電源回路設計の安定化の評価と電子回路の事前検証

アナログ・デバイセズ株式会社
原田秀一

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

"多くの電子回路設計者が遭遇するいろいろなトラブル“

電源回路の設計はその負荷が決まってからでないと詳細に検討できません。必要な機能の設計が終わってから短時間で、そして確実に安定に動く電源回路を設計しなければなりません。今回は電源回路の安定性評価方法と測定器がない時の対処、さらにLTspiceで位相余裕を測定する方法をご紹介いたします。

また高次フィルタを例にとって、部品バラつきによる特性変化に対処する評価をLTspiceで行う方法、

特に.funcコマンドを使用した例をご紹介いたします。

1. 電源回路の安定化評価
   位相余裕の測定方法
   測定装置がない場合の対処方法
   LTspiceによるボード線図の作成方法

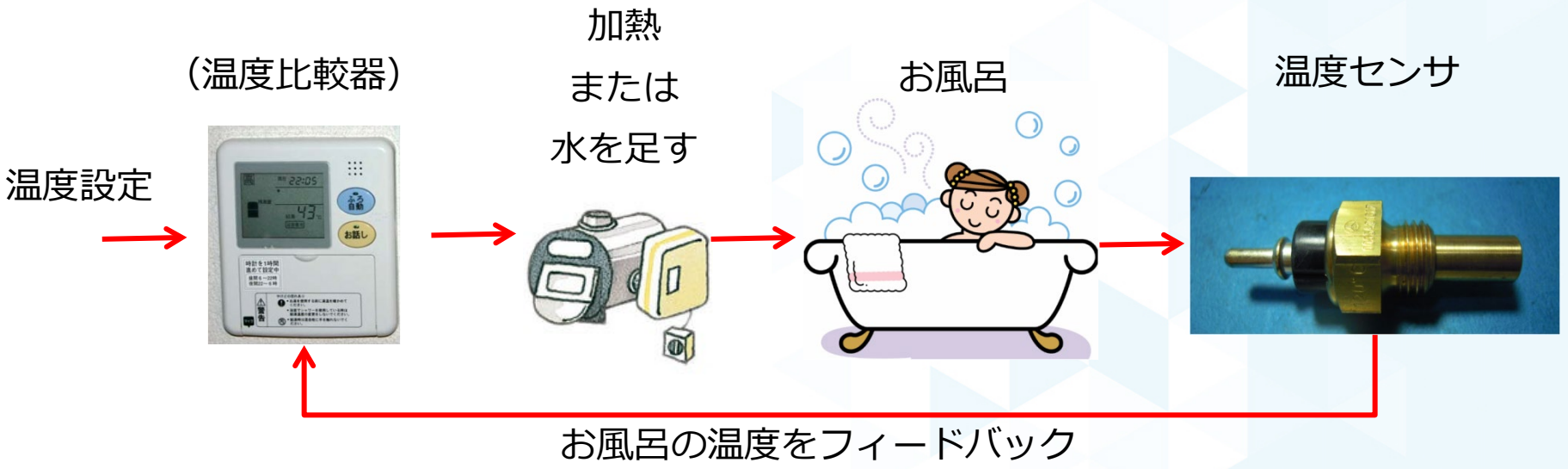
2. 電子回路の事前検証
   受動部品の誤差がどの程度周波数特性に影響するか
   LTspiceでの検証方法
   最悪の場合を .func コマンドを使用して検証

電源回路の安定性評価　（位相余裕の測定）

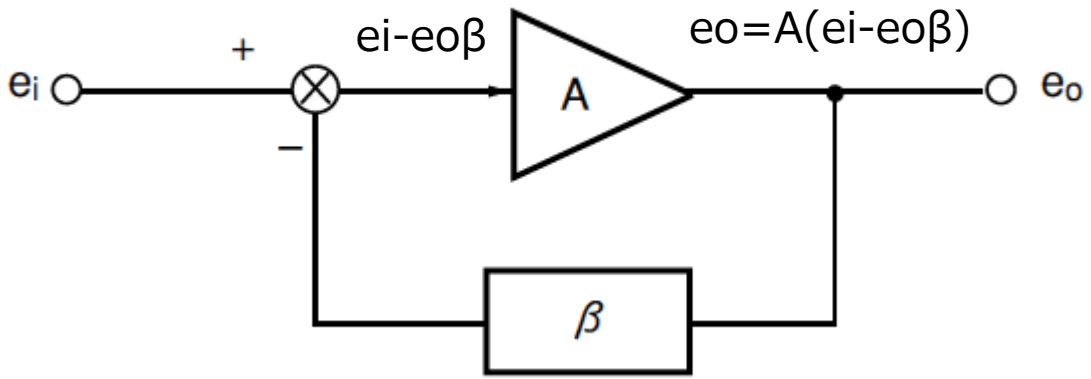
負帰還制御と発振

負帰還：電源回路の出力電圧安定化のために必要

風呂の自動温度設定



負帰還回路のブロックダイアグラム



系のゲインは $\frac{e_0}{e_i} = \frac{A}{1 + A\beta}$

アンプのゲインAは周波数に依存する。
周波数が高くなるとAは低下する

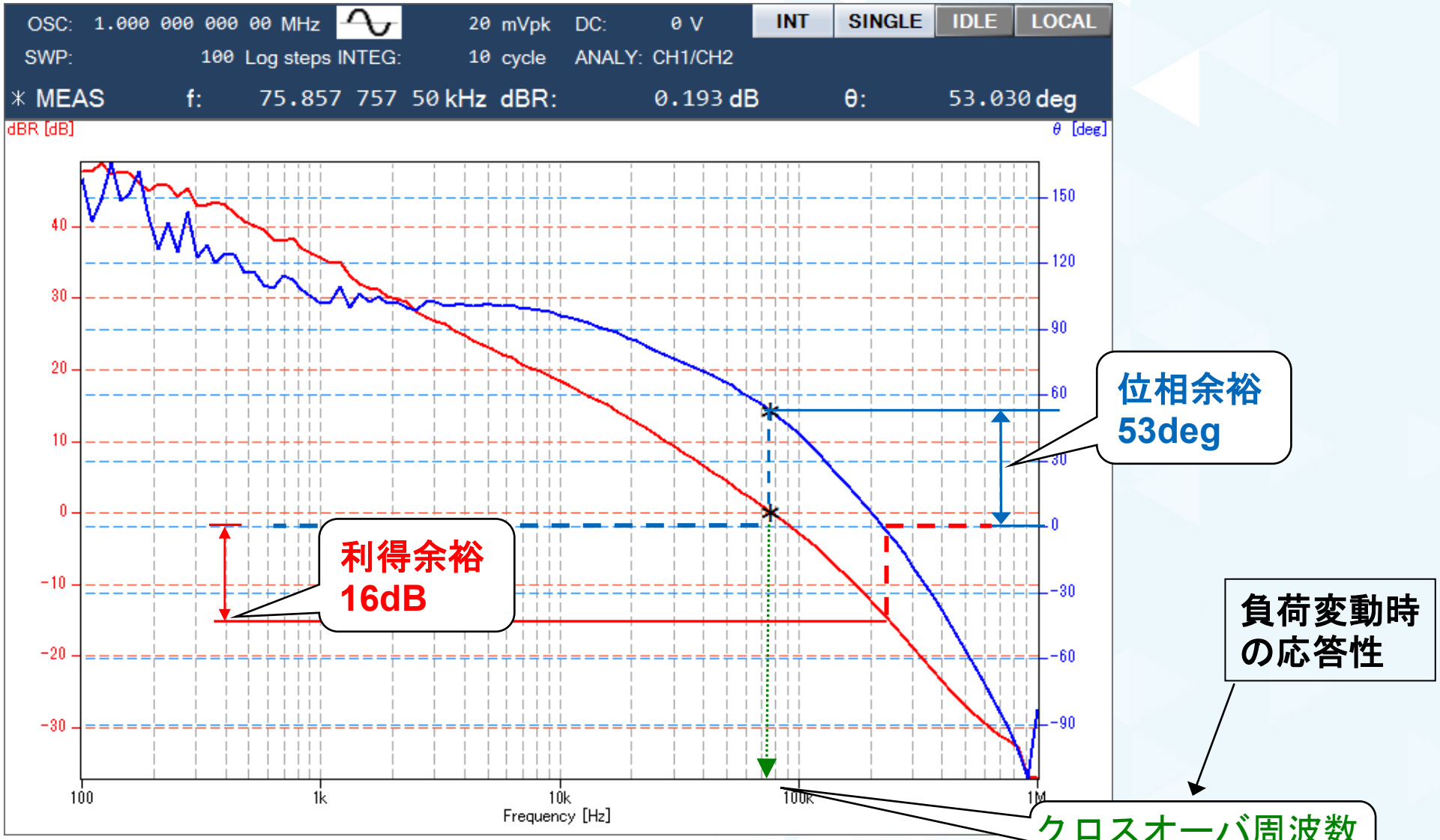
βを変えると通過帯域も変わる

アンプのゲインAが非常に
大きければ系のゲインは1/β
で求められる。

負帰還を掛ける目的は？

- オープンループゲインのばらつきを抑えられる
- ゲイン一定の帯域を広くとれる
- 出力インピーダンスが1/(1+Aβ)と低くなる

ボード線図



一般的な指標

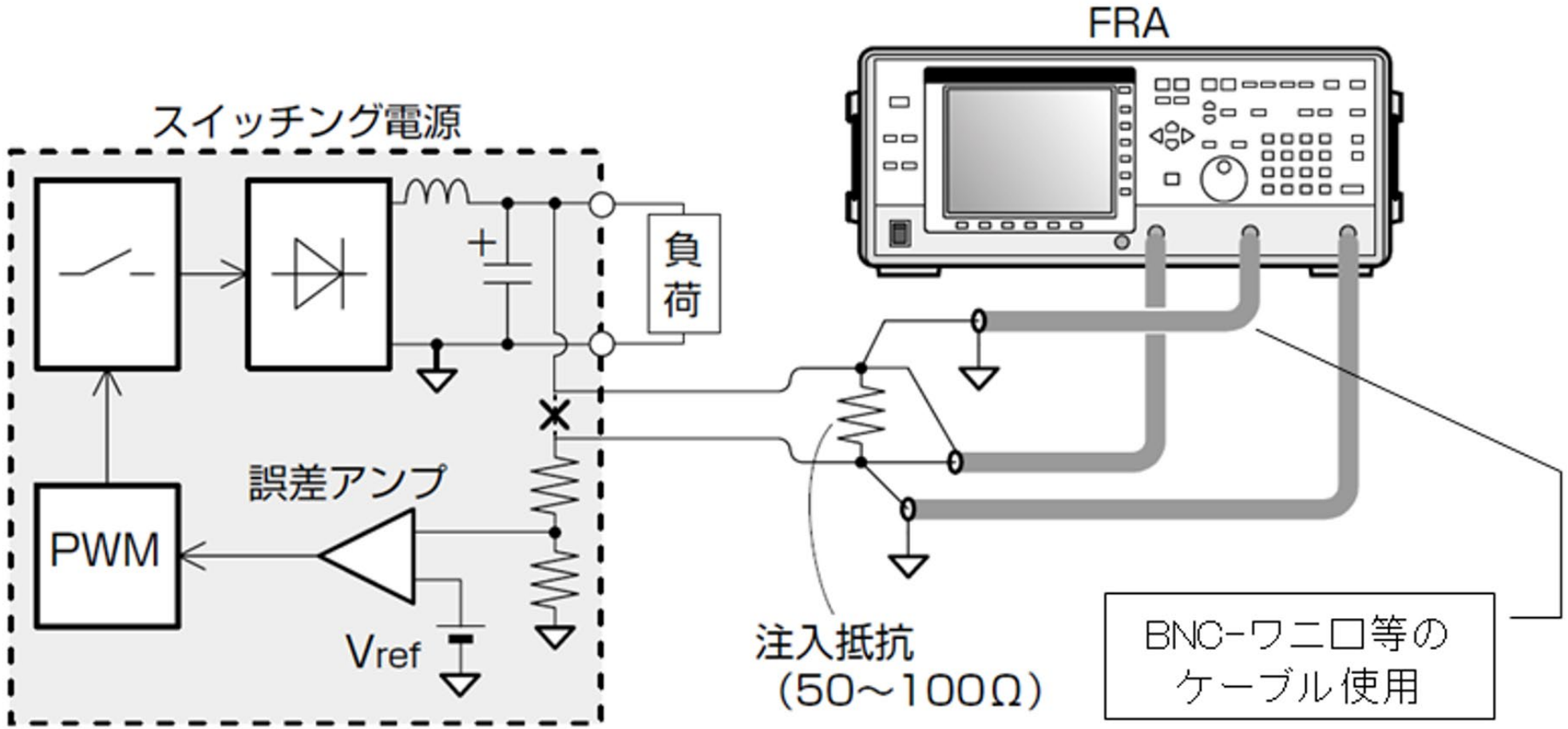
位相余裕[度]	利得余裕[dB]	状態
20	3	ひどりリンギング　絶対的に悪い値
30	5	多少のリンギング　やや悪い値
45	7	きわどいダンピング　最良の応答時間
60	10	一般的に適切な値
72	12	基準としたい値　閉ループ応答でピークが出ない

※上記はあくまでも参考値　　条件、仕様によって変わります。

（株）エヌエフ回路設計ブロック殿の資料より抜粋

ボード線図取得時の実際の接続

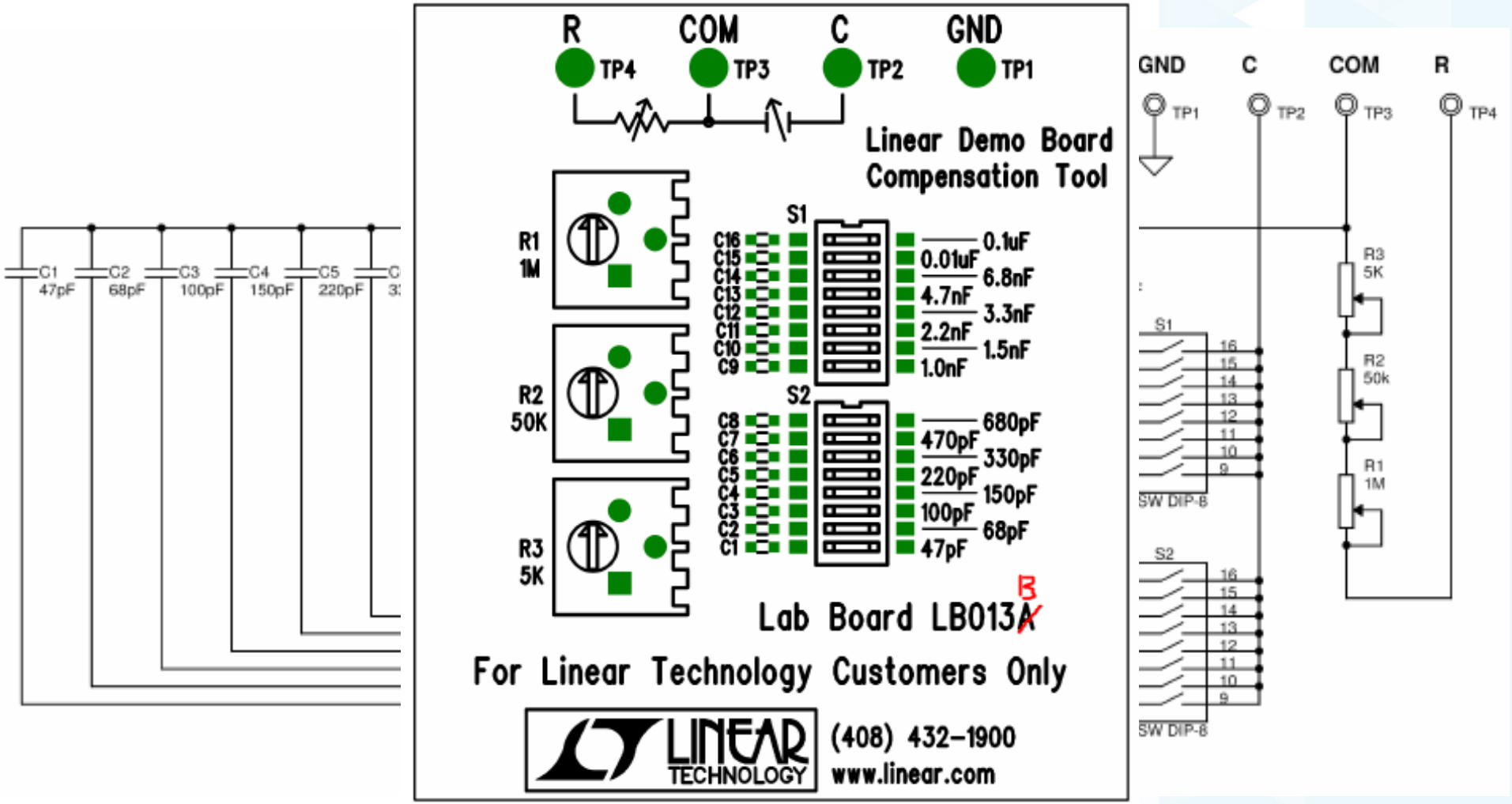
FRA (**F**requency **R**esponse **A**nalyzer) の接続

