

SIM マニュアル

ver.1.82

2020/06/23

Ver.1.81 2024/04/29

Ver.1.82 2024/06/12

三浦 高志

線形回路網シミュレーションプログラム (s i m. e x e) マニュアル

1992/08/25 三浦 高志

2020/04/12 VS 6.0 で32bit版にリビルド

```

素子属性          アナログ回路シミュレーションプログラムVer3.0
独立素子: e      i      r      l      c
従属素子: k...i to i  v...e to e  q...e to i  w...i to e
ブロック素子: b      関数素子: f

コマンド: / + コマンド名
ipart  cpart  dpart  padd  node  parts  free  dir  help
range  point  para  ac  disp  conv  auto  manu  man
cmode  smode  mode  db  val  mk  nomk  gpos  bmode
cal  save  load  bload  fname  def  dvar  dfnc  end
DOS コマンド: ! + MS-DOS コマンド名
バッチファイル実行: @ + ファイル名  ... コメント行は % で始める
バッチファイル作成・終了: #          |___ キー入力待ちは /wait
                                      |___ 中断の指示は /exit
計算式で使用可能な因子  |___ 巻戻しは /rewind, スキップは /skip
$a      j(式)  ~(式)  abs  sqrt  arg  log  log10  db
exp  udb  sin  cos  tan  asin  acos  atan
real  image  pow  unpw  hex  khex  mhex

システム変数
f fl fh fs s pi

```

```

? /range
データファイル名は ? s21-1
ac 入力信号源名は ? el
値は ? 1

range 周波数 最低値 ? 500
range 周波数 最高値 ? 1500
range 周波数 ステップ ? 50
el[1] j 0
f[500]
x5 [6.249430e-005 j -1.310620e-005] DB[ -83.8963] arg[ -11.8443]
f[550]
x5 [1.264000e-004 j -3.189178e-005] DB[ -77.6970] arg[ -14.1607]
f[600]
x5 [2.629061e-004 j -8.073583e-005] DB[ -71.2126] arg[ -17.0712]
f[650]
x5 [5.735657e-004 j -2.187432e-004] DB[ -64.2386] arg[ -20.8756]
f[700]
x5 [ 0.0013 j -6.586564e-004] DB[ -56.4967] arg[ -26.1066]
f[750]
x5 [ 0.0035 j -0.0023 ] DB[ -47.5630] arg[ -33.8111]
f[800]
x5 [ 0.0101 j -0.0106 ] DB[ -36.7113] arg[ -46.3547]
f[850]
x5 [ 0.0251 j -0.0704 ] DB[ -22.5293] arg[ -70.3469]
f[900]
x5 [ -0.5236 j -0.5852 ] DB[ -2.1004] arg[ -131.8198]
f[950]
x5 [ 8.9236 j 0.7883 ] DB[ 19.0446] arg[ 5.0485]
f[1,000]
x5 [ -0.7610 j 0.6253 ] DB[ -0.1317] arg[ 140.5927]
f[1,050]
x5 [ 0.0226 j 0.1132 ] DB[ -18.7576] arg[ 78.7272]
f[1,100]
x5 [ 0.0162 j 0.0228 ] DB[ -31.0791] arg[ 54.6583]

```

目次

目次.....	2
sim プログラムの起動.....	6
sim.pdf (sim マニュアル) の表示	6
機能概要	7
使用できる単位.....	8
使用出来る回路素子(素子属性名 と 素子の種類).....	8
登録されているコマンドの種類.....	9
関数電卓の関数名と因子	9
電卓としての利用法 :	10
factor : 因子 (電卓)	11
システム変数 :	12
シミュレーション用コマンドの紹介 (／に続けてコマンド名を入力する).....	13
!に続けてMS-DOS コマンドを入力して実行する	15
#を入力してバッチファイルを作成する	15
@に続けて、バッチファイル名を入力してバッチファイルを実行する	16
各コマンドの詳細説明と操作方法	17
(1) /ac (alternating current の略)	17
(2) /auto	18
(3) /bload (block load の略)	19
(4) /bmode (block mode の略)	20
(5) /cal (calculation の略)	21
(6) /cmode (clear mode の略)	22
(7) /conv (conversion の略)	22
(8) /cpart (clear parts の略)	24
(9) /db (decibel の略)	24
(10) /def (define functions の略)	24
(11) /dfnc (display functions の略)	25
(12) /dir (directory の略)	25
(13) /disp (display nodes の略)	25
(14) /dpart (display parts の略)	26
(15) /dvar (display variables の略)	26
(16) /end.....	26
(17) /fname (file name の略)	27

目次

(18) /free	27
(19) /gpos (give positions の略)	27
(20) /grph.....	27
(21) /help.....	28
(22) /ipart (initialize parts の略)	28
(23) /load.....	31
(24) /man	31
(25) /manu	31
(26) /mode.....	32
(27) /node.....	32
(28) /padd (parts addition の略)	32
(29) /para (parameter の略)	32
(30) /parts (parts numbers の略)	34
(31) /point.....	34
(32) /range	34
(33) /rewind.....	36
(34) /save	36
(35) /skip.....	38
(36) /smode (set mode の略)	38
(37) /val (value の略)	38
sim.exe の操作練習	39
図 ex-1.....	39
/ipart	39
/dpart	40
/point.....	40
/para.....	41
シミュレーションコマンド /para, /ac, /range における計算回数.....	42
計算回数の実測値.....	45
図 ex-2.....	50
/padd	50
/disp.....	51
/mode, /cmode, /smode	51
/mk, /nomk, /db, /val, /auto, /manu, /gpos, /bmode, /cal, /disp.....	52
Maple にコピーアンドペーストして、方程式を解きます。	56
図 ex-3.....	57
/range.....	58

目次

/ac	59
図 ex4	60
/bload.....	61
/point.....	61
/para.....	62
Maple を利用する 1	63
Maple を利用する 2	75
回路ブロックとトランジスタ・FET・IC などの等価回路.....	78
回路ブロックのデータファイルを作成するには	79
図 ex5	80
サンプルデータファイルの内容.....	83
バッチファイルのサンプル	85
サンプル回路	93
図 s1.sb で設計する回路図	93
図 s2.sb で設計する回路図	93
図 s3.sb で設計する回路図	94
サンプル回路の概要説明	94
Sim.exe で使用する標準部品として準備してあるブロック素子について.....	103
図 npn トランジスタの回路図記号と等価回路 TRN. CIR, TRN2. CIR	103
図 pnp トランジスタの回路図記号と等価回路 TRP. CIR, TRP2. CIR	104
図 n 型デプリーションタイプ FET の回路図記号と等価回路 FETD.CIR	105
図 n 型エンハンスメントタイプ FET の回路図記号と等価回路 FETE. CIR	106
サンプルバッチファイルの回路図と説明	107
(1) バッチファイル名 s5. sb で扱う回路図.....	107
(2) バッチファイル名 s6. sb で扱う回路図.....	123
(3) バッチファイル名 s7. sb で扱う回路図.....	129
(4) バッチファイル名 s8. sb で扱う回路図.....	138
(5) バッチファイル名 s9. sb で扱う回路図.....	147
(6) バッチファイル名 s10. sb で扱う回路図	152
(7) バッチファイル名 s11. sb で扱う回路図	160
(8) バッチファイル名 ss1. sb で扱う回路図	171
(9) バッチファイル名 ss2. sb で扱う回路図	178
ss2 の差動増幅回路を回路ブロックにして、部品として利用する。	186

目次

Maple を使って、ss2premp.sb の回路の伝達関数を求めます。	205
差動増幅回路について	209
差動増幅回路.....	210
Maple を利用する	213
(10) バッチファイル s20.sb で扱う3つの回路図	215
Maple を使って、s20.sb の係数方程式 s20.coe を解く。	222
サンプル回路全体を順に実行するバッチファイル ALL.sb	226
Maple の利用方法.....	227
図 ex4	227
独立電源の表示	232
LC 回路 グラフが利用できると分かりやすい	235
RLC 回路 3 個の素子の配置を変えるだけで4通りの特性	237
数式処理ソフトを sim.exe と併用することを推奨します	261
Mathematica を利用する場合	262
フィルタ回路の参考資料	263
5 次の楕円関数 BE フィルタの設計例.....	264
Vector で公開中のソフトとデータ	268

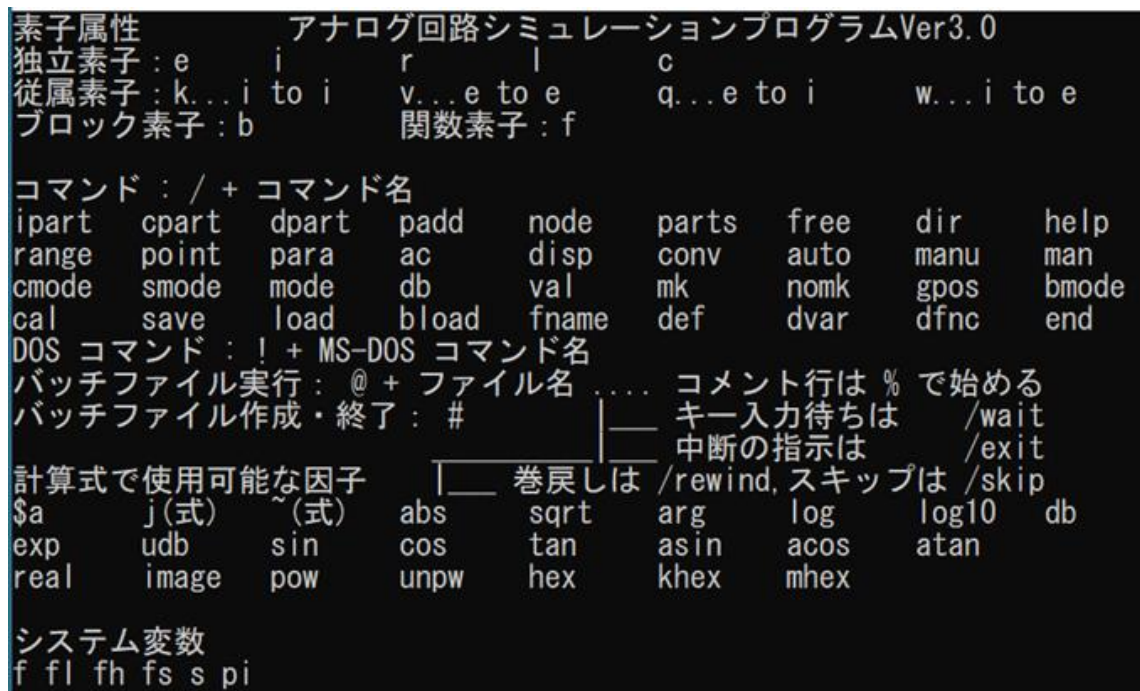
sim プログラムの起動

sim プログラムの起動

ダウンロードした ZIP ファイルを解凍して、使用するドライブ（例えば e:）のルートフォルダに解凍します。例えば、ドライブ e: にフォルダ「sim」が作成されます。

「sim」フォルダに含まれている実行ファイル sim.exe をダブルクリックします。

「Windows によって PC が保護されました」と表示されるので、「詳細情報」をクリックして「実行」をクリックします。



sim.pdf (sim マニュアル) の表示

sim.exe のコマンドラインで /man と入力すると、マニュアル sim.pdf が開きます。

sim.pdf をクローズすると、sim.exe に戻ります。

sim.pdf を開いたままで、sim.exe の作業を続けたい場合は、sim.exe が保存されているフォルダを開いて sim.pdf をダブルクリックしてください。

sim.exe は DOS 窓用のソフトなので同時に 2 つのプログラムを実行できません。

機能概要

1. 線形回路網の入力及び定常状態シミュレーションを実行可能
(グラウンドをノード0として、ノードiとノードjに接続されている部品をすべて登録する)
2. 線型回路網のネットリストをファイルとして入出力可能
(ファイル名及び回路網の入出力ノードの番号を同時に記録する)
3. 線形回路網を機能ブロックとしてファイルに入出力が可能
4. 新たな回路網を入力する時に、ファイルとして作成されている機能ブロックをR, L, Cなどの部品と同様に使用することが可能
5. 入力された回路網の回路方程式を作成し、画面表示やファイルに保存が可能
(ファイル名は conv. coe となる)
6. 作成された回路方程式のファイル conv. coe は、Maple に読み込んで、数式処理により伝達関数を求めることが出来る
7. 操作した内容をバッチファイルとしてファイルに保存する事が出来る
8. 操作内容のバッチファイルを読み込みながら、処理を再現する事が出来る
9. バッチファイルから他のバッチファイルにジャンプすることが出来る
10. バッチファイル実行中に入力を受け付け、バッチファイルの巻き戻し、バッチファイルの最後へのジャンプ、バッチファイルの実行停止などを指定出来る
11. 複素数電卓として使用する事が出来る
(演算処理に名前をつけて、手続きを登録する事が出来る)
12. プログラムに対する入力は、小文字でも大文字でも入力可能
13. ネットリスト入力の際に素子の値は、通常の電子工学で使用されている単位が使用出来る。
抵抗が 1 K オームのときは 1 k (あるいは 1 K) と入力が可能。
同様に 1 M オームなら、1 m (1 M)、
1 ミリオームは 0. 001 または 1/1k と入力する。
1 000 M オームなら、1 g (1 G)
コンデンサで 0. 1 u F なら、0. 1 u (0. 1 U)
0. 001 u F なら、1 n (1 N)
120 p F なら、120 p (120 P)

機能概要

使用できる単位

k	...	10^3
m	...	10^6
g	...	10^9
u	...	10^{-6}
n	...	10^{-9}
p	...	10^{-12}

14。コマンド `/ipart`, `/padd`, `/para` 及び `/ac` において、素子の値などを数式で入力することが可能。

例えば、規格表にトランジスタのコレクタ抵抗がアドミタンスで記載されていた場合（`50uS` となっていた時）

`1/50u` と入力すればよい

また、`re*(1+k)` の様に、変数名も使用可能である

使用出来る回路素子(素子属性名 と 素子の種類)

e	独立電圧源	i	独立電流源
r	抵抗素子	c	コンデンサ素子
l	インダクタンス素子		
v	電圧従属電圧源	q	電圧従属電流源
w	電流従属電圧源	k	電流従属電流源

b ブロックとして登録された素子

f 関数（数式）として登録された素子（現在は対応していない）

登録されているコマンドの種類

(1) シミュレーション用コマンドの実行： / + コマンド名

コマンドの実行は、画面左端に表示される「？」の隣で点滅するカーソル位置から「/」を入力して、以下に示す「コマンド名」を入力して、「Enter」を押します。

?/ipart 「Enter」のように。

ipart	cpart	dpart	padd	node	parts	free	help	
range	point	para	ac	disp	conv	auto	manu	man
cmode	smode	mode	db	val	mk	nomk	gpos	bmode
cal	save	load	bload	fname	def	dvar	dfnc	end

注意： grph コマンドは VS 6.0 でビルドした時に削除した。2020/04/10

(2) DOS コマンドの実行： ! + MS-DOS コマンド名

DOS コマンドの実行は「？」の隣から「!」を入力してから、cls や copy などの「DOS コマンド名」を入力して、「Enter」を押します。

?!cls 「Enter」のように。

(2) バッチファイルの作成及び終了：「？」の隣から「#」を入力して「Enter」を押します。

コマンドの入力などを実行後、作成終了するには、もう一度「#」を入力します。

キー入力待ちは /wait

コメント行は %% で始める

(4) バッチファイルの実行：「？」から「@ + ファイル名」を入力して「Enter」を押します。

?@ex1 のように、拡張子「.sb」は不要です。

バッチファイルの巻き戻しは /rewind

バッチファイルの最後にジャンプは /skip

バッチファイルの終了は /exit

関数電卓の関数名と因子

\$a	j(式)	^(式)	abs	sqrt	arg	log	log10	db
exp	udb	sin	cos	tan	asin	acos	atan	
real	image	pow	unpw	hex	khex	mhex		

システム変数

f	fl	fh	fs	s	pi
---	----	----	----	---	----

電卓としての利用法：

電卓としての利用法：

変数を回路データの変数と合わせて3000個まで使用できる。また、ユーザ関数（手続き）を1500個まで定義して使用できる。変数名、関数名は15文字まで有効である。

1. 変数に値を代入する。変数名＝値（または式）

? a = 100

? a = 1 + j 3 (変数は全て複素数として扱う)

? a b c = a * b / c などのように入力する

? > 変数名 (直前の計算結果が変数に代入される)

2. 関数（手続き）の定義：

/d e f

関数定義： 関数名をアルファベット15文字以内で入力して下さい ?

