰ ¹ ² ³ ⁴ ⁵ ⁶ ⁷ ⁸ ⁹ ⁺ ⁻ ⁼ ⁽ ⁾ ⁿ

Subscripts : ₀ ₁ ₂ ₃ ₄ ₅ ₆ ₇ ₈ ₉ ₊ ₋ ₌ ₍ ₎ ₐ ₑ ₒ ₓ ₔ ₕ ₖ ₗ ₘ ₙ ₚ ₛ ₜ

Unicode Block "Latin-1 Supplement" U+00A1 to U+00BF, U+00D7, U+00F7

Latin-1 : ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿

Math : × ÷

Unicode Block "Greek and Coptic" U+0391 to U+03A9 and U+03B1 to U+03C9

Greek (Capital) : Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω

Greek (Small Letter) : α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω

Unicode Block "Number Forms" between U+2150 to U+218F

Number Forms : ⅐ ⅑ ⅒ ⅓ ⅔ ⅕ ⅖ ⅗ ⅘ ⅙ ⅚ ⅛ ⅜ ⅝ ⅞ ⅟ ↉

Unicode Block "Mathematical Operators" between U+2200 to U+22FF

Math : ∆ ∇ ∑ − ∓ ∘ √ ∛ ∜

Math : ∝ ∞

Math : ∴ ∵ ≤ ≥

- Unicode in Qspice
  - Unicode is only accepted in the "TEXT" of a schematic, not in the attributes of a symbol
  - QSpice schematic files (.qsch) are encoded in ANSI, not UTF-8, and .qsch only partially accept Unicode.

Comment – All ANSI Characters available in Qspice

Qspice : Comment - ANSI Character for Qspice Text and Attribute.qsch

X1
Attribute

! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
[\ ] ^ _ `
a b c d e f g h i j k l m n o p q r s t u v w x y z
{ | } ~ † ¢ £ ¤ ¥ ¦ § © « ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿
À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý
Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

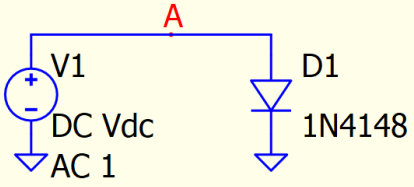
- ANSI in Qspice
  - Qspice files (.qsch, .qraw, .cir, .lib etc..) are in ANSI encoding
  - Symbol attribute only accept ANSI characters but not Unicode

.ac
Small-Signal AC Analysis

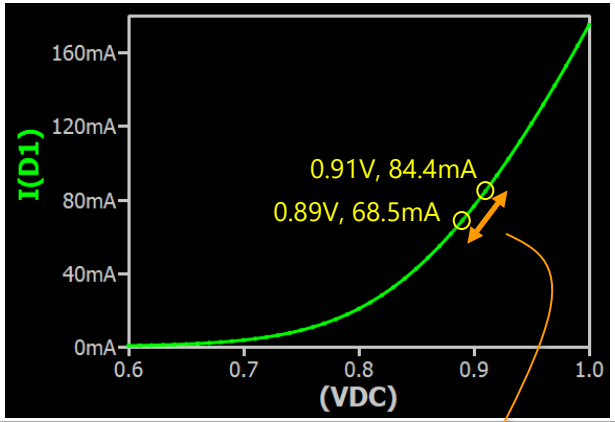
.ac – Small-Signal AC Analysis

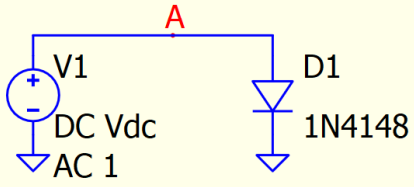
Qspice : .ac - basic.qsch

- Small-Signal AC Analysis
  - .ac performs small-signal AC analysis by computing the frequency domain response at the circuit's DC operating point
    - .op is executed to calculate the DC operating point before running .ac
  - This technique is effective for continuous-time non-switching circuits. For small-signal response analysis of switching circuits, the .bode directive
  - In this example, .step is utilized to sweep through different DC operating points and obtain .ac results at a single frequency point

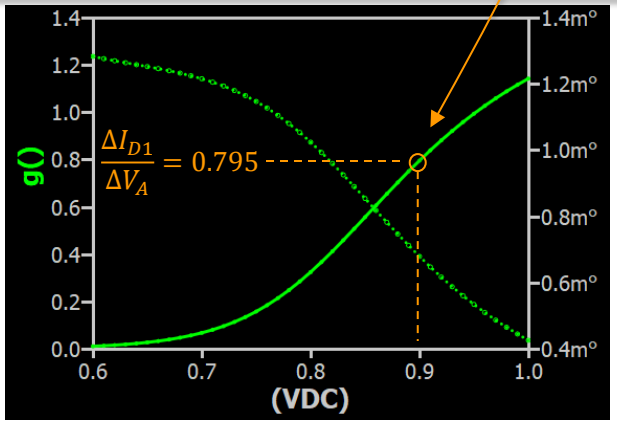



```
.step param Vdc 0.6 1 0.01
.op                .ac list 1K
.plot I(D1)        .func g() I(D1)/V(a)
                   .plot g()
```





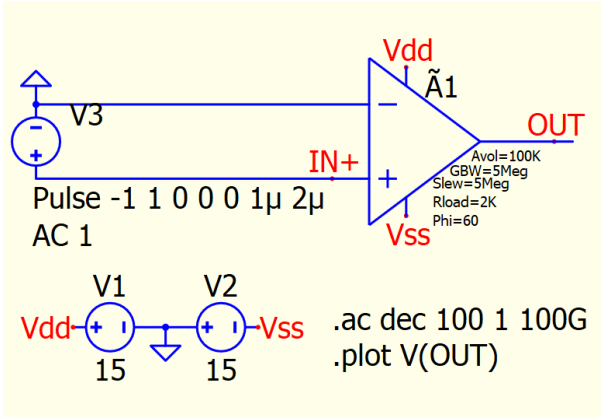

```
.step param Vdc 0.6 1 0.01
.op                .ac list 1K
.plot I(D1)        .func g() I(D1)/V(a)
                   .plot g()
```

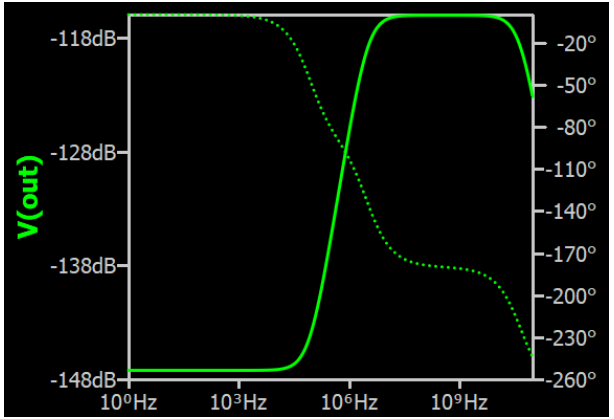


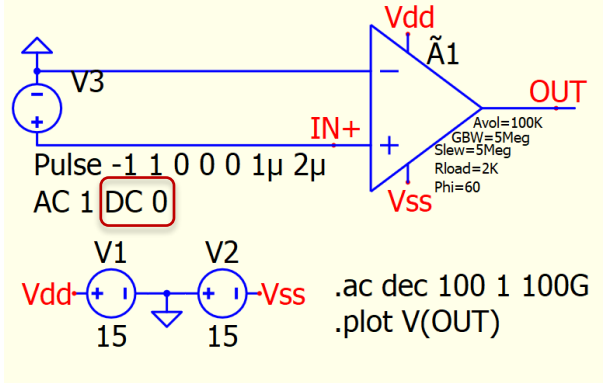
.ac – DC operating point in .ac from V-Source

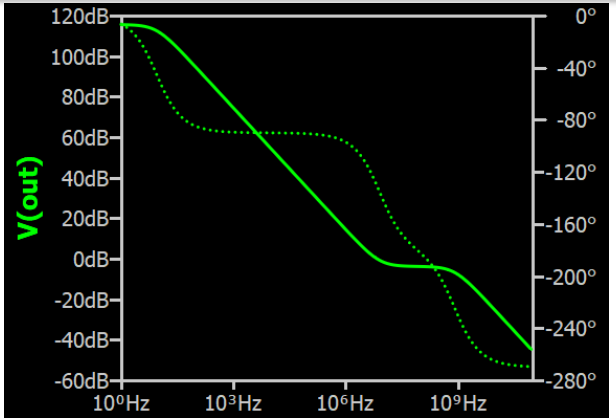
Qspice : .ac - DC in V-source.qsch

- DC operating point
  - There is special attention in .ac with source that support transient analysis
  - This example measures opamp open loop gain, but with a pulse statement included in V3, open loop profile is affected
  - The reason is that, in default, .op is run with source voltage at t=0 if DC is not defined. This pulse statement set -1V at t=0, which saturated this opamp by .op with V(IN+)=-1V
  - Solution is to add DC into V-source to force a DC voltage for .op before .ac is run
  - More information can be found in ".op" section





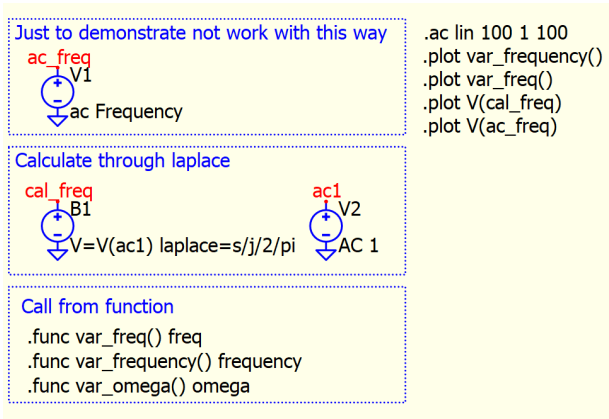




.ac – keyword : freq, frequency and omega

Qspice : .ac - keyword freq and frequency.qsch

- Keyword
  - Freq/Frequency/Omega are keywords used in .ac analysis, it is advisable not to define .param with these keywords, as it may result in errors
  - These keywords can be accessed in post-processing (thus, .func can use freq/frequency/omega), but they are not available during the simulation run
  - It can calculate frequency with Laplace function ($\frac{s}{j2\pi}$) and an AC 1V source

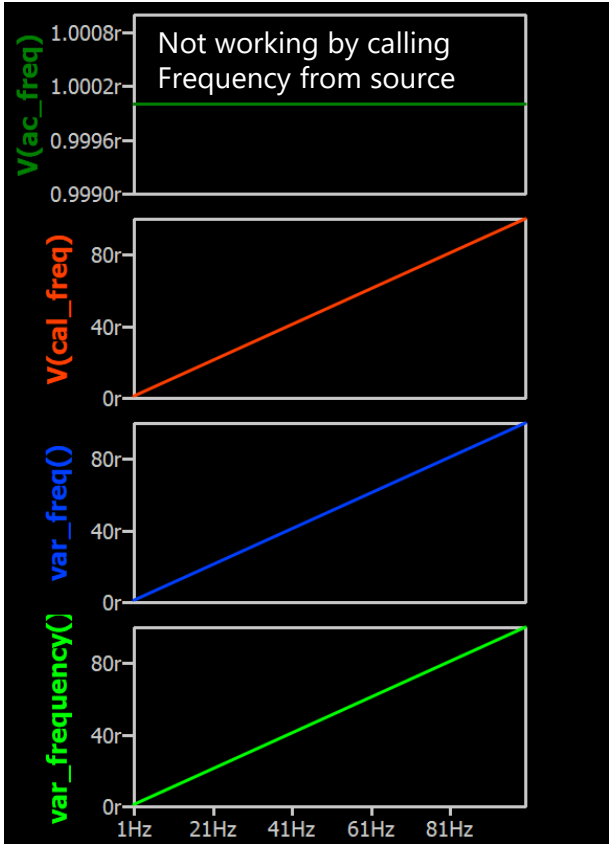



```
.ac lin 100 1 100
.plot var_frequency()
.plot var_freq()
.plot V(cal_freq)
.plot V(ac_freq)
```

.ac - keyword freq and frequency.qraw

```
1   Title: * C:\KSKelvinQspice\01 U
2   Date: Mon May 13 21:10:35 2024
3   Plotname: AC Analysis
4   Plot Suggestion(s): «var_freque
5   Flags: complex
6   Abscissa:     1.000000000000000
7   No. Variables: 7
8   No. Points: 100
9   Command: QSPICE80, Build May 11
10  .param temp=27
11  .func VAR_FREQ()FREQ
12  .func VAR_FREQUENCY()FREQUENCY
13  .alias Freq Frequency
14  .alias Omega 2*pi*Frequency
15  Variables:
16      0   Frequency   frequency
17      1   V(ac_freq)  voltage
18      2   V(ac1)  voltage
19      3   V(cal_freq) voltage
```

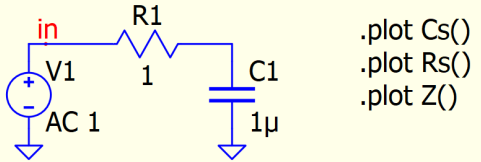
keywords



.ac – use frequency in .ac

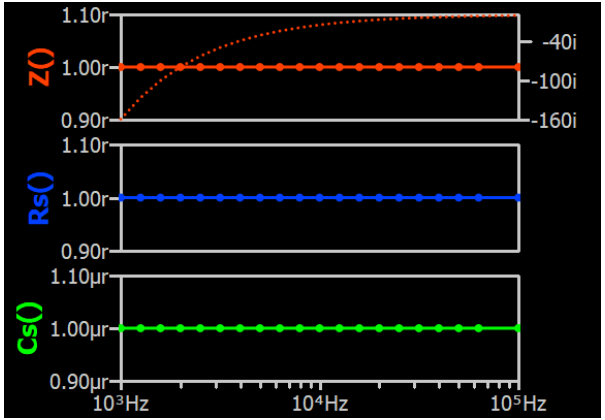
Qspice : .ac - freq from keyword.qsch | .ac - frq with list.qsch

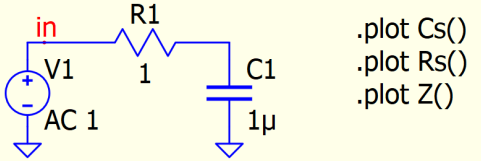
- Use frequency in .ac
  - Examples to demonstrate if simulation requires frequency information
  - Example #1
    - Use keyword (e.g. freq) in post-processing calculation
    - Advantage is faster simulation, but only available for post-processing calculations such as .func
  - Example #2
    - Define .param frq and step a single point frequency analysis .ac list
    - Advantage is that frq is defined as a .param and can be used during simulation. However, simulation is slower because each step requires recalculating .op
    - This method is most reliable way for .ac with behavioral devices




```
.plot Cs()
.plot Rs()
.plot Z()

.ac dec 10 1K 100K

.func Z() V(in)/-I(V1)
.func Rs() re(Z())
.func Cs() -1/im(Z())/2/pi/freq
```

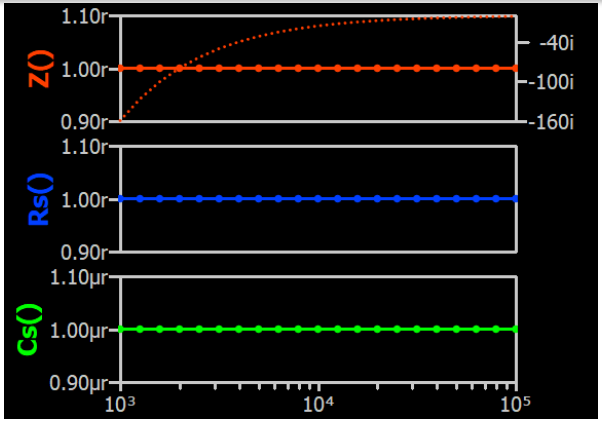
Use keyword freq






```
.plot Cs()
.plot Rs()
.plot Z()

.param frq=1K
.step dec param frq 1K 100K 10
.ac list frq

.func Z() V(in)/-I(V1)
.func Rs() re(Z())
.func Cs() -1/im(Z())/2/pi/frq
```

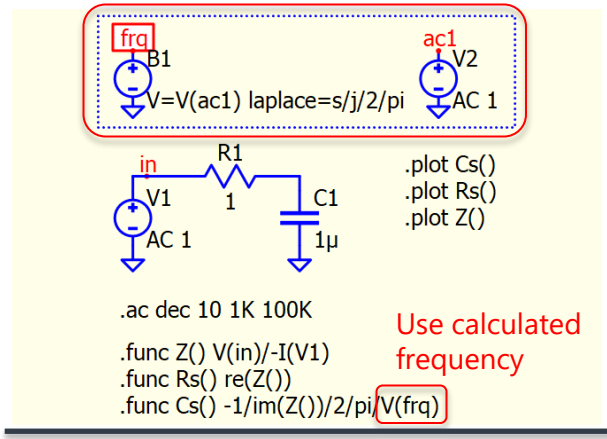
Define param frq
Use .ac list

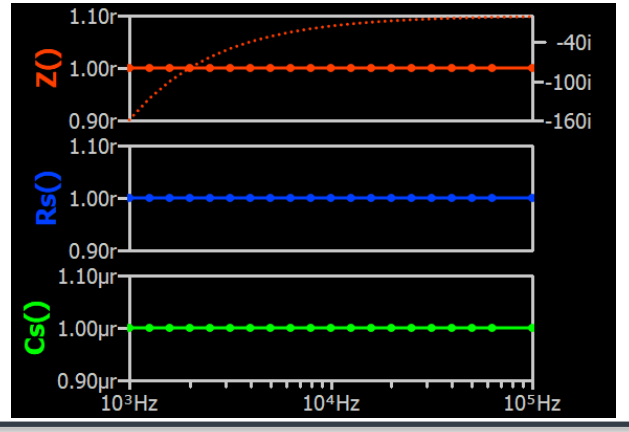


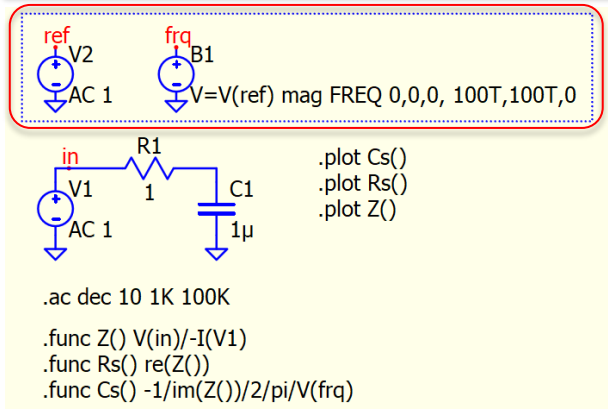
.ac – use frequency in .ac

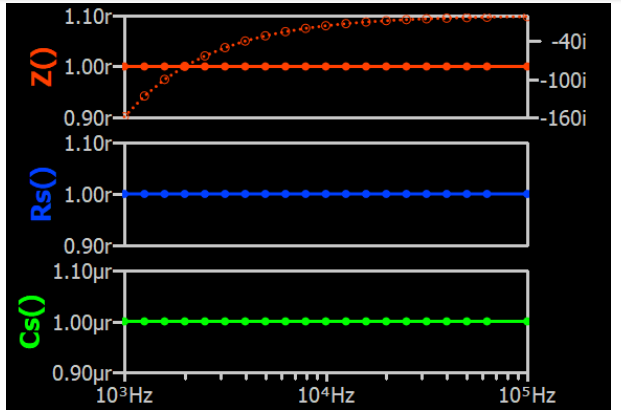
Qspice : .ac - frq with symbol-1.qsch | .ac - frq with symbol-2.qsch

- Use frequency in .ac
  - Example #3 (Not Recommend)
    - Use a frequency calculation behavioral source and utilize its result for calculations
    - Unlike the previous case, this frequency is captured from a voltage node and expressed as V(frq)
    - Be cautious that using V(frq) to define a behavioral device may not work! It is recommended to use the method shown in the 2nd example









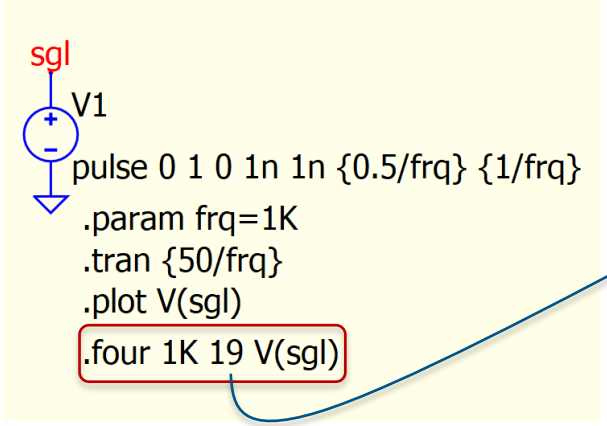
.four
THD Computation

.four THD Computation

Qspice : .four - harmonics.qsch

- Syntax: .four FREQ [HARMONICS] [PERIODS] expr1 [expr2 [expr3 [...]]]
  - FREQ : fundamental frequency
  - HARMONICS : number of harmonics to compute (Default HARMONICS=9)
  - PERIODS : number of period to be used to compute THD
  - expr1 : expression (e.g. V(out), I(V1) etc...)

- [HARMONICS]
  - Number of harmonics to compute
  - In this example, it computed 1st to 19th harmonic of [FREQ] (e.g. 1kHz)




```
.four 1k 19 v(sgl):
Magnitude of Fundamental(RMS): 0.450158
Harmonic  Frequency     Magnitude     Phase
   1      1.000e+03     1.000e+00      0.00°
   2      2.000e+03     5.890e-06    -90.00°
   3      3.000e+03     3.333e-01     -0.00°
   4      4.000e+03     5.890e-06    -90.00°
   5      5.000e+03     2.000e-01     -0.01°
   6      6.000e+03     5.890e-06    -89.99°
   7      7.000e+03     1.429e-01     -0.01°
   8      8.000e+03     5.890e-06    -89.99°
   9      9.000e+03     1.111e-01     -0.01°
  10      1.000e+04     5.890e-06    -89.99°
  11      1.100e+04     9.091e-02     -0.02°
  12      1.200e+04     5.890e-06    -89.99°
  13      1.300e+04     7.692e-02     -0.02°
  14      1.400e+04     5.890e-06    -89.99°
  15      1.500e+04     6.667e-02     -0.02°
  16      1.600e+04     5.890e-06    -89.99°
  17      1.700e+04     5.882e-02     -0.02°
  18      1.800e+04     5.890e-06    -89.99°
  19      1.900e+04     5.263e-02     -0.03°
THD = 45.686%(48.3398%)
```

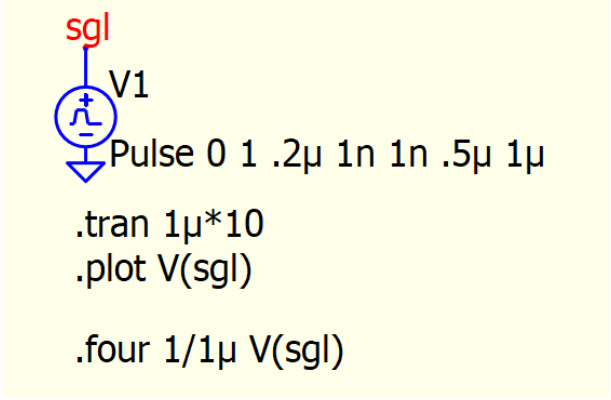
THD : total harmonic distortion

.four meaning and reconstruction

Qspice : .four - explain.qsch | .four - reconstruction

- Meaning from .four
  - [1] Fundamental Return
    - RMS Magnitude (Mag_{rms})
    - Phase (Phs)
    - DC (DC)
  - [2] Harmonic Return
    - Harmonic order (n)
    - Frequency (f_n)
    - Magnitude ($Mag_{nom,n}$)
    - Phase ($Phs_{nom,n}$)
  - Reconstruction
    - .four uses SINE function for reconstruction (unlike .meas four)

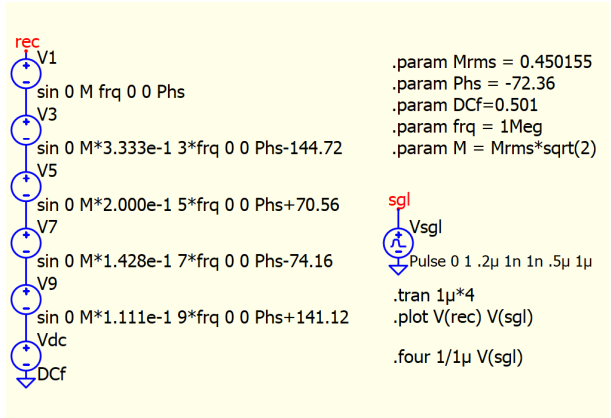
$$Signal = \sum_{n} \sqrt{2}\, Mag_{rms}\, Mag_{nom,n}\, sin\left(2\pi f_n T + Phs + Phs_{nom,n}\right)$$

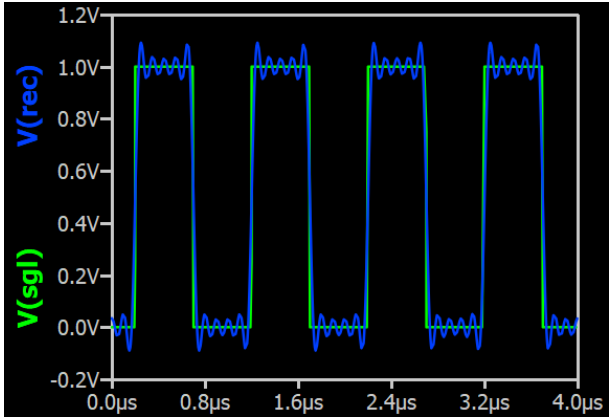



```
.four 1/1µ v(sgl):
 Fundamental: RMS Magnitude=0.450155 Phase=-72.36° DC=0.501
Harmonic  Frequency     Magnitude    Phase
   1      1.000e+06     1.000e+00     0.00°
   2      2.000e+06     3.142e-03    17.64°
   3      3.000e+06     3.333e-01  -144.72°
   4      4.000e+06     3.141e-03  -127.08°
   5      5.000e+06     2.000e-01    70.56°
   6      6.000e+06     3.141e-03    88.20°
   7      7.000e+06     1.428e-01   -74.16°
   8      8.000e+06     3.141e-03   -56.52°
   9      9.000e+06     1.111e-01   141.12°
THD = 42.8782%(48.1732%)
```

[1]

[2]

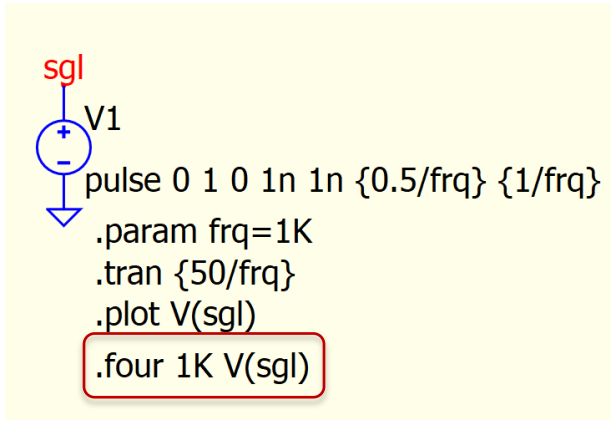




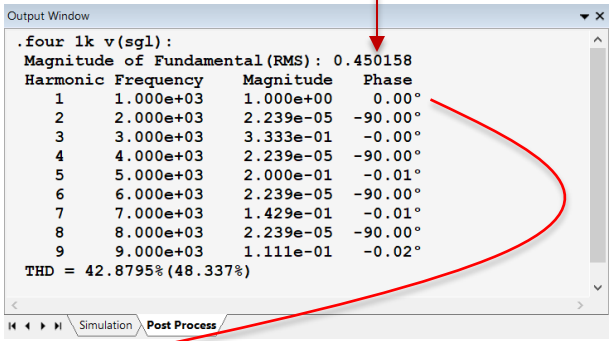
.four THD Computation

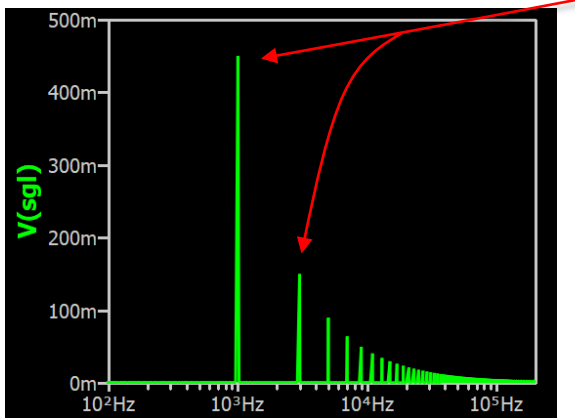
Qspice : .four - basic.qsch

- .four THD
  - [HARMONICS] and [PERIODS] is not necessary input parameters
  - Magnitude and phase are normalized component equal 1 and 0 degree
  - To compare FFT, FFT is to plot magnitude in RMS
    - Therefore, it requires to calculate with Magnitude * Magnitude of Fundamental (RMS) in .four to match calculation in FFT
    - In waveform viewer, right click, select FFT



rms of harmonic 1 (fundamental)



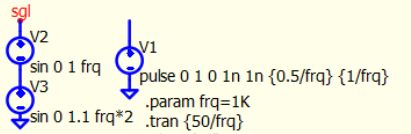


Calculate V(sgl),rms from .four into FFT chart
Harmonic 1 : Magnitude*RMS = 1.000e0*0.45 = 0.45
Harmonic 2 : Magnitude*RMS = 2.239e-5*0.45 = 0.00001
Harmonic 3 : Magnitude*RMS = 3.333e-1*0.45 = 0.15
:
:

.four THD Computation – THD formula

Qspice : .four - THD.qsch

- THD (total harmonic dist)
  - There are two THD (total harmonic distortion) definition
  - THD-F (Fundamental)
    - THD result in Qspice .four
    - Compares harmonic content to the fundamental frequency
    - Commonly used in power system analysis
    - $THD_F = \frac{\sqrt{V_2^2+V_3^2+V_4^4+\cdots}}{V_1}$
  - THD-R (Root Mean Square)
    - Comapres harmonic content to the total RMS value of the waveform
    - Commonly used in audio applications
    - $THD_R = \frac{\sqrt{V_2^2+V_3^2+V_4^4+\cdots}}{\sqrt{V_1^2+V_2^2+V_3^2+\cdots}}$




```
* user input
.param frq=1K ; replace 1K with your fundamental frequency
.func Signal() V(sgl) ; replace V(sgl) with your nodename

* calculate harmonic 1 to 9 with .four
.four frq V(sgl)

* calcuate THD-F and THD-R
* THD-F : ratio of total harmonic content to the fundamental freq
* THD-R : ratio of total harmonic content to RMS value of entire waveform
.meas THD-F param sqrt(SumH)/H1*100
.meas THD-R param sqrt(SumH/(SumH + H1^2))*100

* necessary formula in calculating THD-F and THD-R
.meas SumH param (H2^2+H3^2+H4^2+H5^2+H6^2+H7^2+H8^2+H9^2)
.meas G1 four frq*1 Signal()
.meas G2 four frq*2 Signal()
.meas G3 four frq*3 Signal()
.meas G4 four frq*4 Signal()
.meas G5 four frq*5 Signal()
.meas G6 four frq*6 Signal()
.meas G7 four frq*7 Signal()
.meas G8 four frq*8 Signal()
.meas G9 four frq*9 Signal()
.meas H1 param abs(G1)
.meas H2 param abs(G2)
.meas H3 param abs(G3)
.meas H4 param abs(G4)
.meas H5 param abs(G5)
.meas H6 param abs(G6)
.meas H7 param abs(G7)
.meas H8 param abs(G8)
.meas H9 param abs(G9)
```

In this example, 2nd harmonic amplitude is larger than fundamental amplitude
THD-F can return a result > 100%
THD-R must returns a result < 100%

```
.four frq v(sgl):
 Fundamental: RMS Magnitude=0.706494 Phase=0.000
 Harmonic  Frequency     Magnitude    Phase
    1      1.000e+03     1.000e+00     0.00°
    2      2.000e+03     1.098e+00    -0.01°
    3      3.000e+03     2.880e-04   -10.72°
    4      4.000e+03     7.225e-05    62.83°
    5      5.000e+03     2.819e-04   166.09°
    6      6.000e
    7      7.000e
    8      8.000e+03     7.367e-05    39.19
    9      9.000e+03     1.291e-04   148.70°
 THD = 109.839%(109.839%)
.meas thd-f param sqrt(sumh)/h1*100:
        110.381
.meas thd-r param sqrt(sumh/(sumh + h1^2))*100:
        74.1097
```

Qspice calculates THD-F

THD-F

THD-R

Compare Qspice and LTspice .four log

Qspice

```
.four 1k v(sgl):
 Fundamental: RMS Magnitude=0.450158 Phase=-0.00241709° DC=0.5
 Harmonic  Frequency     Magnitude    Phase
    1      1.000e+03     1.000e+00     0.00°
    2      2.000e+03     2.239e-05   -90.00°
    3      3.000e+03     3.333e-01    -0.00°
    4      4.000e+03     2.239e-05   -90.00°
    5      5.000e+03     2.000e-01    -0.01°
    6      6.000e+03     2.239e-05   -90.00°
    7      7.000e+03     1.429e-01    -0.01°
    8      8.000e+03     2.239e-05   -90.00°
    9      9.000e+03     1.111e-01    -0.02°
 THD = 42.8795%(48.337%)
```

Equivalent
(for order e-05, error between simulator can be expected)

LTspice

Fourier Component in LTspice .four is Peak
Qspice : Magnitude * RMS * sqrt(2) = 1*0.450158*sqrt(2) = 0.6366

```
Fourier components of V(sgl)
DC component:0.500001

Harmonic      Frequency      Fourier       Normalized    Phase       Normalized
 Number         [Hz]        Component      Component   [degree]    Phase [deg]
    1        1.000e+3       6.366e-1      1.000e+0      90.00°        0.00°
    2        2.000e+3       2.000e-6      3.142e-6       0.00°      -90.00°
    3        3.000e+3       2.122e-1      3.333e-1      90.00°        0.00°
    4        4.000e+3       2.000e-6      3.142e-6       0.00°      -90.00°
    5        5.000e+3       1.273e-1      2.000e-1      90.00°        0.00°
    6        6.000e+3       2.000e-6      3.142e-6       0.00°      -90.00°
    7        7.000e+3       9.095e-2      1.429e-1      90.00°        0.00°
    8        8.000e+3       2.000e-6      3.142e-6       0.00°      -90.00°
    9        9.000e+3       7.074e-2      1.111e-1      90.00°        0.00°
Partial Harmonic Distortion: 42.880402%
Total Harmonic Distortion:   48.342146%
```

.func
Function

.func – User-Defined Function

Qspice : .func - math ex1.qsch | .func - math ex2.qsch

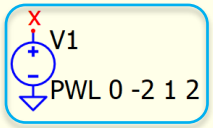
- .func
  - User-Defined Function
  - Mathematic formula can be implemented in multiple ways, here are two examples in .tran and .op
  - Two example to show how to plot formula
    - $y = ax^2 + bx + c$
  - In .tran example, a voltage source is used for input x
  - In .op example, use .step for input x
  - In both examples, .step is used to change coefficient b

Math formula implementation with .tran

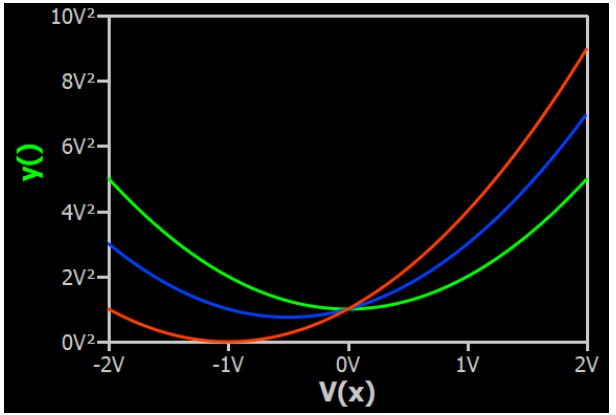
```
.param a=1
.param b=0
.param c=1
```

Change coefficient b

```
.step param b list 0 1 2
```




```
.func y() a*V(x)**2+b*V(x)+c    .plot y()
.option SaveParam

.tran 1
```



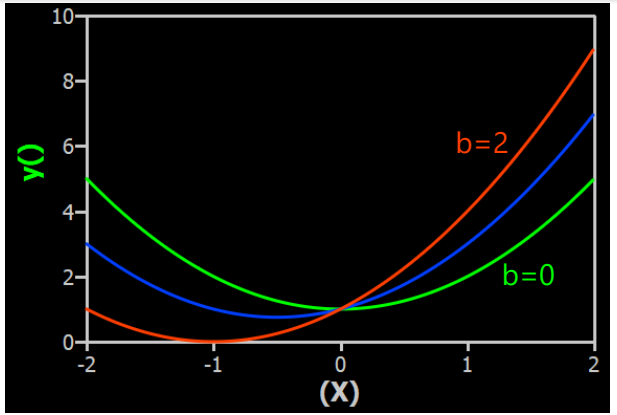
Math formula implementation with .op

```
.param a=1
.param b=0
.param c=1
.step param x -2 2 0.01 param b list 0 1 2
```

Change coefficient b

Need a dummy source for simulation can run

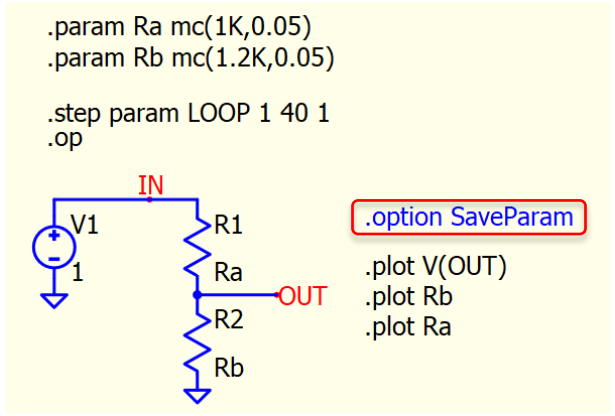
```
.func y() a*x**2+b*x+c    .plot y()
.option SaveParam
.op
```

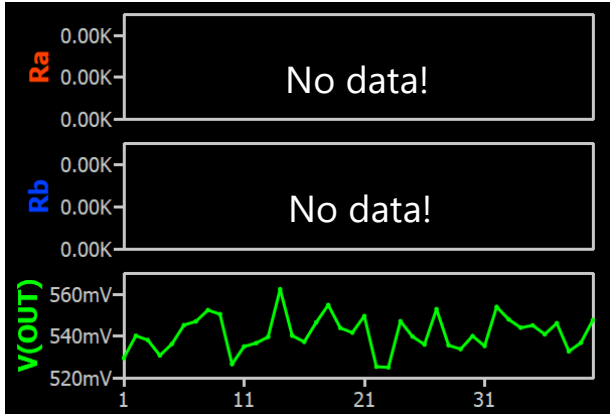


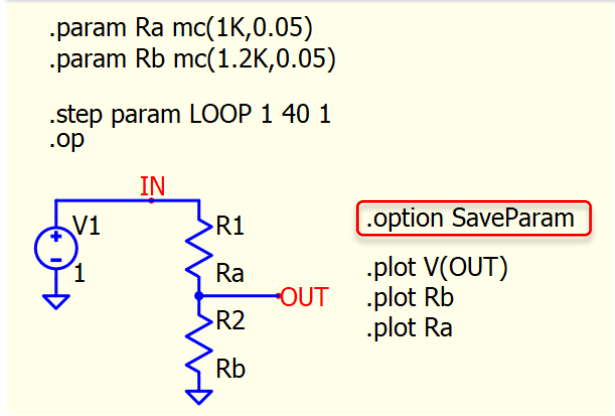
.func – User-Defined Function (with .option SaveParam)

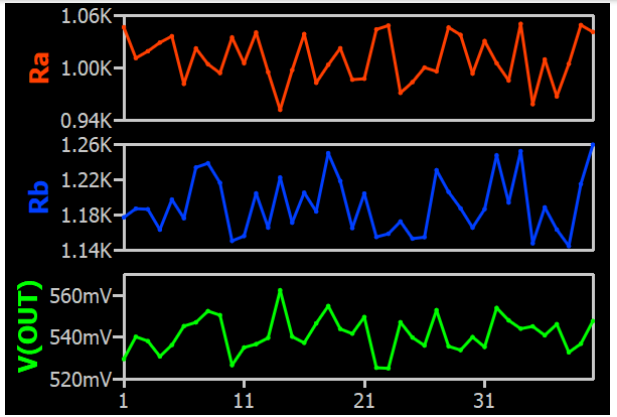
Qspice : .func with listparam.qsch

- .option SaveParam
  - Example to demonstrate .param with formula
  - In this Monte Carlo example, .param defines Ra and Rb as resistance that randomly change by 5% in each simulation step with the help of dummy .step command
  - However, without .option SaveParam, the randomly generated Ra and Rb for each step will not store in the data file as they are only parameters
  - By including .option SaveParam, their randomly generated values will be stored in the data file (.qraw) and can be selected to plot







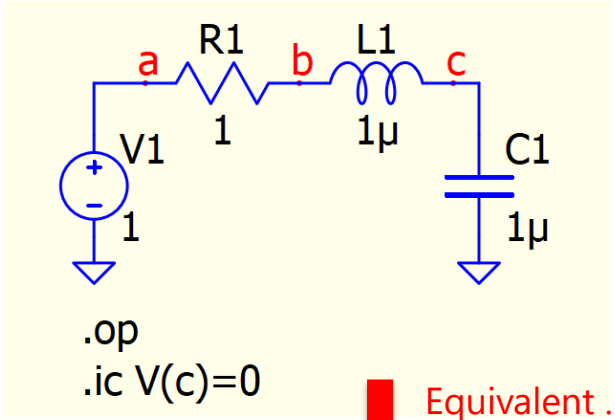


.ic Initial Conditon

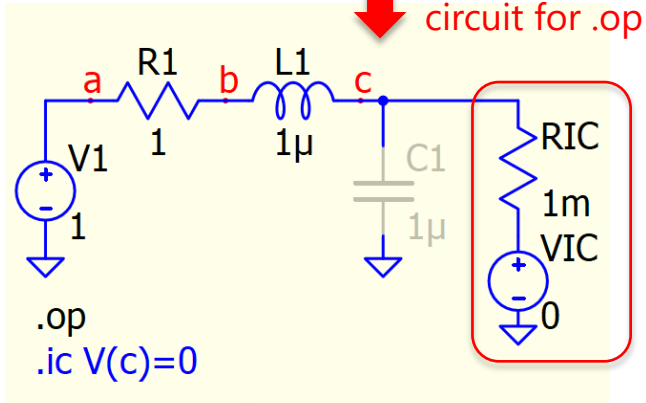
.ic Initial Condition – How Initial Condition works

Qspice : .ic - initial condition in .op (capacitor).qsch

- .ic Initial Condition
  - Initial node voltages and inductor currents of a transient analysis can be specified with .ic statements
- Concept of .ic in SPICE
  - Initial conditions are NOT assigned voltage/current at the node and inductor to compute the DC solution with circuit theory
  - Instead, initial conditions are set with non-zero impedance sources
  - In this example, I(L1) and I(C1) are different, which is NOT possible in circuit theory. However, as SPICE asserts an invisible source, there isn't any current in the capacitor



Equivalent .ic circuit for .op



	Operating Point
V(a)	1
V(b)	0.000999001
V(c)	0.000999001
I(V1)	-0.999001
I(L1)	0.999001
I(C1)	9.99001e-22
P(R1)	0.998003
P(V1)	-0.999001
P(L1)	1.58837e-29
I(R1)	0.999001
P(C1)	0

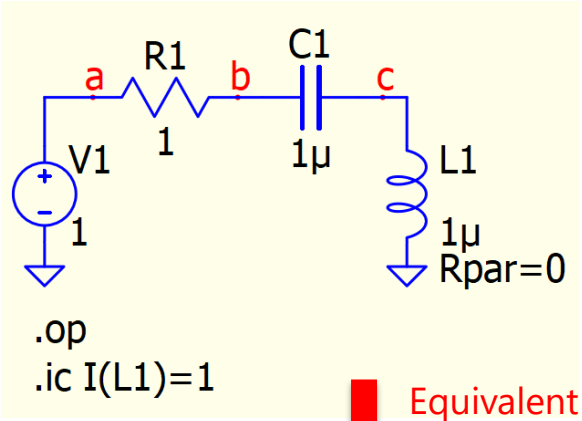
	Operating Point
V(a)	1
V(b)	0.000999001
V(c)	0.000999001
V(n01)	0
I(VIC)	0.999001
I(V1)	-0.999001
I(L1)	0.999001
P(RIC)	0.000998003
P(R1)	0.998003
P(V1)	-0.999001
P(L1)	1.58837e-29
I(RIC)	0.999001
I(R1)	0.999001
P(VIC)	0

Use .option RIC to change impedance of source asserting initial conditions

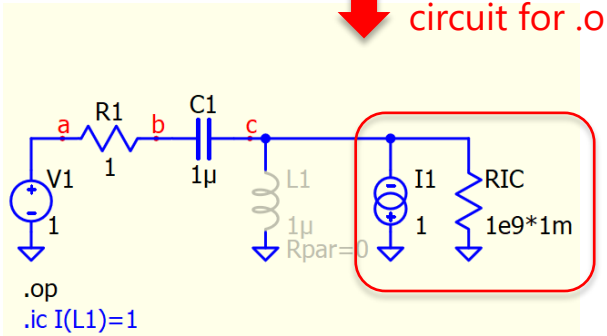
.ic Initial Condition – How Initial Condition works

Qspice : .ic - initial condition in .op (inductor).qsch

- .ic for inductor
  - Initial conditions are asserted by current source in parallel with impedance = 1e9*RIC