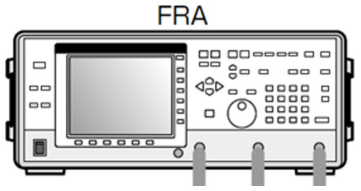


（株）エヌエフ回路設計ブロック殿の資料より抜粋

位相補償の調整

位相調整用　CR基板



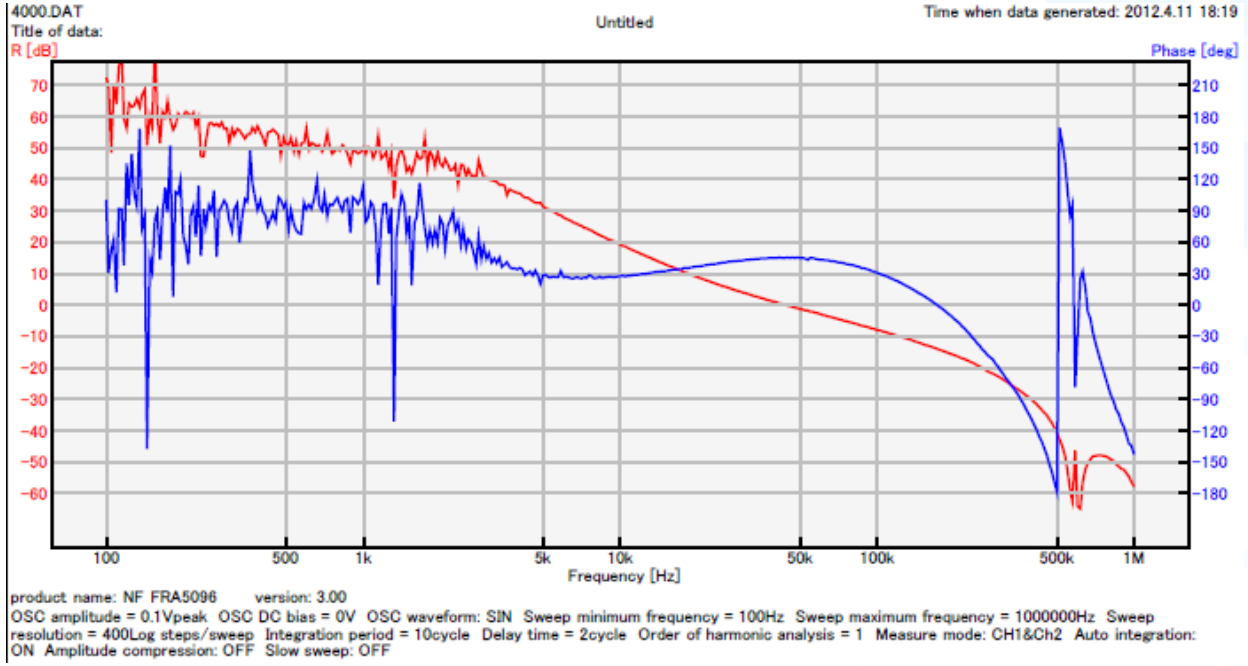


FRAがない時

ボード線図とトランジェントレスポンスの関係

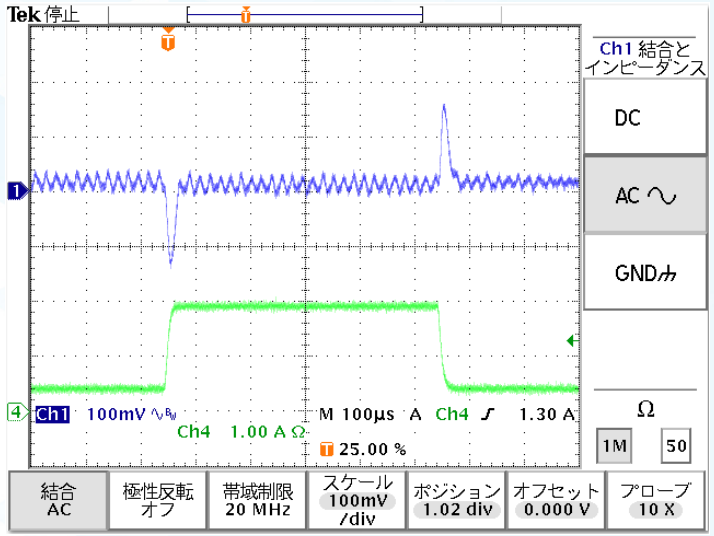
入力電圧：15V

出力電圧電流：5V/2A　CRモード



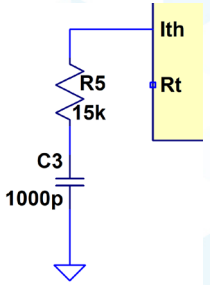
0.5A/2Aスイッチング

電流トランジェント：10us



クロスオーバー周波数： 43kHz

位相余裕：45度

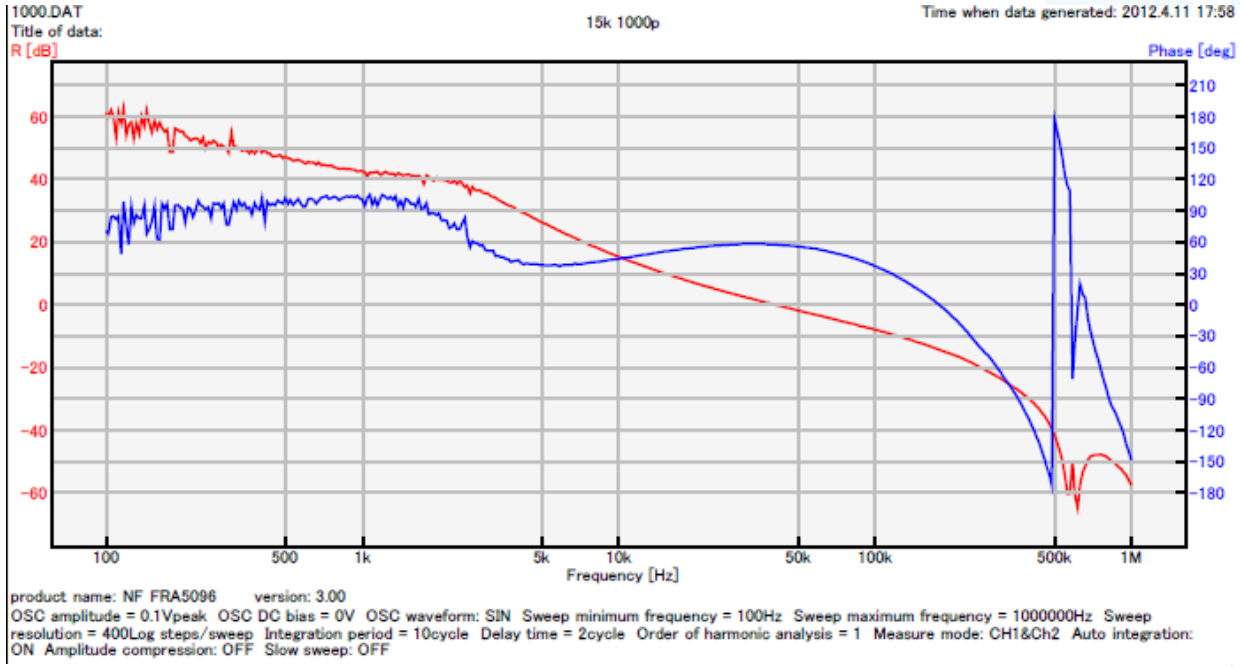


15kΩ+470pF

まず半分でやってみる

入力電圧：15V

出力電圧電流：5V/2A

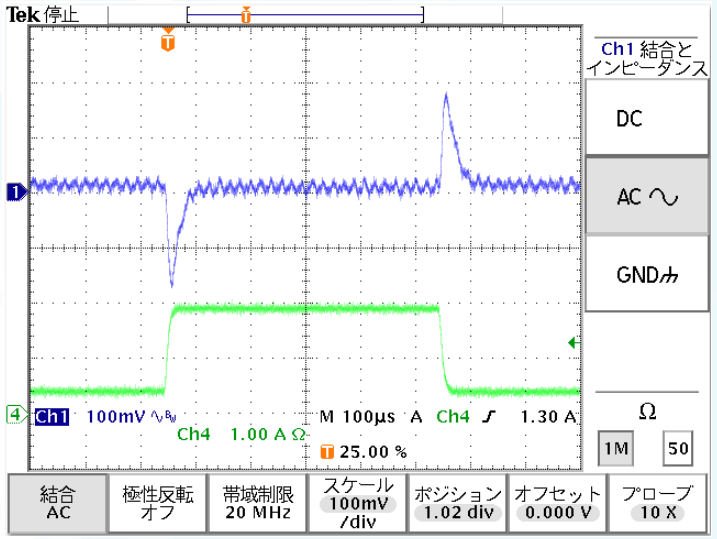


クロスオーバー周波数： 41kHz

位相余裕：58度

0.5A/2Aスイッチング

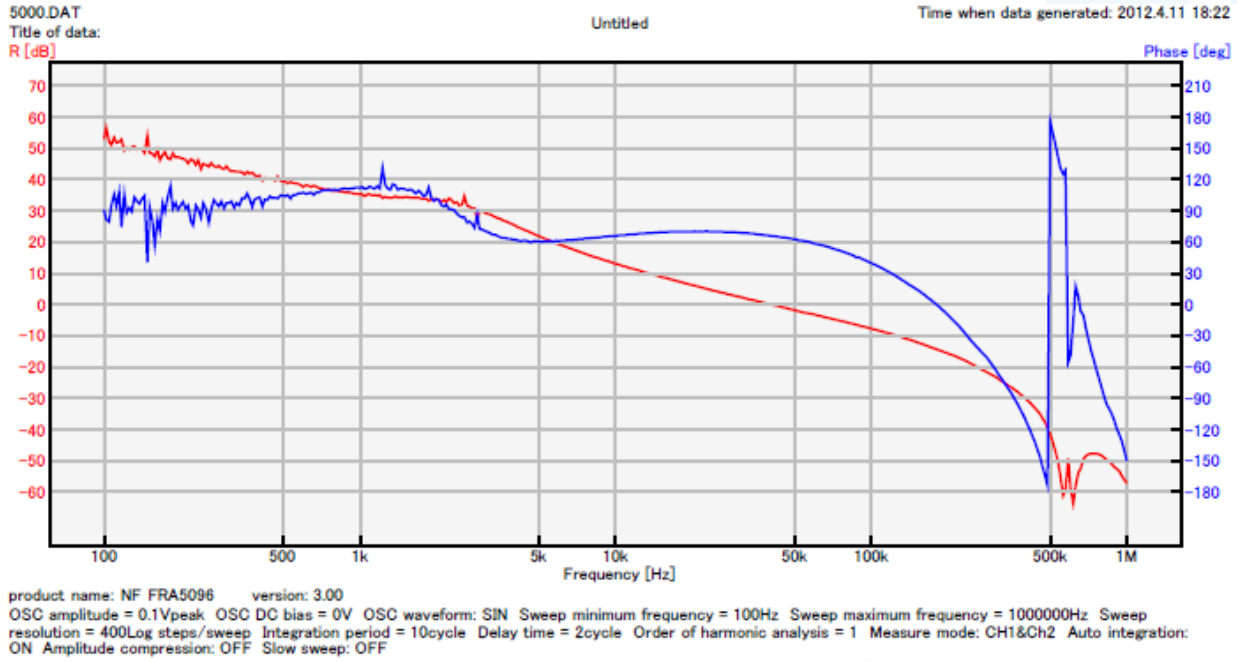
電流トランジェント：10us



15kΩ+1000pF

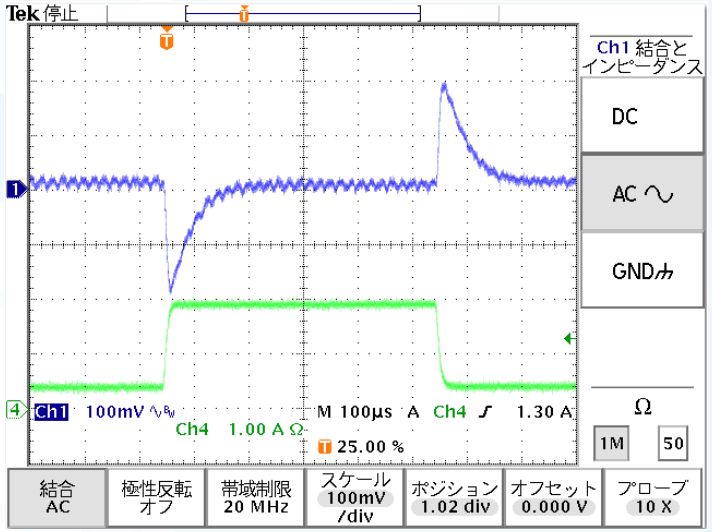
入力電圧：15V

出力電圧電流：5V/2A



0.5A/2Aスイッチング

電流トランジェント：10us



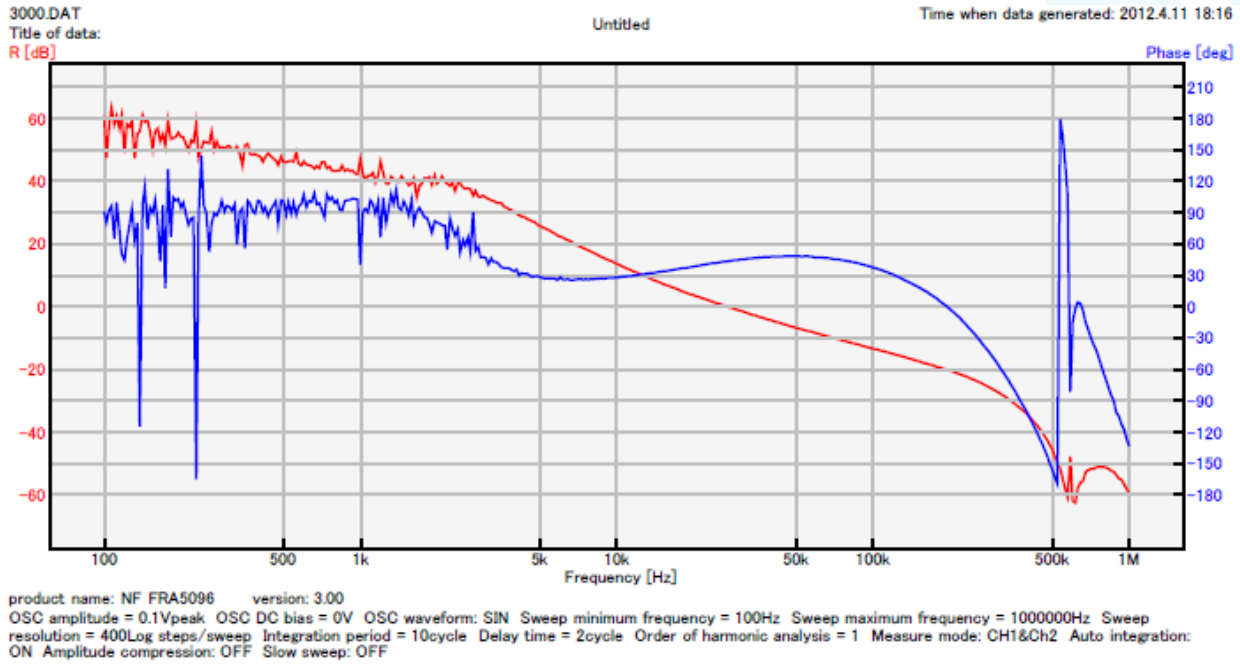
クロスオーバー周波数： 41kHz

位相余裕：66度

15kΩ+2200pF

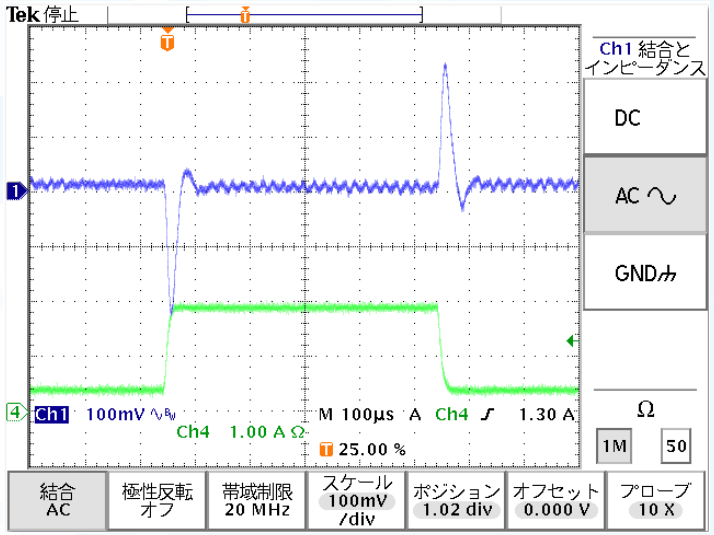
入力電圧：15V

出力電圧電流：5V/2A



0.5A/2Aスイッチング

電流トランジェント：10us



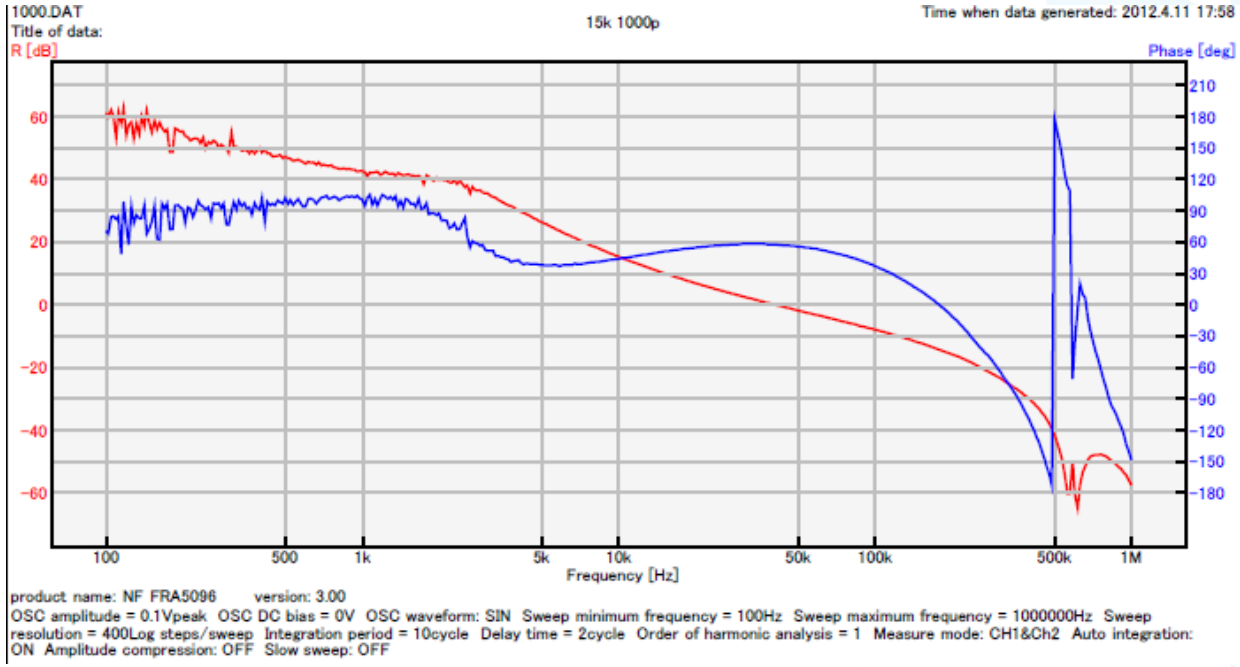
クロスオーバー周波数： 27kHz

位相余裕：43度

7.5kΩ+1000pF

入力電圧：15V

出力電圧電流：5V/2A

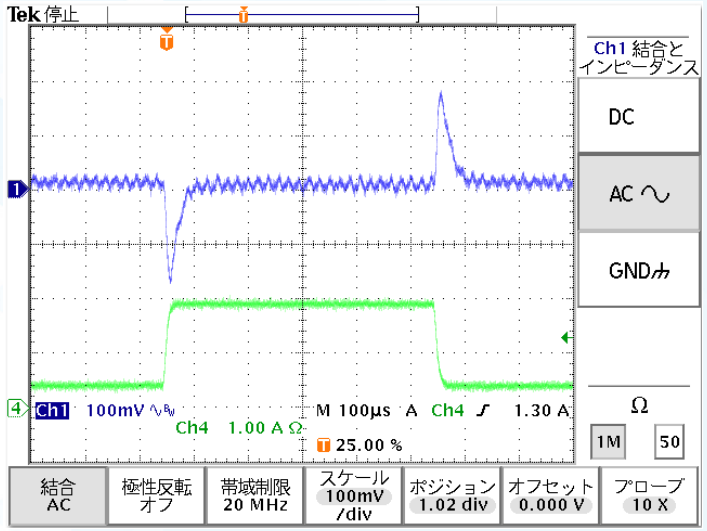


クロスオーバー周波数： 41kHz

位相余裕：58度

0.5A/2Aスイッチング

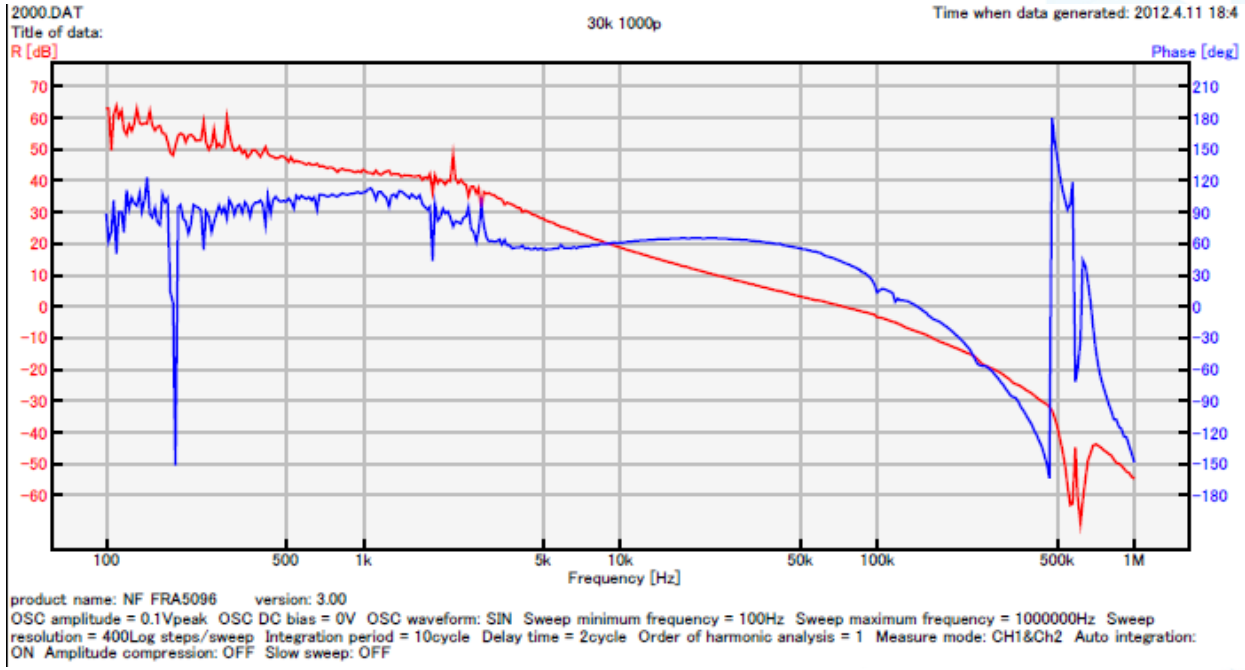