と表示されるので、関数名をアルファベット15文字以内で入力する。

例えば、nyuryoku1

f n c (nyuryoku) = ... リターン ならこのまま

と表示されるので式を入力する。既に、同名の関数が定義されている時には

... の所に式が表示される。

a 1 * \$ b - c のように、関数定義されている他の関数を利用することも可能

この場合、関数名に\$をつける。

関数を定義したら、\$nyuryokuを入力すると、定義した計算が行われる。

3. 計算：

? 100 + 30

? a 1 / (b - c)

? \$ a * (b + c)

? s i n (a)

? p o w (2, b) のように、f a c t o rに含まれる全ての因子を使用可

定義済みの変数名と値は/d v a rで、定義済みの関数名（手続き名）と式は

/d f n cで確認ができる。

定義した関数は、/saveで接続情報(***.cir)と一緒に保存されて、/loadで読み込みます。

変数は/convで係数行列と一緒に保存されますがsim.exeで読み込みません。

電卓としての利用法：

f a c t o r : 因子 (電卓)

- (1) \$ に続けて関数名 (\$ a など) / d e f で定義した手続き (式) を実行して表示する

- (2) j に続けて変数名 (j b 2 など) または、j (式) j は虚数単位の意味

- (3) ~ に続けて変数名 (~ c など) または、~ (式) ~ は共役複素数の意味

- (3) a b s (式) 絶対値 (式は複素数でも可)

- (4) s q r t (式) 平方根 (式は複素数でも可)

- (5) a r g (式) 式が $a + j b$ という値になる時の
偏角 ($\text{atan}(b/a)$)、単位は度 (degree)

- (6) l o g (式) 自然対数 (式は複素数でも可)

- (7) l o g 1 0 (式) 常用対数 (式は複素数でも可)

- (8) d b (式) 式の値をデシベル値に変換する

- (9) e x p (式) 自然対数の底によるべき乗 (複素数対応)

- (10) u d b (式) 式の値をデシベル値から倍率に変換する

- (11) s i n (式) 式の値 (単位は d e g r e e) の正弦

- (12) c o s (式) 式の値の余弦
 $c o s (45) = 0.707$

- (13) t a n (式) 式の値の正接

- (14) a s i n (式) 式の値の逆正弦

- (15) a c o s (式) 式の値の逆余弦

電卓としての利用法：

- | | |
|---------------------------------|---|
| (16) <code>a t a n</code> (式) | 式の値の逆正接
<code>a t a n (1) = 4 5</code> |
| (17) <code>r e a l</code> (式) | 式の値の実数部 |
| (18) <code>i m a g e</code> (式) | 式の値の虚数部 |
| (19) <code>p o w</code> (式1、式2) | 式1の式2乗
<code>p o w (2 , 3) = 8</code> |
| (20) <code>u n p w</code> (b、c) | $b = c ^ d$ となる、dを求める。 |
| (21) <code>h e x</code> (式) | 式の値を16進数から10進数に変換する |
| (22) <code>k h e x</code> (式) | 式の値をキロ単位の16進数から10進数に変換する |
| (23) <code>m h e x</code> (式) | 式の値をメガ単位の16進数から10進数に変換する |

システム変数：

<code>f</code>	周波数特性などの計算に使用する周波数値
<code>f l</code>	／ <code>r a n g e</code> で使用する周波数範囲の最低値
<code>f h</code>	／ <code>r a n g e</code> で使用する周波数範囲の最高値
<code>f s</code>	／ <code>r a n g e</code> で使用する周波数の上昇ステップ値
<code>s</code>	$j 2 * p i * f$ を格納するための変数 (角周波数)
<code>p i</code>	円周率の値 3. 1 4 1 5 9 6

シミュレーション用コマンドの紹介 (／に続けてコマンド名を入力する)

- (1) `i p a r t` 回路情報 (ネットリスト) を初期化して新規に回路情報を
入力する。 ／ `c p a r t` に続けて、／ `p a d d` を行なうのと同等の機能

- (2) `c p a r t` 回路情報を初期化する

- (3) `d p a r t` 回路情報を表示する

- (4) `p a d d` 回路情報に素子を追加する

- (5) `n o d e` 回路情報で使用されている最大ノード番号を表示する

- (6) `p a r t s` 回路情報で使用されている素子の総数を表示する

- (7) `f r e e` 使用可能なメモリの残量を表示する (あまり意味がない)

- (8) `h e l p` ヘルプ画面を表示する (コマンド名などの確認が出来る)

- (8) `r a n g e` 指定された周波数ステップごとに周波数特性を計算して、
指定ノードの電圧振幅を表示する

- (9) `p o i n t` 周波数 $f = 0$ に設定しておく。指定ノードの電圧を計算する。

- (10) `p a r a` 周波数は $f = 0$ に設定しておく。素子名を指定して、素子値を変
化させたときの指定ノードの動作点電圧を計算する。
すべての独立電圧源及び独立電流源は与えられた値として
計算される。

- (11) `a c` ／ `p a r a` で動作点を決定してから、周波数 f を設定する。
入力信号名と値を入力し、変化させたい素子名と変化の
範囲を入力する。回路中の他の独立電圧源及び独立電流源を
0 として指定ノードの電圧振幅を計算する。

- (12) `g r p h` 指定されたデータファイルを読み込んで、周波数特性などの
グラフを表示する → 削除 2020/04/10

シミュレーション用コマンドの紹介（／に続けてコマンド名を入力する）

- (13) `d i s p` 計算するノード番号を指定する
- (14) `c o n v` 出力先を指定すると、回路情報から数式による係数行列（回路方程式の係数ファイル `conv.coe`）を作成して出力する。
必要に応じてリネームしてください。
- (15) `s a v e` 回路情報にファイル名と接続ノード番号をつけて保存する
- (16) `l o a d` 指定されたファイルから回路情報を読み込む
- (17) `b l o a d` 機能ブロックのファイル（ブロック素子）を展開する
- (18) `f n a m e` ／ `l o a d` で読み込んだファイル名を表示する
- (19) `d e f` 関数電卓で使う手続きを定義する、手続き名は15文字まで
- (20) `d v a r` 使用されている変数名（`r3 b1` 等も可）と値を表示する
- (21) `d f n c` 定義されている手続き名と式を表示する
- (22) `e n d` `s i m. e x e` を終了する。
- (23) `c m o d e` モード設定クリア、計算は値で行い、ファイルを作らない
- (24) `s m o d e` モード設定セット、計算はdBで行い、ファイルを作る
- (25) `m o d e` 現在のモードの設定状態を表示する
- (26) `d b` 計算モードをdBにする
- (27) `v a l` 計算モードを値にする
- (28) `m k` ファイルを作成にするモードに切り替える
- (29) `n o m k` ファイルを作らないモードに切り替える

シミュレーション用コマンドの紹介 (／に続けてコマンド名を入力する)

(30) m k /range, /para, /ac において、計算結果をファイルとして保存するモードに切り替える。
(作成されたファイルは /grph でグラフ表示に使われていた。)

！に続けてMS-DOSコマンドを入力して実行する

!dir フォルダに含まれる全ファイル名を表示する。
!dir *.sb 拡張子が「.sb」のファイルを表示する。
!type trn.cir テキストファイル trn.cir を表示する。
!cls 画面をクリアする。
!copy test.d ex-2.d ファイル test.d を ex-2.d にコピーする。
!del test.d ファイル test.d を削除する。
!ren test.d ex-2.d ファイル test.d を ex-2.d に名前を変更する。

#を入力してバッチファイルを作成する

コマンドラインで#を入力すると、バッチファイルを作成するモードに入ります。
このあと操作した内容が全て記録されます。
バッチファイルの作成を終了するには、コマンドラインから再び#を入力します。
test.sb というバッチファイルが出力されます。
コマンドラインで @test と入力すると、操作した手順が実行されます。

例えば、/range コマンドで周波数特性を計算するときなど、入力信号源名、値、最低周波数、最高周波数、周波数ステップなど決まっている入力を何度も繰り返す時などはこれらをバッチファイルにしておくと便利です。

操作ごとに分かりやすいファイル名に変更しておくと、後で利用しやすいと思います。
ファイルの拡張子は .sb ですが、ただの文字列なので、普通のエディタを使って編集することもできます。
行の先頭を%で始めるとコメント行になります。
コメント行はバッチファイルを実行すると、画面に表示されますから操作内容の説明として利用できます。

/wait を挿入すると、バッチファイルを実行中にその行で一時停止します。リターンキーで再開します。

バッチファイルが一時停止している時に、/rewind を入力すると、バッチファイルの先頭に戻ります。/skip を入力すると、バッチファイルの最後にジャンプします。

！に続けてMS-DOSコマンドを入力して実行する

バッチファイルの中で別のバッチファイルを呼んでいる時には、呼ばれたバッチファイルが一時停止しているときに /skip すると元のバッチファイルに戻って続きを実行します。

/exit を入力すると、バッチファイルを強制終了します。呼ばれたバッチファイルの実行中であっても、元のバッチファイルの実行を停止します。

バッチファイルは10レベルまでネスティングが可能です。

ネスティングしたバッチファイルを作成するには、最初に、#コマンドでネスティングしていないバッチファイルに必要な個数作成してから、エディタを使ってそれぞれのバッチファイルと呼び出すバッチファイルを作成します。

ALL.sb, ALL1.sb, ALL2.sb, ALL3.sb のファイルを参照してください。ALL.sb がそれぞれのバッチファイルと呼び出すバッチファイルです。

呼び出されるバッチファイルも操作内容が分かるようにコメントを追加してあります。

適当なタイミングで、/wait や!cls を入れておくと見やすいかもしれません。

@に続けて、バッチファイル名を入力してバッチファイルを実行する

コマンドラインから@bat1のように入力すると、bat1.sbをロードしてキーボード入力の代わりに、ファイルから1行ずつ読んで、自動実行します。

実行がはや過ぎる場合には、エディタを使用して適当な数コマンドごとに、/wait を挿入しておくこと確認しながら実行が可能となる。また、実行後にスクリーンをスクロールして確認することもできます。

各コマンドの詳細説明と操作方法

各コマンドの詳細説明と操作方法

(1) /ac (alternating current の略)

システム変数 f で指定される周波数において、指定の素子を希望の範囲で変化させた時の交流信号の振幅を計算する。入力信号源と指定された素子以外の電圧源は 0 として交流成分だけの計算を行なう。ファイルを作成するモードの場合 (/mk) には指定されたノードの電圧値の絶対値をファイルにセーブする。

- a /auto モードかつ 最高値/最低値の比が 10 以下の時は、
入力されたステップ値で計算を行なう。
- b /manu モードまたは 最高値/最低値の比が 10 を超える時は、
/gps で設定された計算ポイント数で最低値から最高値まで
等比級数で増加する様にステップ値を自動的に変化させる。

/para と /ac を交互に実行し、直流動作点と交流利得の初期設定を行ない、次に /range によって使用する全周波数範囲における特性を計算する。

```

B? /ac
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? r1
ac r1 最低値 ? 3k
ac r1 最高値 ? 4k
ac r1 ステップ ? 100
e1[0.2209 j 0 ]
r1[3000 ] f[1.000 ]
x5 [ -44.2770 j -0.8588 ] abs[ 44.2853] arg[ -178.8888]
x6 [ -44.2769 j -0.8611 ] abs[ 44.2853] arg[ -178.8858]
r1[3100 ] f[1.000 ]
x5 [ -45.0898 j -0.8746 ] abs[ 45.0983] arg[ -178.8888]
x6 [ -45.0897 j -0.8769 ] abs[ 45.0983] arg[ -178.8858]
r1[3200 ] f[1.000 ]
x5 [ -45.8794 j -0.8900 ] abs[ 45.8880] arg[ -178.8887]
x6 [ -45.8793 j -0.8923 ] abs[ 45.8880] arg[ -178.8858]
r1[3300 ] f[1.000 ]
x5 [ -46.6467 j -0.9049 ] abs[ 46.6555] arg[ -178.8886]
x6 [ -46.6467 j -0.9072 ] abs[ 46.6555] arg[ -178.8859]
r1[3400 ] f[1.000 ]
x5 [ -47.3927 j -0.9195 ] abs[ 47.4017] arg[ -178.8886]
x6 [ -47.3927 j -0.9217 ] abs[ 47.4017] arg[ -178.8859]
r1[3500 ] f[1.000 ]
x5 [ -48.1183 j -0.9336 ] abs[ 48.1274] arg[ -178.8885]
x6 [ -48.1183 j -0.9358 ] abs[ 48.1274] arg[ -178.8859]
r1[3600 ] f[1.000 ]
x5 [ -48.8243 j -0.9473 ] abs[ 48.8335] arg[ -178.8884]
x6 [ -48.8243 j -0.9495 ] abs[ 48.8335] arg[ -178.8859]
r1[3700 ] f[1.000 ]
x5 [ -49.5115 j -0.9607 ] abs[ 49.5208] arg[ -178.8884]
x6 [ -49.5114 j -0.9628 ] abs[ 49.5208] arg[ -178.8859]
r1[3800 ] f[1.000 ]
x5 [ -50.1806 j -0.9737 ] abs[ 50.1900] arg[ -178.8883]
x6 [ -50.1805 j -0.9758 ] abs[ 50.1900] arg[ -178.8859]
r1[3900 ] f[1.000 ]
x5 [ -50.8322 j -0.9864 ] abs[ 50.8418] arg[ -178.8883]
x6 [ -50.8322 j -0.9885 ] abs[ 50.8418] arg[ -178.8860]
r1[4000 ] f[1.000 ]
x5 [ -51.4672 j -0.9988 ] abs[ 51.4769] arg[ -178.8883]
x6 [ -51.4672 j -1.0008 ] abs[ 51.4769] arg[ -178.8860]

```

/mk の場合

各コマンドの詳細説明と操作方法

```
? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

値を変化させる素子名は ? r1
ac r1 最低値 ? 3k
ac r1 最高値 ? 4k
ac r1 ステップ ? 100
e1[1] j 0 ]
r1[3000 ] f[1,000 ]
x5 [ -452.5988 j -8.8132 ] abs[ 452.6846] arg[ -178.8845]
x6 [ -452.5988 j -8.8132 ] abs[ 452.6846] arg[ -178.8844]
r1[3100 ] f[1,000 ]
x5 [ -452.5988 j -8.8124 ] abs[ 452.6846] arg[ -178.8846]
x6 [ -452.5988 j -8.8125 ] abs[ 452.6846] arg[ -178.8845]
r1[3200 ] f[1,000 ]
x5 [ -452.5988 j -8.8117 ] abs[ 452.6846] arg[ -178.8846]
x6 [ -452.5988 j -8.8117 ] abs[ 452.6846] arg[ -178.8846]
r1[3300 ] f[1,000 ]
x5 [ -452.5989 j -8.8110 ] abs[ 452.6846] arg[ -178.8847]
x6 [ -452.5989 j -8.8110 ] abs[ 452.6846] arg[ -178.8847]
r1[3400 ] f[1,000 ]
x5 [ -452.5989 j -8.8103 ] abs[ 452.6846] arg[ -178.8848]
x6 [ -452.5989 j -8.8104 ] abs[ 452.6846] arg[ -178.8848]
r1[3500 ] f[1,000 ]
x5 [ -452.5989 j -8.8097 ] abs[ 452.6846] arg[ -178.8849]
x6 [ -452.5989 j -8.8098 ] abs[ 452.6846] arg[ -178.8849]
r1[3600 ] f[1,000 ]
x5 [ -452.5989 j -8.8092 ] abs[ 452.6846] arg[ -178.8850]
x6 [ -452.5989 j -8.8092 ] abs[ 452.6846] arg[ -178.8850]
r1[3700 ] f[1,000 ]
x5 [ -452.5989 j -8.8086 ] abs[ 452.6846] arg[ -178.8850]
x6 [ -452.5989 j -8.8087 ] abs[ 452.6846] arg[ -178.8850]
r1[3800 ] f[1,000 ]
x5 [ -452.5989 j -8.8081 ] abs[ 452.6846] arg[ -178.8851]
x6 [ -452.5989 j -8.8082 ] abs[ 452.6846] arg[ -178.8851]
r1[3900 ] f[1,000 ]
x5 [ -452.5989 j -8.8076 ] abs[ 452.6846] arg[ -178.8852]
x6 [ -452.5989 j -8.8077 ] abs[ 452.6846] arg[ -178.8852]
r1[4000 ] f[1,000 ]
x5 [ -452.5989 j -8.8072 ] abs[ 452.6846] arg[ -178.8852]
x6 [ -452.5989 j -8.8072 ] abs[ 452.6846] arg[ -178.8852]
```

注意：各種コマンドでは、ファイル名などを入力すると小文字に変換されます。漢字を使用すると文字化けが生じます！英数字を入力してください。

(2) /auto

シミュレーションコマンド `/para`, `/ac`, `/range` で計算ステップを最高値／最低値の比の値によって自動的に切り替えるモードに設定する。

比の値が 1.0 以下であれば、基本的には入力されたステップ値で計算を行なうので、指定したステップ値ごとに 300 ポイントまでの計算が実行できる。

最低値が 0 の場合は、

最高値／ステップ値 > 300 なら、ステップ値が最高値／300 に変更される。

最低値が 0 でなければ、

(最高値－最低値) > 300 なら、

ステップ値が (最高値－最低値) / 300 に変更される。

比の値が 1.0 を超えていれば、`/gpos` のステップ数で等比級数で計算を行なう。

各コマンドの詳細説明と操作方法

/gpos で設定できる最大のポイント数は 300 である。

計算が実行される範囲は指定した最低値～最高値以内の範囲となる。

(3) /bload (block load の略)

ブロック素子ファイルを読み込んで解凍する

回路情報の中にブロック素子が使用されている時には、/bload によって

回路を展開しなければ/rangeなどで計算を行なうことができない。

ブロック素子のネスティングは許されている。

ネスティングのレベルに制限はないが、

/bload 後の素子数が 3000 以内、ノード数が 1500 以内に制限される。

```
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0          ]
left[ 0] right[ 2] parts[ r1          ] value[ 1e+004 j 0          ]
left[ 0] right[ 3] parts[ e2          ] value[      24 j 0          ]
left[ 0] right[ 4] parts[ ce          ] value[    0.001 j 0          ]
left[ 0] right[ 4] parts[ re          ] value[    2200 j 0          ]
left[ 0] right[ 6] parts[ r1          ] value[    1000 j 0          ]
left[ 1] right[ 2] parts[ cc          ] value[    0.001 j 0          ]
left[ 2] right[ 3] parts[ r2          ] value[ 5e+004 j 0          ]
left[ 2] right[ 5] parts[ b1          ] value[      0 j 0          ]
@ func trn:2,4,5
left[ 3] right[ 5] parts[ rc          ] value[    3800 j 0          ]
left[ 5] right[ 6] parts[ cc          ] value[    0.001 j 0          ]
最大の 素子番号 = 11
最大のノード番号 = 6
未解凍のブロックを 1 個含んでいます
?
```

```
? /bload
bname b1 sfnc trn:2,4,5

func trn:2,4,5 name trn.cir blk b1
@ func trn:1,2,3

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.6 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
1 2 3 4 5 6

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 5
ノード番号 (0 なら終了) ?

使用した式の個数 0、密度 0.000 %
使用した式のサイズ 463
最大の 素子番号 = 16
最大のノード番号 = 10
```


各コマンドの詳細説明と操作方法

/bload 後の接続情報

```
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0
left[ 0] right[ 2] parts[ r1          ] value[ 1e+004 j 0
left[ 0] right[ 3] parts[ e2          ] value[      24 j 0
left[ 0] right[ 4] parts[ ce          ] value[    0.001 j 0
left[ 0] right[ 4] parts[ re          ] value[    2200 j 0
left[ 0] right[ 6] parts[ rl          ] value[    1000 j 0
left[ 1] right[ 2] parts[ cc          ] value[    0.001 j 0
left[ 2] right[ 3] parts[ r2          ] value[ 5e+004 j 0
left[ 2] right[10] parts[ r1b1        ] value[     0.1 j 0
left[ 3] right[ 5] parts[ rc          ] value[    3800 j 0
left[ 4] right[ 7] parts[ rbb1        ] value[     0.1 j 0
left[ 5] right[ 6] parts[ cc          ] value[    0.001 j 0
left[ 5] right[ 9] parts[ rbb1        ] value[     0.1 j 0
left[ 7] right[ 8] parts[ reb1        ] value[     26 j 0
left[ 7] right[ 9] parts[ kb1         ] value[    100 j 0
@ master[reb1          ] left[ 7] right[ 8]
left[ 8] right[10] parts[ ebb1        ] value[     0.6 j 0
最大の素子番号 = 16
最大のノード番号 = 10
? _
```

ブロック素子を読み込むと、最大のノード番号が 6 から 10 に増えた。

(4) /bmode (block mode の略)

回路リストあるいは回路リストのファイルでブロックを使用している時に、ブロック名を ? で始めると、/bload でブロックを解凍する時に、キーボードから任意のブロックファイル名を入力可能となっている。

