複数の部品の誤差を考慮したシミュレーションによる評価

部品の誤差を考慮した設計を、シミュレーションによって確認する場合、「.STEP」コマンドを利用し、その部品のパラメータの最小値・最大値を代入する手法が一般的である。しかし、「.STEP」は3つまでしかネストできないので、自由度が少ない。もちろん、「TABLE」機能を併用することで、パラメータの組み合わせを一覧表のように記述することで3つを超えるパラメータを取り扱うことも可能である。

しかし、もしn個の部品の「Min-Max」すべての組み合わせとなると、2^n通りのシミュレーションが必要になる。これを「TABLE」で記述するのは現実的ではない。このような場合に便利な手法を以下に紹介する。

.FUNC の基本文法

.FUNC を利用すると、様々な条件を「引数」にして関数として定義することで、部品（要素）のパラメータを条件に応じた値に設定できる。

.func xxxx(zzz,・・・) [・ ・ ・ ・ ・ ・]

計算式を記述する。 .PARAM で利用できる数学関数と IF文 が利用できる。

関数で利用する「引数」を、丸括弧（・・）の中に「カンマ(,)区切り」で並べる。
関数で使う必要数だけパラメータを定義する。計算実行時点で値は確定していなければならない（シミュレーションの実行中にダイナミックには変化させられない）。

ユーザ定義関数の名称（ユーザが任意に決める・・・予約変数になっていないこと）。
この関数名で、回路図中で利用する。

ユーザ定義関数の宣言。

・・・事前準備（１）

Monte Carlo(MC) による一様乱数を使ったシミュレーションでは、Min-Max の限界までパラメータが広がる確率が少ない。 そこで、Min-Max の値だけを各部品に順次割り当て、すべての組み合わせをシミュレーションする。

要素の数：n ・・・とすると、組合せの数は 2^n になる。
n個の部品に番号を付ける（0から始める）

シミュレーションを 2^n （=N）回実行する。 （下に示す関数定義では、変数名を「run」としている）
すなわち 2^n を表現する「2進数」の各桁の係数（0または1）を
シミュレーションの「STEP」ごとに割り当てる。

これらの「N」と「n」から各bit（部品ごと）の係数（0または1）を求める関数を定義する。

$$N = b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_1 2^1 + b_0 2^0 = \sum_{i=0}^{n} b_i 2^i$$

左のように書いたときの各係数 b_n は→
（詳細は末尾の付録を参照）

$$b_n = \left[\frac{N}{2^n}\right] - \left[\frac{N}{2^n}\cdot\frac{1}{2}\right]\times 2 = \left[\frac{N}{2^n}\right] - 2\times\left[\frac{N}{2^{n+1}}\right]$$

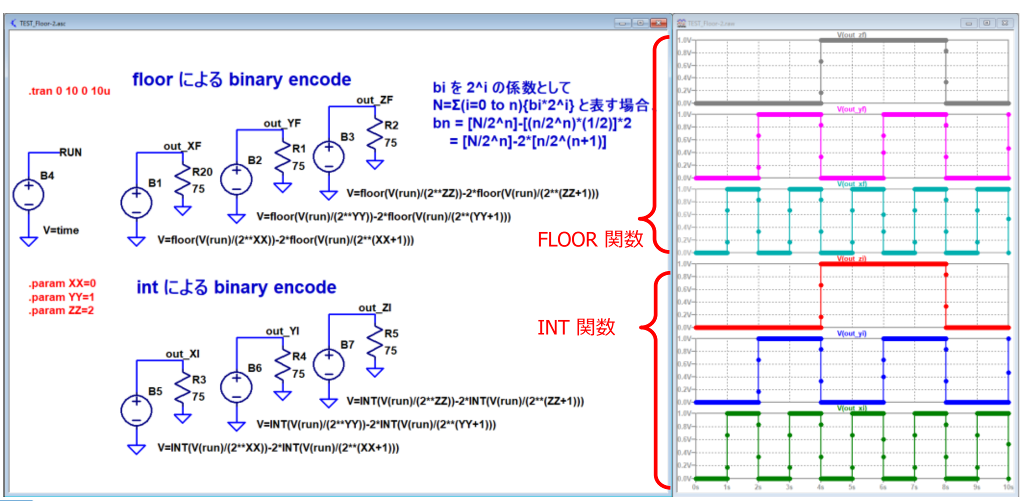
.func binary(run,index) $\underbrace{\text{INT(run/2**index)-2*INT(run/2**(index+1))}}_{b_n \text{ の式}}$