電流トランジェント：10us



15kΩ+1000pF

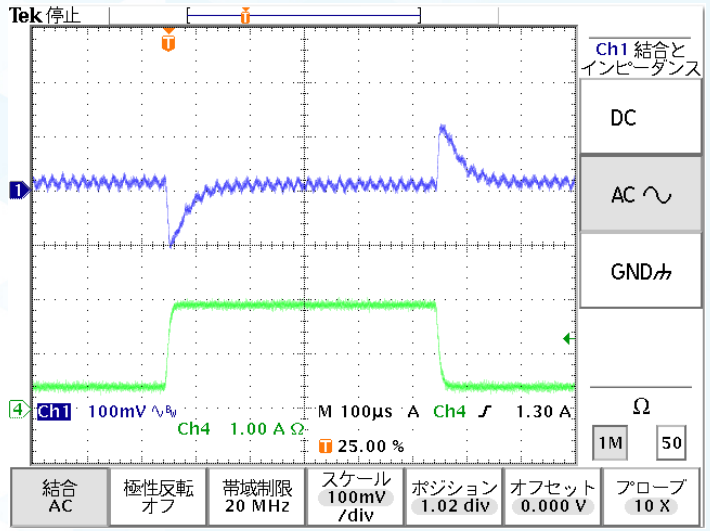
入力電圧：15V

出力電圧電流：5V/2A



0.5A/2Aスイッチング

電流トランジェント：10us

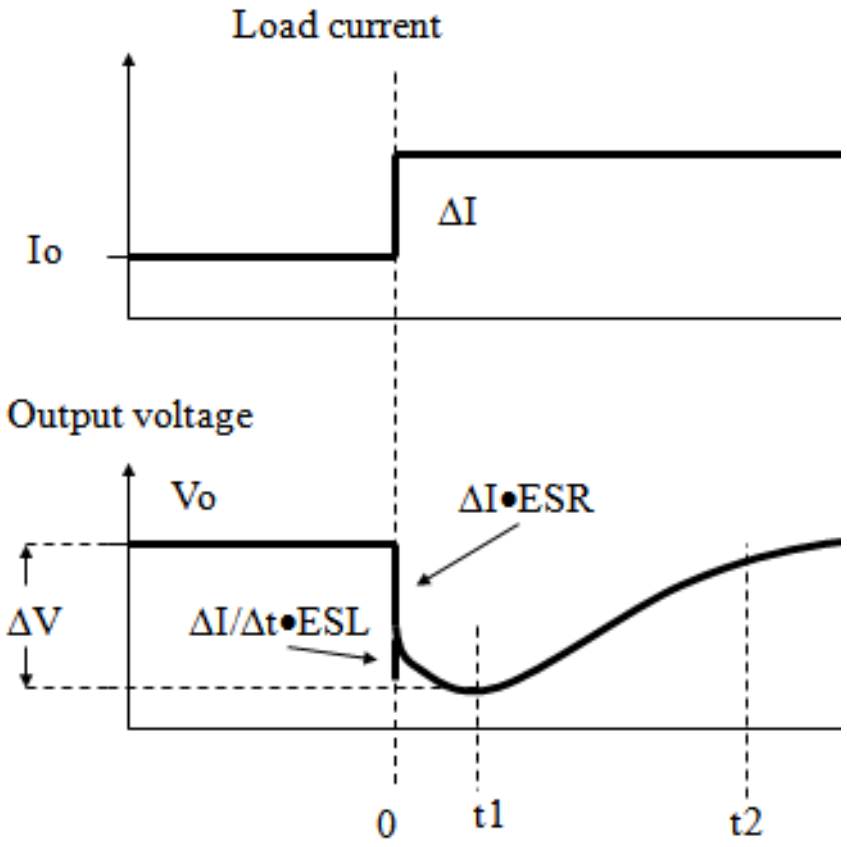


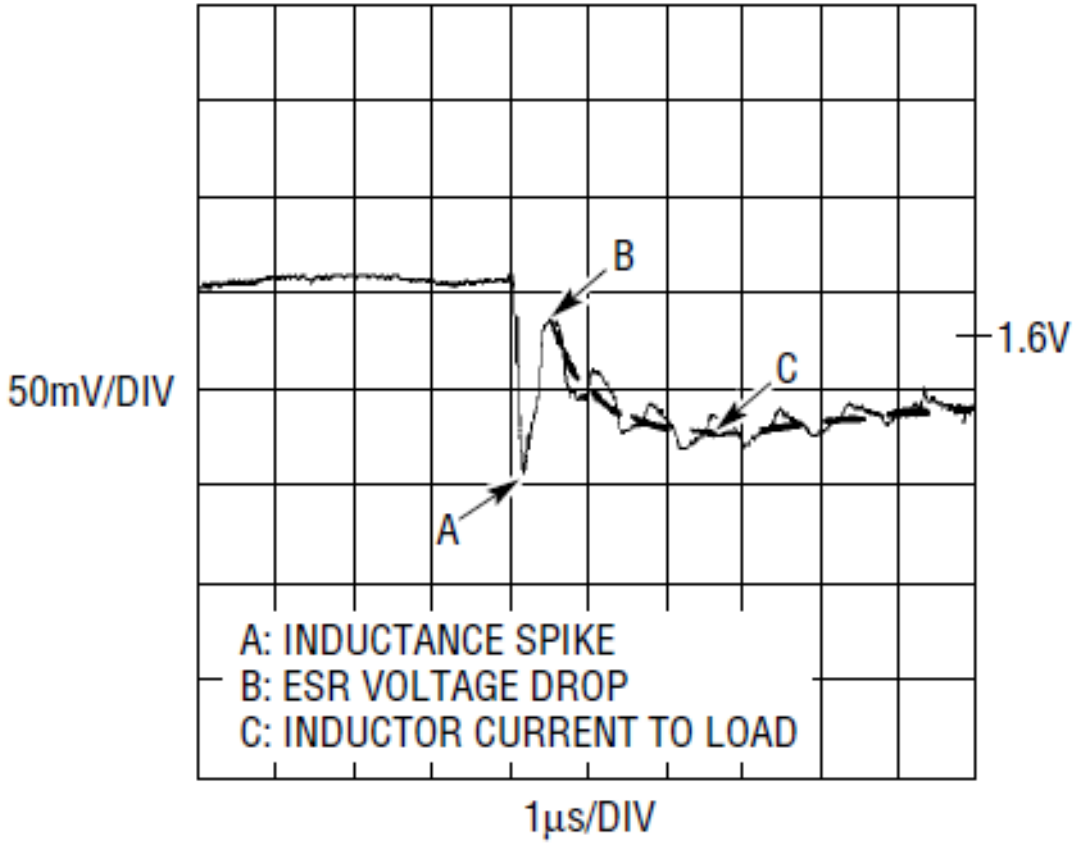
クロスオーバー周波数： 74kHz

位相余裕：42度

30kΩ+1000pF

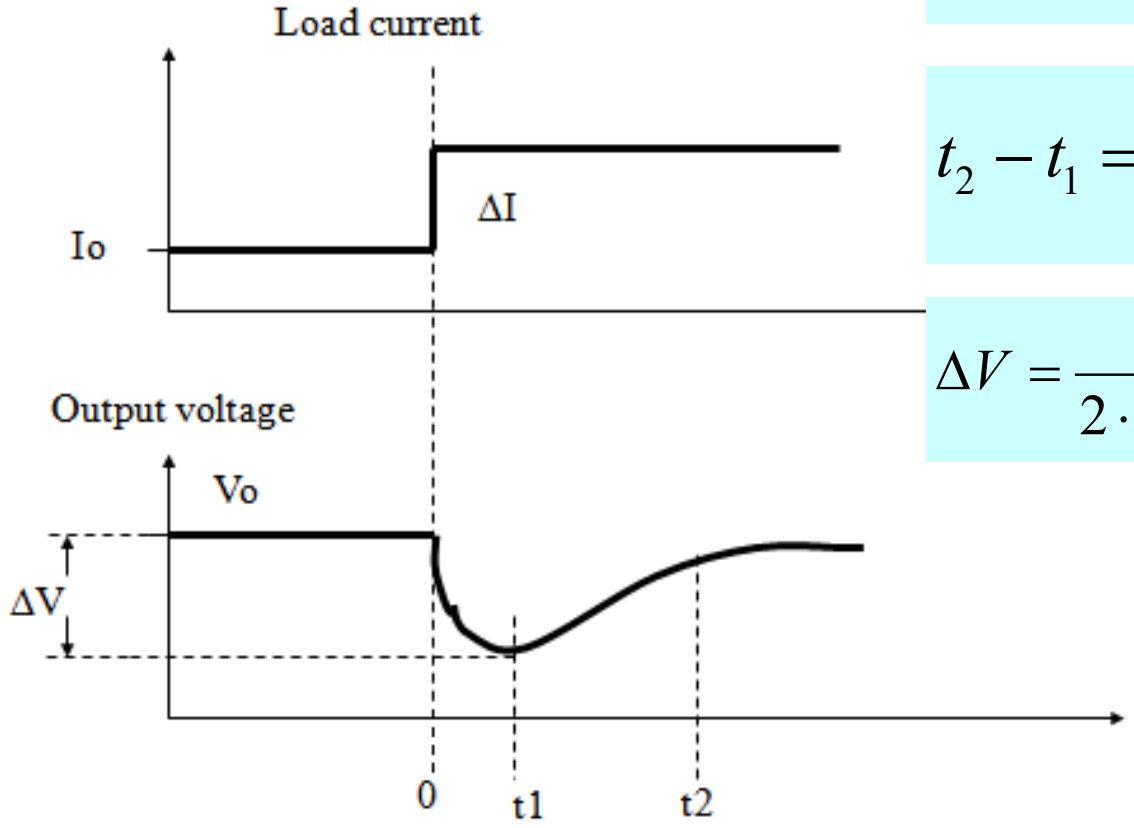
降圧電源のロードトランジェント





ロードトランジェント波形からわかること

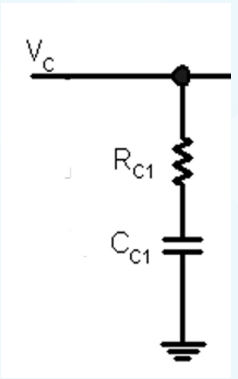
$$20° < \varphi_m < 76°$$



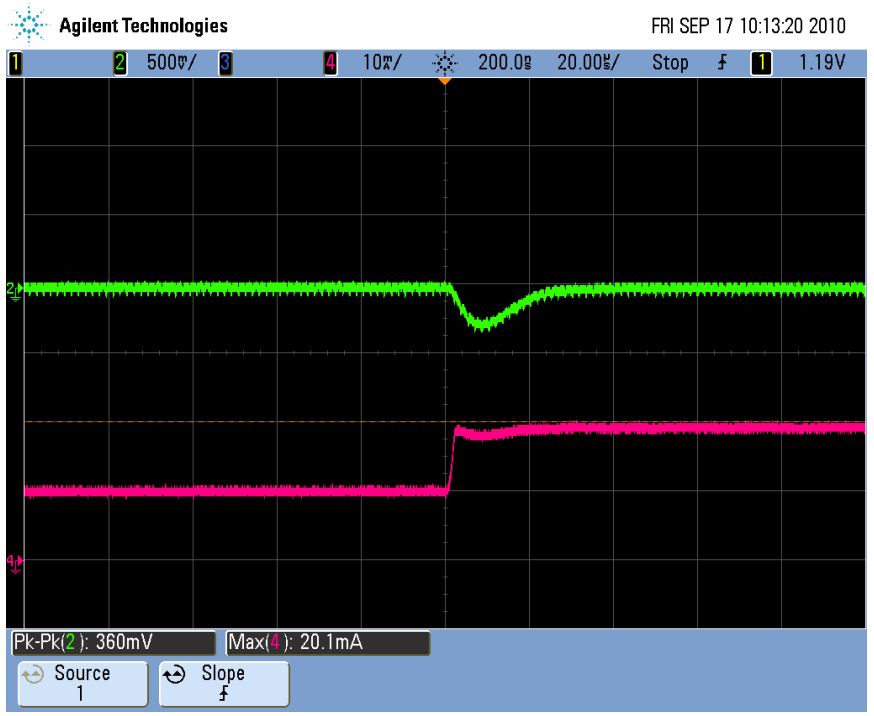
$$t_1 = \frac{1}{\pi \cdot f_c}$$

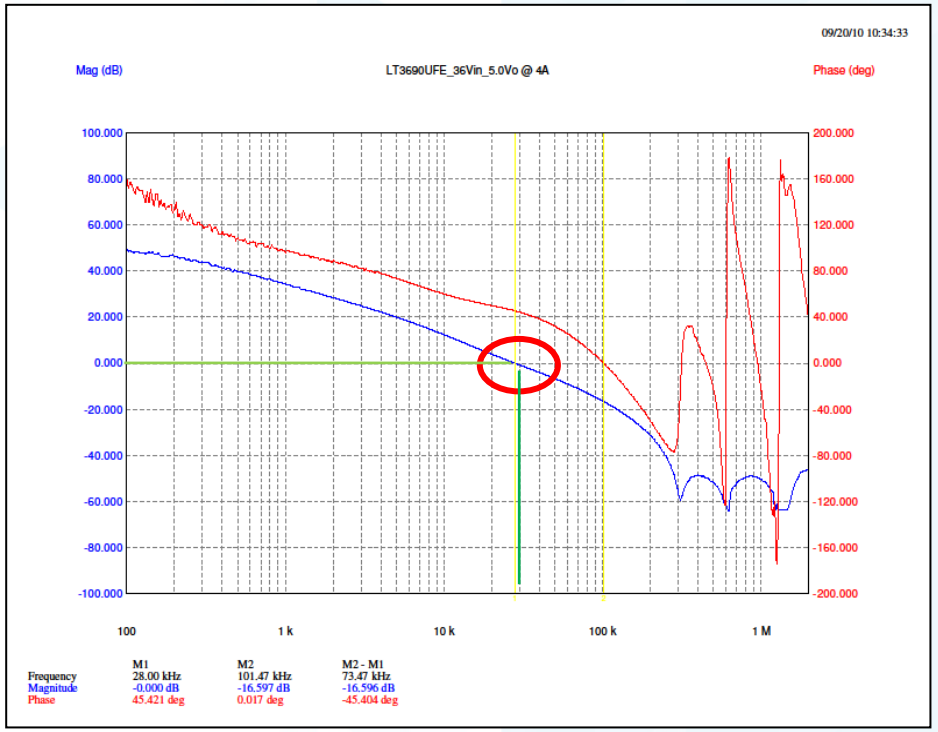
$$t_2 - t_1 = \frac{1}{2\pi \cdot f_z}$$ =Rc1*Cc1　リカバリ時間 (63% of ΔV　)

$$\Delta V = \frac{\Delta I}{2 \cdot \pi \cdot f_c \cdot C_{out}}$$



fc　：　クロスオーバー周波数
φ_m　：　位相余裕
fz　：　ゼロ周波数=1/(2πRc1Cc1)

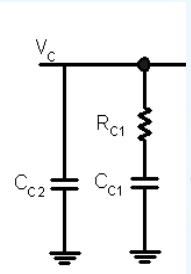




クロスオーバー周波数 $f_c = \frac{1}{\pi \cdot t_1} = \frac{1}{3.14 \cdot 9\mu} = 35kHz$ (ボード線図: 30kHz)

ゼロ周波数 $f_z = \frac{1}{2\pi \cdot R_{c1} \cdot C_{c1}} = \frac{1}{2 \cdot 3.14 \cdot 15k\Omega \cdot 0.68nF} = 15kHz$

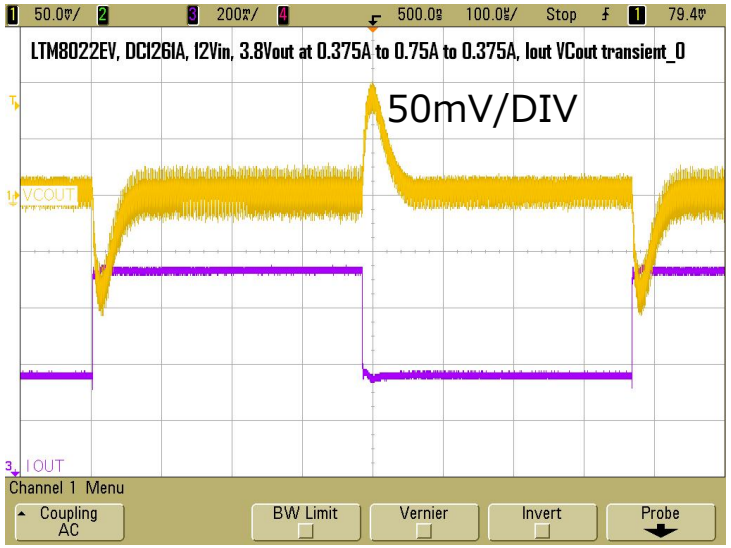
リカバリ時間 $t_2 - t_1 = \frac{1}{2\pi \cdot f_z} = R_{c1} \cdot C_{c1} = 15k\Omega \cdot 0.68nF = 10.2\mu s$