ブロック名入力でリターンだけを入力すると、回路リストに入力されているブロック名を使用する。

/bmode コマンドを入力して、1 を入力すると、? で始めていないブロックファイル名であっても、キーボードから入力するよう出来る。/bmode で 0 を入力した場合は? で始まるブロック名だけがキーボードからファイル名の再指定が可能となる。

/ipart, /padd で、ブロック素子を入力する時場合には、接続情報を入力するが、

trn: 1, 2, 3 と通常は入力する所を、

各コマンドの詳細説明と操作方法

? : 1, 2, 3 あるいは ? t r n : 1, 2, 3 と入力すると、
実際にロードするブロックファイル名をキーボードから入力出来るようになる。

/b m o d e で 1 を入力してあれば、? がなくても、すべての
ブロックファイル名をキーボードから入力して、目的に合ったブロックファイル
名を指定出来るようになる。

(5) /cal (calculation の略)

/range, /para, /ac, /point などのシミュレーションコマンドにおいて、/disp では
指定された表示ノードの電圧値をそのままあるいはデシベル値として計算している。

従って、抵抗 R 1 に流れる電流値を計算して出力することが出来ない。
同様に、

抵抗 R 1 で消費される電力値の出力も出来ない。

このような場合には、コマンド/cal により、上記の要求が満たされる。

もし、単純に電圧値ではなく電流や電力を計算したい時には次の様に設定を行う。

1 /c a l

現在のモードが表示される

ここで 1 を入力する

2 /d i s p

現在の計算式が表示される (最初は何も表示されない)

ここで、リターンキーを入力する

1 番目の計算式を入力して下さい と表示されるので式を入力する
ノードの電圧値は x に ノード番号をつけて、x 3 の様に入力する。

(x 5 - x 4) / r 2 抵抗 R 2 がノード 4 と 5 の間に接続
されていて、抵抗に流れる電流を計算する時

d b (x 8) ノード 8 の電圧をデシベル値に換算したい時
a b s (x 8) ノード 8 の電圧値の絶対値を計算したい時

各コマンドの詳細説明と操作方法

$x_5 - x_4$ ノード4とノード5の電位差を計算したい時
 計算式には、`sin (..)`、`db (..)`、および自作の関数など `sim.exe` で
 定義されているすべての関数を使用することが出来る。

- 3 `/cal` のモード設定を `1` にすると、
 現在設定されている `/disp` で設定した表示ノード番号は無効となり、
`/db` や `/val` も無視される。

`/cal` のモードを `0` に戻すと、
`/disp` で設定した表示ノード番号は再び有効となり、
`/db` や `/val` も有効となる。
`/ac` と `/range` では、ノード電圧の絶対値が使われる。

(6) `/cmode` (clear mode の略)

mode をクリアする

```
? /cmode
データファイルを作成しない..... /mk または /nomk で切り替える
値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /auto (自動切り替え) です
..... /auto または /manu で切り替える
等比級数のポイント数は 32 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
1 2 3 4 5 6
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する
```

(7) `/conv` (conversion の略)

`/save` は回路の接続情報を保存するが、`/conv` は接続情報を回路方程式の係数行列に変換して出力する。回路の各ノード間の抵抗情報が文字式の状態で保存されている。

この係数行列のデータ `conv.coe` を数式処理ソフト Maple に入力すると、回路方程式を解くことが出来る。ファイル名は必要に応じてリネームしてください。

```
? /conv
係数行列の出力先を選択して下さい
ファイル(conv.coe) F, 画面 C
```

出力された係数行列ファイル Maple 用

```
restart: with(linalg):
```

```
siki:=matrix(10,10,0):e:=vector(10,0):
```

```
siki[ 1, 1] := 1 ;
```

(6) `/cmode` (clear mode の略)

各コマンドの詳細説明と操作方法

```

e[1] := +e1 ;
siki[ 2, 1] := -s*cc ;
siki[ 2, 2] := +1/r1+s*cc+1/r2+1/r1b1 ;
siki[ 2, 3] := -1/r2 ;
siki[ 2, 10] := -1/r1b1 ;
siki[ 3, 3] := 1 ;
e[3] := +e2 ;
siki[ 4, 4] := +s*ce+1/re+1/rbb1 ;
siki[ 4, 7] := -1/rbb1 ;
siki[ 5, 3] := -1/rc ;
siki[ 5, 5] := +1/rc+s*cc+1/rbb1 ;
siki[ 5, 6] := -s*cc ;
siki[ 5, 9] := -1/rbb1 ;
siki[ 6, 5] := -s*cc ;
siki[ 6, 6] := +1/r1+s*cc ;
siki[ 7, 4] := -1/rbb1 ;
siki[ 7, 7] := +1/rbb1+1/reb1+kb1*1/reb1 ;
siki[ 7, 8] := -1/reb1-kb1*1/reb1 ;
siki[ 8, 2] := -1/r1b1 ;
siki[ 8, 7] := -1/reb1 ;
siki[ 8, 8] := +1/reb1 ;
siki[ 8, 10] := +1/r1b1 ;
siki[ 9, 5] := -1/rbb1 ;
siki[ 9, 7] := -kb1*1/reb1 ;
siki[ 9, 8] := +kb1*1/reb1 ;
siki[ 9, 9] := +1/rbb1 ;
siki[ 10, 8] := -1 ;
siki[ 10, 10] := 1 ;
e[10] := +ebb1 ;
x:=linsolve(siki,e);
e1 := 1 ;
r1 := 10000 ;
re := 2200 ;
ce := 0.001 ;
r1 := 1000 ;
e2 := 24 ;

```

各コマンドの詳細説明と操作方法

```

cc := 0.001 ;
r2 := 50000 ;
rc := 3800 ;
r1b1 := 0.1 ;
rbb1 := 0.1 ;
reb1 := 26 ;
kb1 := 100 ;
ebb1 := 0.6 ;
;end;

```

変換終了

使用した式の個数 30、 密度 30.000 %

使用した式のサイズ 463

/dpart による接続情報

```

? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0          ]
left[ 0] right[ 2] parts[ r1          ] value[ 1e+004 j 0          ]
left[ 0] right[ 3] parts[ e2          ] value[      24 j 0          ]
left[ 0] right[ 4] parts[ ce          ] value[ 0.001 j 0          ]
left[ 0] right[ 4] parts[ re          ] value[ 2200 j 0          ]
left[ 0] right[ 6] parts[ r1          ] value[ 1000 j 0          ]
left[ 1] right[ 2] parts[ cc          ] value[ 0.001 j 0          ]
left[ 2] right[ 3] parts[ r2          ] value[ 5e+004 j 0          ]
left[ 2] right[10] parts[ r1b1        ] value[ 0.1 j 0          ]
left[ 3] right[ 5] parts[ rc          ] value[ 3800 j 0          ]
left[ 4] right[ 7] parts[ rbb1        ] value[ 0.1 j 0          ]
left[ 5] right[ 6] parts[ cc          ] value[ 0.001 j 0          ]
left[ 5] right[ 9] parts[ rbb1        ] value[ 0.1 j 0          ]
left[ 7] right[ 8] parts[ reb1        ] value[ 26 j 0          ]
left[ 7] right[ 9] parts[ kb1         ] value[ 100 j 0          ]
@ master[reb1      ] left[ 7] right[ 8]
left[ 8] right[10] parts[ ebb1        ] value[ 0.6 j 0          ]
最大の 素子番号 = 16
最大のノード番号 = 10

```

(8) /cpart (clear parts の略)

回路の接続情報をクリアする。

(9) /db (decibel の略)

デシベルで計算するモードにする。

(10) /def (define functions の略)

自作の関数を定義する。

各コマンドの詳細説明と操作方法

```
? /def
関数定義: 関数名を英数字 15 文字以内で入力して下さい ? a2b2
fnc(a2b2) = a*a+b*b ... 関数を入力して下さい。リターンならこのままです
a*a+b*b
fnc(a2b2) = a*a+b*b
? a=2
      2 j 0

? b=3
      3 j 0

? $a2b2
     13 j 0
```

(11) /dfnc (display functions の略)

定義された関数名とその内容を表示する。

```
? /dfnc
No.  0 a          b*3
No.  1 a2b2       a*a+b*b
```

自作の関数は、/save で保存しておく、/load で読み込むことができます。

/cal で数式を用いてシミュレーションの結果を表示する時には、sim.exe に登録されている関数に加えて、自作の関数も利用できます。

(12) /dir (directory の略)

／load, ／bload, @マクロファイル名のコマンドにおいて、
．cir 及び ．sb のファイルが存在するディフォルトディレクトリを
設定、変更する。

注意：各種コマンドでは、ファイル名などを入力すると小文字に変換されます。漢字を使用すると文字化けが生じます！英数字を入力してください。

(13) /disp (display nodes の略)

電圧を計算して表示するノードを指定する

表示ノード番号 (0 なら全て) ? と表示されるので、

／ipart, ／padd の最後で行なったように、表示ノード番号を入力する。

各コマンドの詳細説明と操作方法

```
? /disp
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.6 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
1 2 3 4 5 6 7 8 9 10

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 6
ノード番号 (0 なら終了) ?
? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x6 [ -452.5992 j -8.7920 ] abs[ 452.6846] arg[ -178.8871]
```

(14) /dpart (display parts の略)

回路の接続情報を表示する。

(15) /dvar (display variables の略)

sim システムに記憶されているすべての変数を表示する。

```
? /dvar
No. 0 f 0 0
No. 1 fl 0 0
No. 2 fh 0 0
No. 3 fs 0 0
No. 4 s 0 0
No. 5 pi 3.1416 0
No. 6 r1b 357.4800 0
No. 7 b1 0 0
No. 8 r1a 81.215 0
No. 9 c 1.674000e-008 0
No. 10 r2 280.968 0
No. 11 v1b1 1,000,000 0
No. 12 r1b1 1,000,000 0
No. 13 b 3 0
No. 14 a 2 0
```

(16) /end

sim.exe を終了する。

各コマンドの詳細説明と操作方法

(17) /fname (file name の略)

現在読み込んでいる接続情報ファイル名を表示する。

```
? /load
読み込み
ファイル名は ? bpf11
@ func @bpf11:0,1,4
node 4
独立電圧源素子の個数 0
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

表示ノード番号 (0 なら全て、-1 ならこのまま) ?
未解凍のブロックがあります。 解凍しますか (y/n) y
bname b1 sfnc ic1:0,3,0,4

func ic1:0,3,0,4 name ic1.cir blk b1
@ func ic1:0,1,2,3

従属電圧源素子の接続ノード
i[ 4] j[ 0] 部品名[v1b1] 値[1e+006 j 0]
独立電圧源素子の個数 0
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
1 2 3 4

表示ノード番号 (0 なら全て、-1 ならこのまま) ?

使用した式の個数 0、密度 0.000 %
使用した式のサイズ 202
最大の 素子番号 = 7
最大のノード番号 = 4
最大の 素子番号 = 7
最大のノード番号 = 4
? /fname
現在読み込んでいるファイル名は = bpf11.cir
```

(18) /free

未定義

(19) /gpos (give positions の略)

上記シミュレーションコマンド/range, /para, /ac でステップ値を等比級数で増加させる時に最低値から最高値までいくつのポイントで行なうかを設定する。
デフォルトは3 2、最低のポイント数は5、最高のポイント数は3 0 0である。

(20) /grph

このコマンドは削除したので、使えない。

周波数特性などのグラフを表示する。

/range, /para, /ac で作成した周波数特性などの計算ファイルが必要。

周波数など横軸の最高値／最低値の比が1 0 倍を超える時には、自動的に

各コマンドの詳細説明と操作方法

横軸を対数目盛りに変更して分かり易いグラフ表示を行なう。

データファイル名は `?` と表示されるので、データファイル名を
拡張子を付けずに入力する。`rc` と入力すると `rc.d` がオープンされる。

最低周波数 `0` 最高周波数 `1000` のように、周波数範囲が表示される。
出力値 最低値 `-16.07` 最高値 `0` のように、出力値の範囲が表示される。

基準とする出力値は `?` と表示されるので、
希望の出力値を入力する。例えば、`0`。

縦方向の位置は `(0から399)` `?` と表示されるので、
上で入力した出力値が画面でどの高さに表示したいかを入力する。
一番上の位置が `0` で、一番下の位置が `399` になる。例えば、`0`。

画面の縦方向全体に対する出力の振幅は `?` と表示されるので、
画面の上から下までに対応する出力値の変化幅を入力する。例えば、`20`。

ハードコピーを取りますか `?` (`y/n`) では、`y` か `n`
(現在のプログラムでは、ドライバが古すぎて印刷は出来ないと思われる)
再度違う条件で表示しますか `?` (`y/n`) でも、`y` か `n`
別のデータファイルでグラフを表示しますか `?` (`y/n`) で、
`n` を入力すると `sim.exe` のプロンプトに戻る。

(21) /help

コマンドなどの一覧表示

(22) /ipart (initialize parts の略)

メモリ上のネットリストをクリアして、新規に回路データの入力を行なう。計算できる最大の部品数はブロック素子の解凍後で 3000、ノード数は 1500 までとなっている。

注： ver. 1.4 までは最大の素子数は 300 と最大ノード数は 100 だった。

各コマンドの詳細説明と操作方法

```

B? /ipart

left[ 0] right ? 1
素子名は ? e1
値は ? 1

left[ 0] right ? 3
素子名は ? re
値は ? 1k

left[ 0] right ? 6
素子名は ? rl
値は ? 10k

left[ 0] right ? 5
素子名は ? e2
値は ? 12

left[ 0] right ? /

left[ 1] right ? 2
素子名は ? cc
値は ? 100u

left[ 1] right ? /

left[ 2] right ? 5
素子名は ? rb
値は ? 100k

left[ 2] right ? 4
素子名は ? bl
ファイル名と接続ノードを入力して下さい(ic1:0, 1, 2 の様に)
trn:2, 3, 4

left[ 2] right ? /

left[ 3] right ? /

left[ 4] right ? 5
素子名は ? rc
値は ? 10k

left[ 4] right ? 6
素子名は ? cc
素子名 cc は既に使用されています。
同じ素子名を使用しますか? (y/n) y

left[ 4] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 5] j[ 0] 部品名[e2] 値[12 j 0]
独立電圧源素子の個数 2
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

表示ノード番号 (0 なら全て、-1 ならこのまま) ?
未解凍のブロックがあります。 解凍しますか (y/n)

```

left [0] right ? と表示されるので、ノード0と接続されている素子で最小のノード番号を入力する。 通常は1から入力する。

素子名は ? と表示されるので、上で指定したノード間に接続される素子名を入力する。

各コマンドの詳細説明と操作方法

素子名は素子属性記号に続けて英数字を書く。(e 1, c 2, b 3のように)

既に、部品リストに登録されている素子名を入力すると、同じ素子名を使用するかどうかを確認してくるので、yまたはnで決定する。

素子属性がブロック (bで始まる素子) の時には、ブロック名と接続ノードを聞いてくるので、 b p f 1 1 : 0, 1, 2 の様に入力する。

素子属性が従属素子 (k, v, q, wで始まる素子) の時には、マスター素子名を聞いてくる。入力したマスター素子名が既に他の従属素子で使用中の場合には違うマスター素子名の入力を求めてくる。マスター素子の属性はr, l, cである。値は ? と表示されるので、素子の値を入力する。

(1 0 0 0, 1 e 3, 1 e - 6などのように)

再び

l e f t [0] r i g h t ? と表示されるが、ノード0に接続される素子がない時には / を入力すると、

l e f t [1] r i g h t ? と左ノード番号が1だけ進んで、右ノード番号の入力になる。上と同様に接続される素子が無い時は / を入力する。

全ての素子を入力したら、r i g h t ? で * を入力する。

最大ノード番号 = 2 のように、入力された最大のノード番号を表示する。

表示ノード番号 (0 なら全て) ? と表示される。

これは、/ r a n g eなどで周波数特性を計算するときに画面に表示するノード番号の入力を要求している場面である。例えば、2と入力すると、ノード番号 (0 なら終了) ? と表示される。

これは、他にも画面に計算結果を表示したいノードがあればノード番号を入力する様に要求している場面である。他になければ、0を入力する。

各コマンドの詳細説明と操作方法

(23) /load

回路情報ファイルの読み込み

ファイル名は ? と表示されるのでファイル名を拡張子を付けずに入力する

```
? /load
読み込み
ファイル名は ? test
@ func @test:0,1,2
node 10

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.6 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 5
ノード番号 (0 なら終了) ?
```

注意：各種コマンドでは、ファイル名などを入力すると小文字に変換されます。漢字を使用すると文字化けが生じます！英数字を入力してください。

回路の接続情報に加えて、自作の関数も一緒にロードされます。

sim.exe の関数と自作の関数はすべて、/cal などで利用することができます。

(24) /man

sim マニュアル「sim.pdf」を表示します。

表示を終了すると、sim.exe に戻ることが出来ます。

(25) /manu

シミュレーションコマンド /para, /ac, /range で計算ステップを常に等比級数で増加するように設定し、/gpos で設定されたステップ数で計算する。

/gpos で設定できる最大のポイント数は300である。

最高値／最低値の比が10を超えても越えなくても等比級数で計算する。

ただし、最低値が0の場合は、/autoと同様に、与えられたステップ値で計算する。

最高値／ステップ値 > 300 なら、ステップ値が最高値／300に変更される。

計算が実行される範囲は指定した最低値～最高値以内の範囲となる。

各コマンドの詳細説明と操作方法

(26) /mode

mode の現在の状態を表示する。

```
? /mode
データファイルを作成する..... /mk または /nomk で切り替える
デシベル値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /auto (自動切り替え) です
..... /auto または /manu で切り替える
等比級数のポイント数は 32 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
1 2 3 4 5 6
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する
```

/nomk すると、

```
? /mode
データファイルを作成しない..... /mk または /nomk で切り替える
デシベル値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /auto (自動切り替え) です
..... /auto または /manu で切り替える
等比級数のポイント数は 32 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
1 2 3 4 5 6
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する
```

(27) /node

回路の最大のノード番号を表示する。

(28) /padd (parts addition の略)

現在入力されているネットリストに回路を追加する時に使用する。

現在のネットリストを表示した後、

l e f t [0] r i g h t ? と表示されて、/ i p a r t の素子の接続情報の入力状態になる。

l e f t [0] に接続するものが無ければ、/を入力すると、l e f t [1] に進むので、希望する左ノードまで進んだら、右ノード番号を入力して、素子データを入力する。追加する素子が無くなったら、*を入力すると追加が終了する。

(29) /para (parameter の略)

直流シミュレーションのために f=0 に設定しておきます。指定の素子を希望の範囲で変化させたときの指定ノードの直流電圧を計算する。全ての電圧源は入力された値を使用して計算する。

回路の直流動作点の設定や確認などで利用する。

各コマンドの詳細説明と操作方法

```

B? /para
値を変化させる素子名は ? r1
para r1 最低値 ? 10k
para r1 最高値 ? 50k
para r1 ステップ ? 5k
r1[1e+004 ] f[0 ]
x5 [ 18.3965 j 0 ] abs[ 18.3965] arg[ 0]
r1[1.5e+004 ] f[0 ]
x5 [ 15.9725 j 0 ] abs[ 15.9725] arg[ 0]
r1[2e+004 ] f[0 ]
x5 [ 13.9472 j 0 ] abs[ 13.9472] arg[ 0]
r1[2.5e+004 ] f[0 ]
x5 [ 12.2295 j 0 ] abs[ 12.2295] arg[ 0]
r1[3e+004 ] f[0 ]
x5 [ 10.7544 j 0 ] abs[ 10.7544] arg[ 0]
r1[3.5e+004 ] f[0 ]
x5 [ 9.4739 j 0 ] abs[ 9.4739] arg[ 0]
r1[4e+004 ] f[0 ]
x5 [ 8.3518 j 0 ] abs[ 8.3518] arg[ 0]
r1[4.5e+004 ] f[0 ]
x5 [ 7.3605 j 0 ] abs[ 7.3605] arg[ 0]
r1[5e+004 ] f[0 ]
x5 [ 6.4783 j 0 ] abs[ 6.4783] arg[ 0]
B? キー入力待ち?