Equivalent .ic circuit for .op



Operating Point	
V(a)	1
V(b)	1
V(c)	-1e+06
I(V1)	-1e-12
I(L1)	1e-12
I(C1)	1e-12
P(R1)	1e-24
P(V1)	-1e-12
I(R1)	9.99978e-13
P(L1)	0
P(C1)	0

Operating Point	
V(a)	1
V(b)	1
V(c)	-1e+06
I(V1)	-1e-12
I(I1)	1
I(C1)	1e-12
P(I1)	-1e+06
P(RIC)	1e+06
P(R1)	1e-24
P(V1)	-1e-12
I(RIC)	-1
I(R1)	9.99978e-13
P(C1)	0

Use .option UIC to change impedance of source asserting initial conditions
Inductor currents are asserted with the compliance of 1e9 * RIC

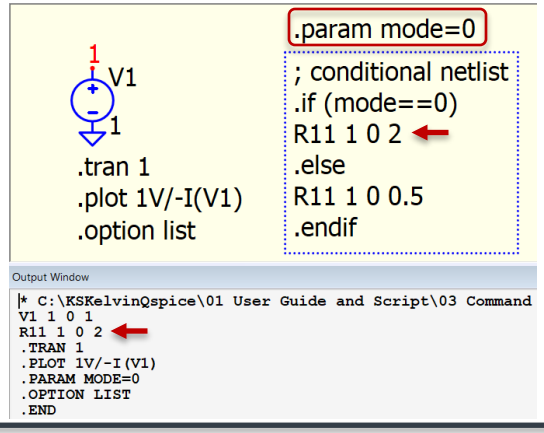
.if/.else/.endif

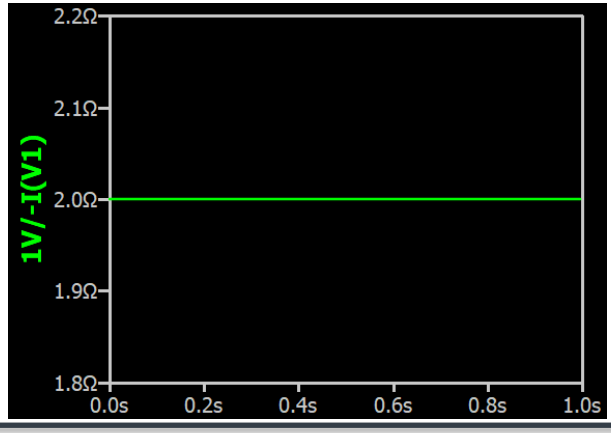
If conditional netlist
(Undocumented)

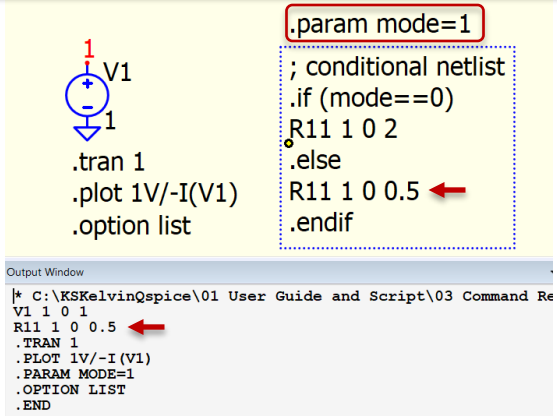
.if/.else/.endif – If conditional netlist processing

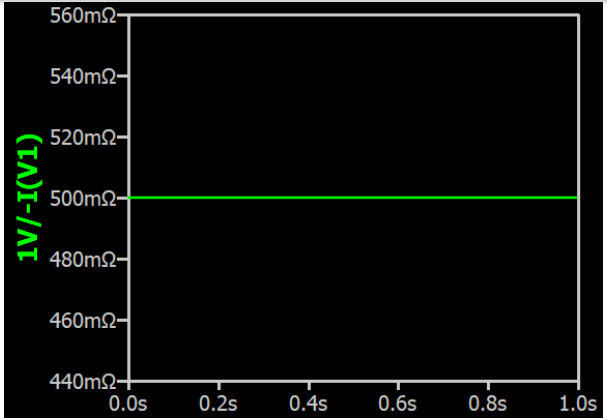
Qspice : .if - netlist device.qsch

- .if/.else/.endif
  - Conditional netlist processing
  - Limitation
    - .elseif not implemented
    - .step param not support as implementation is from the schematic level
    - Mike explained that this implementation is for in-house Qorvo engineers could replace Spectre with QSPICE for some foundry models





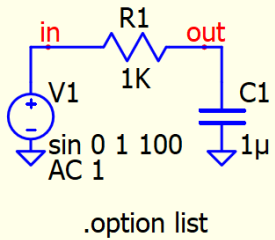




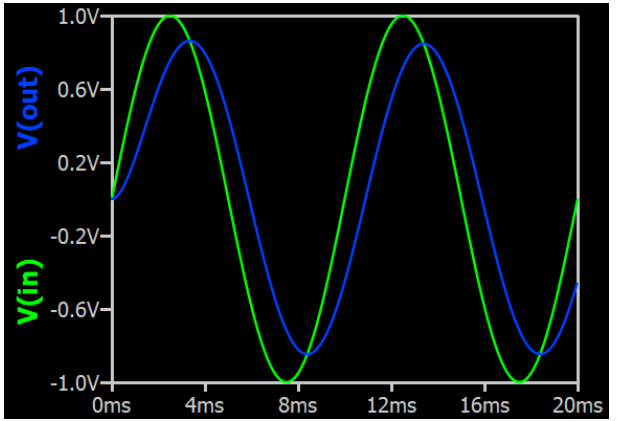
.if/.else/.endif – If conditional netlist processing

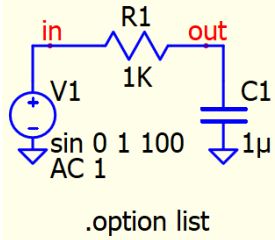
Qspice : .if - simulation directive.qsch

- .if/.else/.endif
  - An example application in controlling simulation directive
  - Another usage is to select .model to be used

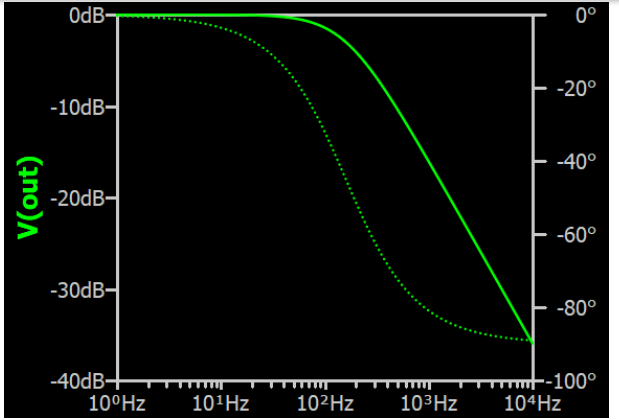



```
.param mode=0
; conditional netlist
.if (mode==0)
  .tran 2/100
  .plot V(in) V(out)
.else
  .ac dec 100 1 10K
  .plot V(out)
.endif
```






```
.param mode=1
; conditional netlist
.if (mode==0)
  .tran 2/100
  .plot V(in) V(out)
.else
  .ac dec 100 1 10K
  .plot V(out)
.endif
```

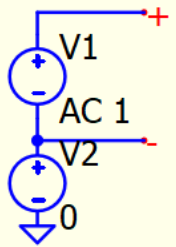


.if/.else/.endif – nested structure

Qspice : .if - nested.qsch

- Nested
  - .elseif is not supported, but nested structure is acceptable
  - This example is to change netlist element based on imaginary part X, where three possible condition may present

```
.param frq=10Meg
.param R=50
.param X=-50
Change netlist based on imaginary X
.if(X==0)
  R + - R
.else
  .if(X>0)
    L + - X/2/pi/frq Rser=R
  .else
    C + - 1/2/pi/frq/(-X) Rser=R
  .endif
.endif
```




```
.ac list 10Meg
.plot V(+,-)/-I(V1)
```

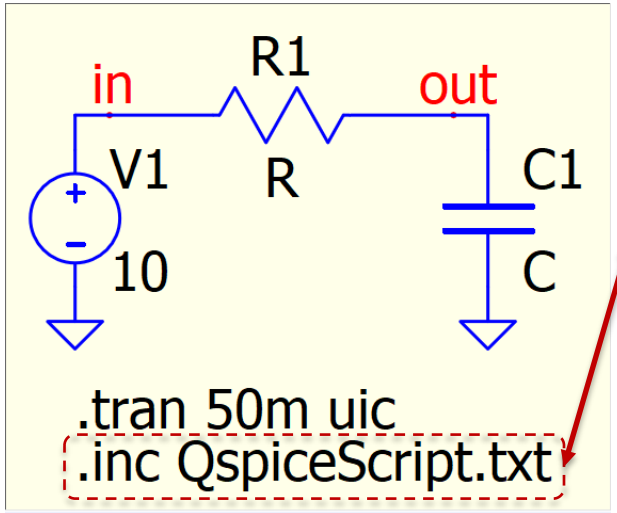
.inc
Include File

Include File (.inc) : HELP > Schematic Capture > Simulator > Include File (.inc)

Qspice : .inc - QspiceScript.qsch

- Include File
  - To include a file to execute by simulator
  - This allow to simplify schematic for directive or reuse purpose

- Example
  - This example use .inc to include a file called QspiceScript.txt
  - This script can
    - Input .param
    - Calculate .func
    - Define .plot
    - Calculate .meas

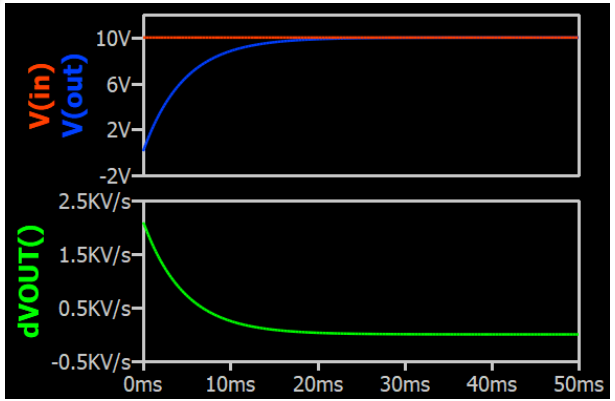


Output Window

```
.meas vo_pk max v(out):
        9.99976 (at Time=0.05)
.meas tau find time when v(out)=0.63*vo_pk:
     0.00467298 0.00467298
```

QspiceScript.txt

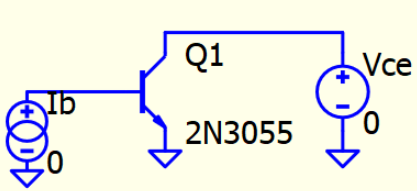
```
.param R=1K
.param C=4.7u
.func dVOUT() d(V(out))
.plot dVOUT()
.plot V(out) V(in)
.meas Vo_pk max V(out)
.meas tau find time when V(out)=0.63*Vo_pk
```



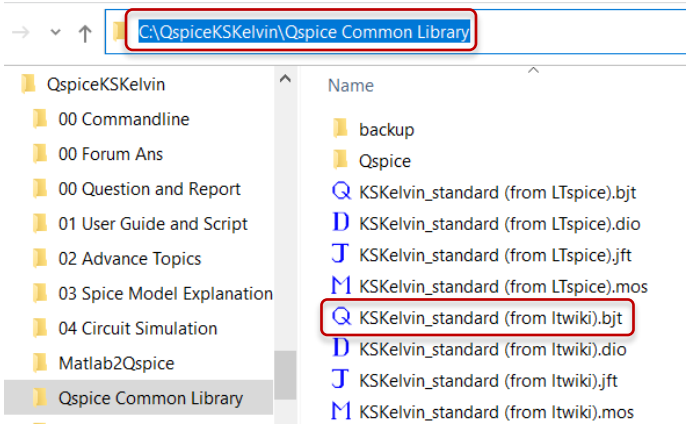
.libpath
Include a Directory into the Library File Search Path

.libpath (Include a Directory into the Library File Search Path)

- .libpath
  - Syntax : .libpath <directory>
  - Search path for library .lib and include .inc directive
  - Search path priority
    1. Absolute path in .lib and .include
    2. Current working directory
    3. .libpath directories
    4. Qspice installation directory
       - Normal Install Path : C:\Program Files\QSPICE
  - If .libpath is used, .lib or .inc must after .libpath in netlist
    - Therefore, recommend to use Ctrl-Enter method after .libpath to add .lib / .inc to ensure this sequence



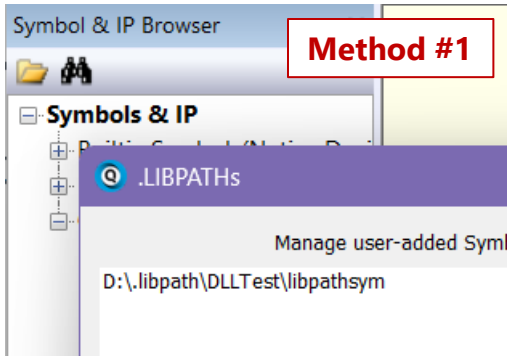

```
.dc Vce 0 10 0.01 Ib list 0.01 0.1 0.4 1 2
.libpath C:\QspiceKSKelvin\Qspice Common Library
.lib "KSKelvin_standard (from ltwiki).bjt"
.plot Ic(Q1)
```

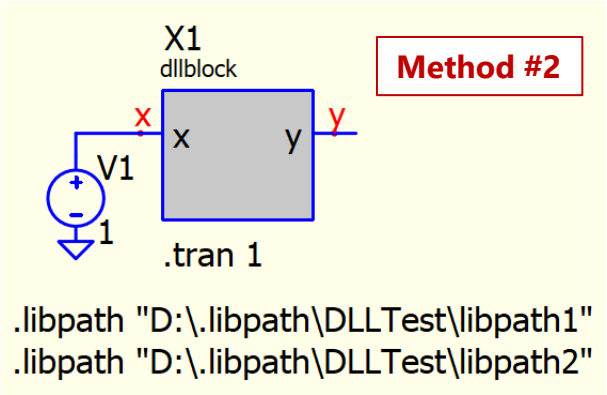


.libpath and DLL Priority (#1 : Understanding .libpath method)

Qspice : [folder – DLLTest] SingleLayer.dllblock.qsch

- .libpath
  - There are two methods to add .libpath, each with a different priority
  - Method #1 : In Symbol & IP Browser > Add a Symbol Directory
  - Method #2 : In the schematic, add .libpath <path>
  - In this example
    - The .libpath added with Method #1 is at the beginning of the netlist, while the .libpath added with Method #2 is placed at the end of the netlist
    - Multiple .libpath entries in Method #2 are arranged in accordance with their order of input in the netlist





** netlist is best way to identify .dll file search sequence (if multiple .dll filename)

```
* D:\.libpath\DLLTest\parent.dllblock.qsch

.libpath "D:\.libpath\DLLTest\libpathsym"

Ø´X1 «x´d» «y´d» «»  dllblock
V1 x 0 1
.tran 1
.libpath "D:\.libpath\DLLTest\libpath1"
.libpath "D:\.libpath\DLLTest\libpath2"
.end
```

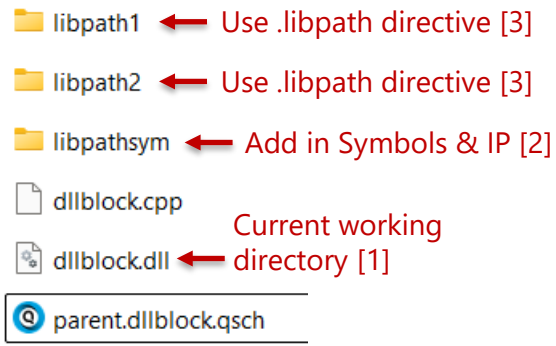
Method #1

Method #2

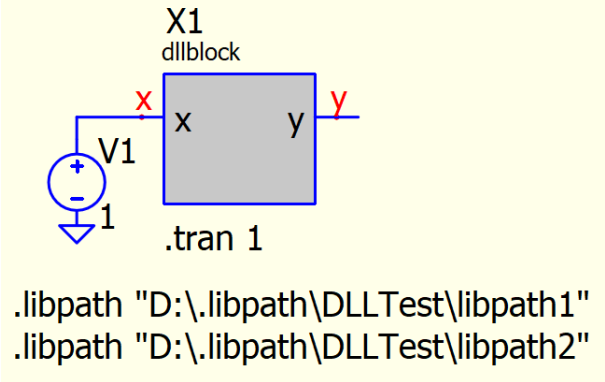
.libpath and DLL Priority (#2 : DLL Priority)

Qspice : [folder – DLLTest] SingleLayer.dllblock.qsch

- DLL Priority
  - A DLL block with its first attribute “dllblock”, represent it search for dllblock.dll to execute
  - A directory setup with all folders and its root with a file “dllblock.dll” includes, and to verify DLL loading priority
  - Priority as follows
    - [1] Current working directory
    - [2] Symbols & IP .libpath path (method #1)
    - [3] .libpath in schematic

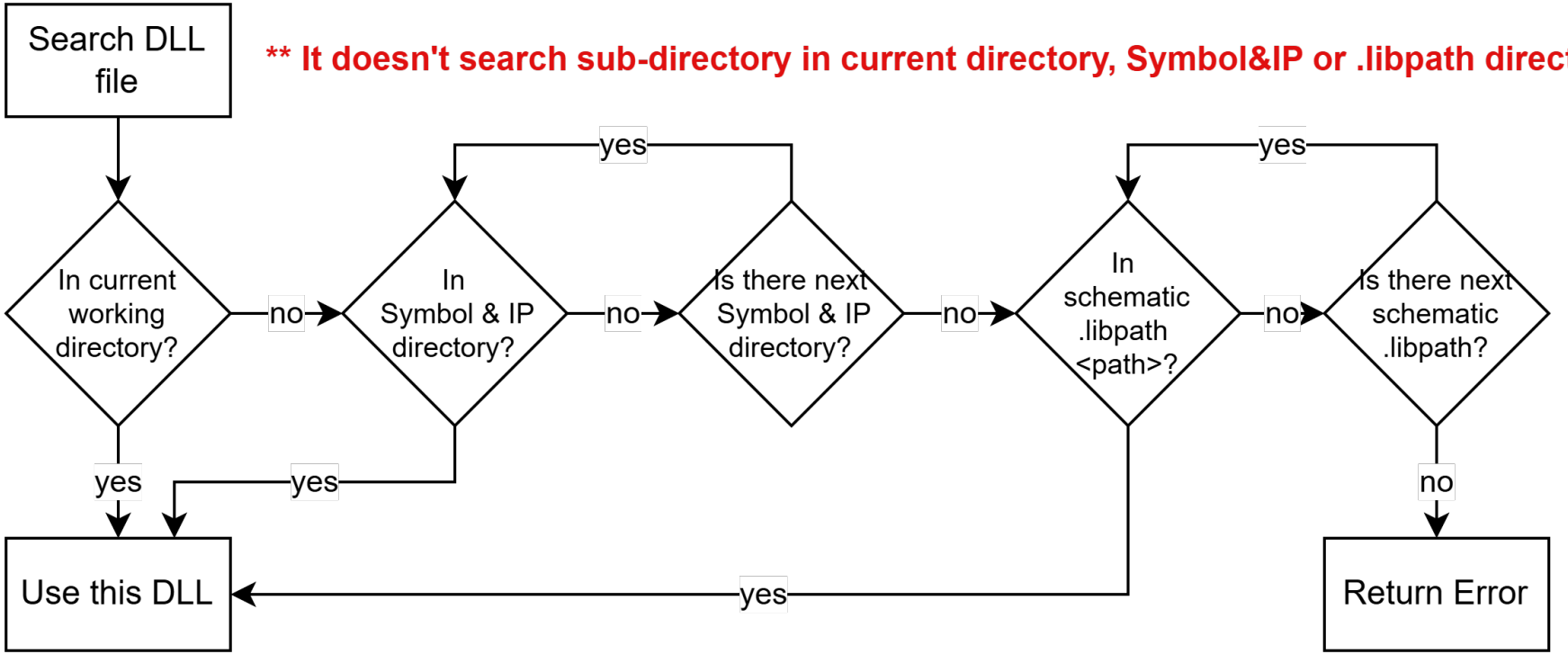



```
[1]* D:\.libpath\DLLTest\parent.dllblock.qsch
[2].libpath "D:\.libpath\DLLTest\libpathsym"

   Ø´X1 «x´d» «y´d» «»  dllblock
   V1 x 0 1
   .tran 1
[3.1].libpath "D:\.libpath\DLLTest\libpath1"
[3.2].libpath "D:\.libpath\DLLTest\libpath2"
   .end
```



.libpath and DLL Priority (#2 : DLL Priority – Flow Chart)

Qspice : [folder – DLLTest] SingleLayer.dllblock.qsch

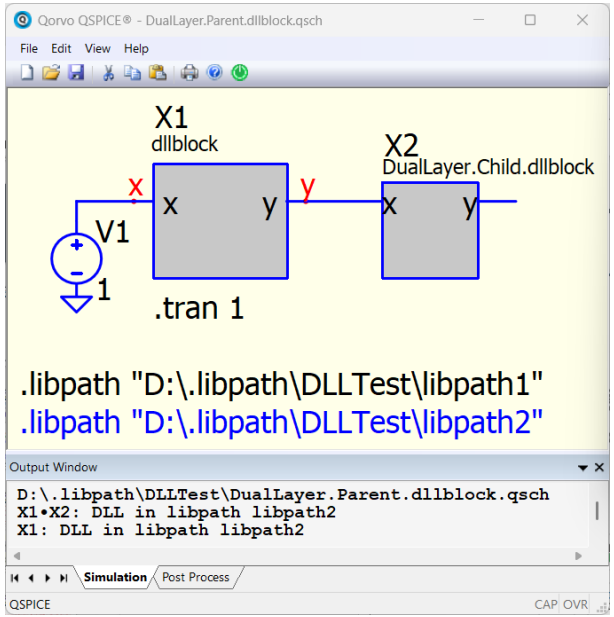


Special Case : Parent and Child

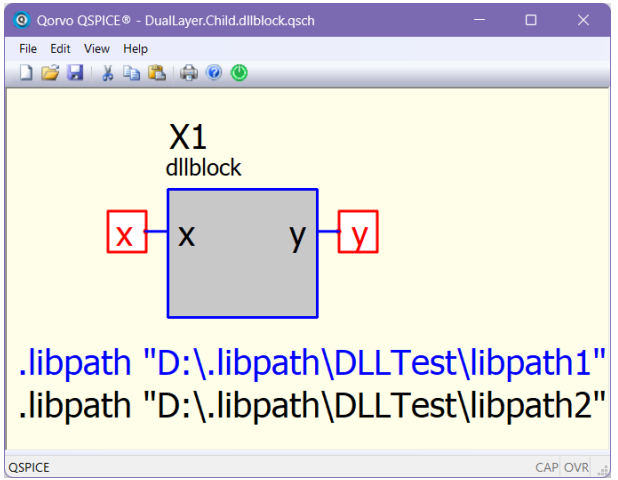
Qspice : [folder – DLLTest] DualLayer.Parent.dllblock.qsch | DualLayer.Child.dllblock.qsch

- Special Case
  - If both the Parent and Child schematics contain a DLL block with the same name
  - If the .dll file is located in the current directory or added in the Symbol & IP directory, that specific .dll file takes priority for both blocks
  - In the scenario where a .libpath is utilized in both the Parent and Child schematics, but they point to different paths, the .libpath specified in the Child schematic takes priority for both blocks

Parent Schematic



Child Schematic



.meas
Measure Statements

Part 1 of 3
General Knowledge

Syntax of Measure (.meas)

HELP > Simulator > Command Reference > Measure(.meas)

- Syntax
  .MEAS[URE] [AC|DC|OP|TRAN|TF|NOISE] <name>
  + [<FIND|DERIV|PARAM> <quantity>]
  + [WHEN <LHS=RHS> | AT=<expr>]]
  + [TD=<td>] [<RISE|FALL|CROSS>=[<count>|LAST]]

- Example of single point along the abscissa (Part 1)
  - .meas <NAME> find <QUANTITY> at <EXPR>
  - .meas <NAME> find <QUANTITY> when <LHS=RHS>
  - .meas <NAME> find <QUANTITY> when <LHS=RHS> td=5n cross=10
  - .meas <NAME> find <QUANTITY> when <LHS=RHS> cross=last
  - .meas <NAME> deriv <QUANTITY> at <EXPR>

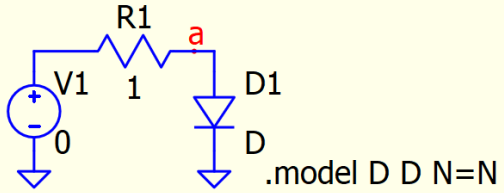
- Example of range over the abscissa (Part 1)
  - .meas <NAME> trig <LHS=RHS>
  - .meas <NAME> targ <LHS=RHS>
  - .meas <NAME> trig <LHS=RHS> targ <LHS=RHS>
  - .meas <NAME> trig <LHS=RHS> rise=1 targ <LHS=RHS> rise=11
  - .meas <NAME> avg|max|min|pp|rms|integ <QUANTITY>
  - .meas <NAME> avg|max|min|pp|rms|integ <QUANTITY> from <EXPR> [to <EXPR>]
  - .meas <NAME> avg|max|min|pp|rms|integ <QUANTITY> trig <LHS=RHS> targ <LHS=RHS>

- Example of fourier and FRA analysis (Part 2)
  - .meas <NAME> four FREQ <EXPR> [...]
  - .meas <NAME> fra FREQ <INPUT> <OUTPUT> [...]

.meas FIND with AT/WHEN (.dc directive)

Qspice : .meas - find (.dc).qsch

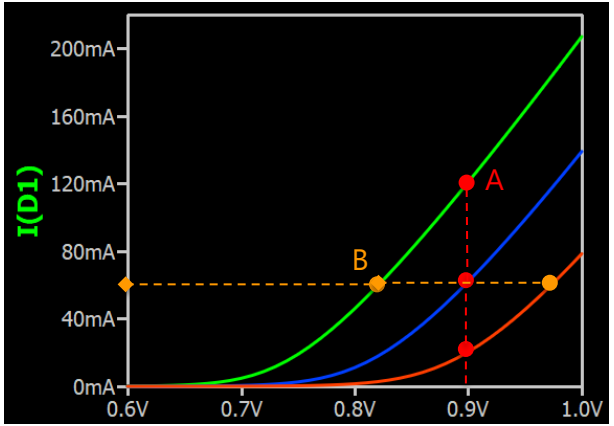
- .meas <NAME> find <QUANTITY> at <expression>
  .meas <NAME> find <QUANTITY> when <LHS=RHS>
  - Find a single point data along the abscissa, keyword **find** can be omitted in this sytax
  - **at** <expression> can be time or .step quantity
  - **when** <LHS>=<RHS> is defined as 1st condition meet this criteria, except with [rise/fall/cross]
  - Example
    - .meas A find I(D1) at 0.9 : A returns value of I(D1) at .dc sweep V1=0.9V
    - .meas B find V(a) when I(D1)=50m : B returns value of V(a) when I(D1)=50m condition is met




```
.dc V1 0.6 1 0.01
.step param N list 1 1.1 1.2
```

```
.meas A find I(D1) at 0.9
.meas B find V(a) when I(D1)=50m
```

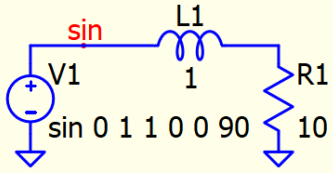
```
.meas a find i(d1) at 0.9:
   0    0.120867          0.9
   1   0.0619638          0.9
   2   0.0203455          0.9
.meas b find v(a) when i(d1)=50m:
   0    0.756249    0.806249
   1    0.831898    0.881898
   2    0.907518    0.957518
```



.meas FIND with AT/WHEN (.tran directive)

Qspice : .meas - find (.tran) - 01 basic.qsch

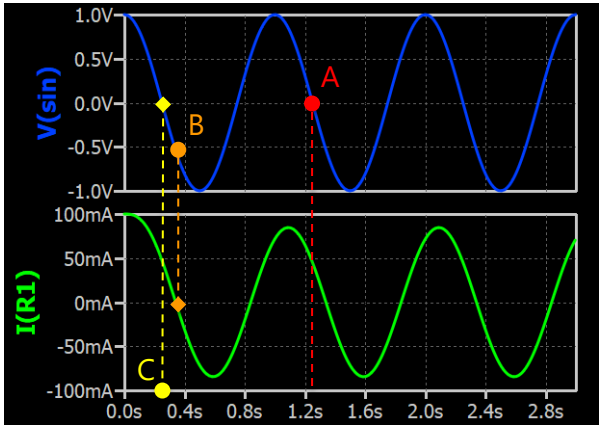
- .meas <NAME> find <QUANTITY> at <expression>
  .meas <NAME> find <QUANTITY> when <LHS=RHS>
  - Find a single point data along the abscissa, keyword **find** can be omitted in this sytax
  - **at** <expression> can be time or .step quantity
  - **when** <LHS>=<RHS> is defined as 1st condition meet this criteria, except with [rise/fall/cross]
  - Example
    - .meas A find V(sin) at 1.25 : A returns value of V(sin) at 1.25s
    - .meas B find V(sin) when I(R1)=0 : B returns value of V(sin) when I(R1)=0 condition is met
    - .meas C1 find time when V(sin)=0 : C1 returns time when V(sin)=0
    - .meas C2 find V(sin)*I(R1) when V(sin)=0 : C2 returns value of V(sin)*I(R1) when V(sin)=0




```
.tran 3   .plot I(R1)
          .plot V(sin)
```

```
.meas A find V(sin) at 1.25
.meas B find V(sin) when I(R1)=0
.meas C1 find time when V(sin)=0
.meas C2 find V(sin)*I(R1) when V(sin)=0
```

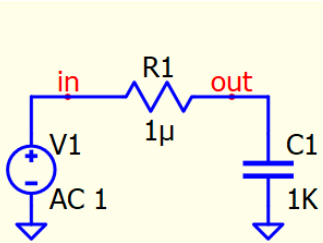
```
.meas a find v(sin) at 1.25:
     7.9986e-08       1.25
.meas b find v(sin) when i(r1)=0:
      -0.541337   0.341042
.meas c1 find time when v(sin)=0:
            0.25       0.25
.meas c2 find v(sin)*i(r1) when v(sin)=0:
                0       0.25
```

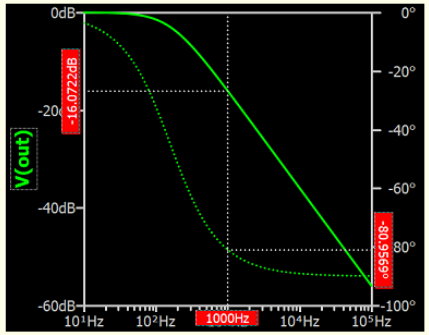


.meas FIND with WHEN (.ac directive) – complex number results

Qspice : .meas - find (.ac).qsch

- .meas results in .ac directive
  - Result is complex number : (real, imag)






```
.ac dec 100 10 100K
.plot V(out)
```

```
.meas v_complex find V(out) when frequency=1000
.meas v_dB_complex find dB(V(out)) when frequency=1000
.meas v_mag find abs(V(out)) when frequency=1000
.meas v_mag_dB dB(abs(V(out))) when frequency=1000
.meas v_phase phase(V(out)) when frequency=1000
```

$V(out) = 0.0247045 - j0.155223$

$20\log_{10} V(out) = -16.0722 - j12.2729$
** it log a complex number!

$|V(out)| = 0.157177\ (in\ Volt)$

$20\log_{10}|V(out)| = -16.0722\ (in\ dB)$

$\angle V(out) = -80.9596°$

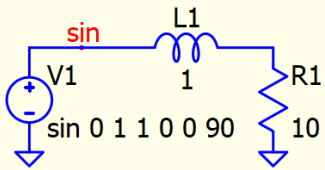
Output Window

```
.meas v_complex find v(out) when frequency=1000:
     ( 0.0247045, -0.155223)
.meas v_db_complex find db(v(out)) when frequency=1000:
     (  -16.0722,  -12.2729)
.meas v_mag find abs(v(out)) when frequency=1000:
     (  0.157177,          0)
.meas v_mag_db db(abs(v(out))) when frequency=1000:
     (  -16.0722,          0)
.meas v_phase phase(v(out)) when frequency=1000:
     (  -80.9569,          0)
```

.meas with keyword RISE, FALL, CROSS and TD in when, trig and targ

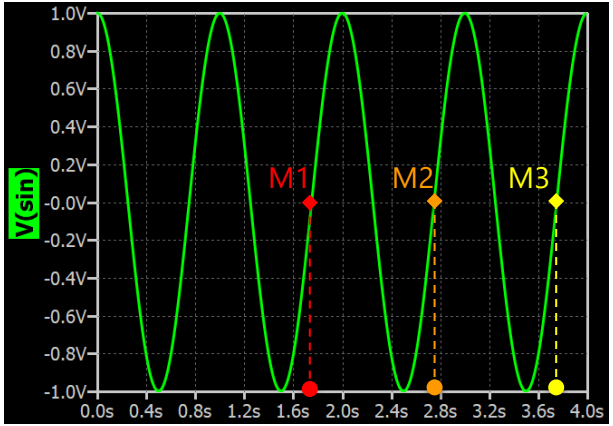
Qspice : .meas - find (.tran) - 02 rise fall cross td.qsch

- .meas with keyword RISE, FALL, CROSS and TD
  - Rise, Fall, Cross and Td can be used in when, trig and targ
  - With <LHS>=<RHS> condition, [rise,fall,cross] can be used to determine when to count this condition as a find event
  - TD is delayed trigger of event action
  - Example
    - .meas M1 find time when V(sin)=0 rise=2 : M1 is time for 2nd count of rising of V(sin)=0
    - .meas M2 find time when V(sin)=0 rise=2 td=1 : M2 is time for 2nd count of rising of V(sin)=0 after t=1s
    - .meas M3 find time when V(sin)=0 cross=last : M3 is time for last count of crossing of V(sin)=0




```
.tran 3
.plot I(R1)
.plot V(sin)
.meas M1 find time when V(sin)=0 rise=2
.meas M2 find time when V(sin)=0 rise=2 td=1
.meas M3 find time when V(sin)=0 cross=last
```

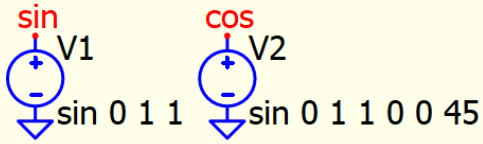
```
.meas m1 find time when v(sin)=0 rise=2:
                1.75       1.75
.meas m2 find time when v(sin)=0 rise=2 td=1:
                2.75       2.75
.meas m3 find time when v(sin)=0 cross=last:
                3.75       3.75
```



.meas DERIV with AT/WHEN

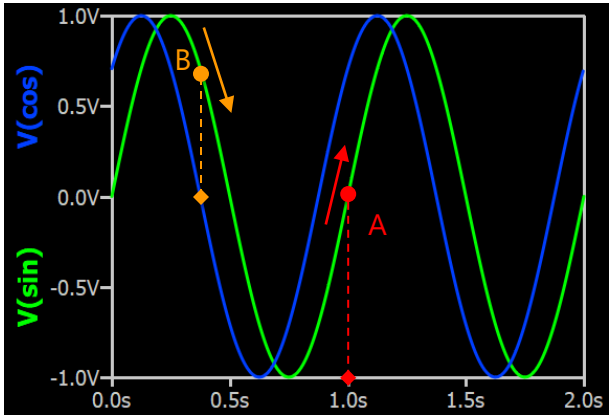
Qspice : .meas - deriv (.tran).qsch

- .meas <NAME> deriv <QUANTITY> at <expression>
  .meas <NAME> deriv <QUANTITY> when <LHS=RHS>
  - Derivative of <QUANTITY> of a single point data along the abscissa
  - Usage is identical as find, but return derivative of data point instead of its value




```
.tran 2
.plot V(sin) V(cos)

.meas A deriv V(sin) at 1
.meas B deriv V(sin) when V(cos)=0
```

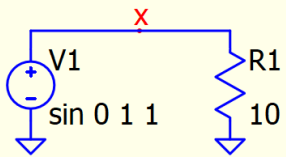
```
.meas a deriv v(sin) at 1:
         6.28308           1
.meas b deriv v(sin) when v(cos)=0:
        -4.44281       0.375
```



.meas PARAM

Qspice : .meas - param.qsch

- .meas <NAME> param <QUANTITY>
  - Returns a value of other .meas statement results
    - Not intended that <QUANTITY> is based on direct simulation data, if direct simulation data is taken, last simulated point is returned
  - Example
    - .meas P-Watts param Vrms*Irms : P-Watts is value of multiple from two .meas results
    - .meas M2 V(x) : Incomplete syntax, direct simulation data V(x) is taken and return last simulated point where time = 4.25s
    - .meas M3 param V(x) : Complete syntax as M2 condition, return only last simulated point without time



.tran 4.25

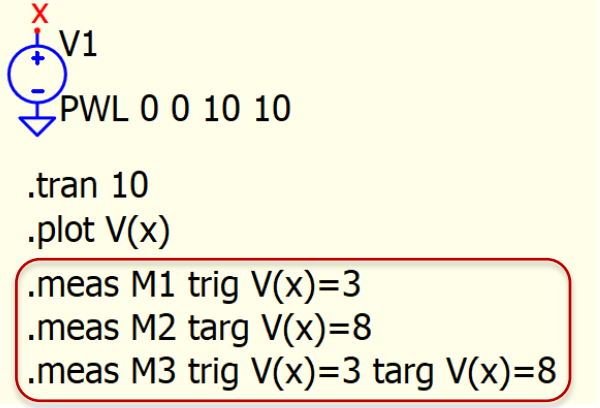
```
.meas Vrms rms V(x)
.meas Irms rms I(R1)
.meas P-Watts param Vrms*Irms
.meas M2 V(x)
.meas M3 param V(x)
```

```
.meas vrms rms v(x):
       0.707107
.meas irms rms i(r1):
      0.0707107
.meas p-watts param vrms*irms:
            0.05
.meas m2 v(x):
                 1         4.25
.meas m3 param v(x):
                 1
```

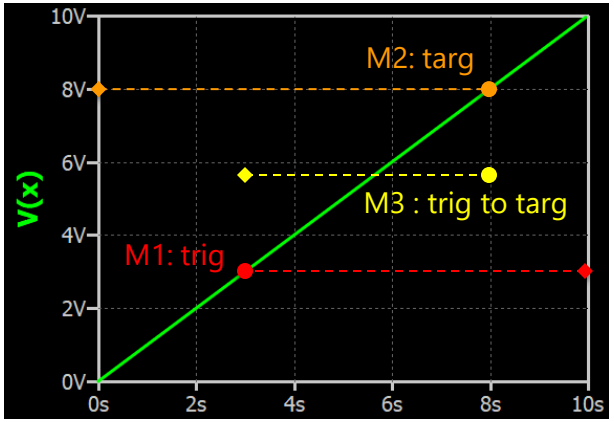
.meas TRIG and TARG

Qspice : .meas - trig targ.qsch

- .meas <NAME> trig <LHS=RHS>
  .meas <NAME> targ <LHS=RHS>
  .meas <NAME> trig <LHS1=RHS1> targ <LHS2=RHS2>
  - The range over the abscissa is specified with the points defined by "TRIG" and "TARG"
  - TRIG measures from condition<LHS=RHS> to the end of simulation data
  - TARG measures from condition <LHS=RHS> to the start of simulation data
  - TRIG… TARG measure distance between condition <LHS1=RHS1> and <LHS2=RHS2>
  - TRIG, TARG also support [<RISE|FALL|CROSS>=[<count>|LAST]]



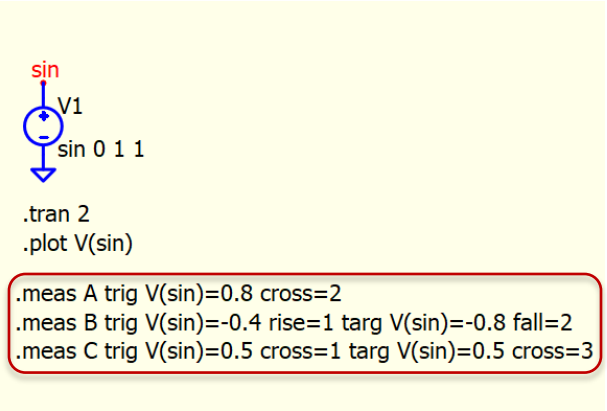

```
.meas m1 trig v(x)=3:
                  7
.meas m2 targ v(x)=8:
                  8
.meas m3 trig v(x)=3 targ v(x)=8:
                  5
```

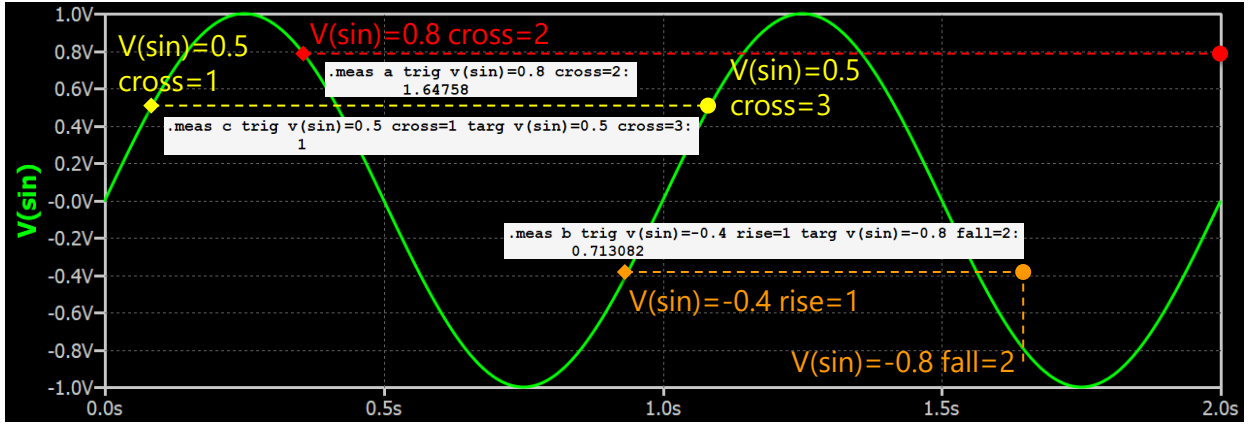


.meas TRIG and TARG : with RISE, FALL, CROSS

Qspice : .meas - trig targ - rise fall cross.qsch

- .meas with keyword RISE, FALL, CROSS and TD
  - Rise, Fall, Cross and Td can be used in when, trig and targ
  - With <LHS>=<RHS> condition, [rise,fall,cross] can be used to determine when to count this condition as a find event
  - TD is delayed trigger of event action
  - Example
    - .meas A trig V(sin)=0.8 cross=2 : A is time of 2nd crossing of V(sin)=0.8 to last data
    - .meas B trig V(sin)=-0.4 rise=1 targ V(sin)=-0.8 fall=2 : B is time of 1st rising of V(sin)=-0.4 to 2nd falling of V(sin)=-0.8
    - .meas C trig V(sin)=0.5 cross=1 targ V(sin)=0.5 cross=3 : C is time between 1st and 3rd crossing of V(sin)=0.5

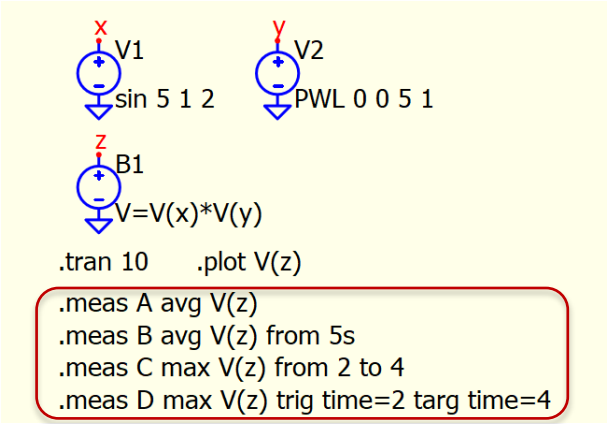




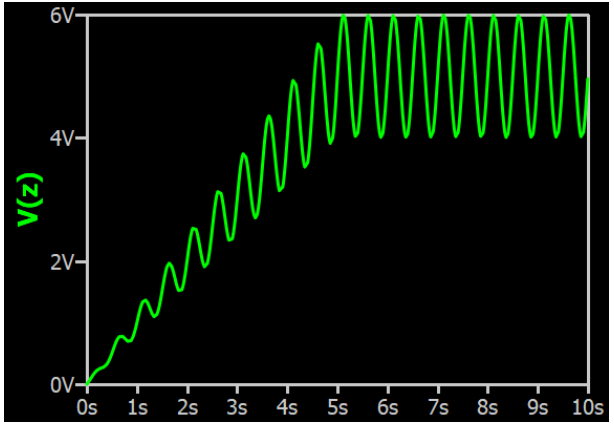
.meas AVG|MAX|MIN|PP|RMS|INTEG

Qspice : .meas - operation.qsch

- .meas <NAME> avg|max|min|pp|rms|integ <QUANTITY>
  .meas <NAME> avg|max|min|pp|rms|integ <QUANTITY> from <EXPR> [to <EXPR>]
  .meas <NAME> avg|max|min|pp|rms|integ <QUANTITY> trig <LHS=RHS> targ <LHS=RHS>
  - Measurement operation over entire data or an interval (with FROM / TO or TRIG / TARG)
  - Avg : Average | Max : Maximum | Min : Minimum
  - PP : Peak to Peak | RMS : Root Mean Square | Integ : Integral
  - Example
    - .meas A avg V(z) : A is average of V(z) over entire range of data
    - .meas B avg V(z) from 5s : B is average of V(z) from 5s to last data
    - .meas C max V(z) from 2 to 4 : C is maximum of V(z) between 2s to 4s
    - .meas D max V(z) trig time=2 targ time=4 : D is maximum of V(z) between 2s to 4s