15k
680p

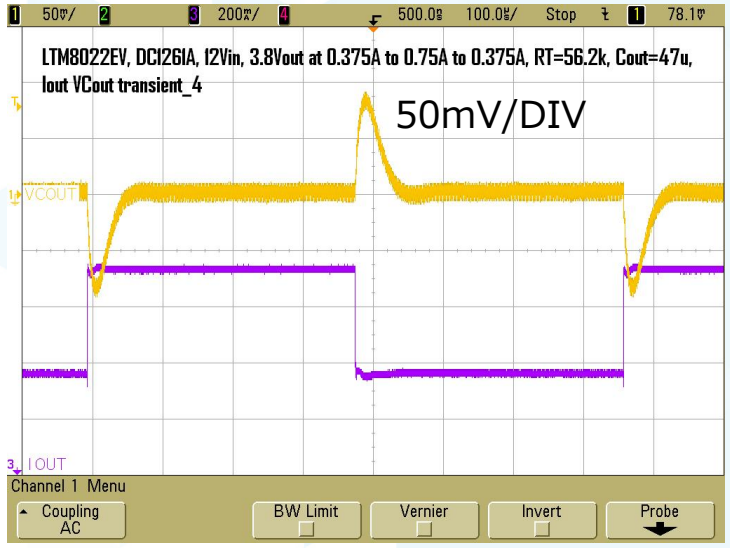
Coutを増やしてもステップレスポンスが変化しない？

80mV のステップレスポンスを 半分 にしようと Cout を増したがあまり効果はなかった



22uF

Coutを増すと



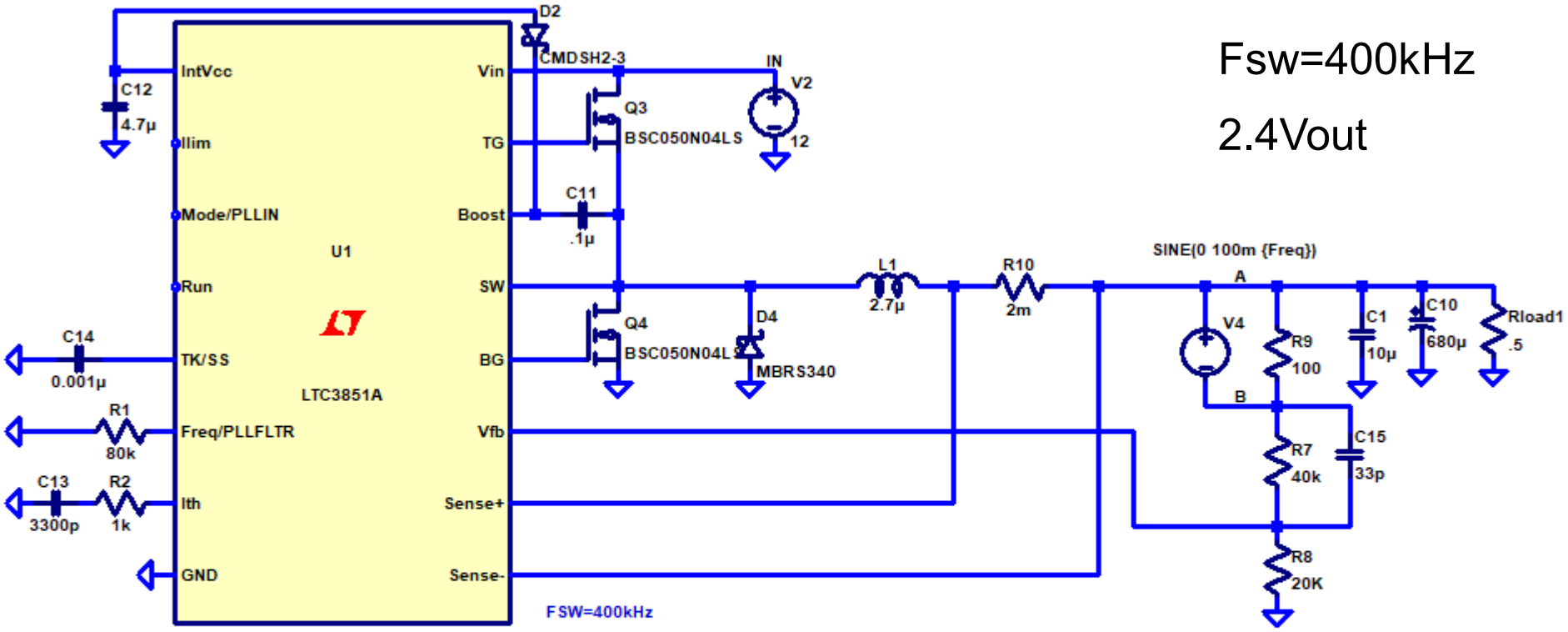
47uF

$$\Delta V = \frac{\Delta I}{2 \cdot \pi \cdot f_c \cdot C_{out}}$$

$$f_c \propto \frac{R_{C1}}{C_{out}} k$$

ΔVはCoutの関数ではない

LTspiceでボード線図を描く



Fsw=400kHz

2.4Vout

```
.measure Aavg avg V(a)
.measure Bavg avg V(b)
.measure Are avg  (V(a)-Aavg)*cos(360*time*Freq)
.measure Aim avg -(V(a)-Aavg)*sin(360*time*Freq)
.measure Bre avg  (V(b)-Bavg)*cos(360*time*Freq)
.measure Bim avg -(V(b)-Bavg)*sin(360*time*Freq)
.measure GainMag param 20*log10(hypot(Are,Aim) / hypot(Bre,Bim))
.measure GainPhi param mod(atan2(Aim, Are) - atan2(Bim, Bre)+180,360)-180
```

```
.step oct param Freq 20K 80K 3
.ic V(A)=2.4
.param t0=1m
.tran 0 {t0+5/Freq} {t0} 10n
```

コマンドの説明

```
.measure Aavg avg V(a)                                                    ①
.measure Bavg avg V(b)
.measure Are avg  (V(a)-Aavg)*cos(360*time*Freq)                          ②
.measure Aim avg -(V(a)-Aavg)*sin(360*time*Freq)
.measure Bre avg  (V(b)-Bavg)*cos(360*time*Freq)                          ③
.measure Bim avg -(V(b)-Bavg)*sin(360*time*Freq)
.measure GainMag param 20*log10(hypot(Are,Aim) / hypot(Bre,Bim))          ④
.measure GainPhi param mod(atan2(Aim, Are) - atan2(Bim, Bre)+180,360)-180 ⑤
```

```
.step oct param Freq 20K 80K 3
.ic V(A)=2.4
.param t0=1m
.tran 0 {t0+5/Freq} {t0} 10n
```

① A点、B点の平均値を求める。（直流電圧に相当）

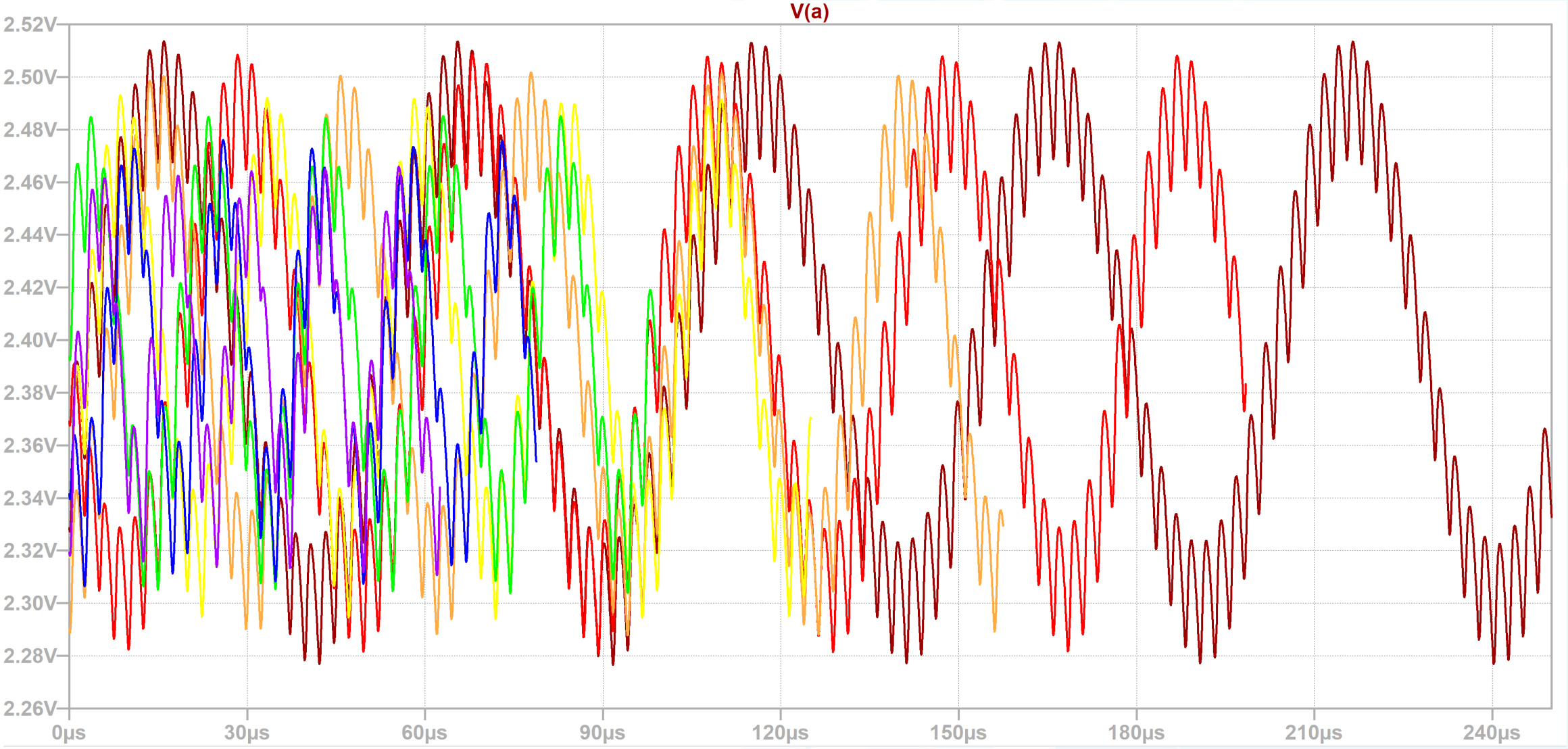
② A点のフーリエ係数のある特定周波数成分の実部と虚部を求める

③ B点でも同様に計算する

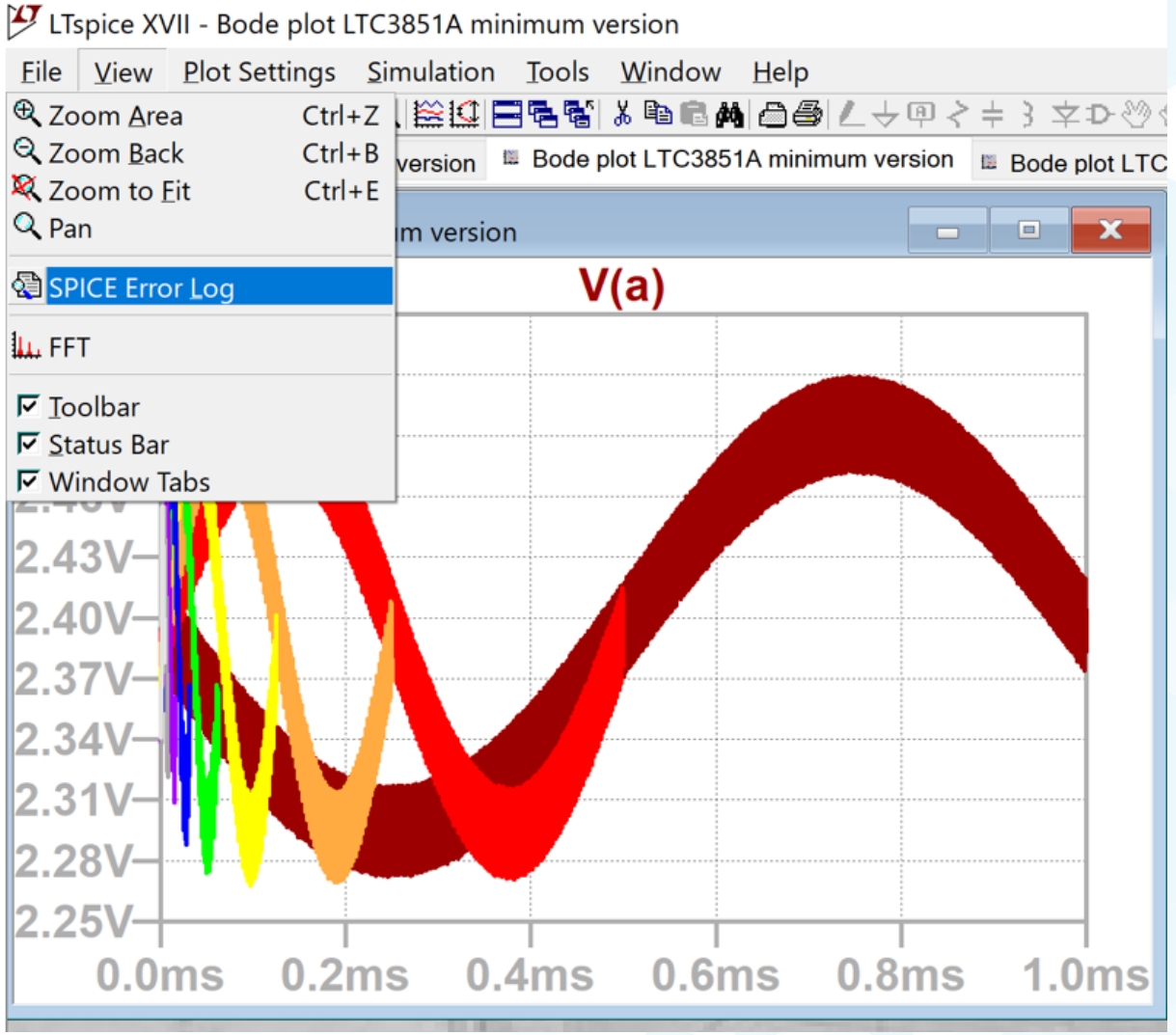
④ A点とB点の実部、虚部の二乗和平方根を求め、除算することによりB点からA点までのゲインを計算し、デシベルに変換している