```

値を変化させる素子名は ? と表示されるので、変化させたい素子名を入力する。

para 最低値 ? と表示されるので、最低値を入力する。

para 最高値 ? と表示されるので、最高値を入力する。

para ステップ ? と表示されるので、変化ステップ値を入力する。

(最高値 - 最低値) / ステップ の値が 300 以内になる

ように設定する。この値が 300 を超えた場合には、

計算回数が 300 となるように、ステップ値は自動的に決定される。

a /auto モードかつ 最高値 / 最低値 の比が 10 以下の時は、

入力されたステップ値で計算を行なう。

b /manu モードまたは 最高値 / 最低値 の比が 10 を超える時は、

/gpos で設定された計算ポイント数で最低値から最高値まで

等比級数で増加する様にステップ値を自動的に変化させる。

/mk, /val のモードが設定されている時には表示ノードの電圧値の実数部を保存する。これは、/para が主に動作点の確認や決定に利用するために絶対値で表示しても意味がないため。

/para で /mk, /db のモードでは表示ノードの電圧値の絶対値をファイルに保存する。

/mk, /val の場合

各コマンドの詳細説明と操作方法

```
? /para
データファイル名は ? test
値を変化させる素子名は ? r1
para r1 最低値 ? 10 k
para r1 最高値 ? 50k
para r1 ステップ ? 5k
r1[1e+004 ] f[0 ]
x5 [ 18.3965 j 0 ] abs[ 18.3965] arg[ 0]
r1[1.5e+004 ] f[0 ]
x5 [ 15.9725 j 0 ] abs[ 15.9725] arg[ 0]
r1[2e+004 ] f[0 ]
x5 [ 13.9472 j 0 ] abs[ 13.9472] arg[ 0]
r1[2.5e+004 ] f[0 ]
x5 [ 12.2295 j 0 ] abs[ 12.2295] arg[ 0]
r1[3e+004 ] f[0 ]
x5 [ 10.7544 j 0 ] abs[ 10.7544] arg[ 0]
r1[3.5e+004 ] f[0 ]
x5 [ 9.4739 j 0 ] abs[ 9.4739] arg[ 0]
r1[4e+004 ] f[0 ]
x5 [ 8.3518 j 0 ] abs[ 8.3518] arg[ 0]
r1[4.5e+004 ] f[0 ]
x5 [ 7.3605 j 0 ] abs[ 7.3605] arg[ 0]
r1[5e+004 ] f[0 ]
x5 [ 6.4783 j 0 ] abs[ 6.4783] arg[ 0]
?
```

(30) /parts (parts numbers の略)

最大の素子番号、最大のノード番号を表示する。

未解凍のブロックの個数を表示する。

(31) /point

システム変数 f=0 に設定してから、/point と入力する。

表示ノードの直流動作点電圧を表示する。

(32) /range

交流シミュレーション。

指定された入力信号源以外の独立電圧源と独立電流源は仮に 0 に設定され、周波数特性の計算を行う。ファイルを作成するモードの場合には指定されたノードの電圧振幅をファイルにセーブする。入力信号源の振幅が 1 ならゲインを表す。

```
B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1 ] j 0 ]
f[1,000 ]
x5 [ -94.6500 j -1.8267 ] abs[ 94.6677] arg[ -178.8944]
x6 [ -94.6498 j -1.8417 ] abs[ 94.6677] arg[ -178.8852]
```

各コマンドの詳細説明と操作方法

/mk の場合

```
? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1          j 0          ]
f[1,000      ]
x5 [ -94.6500 j -1.8267 ] abs[ 94.6677] arg[ -178.8944]
x6 [ -94.6498 j -1.8417 ] abs[ 94.6677] arg[ -178.8852]
```

a c 入力信号源名は ? と表示されるので、
素子属性が e で入力信号源として計算したい素子名を入力する。この時、他の
独立電圧源及び独立電流源の値は 0 として交流シミュレーションを実行する。
通常は e 1 と入力する。

値は ? と表示されるので、上で指定した電圧源の電圧を入力する。
通常はゲインが分かり易いように 1 と入力する。実際の出力振幅が知りたい
時には入力電圧をそのまま入力する。

最低周波数 ? と表示されるので、最低の周波数値を入力する。
最高周波数 ? と表示されるので、最高の周波数値を入力する。
ステップ ? と表示されるので、周波数の上昇ステップ値を入力する。
(最高周波数 - 最低周波数) / ステップ の値が 300 以内になる
ように設定する。この値が 300 を超えた場合には、
計算回数が 300 となるように、ステップ値は自動的に決定される。
a /auto モードかつ 最高値 / 最低値 の比が 10 以下の時は、
入力されたステップ値で計算を行なう。
b /manu モードまたは 最高値 / 最低値 の比が 10 を超える時は、
/gpos で設定された計算ポイント数で最低値から最高値まで
等比級数で増加する様にステップ値を自動的に変化させる。

周波数特性を計算して、/disp、/padd、/ipart において最後に
指定された表示ノードの電圧値と位相を画面表示する。

*** オプション機能 ***

各コマンドの詳細説明と操作方法

／s m o d e、／c m o d e、／d b、／v a l、／m k 及び／n o m k により
ファイルを作成するかどうか、計算を値で行なうか d b で行なうか
の切り替えが出来る。

／m k によりデータファイルを作るモードに切り替えてある場合には
計算結果をファイルとして保存することができる。このモードの時には、
a c 入力信号源名 ? の前に、保存するためのファイル名の入力
を求められる。(l p f 1, r c のように8キャラクタ
以内で、拡張子を付けずに入力する)

／m o d e により計算モードとファイルモードがどのように設定されているかを
確認できる。

(33) /rewind

バッチファイル実行時に、バッチファイルに記述された／w a i t コマンド
により、キー入力待ちになっている時に／r e w i n d とキー入力すると、
そのバッチファイルの先頭まで戻ることが出来る。

(34) /save

回路情報をファイルにして保存する。

ファイル名は ? と表示されるので、ファイル名を入力する。

**注意：各種コマンドでは、ファイル名などを入力すると小文字に変換されます。漢字を
使用すると文字化けが生じます！英数字を入力してください。**

(lowpass のように拡張子を付けずに入力すると、自動的に lowpass.cir として保存
する)

既に、同名のファイルが登録されている時には、

1.. 重書き、他のキーならファイル名を再入力 ?

と表示されるので、重ね書きするときは1を、他のファイル名に変更するとき
には0などを入力する。

外部ノード (1, 2, 3 の様に)

と表示されるので、他の回路の中で、この回路をブロック素子という部品として
使用するときには接続する必要があるノード番号をカンマで区切って入力する。

各コマンドの詳細説明と操作方法

素子の接続情報

```
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[          1 j 0          ]
left[ 0] right[ 2] parts[ r1          ] value[ 2. 6e+004 j 0          ]
left[ 0] right[ 3] parts[ e2          ] value[          24 j 0          ]
left[ 0] right[ 4] parts[ ce          ] value[         0. 001 j 0          ]
left[ 0] right[ 4] parts[ re          ] value[         2200 j 0          ]
left[ 0] right[ 6] parts[ r1          ] value[        1e+006 j 0          ]
left[ 1] right[ 2] parts[ cc          ] value[         0. 001 j 0          ]
left[ 2] right[ 3] parts[ r2          ] value[        5e+004 j 0          ]
left[ 2] right[10] parts[ r1b1        ] value[          800 j 0          ]
left[ 3] right[ 5] parts[ rc          ] value[         3800 j 0          ]
left[ 4] right[ 7] parts[ rbb1        ] value[          0. 1 j 0          ]
left[ 5] right[ 6] parts[ cc          ] value[         0. 001 j 0          ]
left[ 5] right[ 9] parts[ rbb1        ] value[          0. 1 j 0          ]
left[ 7] right[ 8] parts[ reb1        ] value[          26 j 0          ]
left[ 7] right[ 9] parts[ kb1         ] value[         100 j 0          ]
@ master[reb1          ] left[ 7] right[ 8]
left[ 8] right[10] parts[ ebb1        ] value[          0. 6 j 0          ]
最大の素子番号 = 16
最大のノード番号 = 10
```

回路情報の保存

```
? /save
セーブ
ファイル名は ? test
外部ノード (1, 2, 3 の様に)
0, 1, 2
```

保存された接続情報

```
@test:0, 1, 2
10
0 1 @e @e1 @ 0 1 0 @
0 2 @r @r1 @ 0 26000 0 @
0 3 @e @e2 @ 0 24 0 @
0 4 @c @ce @ 0 0. 001 0 @
0 4 @r @re @ 0 2200 0 @
0 6 @r @r1 @ 0 1e+006 0 @
1 2 @c @cc @ 0 0. 001 0 @
2 3 @r @r2 @ 0 50000 0 @
2 5 @* @b1 @ 1 0 0 @trn:2, 4, 5
2 10 @r @r1b1 @ 0 800 0 @
3 5 @r @rc @ 0 3800 0 @
4 7 @r @rbb1 @ 0 0. 1 0 @
```

各コマンドの詳細説明と操作方法

```

5 6 @c @cc @ 0 0.001 0 @
5 9 @r @rbb1 @ 0 0.1 0 @
7 8 @r @reb1 @ 0 26 0 @
7 9 @k @kb1 @reb1 0 100 0 @
8 10 @e @ebb1 @ 0 0.6 0 @
fnc

```

(35) /skip

バッチファイル実行時に、バッチファイルに記述された `/wait` コマンドにより、キー入力待ちになっている時に `/skip` とキー入力すると、そのバッチファイルを最後までスキップすることが出来る。

(36) /smode (set mode の略)

mode をセットする

```

? /smode
データファイルを作成する..... /mk または /nomk で切り替える
デシベル値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /auto (自動切り替え) です
..... /auto または /manu で切り替える
等比級数のポイント数は 32 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
1 2 3 4 5 6
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する

```

(37) /val (value の略)

値で計算するモードにする。

sim.exe の操作練習

sim.exe の操作練習

まず、sim.exe を起動して、/ipart コマンドによって回路図の入力を練習します。

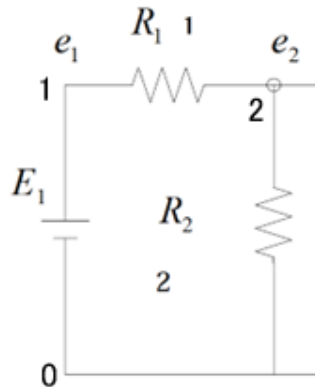


図 ex-1

図で e1, e2 と表示されているのは、対応するノードの電圧を表します。そして、e1 のノード番号は 1、e2 のノード番号は 2、電源 E1 のマイナス側はノード番号 0 とします。

図では、抵抗 R1 の右側に 1 が見えますが、これは抵抗値を表します。回路図の入力時には好きな値を入力して構いません。R2 の値も同様です。

/ipart

? の次から、/ipart 「Enter」と入力します。「各コマンドの詳細説明と操作方法」を参照

```
? /ipart
left[ 0] right ? 1
素子名は ? e1
値は ? 1

left[ 0] right ? 2
素子名は ? r2
値は ? 2

left[ 0] right ? /

left[ 1] right ? 2
素子名は ? r1
値は ? 1

left[ 1] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
独立電圧源素子の個数 1
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

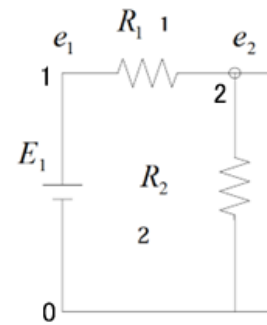
表示ノード番号 (0 なら全て、-1 ならこのまま) ? 2
ノード番号 (0 なら終了) ?
```

「right」で「1」を入力する
「素子名」は「e1」を入力する
「値」は「1」を入力する
「right」に適切な番号を入力
以下同様

入力する素子が無くなったら
「*」を入力する

シミュレーションで確認する
ノード番号を入力する
複数個の指定が可能

sim.exe の操作練習



/dpart

入力した回路データは/dpart によって確認できます。

```
? /dpart
left[ 0] right[ 1] parts[ e1      ] value[      1 j 0      ]
left[ 0] right[ 2] parts[ r2      ] value[      2 j 0      ]
left[ 1] right[ 2] parts[ r1      ] value[      1 j 0      ]
最大の 素子番号 = 3
最大のノード番号 = 2
```

2つの抵抗の値を変更して、確認します。

```
? r1=1k
      1,000 j 0
? r2=2k
      2,000 j 0
? /dpart
left[ 0] right[ 1] parts[ e1      ] value[      1 j 0      ]
left[ 0] right[ 2] parts[ r2      ] value[     2000 j 0      ]
left[ 1] right[ 2] parts[ r1      ] value[     1000 j 0      ]
```

このように、コマンドラインから簡単に素子値を変更することが出来ます。

/point

ノード2の電圧を調べてみます。

E1は直流電源として、f=0に設定してから、/pointを実行します。

```
? f=0
      0 j 0
? /point
f[0      ]
x2 [ 0.6667 j 0 ] abs[ 0.6667] arg[      0]
```

回路図を入力した時に、表示ノード番号を2に指定したのでx2の値が表示される。

sim.exe の操作練習

次に、r1 の値を変化すると e2 がどのように変化するかを調べてみます。

r1 を 10 から 10k まで、5 点の計算を行います。

r1 を変化させる範囲が広いので、計算するポイント数を 5 個に限定して表示画面に収まるように指定します。/gpos コマンドで 5 を入力してから、/manu コマンドを入力します。

r1 の変化の倍率が 1000 なので、 $r = 1000^{\frac{1}{4}} = 5.62341$ の倍率で、10 から 10000 (10k) まで 5 ポイントで増加します。 $r1 = 10 \cdot r^k$, $k=0\ldots4$ の等比級数になります。

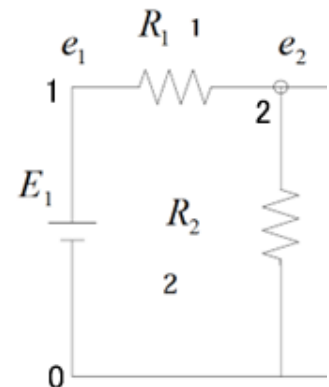
sim.exe を起動直後は、gpos=32 に設定されているので、値を変化させる最大値と最小値の倍率が 10 を超える場合には、/auto でも、等比級数のステップで、自動的に 32 ポイントの計算を行います。

コマンド/manu を実行すると、最大値と最小値の倍率が 10 を超えなくても、等比級数のステップで gpos 回の計算を行います。

/para

次に、/para コマンドを実行します。

```
? /gpos
MANU ON または 最高/最低の比が 10 以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 32 です
5
gpos = 5 に設定しました
? /manu
? /para
値を変化させる素子名は ? r1
para r1 最低値 ? 10
para r1 最高値 ? 10k
para r1 ステップ ? 1k
r1[10] ] f[0] ]
x2 [ 0.9950 j 0 ] abs[ 0.9950] arg[ 0]
r1[56.23] ] f[0] ]
x2 [ 0.9727 j 0 ] abs[ 0.9727] arg[ 0]
r1[316.2] ] f[0] ]
x2 [ 0.8635 j 0 ] abs[ 0.8635] arg[ 0]
r1[1778] ] f[0] ]
x2 [ 0.5293 j 0 ] abs[ 0.5293] arg[ 0]
r1[1e+004] ] f[0] ]
x2 [ 0.1667 j 0 ] abs[ 0.1667] arg[ 0]
```



シミュレーションのコマンドを実行しても、素子の値は元のまま変化しません。

```
? /dpart
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 2] parts[ r2 ] value[ 2000 j 0 ]
left[ 1] right[ 2] parts[ r1 ] value[ 1000 j 0 ]
```

sim.exe の操作練習

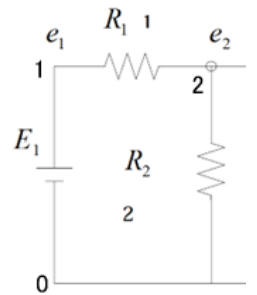
シミュレーションコマンド /para, /ac, /range における計算回数

「各コマンドの詳細説明と操作方法」の「(2) /auto」、「(25) /manu」を参照して下さい。

操作例を示します。

/auto モード、gpos = 5 の場合

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 0
para e1 最高値 ? 1
para e1 ステップ ? 0.25
e1[0]      ] f[0      ]
x2 [      0 j 0      ] abs[      0] arg[      0]
e1[0.25]   ] f[0      ]
x2 [ 0.1667 j 0      ] abs[ 0.1667] arg[      0]
e1[0.5]    ] f[0      ]
x2 [ 0.3333 j 0      ] abs[ 0.3333] arg[      0]
e1[0.75]   ] f[0      ]
x2 [ 0.5000 j 0      ] abs[ 0.5000] arg[      0]
e1[1]      ] f[0      ]
x2 [ 0.6667 j 0      ] abs[ 0.6667] arg[      0]
```



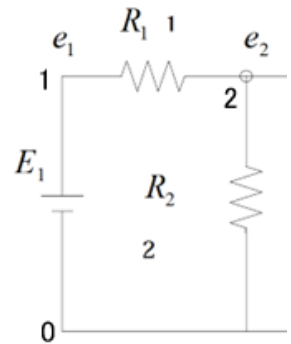
最低値 = 0 で、最高値 / ステップ値 ≤ 300 なので、与えられたステップ値で計算する。

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 0
para e1 最高値 ? 3
para e1 ステップ ? 0.001
e1[0]      ] f[0      ]
x2 [      0 j 0      ] abs[      0] arg[      0]
e1[0.01]   ] f[0      ]
x2 [ 0.0067 j 0      ] abs[ 0.0067] arg[      0]
e1[0.02]   ] f[0      ]
x2 [ 0.0133 j 0      ] abs[ 0.0133] arg[      0]
e1[0.03]   ] f[0      ]
x2 [ 0.0200 j 0      ] abs[ 0.0200] arg[      0]
e1[0.04]   ] f[0      ]
x2 [ 0.0267 j 0      ] abs[ 0.0267] arg[      0]
```