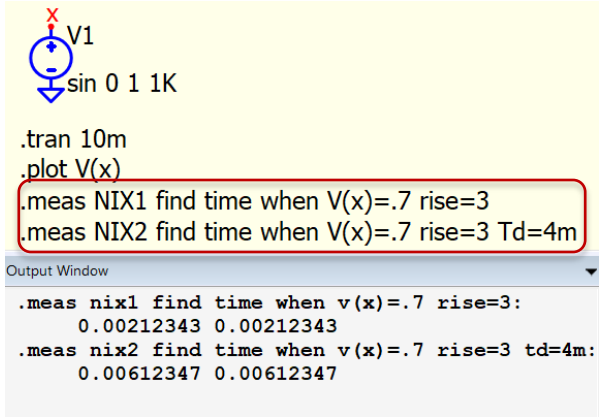

```
.meas a1 avg v(z):
        3.74205
.meas a2 avg v(z) from 5s:
           5
.meas a3 max v(z) from 2 to 4:
        4.35627 (at Time=3.63615)
.meas b max v(z) trig time=2 targ time=4:
        4.35627 (at Time=3.63615)
```

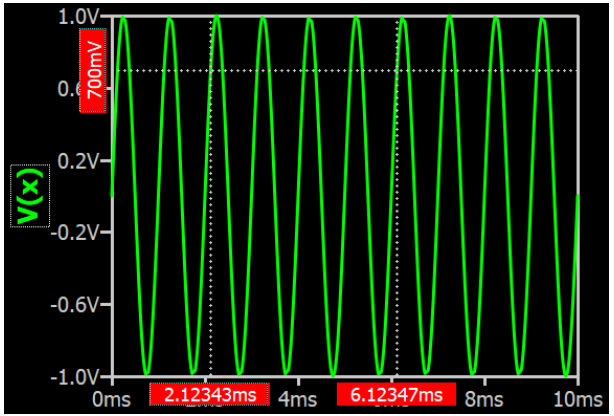


.meas TD – Delay Trigger

Qspice : .meas - Td.qsch

- .meas with TD
  - TD is delay in measurement
  - In this example
    - NIX1 is the time for 3rd cross (rising direction) counts of V(x) = .7
    - NIX2 is the time for 3rd cross (rising direction) counts of V(x) = .7 after time=4ms

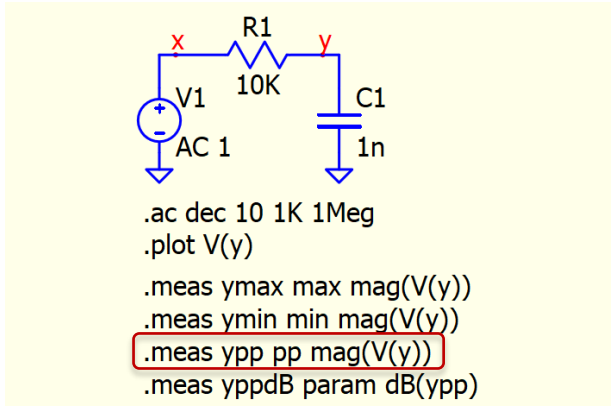




.meas PP in .ac (.ac directive) – Ratio of Amplitude (Special Case)

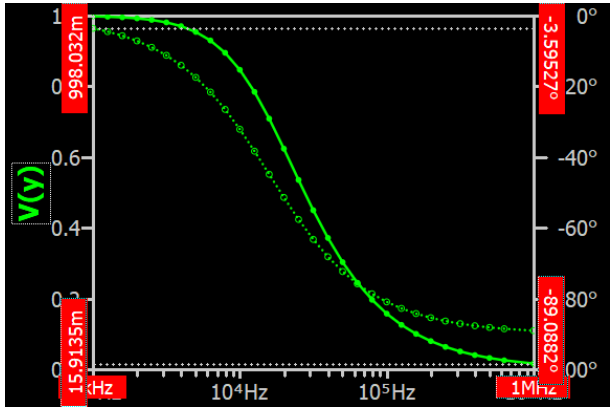
Qspice : .meas - pp (.ac).qsch

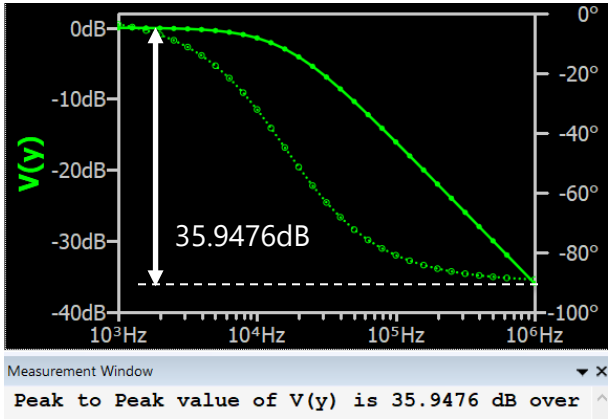
- .meas : pp in .ac
  - PP measures
    - In .tran : Peak-to-Peak
    - In .ac : Ratio
  - In this example
    - V(y) max and min magnitude are 0.998032 and 0.0159135
    - .meas PP gives ratio of the magnitude $= \frac{0.998032}{0.0159135} = 62.7161 = 35.9476\text{dB}$
    - This is peak-to-peak gain in ratio OR gain difference in dB



Output Window

```
.meas ymax max mag(v(y)):
     (  0.998032,          0)
.meas ymin min mag(v(y)):
     ( 0.0159135,          0)
.meas ypp pp mag(v(y)):
     (   62.7161,          0)
.meas yppdb param db(ypp):
     (   35.9476,          0)
```





Measurement Window

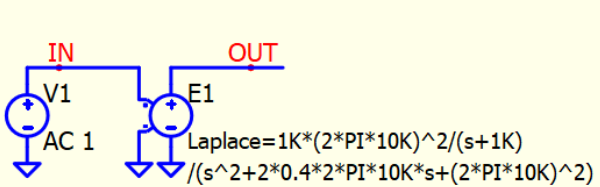
```
Peak to Peak value of V(y) is 35.9476 dB over
```

.meas dB() function (.ac directive)

Qspice : .meas - dB (.ac).qsch

- dB()
  - .meas supports dB()
  - For .ac analysis, results is in complex number (re,im) format
  - dB() returns result in complex number format in .ac
  - Assume complex number $v = v_r + jv_i$
  - $\log_{10} v = \log_{10}(v_r + jv_i) = \log_{10}\left(\sqrt{v_r^2 + v_i^2}\right) + j\,\frac{atan2(v_i, v_r)}{\ln(10)}$
  - where $|v| = v_r^2 + v_i^2$ and $\angle v = \tan^{-1}\left(\frac{v_i}{v_r}\right)$
  - dB()
    - Real = $20\log_{10}|v|$
    - Imag =$20\,\frac{\angle v}{\ln(10)}$

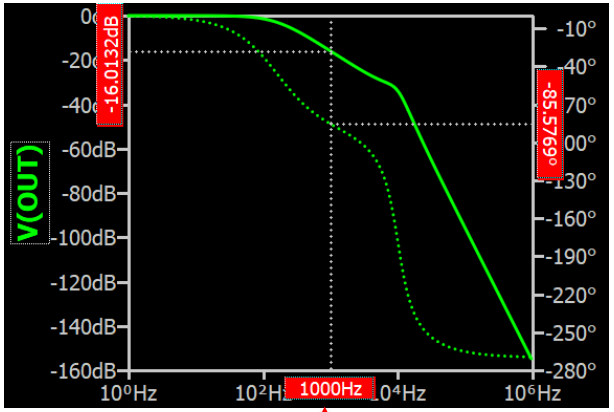
      → $\angle v = \frac{\mathrm{dB}_{\mathrm{imag}}\ln(10)}{20}$




```
.ac dec 100 1 1Meg
.plot V(OUT)

.meas A find V(OUT) at 1K
.meas B find dB(V(OUT)) at 1K
.meas C find dB(mag(V(OUT))) at 1K
```

```
.meas a find v(out) at 1k:
     ( 0.0122044, -0.157777)
.meas b find db(v(out)) at 1k:
     (  -16.0132,  -12.9732)
.meas c find db(mag(v(out))) at 1k:
     (  -16.0132,         0)
```

To prevent confusion in imaginary
Use dB(mag()) instead



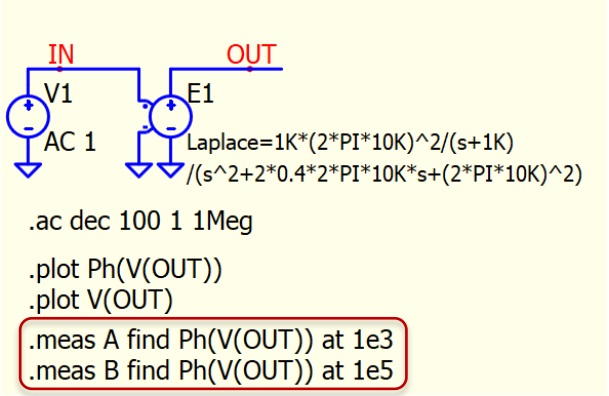
V(out) in R+jX format, or in polar
= -16.0132dB ∠ -85.5769°

Real part = magnitude in dB
Imag part
$\angle V(out) = -\frac{12.9732\ln(10)}{20} = 1.4936$ radian
∠V(out)=-85.5769°

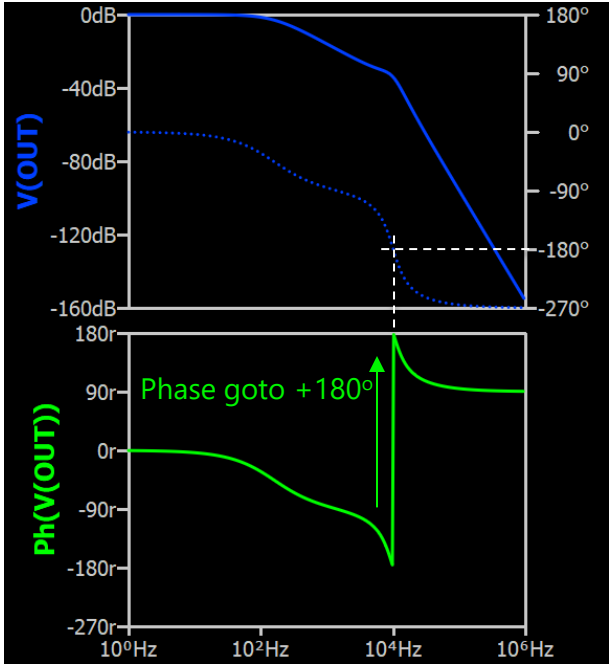
.meas Phase() | Ph() function (.ac directive)

Qspice : .meas - Phase (.ac).qsch

- Phase() | Ph()
  - .meas supports Phase() | Ph() functions
  - Ph() returns angle within (-180,180), but waveform viewer can display unwrap angle in polar plot
  - It is important that when using Ph() in .meas, it should compare to a value between (-180,180)



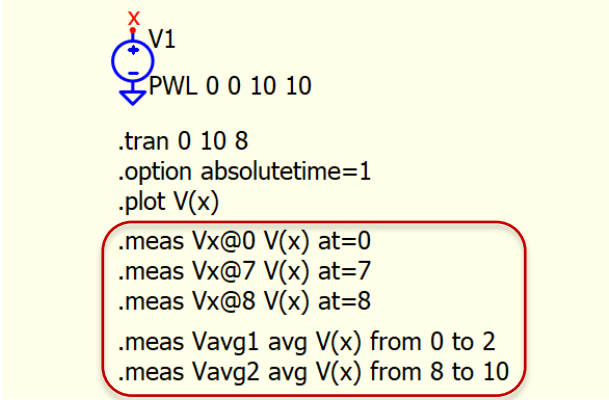

```
.meas a find ph(v(out)) at 1e3:
     (  -85.5769,          0)
.meas b find ph(v(out)) at 1e5:
     (   94.7111,          0)
```



.meas AT | FROM...TO in .tran with Tstart and absolutetime=1

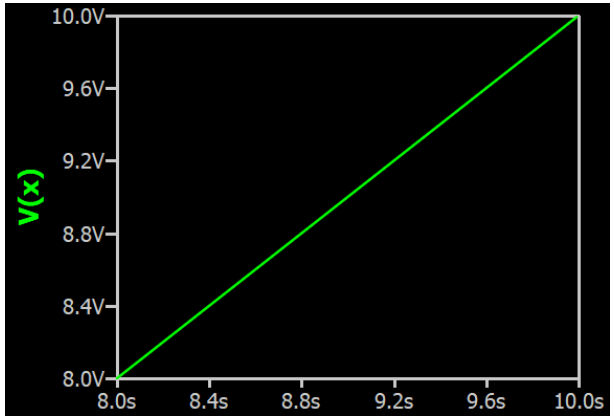
Qspice : .meas - at from (.tran Tstart).qsch

- .meas with Tstart
  - If **.tran with Tstart** and **.option absolutetime=1**, data storage will begin from Tstart instead of 0s
  - Time in .meas is absolute time
    - In this example, .tran only stores data from 8s to 10s
    - .meas at 0s, 7s will give "nan"



Output Window

```
.meas vx@0 v(x) at=0:
     (       nan,        nan)
.meas vx@7 v(x) at=7:
     (       nan,        nan)
.meas vx@8 v(x) at=8:
                8           8
.meas vavg1 avg v(x) from 0 to 7:
          8.00063
.meas vavg2 avg v(x) from 8 to 10:
                9
```



Plot .meas data in Waveform Viewer

Qspice : .meas - waveform viewer.qsch

Method #1 :

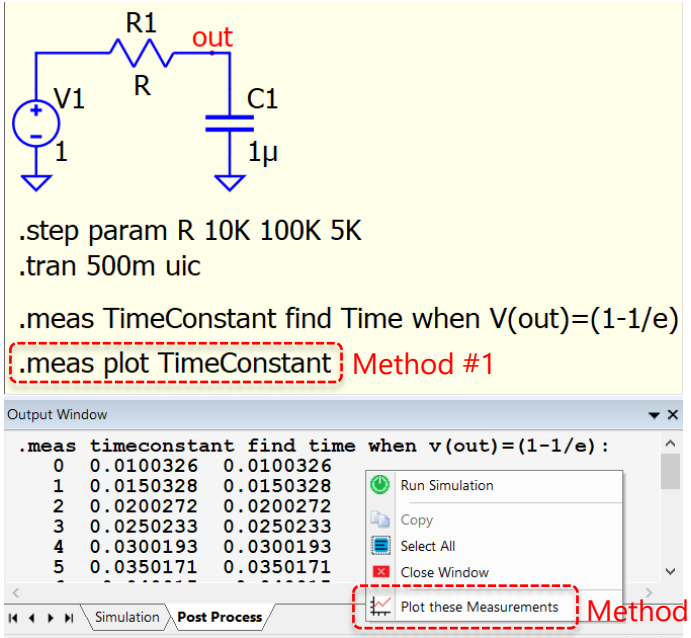
[1] Add ".meas plot [Name]"

Method #2 :

[1] Run Simulation

[2] In Output Window

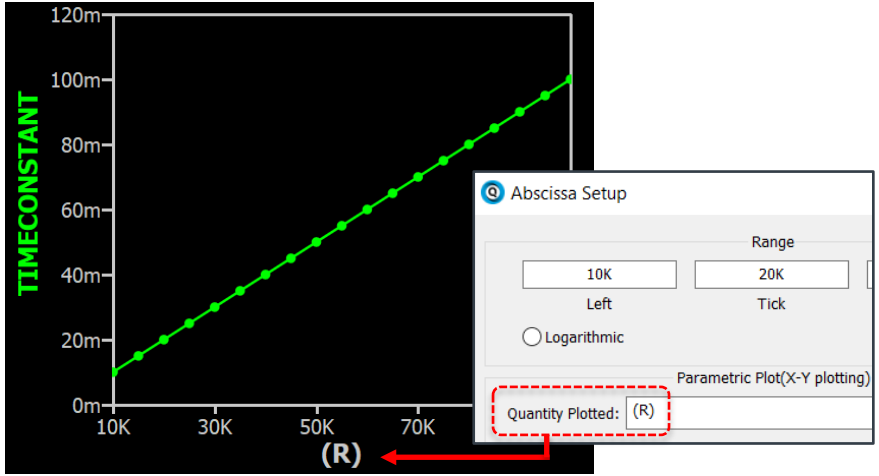
Right click in Post Process > Plot these Measurements



[3] X-axis default is .step parameter

[4] If you want to display X-axis parameter name

Right click x-axis > add bracket (or curly bracket) to parameter



.meas
Measure Statements

Part 2 of 3
FOUR Fourier
FRA Freq Response

.meas FOUR (fourier component) in .tran directive

Qspice : .meas - four (.tran).qsch

```
.func rms2pk(in) in*sqrt(2)

sgl
V1
sin 0 rms2pk(1) 1K
V2
sin 0 rms2pk(0.6) 3K
V3
sin 0 rms2pk(0.2) 5K

.tran 5/1K
.plot V(sgl)
```

THD Total Harmonic Distortion
.four 1K V(sgl)
Fourier component with .meas
.meas xx four 1K V(sgl)
.meas |xx| abs(xx)

THD (.four)

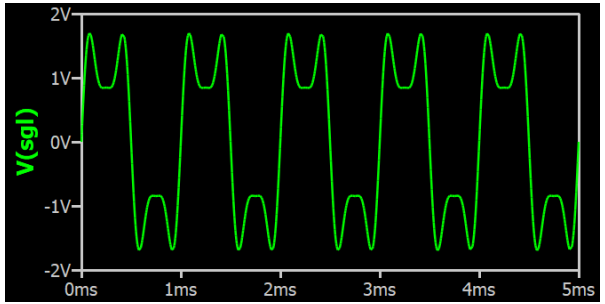
```
.four 1k v(sgl):
 Magnitude of Fundamental(RMS): 0.999922
 Harmonic  Frequency     Magnitude    Phase
    1      1.000e+03     1.000e+00     0.00°
    2      2.000e+03     3.498e-08   105.41°
    3      3.000e+03     5.996e-01     0.00°
    4      4.000e+03     1.713e-08    64.78°
    5      5.000e+03     1.996e-01    -0.00°
    6      6.000e+03     1.341e-08    35.02°
    7      7.000e+03     1.292e-06    -8.50°
    8      8.000e+03     2.028e-08   -30.72°
    9      9.000e+03     2.366e-07    -5.73°
 THD = 63.1981%(63.1981%)
```

.meas with four

```
.meas xx four 1k v(sgl):
      (1.54886e-07, -0.999999)
.meas |xx| abs(xx):
       0.999999       0.005
```

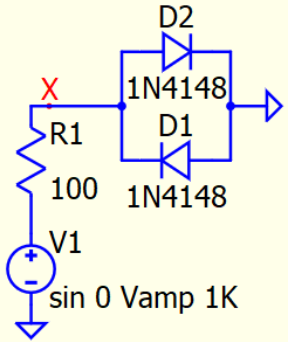
Fourier component is a complex number (re+j*im)

Magnitude (rms) can be calculated with abs()



.meas FOUR (fourier component) in .tran directive [with .step]

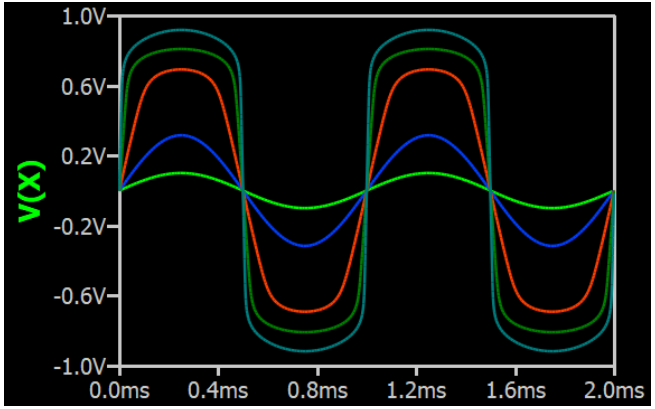
Qspice : .meas - four (.step).qsch




```
.step dec param Vamp 100m 10 2
.tran 2m
.plot V(X)
```

THD Total Harmonic Distortion
.four 1K V(X)
Fourier component with .meas
.meas xx four 1K V(x) format : complex number
.meas |xx| abs(xx) convert complex to magnitude

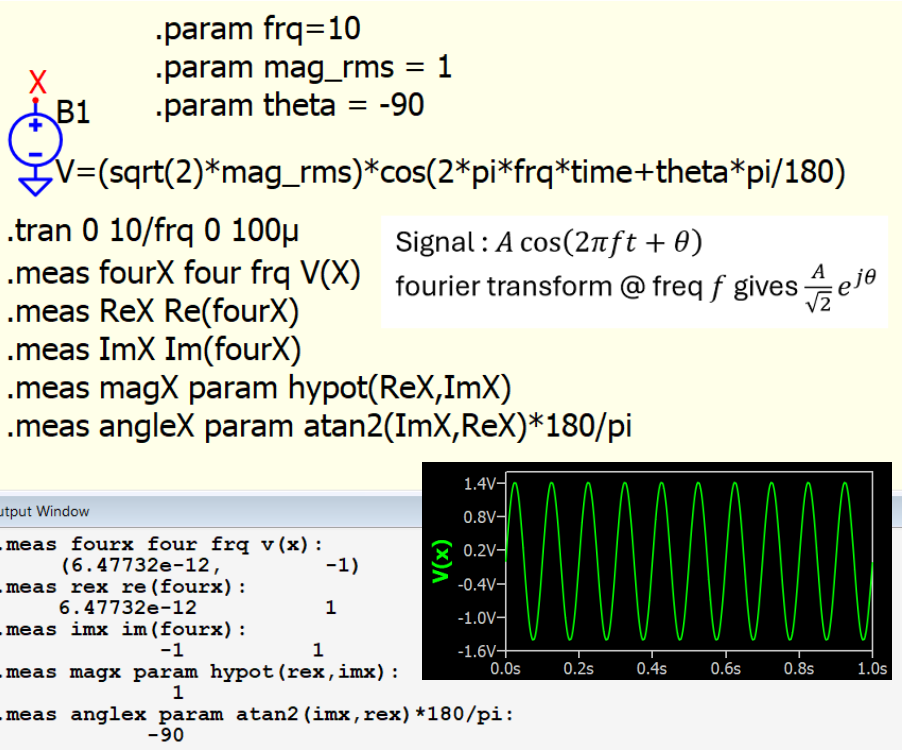
```
.meas xx four 1k v(x):
   0 (-7.73763e-08,-0.0707086)
   1 (-2.48947e-07, -0.223511)
   2 (-3.44697e-06, -0.551818)
   3 (-7.78738e-06, -0.698485)
   4 ( 2.254e-06, -0.797766)
.meas |xx| abs(xx):
   0  0.0707086       0.002
   1   0.223511       0.002
   2   0.551818       0.002
   3   0.698485       0.002
   4   0.797766       0.002
```



.meas FOUR (fourier component) : Mathematic Explanation

Qspice : .meas - four - math.qsch

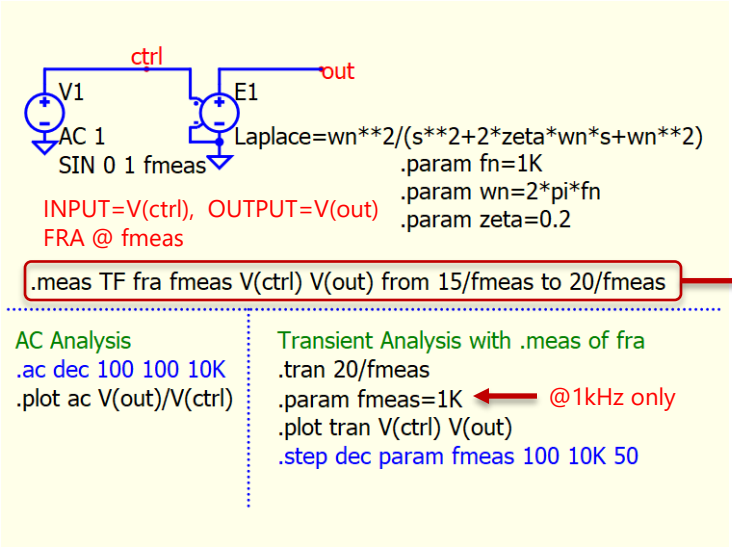
- Mathematic Explanation
  - In Qspice, fourier transform calculate signal magnitude (in rms) and phase and output a complex number R+jX (Re,Im) format
  - Fourier transform in Qspice
    - Signal : $A\cos(2\pi ft+\theta)$
    - Fourier transform @ frequence f gives $\frac{A}{\sqrt{2}}e^{j\theta}$
    - Therefore, fourier transform can give signal magnitude and phase at tested frequency, and to use consine function to reconstruct the waveform in time domain
  - Calculate magnitude and phase from (Re,Im) format
    - Magnitude (rms) : $\sqrt{R^2+X^2}$: hypot(Re,Im)
    - Phase : $\tan^{-1}\frac{X}{R}$: atan2(Im,Re)
      - 4 quadrant arc tangent is required to calculate phase in correct quadrant
  - In this example, $A_{rms}=1$ and $\theta=-90°$, which is a sine wave with 0 degree at t=0
    - Therefore, a sine source with 0 degree will give fourier transform with phase -90 degree! As fourier uses consine as reconstruction

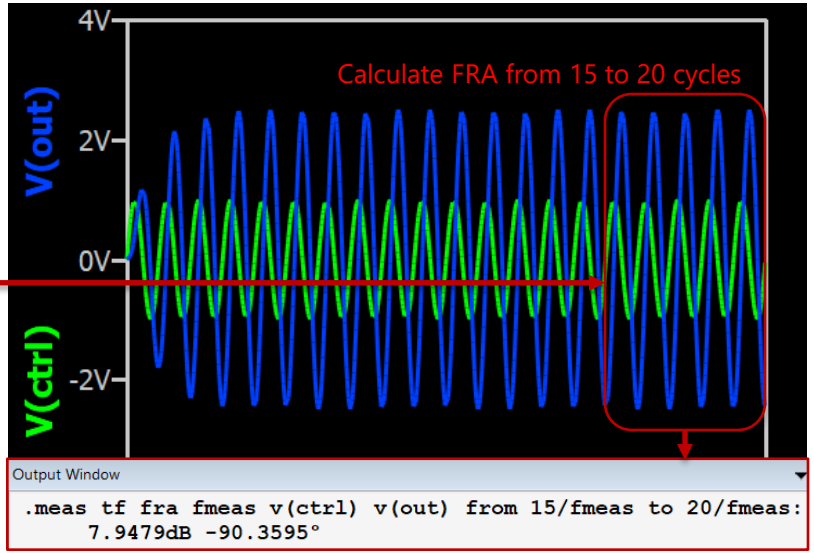


.meas – FRA : fourier component between OUTPUT and INPUT

Qspice : .meas - fra - 01 Basic.qsch

- Syntax : .meas NAME **fra** **FREQ** **INPUT** **OUTPUT** [... range limits ...]
  - Time domain frequency response analysis
  - **FRA** : Fourier component of **OUTUT** at **FREQ** divided by the Fourier component of **INPUT** at **FREQ**
  - Range limits can be set with from/to or trig/targ syntax
  - Normalization is to the time domain RMS

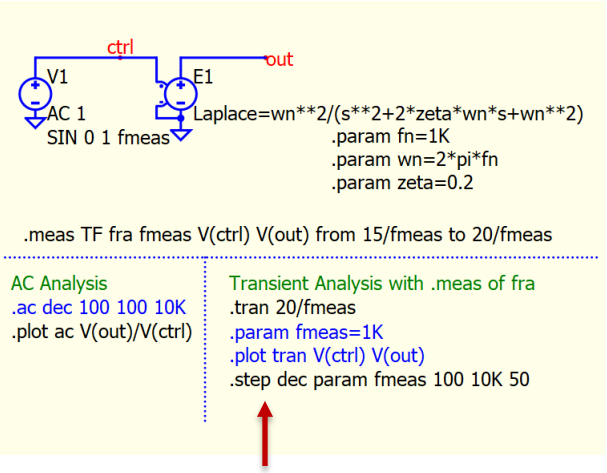




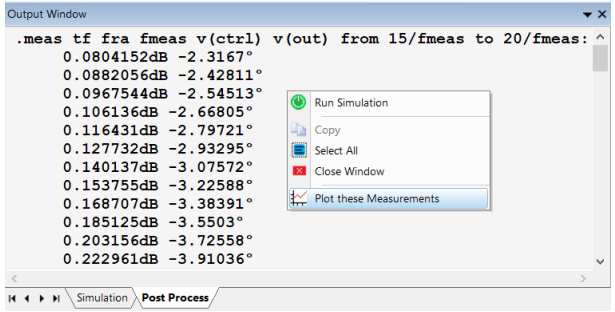
.meas FRA : fourier component between OUTPUT and INPUT

Qspice : .meas - fra - 02 Bode.qsch

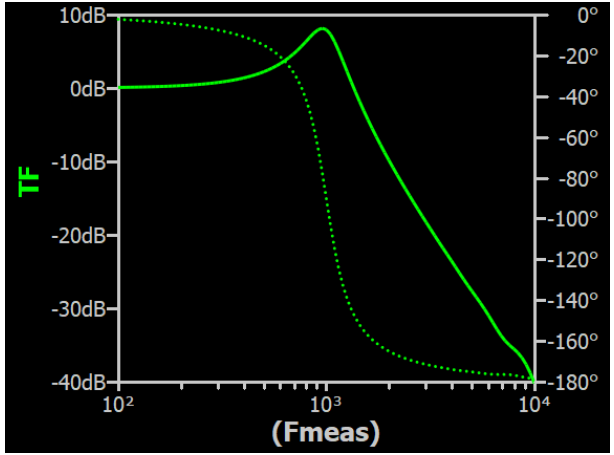
- Frequency response (bode plot) from time domain using FRA
  - In this example, .step is used to sweep FRA frequency
  - Time domain simulation is performed at each FRA frequency. Unlike .ac analysis, which linearizes the circuit around its DC operating point to compute signal magnitude and phase, FRA is performed in the time domain, making it useful for finding the small-signal response of SMPS
  - Qspice includes a demo in File > Open Demo… named *FRA_SMPS.qsch*



fmeas sweep from 100Hz to 10K with 50 points per decade



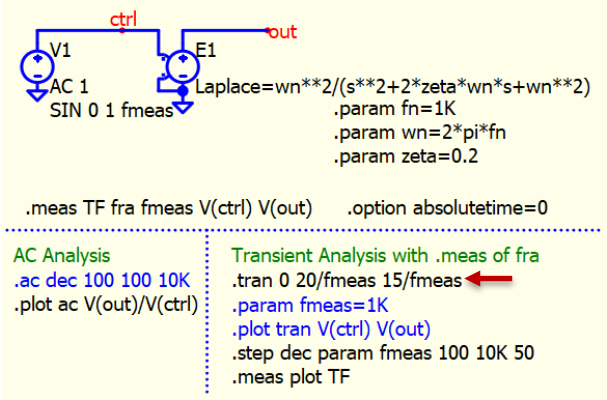
After .meas is ready in output window
Right click → Plot these Measurements
OR add ".meas plot TF" in schematic to get this plot automatically after simulation

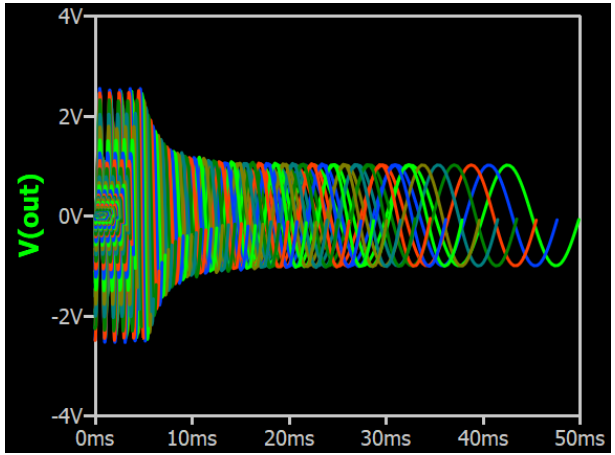


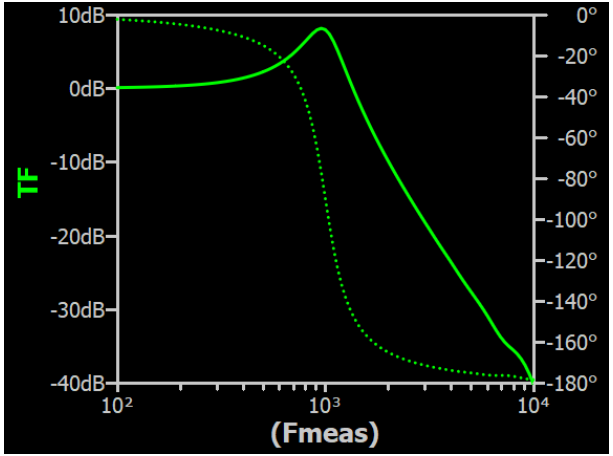
.meas FRA : fourier component between OUTPUT and INPUT

Qspice : .meas - fra - 03 Parameterized Tstart.qsch

- Parameterized Tstart
  - Setup .tran with parameterized Tstart and Tstop, and in this example, only save data in last 5 ac cycles for FRA analysis in each .step
  - To ensure .step can be kept track correctly, **.option absolutetime=0** is required (this is now default setting in .option)





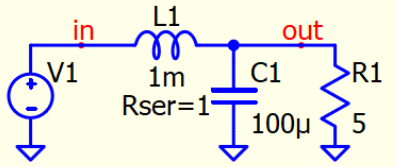


.meas
Measure Statements

Part 3 of 3
Examples

Example (.tran) – Steady State, Rise Time, Power Calculation

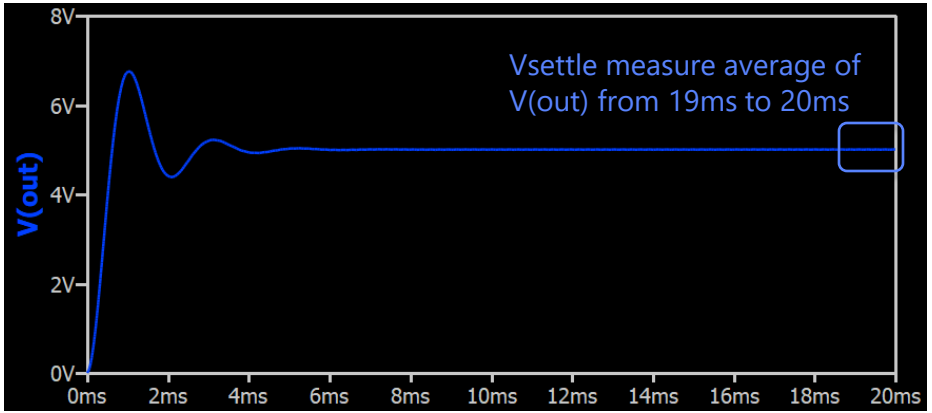
Qspice : .meas - (.tran) RiseTime SteadyState.qsch

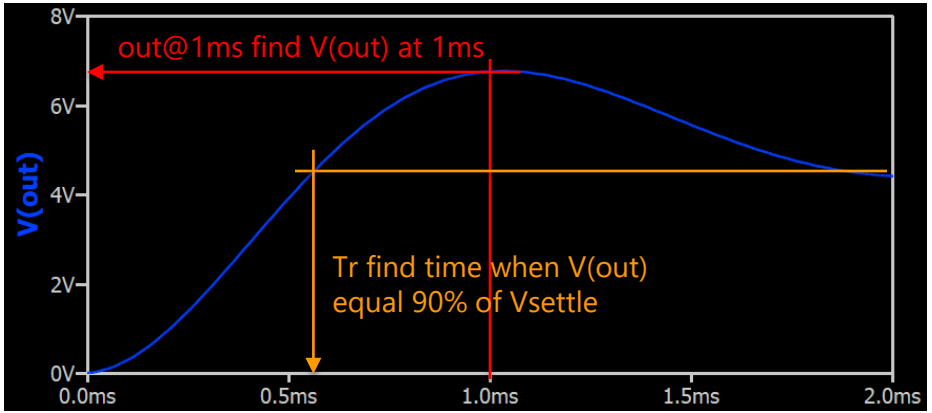



```
pulse 0 10 0 10n 10n 25µ 50µ
.tran 0 20m 0 0.1µ
 .plot V(out)
 .mease out@1ms find V(out) at 1m
 .mease Vsettle avg V(out) from 19m to 20m
 .mease Tr find time when V(out)=0.9*Vsettle

 .meas Pin avg V(in)*-I(V1) from 19m to 20m
 .meas Pout avg V(out)*I(R1) from 19m to 20m
 .meas %eff param Pout/Pin*100
```

Output Window

```
.mease out@1ms find v(out) at 1m:
         5.0911       0.001
.mease vsettle avg v(out) from 19m to 20m:
        4.16903
.mease tr find time when v(out)=0.9*vsettle:
     0.000565244 0.000565244
.meas pin avg v(in)*-i(v1) from 19m to 20m:
        4.17199
.meas pout avg v(out)*i(r1) from 19m to 20m:
        3.47616
.meas %eff param pout/pin*100:
        83.3213
```

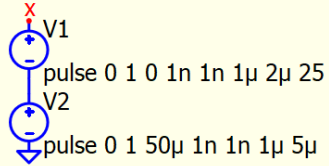




Example (.tran) – measure Pulse Period OnTime Freq and Duty

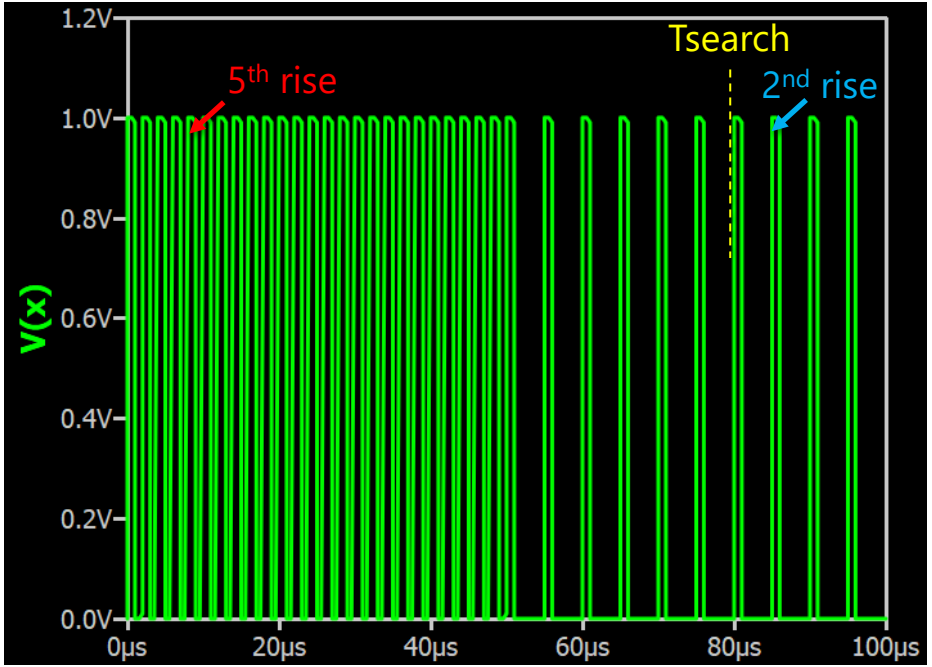
Qspice : .meas - (.tran) Pulse Period OnTime Freq Duty.qsch

```
.tran 100µ
; measure pulse info from 5th rising edge
.meas Tperiod1 trig V(x)=0.5 rise=5 targ V(x)=0.5 rise=6
.meas Ton1 trig V(x)=0.5 rise=5 targ V(x)=0.5 fall=5
.meas Freq1 param 1/Tperiod1
.meas Duty1 param Ton1/Tperiod1

; measure pulse info from 2th rising edge after Tsearch
.param Tsearch=80µ
.meas Tperiod2 trig V(x)=0.5 rise=2 Td=Tsearch targ V(x)=0.5 rise=3 Td=Tsearch
.meas Ton2 trig V(x)=0.5 rise=2 Td=Tsearch targ V(x)=0.5 fall=2 Td=Tsearch
.meas Freq2 param 1/Tperiod2
.meas Duty2 param Ton2/Tperiod2
```



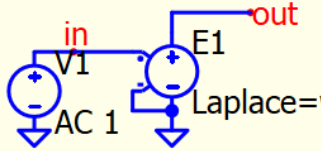
Output Window

```
.meas tperiod1 trig v(x)=0.5 rise=5 targ v(x)=0.5 rise=6:
         2e-06
.meas ton1 trig v(x)=0.5 rise=5 targ v(x)=0.5 fall=5:
      1.001e-06
.meas freq1 param 1/tperiod1:
         500000
.meas duty1 param ton1/tperiod1:
         0.5005
.meas tperiod2 trig v(x)=0.5 rise=2 td=tsearch targ v(x)=0.5 rise=3 td=tsearch:
         5e-06
.meas ton2 trig v(x)=0.5 rise=2 td=tsearch targ v(x)=0.5 fall=2 td=tsearch:
      1.001e-06
.meas freq2 param 1/tperiod2:
         200000
.meas duty2 param ton2/tperiod2:
         0.2002
```



Example (.ac) – Gain, Phase and Frequency meas

Qspice : .meas – (.ac) Gain Phase Freqeuncy.qsch




```
.param fn = 1K
.param wn = 2*pi*fn
.param z = 0.2
Laplace=wn^2/(s^2+2*z*wn*s+wn^2)

.ac dec 100 100 1e4
.plot V(out)

.meas Gain@max max dB(V(out))
.meas Freq@max find frequency WHEN dB(V(out))=Gain@max
.meas phase@2kHz find phase(V(out)) WHEN frequency=2K
.meas freq@+5dB find frequency WHEN dB(V(out))=5 cross=2
```

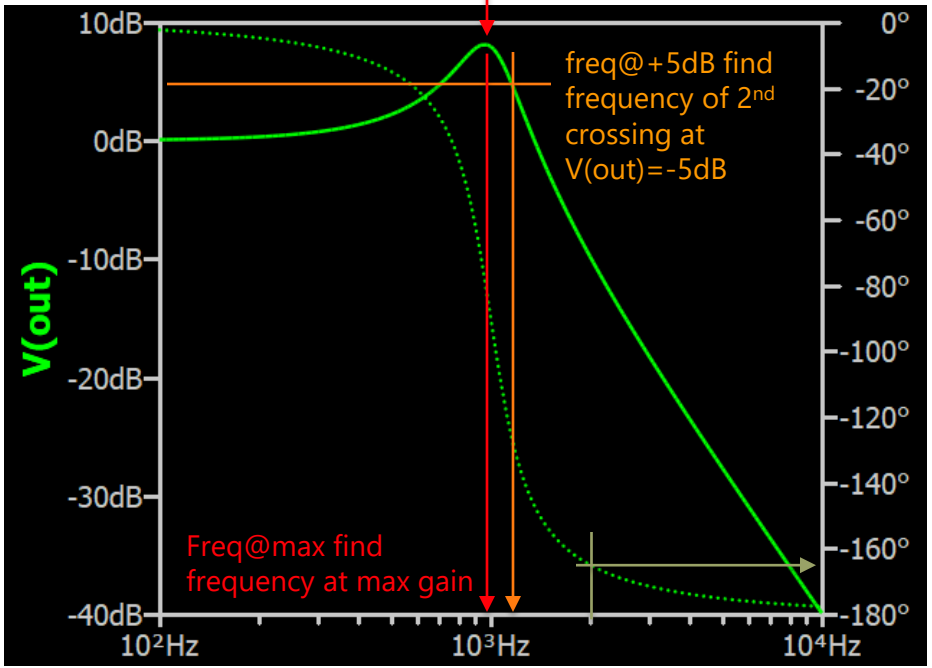
Support 6 types of measure

avg/ave : average
max : maximum
min : minimum
pp : peak to peak
rms : root mean square
integ : integral

Output Window

```
.meas gain@max max db(v(out)):
     (   8.13428, -0.350754) (at Frequency=954.993)
.meas freq@max find frequency when db(v(out))=gain@max:
     (   954.993,          0)
.meas phase@2khz find phase(v(out)) when frequency=2k:
     (  -165.064,          0)
.meas freq@+5db find frequency when db(v(out))=5 cross=2:
     (   1150.32,          0)
```