⑤ 位相差も同様に計算している。

出力電圧波形



ボード線図を表示する方法



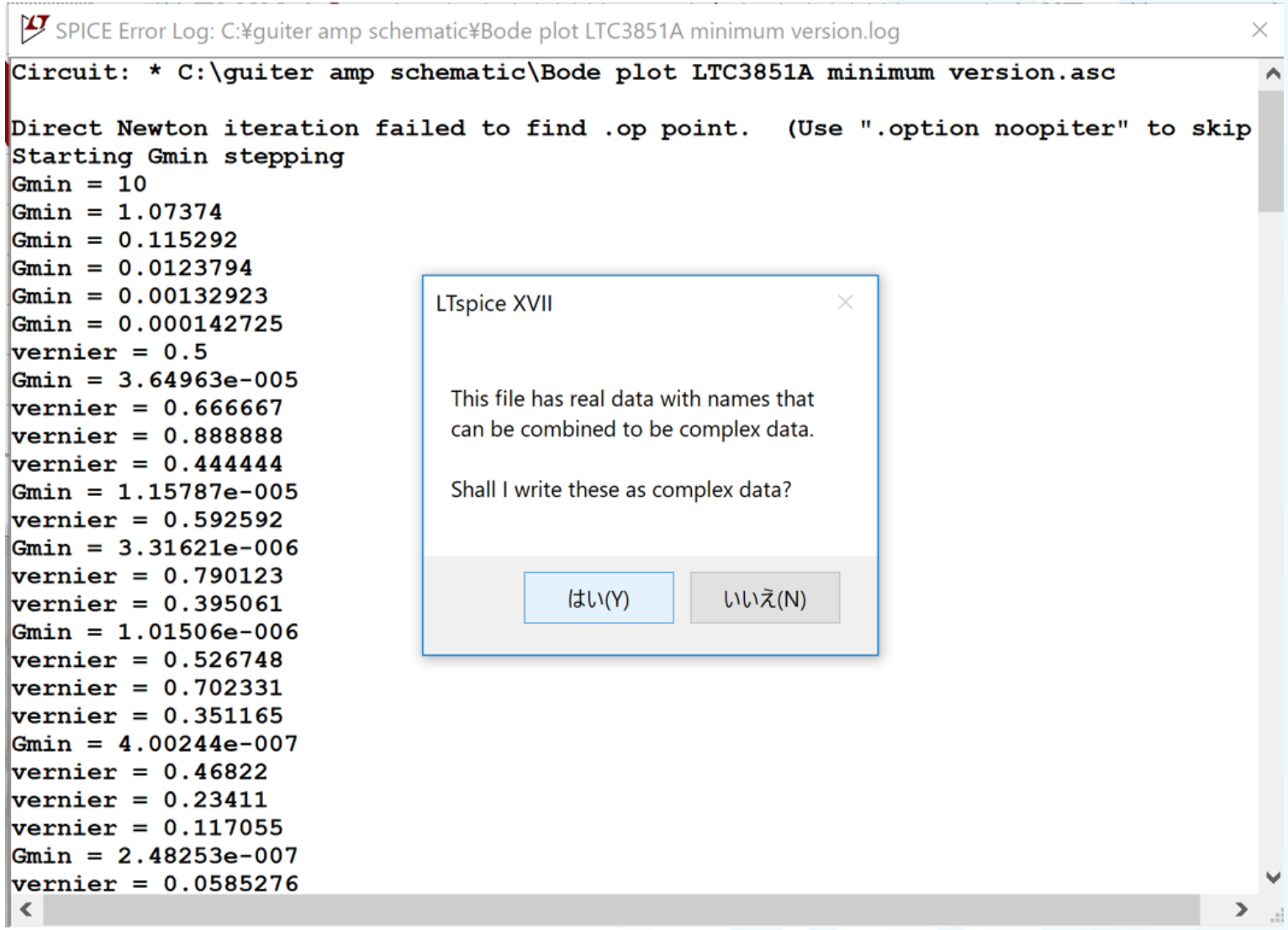
グラフペインを指定して VIEW➔SPICE ERROR LOG

SPICE Error Log: C:¥guiter amp schematic¥Bode plot LTC3851A minimum version.log

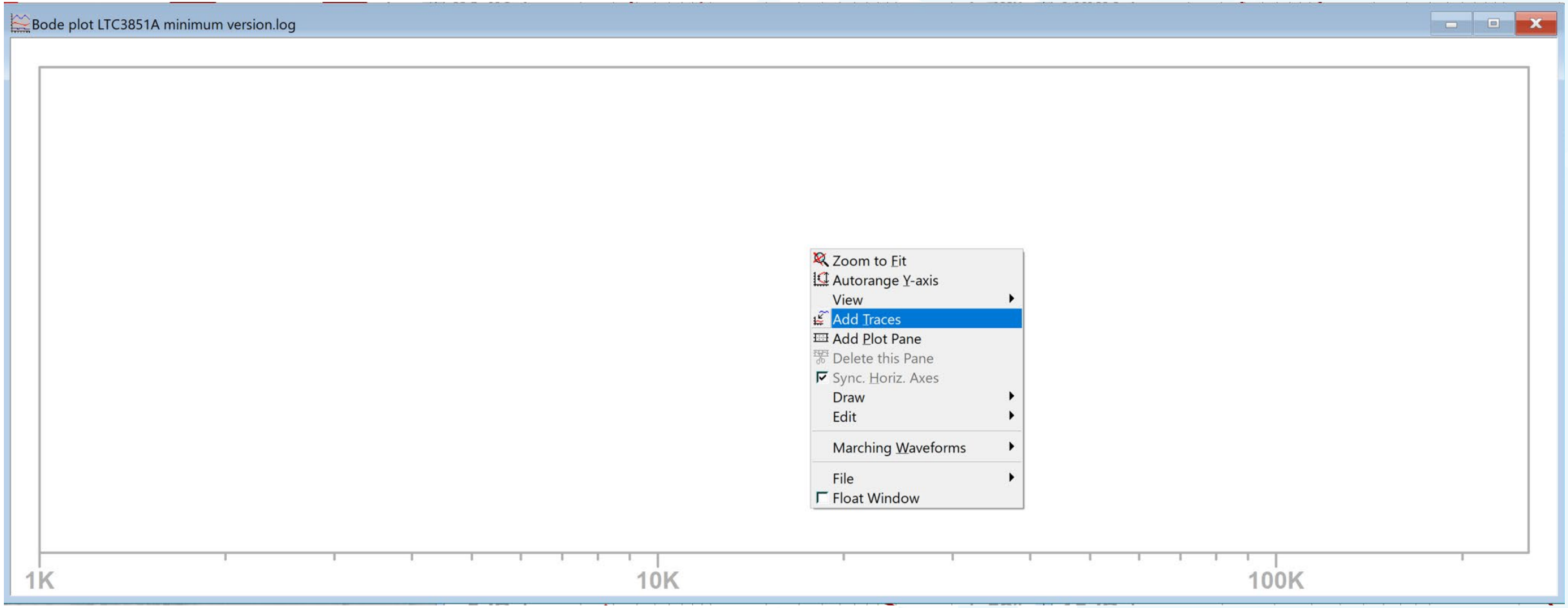
```
Circuit: * C:\guiter amp schematic\Bode plot LTC3851A minimum version.asc

Direct Newton iteration failed to find .op point.  (Use ".option noopiter" to skip
Starting Gmin stepping
Gmin = 10
Gmin = 1.07374
Gmin = 0.115292
Gmin = 0.0123794
Gmin = 0.00132923
Gmin = 0.000142725
vernier = 0.5
Gmin = 3.64963e-005
vernier = 0.666667
vernier = 0.888888
vernier = 0.444444
Gmin = 1.15787e-005
vernier = 0.592592
Gmin = 3.31621e-006
vernier = 0.790123
vernier = 0.395061
Gmin = 1.01506e-006
vernier = 0.526748
vernier = 0.702331
vernier = 0.351165
Gmin = 4.00244e-007
vernier = 0.46822
vernier = 0.23411
vernier = 0.117055
Gmin = 2.48253e-007
vernier = 0.0585276
```

Find Ctrl+F
Plot .step'ed .meas data
Close

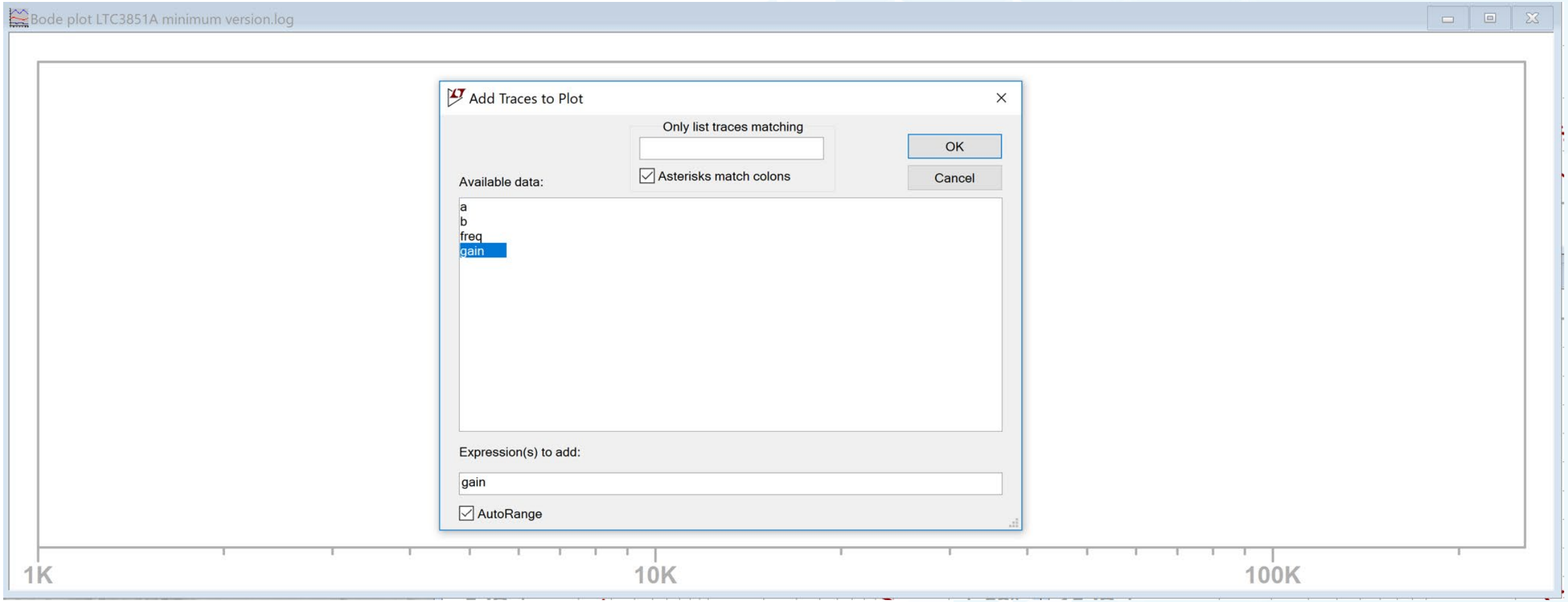
ERROR LOG上で右クリック、Plot step’ed .meas data



はいをクリック

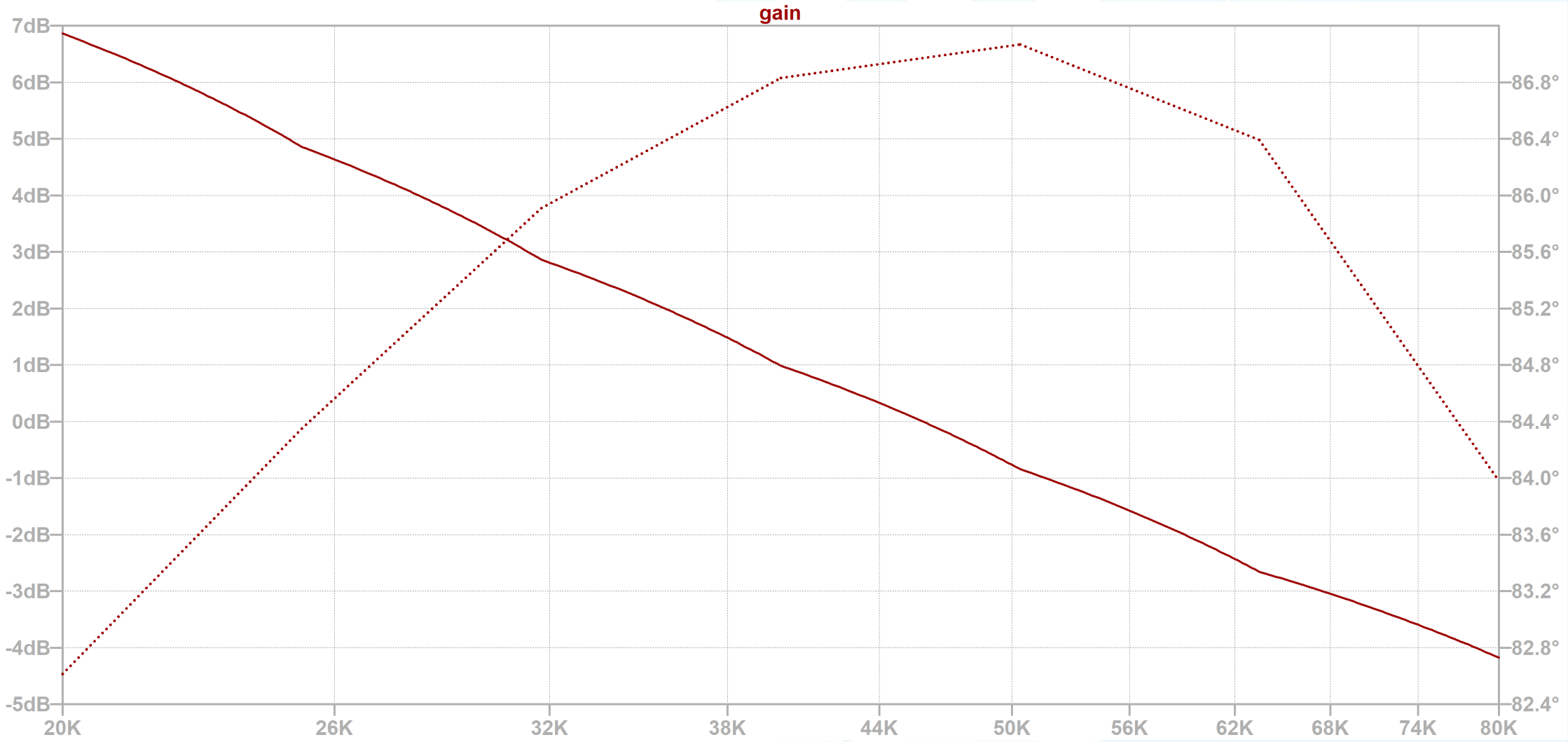


画面上で右クリックした後、Add Tracesをクリック

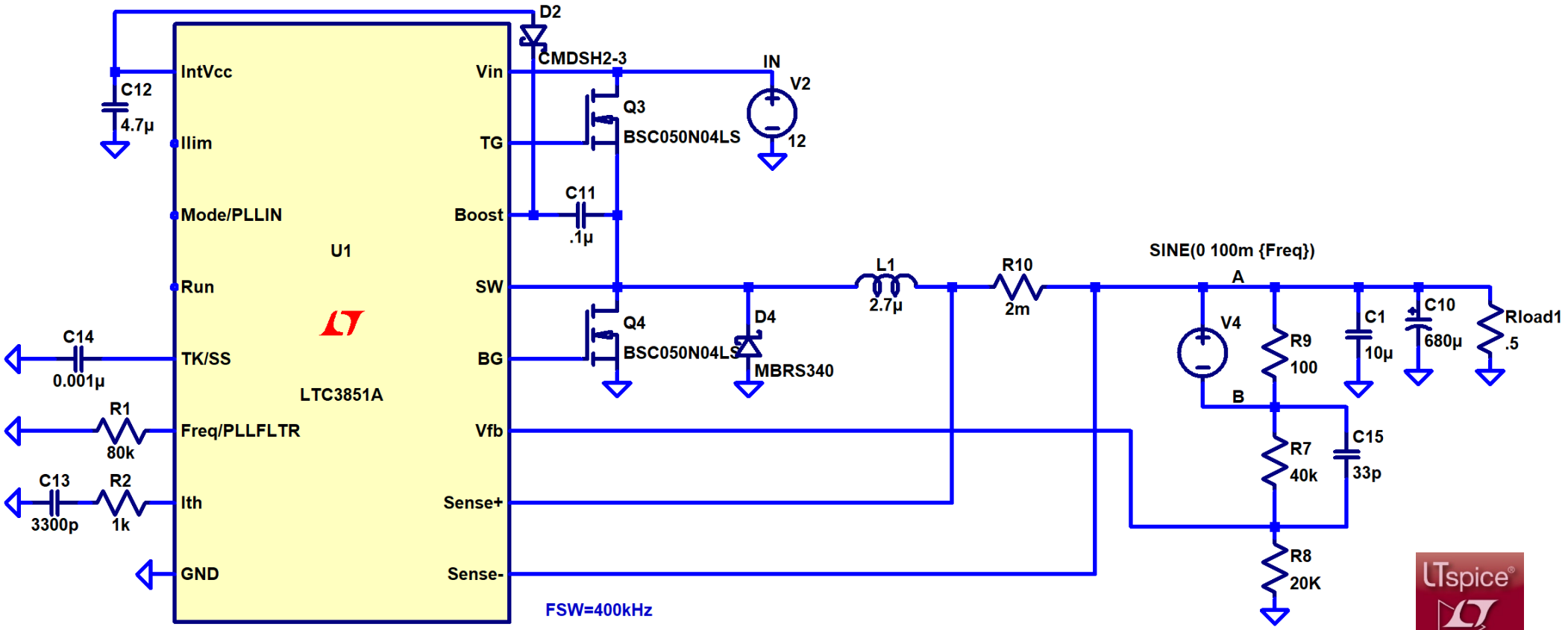


gain をクリック

シミュレーション結果



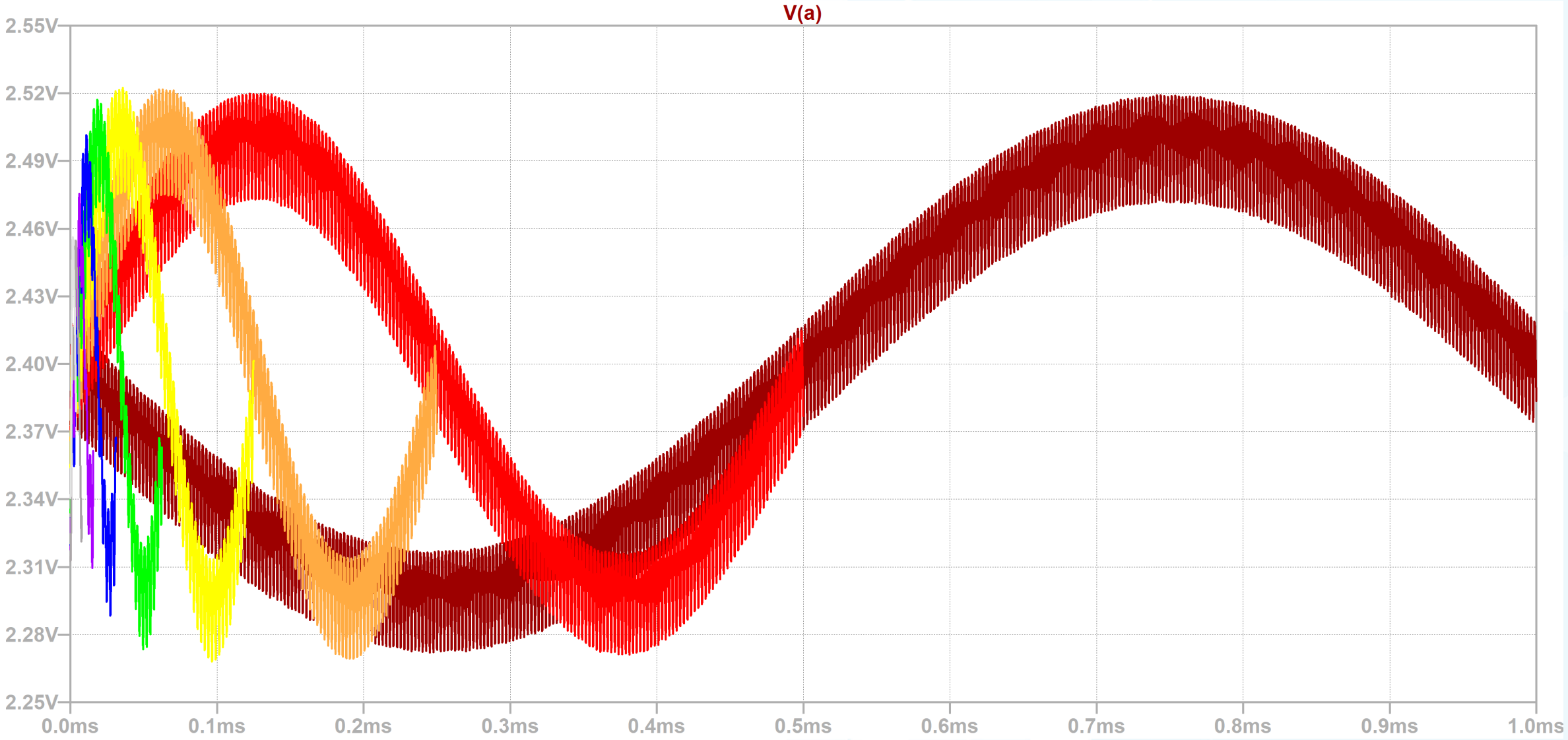
実際の進め方

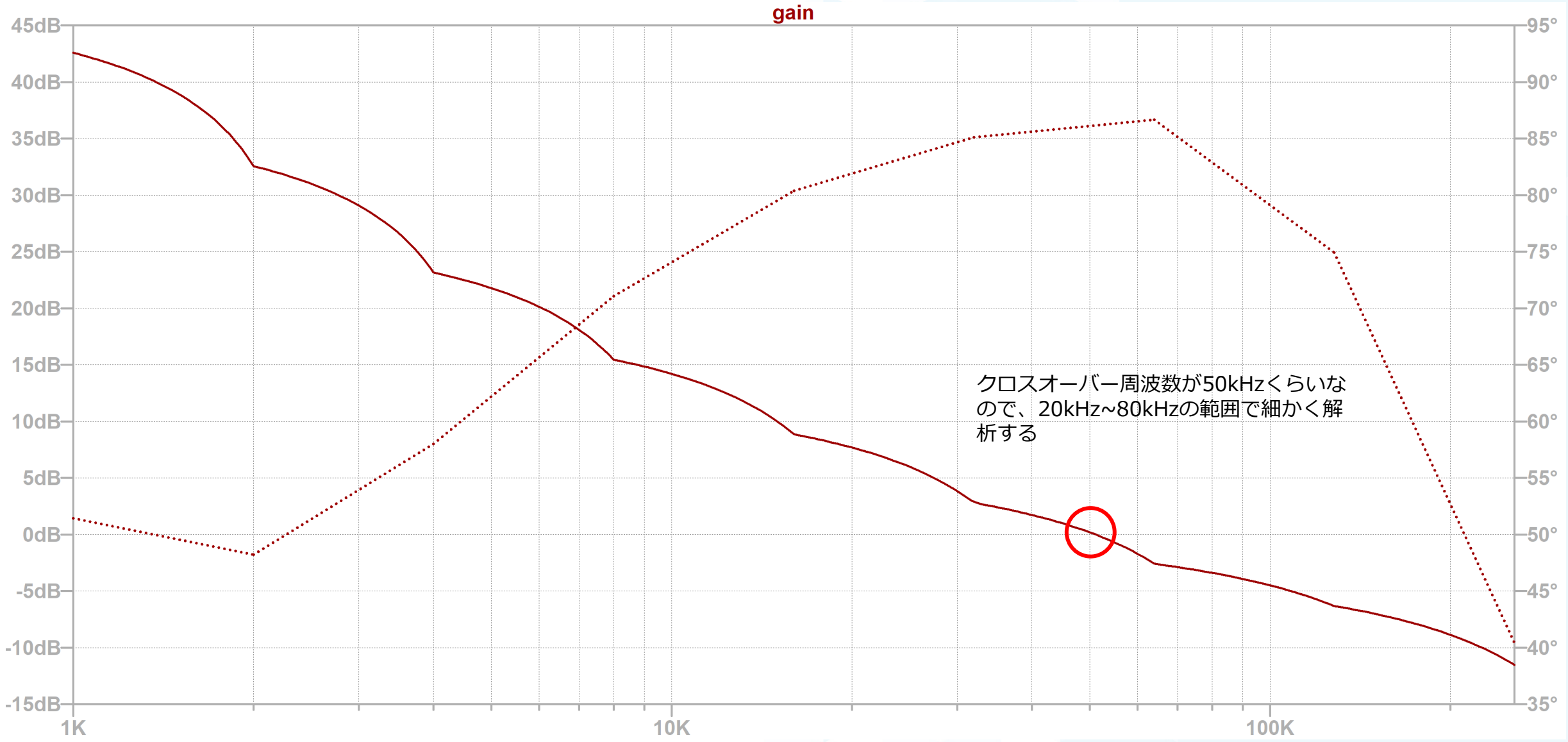



```
.measure Aavg avg V(a)
.measure Bavg avg V(b)
.measure Are avg  (V(a)-Aavg)*cos(360*time*Freq)
.measure Aim avg -(V(a)-Aavg)*sin(360*time*Freq)
.measure Bre avg  (V(b)-Bavg)*cos(360*time*Freq)
.measure Bim avg -(V(b)-Bavg)*sin(360*time*Freq)
.measure GainMag param 20*log10(hypot(Are,Aim) / hypot(Bre,Bim))
.measure GainPhi param mod(atan2(Aim, Are) - atan2(Bim, Bre)+180,360)-180
```

```
.step oct param Freq 1K 256K 1
.ic V(A)=2.4
.param t0=.5m
.tran 0 {t0+1/Freq} {t0} 10n
```