最低値 = 0 で、最高値 / ステップ値 > 300 なので、
ステップ値 = 最高値 / 300 ($\frac{3}{300} = 0.01$) に変更して計算する。

sim.exe の操作練習

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 1
para e1 最高値 ? 5
para e1 ステップ ? 1
e1[1] ] f[0 ] ] abs[ 0.6667] arg[ 0]
x2 [ 0.6667 j 0 ] ]
e1[2] ] f[0 ] ] abs[ 1.3333] arg[ 0]
x2 [ 1.3333 j 0 ] ]
e1[3] ] f[0 ] ] abs[ 2] arg[ 0]
x2 [ 2 j 0 ] ]
e1[4] ] f[0 ] ] abs[ 2.6667] arg[ 0]
x2 [ 2.6667 j 0 ] ]
e1[5] ] f[0 ] ] abs[ 3.3333] arg[ 0]
x2 [ 3.3333 j 0 ] ]
```



最高値／最低値 ≤ 10 かつ (最高値－最低値)／ステップ値 ≤ 300 なので、与えられたステップ値で計算する。

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 1
para e1 最高値 ? 4
para e1 ステップ ? 0.001
e1[1] ] f[0 ] ] abs[ 0.6667] arg[ 0]
x2 [ 0.6667 j 0 ] ]
e1[1.01] ] f[0 ] ] abs[ 0.6733] arg[ 0]
x2 [ 0.6733 j 0 ] ]
e1[1.02] ] f[0 ] ] abs[ 0.6800] arg[ 0]
x2 [ 0.6800 j 0 ] ]
e1[1.03] ] f[0 ] ] abs[ 0.6867] arg[ 0]
x2 [ 0.6867 j 0 ] ]
e1[1.04] ] f[0 ] ] abs[ 0.6933] arg[ 0]
x2 [ 0.6933 j 0 ] ]
```

最高値／最低値 ≤ 10 かつ (最高値－最低値)／ステップ値 > 300 なので、

ステップ値 = (最高値－最低値)／300 ($\frac{4-1}{300} = 0.01$) に変更して計算する。

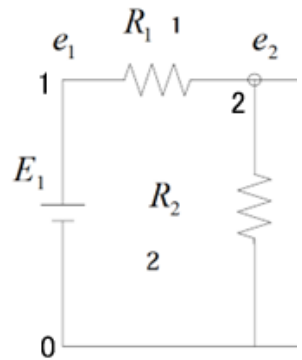
```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 1
para e1 最高値 ? 11
para e1 ステップ ? 1
e1[1] ] f[0 ] ] abs[ 0.6667] arg[ 0]
x2 [ 0.6667 j 0 ] ]
e1[1.821] ] f[0 ] ] abs[ 1.2141] arg[ 0]
x2 [ 1.2141 j 0 ] ]
e1[3.317] ] f[0 ] ] abs[ 2.2111] arg[ 0]
x2 [ 2.2111 j 0 ] ]
e1[6.04] ] f[0 ] ] abs[ 4.0267] arg[ 0]
x2 [ 4.0267 j 0 ] ]
e1[11] ] f[0 ] ] abs[ 7.3333] arg[ 0]
x2 [ 7.3333 j 0 ] ]
```

最高値／最低値 > 10 なので、gpos=5 のステップ数で等比級数で計算する。

sim.exe の操作練習

/manu モード、gpos = 5 の場合

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 0
para e1 最高値 ? 1
para e1 ステップ ? 0.1
e1[0] ] f[0] ]
x2 [ 0 j 0 ] abs[ 0] arg[ 0]
e1[0.1] ] f[0] ]
x2 [ 0.0667 j 0 ] abs[ 0.0667] arg[ 0]
e1[0.2] ] f[0] ]
x2 [ 0.1333 j 0 ] abs[ 0.1333] arg[ 0]
e1[0.3] ] f[0] ]
x2 [ 0.2000 j 0 ] abs[ 0.2000] arg[ 0]
e1[0.4] ] f[0] ]
x2 [ 0.2667 j 0 ] abs[ 0.2667] arg[ 0]
```



最低値 = 0 の場合は、/auto の場合と同様に、与えられたステップ値で計算する。

最高値 / ステップ値 > 300 の場合も /auto の場合と同様に動作する。

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 1
para e1 最高値 ? 10
para e1 ステップ ? 1
e1[1] ] f[0] ]
x2 [ 0.6667 j 0 ] abs[ 0.6667] arg[ 0]
e1[1.778] ] f[0] ]
x2 [ 1.1855 j 0 ] abs[ 1.1855] arg[ 0]
e1[3.162] ] f[0] ]
x2 [ 2.1082 j 0 ] abs[ 2.1082] arg[ 0]
e1[5.623] ] f[0] ]
x2 [ 3.7489 j 0 ] abs[ 3.7489] arg[ 0]
e1[10] ] f[0] ]
x2 [ 6.6667 j 0 ] abs[ 6.6667] arg[ 0]
```

最低値が 0 でなく /manu モードなので、

最高値 / 最低値の比に関係なく、gpos=5 のステップ数で等比級数で計算する。

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 1
para e1 最高値 ? 11
para e1 ステップ ? 1
e1[1] ] f[0] ]
x2 [ 0.6667 j 0 ] abs[ 0.6667] arg[ 0]
e1[1.821] ] f[0] ]
x2 [ 1.2141 j 0 ] abs[ 1.2141] arg[ 0]
e1[3.317] ] f[0] ]
x2 [ 2.2111 j 0 ] abs[ 2.2111] arg[ 0]
e1[6.04] ] f[0] ]
x2 [ 4.0267 j 0 ] abs[ 4.0267] arg[ 0]
e1[11] ] f[0] ]
x2 [ 7.3333 j 0 ] abs[ 7.3333] arg[ 0]
```

最高値 / 最低値の比に関係なく、gpos=5 のステップ数で等比級数で計算する。

sim.exe の操作練習

計算回数の実測値

mode	最低 Lo	最高 Hi	ステップ st	gpos	L0=0?	Hi/Lo>10? または Lo/Hi>10?	Hi/st>300? または (Hi-Lo)/st > 300?	等比級数?	修正 st	計算回数
auto	0	1	0.25	5	○					5
auto	0	3	0.001	5	○		○		0.01	301
auto	1	5	1	5						5
auto	1	4	0.001	5			○		0.01	301
auto	1	11	1	5		○		○		gpos
auto	1	5	2	5						3
auto	1	5	3	5						2
auto	1	5	4	5						2
auto	1	5	5	5						1
auto	-11	-1	1	5		○		○		gpos
auto	-11	2	1	5						14
auto	-11	0	0.001	5			○		0.03667	301
manu	0	1	0.1	5	○					11
manu	1	10	1	5				○		gpos
manu	1	11	1	5		○		○		gpos
manu	1	5	2	5				○		gpos
manu	0	1	0.001	5	○		○		0.01	301
manu	-11	-1	1	5		○		○		gpos
manu	-11	2	1	5						14
manu	-11	0	0.001	5			○		0.03667	301

ステップ値を利用する場合、ステップ値が大きいときには、（黄色で示したように）最高値に達する前に終了する。

最低値と最高値がどちらも負の場合には、 $Lo/Hi > 10$ なら等比級数を利用する。

最低値が負で、最高値がゼロまたは正ならば、与えられたステップ値を利用する。

しかし、 $(Hi - Lo)/st > 300$ の時は、 $st = (Hi - Lo)/st$ に修正する。

sim.exe の操作練習

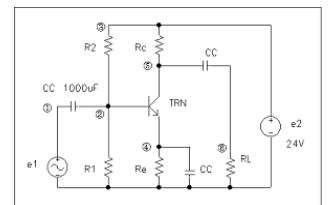
シミュレーションコマンドを使う場合には、/auto か/manu の選択、/val か/db の選択、/mk か/nomk の選択、/cal の設定、/gpos の設定、最高値・最低値・ステップ値等を目的に応じて適切に設定すると、短時間で目的の結果を確認できるようになります。

例えば、オーディオ用の回路を作成してその周波数特性を確認するなら、周波数は 10 Hz から 50 KHz の範囲のゲインをデシベル値で計算してファイルに出力し、その数値を横軸が対数目盛のグラフにして確認すると分かりやすいと思います。

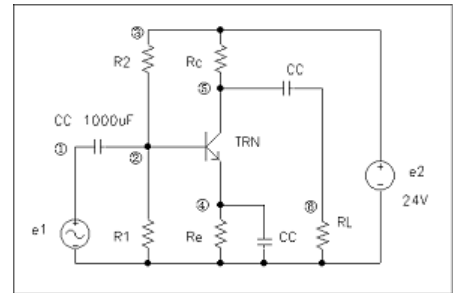
```
? /gpos
MANU ON または 最高/最低の比が 10 以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 24 です
12
gpos = 12 に設定しました
? /db
mode = デシベル値で計算するモード
? /range
データファイル名は ? amptest
ac 入力信号源名は ? e1
値は ? 1
range 周波数 最低値 ? 10
range 周波数 最高値 ? 50k
range 周波数 ステップ ? 100
```

例えば、「sim.exe の操作練習」の「ex4.sb」を実行して、/para コマンドによって Rc を変更してトランジスタのコレクタの動作点を 12V に調整したい場合には、/gpos でポイント数を 5 程度に少なく設定して、抵抗の範囲は広めに指定して、/manu で等比級数で何度か範囲を狭めて実行してから、/auto に戻して、狭い範囲を細かいステップで計算するのが良いと思います。

```
? /manu
? /gpos
MANU ON または 最高/最低の比が 10 以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 5 です
5
gpos = 5 に設定しました
? /para
値を変化させる素子名は ? rc
para rc 最低値 ? 1k
para rc 最高値 ? 50k
para rc ステップ ? 100
rc[1000] ] f[0] ]
x5 [ 22.6348 j 0 ] abs[ 22.6348] arg[ 0]
rc[2659] ] f[0] ]
x5 [ 20.3698 j 0 ] abs[ 20.3698] arg[ 0]
rc[7071] ] f[0] ]
x5 [ 14.3468 j 0 ] abs[ 14.3468] arg[ 0]
rc[1.88e+004] ] f[0] ]
x5 [ -1.6692 j 0 ] abs[ 1.6692] arg[ 180]
rc[5e+004] ] f[0] ]
x5 [ -44.2583 j 0 ] abs[ 44.2583] arg[ 180]
```



sim.exe の操作練習



```
rc[7071      ] f[0      ]
x5 [ 14.3468 j 0      ] abs[ 14.3468] arg[      0]
rc[1.88e+004 ] f[0      ]
x5 [ -1.6692 j 0      ] abs[ 1.6692] arg[    180]
```

Rc=7071 と 18.8K の間で、x5=12 になることが分かりました。

この範囲を/gpos でポイント数を 32 に増加して調べます。

```
? /gpos
MANU ON または 最高/最低の比が 10 以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 5 です
32
gpos = 32 に設定しました
? /para
値を変化させる素子名は ? rc
para rc 最低値 ? 7071
para rc 最高値 ? 18.8k
para rc ステップ ? 0.1
```

```
rc[8544      ] f[0      ]
x5 [ 12.3356 j 0      ] abs[ 12.3356] arg[      0]
rc[8818      ] f[0      ]
x5 [ 11.9618 j 0      ] abs[ 11.9618] arg[      0]
```

Rc=8544 と 8818 の間で、x5=12 になることが分かりました。

この範囲をポイント数 32 のままで調べます。

```
? /para
値を変化させる素子名は ? rc
para rc 最低値 ? 8544
para rc 最高値 ? 8818
para rc ステップ ? 0.1
```

```
rc[8782      ] f[0      ]
x5 [ 12.0109 j 0      ] abs[ 12.0109] arg[      0]
rc[8791      ] f[0      ]
x5 [ 11.9987 j 0      ] abs[ 11.9987] arg[      0]
```

Rc=8782 と 8791 の間で、x5=12 になることが分かりました。

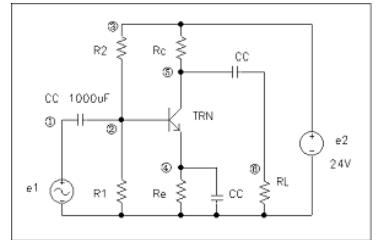
この範囲を/gpos でポイント数 10 に変更して調べます。

sim.exe の操作練習

```

? /para
値を変化させる素子名は ? rc
para rc 最低値 ? 8782
para rc 最高値 ? 8791
para rc ステップ ? 0.01
rc[8782] ] f[0] ] abs[ 12.0111] arg[ 0]
x5 [ 12.0111 j 0 ]
rc[8783] ] f[0] ] abs[ 12.0097] arg[ 0]
x5 [ 12.0097 j 0 ]
rc[8784] ] f[0] ] abs[ 12.0084] arg[ 0]
x5 [ 12.0084 j 0 ]
rc[8785] ] f[0] ] abs[ 12.0070] arg[ 0]
x5 [ 12.0070 j 0 ]
rc[8786] ] f[0] ] abs[ 12.0057] arg[ 0]
x5 [ 12.0057 j 0 ]
rc[8787] ] f[0] ] abs[ 12.0043] arg[ 0]
x5 [ 12.0043 j 0 ]
rc[8788] ] f[0] ] abs[ 12.0029] arg[ 0]
x5 [ 12.0029 j 0 ]
rc[8789] ] f[0] ] abs[ 12.0016] arg[ 0]
x5 [ 12.0016 j 0 ]
rc[8790] ] f[0] ] abs[ 12.0002] arg[ 0]
x5 [ 12.0002 j 0 ]
rc[8791] ] f[0] ] abs[ 11.9988] arg[ 0]
x5 [ 11.9988 j 0 ]

```



Rc=8790 と 8791 の間で、x5=12 になることが分かりました。

/auto に戻して、この範囲を/gpos でポイント数 10 に変更して調べます。

```

? /auto
? /para
値を変化させる素子名は ? rc
para rc 最低値 ? 8790
para rc 最高値 ? 8791
para rc ステップ ? 0.1
rc[8790] ] f[0] ] abs[ 12.0002] arg[ 0]
x5 [ 12.0002 j 0 ]
rc[8790] ] f[0] ] abs[ 12.0001] arg[ 0]
x5 [ 12.0001 j 0 ]
rc[8790] ] f[0] ] abs[ 11.9999] arg[ 0]
x5 [ 11.9999 j 0 ]
rc[8790] ] f[0] ] abs[ 11.9998] arg[ 0]
x5 [ 11.9998 j 0 ]
rc[8790] ] f[0] ] abs[ 11.9996] arg[ 0]
x5 [ 11.9996 j 0 ]
rc[8791] ] f[0] ] abs[ 11.9995] arg[ 0]
x5 [ 11.9995 j 0 ]
rc[8791] ] f[0] ] abs[ 11.9994] arg[ 0]
x5 [ 11.9994 j 0 ]
rc[8791] ] f[0] ] abs[ 11.9992] arg[ 0]
x5 [ 11.9992 j 0 ]
rc[8791] ] f[0] ] abs[ 11.9991] arg[ 0]
x5 [ 11.9991 j 0 ]
rc[8791] ] f[0] ] abs[ 11.9990] arg[ 0]
x5 [ 11.9990 j 0 ]

```

Rc の小数点以下が表示されないので、Rc の変化が分かりません。

sim.exe の操作練習

/cal を使って、Rc と x5 の 1000 倍の値を表示して小数点以下を確認します。

Rc=8790 から 8790.2 までの範囲を調べます。

```
? /cal
シミュレーションコマンドの計算は /disp で設定した式を計算するモードです
従来通りノード電圧のみで行なうなら 0
/disp で設定した式を計算するなら 1 1

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.7 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0

シミュレーションにおける 計算式設定状況

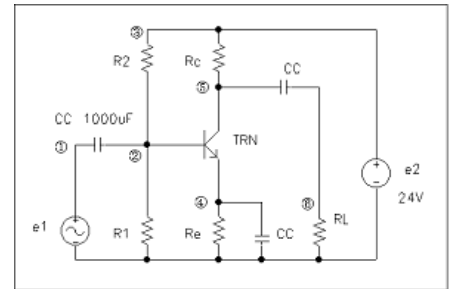
計算式 (0 なら新規設定、-1 なら設定無し) ? 0

ノード電圧は x1, x2 の様に入力して下さい。
因子として定義されているすべての関数が使用出来ます。
1 番目の計算式を入力して下さい
rc*1000

ノード電圧は x1, x2 の様に入力して下さい。
因子として定義されているすべての関数が使用出来ます。
2 番目の計算式を入力して下さい
x5*1000

ノード電圧は x1, x2 の様に入力して下さい。
因子として定義されているすべての関数が使用出来ます。
3 番目の計算式を入力して下さい

? /para
値を変化させる素子名は ? rc
para rc 最低値 ? 8790
para rc 最高値 ? 8790.2
para rc ステップ ? 0.01
```



```
rc[8790 ] f[0 ]
rc*1000
p1 [ 8,790,130 j 0 ] abs[ 8,790,130] arg[ 0]
x5*1000
p2 [12,000.0145 j 0 ] abs[12,000.0145] arg[ 0]
rc[8790 ] f[0 ]
rc*1000
p1 [ 8,790,140 j 0 ] abs[ 8,790,140] arg[ 0]
x5*1000
p2 [12,000.0008 j 0 ] abs[12,000.0008] arg[ 0]
rc[8790 ] f[0 ]
rc*1000
p1 [ 8,790,150 j 0 ] abs[ 8,790,150] arg[ 0]
x5*1000
p2 [11,999.9872 j 0 ] abs[11,999.9872] arg[ 0]
```

実際の Rc と x5 の値は p1 と p2 の数値を 1000 分の 1 にすれば良いです。

Rc=8790.14 で、x5=12.0000008 になることが分かります。

このように、必要に応じて/cal を利用してノードの電圧や素子値の小数点以下の数値を確認することが出来ます。

sim.exe の操作練習

次は、回路に部品を追加してみます。コマンド/padd を使用します。

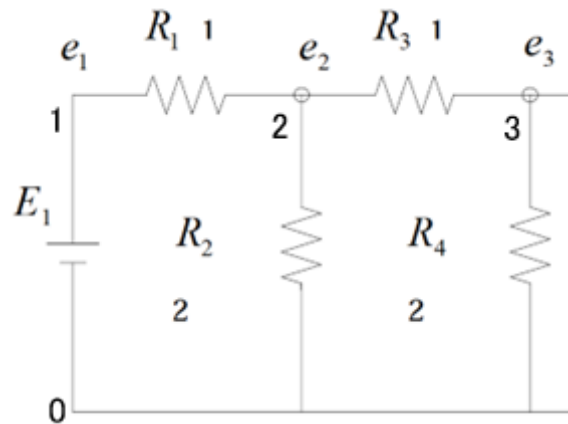


図 ex-2

/padd

コマンドを入力すると、現在の回路情報が表示されて、次に左ノード 0 に対する追加素子のノード番号の入力が求められます。