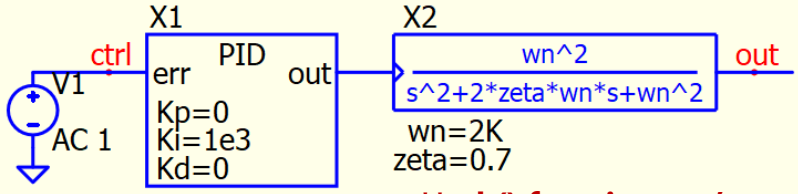
Gain@max find maximum of V(out) in dB



phase@2kHz find phase at frequency is 2kHz

Example (.ac) – Gain and Phase Margin, Crossover Frequency

Qspice : .meas - (.ac) Gain and Phase margin.qsch



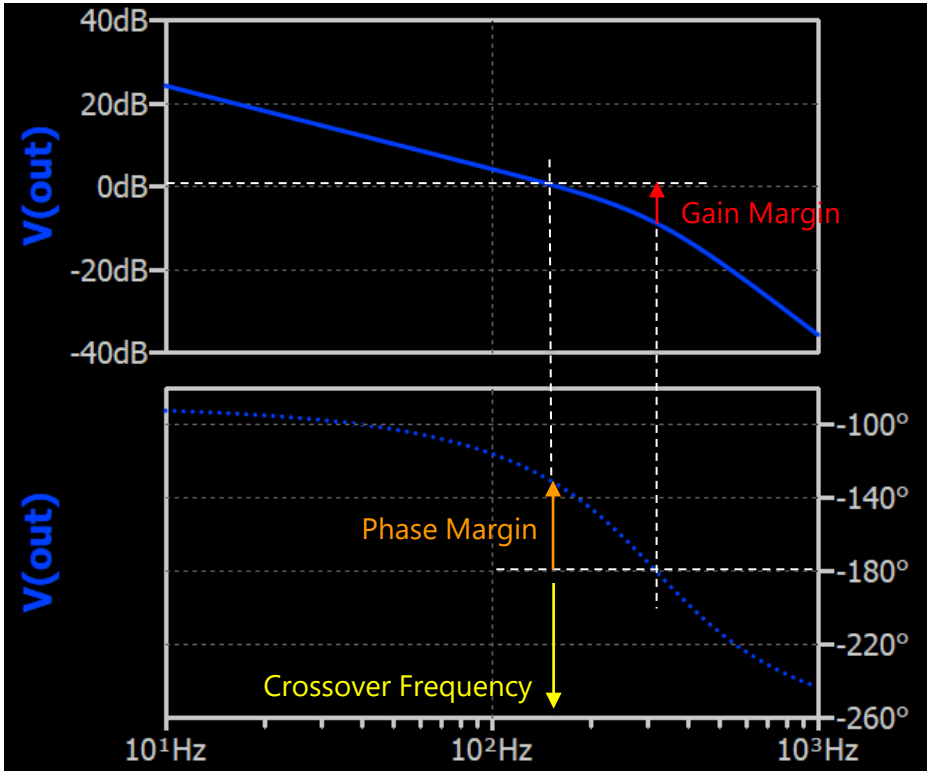

```
.ac dec 100 10 1K
.plot V(out)
.plot V(out)
```

**** ph() function can't unravel phase and only return -180 to 180, it cannot detect equality at exactly -180**

```
.meas GainMargin1 find -dB(abs(V(out))) when ph(V(out))=-179
.meas GainMargin2 find -dB(abs(V(out))) when im(V(out))=0
.meas Ph@0dB find ph(V(out)) when dB(V(out))=0
.meas PhaseMargin param Ph@0dB+180
.meas fcrossover find Frequency when dB(V(out))=0
```

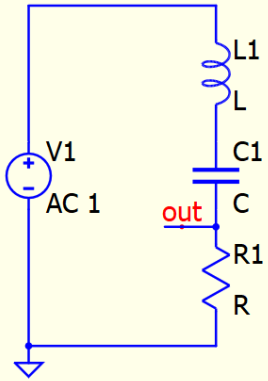
Output Window

```
.meas gainmargin1 find -db(abs(v(out))) when ph(v(out))=-179:
     (   8.73229,         -0)
.meas gainmargin2 find -db(abs(v(out))) when im(v(out))=0:
     (     8.947,         -0)
.meas ph@0db find ph(v(out)) when db(v(out))=0:
     (  -131.934,          0)
.meas phasemargin param ph@0db+180:
     (   48.0659,          0)
.meas fcrossover find frequency when db(v(out))=0:
     (   155.514,          0)
```



Example (.ac) – Q measurement

Qspice : .meas - (.ac) Q of LCR Resonant.qsch

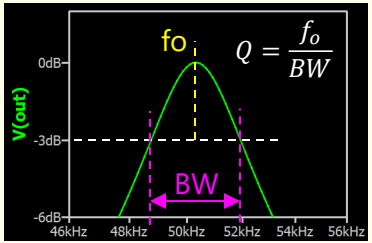



```
.param L=10µ
.param C=1µ
.param R=0.2

.ac dec 1000 1 1G
.options LISTPARAM
.plot V(out)
```

[1] Add this option to display parameter evaluations result in output window

Series RLC:

$$Q = \frac{1}{R}\sqrt{\frac{L}{C}}$$



Q formula of Series RLC

```
.param Qcal 1/R*(L/C)**0.5
```

Q calculation from Bandwidth (BW) and Center Frequency (fo) : Q=fo/BW

```
.meas Vmax max mag(V(out))
.meas fo FIND frequency WHEN mag(V(out))=Vmax
.meas fL FIND frequency WHEN mag(V(out))=Vmax/sqrt(2) rise=1
.meas fH FIND frequency WHEN mag(V(out))=Vmax/sqrt(2) fall=last
.meas BW fH-fL
.meas Q fo/BW
```

Output Window

```
   --- Parameter Evaluations ---
TEMP        = 27        "CKTTEMP"
L           = 10µ       "10µ"
C           = 1µ        "1µ"
R           = 200M      "0.2"
QCAL        = 15.8114   "1/R*(L/C)**0.5"

C:\QspiceKSKelvin\01 User Guide and Script\01 Qspice Refer

Total elapsed time: 0.129028 seconds.
```

In simulation, it has .param calculation results

Simulation | Post Process

Output Window

```
.meas vmax max mag(v(out)):
     (  0.999835,          0)  (at Frequency=50350.1)
.meas fo find frequency when mag(v(out))=vmax:
     (   50350.1,          0)
.meas fl find frequency when mag(v(out))=vmax/sqrt(2) rise=1:
     (   48762.3,          0)
.meas fh find frequency when mag(v(out))=vmax/sqrt(2) fall=last:
     (   51946.9,          0)
.meas bw fh-fl:
     (   3184.61,          0)
.meas q fo/bw:
     (   15.8104,          0)
```

In post process, it has .meas calculation results

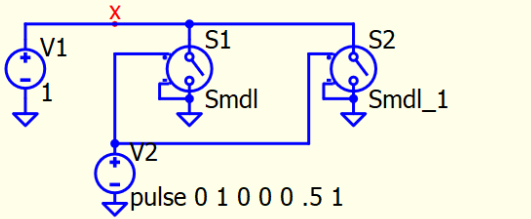
Simulation | Post Process

.model
Define Model

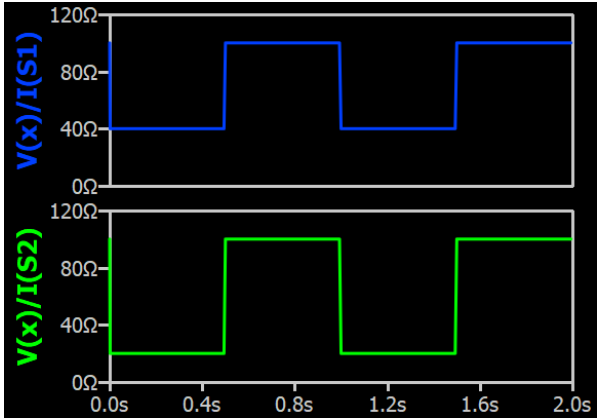
.model – ako Aliases (A Kind Of)

Qspice : model - ako.qsch

- ako (undocumented)
  - Aliases (**A K**ind **O**f)
  - Syntax : **ako:**
  - Modify parameters of an existing model
  - Example
    - Smdl_1 aliases model Smdl, but only changed Ron from 40 to 20

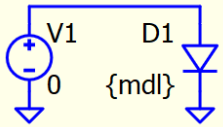



```
.plot V(x)/I(S2)
.plot V(x)/I(S1)

.tran 2
.model Smdl SW Ron=40 Roff=100 Vt=0.5 Vh=0
.model Smdl_1 ako: Smdl Ron=20
```

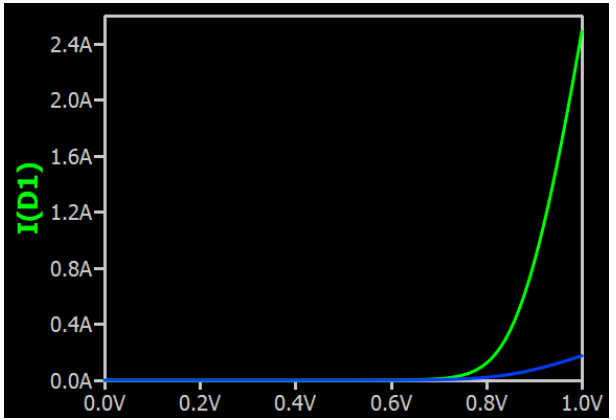


- Step model with ako!
  - With ako, it is possible to step a model in simulation
  - .step only accept numerical value
  - .model use numerical value for ako model name
  - Model name of device must be in curly bracket {}

{} must be used to step model




```
.dc V1 0 1 0.01
.plot I(D1)

.step param mdl list 4148 4007

.model 4148 ako:1N4148
.model 4007 ako:1N4007
```

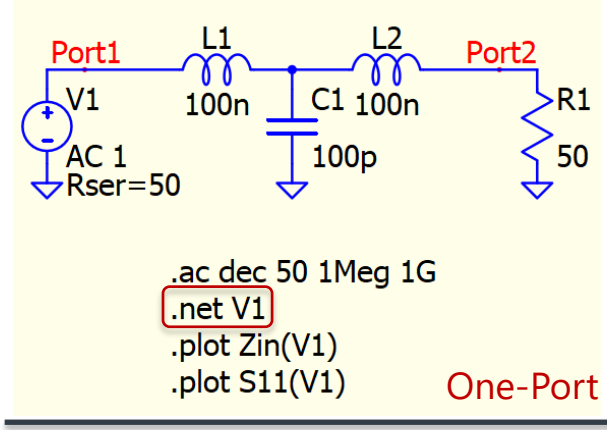


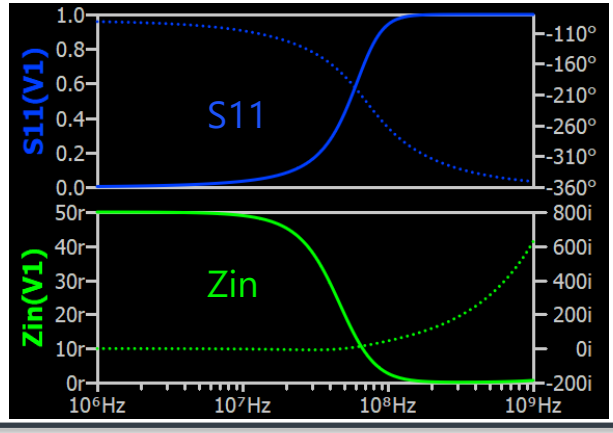
.net
Network Parameters

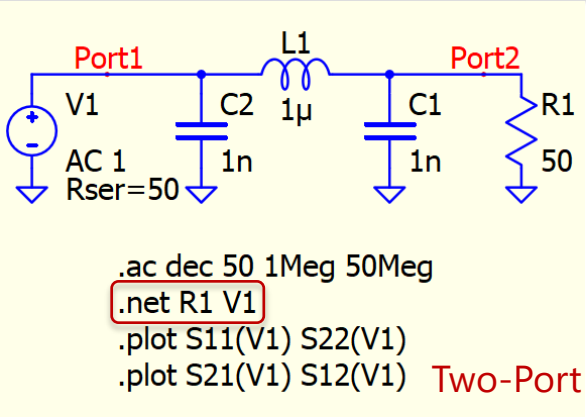
.net Network Parameters – Basic

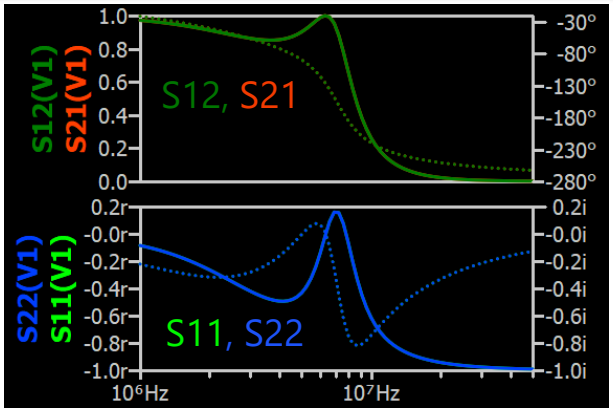
Qspice : .net - One-Port.qsch | .net - Two-Port.qsch

- .net
  - To extract small signal One- or Two-port network parameters : S-, Y-, Z- and H-parameters
  - **Syntax :**
    - 1-port : .net <Vin>
    - 2-port: .net <Rout> <Vin>
  - Note
    - Port 1 (Input Port) : V-source must have Rser, which is Zo of S-parameters at Port 1
    - Port 2 (Output Port) : Resistor and its resistance is Zo at Port 2
      - where Zo is characteristic impedance





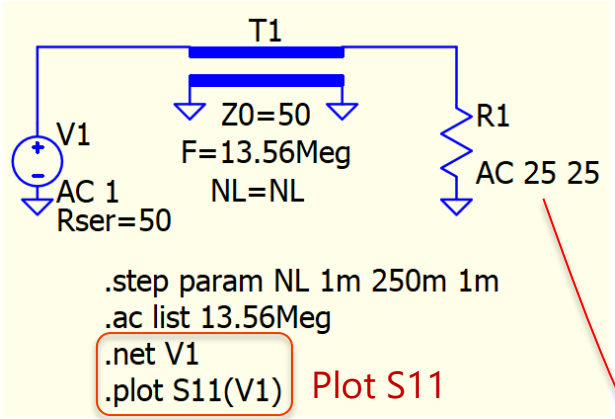


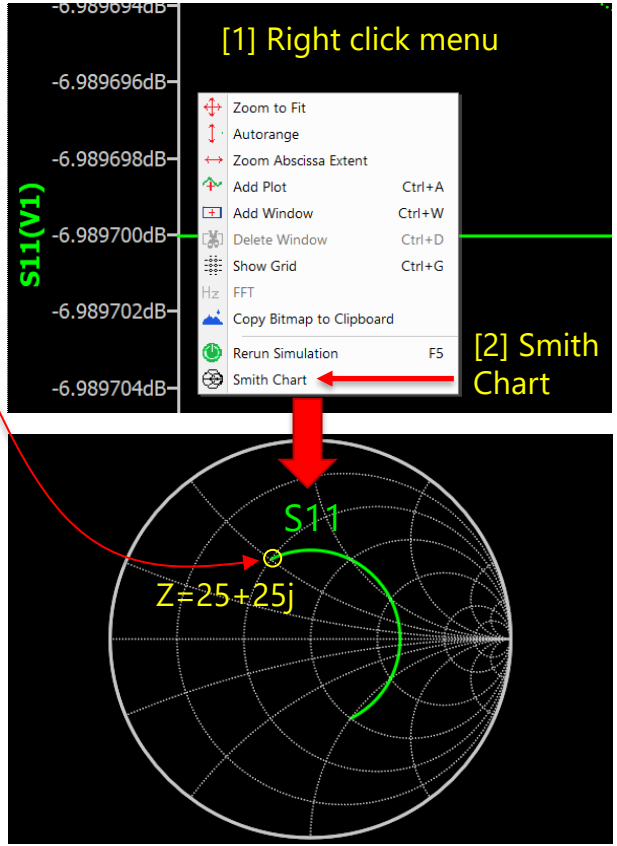


.net Network Parameters – SmithChart and R with AC

Qspice : .net - SmithChart and R-AC.qsch

- SmithChart
  - To plot the S11 trace on a Smith Chart, right-click on the axis and select [Smith Chart] in the Representation options
- AC resistor (complex)
  - In this example, R1 is defined to represent complex impedance in .ac/.net analysis
  - Resistor with instance parameter AC [Re] [Im] represent Z = [Re] + j [Im], where impedance is frequency-independent

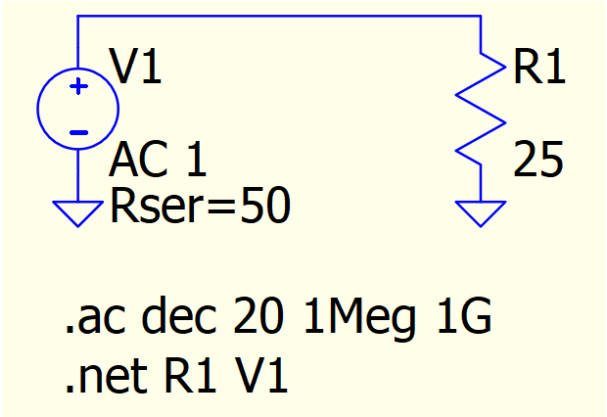


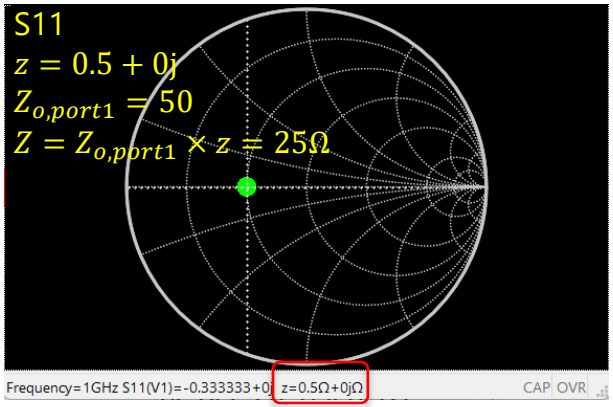


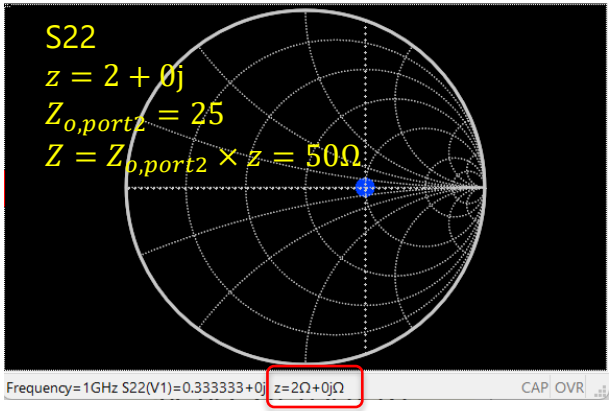
.net Network Parameters – SmithChart in Two-Ports

Qspice : .net - Two-Port SmithChart.qsch

- SmithChart in Two-Ports
  - Rser of <VIN> determines the characteristic impedance (Zo) of Port 1, and the resistance of <Rout> determines Zo of Port 2
  - In general, for Zo=50 systems, both Rser and R of <Rout> should be set to 50
  - In SmitchChart (S11, S22), it can read normalize impedance $z = \frac{Z}{Z_o}$
  - If the characteristic impedance of Port 1 and Port 2 are different, the SmithChart characteristic impedance of S11 follows Port 1 Zo, port 1, and S22 follows Port 2 Zo, port 2







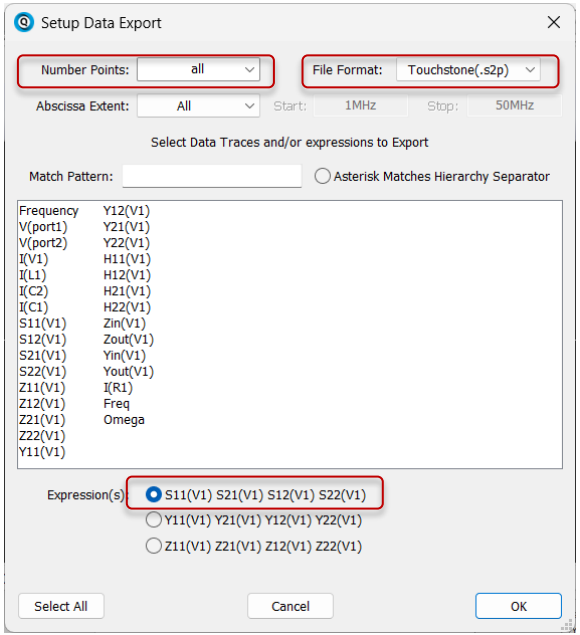
Export and Read Touchstone .s2p data file

- Export Touchstone .s2p
  - Two-ports .net results can be exported as Touchstone .s2p from waveform viewer
- Read Touchstone .s2p
  - Touchstone .s2p file can be opened with QUX.exe (Schematic or Waveform viewers) and plot in waveform viewer
  - OR, directly drag a .s2p file from browser to Waveform Viewer
    - Drag a .s2p file from browser to Schematic viewer will trigger auto-generation of a S-parameter model block, which is an operation describe in next slide

Export Touchstone (.s2p)

In Waveform Viewer: File > Export Data

- Number Points : all
- File Format : Touchstone(.s2p)
- Expression(s) : S11(V1)... S22(V1)

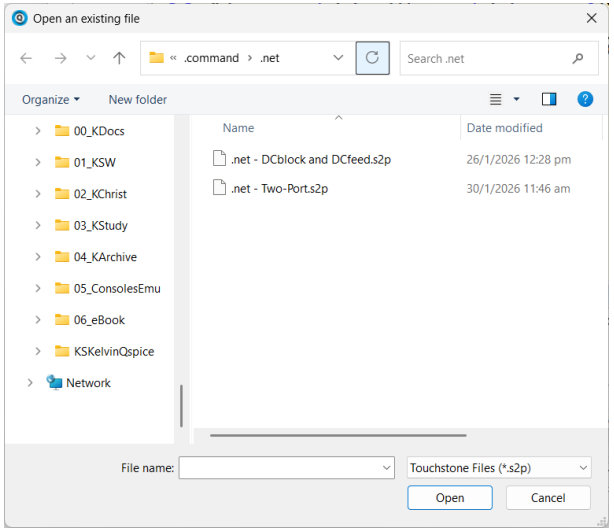


Open Touchstone (.s2p)

In Schematic or Waveform Viewer

File > Open

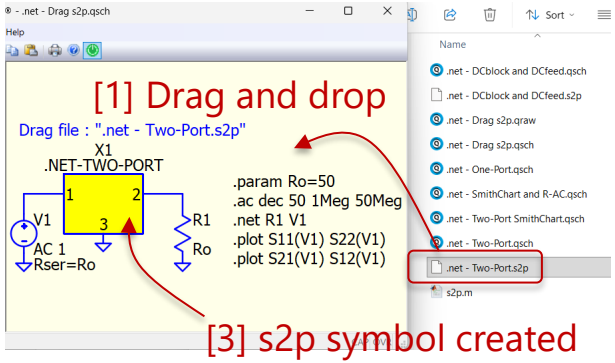
File Type : Touchstone Files (*.s2p)

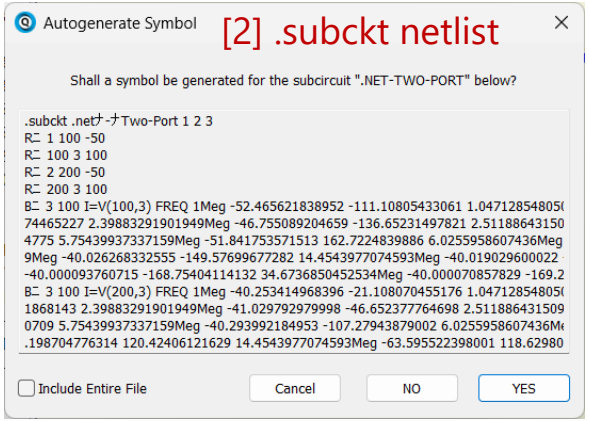


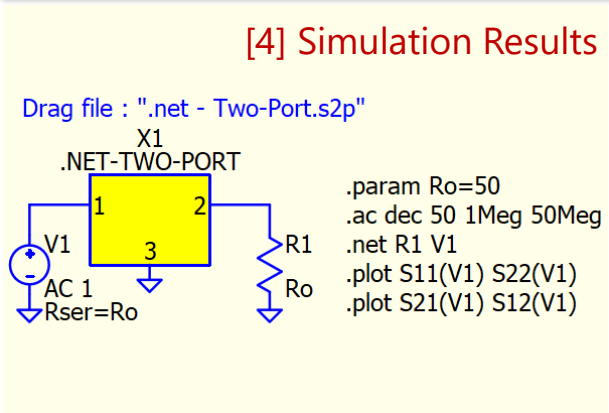
S-parameters Model from Touchstone .s2p

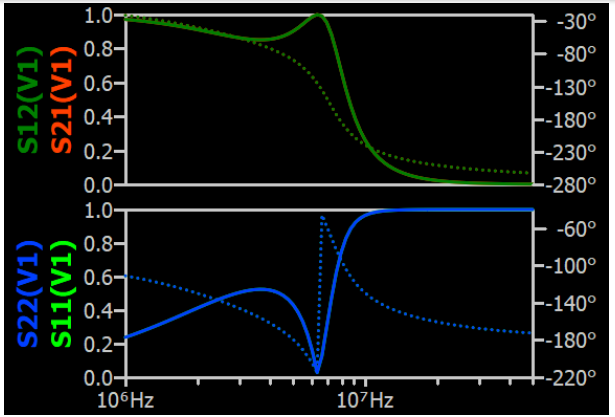
Qspice : .net - Drag s2p.qsch

- S-param Model from Touchstone .s2p
  - A two-port network can be generated by dragging and dropping a Touchstone **.s2p** file from the file explorer to the schematic window directly
  - An Autogenerate Symbol window will pop up to create a subcircuit representing this touchstone .s2p data









Technical note about .net in Qspice

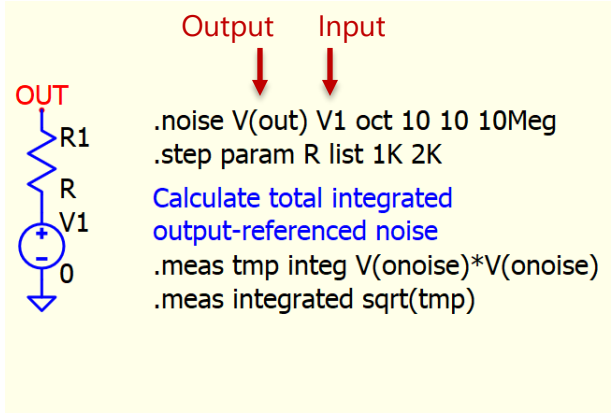
- AC voltage level in Port 1 AC source
  - The AC voltage level will not affect .net calculations. Even **AC 0** can return correct .net results
- Characteristic Impedance Zo
  - In general, it is recommended that **Rser** in <Vin> and **<Rout>** be set to the same value equals characteristic impedance Zo (e.g., 50), to prevent confusion in obtaining the S-parameters from the system
  - For one-ports, if Rser=1 in <Vin>, nominal impedance (*z*) in SmithChart plot is directly the impedance reading
- Two-ports .net workflow : .net [<Rout>] <Vin>
  - When calculating S11 and S21: Port 1 is <Vin> in series with Rser, and Port 2 is Rout
  - When calculating S22 and S12: Port 1 is just Rser (voltage source removed), and Port 2 is a current in parallel with Rout (Norton equivalent)
    - Mike Engelhardt explanation : I use the Norton equivalent, i.e., Rout with a current in parallel. Using the Thévenin equivalent for the input and the Norton equivalent for the output means that the circuit topology never changes. I just manipulate the excitations to allow extraction of the S-parameters by inspection.

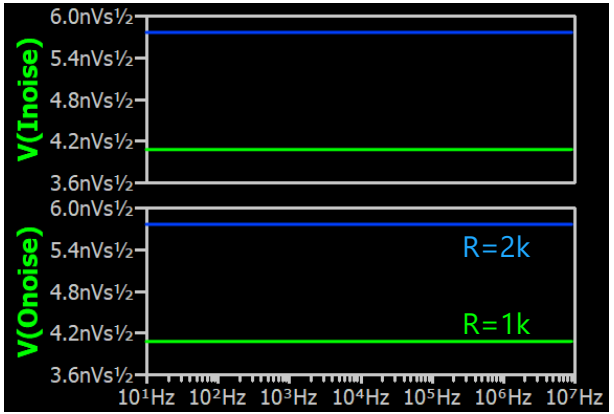
.noise
Stochastic Noise Analysis

.noise Stochastic Noise Analysis

Qspice : noise - basic.qsch

- .noise
  - Stochastic Noise Analysis
  - .noise calculated results
    - Noise spectrum density per unit square root bandwidth ($\mathrm{V}/\sqrt{\mathrm{Hz}}$)
    - Input : Inoise
    - Output : Onoise
  - Notes
    - Noise analysis is performed without the needs of input voltage
    - In output window, it calculates total integrated output and input-referenced noise in rms
      - $\sqrt{\int (V_{noise})^2 \, df}$





Output Window (Simulation)

```
1 of 2 steps:  .step r=1000
Total integrated output-referenced noise: 12.8748µV rms
Total integrated input-referenced noise: 12.8748µV rms
2 of 2 steps:  .step r=2000
Total integrated output-referenced noise: 18.2077µV rms
Total integrated input-referenced noise: 18.2077µV rms
```

Output Window (Post-Processing)

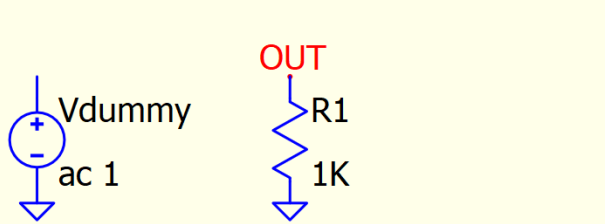
```
.meas tmp integ v(onoise)*v(onoise):
   0 1.65761e-10
   1 3.31521e-10
.meas integrated sqrt(tmp):
   0 1.28748e-05      1e+07
   1 1.82077e-05      1e+07
```

Thermal Noise Formula of Resistor

Qspice : .noise - resistor (thermal noise).qsch

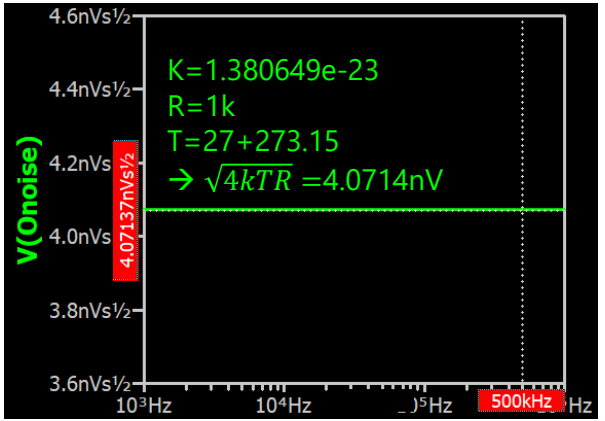
- Thermal noise of R
  - $V_{noise} = \sqrt{4kTRB}$
    - V_{noise} = Thermal noise in voltage V
    - k = Boltzmann's constant 1.380649×10^{-23} J/K
    - R = Resistance in Ohms
    - T = Temperature in Kelvin 0K = -273.15℃
    - B (Δf) = Bandwidth of system in Hz
  - $\frac{V_{noise}}{\sqrt{1\text{Hz}}} = \sqrt{4kTR}$
    - Unit is $V/\sqrt{Hz}$ or $Vs^{1/2}$

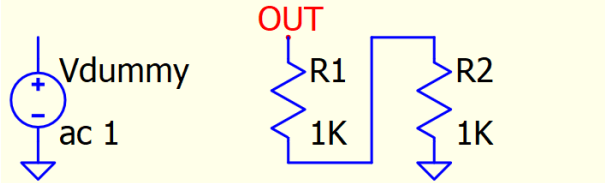
- Two Resistors in series
  - $V_{n,Total} = \sqrt{V_{n1}^2 + V_{n2}^2}$



.noise V(out) Vdummy oct 10 1K 1Meg

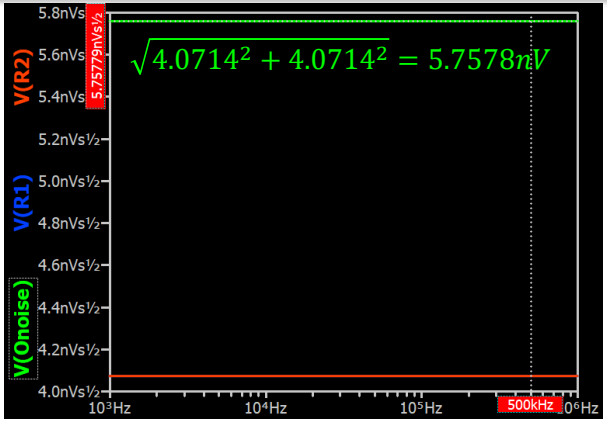
.plot V(Onoise)





.noise V(out) Vdummy oct 10 1K 1Meg

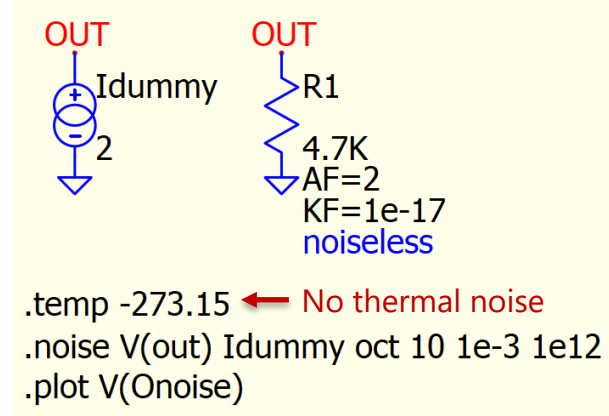
.plot V(Onoise)

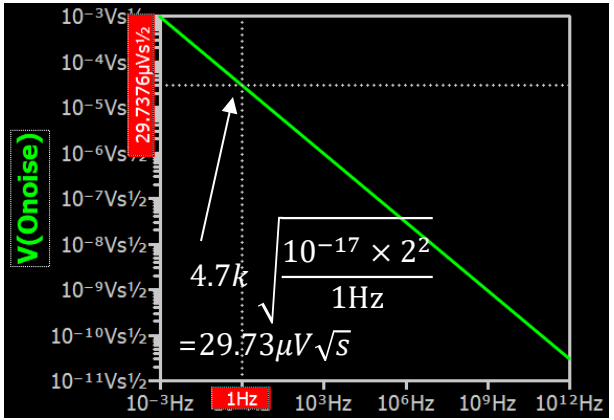


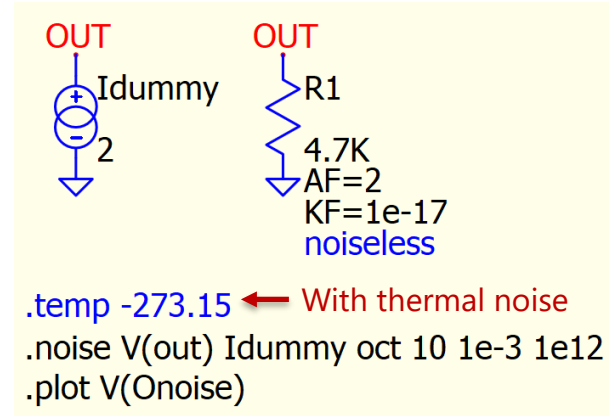
Flicker Noise Formula of Resistor

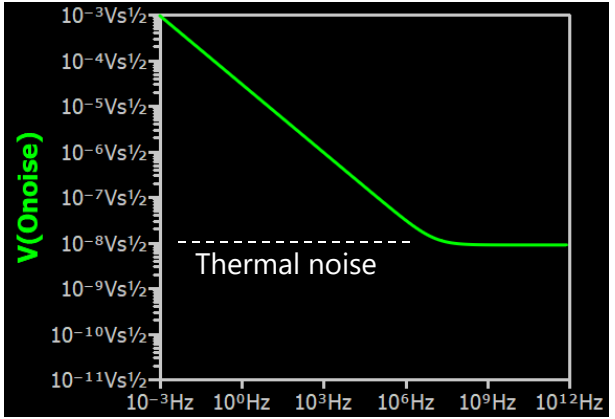
Qspice : .noise - resistor (flicker noise).qsch | flickernoise.m

- Flicker Noise
  - Flicker noise also known as 1/f noise or pink noise, which is current dependent
  - Instance parameter of flicker noise includes
    - AF : Flicker noise exponent
    - KF : Flicker noise coefficient
  - Formula
    - $V_{noise} = R\sqrt{\frac{KF \times I^{AF}}{f}}$
    - Where I is current, f is frequency and R is resistance value



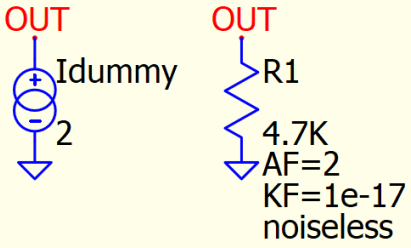




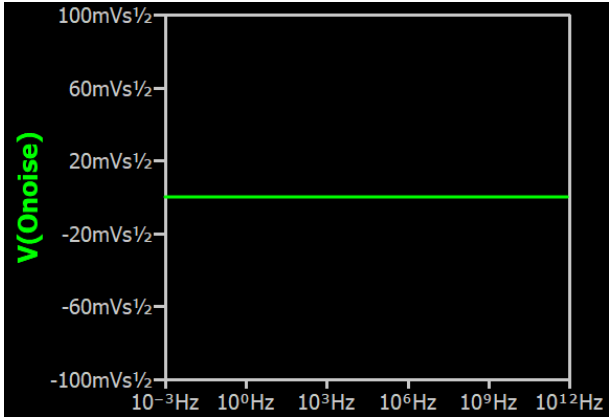


Noiseless

- Noiseless
  - Remove noise from devices




```
.temp -273.15
.noise V(out) Idummy oct 10 1e-3 1e12
.plot V(Onoise)
```

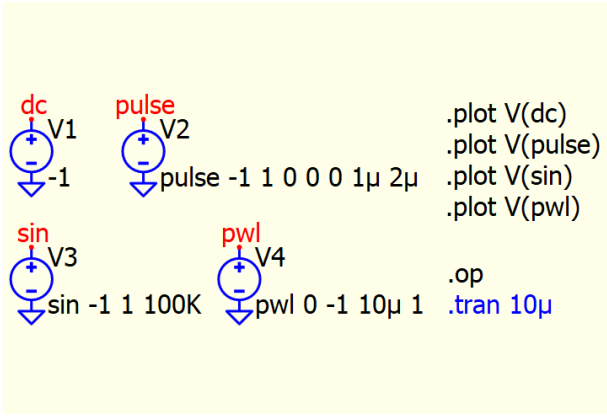


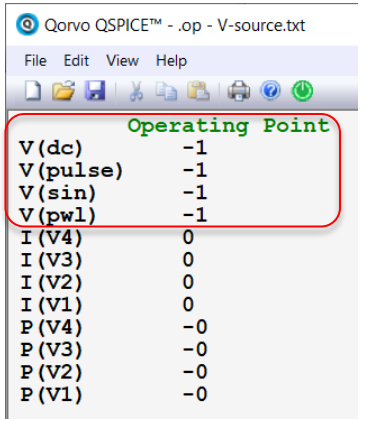
.op
Bias Point Analysis

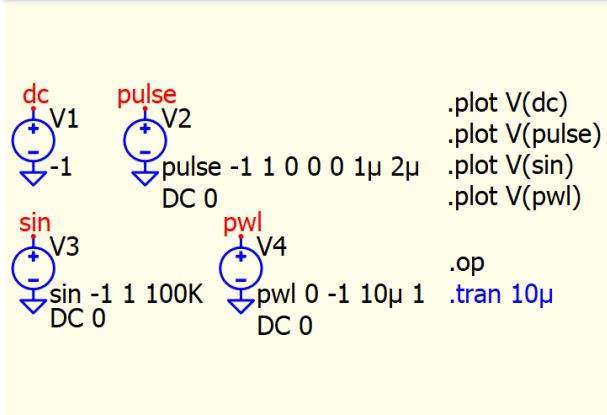
.op – DC Solution in Source

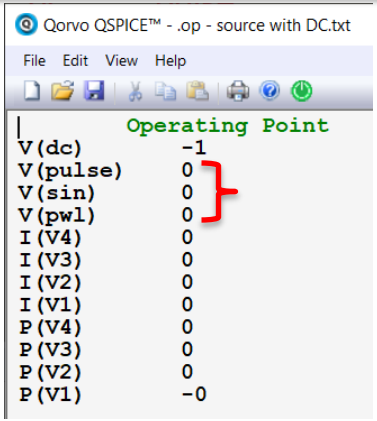
Qspice : .op - source without DC.qsch | .op - source with DC.qsch

- DC Solution in Source
  - The DC solution for source will be its voltage at t=0 unless a DC value is specified









.option / .options
Set Simulator Options

Set Simulator Options

Set Simulator Options

Syntax: .option NAME1=VALUE1 [NAME2=VALUE2 [...]]

Recognized Options

Name	Description	Default
ABSTOL	Absolute error tolerance	1e-12A
ACCT	Print accounting information	(not set)
ASCII	ASCII .qraw file	(not set)
BINARY	Override command line switch to use ASCII .qraw file	(not set)
BODEAMPFREQ	Frequency with the minimum perturbation amplitude. Set to 0. for constant amplitude.	(not set)
BODEHIPOW	Controls perturbation amplitude for above BODEAMPFREQ by pow (freq/BODEAMPFREQ, BODEHIPOW)	1.
BODEINPUT[1]	Override input node for transfer function computation(aka BODEIN)	auto
BODELOPOW	Controls perturbation amplitude for below BODEAMPFREQ by pow (freq/BODEAMPFREQ, BODELOPOW)	1.
BODEPERIODS	Maximum number of periods to include in deconvolution	20
BODEREF	Reference node to use for Frequency Response Analysis	Node 0 (global ground)
BODEOUTPUT[1]	Override output node for transfer function computation(aka BODEOUT)	auto
BODETOL	A Frequency Response Analysis relative tolerance	10.
CAPOP	0: Use model value 1: Use Meyer, >1 Use BSIM1	0
CHGTOL	Charge error tolerance	1e-14C
CSHUNT	Capacitance added from every node to ground(aka CMIN)	0F
DEFAD	Default MOSFET area of drain	$0m^2$
DEFAS	Default MOSFET area of source	$0m^2$
DEFL	Default MOSFET length	10µm
DEFW	Default MOSFET width	10µm
FEATHER	Trap integration damping factor	0
GMIN	Minimum conductance	1e-12℧
GMINSTEPS[2]	Number of Gmin steps	10
GSHUNT	Conductance added from every node to ground	0℧
ITL1	DC iteration limit	100
ITL2	DC transfer curve iteration limit	50
ITL4	Transient analysis iteration limit	10
KEEPOPINFO	Record operating point for small-signal analysis	(not set)
LAUNCHQUX[3]	Open the .qraw file in the waveform viewer after the simulation	(not set)
LIST[4]	Print an expanded netlist	(not set)
LISTPARAM	Print a list of the evaluated parameters(also sets SAVEPARAM)	(not set)
LFT	Frequency at which default inductor damping has parallel resistance equal to reactance.	(not set)
MAXORD	Maximum integration order	2
MAXSTEP	Maximum timestep size for .bode and .tran	infinite
MAX1STSTEP	Maximum timestep size the very first timestep for a .tran	100ns
METHOD	Integration method(trap or Gear)	trapezoidal
MINBREAK[5]	Minimum time between breakpoints	0s
NOOPITER	Go directly to Gmin stepping	(not set)
NUMDGT	Number of significant digits in an ASCII .qraw file	15
PIVREL	Minimum relative matrix pivot	1e-3
PIVTOL	Minimum absolute matrix pivot	1e-13
RELTOL	Relative error tolerance	0.1%
RIC[6]	Impedance of source asserting initial conditions	1mΩ
SAVEPARAM	Include evaluted user-defined parameters as waveform data	(not set)
SAVEPOWERS[7]	Compute and save the dissipation of components	(not set)
SEED[8]	Initialize the random number generater used in .param statements	
SEEDCLOCK	Initialize the random number generater with a 10Mhz clock and the process ID number(aka SEEDCLK).	(not set)
SRCSTEPS[2]	Number of source steps(aka ITL6)	10
TEMP	Operating temperature	27°C
TNOM	Nominal temperature(aka TREF)	27°C
TRTOL	Truncation error overestimation factor	2.5
TRTOL2	Another dimensionless truncation error guidance	1e-8
TRYTOCOMPACT	Try compaction for LTRA lines	(not set)
VNTOL	Voltage error tolerance	1µV

Bode

1] If a resistor is used to indicate where to insert the perturbation, the resistive divider's contribution is excluded.
2] Since an adaptive step size algorithms are used, the value of GMINSTEPS or SRCSTEPS is irrelevant unless set to zero, which means don't try the stepping algorithm.
3] Useful when running simulations from the command line. Don't use it if QSPICE64.exe or QSPICE80.exe are launched from the GUI.
4] Solely for internal diagnostic purposes. Probably not what you're looking for.
5] MINBREAK is automatically computed if left zero.
6] Inductor currents are asserted with the compliance of 1e9 * RIC.
7] Computes the true power dissipation while ignoring displacement currents. Implemented for BJTs, Capacitors, Diodes, Inductors, JFETs, MOSFET level 1, MOSFET level 2010 and VDMOS.
8] Used in .param functions Random() and Gauss(double sigma).