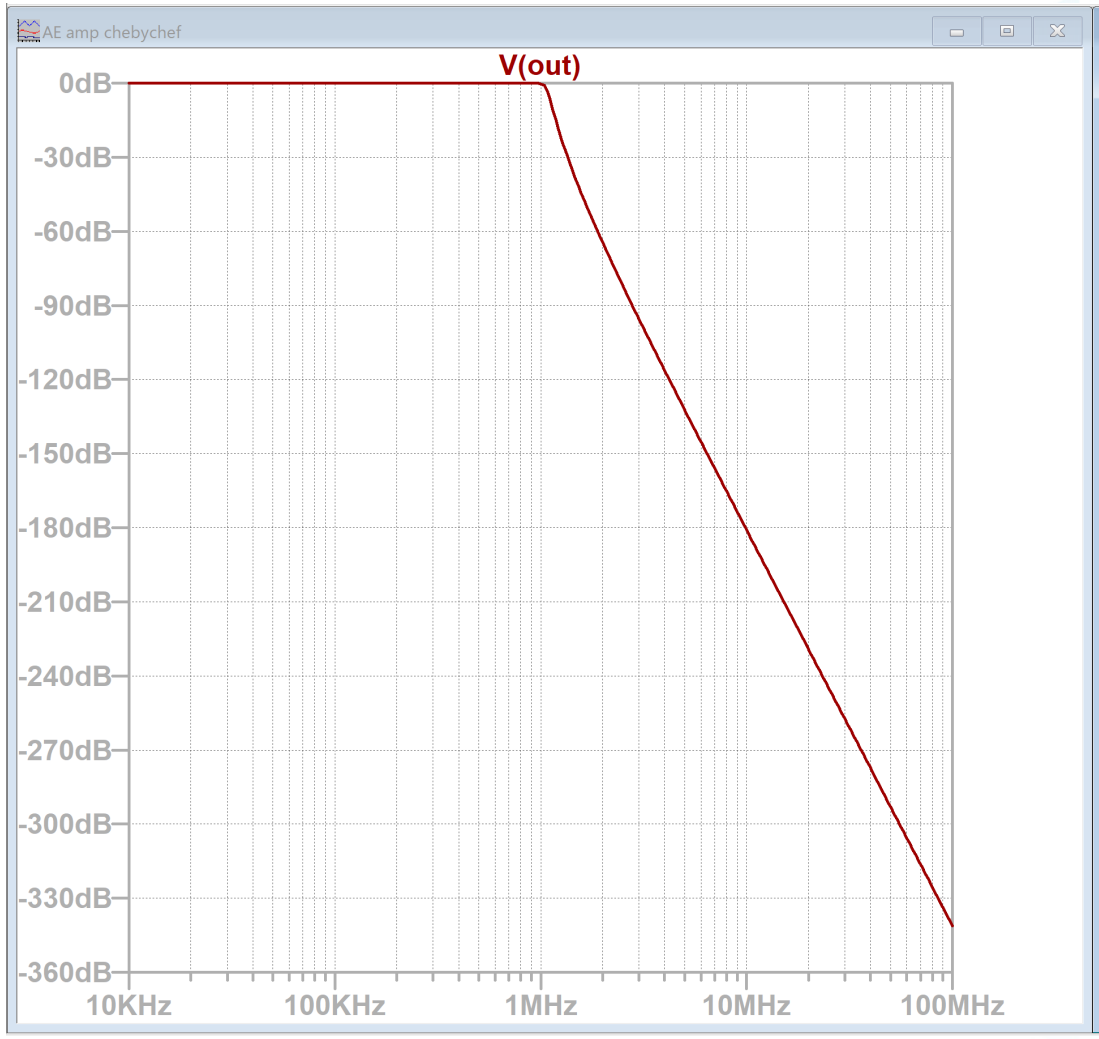
周波数範囲は広く、オクターブ中の解析ポイント数は少なく（この例では１）平均化のための波形数を少なく（この場合は1波形）すると全体が短時間で見通せます。ただし、低い周波数から始めるとシミュレーションに時間がかかる

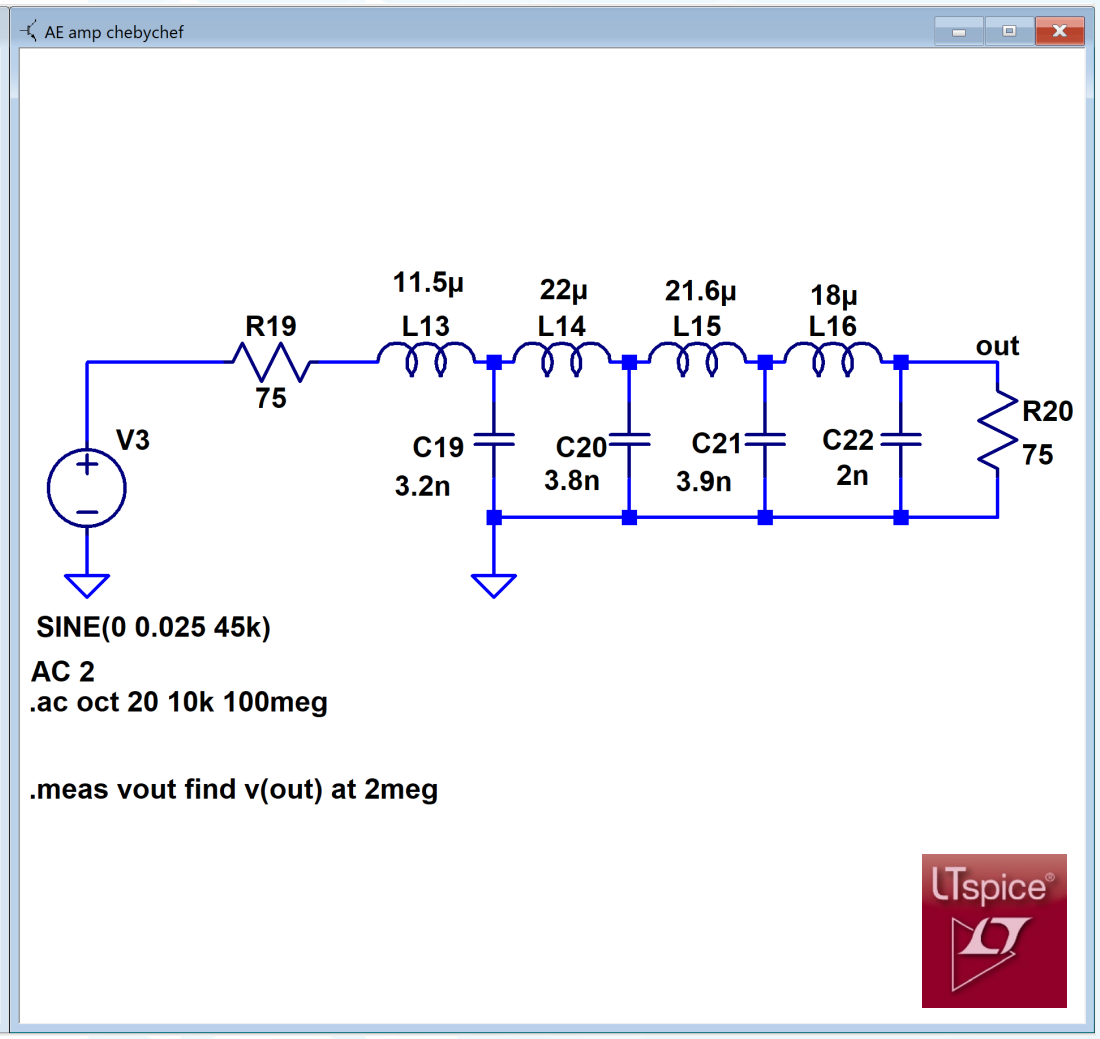




電子回路の事前検証
（フィルタを例にとって）

カットオフ周波数 1MHz のチェビシェフフィルタの例を示す。





抵抗値、コンデンサ容量、インダクタンス値が正確ならこのような特性が得られる。
しかし、通常使用する部品にはばらつきがある。
このため最悪の場合どのような特性になるのかをあらかじめ予測しておく必要がある。

抵抗は1%精度のものは容易に入手できる。
コンデンサは10%、インダクタは20%精度のものが一般的である。

これらがばらついた時にフィルタの特性がどう変化するかを調べてみる

フィルタの周波数特性のバラつきを調べる

LTspiceでは、パラメータの値を可変するために以下のようないくつかの方法を提供してる。

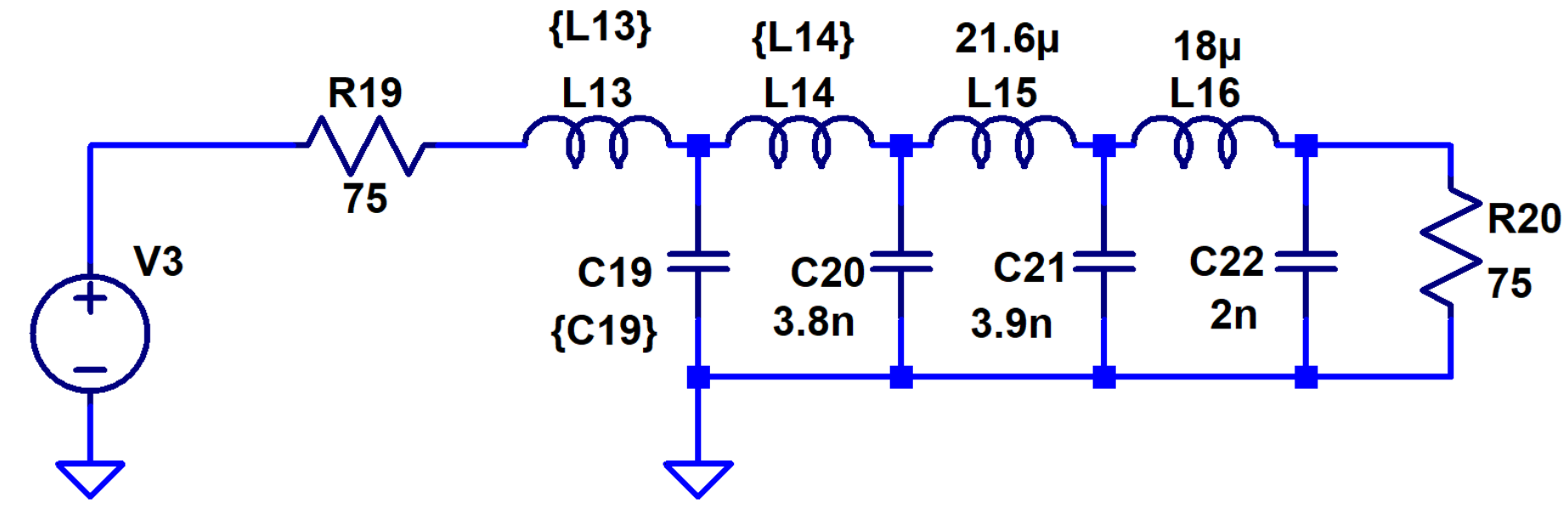
- .step param　　：ユーザ定義変数のパラメータ・スイープ
- mc(x,y)　　：一様分布のx*(1+y)からx*(1-y)間の乱数（モンテカルロ法）
- gauss(x)　　：標準偏差xの正規分布からの乱数

これらの内、関数 Gauss(x)およびmc(x,y)は、分布の観点で結果を見たい場合には非常に有用である。

しかし、最悪条件を見たい場合には、統計的に有意な回数※だけシミュレーションを実行し、分布を見て、標準偏差から最悪値を推定する必要がある。
これは、最悪値の結果を得るだけであれば、最速の方法ではない。

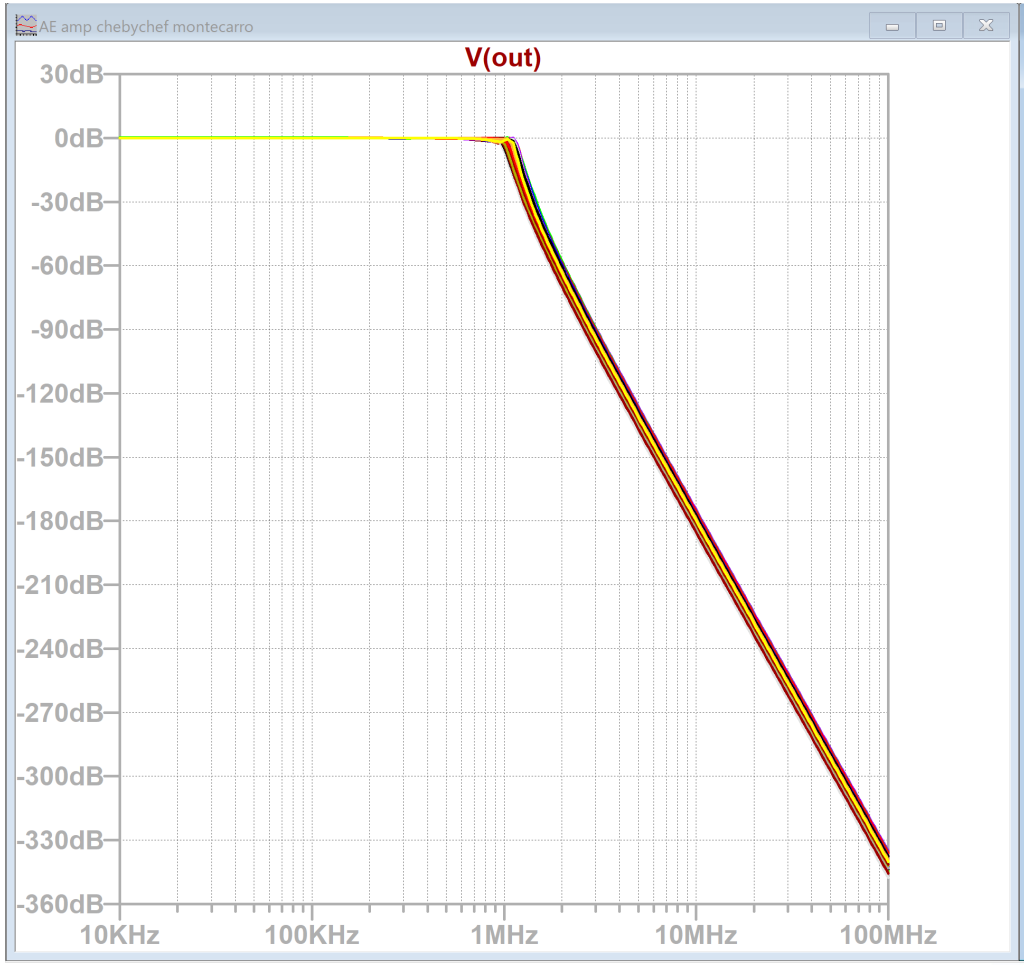
※　許容誤差1%で約10000回、10%でも約100回のサンプルが必要。

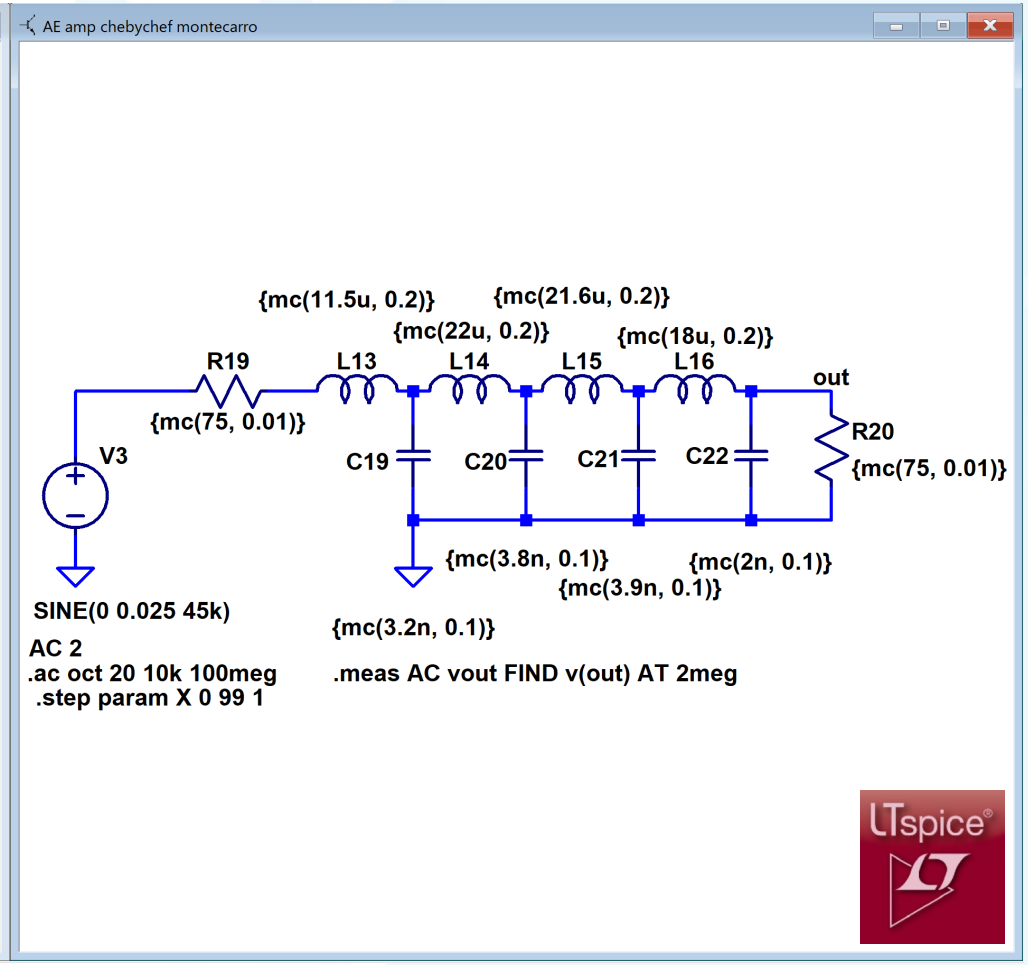
.stepコマンドでは3階層までのネストしか許されていないため部品が多い回路では制約が生じる