```
? /padd
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0          ]
left[ 0] right[ 2] parts[ r2          ] value[    2000 j 0          ]
left[ 1] right[ 2] parts[ r1          ] value[    1000 j 0          ]
最大の 素子番号 = 3
最大のノード番号 = 2
```

```
left[ 0] right ? 3
素子名は ? r4
値は ? 2k

left[ 0] right ? /
left[ 1] right ? /
left[ 2] right ? 3
素子名は ? r3
値は ? 1k

left[ 2] right ? *
```

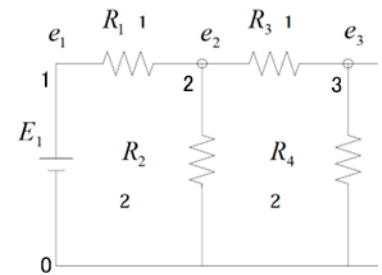
左ノード番号を次に進めるには、/を入力します。

追加する素子が無くなったら、*を入力します。

sim.exe の操作練習

素子の追加を終了すると、表示ノードの指定になります。

```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
独立電圧源素子の個数 1
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
2
表示ノード番号 (0 なら全て、-1 ならこのまま) ? 0
```



/disp

表示ノード番号を変更します。

ノード 2 と 3 を指定します。

```
? /disp

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
独立電圧源素子の個数 1
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
1 2 3
表示ノード番号 (0 なら全て、-1 ならこのまま) ? 2
ノード番号 (0 なら終了) ? 3
ノード番号 (0 なら終了) ?
```

/mode, /cmode, /smode

sim.exe の現在設定されているモードを確認するには、/mode を使います。

```
? /mode
データファイルを作成しない..... /mk または /nomk で切り替える
値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /manu (等比級数) です
..... /auto または /manu で切り替える
等比級数のポイント数は 5 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
2 3
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する
```


sim.exe の操作練習

モードをクリアするには、/cmode を使用します。

```
? /cmode
データファイルを作成しない..... /mk または /nomk で切り替える
値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /auto (自動切り替え) です
..... /auto または /manu で切り替える
等比級数のポイント数は 32 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
2 3
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する
```

モードをセットするには、/smode を使います。

```
? /smode
データファイルを作成する..... /mk または /nomk で切り替える
デシベル値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /auto (自動切り替え) です
..... /auto または /manu で切り替える
等比級数のポイント数は 32 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
2 3
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する
```

/mk, /nomk, /db, /val, /auto, /manu, /gpos, /bmode, /cal, /disp

などのコマンドを利用して、個別のモードだけを切り替えます。

ここでは、/val モードに設定します。

部品を追加した回路情報を確認します。

```
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0      ]
left[ 0] right[ 2] parts[ r2          ] value[    2000 j 0      ]
left[ 0] right[ 3] parts[ r4          ] value[    2000 j 0      ]
left[ 1] right[ 2] parts[ r1          ] value[    1000 j 0      ]
left[ 2] right[ 3] parts[ r3          ] value[    1000 j 0      ]
```

r1 を 1k から 10k まで変化させると、e2 と e3 がどのように変化するかを確認します。

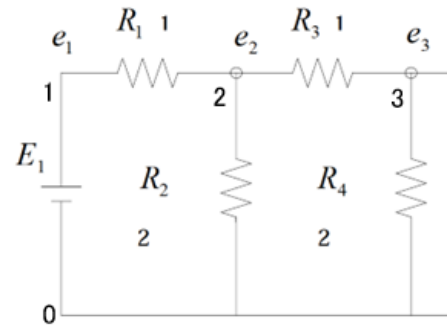
```
? /auto
? /gpos
MANU ON または 最高/最低の比が 10 以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 32 です
10
```

sim.exe の操作練習

```

? /para
データファイル名は ? test
値を変化させる素子名は ? r1
para r1 最低値 ? 1k
para r1 最高値 ? 10k
para r1 ステップ ? 1k
r1[1000] f[0] ]
x2 [ 0.5455 j 0 ] abs[ 0.5455] arg[ 0]
x3 [ 0.3636 j 0 ] abs[ 0.3636] arg[ 0]
r1[2000] f[0] ]
x2 [ 0.3750 j 0 ] abs[ 0.3750] arg[ 0]
x3 [ 0.2500 j 0 ] abs[ 0.2500] arg[ 0]
r1[3000] f[0] ]
x2 [ 0.2857 j 0 ] abs[ 0.2857] arg[ 0]
x3 [ 0.1905 j 0 ] abs[ 0.1905] arg[ 0]
r1[4000] f[0] ]
x2 [ 0.2308 j 0 ] abs[ 0.2308] arg[ 0]
x3 [ 0.1538 j 0 ] abs[ 0.1538] arg[ 0]
r1[5000] f[0] ]
x2 [ 0.1935 j 0 ] abs[ 0.1935] arg[ 0]
x3 [ 0.1290 j 0 ] abs[ 0.1290] arg[ 0]
r1[6000] f[0] ]
x2 [ 0.1667 j 0 ] abs[ 0.1667] arg[ 0]
x3 [ 0.1111 j 0 ] abs[ 0.1111] arg[ 0]
r1[7000] f[0] ]
x2 [ 0.1463 j 0 ] abs[ 0.1463] arg[ 0]
x3 [ 0.0976 j 0 ] abs[ 0.0976] arg[ 0]
r1[8000] f[0] ]
x2 [ 0.1304 j 0 ] abs[ 0.1304] arg[ 0]
x3 [ 0.0870 j 0 ] abs[ 0.0870] arg[ 0]
r1[9000] f[0] ]
x2 [ 0.1176 j 0 ] abs[ 0.1176] arg[ 0]
x3 [ 0.0784 j 0 ] abs[ 0.0784] arg[ 0]
r1[1e+004] f[0] ]
x2 [ 0.1071 j 0 ] abs[ 0.1071] arg[ 0]
x3 [ 0.0714 j 0 ] abs[ 0.0714] arg[ 0]

```



/mk のモードだったので、計算結果が test.d に保存されています。
 計算モードが /val だったので、数値（電圧値）で記録されます。
 /db ならデシベル値で記録されます。

test.d の内容

2 10	2 は表示ノードの個数、10 は計算するポイント数
1000	値を変化させる素子の値
2 0.545455	2 はノード番号、ノード 2 の計算値
3 0.363636	3 はノード番号、ノード 3 の計算値
0 0	表示ノードの終わり
2000	以下、同様に繰り返す
2 0.375	
3 0.25	
0 0	
3000	
2 0.285714	
3 0.190476	
0 0	
4000	

sim.exe の操作練習

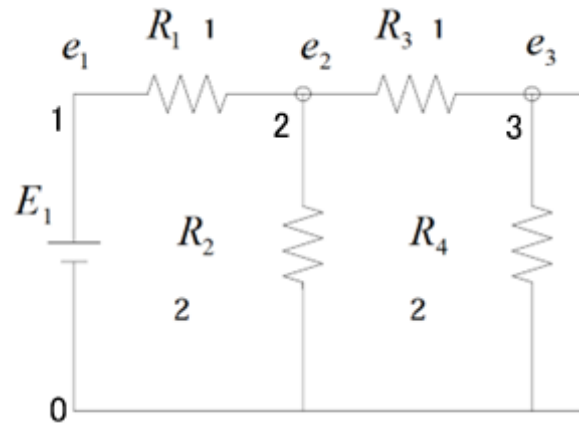
```
2 0.230769
3 0.153846
0 0
5000
2 0.193548
3 0.129032
0 0
6000
2 0.166667
3 0.111111
0 0
7000
2 0.146341
3 0.097561
0 0
8000
2 0.130435
3 0.0869565
0 0
9000
2 0.117647
3 0.0784314
0 0
10000
2 0.107143
3 0.0714286
0 0
```

sim.exe の操作練習

ex2.sb の係数行列 ex2.coe を保存して、Maple で回路方程式を解いてみます。

```
? /conv
係数行列の出力先を選択して下さい
ファイル(conv.coe) F , 画面 C f

? !ren conv.coe ex2.coe
```



ex2.coe の内容

```
restart: with(linalg):
siki:=matrix(3,3,0):e:=vector(3,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 1] := -1/r1 ;
siki[ 2, 2] := +1/r2+1/r1+1/r3 ;
siki[ 2, 3] := -1/r3 ;
siki[ 3, 2] := -1/r3 ;
siki[ 3, 3] := +1/r4+1/r3 ;
x:=linsolve(siki,e);

e1 := 1 ;
r2 := 2 ;
r1 := 1 ;
r4 := 2000 ;
r3 := 1000 ;
```

```
;end;
```

変換終了

使用した式の個数 7、 密度 77.778 %

使用した式のサイズ 100

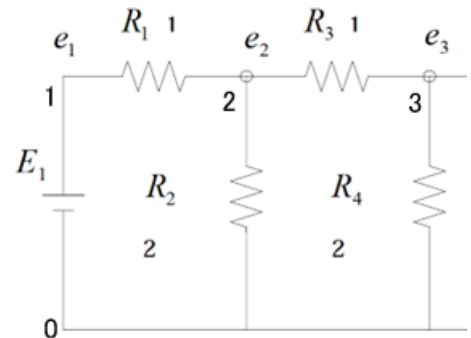
sim.exe の操作練習

Maple にコピーアンドペーストして、方程式を解きます。

```

ワークシート(1)
> restart: with(linalg):
siki:=matrix(3,3,0):e:=vector(3,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 1] := -1/r1 ;
siki[ 2, 2] := +1/r2+1/r1+1/r3 ;
siki[ 2, 3] := -1/r3 ;
siki[ 3, 2] := -1/r3 ;
siki[ 3, 3] := +1/r4+1/r3 ;
x:=linsolve(siki,e);

```



計算を実行すると、

$$\begin{aligned}
 siki_{1,1} &:= 1 \\
 e_1 &:= e1 \\
 siki_{2,1} &:= -\frac{1}{r1} \\
 siki_{2,2} &:= \frac{1}{r2} + \frac{1}{r1} + \frac{1}{r3} \\
 siki_{2,3} &:= -\frac{1}{r3} \\
 siki_{3,2} &:= -\frac{1}{r3} \\
 siki_{3,3} &:= \frac{1}{r4} + \frac{1}{r3} \\
 x &:= \left[e1, \frac{(r3+r4) e1 r2}{r2 r3 + r1 r3 + r2 r1 + r1 r4 + r2 r4}, \frac{r4 e1 r2}{r2 r3 + r1 r3 + r2 r1 + r1 r4 + r2 r4} \right]
 \end{aligned}$$

ノード 2 とノード 3 の値を確認します。

```

> x2:=x[2];x3:=x[3];

```

$$\begin{aligned}
 x2 &:= \frac{(r3+r4) e1 r2}{r2 r3 + r1 r3 + r2 r1 + r1 r4 + r2 r4} \\
 x3 &:= \frac{r4 e1 r2}{r2 r3 + r1 r3 + r2 r1 + r1 r4 + r2 r4}
 \end{aligned}$$

sim.exe は Maple と同様の数式処理をして、最後に素子値を代入して数値で表示します。

sim.exe の操作練習

新しい回路図を入力します。

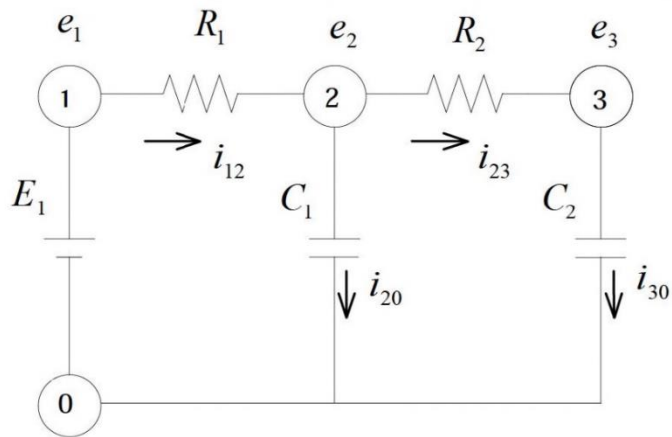


図 ex-3

```
? /ipart
left[ 0] right ? 1
素子名は ? e1
値は ? 1

left[ 0] right ? 2
素子名は ? c1
値は ? 0.01u

left[ 0] right ? 3
素子名は ? c2
値は ? 0.02u

left[ 0] right ? /

left[ 1] right ? 2
素子名は ? r1
値は ? 1k

left[ 1] right ? /

left[ 2] right ? 3
素子名は ? r2
値は ? 1k

left[ 2] right ? *
```

回路情報を確認します。

```
? /dpart
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 2] parts[ c1 ] value[ 1e-008 j 0 ]
left[ 0] right[ 3] parts[ c2 ] value[ 2e-008 j 0 ]
left[ 1] right[ 2] parts[ r1 ] value[ 1000 j 0 ]
left[ 2] right[ 3] parts[ r2 ] value[ 1000 j 0 ]
```

図 ex-3

sim.exe の操作練習

`/range`

この回路では C 素子を使用されているので、入力信号源 E1 の周波数を変化させた場合のノード 2 と 3 の電圧値を確認する。

```
? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 0
range 周波数 最高値 ? 10k
range 周波数 ステップ ? 1k
e1[1 j 0 ]
f[0 ]
x2 [ 1 j 0 ] abs[ 1] arg[ 0]
x3 [ 1 j 0 ] abs[ 1] arg[ 0]
f[1,000 ]
x2 [ 0.9526 j -0.1750 ] abs[ 0.9685] arg[ -10.4084]
x3 [ 0.9161 j -0.2901 ] abs[ 0.9609] arg[ -17.5709]
f[2,000 ]
x2 [ 0.8452 j -0.2889 ] abs[ 0.8932] arg[ -18.8681]
x3 [ 0.7267 j -0.4715 ] abs[ 0.8663] arg[ -32.9759]
f[3,000 ]
x2 [ 0.7334 j -0.3382 ] abs[ 0.8076] arg[ -24.7585]
x3 [ 0.5305 j -0.5382 ] abs[ 0.7557] arg[ -45.4145]
f[4,000 ]
x2 [ 0.6426 j -0.3490 ] abs[ 0.7313] arg[ -28.5047]
x3 [ 0.3730 j -0.5365 ] abs[ 0.6534] arg[ -55.1913]
f[5,000 ]
x2 [ 0.5751 j -0.3428 ] abs[ 0.6695] arg[ -30.7931]
x3 [ 0.2579 j -0.5048 ] abs[ 0.5669] arg[ -62.9350]
f[6,000 ]
x2 [ 0.5257 j -0.3309 ] abs[ 0.6211] arg[ -32.1916]
x3 [ 0.1761 j -0.4637 ] abs[ 0.4960] arg[ -69.2072]
f[7,000 ]
x2 [ 0.4888 j -0.3185 ] abs[ 0.5834] arg[ -33.0852]
x3 [ 0.1176 j -0.4219 ] abs[ 0.4380] arg[ -74.4216]
f[8,000 ]
x2 [ 0.4605 j -0.3073 ] abs[ 0.5536] arg[ -33.7134]
x3 [ 0.0754 j -0.3830 ] abs[ 0.3904] arg[ -78.8651]
f[9,000 ]
x2 [ 0.4380 j -0.2978 ] abs[ 0.5296] arg[ -34.2179]
x3 [ 0.0444 j -0.3480 ] abs[ 0.3508] arg[ -82.7350]
f[10,000 ]
x2 [ 0.4194 j -0.2902 ] abs[ 0.5101] arg[ -34.6798]
x3 [ 0.0212 j -0.3169 ] abs[ 0.3176] arg[ -86.1679]
```

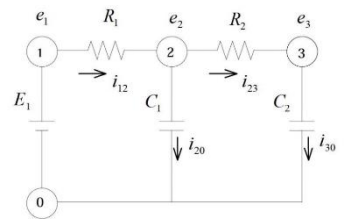
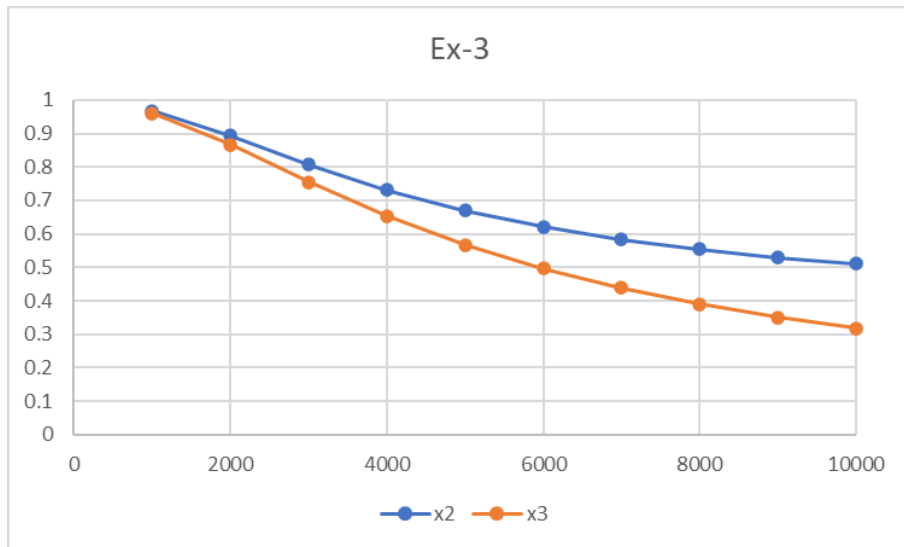
ノード 2 と 3 の電圧値は周波数が高くなるにしたがって減衰して、ローパスフィルタの特性を示すことが分かる。

交流信号の大きさは、`abs[0.9685]` で表示される絶対値でグラフを描きます。

信号の変化が大きい場合には、`/db` によってデシベル値で計算することもできます。

sim.exe の操作練習

Excel でグラフを作成すると、



/ac

周波数 f を 1k に固定して、 $C2$ を 0.02u から 0.1u まで変化させるとどのように変化するかを調べる。

```
? /ac
交流シミュレーションを行ないますが、現在の周波数は 0.000000 です
周波数を変更しますか (y/n) y
周波数は? 1000
ac 入力信号源名は ? e1
値は? 1