One Page Summary for Convergence Related Option

- Convergence (Voltage / Current)
  - ABSTOL : Absolute Current Tolerance [1pA]
  - VNTOL : Absolute Voltage Tolerance [1uV]
  - RELTOL : Relative Error Tolerance [0.1% = 0.001]
  - CHGTOL : Charge Error Tolerance [1e-14C]

- Convergence (Impedance Insertion)
  - Gshunt : Conductance (G) added between each node and ground [0℧]
  - Gmin : Conductance (G) added between PN junction [1e-12℧]
  - Cshunt (Cmin) : Capacitance (C) added between each node and ground [0F]

- DC Analysis ONLY
  - NoOpiter : Skip Direct Newton Iteration [not set]
  - GminSteps : Enable/Disable Adaptive Gmin Stepping [10]
  - SrcSteps (ITL6) : Enable/Disable Adaptive Source Stepping [10]
  - ITL1 : DC iteration limit [100]
  - ITL2 : DC transfer curve iteration limit [50]

- Transient Analysis ONLY
  - ITL4 : Transient analysis iteration limit [10]
  - Maxstep : Maximum timestep size [not set]
  - TRTOL : Truncation error overestimation factor – affect timestep accuracy [2.5]
  - TRTOL2 : Another dimensionless truncation error guidance – affect TTOL over simulation time [1e-8]

- Solver
  - Fastmath : Preferred 64 bits or 80 bits solver (1 : QSPICE64.exe or 0 : QSPICE80.exe) [none]
  - MaxOrd : Integration method (1 : Backward Euler , 2 : Trapezoidal / Gear) [2]
  - Method : Integration method (Trapezoidal / Gear) [Trap]
  - Feather : Damping Factor for Trapezoidal Integration [0]

.option – Reltol, Abstol, Vntol (Convergence Criteria Parameters)

** this slide may be incorrect in describing Qspice, use as reference only

- Solving Nonlinear Equation in SPICE : Newton-Raphson Method
  Newton-Raphson algorithm (aka Newton's method)
  - Initial guess on the solution $v^{<0>}$
  - Linearizes around the initial guess with Jacobian $J(v^{<0>}) = \frac{d}{dv} f(v^{<0>})$
  - Solves the resulting system of linear equations $v^{<k>}$
  - Re-linearize around the new point $J\left(v^{<k>}\right) = \frac{d}{dv} f\left(v^{<k>}\right)$
  - Repeats until the solution **converges (Residue and Update Criterion)**

- Residue Criterion (1st Convergence criteria)
  - KCL should be satisfied to a certain degree (ideally, the sum of currents at each node equal zero)
  - In practice : $\sum \boldsymbol{I(node_i)} < \textbf{reltol} \times |\boldsymbol{I_{max}}| + \textbf{abstol}$
    - Imax is the maximum current in this circuit simulation
  - **Relative error tolerance** : default **reltol** = 0.1% = 0.001
  - **Absolute current error tolerance** : default **abstol** = 1pA (eliminate the need to absolute zero)

- Update Criterion (2nd Convergence criteria)
  - The difference in error between the results of the last two iterations is small
  - In practice : $\left|v^{<k>} - v^{<k-1>}\right| < \textbf{reltol} \times \max\left(v^{<k>}, v^{<k-1>}\right) + \textbf{vntol}$
  - **Absolute voltage error tolerance** : default **vntol** = 1uA

Reference about How SPICE solve a circuit

- Modified Nodal Analysis
  - https://en.wikipedia.org/wiki/Modified_nodal_analysis
- Qspice forum discussion
  - https://forum.qorvo.com/t/current-imbalance-with-behavioural-sources/19312/7

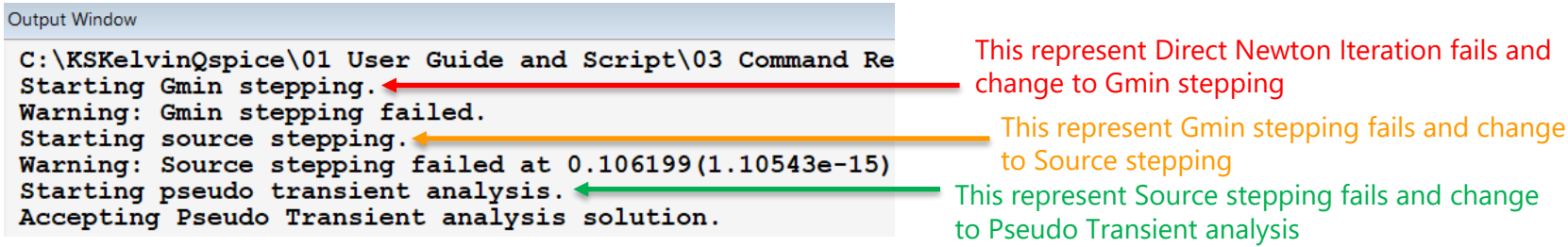
.option – NoOpIter, GminSteps, SrcSteps (DC Solution Strategy)

- NoOpIter, GminSteps, SrcSteps
  - NoOpIter : Go directly to Gmin Stepping (**Default NoOpiter=(not set)**)
  - GminSteps : Number of Gmin steps (**Default GminSteps=10**)
  - SrcSteps : Number of Source steps (**Default SrcSteps=10**)
  - These options are related to finding the DC operation point
- .OP – Find the DC Operation Point
  - DC operation point (.op) is usually performed as part of another analysis (e.g. .ac, .tran [without UIC]) in order to find the operating point of the circuit
  - But there is no guarantee that the operating point of a general nonlinear circuit can be found with successive linear approximations as is done in Newton-Raphson iteration
  - Method to find an operating point is in this sequence

Method	Directive to Disable
1. Direct Newton Iteration	.option NoOpiter
2. Adaptive Gmin Stepping	.option GminSteps=0
3. Adaptive Source Stepping	.option SrcSteps=0
4. Pseudo Transient	

.option – NoOpIter, GminSteps, SrcSteps (DC Solution Strategy)

- Example in Output Window
  - If without any warning message, represent Direct Newton Iteration can successfully find operation point



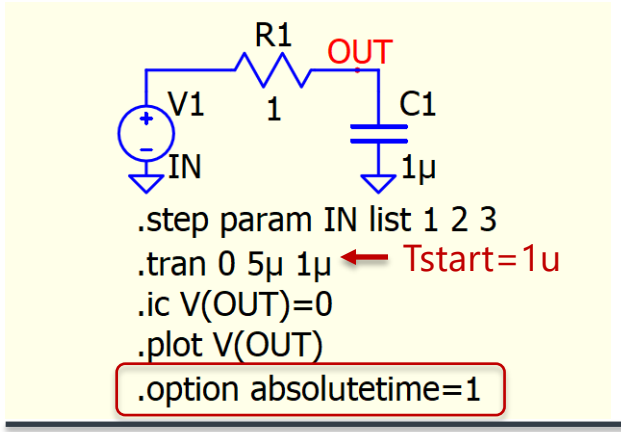
.option – NoOpIter, GminSteps, SrcSteps (DC Solution Strategy)

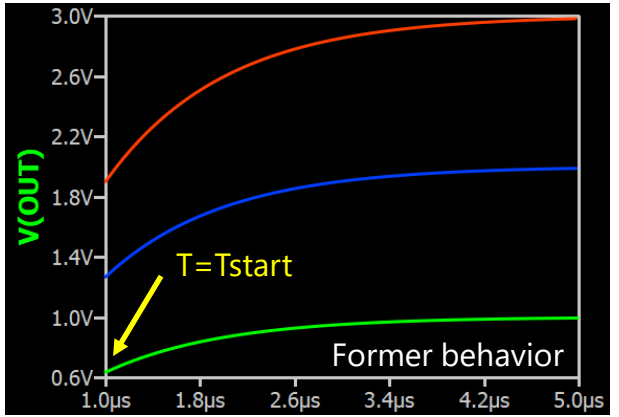
- Quote from Mike Engelhardt about Gmin Stepping and Source Stepping
  - Gmin Stepping
    - Gmin stepping is a DC solution strategy and is poorly named, as it has nothing to do with Gmin (minimum conductivity)
    - If the solver can't find the solution of the original circuit with Direct Newton Iteration, start with a related circuit that it can find the solution to. The circuit is the original but with a conductivity from every node to ground. Like an Ohm(or even less). Odds are, the solver can find the solution of the circuit if every node is shorted to ground. Give the solution for the circuit with every node shorted to ground, use that solution as a starting point for the Newton iteration when instead of 1 Ohm to ground but, say 1.1 Ohm to ground. If that succeeds, keep decreasing the conductivity until it's some very small number, like GMIN. Then remove the conductivity entirely and see of the solution can be found with Newton iteration using the prior solution as a starting point.
    - Refinements to the algorithm entail (i) first increasing the initial conductively until a first solution can be found and (ii) adapting the change in conductively between steps to ensure that a solution is kept as the conductivity is reduced
    - GMIN stepping is extremely effective for ICs or most any physical circuit
  - Source Stepping
    - GMIN stepping is not so good for non-physical macromodels like the TI OpAmp models, Source stepping is often better
    - Source stepping starts with the circuit turned off and then gradually increases the supply voltages
    - There are three types of source stepping
      - DC source stepping - does not take any hints from the reactance in the solution
      - Pseudo transient - take hints from the reactance in the solution
      - .tran with the sources written as PWLs starting a 0

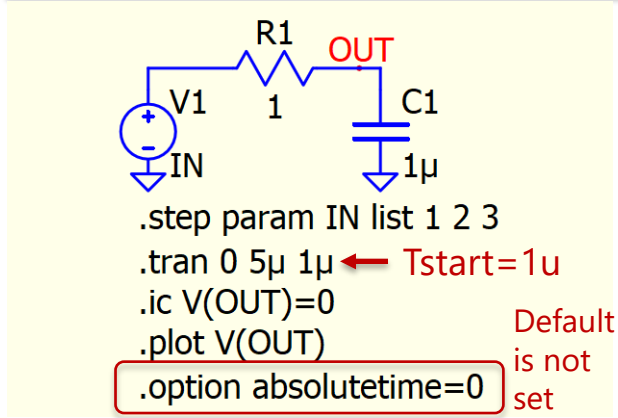
.option – AbsoluteTime (example of .tran with Tstart)

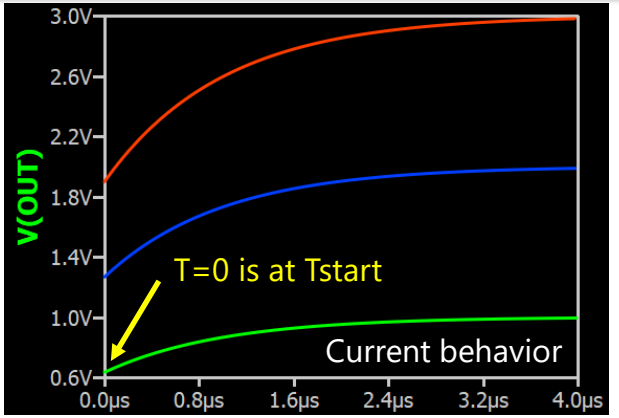
Qspice : .option - absolutetime (Tstart).qsch

- Absolute Time
  - Don't reset t=0 at the beginning of saving waveform data
  - **Default : (not SET) OR ABSOLUTETIME=0**
  - Revision History
    - 05/19/2025 T=0 is now set to the beginning of saving waveform data. To revert to the former behavior, add ".options ABSOLUTETIME" or ".options ABSOLUTETIME=1"
  - e.g.: .tran with Tstart





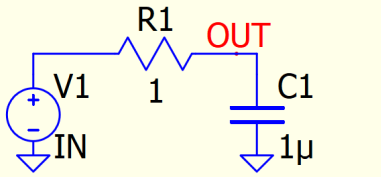




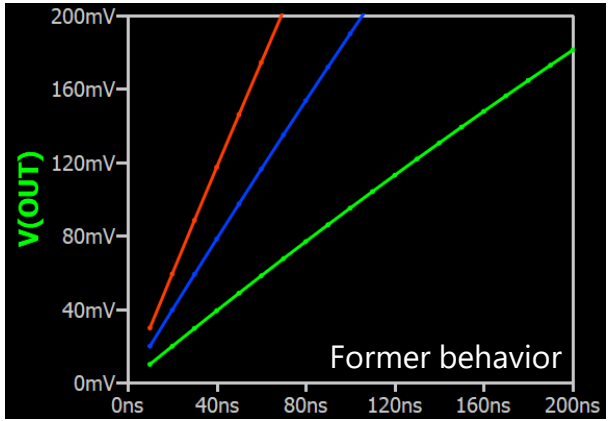
.option – AbsoluteTime (example of .tran with UIC)

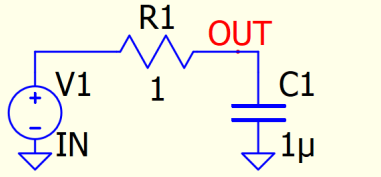
Qspice : .option - absolutetime (UIC).qsch

- Absolute Time
  - e.g.: .tran with UIC
    - .tran with UIC employs an incorrect initial condition to start the simulation, data at t=0 is absent. Qspice has modified its behavior by offset the time by one step toward t=0s



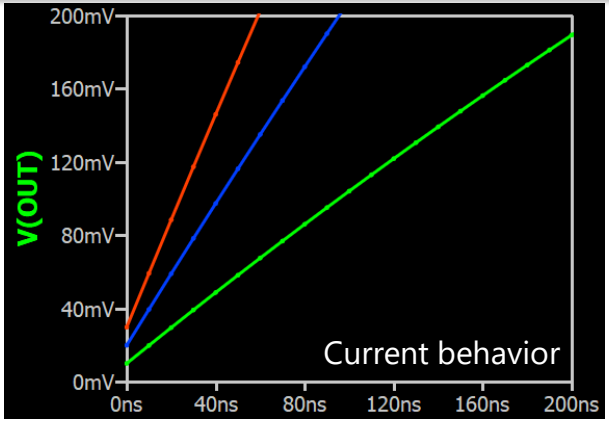

```
.step param IN list 1 2 3
.tran 10m uic
.option maxstep=10n
.plot V(OUT)
.option absolutetime
```






```
.step param IN list 1 2 3
.tran 10m uic
.option maxstep=10n
.plot V(OUT)
.option absolutetime
```

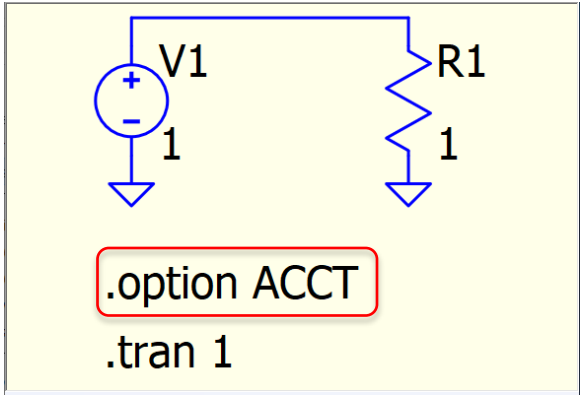
Default is not set



.option – ACCT (Print accounting information)

Qspice : option - ACCT.qsch

- ACCT
  - Print accounting information
  - **Default ACCT : (not SET)**



Output Window

```
C:\KSKelvinQspice\01 User Guide and Script\03

Total elapsed time: 0.0134455 seconds.
Circuit Equations: 2
Total iterations: 2081
Total Analysis Time: 5.433ms
Load time: 171.3us
Matrix reordering time: 5.2us
L-U decomposition time: 2.1us
Matrix solve time: 95.7us
Operating temperature: 27°C
Nominal temperature: 27°C
Transient iterations: 2076
Transient iters per point: 0
Transient timepoints: 1039
Accepted timepoints: 1039
Rejected timepoints: 0
Transient time: 1.506ms
Transient L-U decomp time: 84.4us
Transient solve time: 95.5us
Integration Method: trap
```

.option – CAPOP (MOSFET Charge/Capacitance Model)

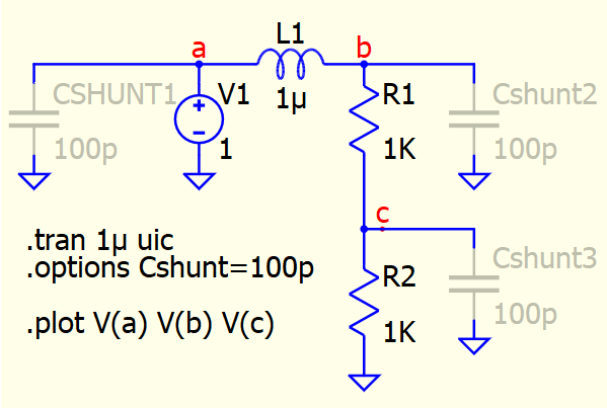
- CAPOP
  - Set to override model value in MOSFET native model statement
  - **Default : Not Set**
  - 0: Use Meyer Capacitance model
  - 1: Use BSIM1 Charge model

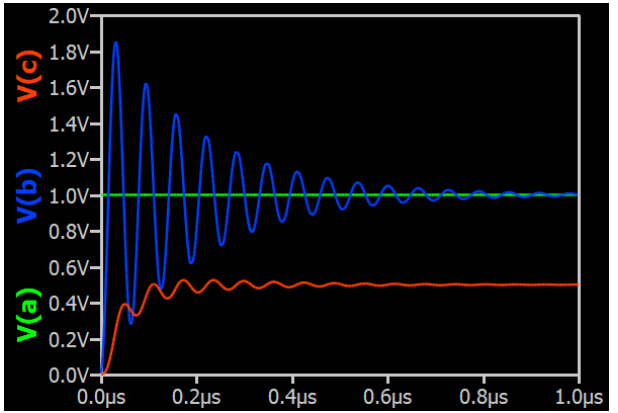
Charge Model	Qspice		LTspice	Pspice
	.option (ifused, override .model)	.model		
SPICE Meyer Capacitance Model	CAPOP=0	CAPOP=0 (default)	Not Support	Not Support
BSIM1 Charge Model	CAPOP=1	CAPOP=1	(default)	(default)

.option – Cshunt and Gshunt (Capacitance and Conductance Shunt)

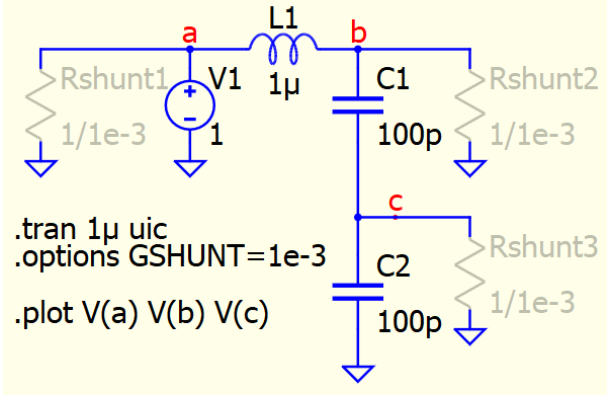
Qspice : option - CSHUNT.qsch ; option - GSHUNT.qsch

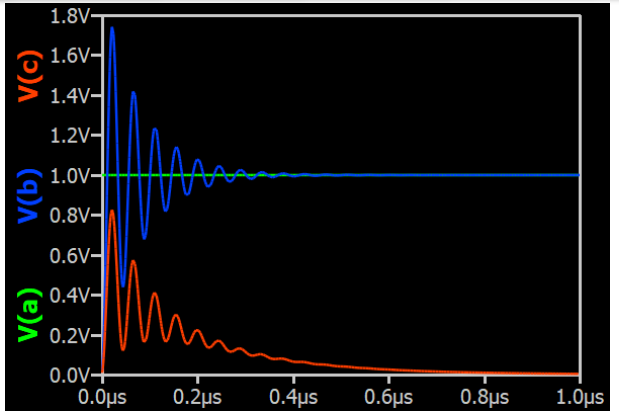
- Cshunt
  - Capacitance added from every node to ground (aka CMIN)
  - **Default CSHUNT=0F**
  - Example to explain
    - Cshunt is equivalent to add Cshunt1/2/3 in node a/b/c





- Gshunt
  - Conductance added from every node to ground
  - **Default GSHUNT=0℧**
  - Example to explain
    - Gshunt is equivalent to add Rshunt1/2/3 = $\frac{1}{\text{GSHUNT}}$ in node a/b/c

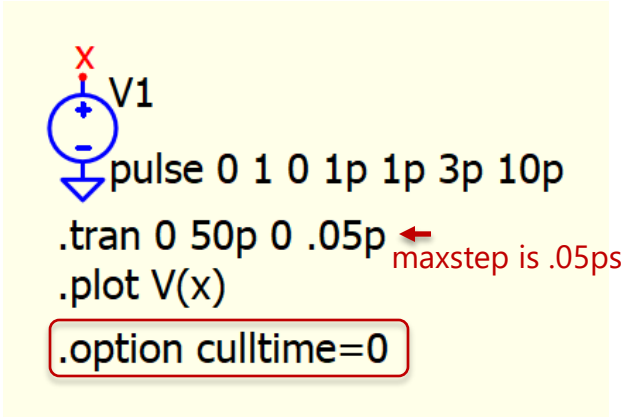


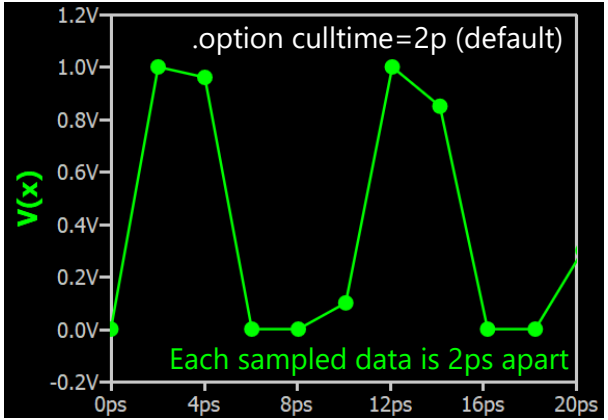


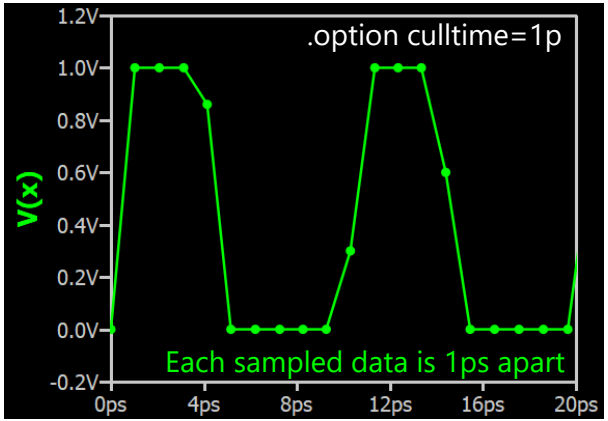
.option – CullTime (Minimum time to elapse for data saving)

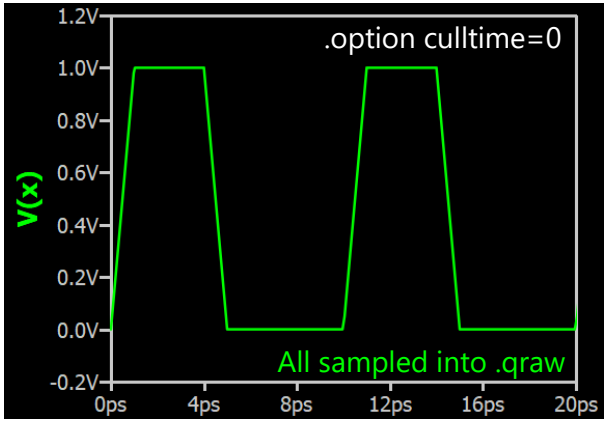
Qspice : option - CullTime.qsch

- CullTime
  - Minimum time to elapse before dumping transient analysis data to disk
  - **Default CULLTIME=2ps**
  - In default, Qspice only updates the .qraw file if at least 2ps have passed since the last update
  - This option is useful for simulating picosecond-level applications
  - ** CullTime doesn't affect actual simulation step size, but only the minimum sampling time between two samples that are written into the .qraw data file





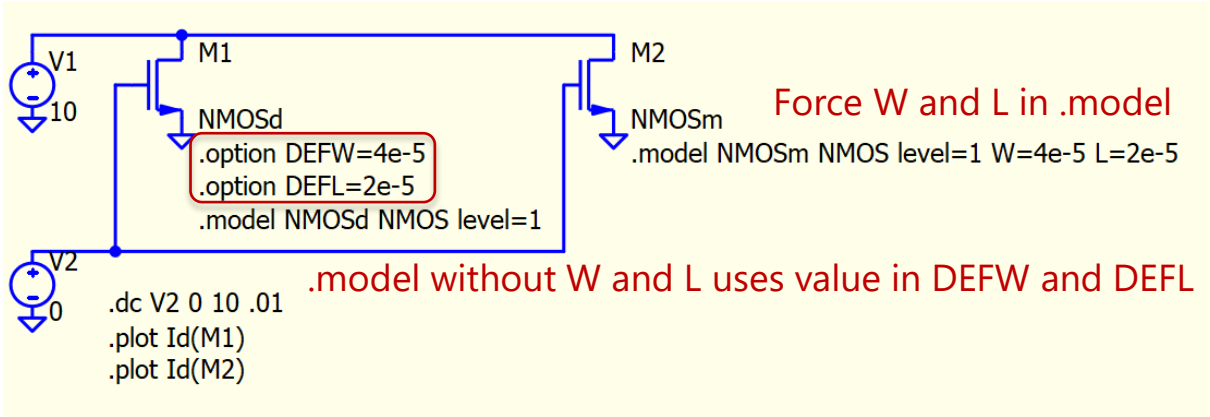


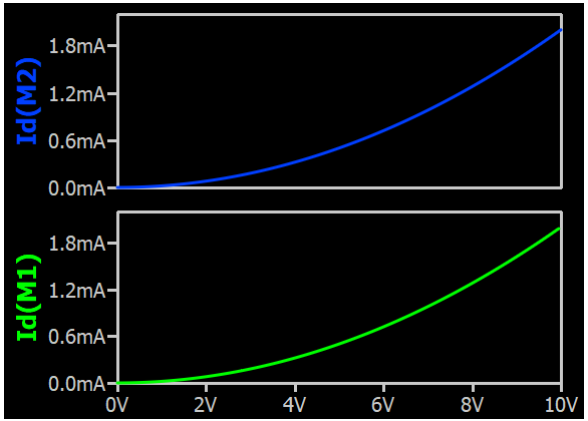


.option – DEFL, DEFW (Default MOSFET Length and Width)

Qspice : .option - DEFL DEFW.qsch

- DEFL, DEFW
  - DEFL : Default MOSFET Length
  - DEFW : Default MOSFET Width
  - **Default DEFL=100um**
  - **Default DEFW=100um**

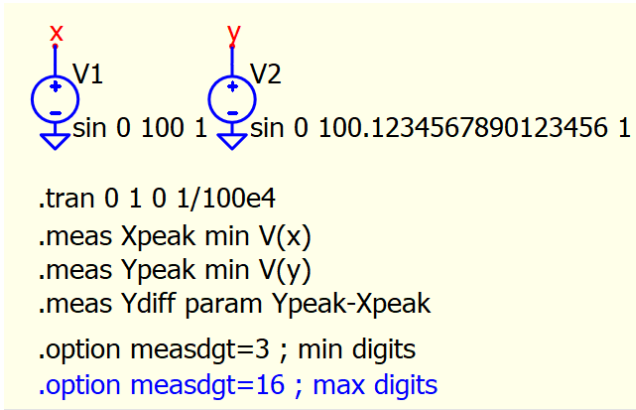


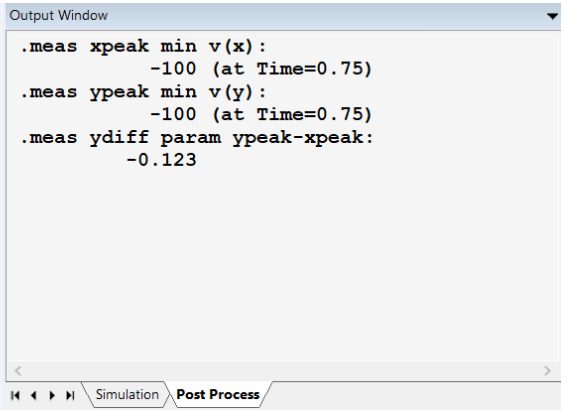


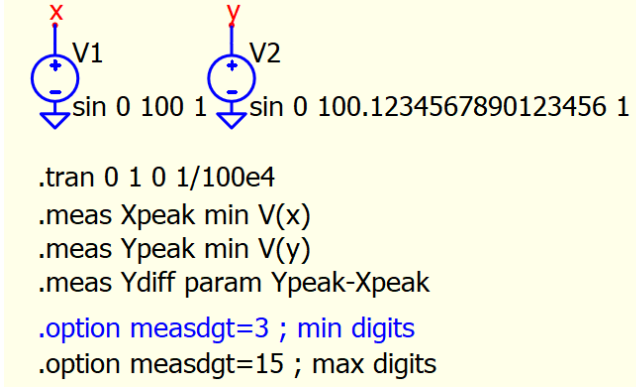
.option – MEASDGT (number of significant figure in .meas)

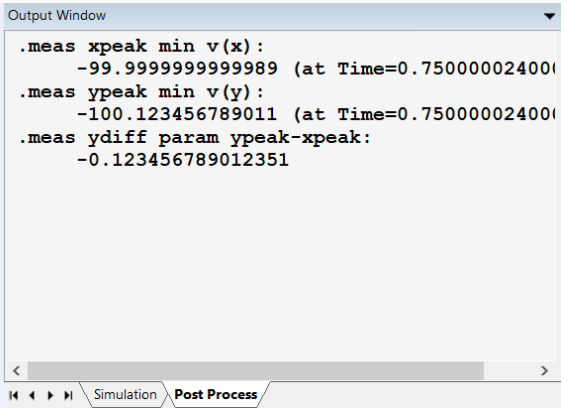
Qspice : .option - measdgt.qsch

- MEASDGT
  - Number of significant figures for **.meas**ure statement output
  - **Default measdgt=6**
  - measdgt range from 3 to 15



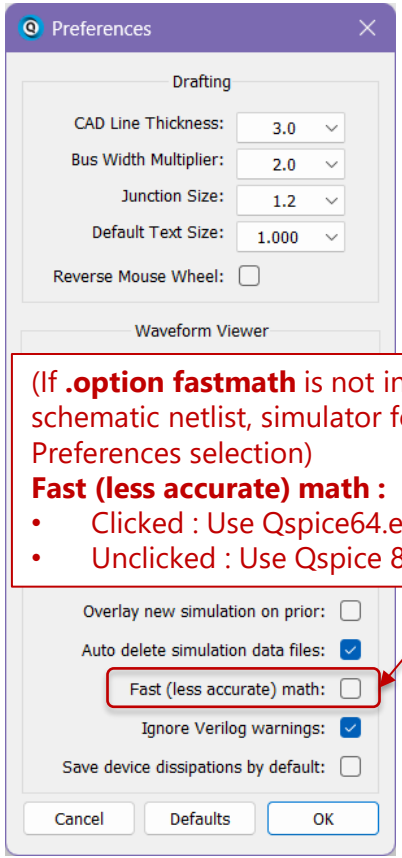






.option – FastMath (64-bit or 80-bit math solver)

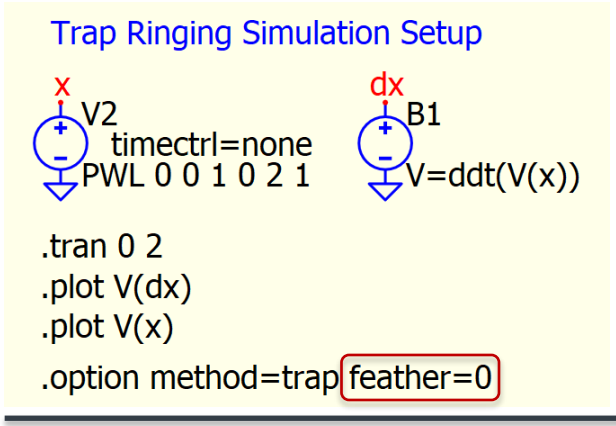
- FastMath
  - Annotates the simulation with the preferred solver
  - Switches between using QSPICE64.exe and QSPICE80.exe
  - On Intel hardware, there are three types of floating points:
    - float 32bit precision
    - double 64bit precision
    - long double 80bit precision
  - Both QSPICE64 and QSPICE80 utilize a mix of 64-bit and 80-bit math. QSPICE80 utilizes 80-bit math more, but runs slower
  - This setting can be adjusted in preferences if the GUI is used or via a netlist command
    - .option fastmath=1 (or true) : use Qspice64.exe as solver
    - .option fastmath=0 (or false): use Qspice80.exe as solver
    - Netlist command override Preferences selection
  - Users can verify the solver being used by examining the .qraw line #8 with a text editor

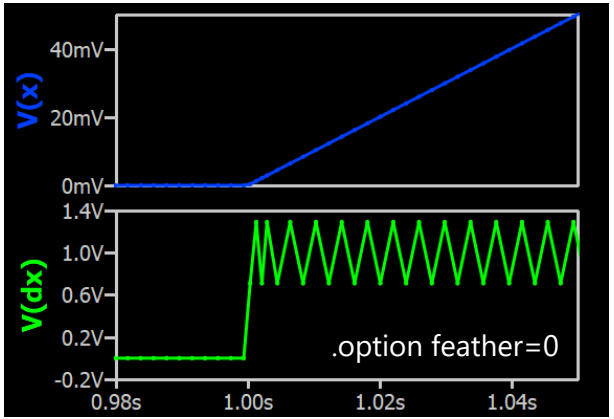


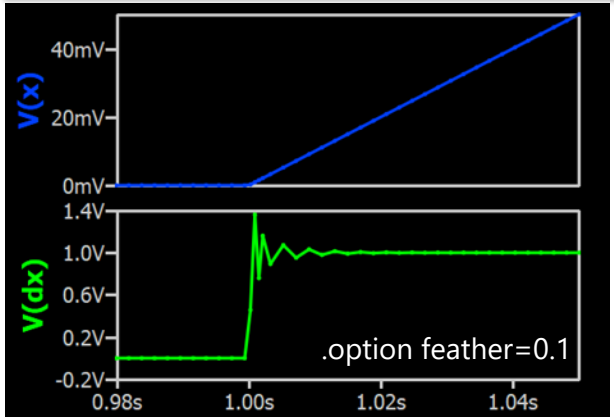
.option – Feather (Trap Integration Damping Factor)

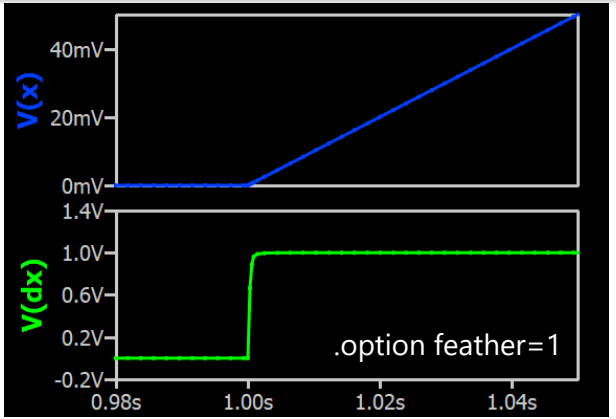
Qspice : option - Feather.qsch

- Feather
  - Trap Integration Damping Factor
  - **Default Feather=0**
  - If trapezoidal ringing is observed when using trapezoidal integration, the damping factor can be employed to reduce it
  - For more information about trapezoidal ringing, refer to section ***.option – Method : Trapezoidal Ringing***









.option – Feather (Trap Integration Damping Factor)

- Mike Engelhardt comment on **.option Feather**

Engelhardt Jul 2023

FEATHER is an experimental parameter. It can be used to duplicate the de-tuned trap integration of HSPICE. It's use is not recommended.

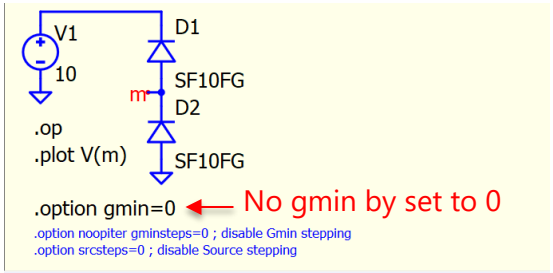
As far as trap ringing, I realize it's disconcerting, but in a sense it's giving the right answer: The area under each trapezoid is correct, so one knows that the differential equations are correctly integrated. It can be reduced by either (i) using ".options method=Gear" (ii) stipulating a lower trtol, or (iii) stipulating a smaller maximum timestep. Gear is not recommended because it adds a substantial artificial damping to the circuit. I know of two cases where the use of Gear integration let an IC designer believe his circuit was stable until silicon said different and a turning Gear off confirmed.

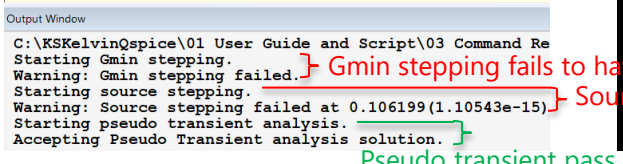
Unlike some prior art, I don't smooth trap ringing out it out at all because one needs to know what the gates and flop truly see.

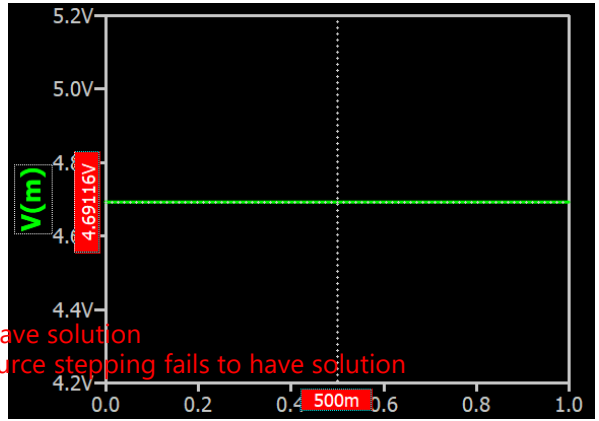
.option – Gmin (Minimum conductance)

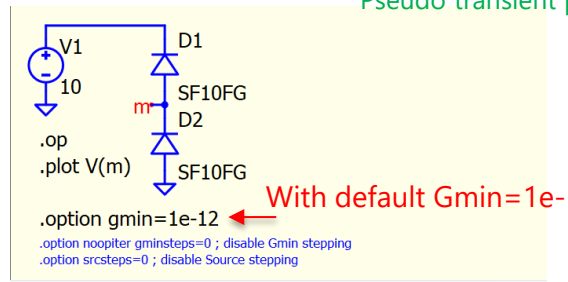
Qspice : option - Gmin.qsch

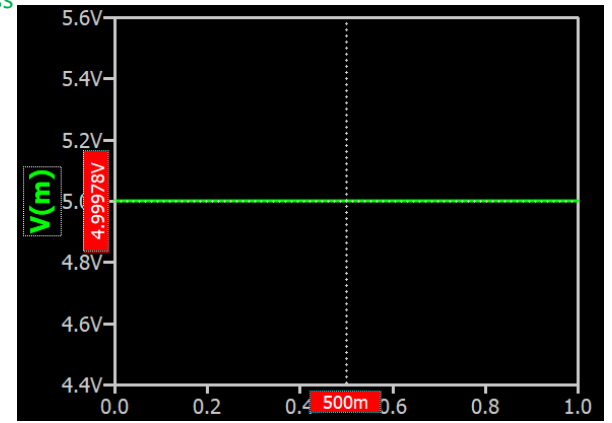
- Gmin
  - Minimum conductance
    - Default Gmin=1e-12℧
    - Minimum conductivity that is added in parallel to every PN junction (diode, JFET, bipolars, MOSFET substrate diodes)
    - Reason : As double precision math isn't accurate enough to find the bias point of two diode in series when reversed bias
  - Example
    - In 1st example, without Gmin, Gmin stepping and Source stepping both failed to calculate solution of .op







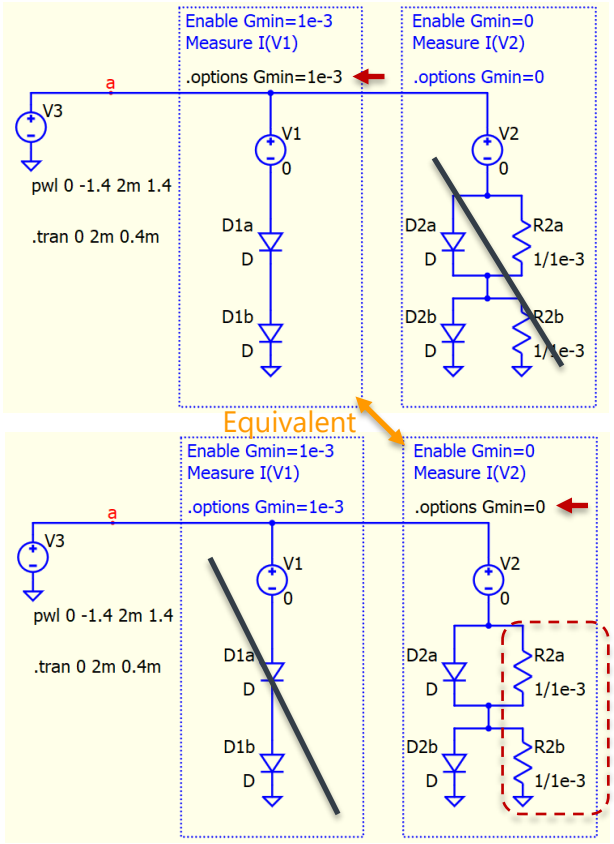


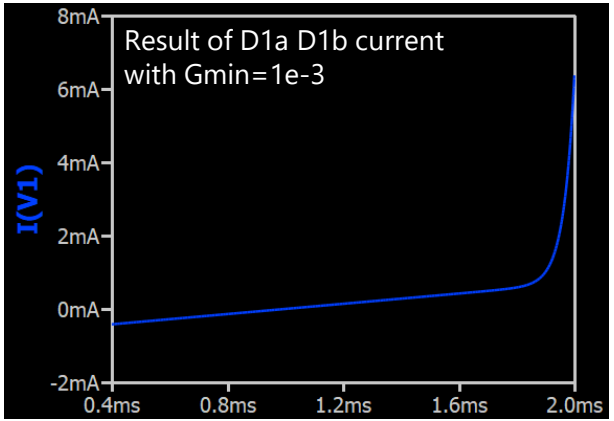


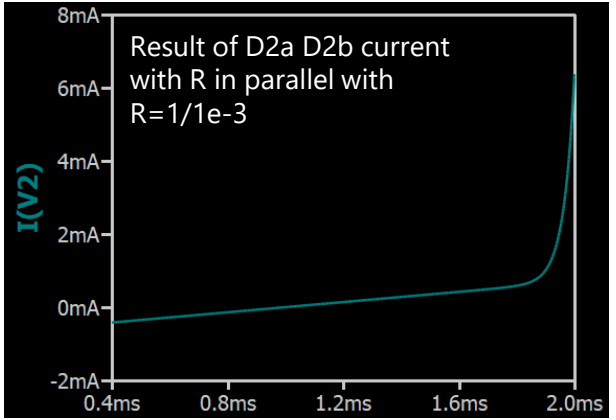
.option – Gmin (Minimum conductance)

Qspice : option - Gmin Diode.qsch

- Effect of Gmin
  - Examples to demonstrate Gmin is equivalent to add shunt conductance for every PNjunction
  - First example uses Gmin=1e-3 and measure I(V1) profile of D1a/D2a
  - Second example force Gmin=0 (no effect of Gmin) and added R2a = R2b = $\frac{1}{1e-3}$, and measure I(V2) profile of D2a and D2b



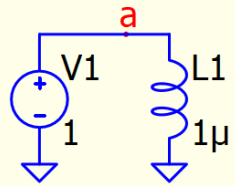




.option – Gmin (Minimum conductance)

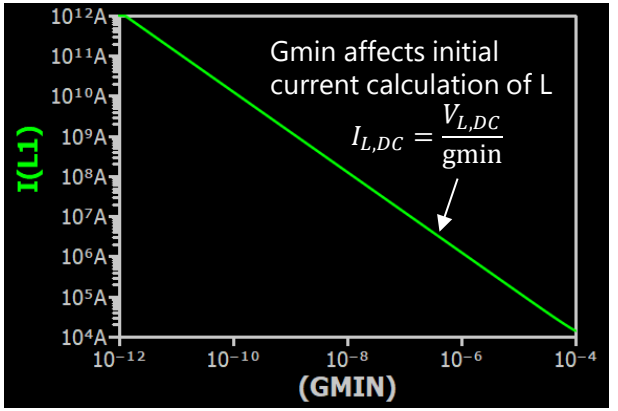
Qspice : option - Gmin L (.op).qsch ; option - Gmin L (.tran).qsch

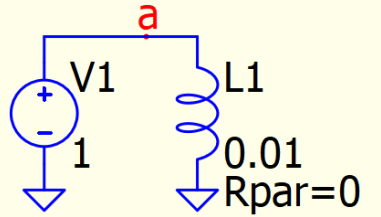
- Gmin in Inductor
  - Gmin is also applied to the inductor in .op and .tran for initial inductor current calculation
  - However, gmin, despite its definition as a conductance, functions as a series resistance value utilized for calculating the initial inductor current
  - Unlike PN junctions, gmin is solely employed in the initial current calculation and is removed during transient analysis



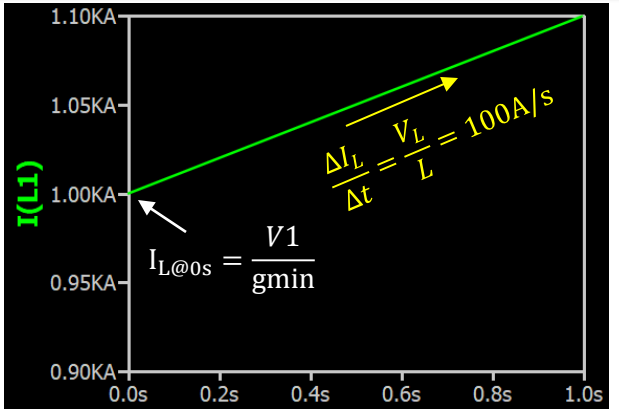

```
.step dec param gmin 1e-12 1e-4 10
.options GMIN=gmin
.plot I(L1)
.op
```

DC Operation Point Analysis






```
.options gmin=1e-3
.plot I(L1)
.tran 1
```

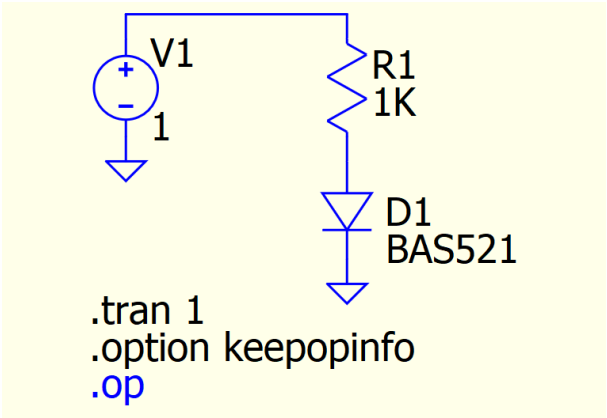


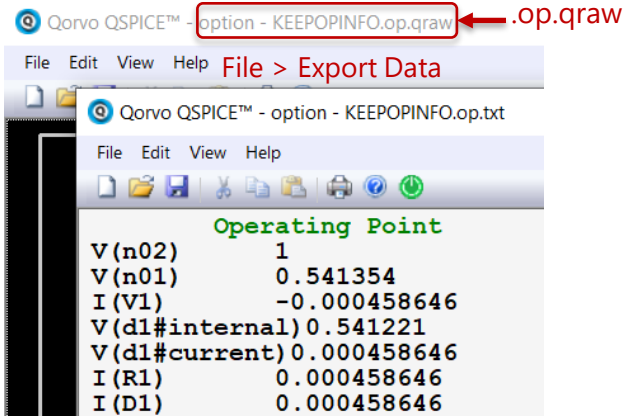
.option – ITL1, ITL2 and ITL4 (Iteration Limit for Simulation Directive)

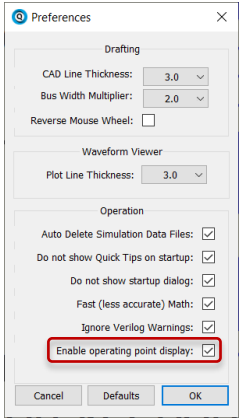
- ITL1 : DC iteration limit – used in .op
  - Default ITL1=100
- ITL2 : DC transfer curve iteration limit – used in .dc
  - Default ITL2 = 50
- ITL4 : Transient analysis iteration limit – used in .tran
  - Default ITL4 = 10

.option – KEEPOPINFO (Record .OP Info)

- KeepOpInfo
  - Record operation point into .qraw for .ac and .tran
  - A file with extension .op.qraw is forced to create
    - This .op.qraw file contains DC operating point data
    - Open .op.qraw file with waveform viewer, a list of operating point is generated
  - .option keepopinfo is equivalent to use
    - Edit > Preferences
    - Click "Enable operating point display"
      - Hover device to display data only work with this enable



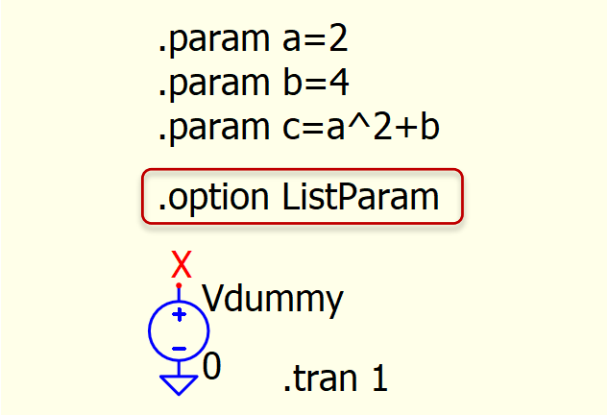




.option – ListParam (Print a list of evaluated parameters)

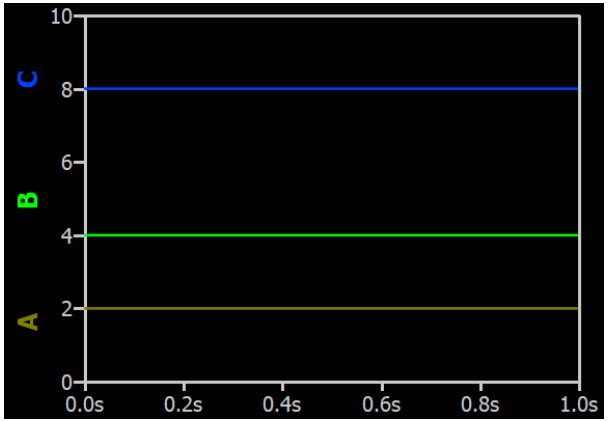
Qspice : .option - ListParam.qsch

- ListParam
  - Print a list of the evaluated parameters (also sets SAVEPARAM)
  - This option can return a list of parameter values in the console output
  - It also sets the .option SaveParam, where all parameter values are saved into the output data file .qraw, which can be plotted in the waveform viewer for post-processing



Output Window

```
   --- Parameter Evaluations ---
TEMP          = 27          "CKTTEMP"
A             = 2           "2"
B             = 4           "4"
C             = 8           "A^2+B"
```



.option – MAXORD (Maximum Integration Order)

- Maxord
  - Maximum integration order : determine integration method in solver
  - **Default MAXORD=2**

.option MAXORD	.option METHOD	Integration Method	Comment
1	[Ignore]	Backward Euler	Less accurate than Gear [If .option maxord=1, force to backward euler and ignore .option method]
2 (Default)	Gear	Gear	It dampens all ringing, not just numerical ringing but even physical ringing
	Trap (Default)	Trapezoidal	Trap integration is recommended but can give rise to trap ringing and cause the user to be suspicious of the correctness of the simulator even through each trapezoid contains the correct integrated area

.option – Method (Integration Method : Trapezoidal or Gear)

- Method
  - Integration method : Trapezoidal or Gear
  - **Default METHOD = Trapezoidal**
  - Trapezoidal (.option method=trap) : Area under every trapezoid is correct which can yield accurate results
    - One disadvantage of using the trapezoidal method is that it can lead to trapezoidal ringing when working with unrealistic circuit elements
    - In LTspice, trap ringing is smoothed out (method : modified trap), but in Qspice, one needs to know what the gates and flop truly perceive in order to address it
  - Gear (.option method=gear) : Adds damping that isn't present in the circuit and does not provide as exact of a result. Generally, it is slower and less accurate compared to the trapezoidal method.
    - This is the default method in Pspice
  - Reference article : SPICE Differentiation by Mike Engelhardt in Jun 19 2015
    - https://www.analog.com/en/resources/technical-articles/spice-differentiation.html

.option – Method (Integration Method : Trapezoidal or Gear)

Mike Engelhardt comment in Qspice forum

- https://forum.qorvo.com/t/need-guidance-on-qspice-integration-method-and-this-feather-parameter/14393
- https://forum.qorvo.com/t/apparent-kirchhoff-s-law-violation/15048/2
- https://forum.qorvo.com/t/gear-vs-trap-what-are-those-and-how-to-optimize/16431
- https://ltwiki.org/files/LTspiceHelp.chm/html/integration_method_issues.htm

Engelhardt 26m

Aside from a few exceptions, Gear is both slower and less accurate than trap. The main problem with trap is a ringing artifact that bothers people, even though the area under each trapezoid is correct.