モンテカルロ法でLPFの特性悪化を検証する

抵抗、コンデンサ、インダクタの値がばらついた場合（一様分布）に
どのくらいの影響があるかを調べる。





バラつきが正規分布に従うと仮定する場合にはgauss関数を使用する

ガウス関数を使用するには標準偏差を入力しなければならない

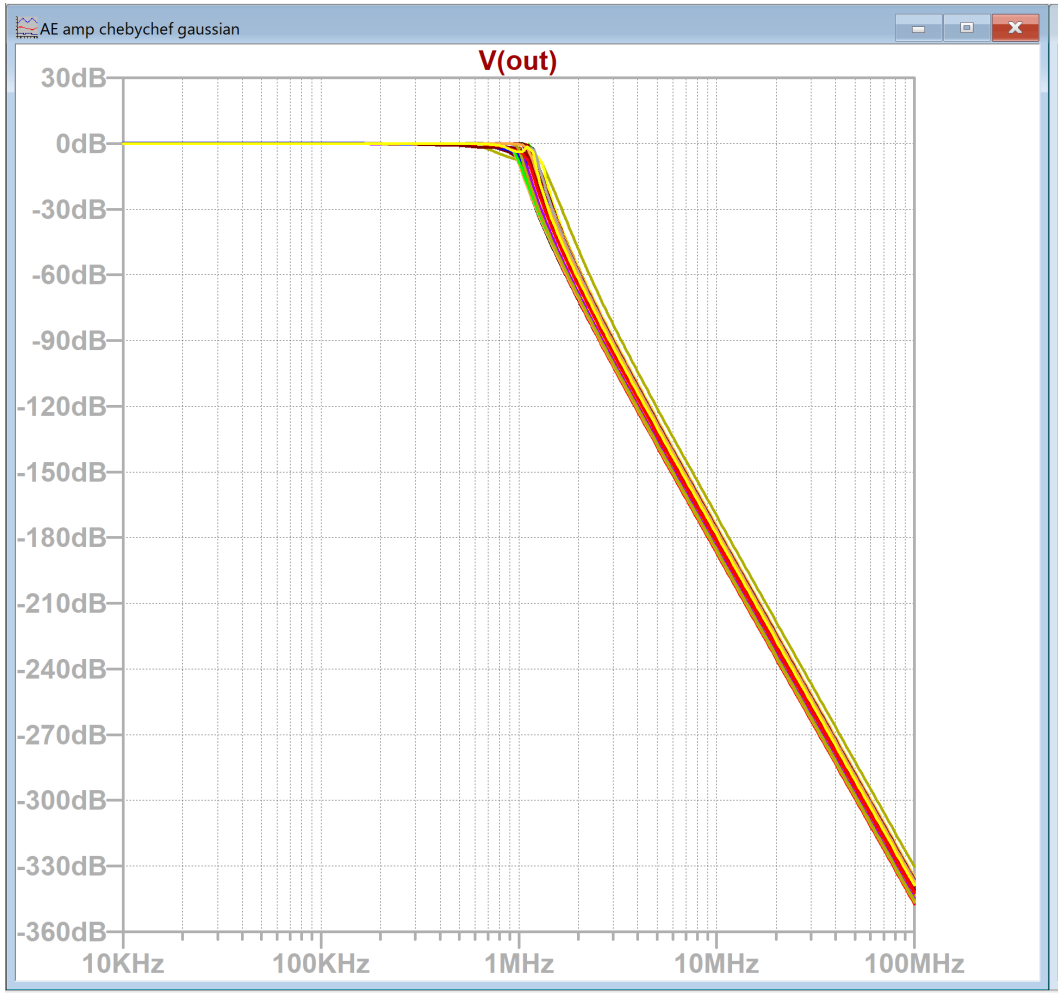
{11.5u+gauss(1.958u)}

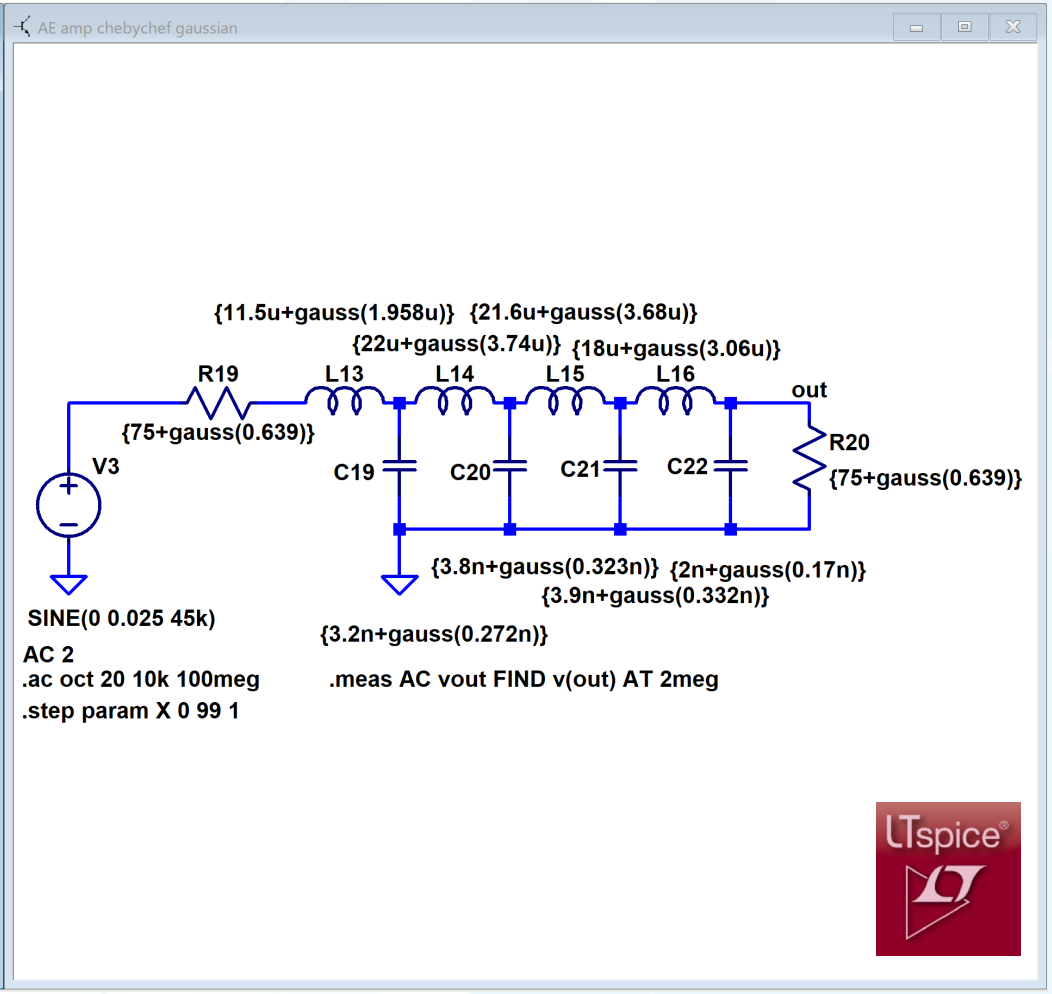
11.5uは中心値、1.958uが標準偏差σ。ここで標準偏差 σ は11.5uHの±20%に相当する4.6uHをガウス関数の半値幅（半値全幅）と定義して求めている。半値全幅とσの関係は

σ=半値全幅/2.35で求めることができる。

また統計的にはインダクタンスが11.5uH±1.958uHに入る確率が約68%であることを示している。

ガウス関数によるシミュレーション





ひとつひとつの部品バラつきを考慮するには

使用するインダクタ、コンデンサのひとつひとつのバラつきの最大値、最小値を使用して
フィルタのシミュレーションを行う方法を考える。

インダクタは+20%の時、-20%の時を、コンデンサは同様に+10%と-10%の場合を考える。
素子の数は合計8個なので組み合わせは２の8乗すなわち256通りが考えらえる。
最後にすべてが公称値の時の特性もシミュレーションする。

このような作業を自動化するためにユーザ定義関数　.func　を使用する。

.func コマンドとは

.func　　ユーザー定義関数

構文　　.func <名前>（[引数]）{<式>}

例　　.func Pythagoras(x,y）{sqrt（x*x + y*y）}

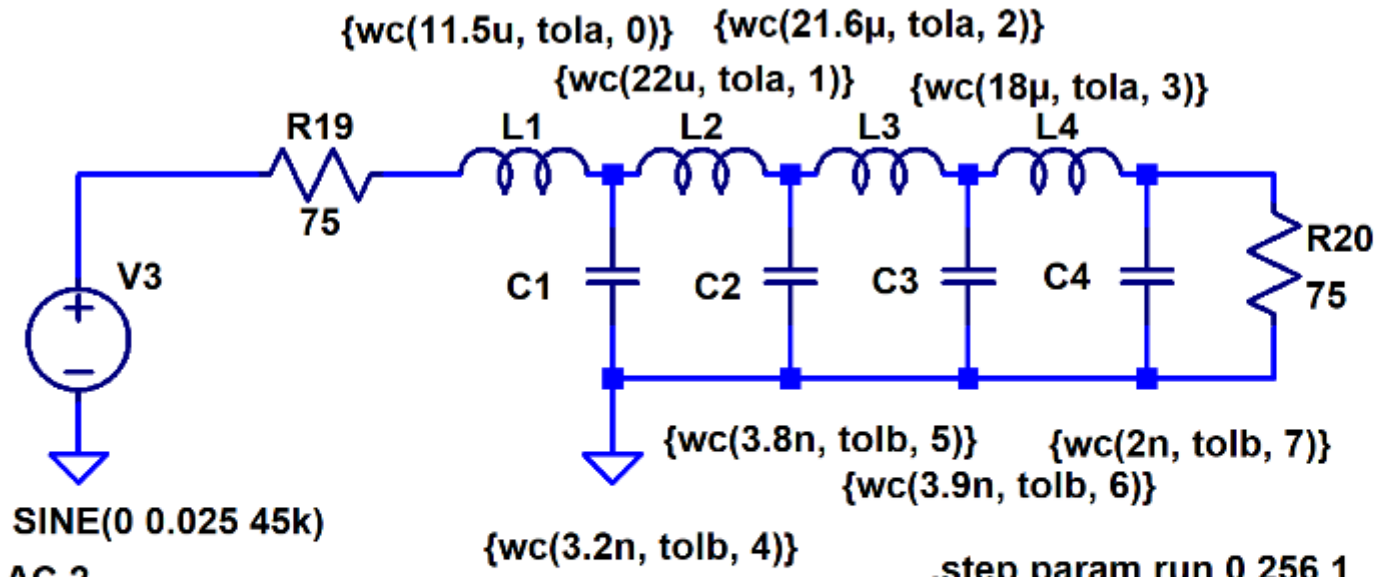
Pythagorasで定義された関数は二つの数 x , y の平方和の平方根を求めるもの

x=300, y=400 の場合、Pythagoras(x,y)は 500 となる

すべてのパラメータ置換評価はシミュレーションが始まる前に行われる

.funcコマンドでLPFの特性悪化を検証する

インダクタ、コンデンサの値が関数wc(nominalvalue, tolerance, index)で置き換えられており、.funcコマンドによって定義されている




```
.func wc(nom,tol,index) if(run==numruns,nom,if(binary(run,index),nom*(1+tol),nom*(1-tol)))
.func binary(run,index) floor(run/(2**index))-2*floor(run/(2**(index+1)))
```

さあどうする

各素子に番号（Index）を付ける
作業番号（run）に対して
右の表に従いそれぞれに0か1を当てる
0が-20%、1が+20%（インダクタ）
0が-10%、1が+10%（コンデンサ）

よく見るとIndex=0がLSB
Index=7がMSBでrunが10進数を表す
8ビットの数値テーブルになっている
ことがわかる

	run	0	1	2	3	4
index	0 (L1)	0	1	0	1	0
	1 (L2)	0	0	1	1	0
	2 (L3)	0	0	0	0	1
	3 (L4)	0	0	0	0	0
	4 (C1)	0	0	0	0	0
	5 (C2)	0	0	0	0	0
	6 (C3)	0	0	0	0	0
	7 (C4)	0	0	0	0	0

これを　.func　で記述すれば

.func binary(run,index) floor(run/(2**index))-2*floor(run/(2**(index+1)))

例１　　Run=4、Index=2の場合（5回目のL3の値）

- binary(4,2)=floor(4/2^2)-2*floor(4/2^3)=floor(4/4)-2*floor(4/8)=1-0=1

例２　　Run=249、Index=1の場合（250回目のL2の値）

- binary(249,1)=floor(249/2^1)-2*floor(249/2^2)=floor(249/2)-2*floor(249/4)=124-2*62=0

	run	0	1	2	3	4	----	245	246	247	248	249	250	----	----
index	0 (L1)	0	1	0	1	0		1	0	1	0	1	0		
	1 (L2)	0	0	1	1	0		0	1	1	0	0	1		
	2 (L3)	0	0	0	0	1		1	1	1	0	0	0		
	3 (L4)	0	0	0	0	0		0	0	0	1	1	1		
	4 (C1)	0	0	0	0	0		1	1	1	1	1	1		
	5 (C2)	0	0	0	0	0		1	1	1	1	1	1		
	6 (C3)	0	0	0	0	0		1	1	1	1	1	1		
	7 (C4)	0	0	0	0	0		1	1	1	1	1	1		

次に　.func wc(nom,tol,index)　について

関数　.func wc(nom,tol,index) は　binary関数によって得られた各素子のrun（作業番号）の値に対する数値（0または1）から各素子の最小値と最大値を返す。

.func wc(nom,tol,index)
if(run==numruns,nom,if(binary(run,index),nom*(1+tol),nom*(1-tol)))

nomはそれぞれの素子の公称値
tolは誤差（許容値）で0.2（インダクタの場合）か0.1（コンデンサの場合）を取る
indexは素子番号

この関数は、binary(run、index)関数とともに、各部品の最大値と最小値を与える

まとめると

例１　　Run=4、Index=2の場合（5回目のL3の値）

- binary(4,2)=floor(4/2^2)-2*floor(4/2^3)=floor(4/4)-2*floor(4/8)=1-0=1
- wc(21.6u, 0.2, 2)=21.6u*(1+0.2)=21.6u*1.2=25.92uH

例２　　Run=249、Index=1の場合（250回目のL2の値）

- binary(249,1)=floor(249/2^1)-2*floor(249/2^2)=floor(249/2)-2*floor(249/4)=124-124=0
- wc(22u, 0.2, 1)=22u*(1-0.2)=22u*0.8=17.6u

	run	0	1	2	3	4	----	245	246	247	248	249	250	----	----
index	0 (L1)	0	1	0	1	0		1	0	1	0	1	0		
	1 (L2)	0	0	1	1	0		0	1	1	0	0	1		
	2 (L3)	0	0	0	0	1		1	1	1	0	0	0		
	3 (L4)	0	0	0	0	0		0	0	0	1	1	1		
	4 (C1)	0	0	0	0	0		1	1	1	1	1	1		
	5 (C2)	0	0	0	0	0		1	1	1	1	1	1		
	6 (C3)	0	0	0	0	0		1	1	1	1	1	1		
	7 (C4)	0	0	0	0	0		1	1	1	1	1	1		

Runの回数は2^N+1としている。ここで、Nは素子数8で、各素子の最大および最小すべての組み合わせでシミュレーションを実行し、最後に公称値で実行する
本例の場合、256+1回のシミュレーションが必要で、.step命令と.param命令でそれを定義している。

```
.step param run 0 256 1    run=0から256まで1ステップ
.param numruns=256         これは定数の設定
```

最後に、シミュレーション用に.param命令でtolaとtolbを定義している

```
.param tola=0.2    インダクタの場合の公差20%
.param tolb=0.1    コンデンサは10%
```

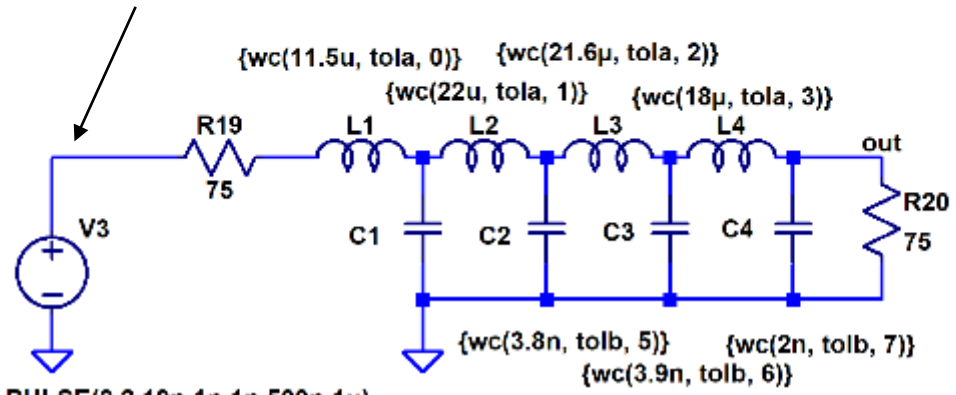
.funcコマンドでLPFの特性悪化を検証する（時間軸）

0:24

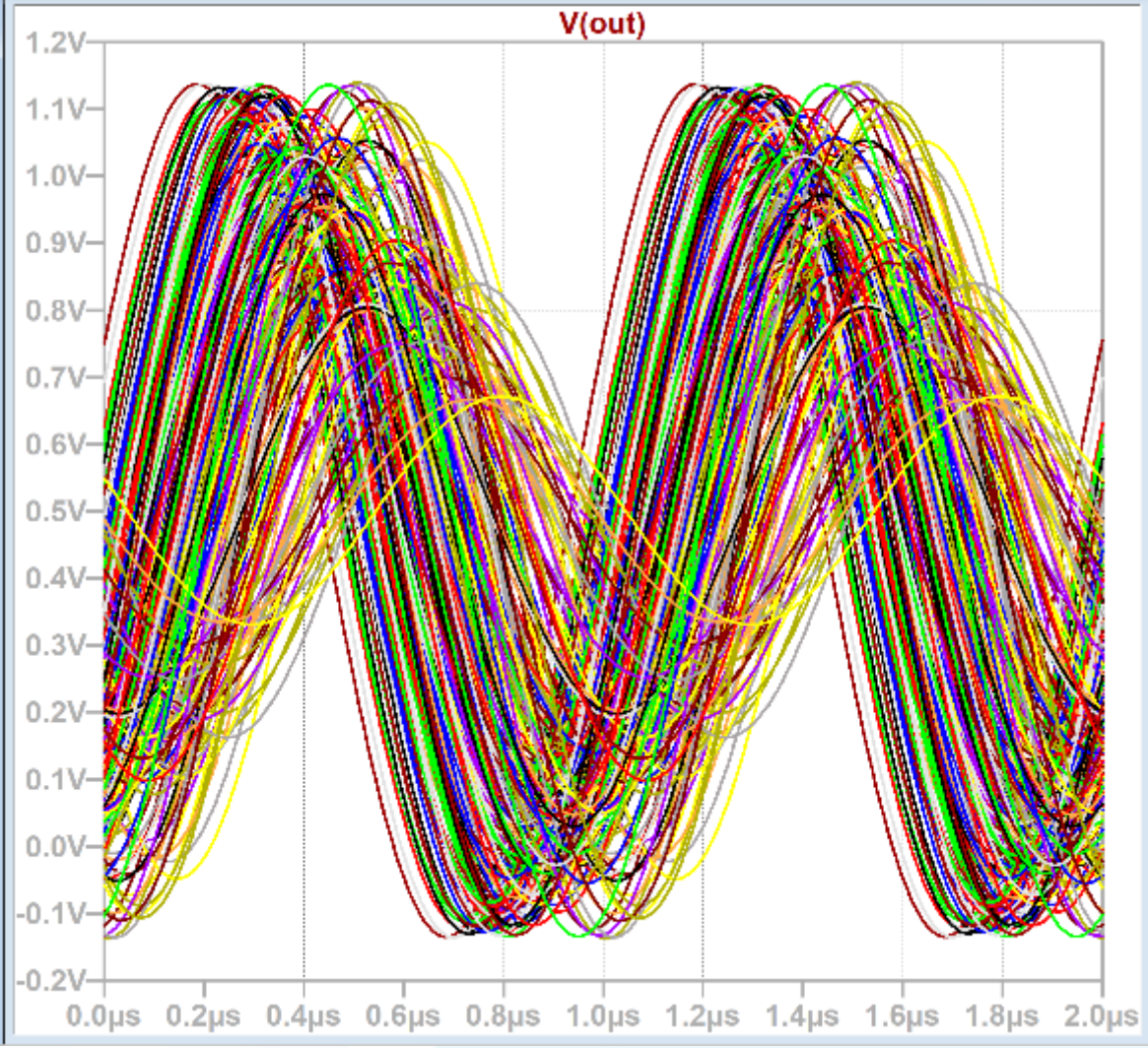
AE amp chebychef .func command time domain.