関数名

引数・・・run：「.STEP」の実行中の回数、index：部品番号（0から付ける）

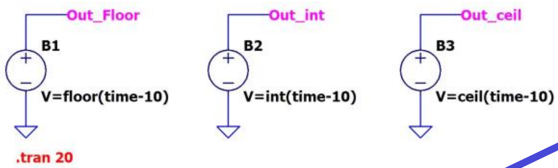
・・・事前準備（補足説明-1）

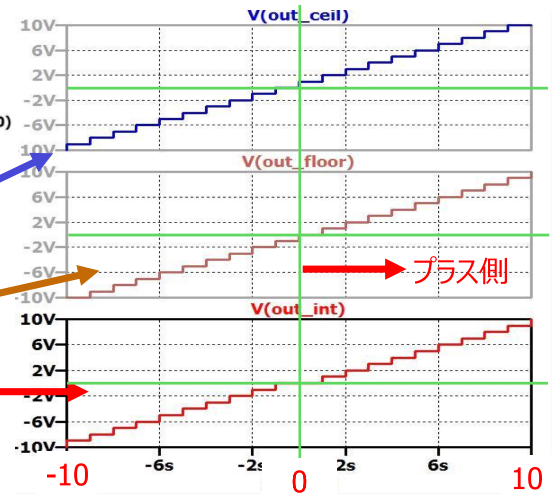
前頁の関数の計算式を試してみる（バイナリー・コードの各桁の係数が計算できているか？）
また、関数「INT」と「FLOOR」の違いは・・・？ （それぞれの引数が「正」の場合は等価）



・・・事前準備（補足説明-2）

関数「INT」と「FLOOR」は・・・それぞれの引数が「正」の場合は等価
関数「INT」と「CEIL」は　・・・それぞれの引数が「負」の場合は等価





CEIL(x)・・・
その数の小数部分を切り上げ

FLOOR(x)・・・
その数の小数部分を切り捨て

INT(x)・・・[x]（ガウスのｘ）と同じ。その数を超えない最大の整数

得られる結果はどれも整数になる

・・・事前準備（２）

Binary Code に従った条件で、誤差（tolerant）を変化させる関数を作る。

ユーザ定義関数名：wc　（・・・mcのパロディー風）

引数：　nom・・・　部品パラメータの標準値（nominal）
tol　・・・　誤差（実際の利用では部品ごとに「.PARAM」で指定する）
index・・・部品番号（０から順に番号を付ける）

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

この後に続けて1行に書く

If 文の条件式が成立したら・・・
nomの値を使う

この条件がtrueかどうか・・・
デフォルトは>0.5がtrue

はじめのIf 文の条件式が
成立しなかったら
・・・このif文を実行する

If 文の条件式
run の値が設定値「numruns」に一致したら・・・true。
「変数名、あるいは数」の値との比較は「==」とする。

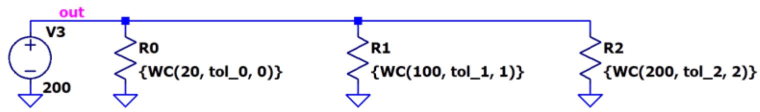
true の場合これを実行
（標準値に誤差を加算）

false の場合これを実行
（標準値から誤差を減算）

・・・事前準備（３）

Binary Code に従った条件で、誤差(Tolerant)を変化させる関数を確認する。

部品の値（パラメータ）を設定するには {・・・・} で関数をくくる。



実験として、3つの抵抗器にそれぞれの標準値と誤差を設定し、誤差のMin-Maxのすべての組み合わせをシミュレーションできるか・・・確認する。

```
.tran 2u
.step param run 0 8 1
.param numruns=8
.param tol_0=0.2
.param tol_1=0.4
.param tol_2=0.5
```

部品数が3個なので・・・ 2^3 =8

結果をLOGに残すコマンド

```
.meas I0 find I(R0) at 1u
.meas I1 find I(R1) at 1u
.meas I2 find I(R2) at 1u
```

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

LOGに表示させたのは、各抵抗を流れる電流。
i(r0)は1つおき、i(r1)は2つおき、i(r2)は4つおき・・・に値が変化していることが確認できる。

Measurement: i0

step	i(r0)	at
1	12.5	1e-06
2	8.33333	1e-06
3	12.5	1e-06
4	8.33333	1e-06
5	12.5	1e-06
6	8.33333	1e-06
7	12.5	1e-06
8	8.33333	1e-06
9	10	1e-06

Measurement: i1

step	i(r1)	at
1	3.33333	1e-06
2	3.33333	1e-06
3	1.42857	1e-06
4	1.42857	1e-06
5	3.33333	1e-06
6	3.33333	1e-06
7	1.42857	1e-06
8	1.42857	1e-06
9	2	1e-06