You've discovered the main point to offering both in the same simulator: if you get the same answer using both methods, you know that the solution is not affected by integration artifacts.

–Mike

Engelhardt Jul '23

FEATHER is an experimental parameter. It can be used to duplicate the de-tuned trap integration of HSPICE. It's use is not recommended.

As far as trap ringing, I realize it's disconcerting, but in a sense it's giving the right answer: The area under each trapezoid is correct, so one knows that the differential equations are correctly integrated. It can be reduced by either (i) using ".options method=Gear" (ii) stipulating a lower trtol, or (iii) stipulating a smaller maximum timestep. Gear is not recommended because it adds a substantial artificial damping to the circuit. I know of two cases where the use of Gear integration let an IC designer believe his circuit was stable until silicon said different and a turning Gear off confirmed.

Unlike some prior art, I don't smooth trap ringing out it out at all because one needs to know what the gates and flop truly see.

Engelhardt Aug '23

That looks like trapezoidal ringing - a situation where the numerically-integrated solution oscillates about the true solution timestep to timestep. A side effect is that current monitoring can look off[1].

In a sense, it's giving the correct answer, in that the area under each trapezoid has the correct area, but it is certainly disconcerting.

You can reduce or eliminate it by some of a combination of these methods:

1. Reduce trtol(.options trtol=1)
2. Reduce the maximum allow time step size(4th number on the .tran command)
3. Change to Gear integration.(.options method=Gear)

The last option is the most empathic way of getting rid of it, but it also introduces the greatest error. A simulation done with Gear integration can look stable when in fact, it is not. I've seen IC designers go to silicon only to learn they made the mistake of using Gear integration to check the IC's operation.

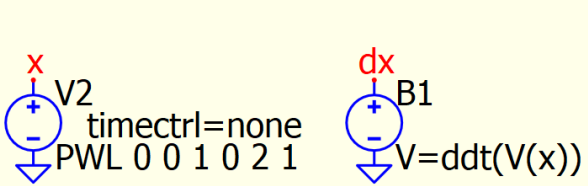
–Mike

1] Current monitoring is not really part of the solution of the circuit. It's the node voltages that are solved for, except for the voltage sources where the currents are part of the solution. Current monitoring is done as a forensic analysis of the circuit. Current reporting can be in error even if the rest of the solution is correct. I'll fix errors in report as they come up, e.g., today I fixed an issue in capacitor current reporting.

.option – Method : Trapezoidal Ringing

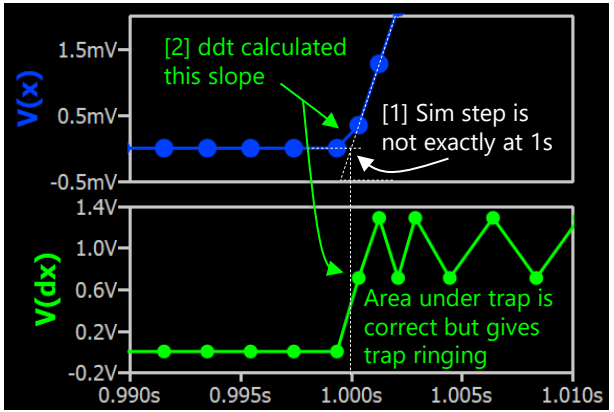
Qspice : option - Method (trap ringing).qsch

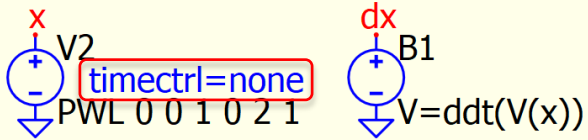
- Trapezoidal Ringing
  - Trapezoidal method accurately calculates the area under each trapezoid
  - However, trapezoidal ringing can occur when dealing with unrealistic circuit elements or when the function being integrated has sudden changes or contains high-frequency components
  - This example demonstrates how trapezoidal ringing occurs in a simulation when using a ramp source and a derivative function
  - The first example illustrates the occurrence of trapezoidal ringing, while the second example forces the simulation step to the exact break point without any trapezoidal ringing




```
.option maxstep=1.94e-3
.tran 0 2
.plot V(dx)
.plot V(x)
```

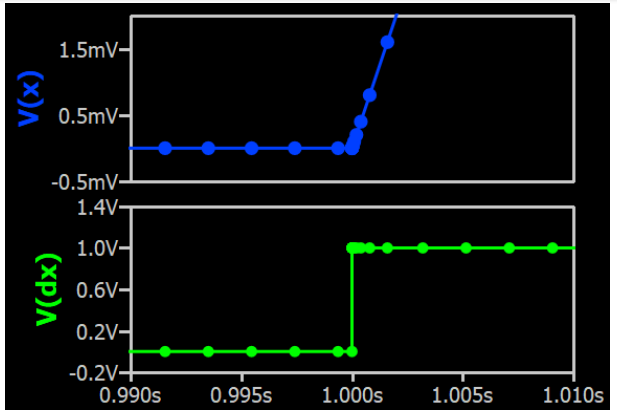
In this setup, if you set different maxstep, you can get different magnitude of trap ringing






```
.option maxstep=1.94e-3
.tran 0 2
.plot V(dx)
.plot V(x)
```

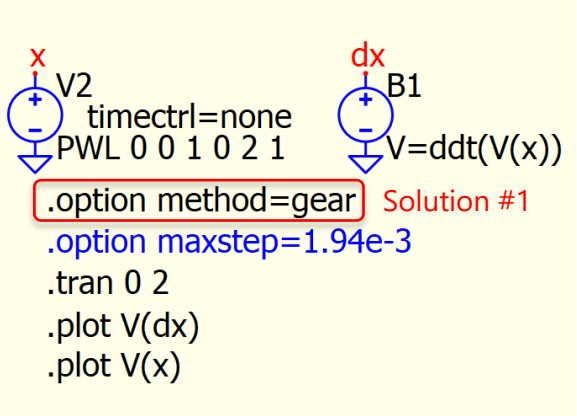
Enable timectrl for Vsource Qspice simulation step can happen at exact break point (i.e. at 1s)

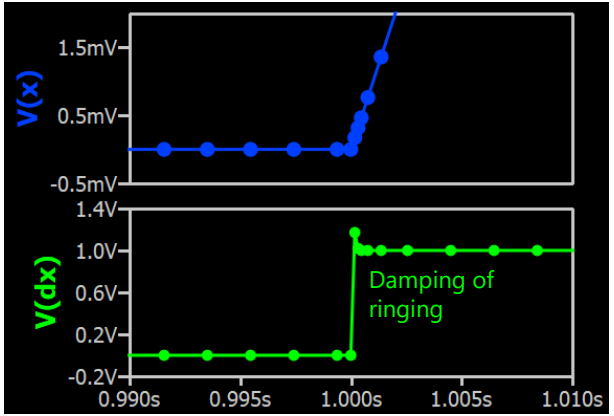


.option – Method : Trapezoidal Ringing

Qspice : option - Method (trap ringing).qsch

- Trapezoidal Ringing (Solution)
  - #1 : Gear integration (.option method=gear) introduces artificial damping that is not present in the circuit, effectively reducing ringing during integration
    - However, this approach is not generally recommended as it does not provide exact results
  - #2 : Reduce trtol (.option trtol)
  - #3 : Reduce maximum timestep (.option maxstep) or use a device with TTOL to reduce temporal timestep when approach trap ringing
  - #4 : Damping factor can be added when trap is used (but not recommended), refer to section **.option – Feather**

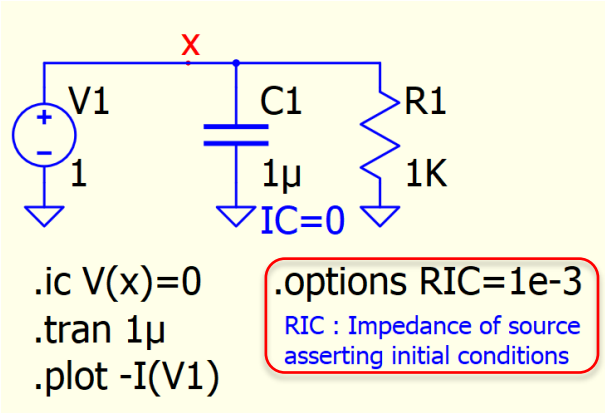


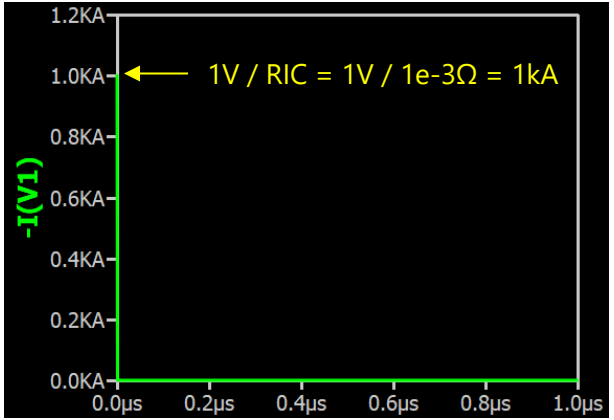


.option – RIC (Impedance of source asserting initial conditions)

Qspice : option - RIC C.qsch

- RIC
  - Impedance of source asserting initial conditions
  - Default RIC=1mΩ
  - In this example, initial voltage of V(x) is defined at 0V. A 0V capacitor connected in parallel to a voltage source will force an infinite current in first time step. RIC is source impedance inserted in source for initial condition calculation purpose. In this case, source current at t=0 becomes $I_{V1} = \frac{V_{V1}}{RIC}$





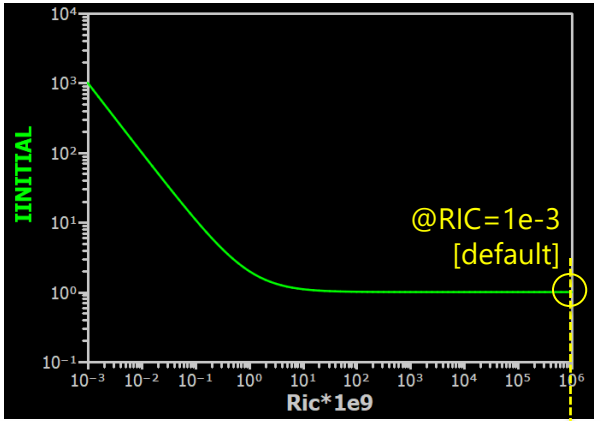
.option – RIC (Impedance of source asserting initial conditions)

Qspice : option - RIC L.qsch

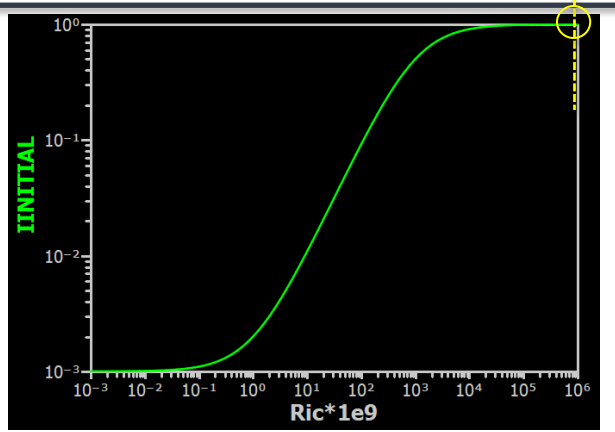
- RIC
  - Impedance of source asserting initial conditions
  - Inductor currents are asserted with the compliance of 1e9 * RIC
  - Default RIC=1mΩ

- Important note
  - RIC only affect inductor current if .ic is used to define inductor initial current
  - In this simulation example, initial inductor current is plotted with Gmin=1e-12 and Gmin=1e3 with .ic I(L1)=1
    - When RIC=1e-3 (default), initial current is always equal .ic defined value

```
.options Gmin=1e-12
.options RIC=RIC
.step dec param RIC 1e-12 1e-3 10

V1
1
L1
1µ  Rpar=0

.ic I(L1)=1
.plot I(L1)
.tran 1n
.meas Iinitial find I(L1) at 0
```



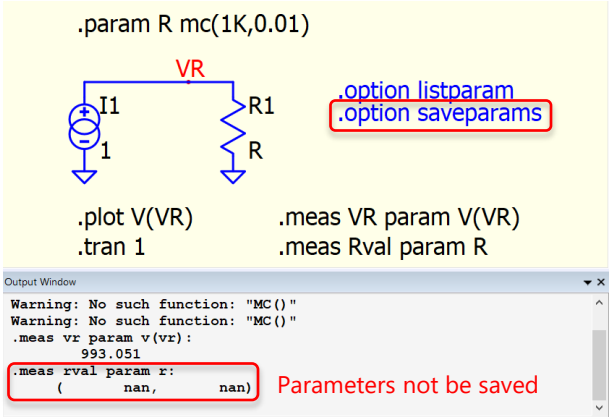

```
.options Gmin=1e3
.options RIC=RIC
.step dec param RIC 1e-12 1e-3 10

V1
1
L1
1µ  Rpar=0

.ic I(L1)=1
.plot I(L1)
.tran 1n
.meas Iinitial find I(L1) at 0
```

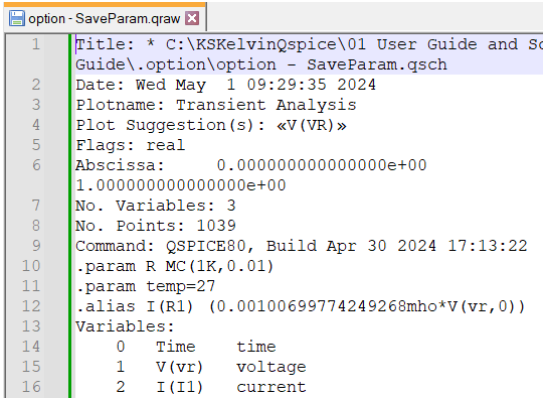


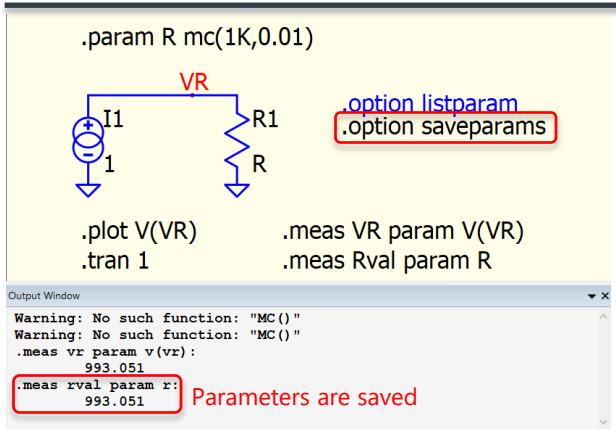
.option – SaveParam (Save .param into waveform data)

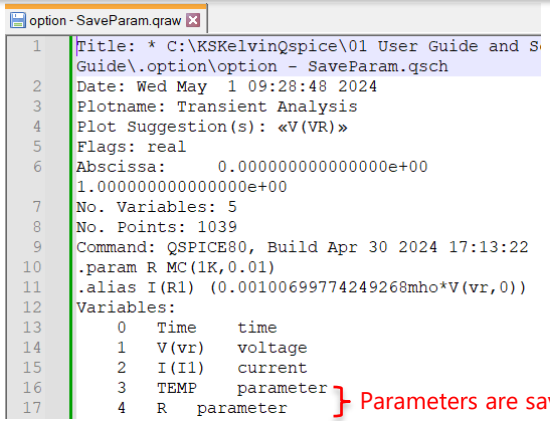
Qspice : option - SaveParam.qsch

- SaveParam
  - Include evaluated user-defined parameters as waveform data
  - In default, Qspice does NOT store user-defined parameters (.param) into output data file (.qraw)
  - If user-defined parameters are needed in .plot or .meas, enabling this option is necessary
  - Two ways to enable this
    - .option SaveParam
    - .option ListParam
      - This automatically sets SaveParam option, but as param is listed in Output Window, it will slow the simulation as display requires msleep()





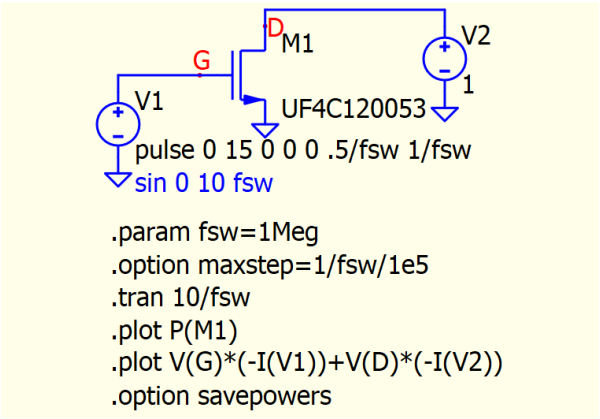


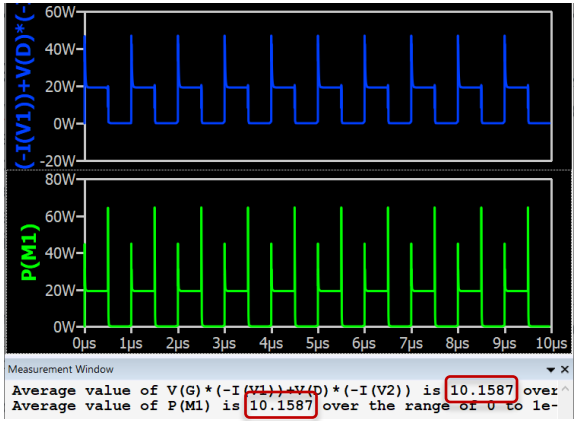


.option – SavePowers (Compute and save dissipation power)

Qspice : option - SavePower (nmos).qsch

- SavePowers
  - Compute and save the dissipation of components
  - Computes the true power dissipation while ignoring displacement currents
    - Implemented for BJTs, Capacitors, Diodes, Inductors, JFETs, MOSFET level 1, MOSFET level 2010 and VDMOS
  - P() will be available in plot expression, and P() represent dissipation power of entire component
    - e.g. both gate and drain loss for nmos



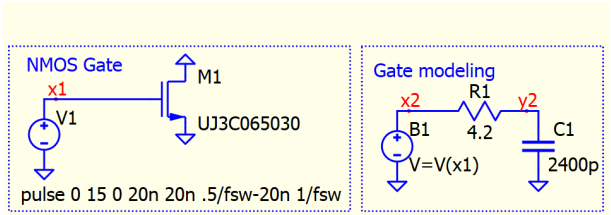


- NMOS example
  - In this example, the NMOS is the only dissipation device, and its power is equal to the input power from V1 and V2
  - Input Power = V(G) * -I(V1) + V(D) * -I(V2)
  - Qspice calculated NMOS power = P(M1)
  - Average power of Input Power and P(M1) are equal (but Instantaneous power profile are different!!)
    - It can conclude that P(M1) include gate and drain power dissipation!

.option – SavePowers (Compute and save dissipation power)

Qspice : option - SavePower (nmos-gate).qsch

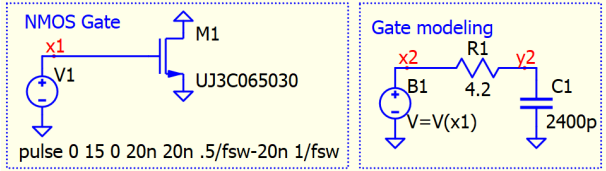
- Instantaneous Power
  - Example of gate power consumption. Instantaneous input power from V1 and dissipation power P(M1) is different, but their average power is the same
  - Gate model circuit R1 and C1 explains why instantaneous power profiles are different, as gate is not purely resistive but with reactive power in capacitance
  - Therefore, P(M1) in savepower option calculate instantaneous real power in device



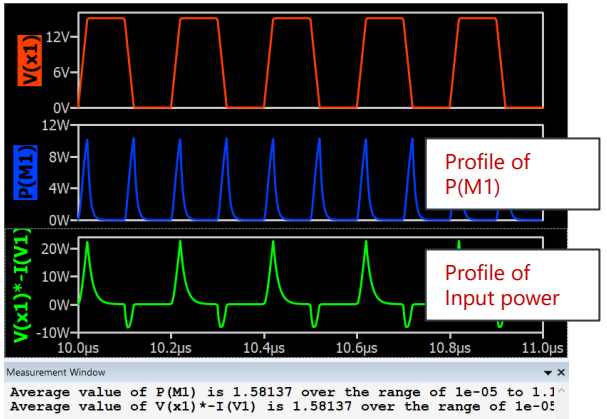

```
.param fsw=5Meg
.option savepower
.tran 0 55/fsw 50/fsw 1/fsw/1e4
.plot V(x1)*-I(V1)
.plot P(M1)
.plot V(x1)
.plot V(x2)*-I(B1)
.plot V(x2,y2)*-I(B1)
.plot V(x2)
```

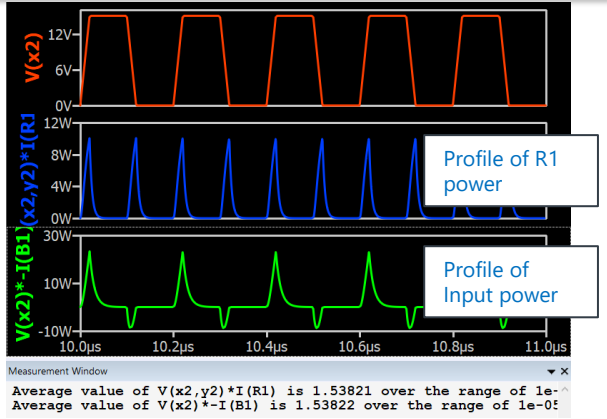
Same simulation

plots NMOS Gate or Gate modeling




```
.param fsw=5Meg
.option savepower
.tran 0 55/fsw 50/fsw 1/fsw/1e4
.plot V(x1)*-I(V1)
.plot P(M1)
.plot V(x1)
.plot V(x2)*-I(B1)
.plot V(x2,y2)*I(R1)
.plot V(x2)
```



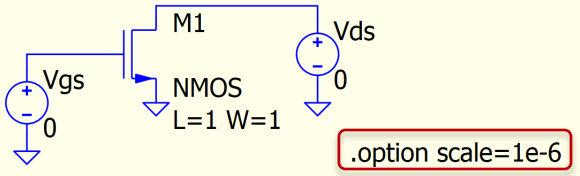


.option – Scale (Geometric element parameters scaling factor)

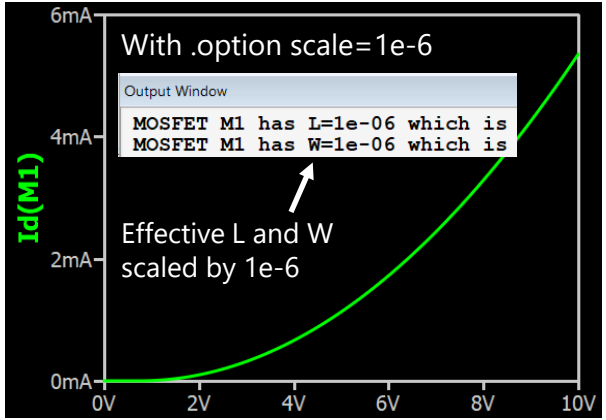
Qspice : option - Scale.qsch

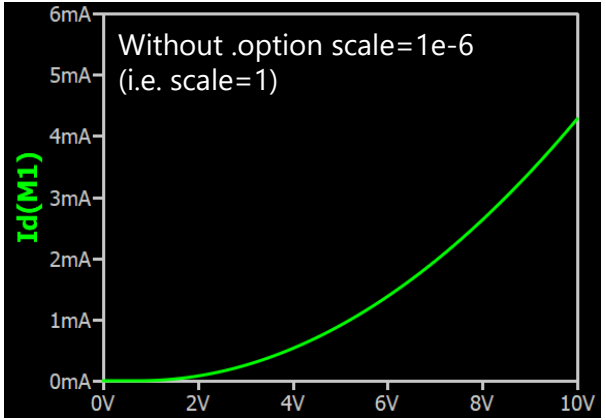
- Scale
  - This scaling factor applies to geometric element parameters and has a default unit of meters
  - **Default Scale=1**
  - An external reference in the Ngspice User's Manual on page 306, as quoted below

NGspice User's Manual : page 306 (.option scale)




```
.model nmos nmos (kp=100µ vto=0.75 LD=1e-7)
.dc Vgs 0 10 0.1 Vds list 10
.plot Id(M1)
```





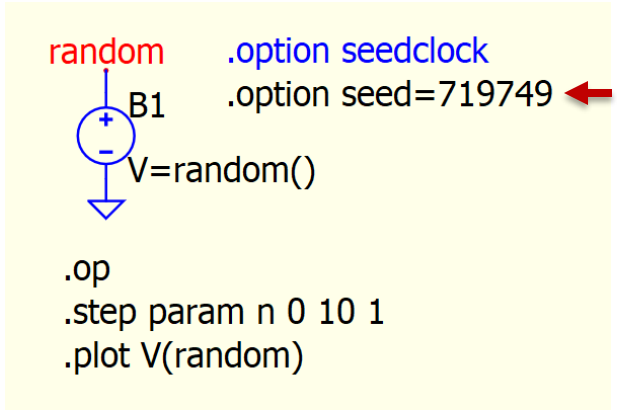
SCALE=x set the element scaling factor for geometric element parameters whose default unit is meters. As an example: `scale=1u` and a MOSFET instance parameter `W=10` will result in a width of $10\mu m$ for this device. An area parameter `AD=20` will result in $20e-12\ m^2$. Following instance parameters are scaled:

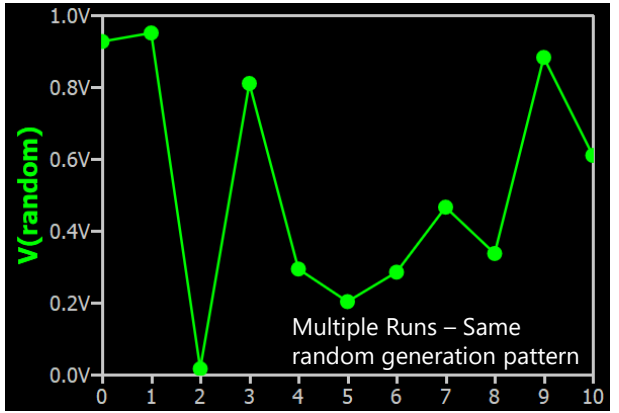
- Resistors and Capacitors: W, L
- Diodes: W, L, Area
- JFET, MESFET: W, L, Area
- MOSFET: W, L, AS, AD, PS, PD, SA, SB, SC, SD

.option – Seed and Seedclock (Seed of random number generation)

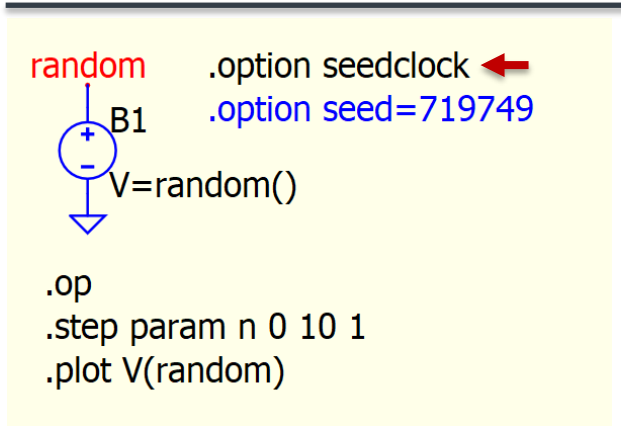
Qspice : option - Seed Seedclock.qsch

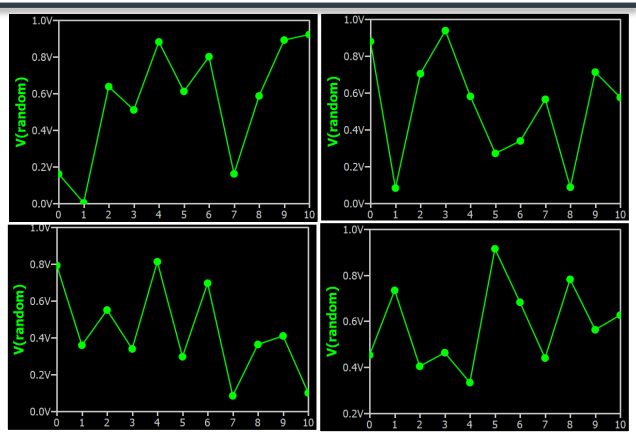
- Seed
  - Initialize the random number generator used in .param statements
  - Same random pattern is generated between Simulation Run





- Seedclock (aka Seedclk)
  - Initialize the random number generator with a 10Mhz clock and the process ID number(aka SEEDCLK)
  - Different random pattern is generated between Simulation Run





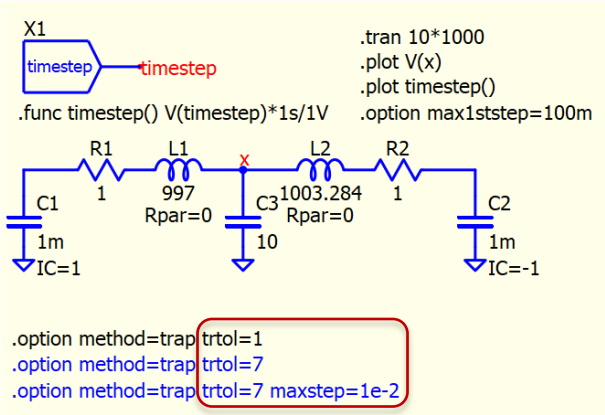
.option – TRTOL (Truncation error overestimation factor)

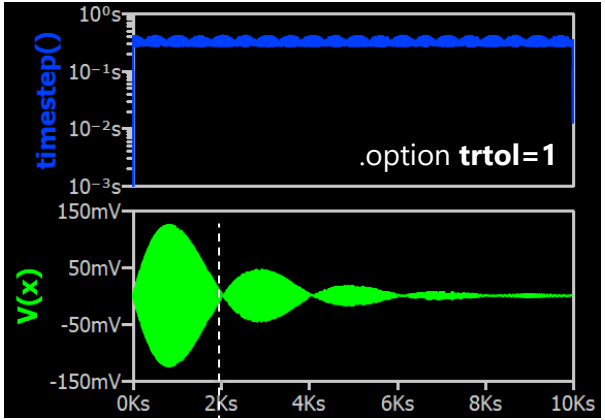
- TRTOL
  - Truncation error overestimation factor (Default Value in Spice Simulator)
    - Qspice : TRTOL = 2.5, Method=Trap
    - LTspice : TRTOL = 2, Method=Trap
    - TI-TINA : TRTOL = 7, Method=Gear
    - TI-PSpice : TRTOL = 7, Method=Gear
    - *** For Qspice to run with Pspice model, may have to consider to change TRTOL=7 and Method=Gear*
  - Trtol affects the timestep strategy more than directly affecting the accuracy of the simulation
  - For transistor-level simulations, a value larger than 1 is usually a better overall solution. You might find that you get a speed of 2x if you increase trtol without adversely affecting simulation accuracy

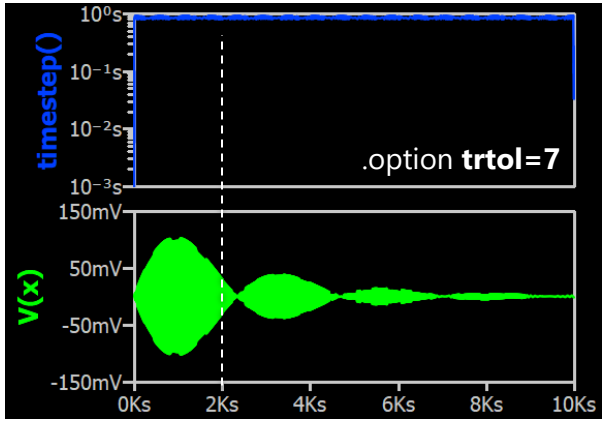
.option – TRTOL (Truncation error overestimation factor)

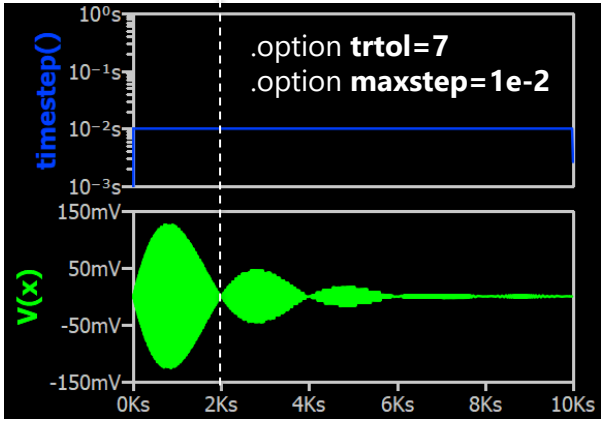
Qspice : option - TRTOL - PrecisionTestSimulations.qsch

- TRTOL
  - The accumulation of truncation errors over the present and past time step is managed by Truncation error factor (TRTOL)
  - This precision test circuit intermodulates two frequencies separated by $1/2000$ Hz, resulting in a beat frequency at $1/2000$ Hz in the envelope
  - In this example, TRTOL=7 produces an imprecise solution (error in beat frequency). However, limiting the maxstep can yield more accurate results









.param User-Defined Parameter

.param – User-defined Parameters

- .option relates to .param
  - .option ListParam : Print a list of the evaluated parameters(also sets SAVEPARAM)
  - .option SaveParam : Include evaluted user-defined parameters as waveform data
  - .option Seed : Initialize the random number generater used in .param statements

- .param
  - .param is a preprocessing process, value is calculated before simulation begin
  - .param accepts parameter (variable) assignment from a number or equation (unknown in equation must be defined by other .param)
  - Preprocessor Functions are supported
    List of function : Qspice Help > Simulator > General Conversions > Preprocessor Functions

.param – User-defined Parameters

Qspice : .param from param - SaveParam.qsch

- .param
  - If a .param argument is derived from another .param argument or from a formula, it is necessary to set .option SaveParam in order to save the parameters in the .qraw file
  - This is because the .qraw file only stores the values of .param arguments if they are direct values (e.g., param a=1)
  - If the value is derived indirectly, the .qraw file will not save it in the pre-headers lines. By using the .option SaveParam, parameters can be written as data channels

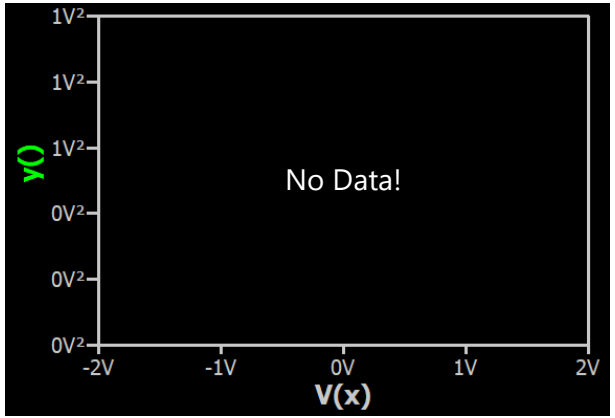
Math formula implementation with .tran

```
.param a=1
.param b=var
.param c=1

V1 x
PWL 0 -2 1 2

.func y() a*V(x)**2+b*V(x)+c    .plot y()
.option SaveParam
.tran 1
```

.param from another .param

.step param var list 0 1 2

Without parameters save into .qraw



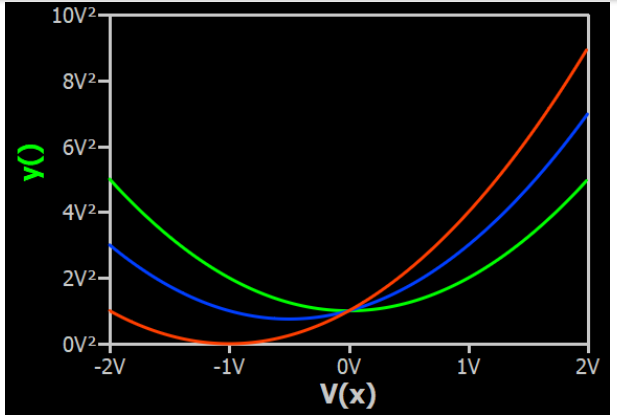
Math formula implementation with .tran

```
.param a=1
.param b=var
.param c=1

V1 x
PWL 0 -2 1 2

.func y() a*V(x)**2+b*V(x)+c    .plot y()
.option SaveParam
.tran 1
```

.param from another .param

.step param var list 0 1 2

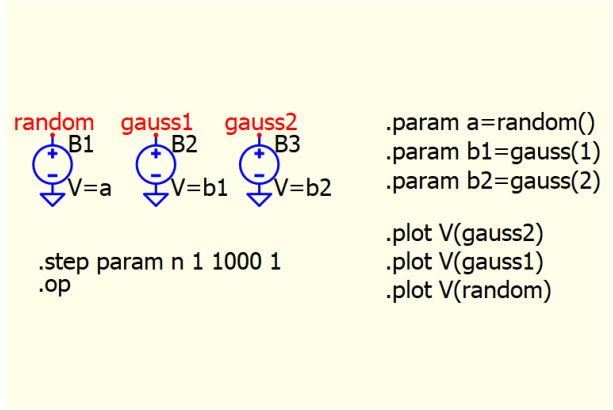
With parameters save into .qraw

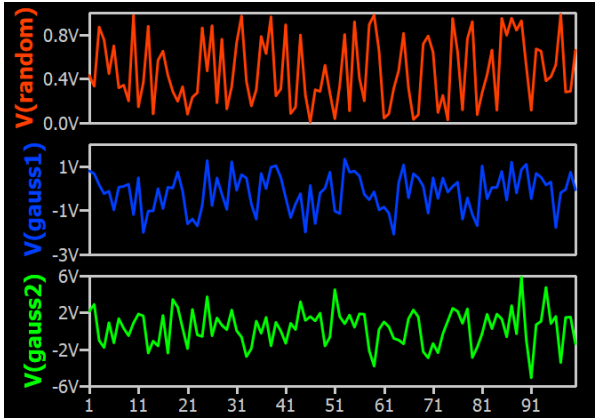


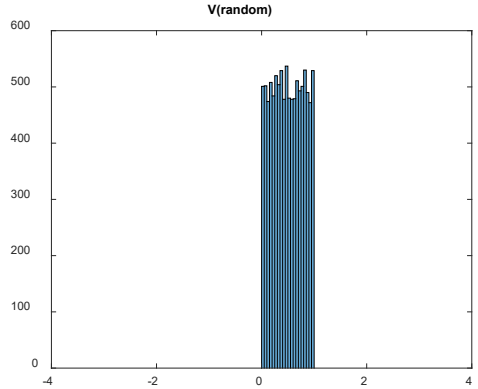
.param functions – random() and gauss(double sigma)

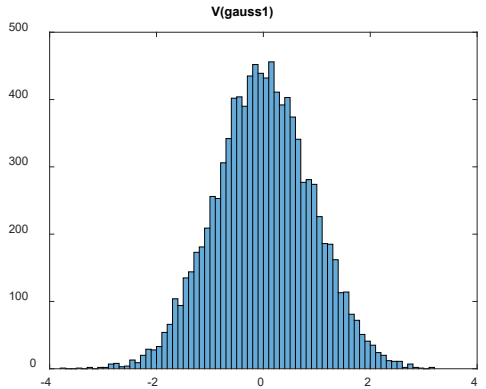
Qspice : param - random and gauss.qsch

- .param functions
  - Preparsing evaluation functions for .param
  - Random()
    - Random value between 0 and 1, uniform distribution
  - Gauss(σ)
    - Random number from Gaussian/Normal distribution with standard derivation σ and mean value at 0
  - Flat(x)
    - Random number between -x and x with uniform distribution
  - Mc(x,y)
    - Random number between x*(1+y) and x*(1-y) with uniform distribution







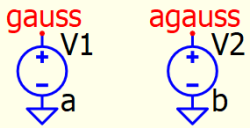


.param functions – Gauss(nom,rvar,sigma) and Agauss(nom,avar,sigma)

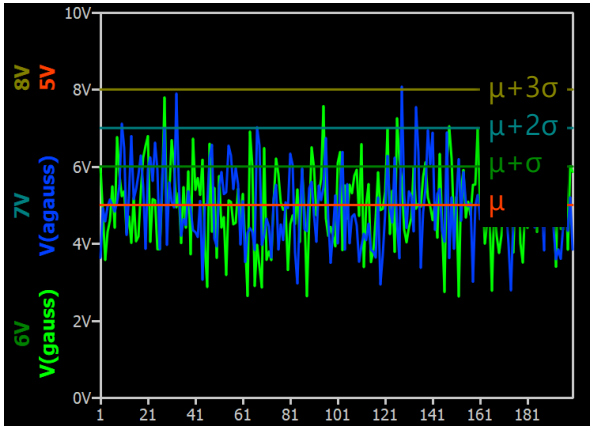
Qspice : param - gauss and agauss.qsch

- Gauss(nom,rvar,sigma) and **A**gauss(nom,**a**var,sigma)
  - Both are gaussian distriubtion (i.e. normal distribution), and different is how standard deviation is defined
  - GAUSS(Nominal Value, Relative variation, Sigma) [Undocumented]
    - Mean (μ) = nom = Nominal Value
    - Standard Deviation (σ) = $\frac{\text{nom} \times \text{rvar}}{\text{sigma}}$
    - ** If mean (nom) is zero, standard deviation is also zero
  - AGAUSS(Nominal Value, Absolute variation, Sigma) [Undocumented]
    - Mean (μ) = nom = Nominal Value
    - Standard Deviation (σ) = $\frac{\text{avar}}{\text{sigma}}$

```
.param a=gauss(5,.2,1)
.param b=agauss(5,1,1)

.step param LOOP 1 200 1
```



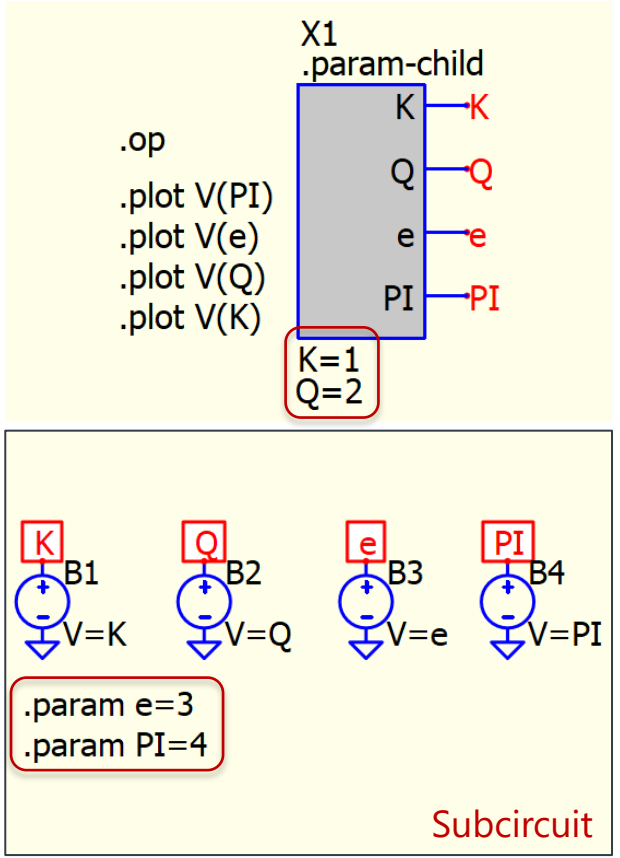

```
.op
.plot V(gauss),V(agauss),5V,6V,7V,8V
```



.param – Reserved Keywords for Fundamental Constants

Qspice : \.param\Global Constant\

- Reserved Keywords for Fundamental Constants
  - K, Q, e and PI are fundamental constants
    - K 1.380649e-23 Boltzmann constant
    - Q 1.602176487e-19 Elementary charge
    - E 2.7182818284590452354 Euler's number
    - PI 3.14159265358979323846 Ratio of circumference to diameter
  - They cannot be used as parameter name in global scope
  - However, other values can be assigned when scoped to a subcircuit



.prot (+ .unprot)
Protect(ed) with
Encryption

.prot – Protect(ed) with Encryption

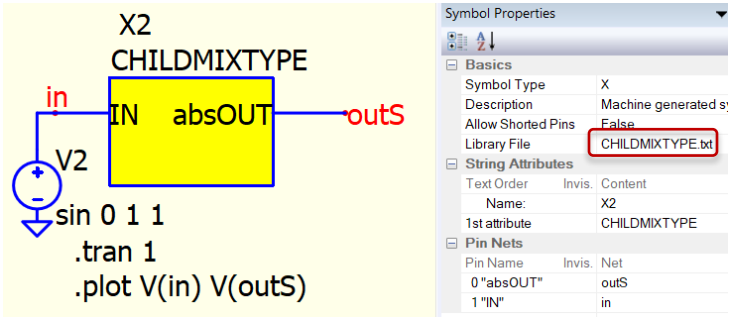
- .prot (+ .unprot)
  - Meaning #1 : During simulation, netlist section between .prot and .unprot is encrypted such that it can only be read by QSPICE
  - Meaning #2 : If the simulator (QSPICE64.exe or QSPICE80.exe) is invoked with the command line option –ProtectSelections, section between .prot and unprot should be encrypted
- Encrypt a netlist
  - Use QSPICE64.exe or QSPICE80.exe with command line option
    -ProtectSelections : Protect sections marked with .prot/.unprot with encryption
    -ProtectSubcircuits : Protect the body of subcircuits with encryption

Procedure to Protect a Subcircuit

Qspice : \.command\.protect\SubcktWithDLL

Condition of Example

This example includes a subcircuit that contains an Ø-device and a child subcircuit. The X-device calls this subcircuit's netlist (CHILDMIXTYPE.txt) for its use.