値を変化させる素子名は ? c2
ac c2 最低値? 0.02u
ac c2 最高値? 0.1u
ac c2 ステップ? 0.02u
e1[1] j 0
c2[2e-008] f[1,000]
x2 [ 0.9526 j -0.1750 ] abs[ 0.9685] arg[ -10.4084]
x3 [ 0.9161 j -0.2901 ] abs[ 0.9609] arg[ -17.5709]
c2[4e-008] f[1,000]
x2 [ 0.8742 j -0.2469 ] abs[ 0.9084] arg[ -15.7721]
x3 [ 0.7639 j -0.4389 ] abs[ 0.8810] arg[ -29.8799]
c2[6e-008] f[1,000]
x2 [ 0.7926 j -0.2769 ] abs[ 0.8396] arg[ -19.2609]
x3 [ 0.6025 j -0.5041 ] abs[ 0.7856] arg[ -39.9169]
c2[8e-008] f[1,000]
x2 [ 0.7241 j -0.2797 ] abs[ 0.7763] arg[ -21.1168]
x3 [ 0.4659 j -0.5138 ] abs[ 0.6936] arg[ -47.8034]
c2[1e-007] f[1,000]
x2 [ 0.6719 j -0.2688 ] abs[ 0.7236] arg[ -21.8049]
x3 [ 0.3606 j -0.4954 ] abs[ 0.6127] arg[ -53.9469]
```

$C2$ の素子値が大きくなるほど、信号が減衰することが確認できます。

sim.exe の操作練習

次は、ブロック素子を利用する回路を入力します。

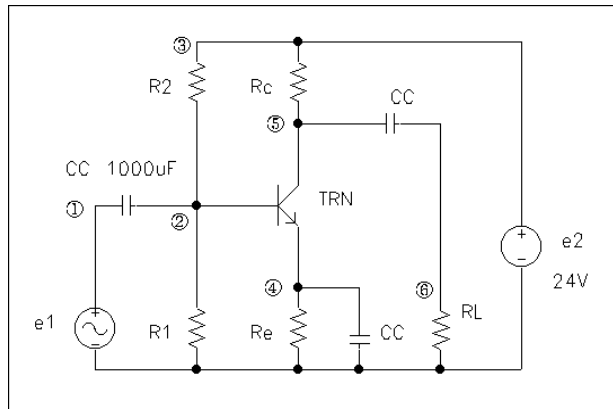


図 ex4

バッチファイル@ex4 を実行すると、

```
B? /ipart
left[ 0] right ? 1
素子名は ? e1
値は ? 1

left[ 0] right ? 2
素子名は ? r1
値は ? 10k

left[ 0] right ? 4
素子名は ? re
値は ? 2.2k

left[ 0] right ? 4
素子名は ? ce
値は ? 1000u

left[ 0] right ? 6
素子名は ? rl
値は ? 1k

left[ 0] right ? 3
素子名は ? e2
値は ? 24

left[ 0] right ? /

left[ 1] right ? 2
素子名は ? cc
値は ? 1000u

left[ 1] right ? /

left[ 2] right ? 3
素子名は ? r2
値は ? 50k

left[ 2] right ? 5
素子名は ? b1
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn2:2,4,5

left[ 2] right ? /

left[ 3] right ? 5
素子名は ? rc
値は ? 3.8k

left[ 3] right ? /

left[ 4] right ? /

left[ 5] right ? 6
素子名は ? cc
素子名 cc は既に使用されています。
同じ素子名を使用しますか? (y/n) y

left[ 5] right ? *
```

トランジスタの接続ノードの入力は、
トランジスタ名(trn2)「コロ」の後に、
ベースの接続ノード番号「」カンマ
エミッタの接続ノード番号「」カンマ
コレクタの接続ノード番号
のように行います

sim.exe の操作練習

まだブロックを解凍していません。

```
? /dpart
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 2] parts[ r1 ] value[ 1e+004 j 0 ]
left[ 0] right[ 3] parts[ e2 ] value[ 24 j 0 ]
left[ 0] right[ 4] parts[ ce ] value[ 0.001 j 0 ]
left[ 0] right[ 4] parts[ re ] value[ 2200 j 0 ]
left[ 0] right[ 6] parts[ r1 ] value[ 1000 j 0 ]
left[ 1] right[ 2] parts[ cc ] value[ 0.001 j 0 ]
left[ 2] right[ 3] parts[ r2 ] value[ 5e+004 j 0 ]
left[ 2] right[ 5] parts[ b1 ] value[ 0 j 0 ]
@ func trn:2,4,5
left[ 3] right[ 5] parts[ rc ] value[ 3800 j 0 ]
left[ 5] right[ 6] parts[ cc ] value[ 0.001 j 0 ]

/bload
```

未解凍のブロックをロードします。

```
? /dpart
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 2] parts[ r1 ] value[ 1e+004 j 0 ]
left[ 0] right[ 3] parts[ e2 ] value[ 24 j 0 ]
left[ 0] right[ 4] parts[ ce ] value[ 0.001 j 0 ]
left[ 0] right[ 4] parts[ re ] value[ 2200 j 0 ]
left[ 0] right[ 6] parts[ r1 ] value[ 1000 j 0 ]
left[ 1] right[ 2] parts[ cc ] value[ 0.001 j 0 ]
left[ 2] right[ 3] parts[ r2 ] value[ 5e+004 j 0 ]
left[ 2] right[ 10] parts[ r1b1 ] value[ 300 j 0 ]
left[ 3] right[ 5] parts[ rc ] value[ 3800 j 0 ]
left[ 4] right[ 7] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 5] right[ 6] parts[ cc ] value[ 0.001 j 0 ]
left[ 5] right[ 9] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 7] right[ 8] parts[ reb1 ] value[ 26 j 0 ]
left[ 7] right[ 9] parts[ kb1 ] value[ 50 j 0 ]
@ master[reb1 ] left[ 7] right[ 8]
left[ 8] right[ 10] parts[ ebb1 ] value[ 0.7 j 0 ]

/point
```

f=0 に設定してから使用します。入力信号源 e1 の値には関係なく、表示ノード 5 の直流電圧が表示されます。ノード 5 が E2 の半分程度の電圧になるように R2 を調整します。現在の電圧を調べます。

```
? /point
f[0 ]
x5 [ 18.8124 j 0 ] abs[ 18.8124] arg[ 0]
```

周波数を f=1K にすると、

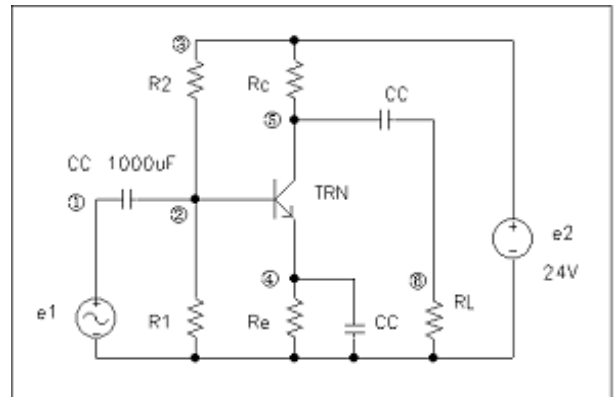
```
? /point
f[1,000 ]
x5 [ -30.8431 j -0.8852 ] abs[ 30.8558] arg[ -178.3561]
```

電源 E2=24 よりも大きな負の数値が示されました。

/point は直流動作点の確認に利用するので、f=0 に設定してから使ってください。

f=0 に戻しておきます。

sim.exe の操作練習



/para

現在は $R2=50K$ で、 $x5$ は $E2$ の半分より高い電圧です。 $R2$ を小さくするとコレクタ電流が増加して $e5$ が下がるので、/para で $R2$ を $10k$ から $20K$ の間で変化させてみます。

```
r2[1.9e+004 ] f[0 ]
x5 [ 11.9125 j 0 ] abs[ 11.9125] arg[ 0]
r2[2e+004 ] f[0 ]
x5 [ 12.3639 j 0 ] abs[ 12.3639] arg[ 0]
```

$R2$ を $19k$ から $20K$ の間で変化させてみます。

```
r2[1.92e+004 ] f[0 ]
x5 [ 12.0053 j 0 ] abs[ 12.0053] arg[ 0]
r2[1.93e+004 ] f[0 ]
x5 [ 12.0512 j 0 ] abs[ 12.0512] arg[ 0]
```

$R2=19.2 K$ に決定します。

$f=0$ に設定してから、直流動作点を表示します。

```
? r2=19.2k
19,200 j 0

? f=0
0 j 0

? /point
f[0 ]
x5 [ 12.0053 j 0 ] abs[ 12.0053] arg[ 0]
```

これでノード 5 の直流動作点が電源電圧 $24V$ の半分に調整出来ました。

sim.exe の操作練習

Maple を利用する 1

ここで、回路の係数行列を作成して Maple を使って直流の動作点を求めてみます。

```
? /point
f[0
x5 [ 12.0053 j 0 ] abs[ 12.0053] arg[ 0]
? /conv
係数行列の出力先を選択して下さい
ファイル(conv.coe) F , 画面 C f

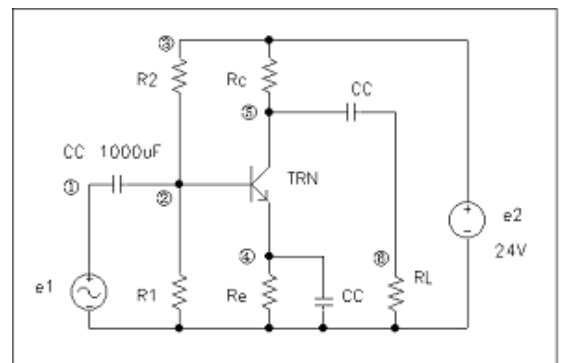
? /point
f[0
x5 [ 12.0053 j 0 ] abs[ 12.0053] arg[ 0]
? /conv
係数行列の出力先を選択して下さい
ファイル(conv.coe) F , 画面 C f

? !ren conv.coe ex4dc.coe
```

直流の動作点を計算する時は、係数行列に含まれる変数 s を 0 に設定します。

また、 $r2=r$ として未定の状態にしておきます。

```
ex4dc.mws
> restart: with(linalg):with(plots):
e1 := 1 ;
r1 := 10000 ;
re := 2200 ;
ce := 0.001 ;
rl := 1000 ;
e2 := 24 ;
cc := 0.001 ;
r2 := r ;
rc := 3800 ;
r1b1 := 300 ;
rbb1 := 0.1 ;
reb1 := 26 ;
kb1 := 50 ;
ebb1 := 0.7 ;
s:=0;
```



sim.exe の操作練習

```

> siki:=matrix(10,10,0);e:=vector(10,0);
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 1] := -s*cc ;
siki[ 2, 2] := +1/r1+s*cc+1/r2+1/r1b1 ;
siki[ 2, 3] := -1/r2 ;
siki[ 2, 10] := -1/r1b1 ;
siki[ 3, 3] := 1 ;
e[3] := +e2 ;
siki[ 4, 4] := +s*ce+1/re+1/rbb1 ;
siki[ 4, 7] := -1/rbb1 ;
siki[ 5, 3] := -1/rc ;
siki[ 5, 5] := +1/rc+s*cc+1/rbb1 ;
siki[ 5, 6] := -s*cc ;
siki[ 5, 9] := -1/rbb1 ;
siki[ 6, 5] := -s*cc ;
siki[ 6, 6] := +1/r1+s*cc ;
siki[ 7, 4] := -1/rbb1 ;
siki[ 7, 7] := +1/rbb1+1/reb1+kb1*1/reb1 ;
siki[ 7, 8] := -1/reb1-kb1*1/reb1 ;
siki[ 8, 2] := -1/r1b1 ;
siki[ 8, 7] := -1/reb1 ;
siki[ 8, 8] := +1/reb1 ;
siki[ 8, 10] := +1/r1b1 ;
siki[ 9, 5] := -1/rbb1 ;
siki[ 9, 7] := -kb1*1/reb1 ;
siki[ 9, 8] := +kb1*1/reb1 ;
siki[ 9, 9] := +1/rbb1 ;
siki[ 10, 8] := -1 ;
siki[ 10, 10] := 1 ;
e[10] := +ebb1 ;
x:=linsolve(siki,e);

> x5:=x[5];

$$x5 = 2735978112 \cdot 10^{-7} \frac{.1091148117 \cdot 10^{43} r - .6128104967 \cdot 10^{46}}{.1190085061 \cdot 10^{34} r + .1092959420 \cdot 10^{38}}$$

> vn:=numer(x5);

$$vn = 2985357365 \cdot 10^{35} r - .1676636106 \cdot 10^{39}$$

> vd:=denom(x5);

$$vd = .1190085061 \cdot 10^{34} r + .1092959420 \cdot 10^{38}$$

> kn:=coeff(vn,r,1);

$$kn = 2985357365 \cdot 10^{35}$$

> vn2:=vn/kn;

$$vn2 = .9999999999 r - 5616.199004$$

> kd:=coeff(vd,r,1);

$$kd = .1190085061 \cdot 10^{34}$$

> vd2:=vd/kd;

$$vd2 = 1.000000000 r + 9183.876480$$

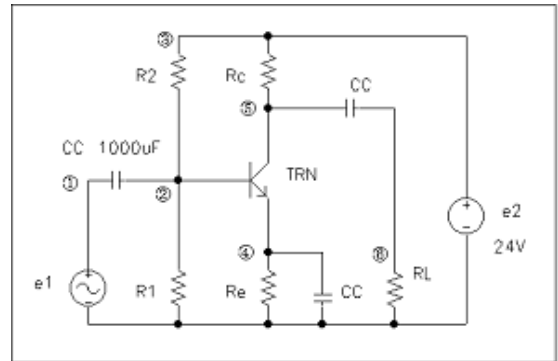
> val:=(kn/kd)*(vn2/vd2);

$$val = 25.08524359 \frac{.9999999999 r - 5616.199004}{1.000000000 r + 9183.876480}$$

> dc:= r ->
25.08524359*(.9999999999*r-5616.199004)/(1.000000000*r+9183.876480);

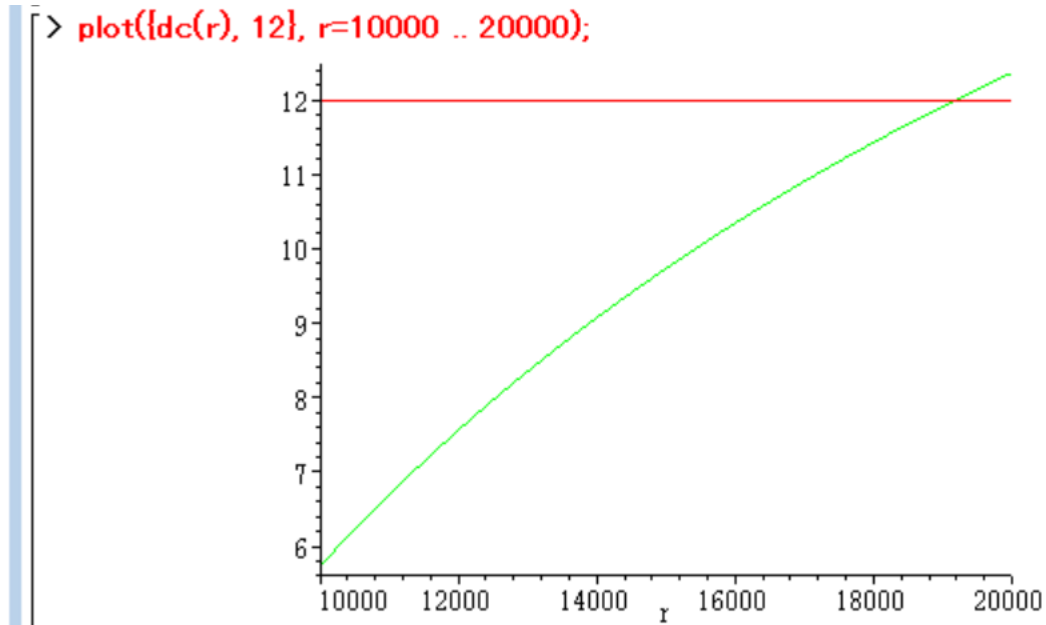
$$dc = r \rightarrow 25.08524359 \frac{.9999999999 r - 5616.199004}{1.000000000 r + 9183.876480}$$


```

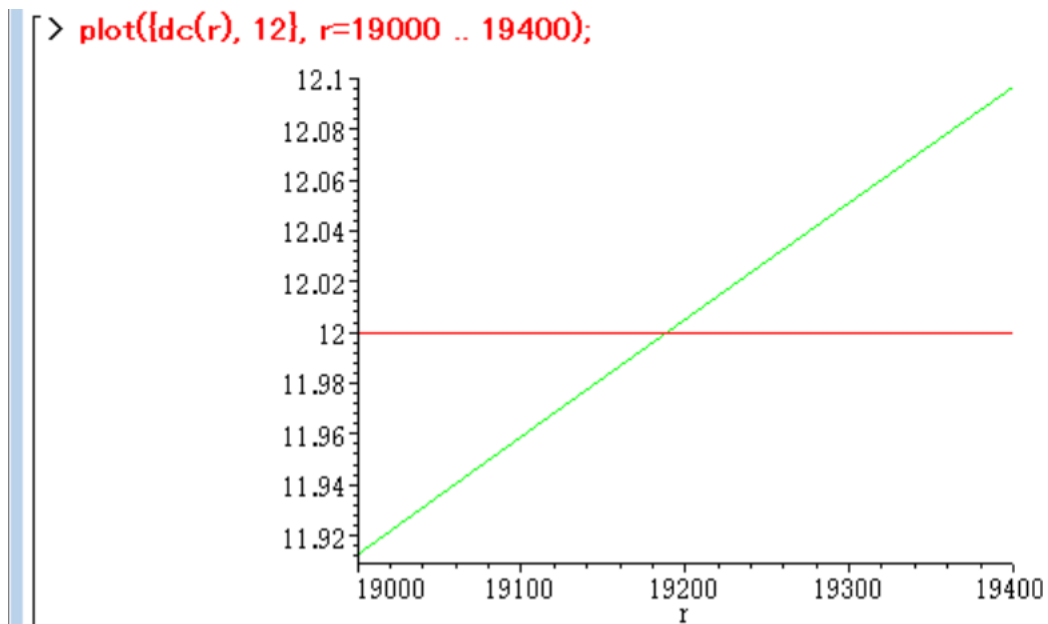


sim.exe の操作練習

このようにして、ノード **x5** の電圧が **r** を変数として、関数 **dc** のように表されました。
r2 が 10K から 20K までの範囲で、**x5** がどのように変化するかをグラフで示します。



r2=19K から **19.4K** の間で **x5=12** になるので、グラフで確認します。



```
> dc(19200);
```

12.00515924

R2=19.2K で **x5=12** になることがわかります。

sim.exe の操作練習

何度か探索範囲を狭めてグラフを作成すると、目標の抵抗値を求めることができます。

Maple の数式処理によって、目標のノード値を達成する抵抗値の数式を作成できれば、何度も探索する必要が無く、即座に抵抗値が得られます。