Measurement: i2

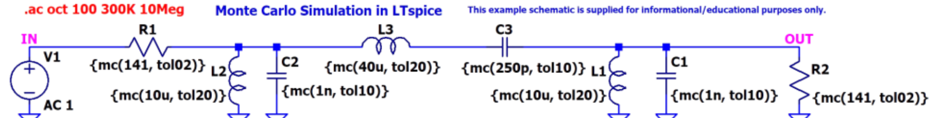
step	i(r2)	at
1	2	1e-06
2	2	1e-06
3	2	1e-06
4	2	1e-06
5	0.666667	1e-06
6	0.666667	1e-06
7	0.666667	1e-06
8	0.666667	1e-06
9	1	1e-06

最後の1回は3つとも標準値

LCバンドパスフィルターの評価（１）

LTspice ライブラリーの　Examples → Educational フォルダーにある・・・
MonteCarlo.ASC　の回路を使って、8個の部品の誤差の評価をする

.ac oct 100 300K 10Meg　　Monte Carlo Simulation in LTspice　　This example schematic is supplied for informational/educational purposes only.



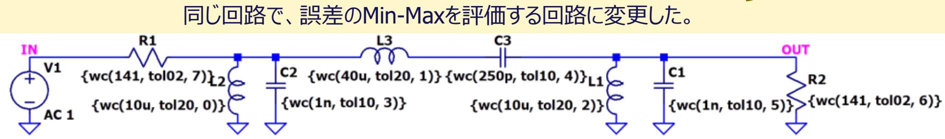

```
.param tol20=0.20 ; +/-20%
.param tol10=0.10 ; +/-10%
.param tol02=0.02 ; +/-2%
```

この後にシミュレーションでは10000回実行している

```
.step param X 0 1024 1 ; a dummy paramter to cycle Monte Carlo runs
.OPTIONS meascplxfmt=cartesian
.meas AC LF FIND freq WHEN abs(V(out))=0.25 TD=0.5Meg RISE=1
.meas AC HF FIND freq WHEN abs(V(out))=0.25 TD=2Meg FALL=1
```

-12dBを通過する周波数を読み取る

同じ回路で、誤差のMin-Maxを評価する回路に変更した。

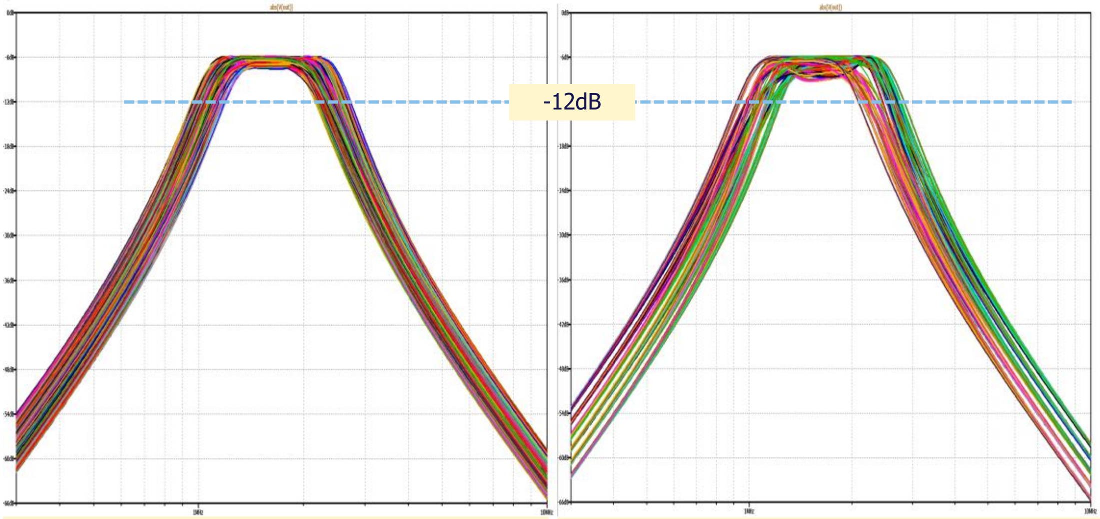



```
.step param run 0 256 1
.param numruns=256
.param tol02=0.02
.param tol10=0.1
.param tol20=0.2
```

```
.func wc(nom,tol,index) if(run==numruns,nom,if(binary(run,index),nom*(1+tol),nom*(1-tol)))
.func binary(run,index) INT(run/2**index)-2*INT(run/2**(index+1))
.OPTIONS meascplxfmt=cartesian
.meas AC LF FIND freq WHEN abs(V(out))=0.25 TD=0.5Meg RISE=1
.meas AC HF FIND freq WHEN abs(V(out))=0.25 TD=2Meg FALL=1
```