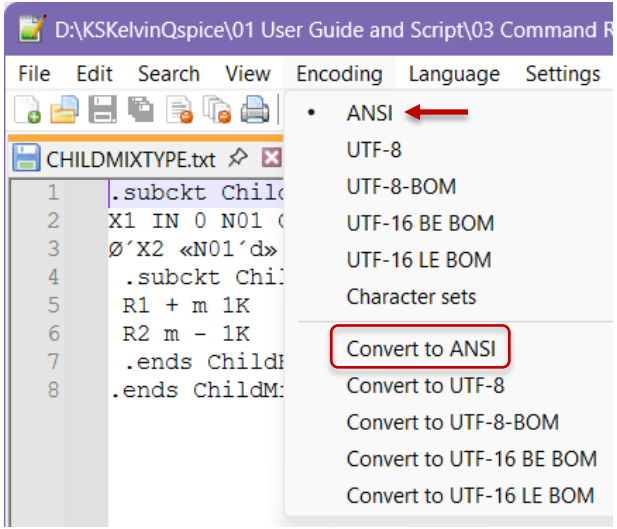


CHILDMIXTYPE.txt

```
1  .subckt ChildMixType absOUT IN
2  X1 IN 0 N01 ChildHierarchy
3  Ø´X2 «N01´d» «absOUT´d» «» ChildDLL
4   .subckt ChildHierarchy + - m
5   R1 + m 1K
6   R2 m - 1K
7   .ends ChildHierarchy
8  .ends ChildMixType
```

Subcircuit netlist encoding format

Qspice reads netlists in ANSI format to handle special characters. Typically, when a netlist is copied from Qspice and pasted into a text document, it is encoded in UTF-8. If the encoding is not ANSI, ensure to convert it to ANSI (and save it), or Qspice may encounter errors while reading the netlist. I recommend using **Notepad++** for text editing purposes.

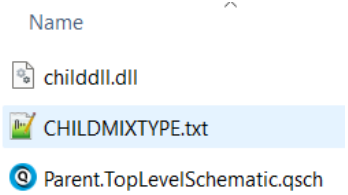


Procedure to Protect a Subcircuit

Qspice : \.command\.protect\SubcktWithDLL

Procedure to Protect Subcircuit

[1] A directory contains subcircuit netlist file
Assume directory is C:\QspiceSubckt



[2] Remember to backup subcircuit netlist file, encryption will directly write and replace this file

[3] Open command window with CMD

[4] Type following command

>> cd C:\QspiceSubckt

>> path C:\Program Files\QSPICE

this command allow Qspice64.exe to be call

>> Qspice64 -ProtectSubcircuits CHILDMIXTYPE.txt

use ProtectSubcircuits option, this option doesn't need to specify .prot and .unprot

```
C:\Users\      >cd C:\QspiceSubckt

C:\QspiceSubckt>path C:\Program Files\QSPICE

C:\QspiceSubckt>Qspice64 -ProtectSubcircuits CHILDMIXTYPE.txt

C:\QspiceSubckt>
```

CHILDMIXTYPE.txt

```
1  .subckt ChildMixType absOUT IN
2  .prot
3  JZHHTl2zhlT0DSByPItq12Wr7Jbboy6sBxFFymGyk73uz3hwjr&V4i5ACaPpielwadJytiXjKCMN
4  K3pDcp,,v6VNAfiWHzksBjn6jYch0nq,e96,uNzu9VcLEsueL&XBm50ETrTXFSdqpNjPUaP6RnwU
5  3X44LKV2DnEs9t&IhhlFz&R4RtSWhtvXXl,x67lk8SZYEzMglv3icRsKO1
6  .unprot
7  .ends ChildMixType
```

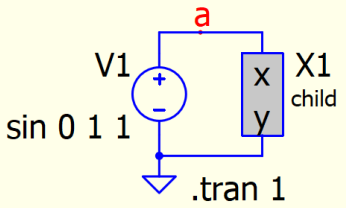
encrypted

.save
Limit Saved Data
Traces

.save - Limit Saved Data Traces

Qspice : .save - Parent.child.qsch

- .save
  - Qspice saves all available data in default, including sub-circuit data
  - To limit the size of the output file, .save command allows to save only the data of interest
  - Characters
    - '*' : any characters except the hierarchy separator "•"
    - '?' : any one character
  - To save hierarchy data, uses • or ? in expression
  - Use **.save V(*) I(*) P(*)** to only save parent schematic waveform data

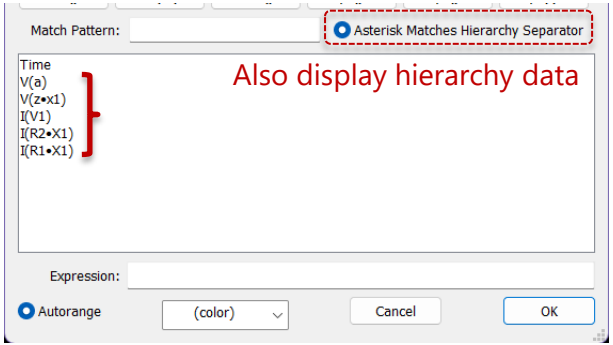


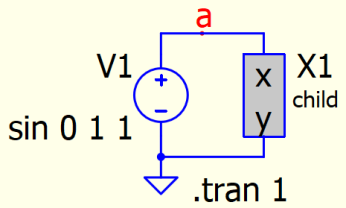

```
.subckt child x y
R1 x z 1
R2 z y 1
.ends child
```

```
.save V(a) V(z?*)
.save V(a) V(z•x1)
```

without .save

Waveform Viewer > Add Plot



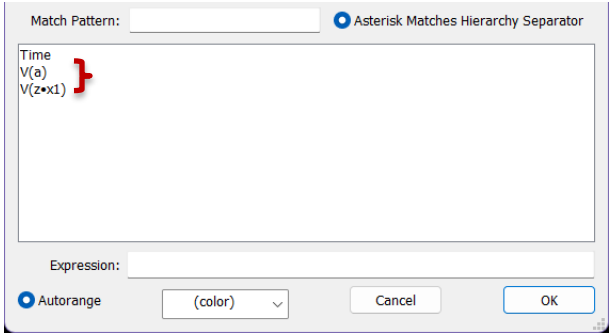



```
.subckt child x y
R1 x z 1
R2 z y 1
.ends child
```

```
.save V(a) V(z?*)
.save V(a) V(z•x1)
```

with .save

Waveform Viewer > Add Plot

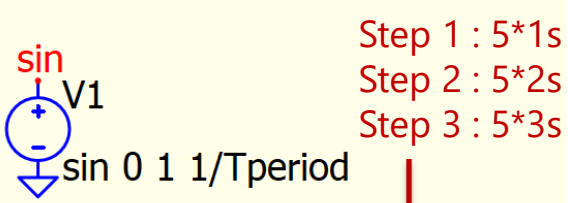


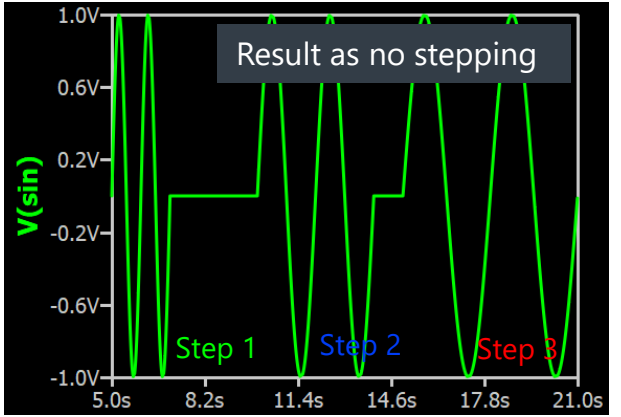
.step
Step User-Defined Parameter

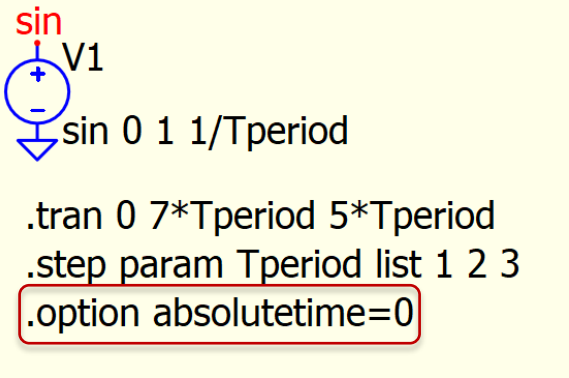
.step - .tran with Tstart is parameterized

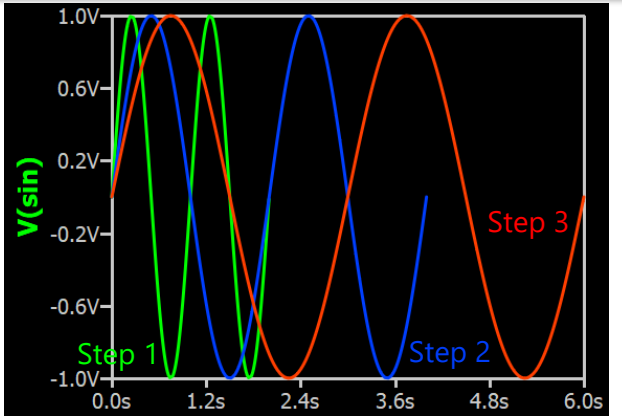
Qspice : .step - Parameterized Tstart.qsch

- .step with Tstart is parameterized
  - QSPICE looks for the time to go backwards to detect one step from the next
  - With absolutetime=1, data is stored in absolute time and step a parameterized Tstart moves time forwards in each step. As a result, Qspice can't keep track of the steps in waveform viewer
  - **Solution** : Disable absolutetime to force time to reset to t=0 in each step (this is Qspice default behavior now)







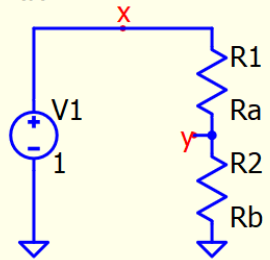


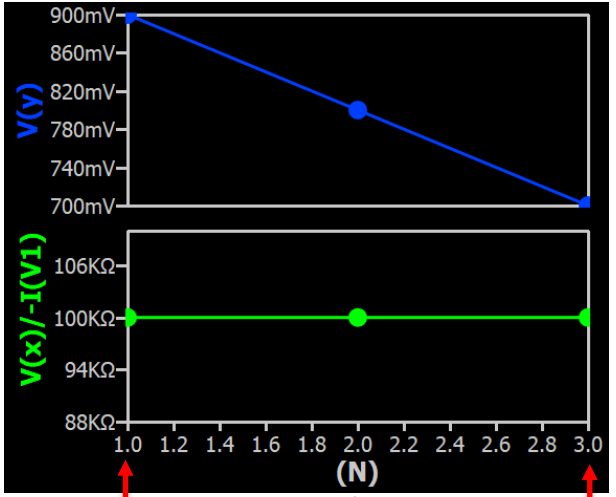
.step with table : Batch Simulation

Qspice : Step with Table.qsch

- .step with table
  - By using table function, user can step integer N from 1, 2, 3, … and assign value to different parameters according to table
  - This is an example to sweep upper and lower resistor network with different resistance combination

```
.op
.step param N list 1 2 3
.param Ra table(N, 1,10K, 2,20K, 3,30K)
.param Rb table(N, 1,90K, 2,80K, 3,70K)
.plot V(x)/-I(V1)
.plot V(y)
```





N=1 Ra=10K Rb=90K

N=2 Ra=20K Rb=80K

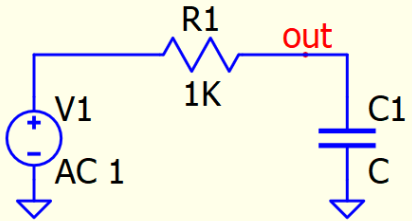
N=3 Ra=30K Rb=70K

.step in Single Point Simulation – Explanation (page 1)

Qspice : Step in Single Point - 01 Explain.qsch

- Single Point Simulation
  - The .step command is treated differently in the output data file depending on whether it is a single point or multiple points simulation directive
  - Single Point Simulation
    - Includes .op and .ac list
    - In .qraw, flags doesn't have "stepped", therefore, .step is not treated as stepped in single point simulation
    - In variables, x-axis variable "0" is C
  - Multiple Point Simulation
    - If simulation directive is multiple points, data file will set stepped flag and with x-axis as Frequency / Voltage / Current depends on it is .ac list or .op

```
.ac list 100  Single point simulation
.param C = 1µ
.step param C 1µ 10µ 1µ
```

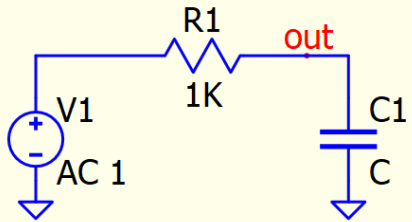


Step in Single Point Simulation.qraw

```
1  Title: * C:\KSKelvinQspice\01 User Guide and S
2  Date: Sun May  5 01:01:07 2024
3  Plotname: AC Analysis
4  Flags: complex
5  Abscissa:     1.0000000000000000e-06     9.9999
6  No. Variables: 6
7  No. Points: 10
8  Command: QSPICE80, Build Apr 30 2024 17:13:22
9  .param C=1µ
10 .param temp=27
11 .alias Freq Frequency
12 .alias Omega 2*pi*Frequency
13 Variables:
14    0   C   parameter
15    1   V(n01)   voltage
16    2   V(out)   voltage
17    3   I(V1)   current
18    4   I(R1)   current
19    5   I(C1)   current
```

x-axis

```
.ac list 100 1K  Multiple points simulation
.param C = 1µ
.step param C 1µ 10µ 1µ
```



Step in Single Point Simulation.qraw

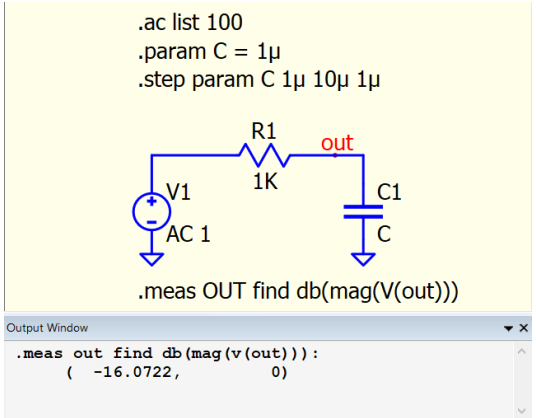
```
1  Title: * C:\KSKelvinQspice\01 User Guide and S
2  Date: Sun May  5 01:02:59 2024
3  Plotname: AC Analysis
4  Flags: complex stepped
5  Abscissa:     1.0000000000000000e+02     1.0000
6  No. Variables: 7
7  No. Points: 20
8  Command: QSPICE80, Build Apr 30 2024 17:13:22
9  .param C=1µ
10 .param temp=27
11 .alias Freq Frequency
12 .alias Omega 2*pi*Frequency
13 Variables:
14    0   Frequency   frequency
15    1   V(n01)   voltage
16    2   V(out)   voltage
17    3   I(V1)   current
18    4   I(R1)   current
19    5   I(C1)   current
20    6   C   parameter
```

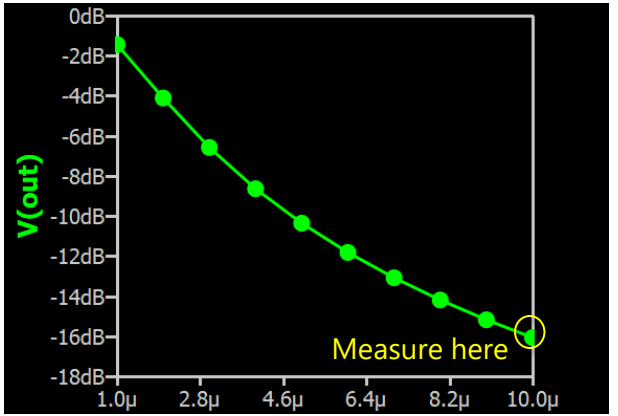
x-axis

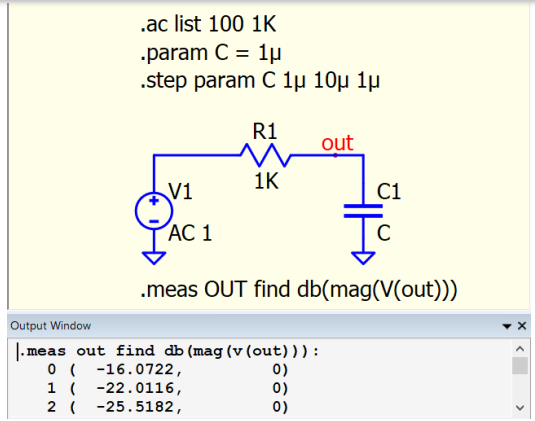
.step in Single Point Simulation – Impact (page 2)

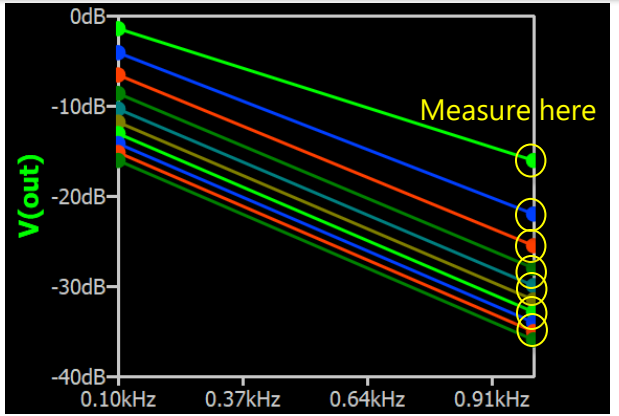
Qspice : Step in Single Point - 02 Impact.qsch

- Impact
  - In the .plot command, for a single point simulation with .step, the stepped parameters are plotted on the x-axis, resulting in only one curve. However, in a multiple point simulation with .step, each stepped parameter generates a separate curve
  - In the .meas command, for a single point simulation, as the data set is not treated as stepped data, only one measurement result is output. However, in a multiple point simulation, the .meas command returns results equal to the total number of steps









.system
Execute System Commands

.system – Environment Strings

- Environment Strings
  - %DECK%
    - (depends on to run a .qsch, or [View] > [Netlist] for a .cir)
    - [userpath]\[filename].qsch OR
    - [userpath]\[filename].cir
  - %RAWFILE%
    - [userpath]\[filename].qraw
  - %QUX%
    - [QspiceInstallPath]\QUX.exe
  - %QPOST%
    - [QspiceInstallPath]\QPOST.exe
  - %QSPICE64%
    - [QspiceInstallPath]\QSPICE64.exe
  - %QSPICE80%
    - [QspiceInstallPath]\QSPICE80.exe
  - %QSPICE%
    - (depends on which simulator is selected in Qspice preference or .option fastmath)
    - [QspiceInstallPath]\QSPICE64.exe OR
    - [QspiceInstallPath]\QSPICE80.exe

Declared Environmental Strings

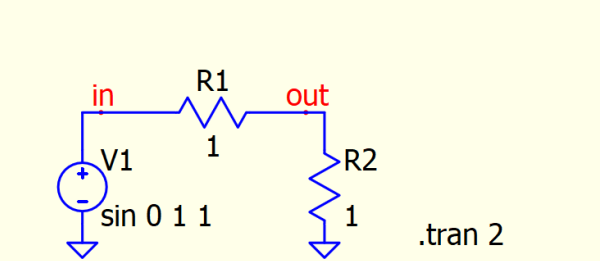
Name	Description
DECK	Path of the input netlist
RAWFILE	Path of the output waveform file
QUX	Path to QUX.exe, the waveform viewer
QPOST	Path to QPOST.exe, the post processor for .meas and .four
QSPICE64	Path to QSPICE64.exe, the simulator using fast math
QSPICE80	Path to QSPICE80.exe, the simulator
QSPICE	Path to the current simulator in use

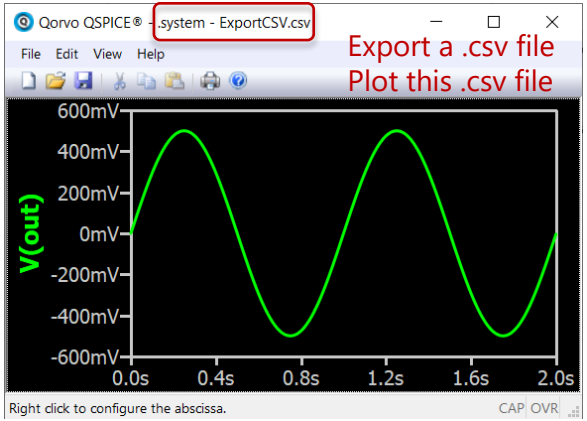
- Environmental parameters can return entire path and filename for user to quickly access entire filepath for system command

.system – Execute a Command, Export CSV example

Qspice : .system - ExportCSV.qsch

- .system – export CSV
  - To execute commands once the simulation completes
  - Command will be executed as if you had typed them at the DOS C:\> prompt
  - Environment Strings
    - %DECK%
      - [userpath]\[filename].qsch OR
      - [userpath]\[filename].cir
    - %RAWFILE%
      - [userpath]\[filename].qraw
    - %QUX%
      - [QspiceInstallPath]\QUX.exe



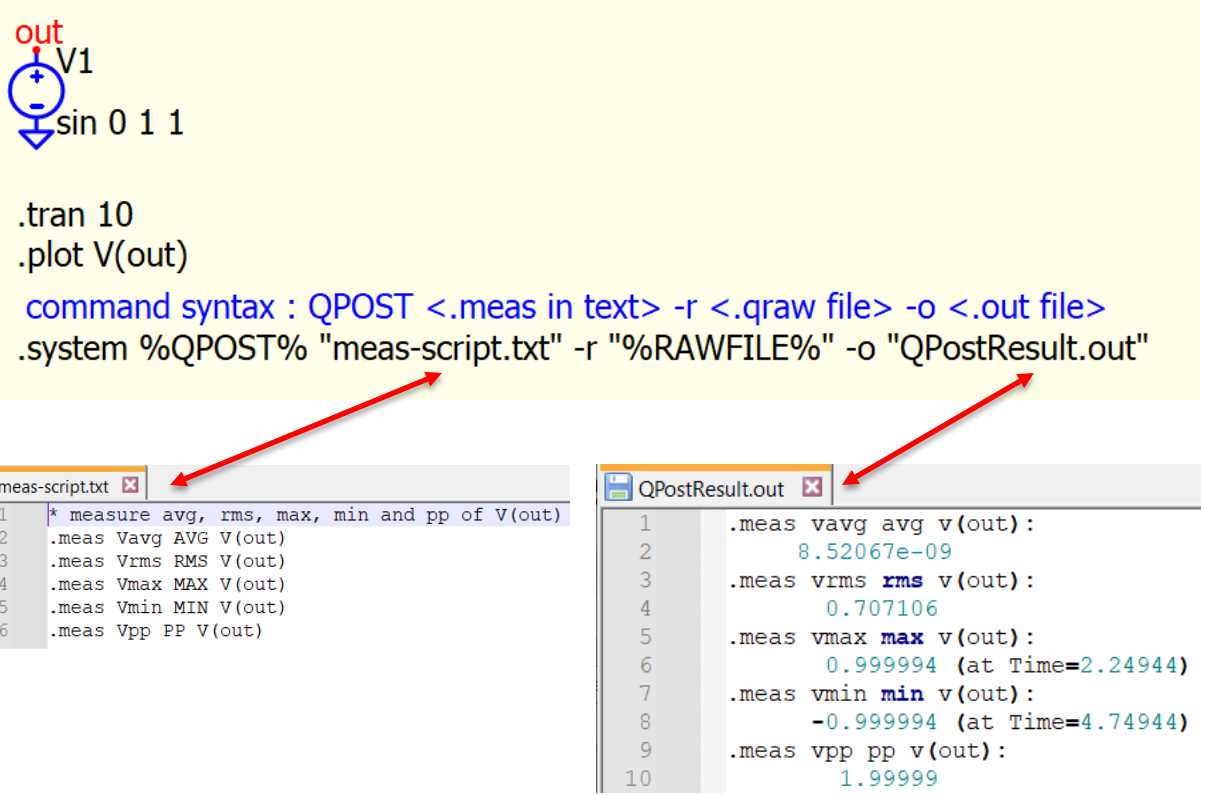


- Explanation of example
  - First .system is to export *.qraw into csv file with QUX.exe for V(out)
  - Second .system is to plot this exported csv file
  - "" is require if file contains space in environment strings

.system – Execute a Command, Export .meas results

Qspice : .system - ExportQPost.qsch | meas-script.txt

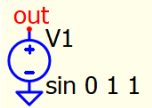
- .system – export .meas
  - Exporting the .meas output file can be a bit tricky
  - In the command line, QPOST.exe is used to post-process the .qraw waveform data through a text file that contains the .meas statement (*.cir or *.txt file)
  - However, Qspice runs from the GUI will not save a *.cir file from .qsch; therefore, the .meas output is not in a text file format
  - Solution is to create a text file for .meas script, and with .system to call QPOST.exe to execute this script file for .qraw waveform data and generate post-processing result



.system – Execute a Command, Export .meas results

Qspice : .system - ExportQPost-Netlist.qsch

- .system – export .meas
  - If the .meas statement is only in the schematic file and you intend to use %DECK% to load a .cir file in post-processing, the user should run the simulation from the netlist
  - Procedure
    - View > Netlist in schematic window
    - In Netlist, File > Save, save file into *.cir
    - Run the simulation from netlist
    - Qspice will directly load a netlist file (*.cir) instead of (*.qsch) when utilizing the environment parameter %DECK%



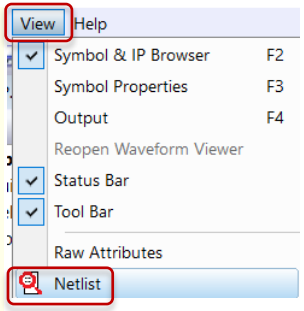

```
.tran 10
.plot V(out)
* measure avg, rms, max, min and pp of V(out)
.meas Vavg AVG V(out)
.meas Vrms RMS V(out)
.meas Vmax MAX V(out)
.meas Vmin MIN V(out)
.meas Vpp PP V(out)
```

command syntax : QPOST <.cir file> -r <.qraw file> -o <.out file>

```
.system %QPOST% "%DECK%" -r "%RAWFILE%" -o "QPostResult.out"
```

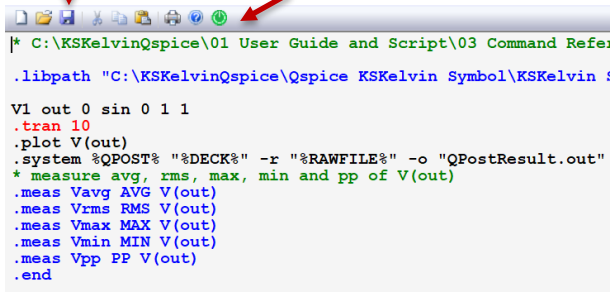
** Procedure for this .system to work (i.e. %DECK% to call a .cir instead of .qsch)

1. View > Netlist
2. In Netlist Window, File > Save, save file into .cir
3. Run simulation from Netlist



Save as a .cir

Run from netlist



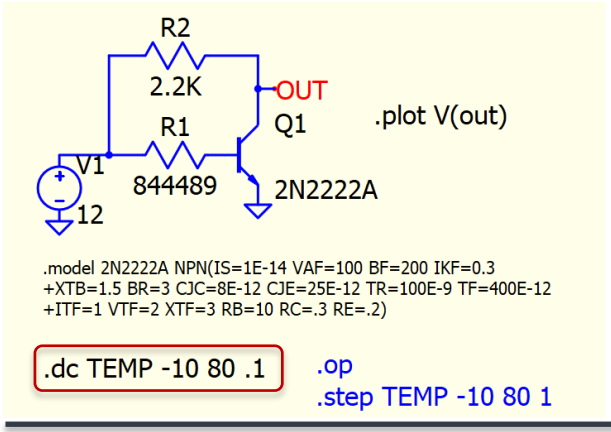
```
* C:\KSKelvinQspice\01 User Guide and Script\03 Command Refe
.libpath "C:\KSKelvinQspice\Qspice KSKelvin Symbol\KSKelvin
V1 out 0 sin 0 1 1
.tran 10
.plot V(out)
.system %QPOST% "%DECK%" -r "%RAWFILE%" -o "QPostResult.out"
* measure avg, rms, max, min and pp of V(out)
.meas Vavg AVG V(out)
.meas Vrms RMS V(out)
.meas Vmax MAX V(out)
.meas Vmin MIN V(out)
.meas Vpp PP V(out)
.end
```

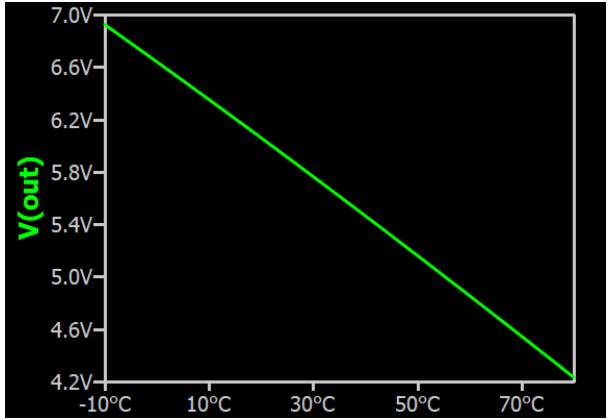
.temp
Temperature

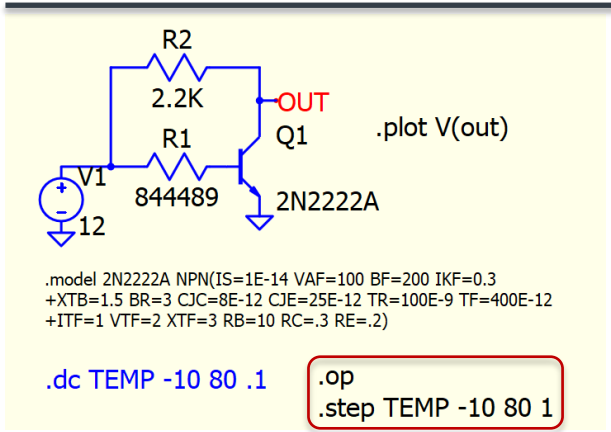
.temp – Circuit Temperature in DC analysis

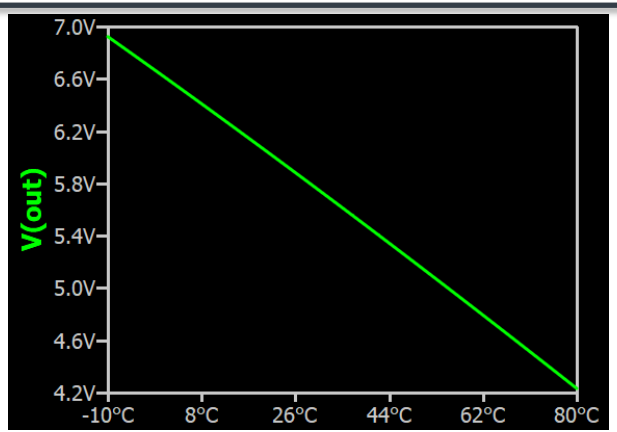
Qspice : .temp - dc directive.qsch

- .temp in DC analysis
  - Method 1
    - Run DC Sweep (.dc) analysis with sweep parameter as TEMP (circuit temperature)
    - Recommend as run faster since it doesn't have to reparse the netlist
  - Method 2
    - Run Bias Point Analysis (.op) with step TEMP parameter (.step)





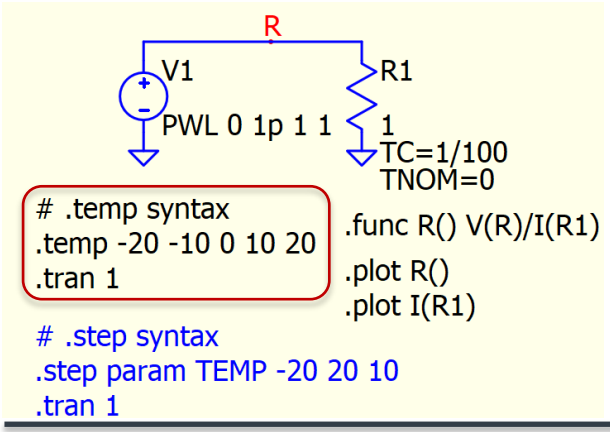


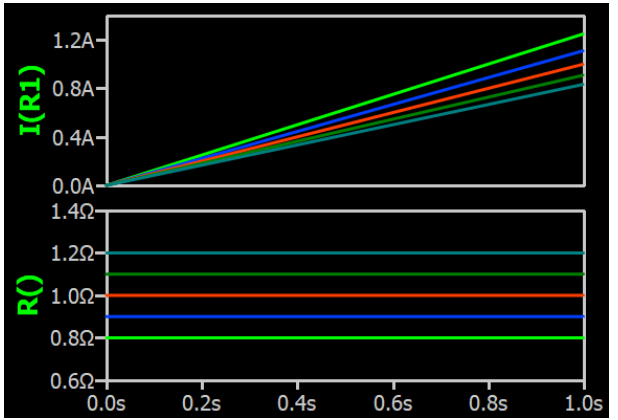


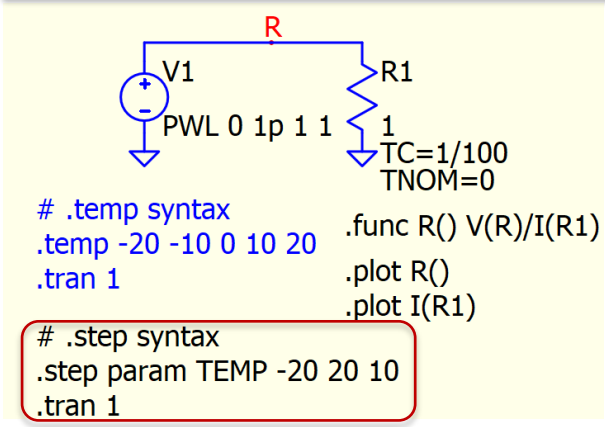
.temp – Circuit Temperature in Transient analysis

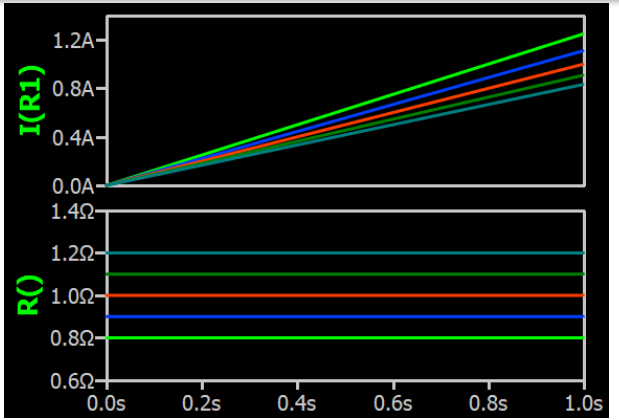
Qspice : .temp - tran directive.qsch

- .temp in Transient
  - Method 1
    - Run Circuit Temperature (.temp) analysis
  - Method 2
    - Run with step TEMP parameter (.step)









.tran
Non-Linear Transient Analysis

Syntax of .tran (Non-Linear Transient Analysis)

- Two syntax of .tran Non-Linear Transient Analysis
  - .tran TSTOP [UIC]
    - If MAXSTEP is required in this syntax, use *.options MAXSTEP* instead
  - .tran IGNORED TSTOP [TSTART [MAXSTEP]] [UIC]
    - Recommend to fill 0 at IGNORED. This syntax allows to set start recording time and maxstep
    - If need to limit .qraw file size, consider to specify Tstart and limit data to disk .qraw file size

1. Specify only the stop time

Syntax: .tran TSTOP [UIC]

Name	Description	Units
TSTOP	Total amount of time to simulate	s
UIC	Use initial conditions instead of solving for the initial bias point(SKIPBP)	

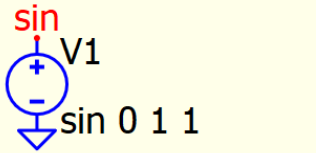
2. Traditional Berkeley Syntax

Syntax: .tran IGNORED TSTOP [TSTART [MAXSTEP]] [UIC]

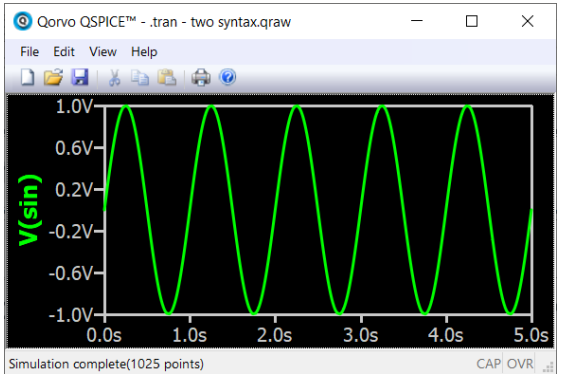
Name	Description	Units
IGNORED[2]	An ignored value	s
TSTOP	Total amount of time to simulate	s
TSTART	Time to start recording waveform data to disk	s
MAXSTEP	Maximum time step size to allow	s
UIC	Use initial conditions instead of solving for the initial bias point(SKIPBP)	

.tran (Non-Linear Transient Analysis) : Syntax Examples

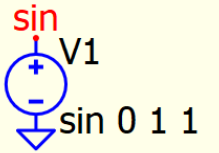
Only Stop Time Syntax




```
.tran 5   .options MAXSTEP=10µ
.tran 0 5 2 10µ
.plot V(sin)
```

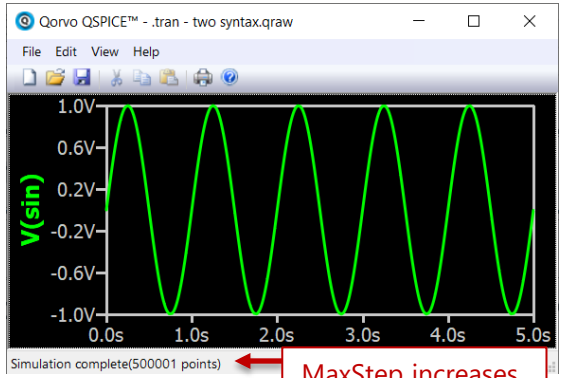


Only Stop Time Syntax with .option MAXSTEP



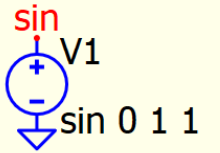
Use .option to set MaxStep

```
.tran 5   .options MAXSTEP=10µ
.tran 0 5 2 10µ
.plot V(sin)
```



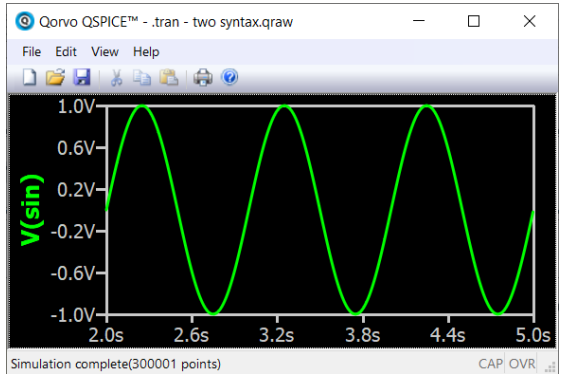
MaxStep increases simulation points

Traditional Berkeley Syntax




```
.tran 5   .options MAXSTEP=10µ
.tran 0 5 2 10µ
```