AE amp chebychef .func command time domain

1MHzの矩形波を入力

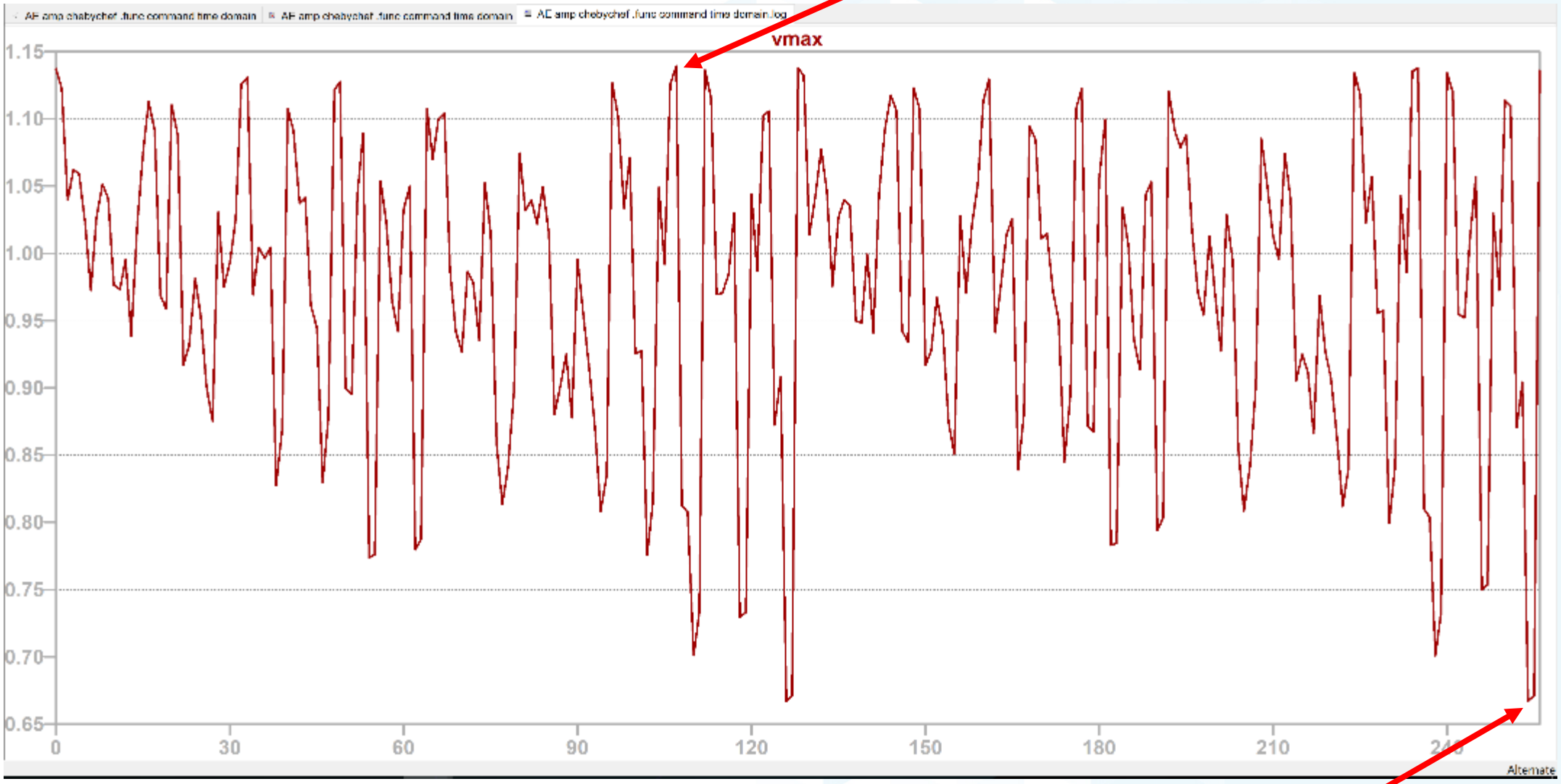



```
PULSE(0 2 10n 1n 1n 500n 1u)

.tran 0 20u 18u

.step param run 0 256 1
.param numruns=256
.param tola=0.2
.param tolb=0.1

.func wc(nom,tol,index) if(run==numruns,nom,if(binary(run,index),nom*(1+tol),nom*(1-tol)))
.func binary(run,index) floor(run/(2**index))-2*floor(run/(2**(index+1)))

.meas Vmax MAX V(out)
```



ピーク値を求める

RUN=107,
peak=1.1394



RUN=254, peak=0.667

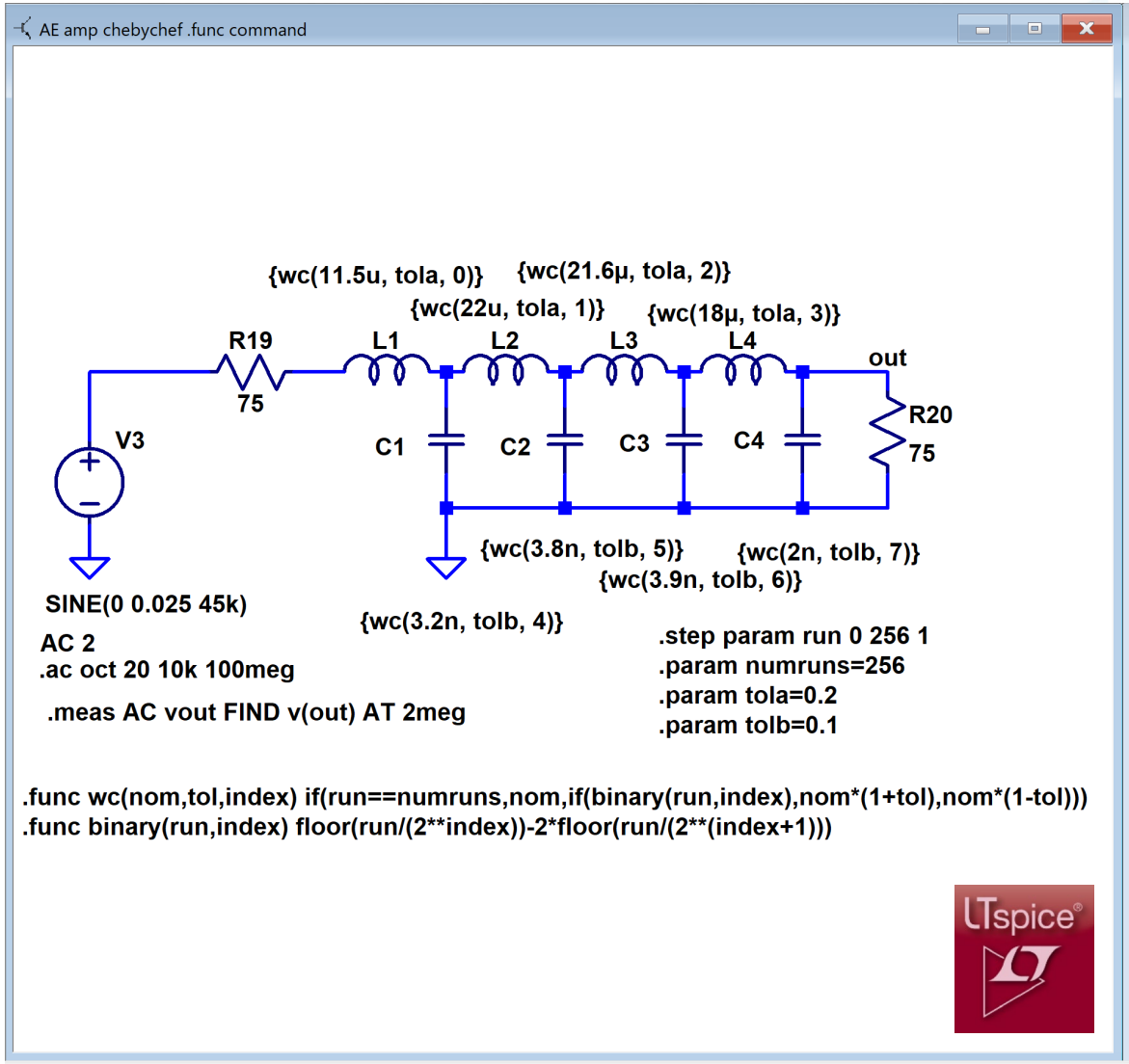
振幅最大、最小の時の定数を求める

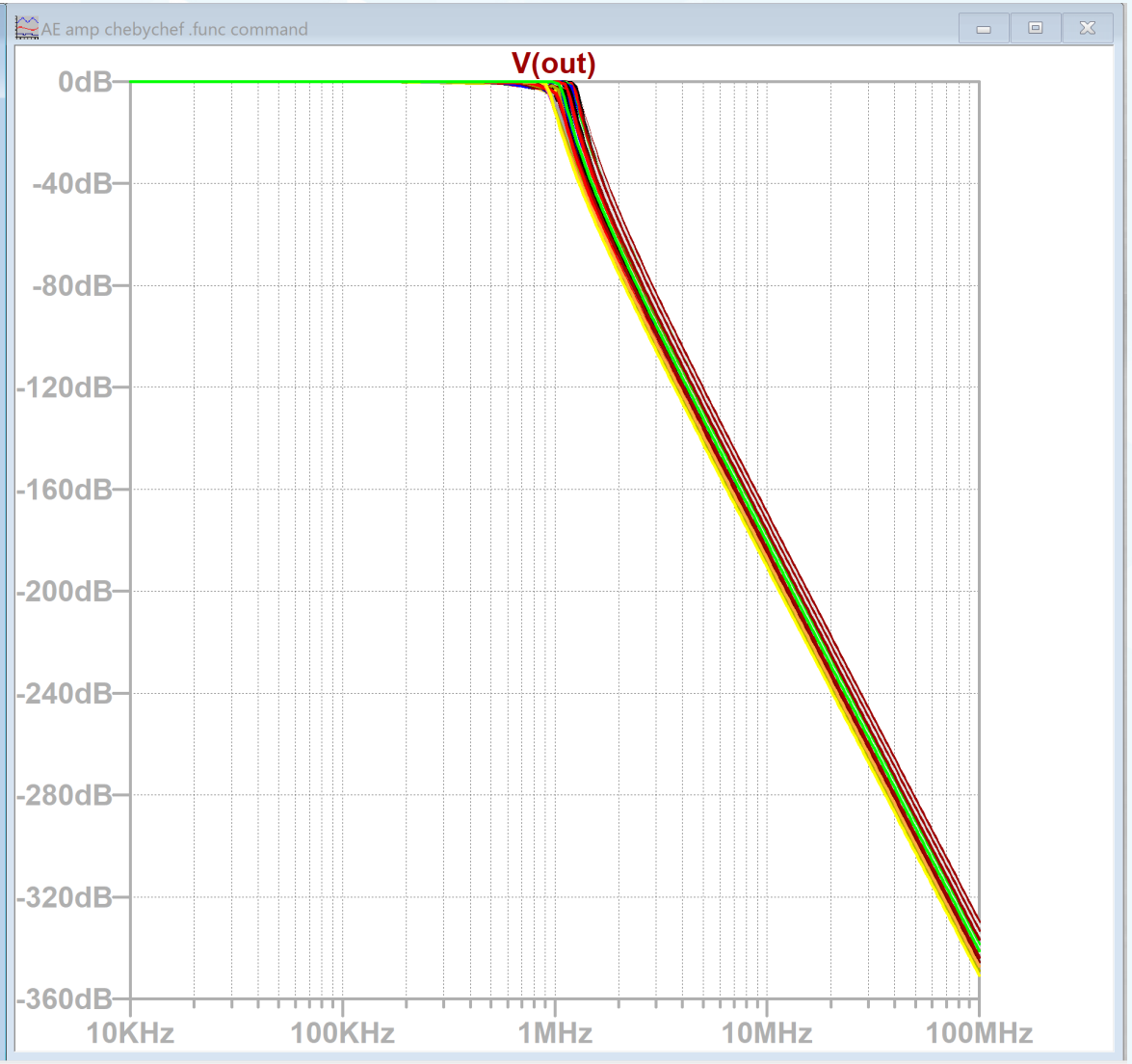
RUN=107、254のインデックス0から7に対応する
バイナリ値を右に示す。

振幅最大の場合	振幅最小の場合
L1=13.8uH	L1=9.2uH
L2=26.4uH	L2=26.4uH
L3=17.28uH	L3=25.92uH
L4=21.6uH	L4=21.6uH
C1=2.88nF	C1=2.816nF
C2=4.18nF	C2=4.18nF
C3=4.29nF	C3=4.29nF
C4=1.8nF	C4=2.2nF

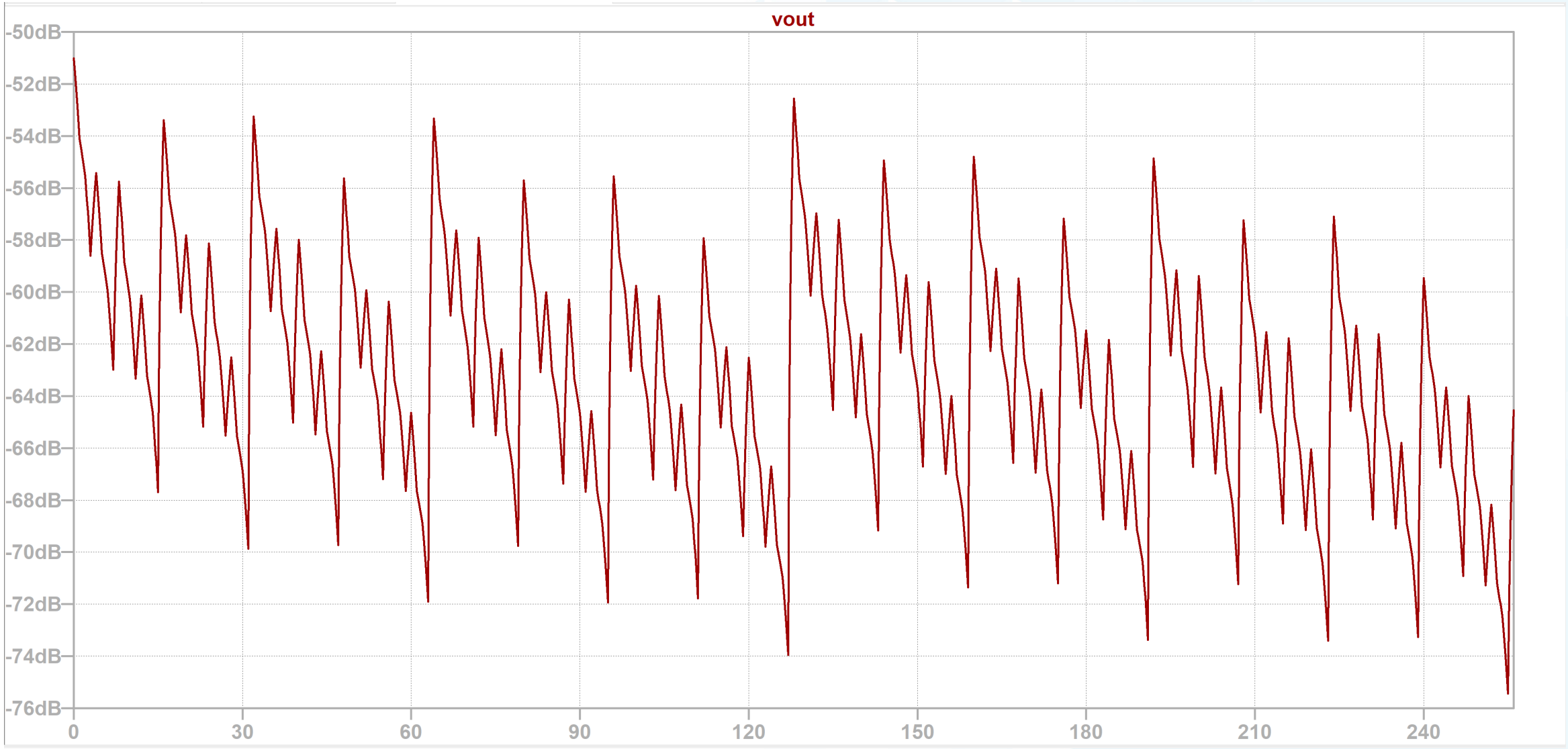
	run	107	254
index	0 (L1)	1	0
	1 (L2)	1	1
	2 (L3)	0	1
	3 (L4)	1	1
	4 (C1)	0	1
	5 (C2)	1	1
	6 (C3)	1	1
	7 (C4)	0	1

.funcコマンドでLPFの特性悪化を検証する（周波数軸）





2MHzにおける減衰特性を調べる



本日のまとめ

1. 電源回路に関して

   位相余裕の測定方法

   測定装置がない場合の対処方法

   LTspiceによるボード線図の作成方法

2. 電子回路の事前検証（高次フィルタの例）

   受動部品の誤差がどの程度周波数特性に影響するか

   LTspiceでの検証方法（.step、モンテカルロ、ガウス関数）

   最悪の場合を .func コマンドを使用して検証