-12dBを通過する周波数を読み取る

LCバンドパスフィルターの評価（２）



mc によるシミュレーション結果（10000回）

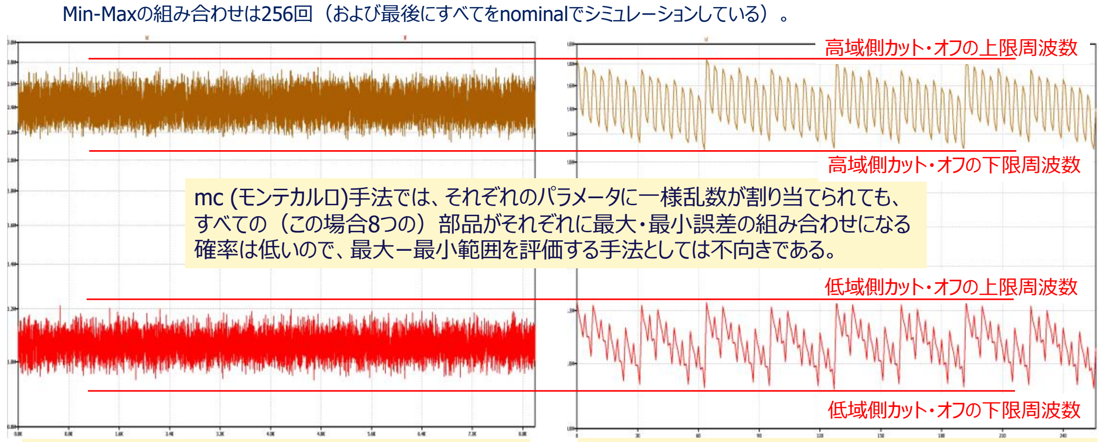
wc によるシミュレーション結果（10000回）

乱数(mc)とMin-Max組み合わせ比較

出力レベルがnominalのパスバンドの最大値(入力レベルを基準に-6dB)から-6dB減衰した周波数の低周波側と高周波側の周波数をプロットした。縦軸は周波数(一番下の目盛りが800kHz、一番上の目盛りが3MHz)、横軸は「.STEP」の回数。

.mcは10000回シミュレーションしたもの。

Min-Maxの組み合わせは256回（および最後にすべてをnominalでシミュレーションしている）。



mc によるシミュレーション結果（10000回）

wc によるシミュレーション結果（10000回）

まとめ

回路のばらつき具合を、統計的（確率的）に評価するには「mc」（一様乱数）による、それぞれの部品のばらつきの組み合わせが有効である。
しかし、すべての部品の誤差が個別に最大・最小になった場合をすべて組み合わせる時には、ここに示したような「Min-Max」手法を利用しなければならない。

＜注意点＞
.func を使う場合、まずは簡単な回路で評価をし、目的とする動作を正しく行えているかどうか・・・場合分けなどの間違いがないか・・・確認する。
if 文を使う場合、条件式とその判定結果(true, false)の動作（計算）が正しく実行されているか、簡単な回路で確認する。

＜if 文で使う判別記号＞

L==R	L は R と等しい
L>=R	L は R 以上
L<=R	L は R 以下
L>R	L は R を超える
L<R	L は R 未満
L!=R	L は R と等しくない

＜注意＞「＝」1つの場合は、
右辺の結果を左辺に代入する意味。
if文の判定には使えない。

参考文献

- アナログ技術セミナー　2019（講演集・冊子）
  - 「LTspiceセミナー　～最悪ケース解析の実行と短縮～
    - アバログ・デバイセズ株式会社　馬場　正幸　氏
- LTspice　ライブラリー
  - Examples　→　Educational
    - MonteCarlo.asc（シミュレーション用参考回路）

付録　（p進数・q桁目の値）

例題として・・・
日常的になじみのある10進数を例に考えてみる

10進数 の 例

p進数 の p^q の q 桁目は?

1番下の桁を「0桁目」とする

$N = 543|21$

この桁 (10^2) の 値を求める

以下の桁を切り捨てる… $[N/10^2]$

543 が 残る

さらに 10 で割って 整数部分を とる

$[N/10^2/10] = [N/10^{2+1}]$

これを 10倍 する… $[N/10^{2+1}] \times 10$

54 が 残る

最後に $543 - 540 = 3$ となる。

q 桁目の 値は 一般的に、

$[N/p^q] - p \times [N/p^{q+1}]$ になる

2進数の場合は → $\left[\frac{N}{2^n}\right] - 2 \times \left[\frac{N}{2^{n+1}}\right]$