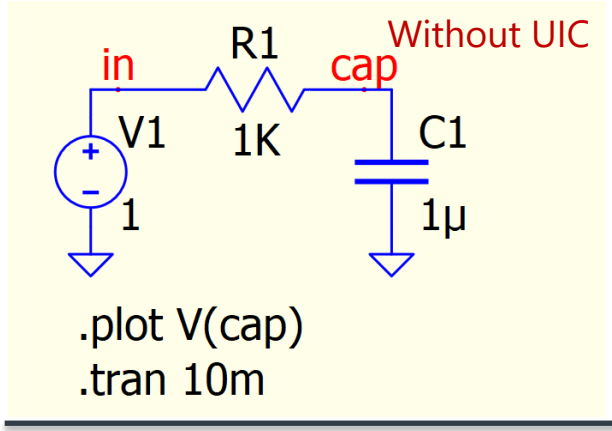
.tran Ignore Tstop Tstart MaxStep

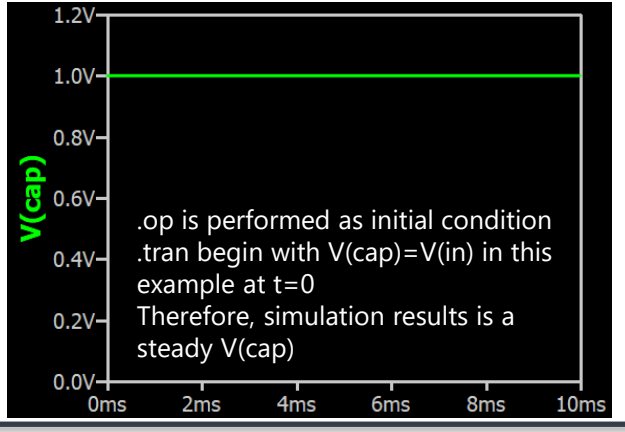


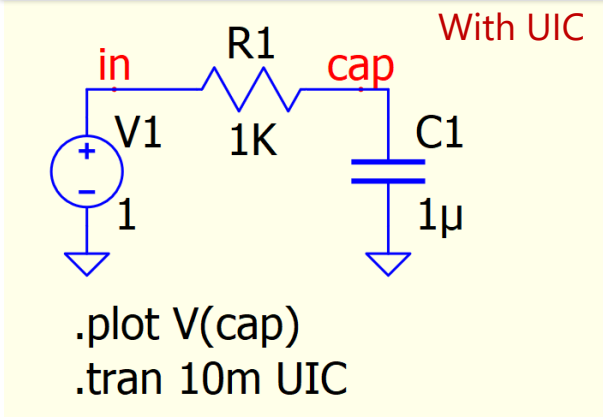
.tran – UIC (Use Initial Condition)

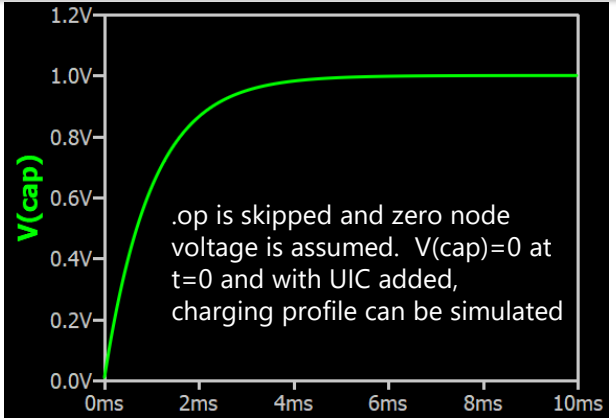
Qspice : .tran - UIC.qsch

- UIC
  - **U**se **I**nitial **C**ondition
  - A DC operating point analysis (.op) is performed before starting the transient analysis (.tran). This directive suppresses this initialization
  - The node voltage is taken as zero if not specified
  - However, UIC is not a particularly recommended feature of SPICE (refer to UIC Help in LTspice), and the reason is explained in next page
    - An alternative method for this example without UIC is to add initial condition directive
      .IC V(cap)=0







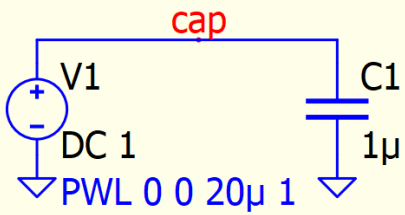


.tran – UIC (Use Initial Condition) and its Limitation

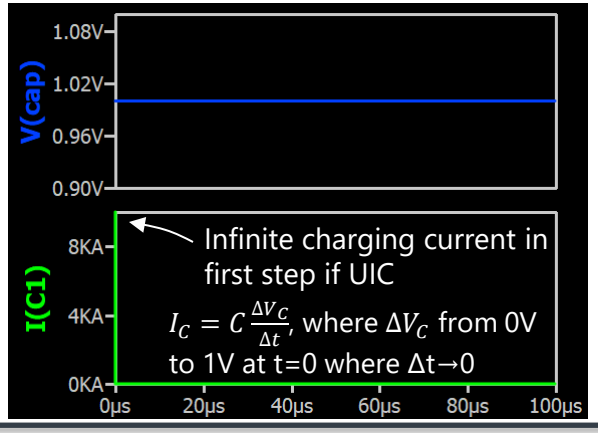
Qspice : .tran - UIC limitation.qsch

- UIC Limitation
  - Skipping the DC operating point analysis leads to nonphysical initial conditions and may introduce difficulty in simulation
  - For example, voltage source in parallel to a capacitor required infinite current to charge in the first time step, which may result in a "time step too small" convergence failure

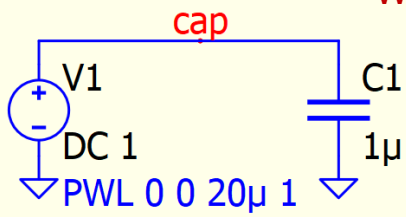
With UIC



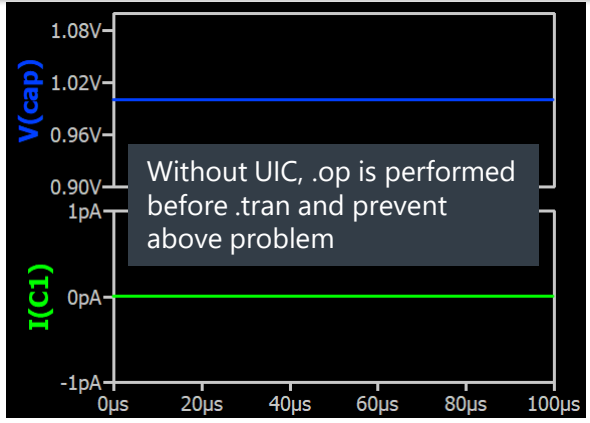
.tran 100µ UIC
.option maxstep=100p
.plot I(C1)
.plot V(cap)



Without UIC



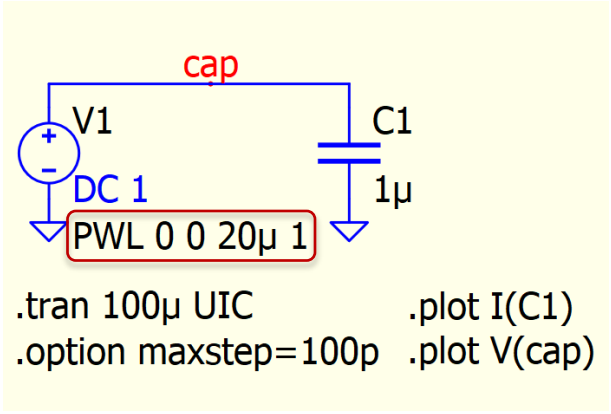
.tran 100µ
.option maxstep=100p
.plot I(C1)
.plot V(cap)

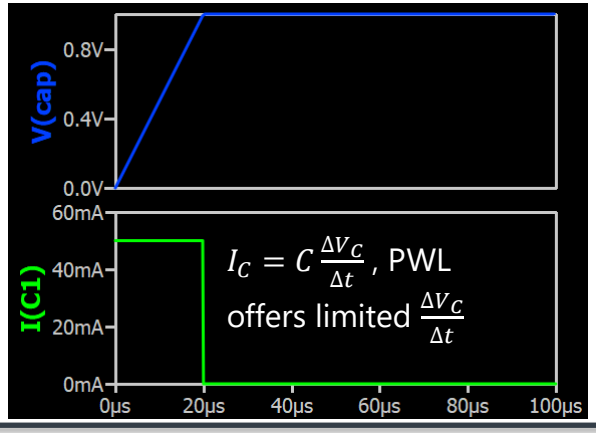


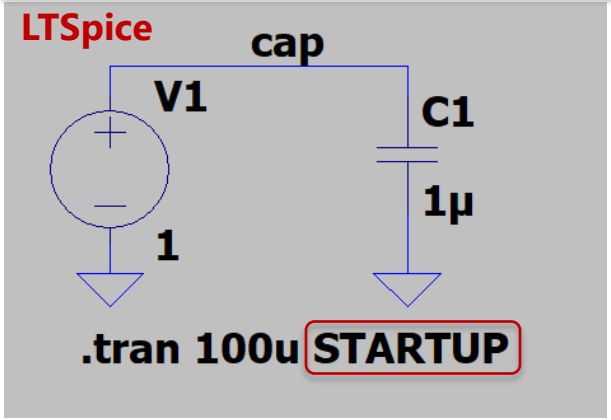
.tran – STARTUP (Not in Qspice)

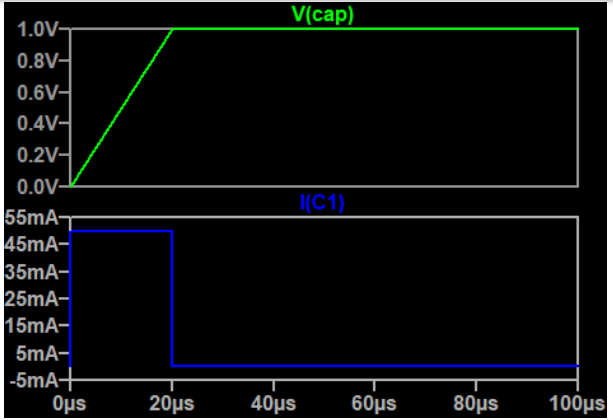
LTspice : .tran - startup.asc

- Startup
  - This is not a modifier directive in Qspice but in LTspice
    - It is needed for many of the switcher models in LTspice according to Mike Engelhardt explanation
  - Startup modifier means independent source should be ramped on during the first 20us of the simulation
    - In my test, only DC source with ramp added, but not PULSE, SINE, PWL etc…
  - An equivalent approach in Qspice is to change voltage source from DC to ***PWL 0 0 20u [VDC]***





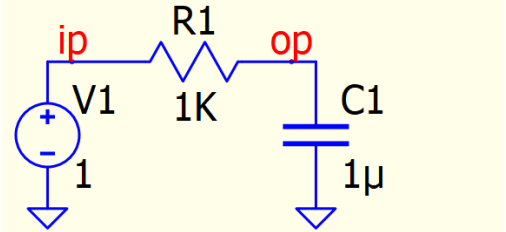


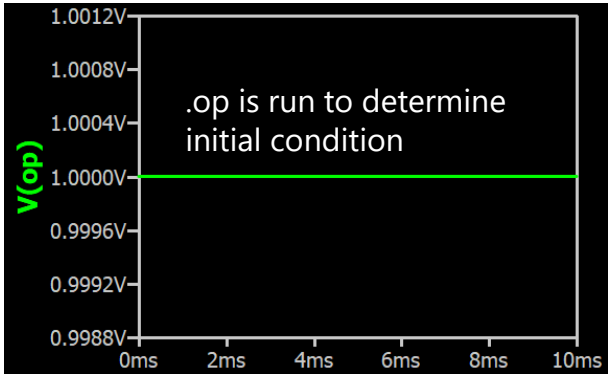


.tran (Non-Linear Transient Analysis) : .IC/IC Without UIC

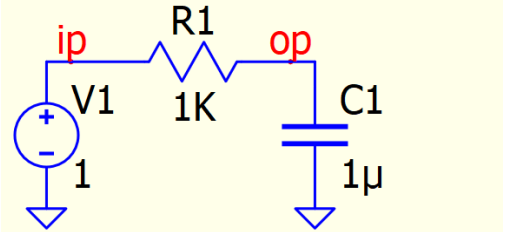
Qspice : .tran - UIC and IC.qsch

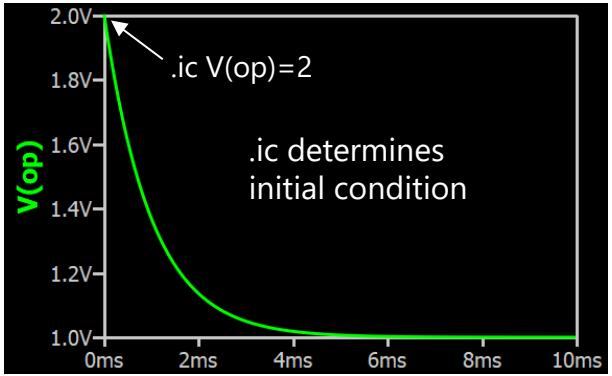
Without UIC



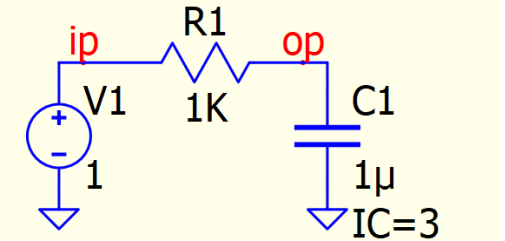


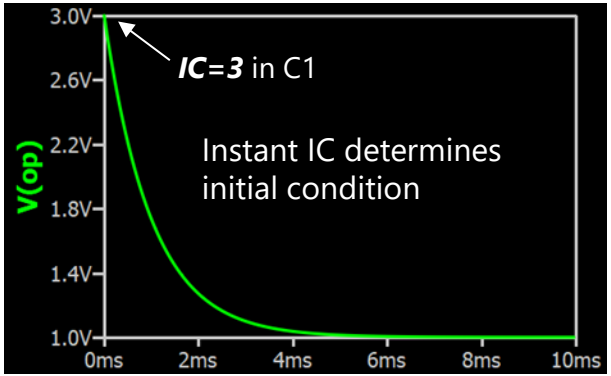
.IC Without UIC





Instance IC W/o UIC

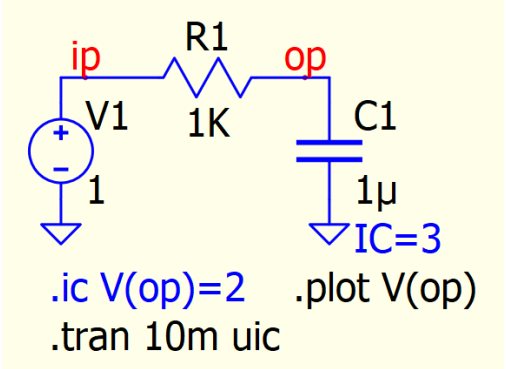


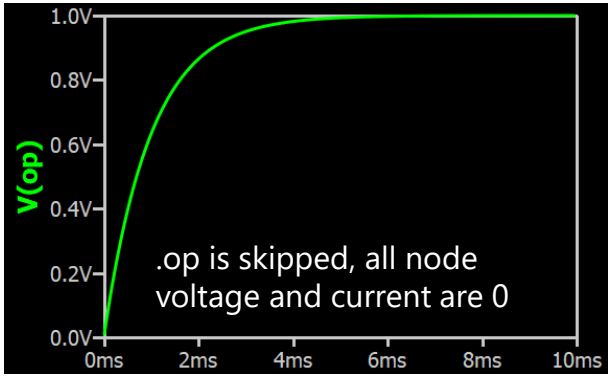


.tran (Non-Linear Transient Analysis) : .IC/IC With UIC

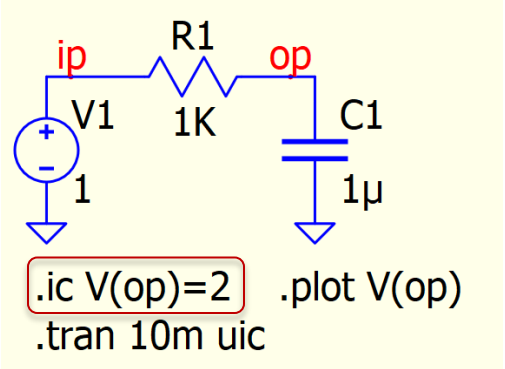
Qspice : .tran - UIC and IC.qsch

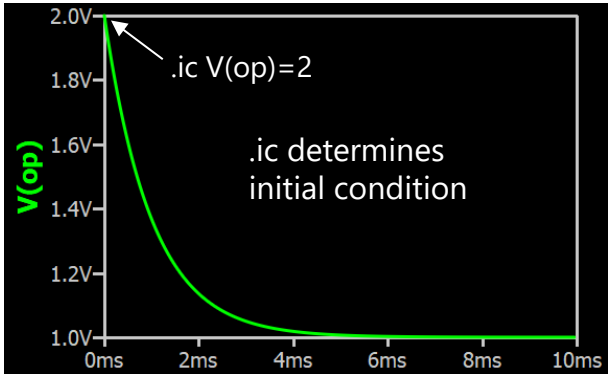
With UIC



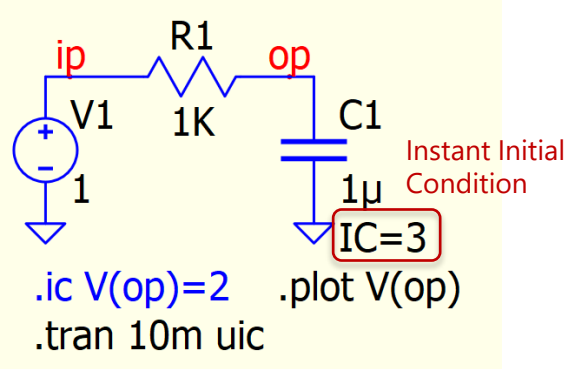


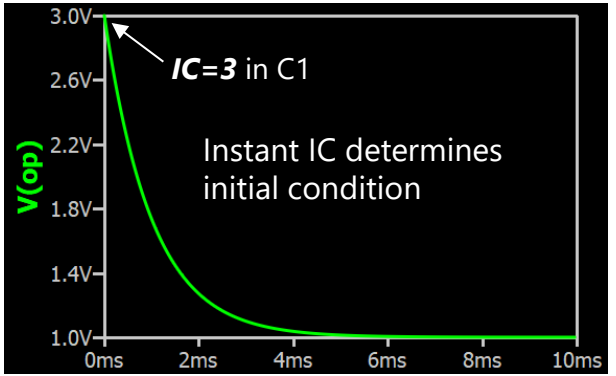
.IC With UIC





Instance IC With UIC

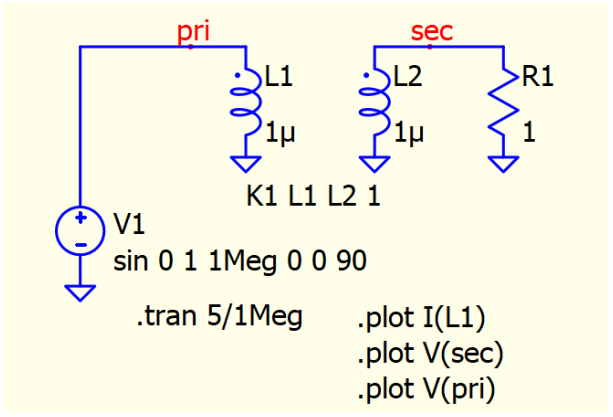


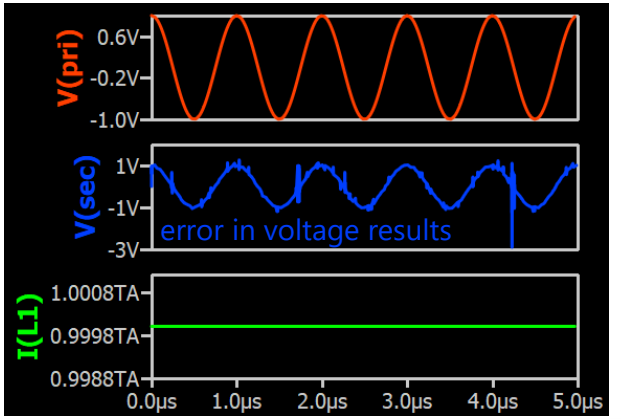


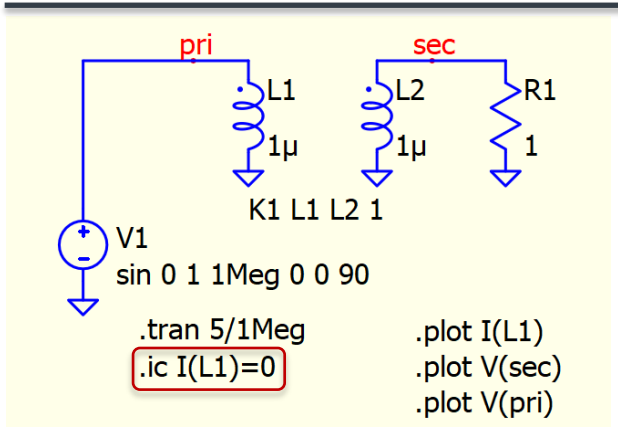
Importance of initial condition for .tran

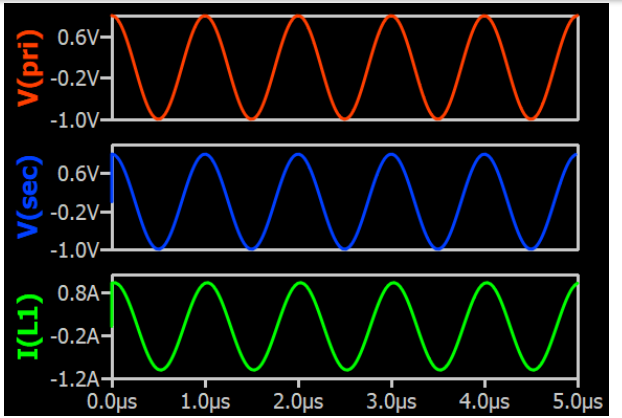
Qspice : .tran - importance - example1.qsch

- Importance of Initial Condition for .tran
  - Example 1
    - In a coupled inductor circuit with a voltage source in parallel with the primary inductor, there may be a significant DC current during .op analysis. This TA level (10^{12}) of current can lead to errors in voltage/current results
  - Solution #1
    - Set an initial condition using the .ic directive





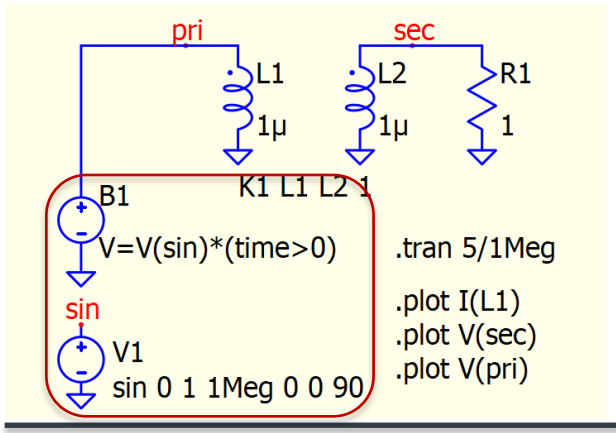


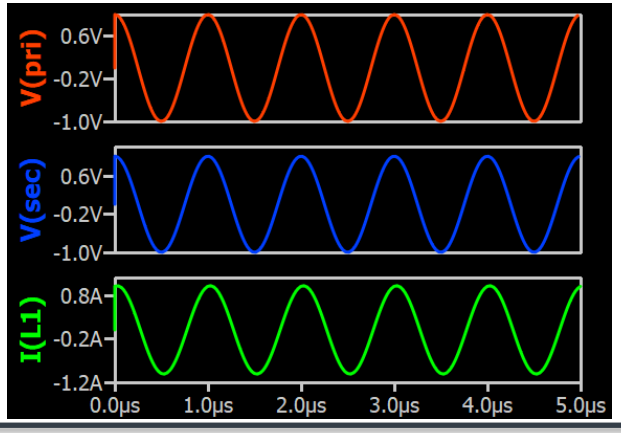


Importance of initial condition for .tran

Qspice : .tran - importance – example2.qsch

- Importance of Initial Condition for .tran
  - Solution #2
    - Setup a B-source with (time>0) to only activity source when simulation time > 0 (i.e. 0V in DC condition)





Batch mode Command

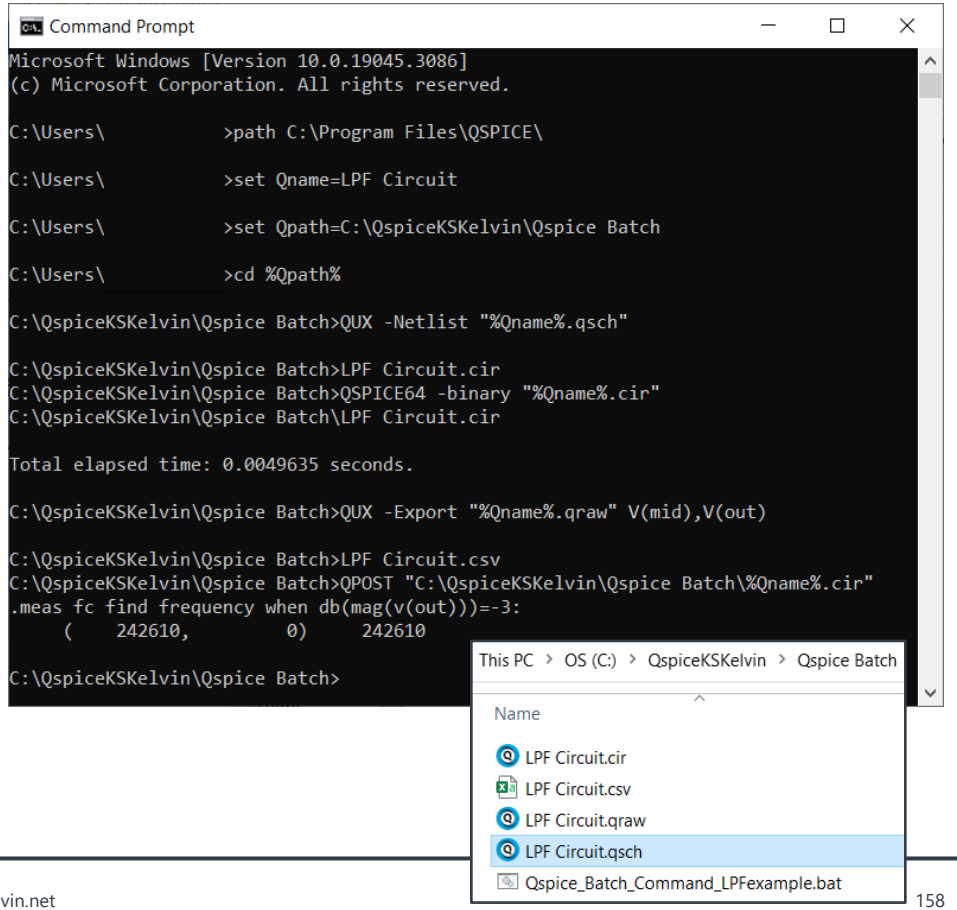
Qspice Execution Files

- Qspice execution files
  - Directory (default installation) : C:\Program Files\QSPICE
  - Schematic Capture and Waveform Viewer Program (HELP > Waveform Viewer)
    - Execution file : QUX.exe
    - Function #1 : Convert .qsch schematic to .cir
    - Function #2 : Export data from data file .qraw
    - Function #3 : Plot data file .qraw or .csv
  - QSPICE Simulator (HELP > Simulator)
    - Execution file : QSPICE64.exe [Enable Fast (less accurate) Math]
    - Execution file : QSPICE80.exe
    - Function : Run simulation from .cir
  - Post Processor (HELP > Post Processor)
    - Execution file : QPOST.exe
    - Function : Execute .meas and .four from .qraw

Batch command basic workflow

Qspice : Qspice_Batch_Command_LPFexample.bat / LPF Circuit.qsch

- Batch command workflow
  - Run CMD in Windows, in Command Prompt
  - Set path for Qspice program
    - path C:\Program Files\QSPICE\
  - Set variable name for working folder
    - set Qname=LPF Circuit
    - set Qpath=C:\QspiceKSKelvin\Qspice Batch
  - Goto schematic .qsch directory
    - cd %Qpath%
  - Convert .qsch to .cir (netlist)
    - QUX -Netlist "%Qname%.qsch"
  - Run Qspice simulation for .qraw
    - QSPICE64 -binary "%Qname%.cir"
    - QSPICE64 -ascii "%Qname%.cir" -r "%Qname%-ascii.qraw"
  - Export data from .qraw to .csv
    - QUX -Export "%Qname%.qraw" V(mid),V(out)
  - Post Process .meas and .four
    - QPOST "%Qname%.cir" -o "%Qname%.out"



QUX.exe : Netlist a Schematic (.qsch)

- Syntax for QUX buildtimestamp
  - QUX.exe -buildtimestamp

```
C:\Program Files\QSPICE>QUX.exe -buildtimestamp

C:\Program Files\QSPICE>Build Nov  3 2023 09:11:08
```

- Syntax for –Netlist (** beware, -Netlist is case sensitive)
  - QUX.exe -Netlist <schematicfile> [-stdout]
    - <schematicfile> : name (+path) of a .qsch schematic, adds " " quotation for filename
    - If "-stdout" is not specified, the name of the netlist(.cir) file is computed from the name of the input .qsch file
    - [-stdout] : the netlist is printed on the console instead of to a file (not recommended since QSPICE employs a character set that most terminals can't handle

```
C:\QspiceKSKelvin\Qspice Batch>QUX -Netlist "%Qname%.qsch" -stdout

C:\QspiceKSKelvin\Qspice Batch>* LPF Circuit.qsch
L1 in mid 1
C1 mid 0 1
R1 out 0 1
V1 in 0 AC 1
L2 mid out 1
.ac dec 100 10K 1Meg
.plot V(mid) V(out)
.MEAS fc FIND frequency WHEN db(mag(V(out)))=-3
.end
```

QUX.exe : Export Datafile (.qraw)

- Syntax for -Export
  - QUX.exe -Export <datafile> <expr1[,expr2[,...]]> [Npoints] [CSV|SPICE|ASCII] [-stdout]
    - <datafile> : name of a .qraw file
    - <expr1[,expr2[,...]]> : expressions of data to extract
      - No space are allowed in the expression
      - Comma-separated expressions
    - [Npoints] : number of equally-spaced data points to extract
      - Default Npoints=1000
      - Npoints=1e308 or Npoints=all : all datapoints are extracted, waveform is not interpolated
    - [CSV|SPICE|ASCII]
      - CSV : Comma-Separated Value file
      - SPICE : .qraw in binary
      - ASCII : .qraw in ASCII
    - [-stdout] : extracted data is printed on the console instead of to a file
    - Example
      - QUX -Export "<filepath filename>" expr,expr2 all ascii ← no quotation mark is required for [Npoints] and [CSV|SPICE|ASCII]

QUX.exe : DLLvariables

- Syntax for –DLLvariables
  - Generate the variable declarations for the .DLL blocks in the schematic given by PATH
  - QUX.exe -DLLvariables <PATH> [-stdout]
    - -DLLvariables : case sensitive, generate the variable declaration for .DLL blocks
    - <PATH> : file path and file name as .qsch
    - [-stdout] : With [-stdout], result prints in Command Window. Without [-stdout], result prints to a file (.c).

```
C:\>QUX -DLLvariables "D:\KSKelvinQspice\01 User Guide and Script\03 Command Reference Guide\Batch Command\QUX\DLLvariables\example.QUXDLL.qsch" -stdout

C:\>// Instance: X1
// Module  : QUXDLL
   double  x    = data[0].d; // input
   double  gain = data[1].d; // input parameter
   double &y    = data[2].d; // output


// Instance: X2
// Module  : QUXDLL2
   double  a = data[0].d; // input
   double  n = data[1].d; // input parameter
   double &b = data[2].d; // output
```

QUX.exe : Plot Datafile (.qraw, .csv)

- Syntax for Plot
  - QUX.exe <datafile>
    - <datafile> : name of a .qraw or .csv file
    - Example : to plot a QUX-plot.qraw in C:\KSKelvinQspice\
      - path C:\Program Files\QSPICE\
      - QUX "C:\KSKelvinQspice\QUX-plot.qraw"

QSPICE64.exe and QSPICE80.exe : QSPICE Simulator

- Syntax for output data .qraw name same as netlist .cir name
  - QSPICE64.exe -binary <netlistname> : Binary file format for output data .qraw
  - QSPICE64.exe -ascii <netlistname> : Ascii file format for output data .qraw
  - If 80 bit is used, change QSPICE64 to QSPICE80
- Syntax for specify output data .qraw name
  - QSPICE64.exe -[ascii/binary] <netlistname> -r <path> : specify the name of output data file
  - Example
    - set Qname=LPF Circuit
    - QSPICE64 -ascii "%Qname%.cir" -r "%Qname%-ascii.qraw"
- Syntax to directs the .qraw output to null (not saving a .qraw)
  - QSPICE64.exe <netlistname> -r NUL
    - This special usage is for user who write C++ dll datalogger and not prefer a .qraw to generate when simulating with batch mode

.qraw Binary Data format

Binary-binary.qraw

```
Title: * Binary.qsch
Date: Sun Nov  5 21:41:33 2023
Plotname: DC Transfer Characteristic
Flags: real
Abscissa: 1.000000000000000e+00 5.000000000000000e+00 lin
No. Variables: 4
No. Points: 3
Command: QSPICE64, Build Nov  3 2023 09:29:29
.param temp=27
Variables:
    0   V1  voltage
    1   V(a)    voltage
    2   I(V1)   current
    3   P(V1)   power
Binary:
NUL NUL NUL NUL NUL NUL ð? NUL NUL NUL NUL NUL NUL ð? NUL NUL NUL NUL NUL
```

Binary vs Ascii

Binary-binary.qraw | Binary-ascii.qraw

```
Title: * Binary.qsch
Date: Sun Nov  5 21:41:38 2023
Plotname: DC Transfer Characteristic
Flags: real
Abscissa: 1.000000000000000e+00 5.000000000000000e+00 lin
No. Variables: 4
No. Points: 3
Command: QSPICE64, Build Nov  3 2023 09:29:29
.param temp=27
Variables:
    0   V1  voltage
    1   V(a)    voltage
    2   I(V1)   current
    3   P(V1)   power
Values:
0       1.000000000000000e+00
    1.000000000000000e+00
    0.000000000000000e+00
    0.000000000000000e+00
1       3.000000000000000e+00
    3.000000000000000e+00
    0.000000000000000e+00
    0.000000000000000e+00
```

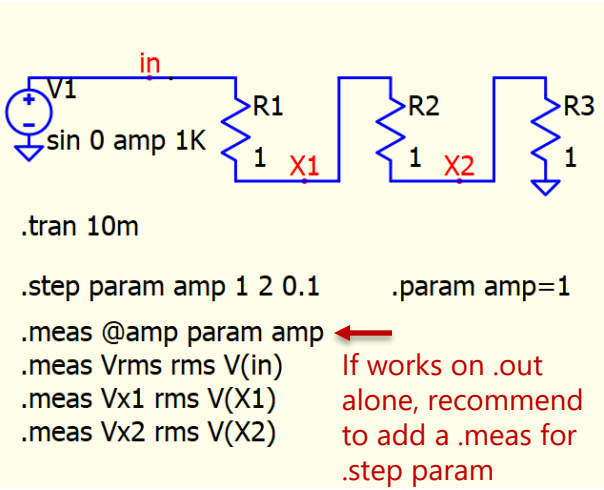
Newline </n>

```
00000130 76 6f 6c 74 61 67 65 0a 09 32 09 49 28 56 31 29 voltage..2.I(V1)
00000140 09 63 75 72 72 65 6e 74 0a 09 33 09 50 28 56 31 .current..3.P(V1
00000150 29 09 70 6f 77 65 72 0a 42 69 6e 61 72 79 3a 0a ).power.Binary:.
00000160 00 00 00 00 00 00 f0 3f 00 00 00 00 00 00 f0 3f .......ð?.......ð?
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 00 00 00 00 00 00 08 40 00 00 00 00 00 00 08 40 .......@.......@
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001a0 00 00 00 00 00 00 14 40 00 00 00 00 00 00 14 40 .......@.......@
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Binary Format : Float 64

QPOST.exe : Post Processor to execute .meas and .four

- Syntax for Qpost.exe
  - Qpost <netlistname> -o <consoleoutput>
    - This will write .meas and .four results into a file for the console output
    - This result is equivalent in Qspice Post Process Output Window after Simulation is run




```
path C:\Program Files\QSPICE\

set Qname=resistor_network

set Qpath=C:\QspiceKSKelvin\01 User Guide and Script\03 Qspice Reference Guide\Batch Command\Qpost

cd %Qpath%

QUX -Netlist "%Qname%.qsch"

QSPICE64 -binary "%Qname%.cir"

QPOST "%Qname%.cir" -o "%Qname%.out"
```

resistor_network.out

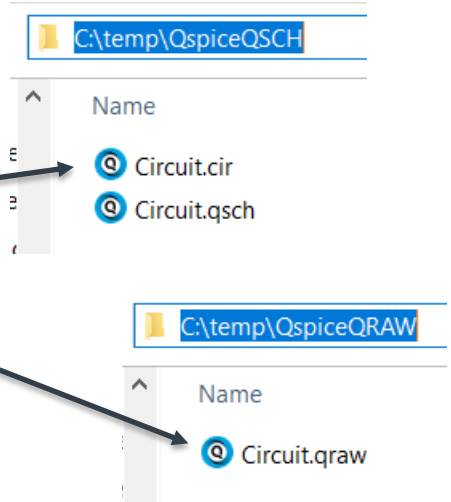
```
.meas @amp param amp:
    0         1
    1       1.1
    2       1.2
    3       1.3
    4       1.4
    5       1.5
    6       1.6
    7       1.7
    8       1.8
    9       1.9
   10         2
.meas vrms rms v(in):
    0   0.707107
    1   0.777817
    2   0.848528
    3   0.919239
    4   0.989949
    5    1.06066
    6    1.13137
    7    1.20208
    8    1.27279
    9    1.3435
```

QPOST.exe : Post Processor to execute .meas and .four

- Syntax for Qpost.exe
  - Qpost <netlistname> -r <.qraw file> -o <consoleoutput>
    - This is to load .meas/.four from a netlist file (*.cir)
    - e.g. QPOST "%Qname%.cir" -r "%Qname%.qraw" -o "%Qname%.out“
  - Qpost <meas script> -r <.qraw file> -o <consoleoutput>
    - This is to load .meas/.four from a separated text file contains these directives
    - e.g. QPOST "meas script.txt" -r "%Qname%.qraw" -o "%Qname%.out“

Example : Save data file (.qraw) into different directory

- Save data file (.qraw) into different directory
  - This example assume
    - Qspice installed in C:\Program Files\QSPICE\
    - Schematic file in C:\temp\QspiceQSCH\
    - Circuit schematic named Circuit.qsch
    - You want the data file to be stored in C:\temp\QspiceQRAW\
  - Batch Command in CMD
    - path C:\Program Files\QSPICE\
    - cd C:\temp\QspiceQSCH\
    - QUX -Netlist "Circuit.qsch"
    - QSPICE64 -binary "Circuit.cir" -r "C:\temp\QspiceQRAW\Circuit.qraw"
    - QUX "C:\temp\QspiceQRAW\Circuit.qraw"

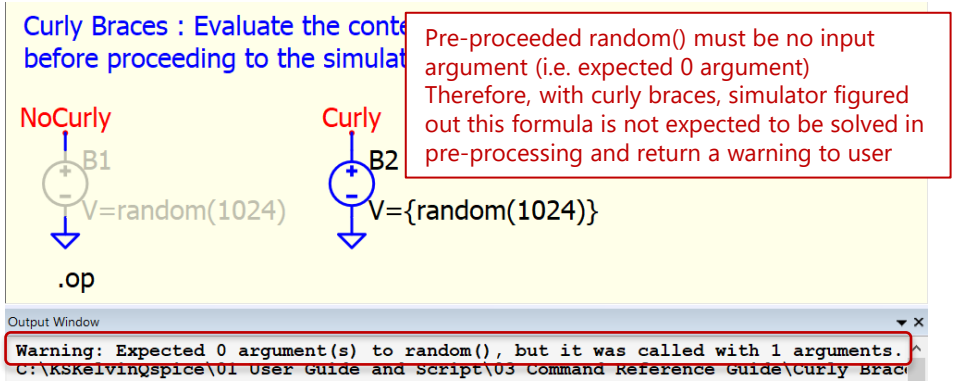


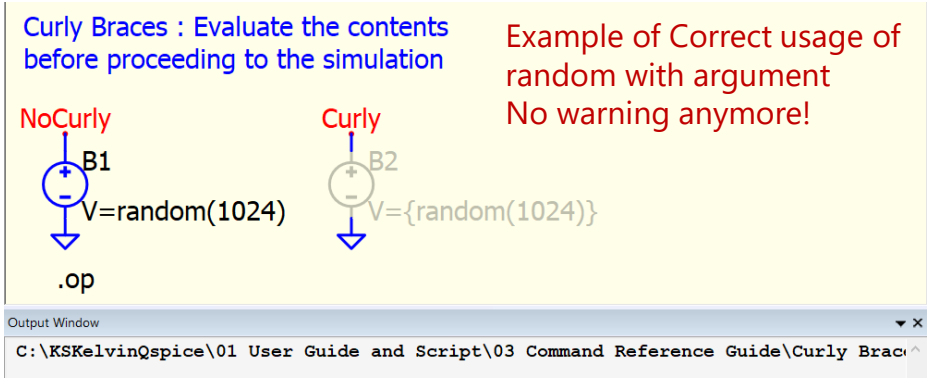
Curly Braces {}

Curly Braces { }

Qspice : Curly Braces with Random.qsch

- Curly Braces { }
  - In Pspice and LTspice, the curly braces always meant evaluate the contents before proceeding to the simulation
  - But Qspice tries to figure out what can be solved before the simulation by itself and remove the necessary of using curly braces { }
    - Therefore, Qspice can call to use a parameter or a formula without curly braces { }
  - But user can still observe effect of curly braces in Qspice, an example is B-source with random(x) [with argument] function in B-source





** Qspice determines it is not a pre-process function and continues to run after warning is returned

Timestep too small

Timestep too small (.tran) – Guideline for .option

- Timestep too small (.tran) – Guideline for .option
  - **.option fastmath=0** : Disable fastmath (Qspice64.exe) and use Qspice80.exe. This will allow for 80-bit math calculations, which are more precise but may result in slightly slower simulations [Only for Qspice]
  - .**option trtol=7 method=gear** : If the model is a PSpice model, you may want to consider this. PSpice uses a different trtol and integration method. Be cautious as gear integration adds dampers to the circuit, suppressing oscillations but also introducing more errors. PSpice uses gear integration to prevent trapezoidal oscillations from trapezoidal integration
  - **.option gshunt=<value>** or **.option cshunt=<value>**, These options add conductance or capacitance from every node to the ground. They may help the simulation to converge by providing a ground path for high-frequency oscillations or numerical noise. Start by setting the value to 1e-12 or 1e-11
  - **.option maxstep=<value>** : Force a maximum timestep! Check your switching frequency or highest operating frequency and set a reasonable maxstep
  - **.option ITL4=<value>** : This option increases the number of transient iteration limits. Consider setting it to 100 or 1000 to see if there is any improvement by allowing more iterations
  - If a timestep too small occurs right after bias point analysis in .tran, it may be related to the initial voltage/current condition, and you may consider trying one of the following
    - Add UIC in .tran to skip the bias point analysis (.op)
    - Change the DC supply source to a PWL source, which ramps the voltage from 0 to the desired setpoint (e.g., change DC 10 to PWL 0 0 1n 10)
  - Investigate the 3rd party sub-circuit model, as it may consist of devices or equations that can cause the timestep to be too small. However, this troubleshooting normally requires a lot of experience
  - Some people may change **vntol**, **abstol**, **reltol**, but user needs deeper knowledge if they change these parameters in .option and I normally not to do that

SPICE methods

- SPICE methods (All you need to know)
  - [G][V]=[I]
    - Node count is more important than component count
    - Avoid circuit elements represented as Thevenin or trans-Thevenin equivalents
  - Newton-Raphson Iteration
    - All I-V curves must be continuous in value and slope
    - Don’t make anything more non-linear than necessary
    - Every nonlinear element should be bypassed with some capacitance

- ** SPICE-based and PWL-based
  - There are primarily two different circuit simulation concepts
    - SPICE-based (SPICE engine) : Qspice, PSpice, LTspice, TINA, NgSPICE etc…
    - PWL-based (Piecewise Linear simulation engine) : SIMPLIS, PSIM, NL5 etc…
  - SPICE-based models device equations. To simulate a near-ideal device, it generally requires more attention and a higher level of understanding of how to set up the simulation as it was not designed to handle discontinuity in I-V curves
  - PWL simulation engines model devices with piecewise linear functions and focus on resolving ideal switching circuit elements. They are designed to handle steep I-V curves

Prevent discontinuities in the I-V curve

Qspice : SmoothIV-Diode-Epsilon.qsch | SmoothIV-SW-VH.qsch

- Remove discontinuities in the I-V curve
  - For behavioral diode model, consider the following
    - Ratio between Ron and Roff not too extreme
    - Add an instance parameter epsilon to smooth the transition
  - For switch, consider the following
    - Use –ve Vh for smoothly transition switch
    - Add ETA if further smoothen is required