```

> val;
      25.08524359  .9999999999 r-5616.199004
      -----
      1.0000000000 r+9183.876480
> vn:=numer(val);vd:=denom(val);
      vn := 25.08524359 r-140883.7201
      vd := 1.0000000000 r+9183.876480
> eq:=n5*vd-vn;
      eq := n5(1.0000000000 r+9183.876480) - 25.08524359 r+140883.7201
> fnc:=collect(eq, r);
      fnc := (1.0000000000 n5-25.08524359) r+9183.876480 n5+140883.7201
> ans:=solve(fnc=0, r);
      ans := -2000.  .459193824 109 n5+.7044186005 1010
      -----
      .1000000000 109 n5-.2508524359 1010
> subs(n5=12,ans);
      19188.80883

```

上の数式で、n5 はノード 5 の目標値を表します。

subs(n5=12, ans); は数式 ans の n5 に 12 を代入した時の値を計算して返します。

先程の関数 dc(r)に、この値を代入して確認します。

```

> dc(19188.80883);
      12.00000000

```

このように数式 ans を作成すると、ノード 5 の目標値を 10 に変更しても、すぐに抵抗値を計算しなおすことができます。

```

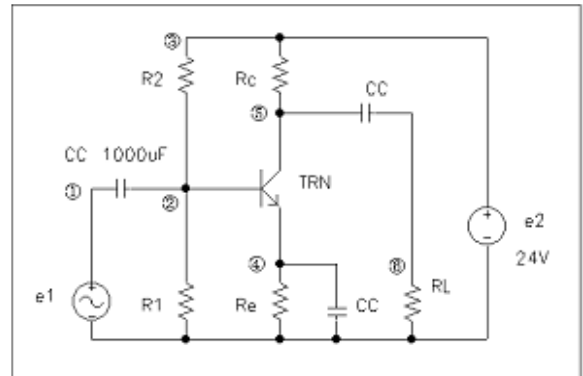
> subs(n5=10,ans);
      15427.16123

```

この回路では、目標のノード値からそれを達成する抵抗値を求める数式が簡単に得られましたが、回路によっては容易に数式を作成できない場合もあります。

その時は、グラフを利用してください。

sim.exe の操作練習



/range で、交流のゲインを測定します。

```
? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x5 [ -119.4764 j -2.9742 ] abs[ 119.5135] arg[ -178.5740]
```

/range による交流シミュレーションでは、ノード 5 の数値は直流電圧ではありません。
入力信号源 e1 の振幅に対するノード 5 の振幅の大きさを表します。

従って、e1=1 なら、x5 の値はこの回路のゲインを表します。

/ac コマンドでも、表示される数値は電圧値ではなく信号源の振幅に対する対象ノードの
振幅を表します。

ここまでのシミュレーション操作によって、分ったことは次の通りです。

R2=19.2K に設定してノード 5 の直流動作点を電源電圧 24V の半分に調整できた。

周波数 1KHz における交流信号の増幅度がおよそ 119 倍で、位相が反転する。

sim.exe の操作練習

次に、もう少し詳しくこの回路の動作を調べるために、sim.exe でシミュレーションした結果の数値とその意味を説明したいと思います。

sim.exe で使用するトランジスタなどの素子は非常に単純な等価回路によって、動作を近似しているために、現実とは異なる結果が得られる場合があります。
数値を見て、それが適切な数値かどうかを判断して利用する必要があります。

例えば、トランジスタ TRN のベースに適切な電圧を与えると、ベース電流が流れて、エミッタ電流とコレクタ電流も流れます。

従って、トランジスタ TRN が正常に動作している時には、エミッタ電流によってノード 4 の電圧が正の値になるはずです。

また、この回路の電源 e2 は 24V ですから、コレクタ電流が変化しても、ノード 5 の電圧が 24V 以上になったり、負の電圧になることはありません。

ノード 2 の電圧を変化させたときに、トランジスタが正常に動作していると判断できるノード 5 の電圧変化の範囲を調べる必要があります。

このために、/padd コマンドによってノード 1 とノード 2 の間に、抵抗 ri=0.1 を追加して、/para コマンドで e1 を変化させてノード 4 とノード 5 の電圧変化を確認します。

```
? /padd
left[ 0] right[ 1] parts[ e1      ] value[ 1 j 0 ]
left[ 0] right[ 2] parts[ r1      ] value[ 1e+004 j 0 ]
left[ 0] right[ 3] parts[ e2      ] value[ 24 j 0 ]
left[ 0] right[ 4] parts[ ce      ] value[ 0.001 j 0 ]
left[ 0] right[ 4] parts[ re      ] value[ 2200 j 0 ]
left[ 0] right[ 6] parts[ rl      ] value[ 1000 j 0 ]
left[ 1] right[ 2] parts[ cc      ] value[ 0.001 j 0 ]
left[ 2] right[ 3] parts[ r2      ] value[ 2e+004 j 0 ]
left[ 2] right[ 10] parts[ rlb1   ] value[ 300 j 0 ]
left[ 3] right[ 5] parts[ rc      ] value[ 3800 j 0 ]
left[ 4] right[ 7] parts[ rbb1   ] value[ 0.1 j 0 ]
left[ 5] right[ 6] parts[ cc      ] value[ 0.001 j 0 ]
left[ 5] right[ 9] parts[ rbb1   ] value[ 0.1 j 0 ]
left[ 7] right[ 8] parts[ reb1   ] value[ 26 j 0 ]
left[ 7] right[ 9] parts[ kb1     ] value[ 50 j 0 ]
@ master[rebl ] left[ 7] right[ 8]
left[ 8] right[ 10] parts[ ebb1   ] value[ 0.7 j 0 ]
最大の 素子番号 = 16
最大のノード番号 = 10

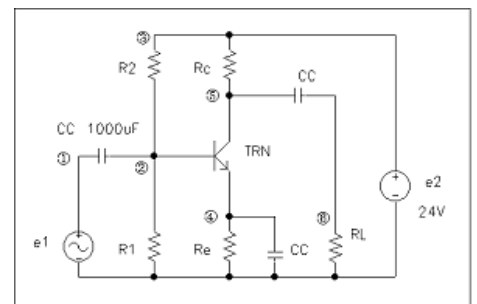
left[ 0] right ? /

left[ 1] right ? 2
素子名は ? ri
値は ? 0.1

left[ 1] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.7 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
5

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 4
ノード番号 (0 なら終了) ? 5
ノード番号 (0 なら終了) ?
```



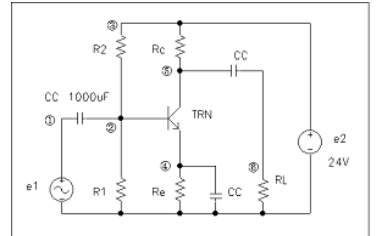
sim.exe の操作練習

抵抗 r_i を接続したので、TRN のベース電圧を希望の電圧に設定してその時のエミッタ電圧やコレクタ電圧を観察できるようになりました。

e1 を 0V から 24V まで、0.1V ステップで計算してファイルに記録することになります。

/auto、/gpos → 300 ポイントを設定する、/mk

```
? /auto
? /gpos
MANU ON または 最高/最低の比が 10 以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 32 です
300
gpos = 300 に設定しました
? /mk
データファイルを作成する..... /mk または /nomk で切り替える
値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /auto (自動切り替え) です
..... /auto または /manu で切り替える
等比級数のポイント数は 300 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
4 5
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する
```



/para → ファイル名 ex4-tst1 を入力して、

e1 を 0V から 24V まで、0.1V ステップで計算してファイルに記録する

```
? /para
データファイル名は ? ex4-tst1
値を変化させる素子名は ? e1
para e1 最低値 ? 0
para e1 最高値 ? 24
para e1 ステップ ? 0.1
```

出力されたファイル ex4-tst1.d を windows の「メモ帳」などで開いてノード 4 とノード 5 の電圧を確認します。

2 241	2 は測定するノードの個数、241 は測定するポイント数
0	0 は/para コマンドで変更する素子 e1 の値が 0V であること
4 -0.69782	4 は測定ノードの番号、-0.69782 はその電圧
5 25.1817	5 は測定ノードの番号、25.1817 はその電圧
0 0	測定するノードの終わり
0.1	e1=0.1V の場合 以下同様に繰り返す
4 -0.598116	
5 25.0129	ノード 4 や 5 が負の値になったり、ノード 5 が 24 を超える
0 0	場合には、トランジスタは動作していないと判断する
.	
.	
0.7	

sim.exe の操作練習

4 0.00011

5 23.9998

0 0

ノード電圧を確認すると、e1 が 0V から 0.7V までは動作していないと考えられる。

0.8 e1=0.8V では、ノード 4 がわずかに正になるが、余裕が少ない

4 0.0998134

5 23.831

0 0

0.9 e1=0.9V 以上になると、動作している

4 0.199518

5 23.6621

0 0

1

4 0.299222

5 23.4933

0 0

1.1

4 0.398926

5 23.3245

0 0

e1 が 0.8V を越えると、トランジスタが動作していると思われる。

しかし、e1=0.8V の時のノード 5 の電圧は 23.831V なので、電源 e2 との差がわずか 0.169V しかないので、正常に動作する e1 の電圧は 0.9V 以上と考える。

e1 をさらに高くすると、ノード 5 が負の値に変化する。

正常に動作する範囲は、ノード 5 が 0.2V 程度以上と考える。

14.6

4 13.859

5 0.53112

0 0

14.7

4 13.9587

5 0.362281

0 0

sim.exe の操作練習

```

14.8          e1=14.8V ではノード 5 が 0.2V 程度なので、動作している
4 14.0584
5 0.193441
0 0
14.9          e1=14.9V 以上では、ノード 5 が 0 V 付近から負の値に変化する
4 14.1581      トランジスタが動作していないと判断する
5 0.0246018
0 0
15
4 14.2578
5 -0.144238
0 0

```

上記測定データより、ノード 5 の電圧範囲は、0.193441V から 23.6621V だと分かった。
 従って、ノード 5 の動作の中心電圧は 11.93V と計算される。

R2=19.2 K に設定して定めたノード 5 の動作点 12 V は適切だったと言える。

また。ノード 5 の最大変化幅は $23.47 V_{p-p}$ になるので、交流電圧の実効値に換算すると

$$\frac{23.47}{2\sqrt{2}} = 8.3V \text{ という事になる。}$$

e1 が 0.9 ～14.8 V まで変化すると、x5 は 23.7 ～0.2 V まで変化する。従って、ex4 の回路の直流ゲインは $(0.2-23.7)/(14.8-0.9)=-1.69$ となる。

この値は、コレクタ抵抗 R_c とエミッタ抵抗 R_e の比 $R_c/R_e=3800/2200=1.72$ に近い。

この回路の交流ゲインはおよそ 119 倍と求められている。

最大振幅の出力を得るために必要な交流入力信号の大きさは、

$$\text{振幅で } \frac{(23.7-0.2)}{119} = 0.197 V_{p-p} \text{、実効値では } \frac{0.197}{2\sqrt{2}} \cdot 1000 = 69.6 mV \text{ と求められます。}$$

交流ゲインがこのような大きくなるのは、周波数が高くなるとエミッタ抵抗に並列に付けられているキャパシタのインピーダンスがほとんど 0 になり、トランジスタの内部抵抗 $reb1=26$ 程度まで減少するので、ゲインが $R_c/reb1=3800/26=146$ に近付くためである。

sim.exe の操作練習

/range で周波数 1KHz の時のノード電圧を確認する。

```
? /range
ac 入力信号源名は ? e1
値は ? 1

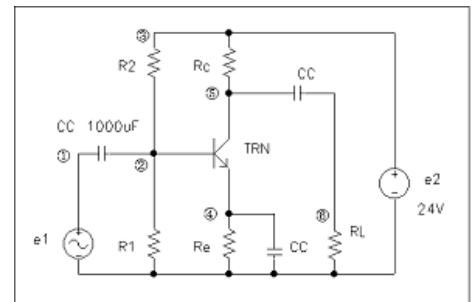
range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1
j 0
]
f[1,000
]
x2 [ 1 j 5.045887e-004] abs[ 1] arg[ 0.0289]
x4 [6.147477e-004 j -0.0245] abs[ 0.0245] arg[ -88.5626]
x5 [ -119.4764 j -2.9742] abs[ 119.5135] arg[ -178.5740]
```

エミッタの交流電圧はほとんど 0V になっていることが分かる。

最後に、ri を追加する前の回路の、直流から 1KHz までの周波数特性を確認する。

```
? /range
データファイル名は ? ex4-range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 0
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 10
```



```
1 101      表示するノード数 1   計算するポイント数 101
0          f=0 Hz      e1 は直流 1 V
5 0        x5=0 V      直流は増幅できないことを示している
0 0
10         f=10 Hz     e1 は交流 10 Hz 振幅 1
5 44.3041  x5=44.3041  ゲインは 44.3041 倍
0 0
20         f=20 Hz
5 74.5636  x5=74.5636  周波数が高くなるとゲインが急速に増加する
0 0
30
5 91.7421  x5 の数値は交流信号の値[-17.12 j -40.86] などの絶対値が保存されている。
0 0
40
5 101.304
0 0
```

sim.exe の操作練習

```
.  
. 300          f=300 Hz  
5 119.136      x5=119.136  300Hz 位でゲインが最大値に近くなった  
0 0  
. 990  
5 119.513  
0 0  
1000  
5 119.513  
0 0
```

データを保存する時には、`/val` によって信号の絶対値を計算したり、`/db` によって信号のデシベル値を計算する方法を選ぶ必要があります。

sim.exe の操作練習

周波数特性をデシベル値で計算したデータを使って、ex4 のゲインの周波数特性をグラフにしました。

1 101

0

5 -1.#INF デシベルに変換すると、x5=0 なので計算不能となります

0 0

10

5 32.9289 x5 32.9289 db

0 0

.

.

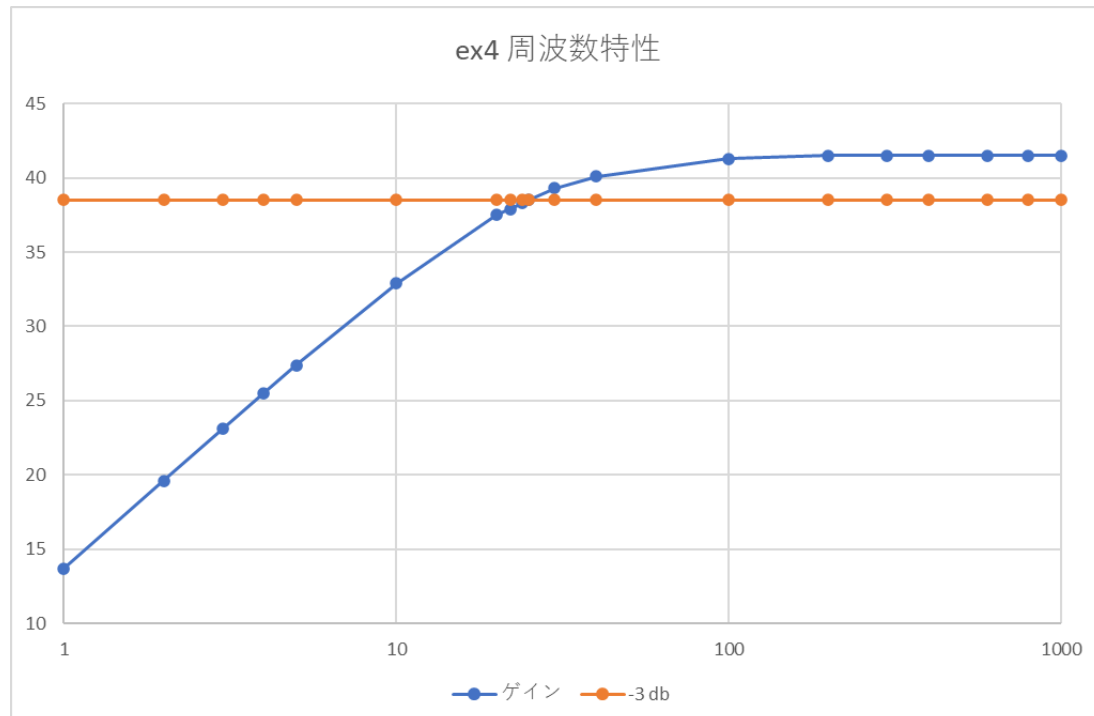
90

5 41.2266

0 0

100

5 41.2864



カットオフ周波数が **25Hz** のハイパスフィルタ特性を示しています。

sim.exe の操作練習

Maple を利用する 2

保存してある ex4dc.coe を使用して、Maple を使って交流のゲインを計算します。

交流シミュレーションを行うためには、回路図の入力信号源 e1 以外のすべての独立電圧源と独立電流源を 0 に設定して、 $s = I \cdot 2 \cdot \pi \cdot f$, $I = \sqrt{-1}$, $\pi = 3.141592$ と設定します。

```

ex4ac.mws
> restart: with(linalg):with(plots):
Warning, new definition for norm
Warning, new definition for trace
> e1 := 1;
r1 := 10000;
re := 2200;
ce := 0.001;
rl := 1000;
e2 := 0;
cc := 0.001;
r2 := 19200;
rc := 3800;
r1b1 := 300;
rbb1 := 0.1;
reb1 := 26;
kb1 := 50;
ebb1 := 0;

> s:=I*2*pi*f;pi:=3.141592;
                                     s:=2 I pi f
                                     pi:=3.141592

> siki:=matrix(10,10,0):e:=vector(10,0):
siki[ 1, 1] := 1;
e[1] := +e1;
siki[ 2, 1] := -s*cc;
siki[ 2, 2] := +1/r1+s*cc+1/r2+1/r1b1;
siki[ 2, 3] := -1/r2;
siki[ 2, 10] := -1/r1b1;
siki[ 3, 3] := 1;
e[3] := +e2;
siki[ 4, 4] := +s*ce+1/re+1/rbb1;
siki[ 4, 7] := -1/rbb1;
siki[ 5, 3] := -1/rc;
siki[ 5, 5] := +1/rc+s*cc+1/rbb1;
siki[ 5, 6] := -s*cc;
siki[ 5, 9] := -1/rbb1;
siki[ 6, 5] := -s*cc;
siki[ 6, 6] := +1/rl+s*cc;
siki[ 7, 4] := -1/rbb1;
siki[ 7, 7] := +1/rbb1+1/reb1+kb1*1/reb1;
siki[ 7, 8] := -1/reb1-kb1*1/reb1;
siki[ 8, 2] := -1/r1b1;
siki[ 8, 7] := -1/reb1;
siki[ 8, 8] := +1/reb1;
siki[ 8, 10] := +1/r1b1;
siki[ 9, 5] := -1/rbb1;
siki[ 9, 7] := -kb1*1/reb1;
siki[ 9, 8] := +kb1*1/reb1;
siki[ 9, 9] := +1/rbb1;
siki[ 10, 8] := -1;
siki[ 10, 10] := 1;
e[10] := +ebb1;
x:=linsolve(siki,e);

```

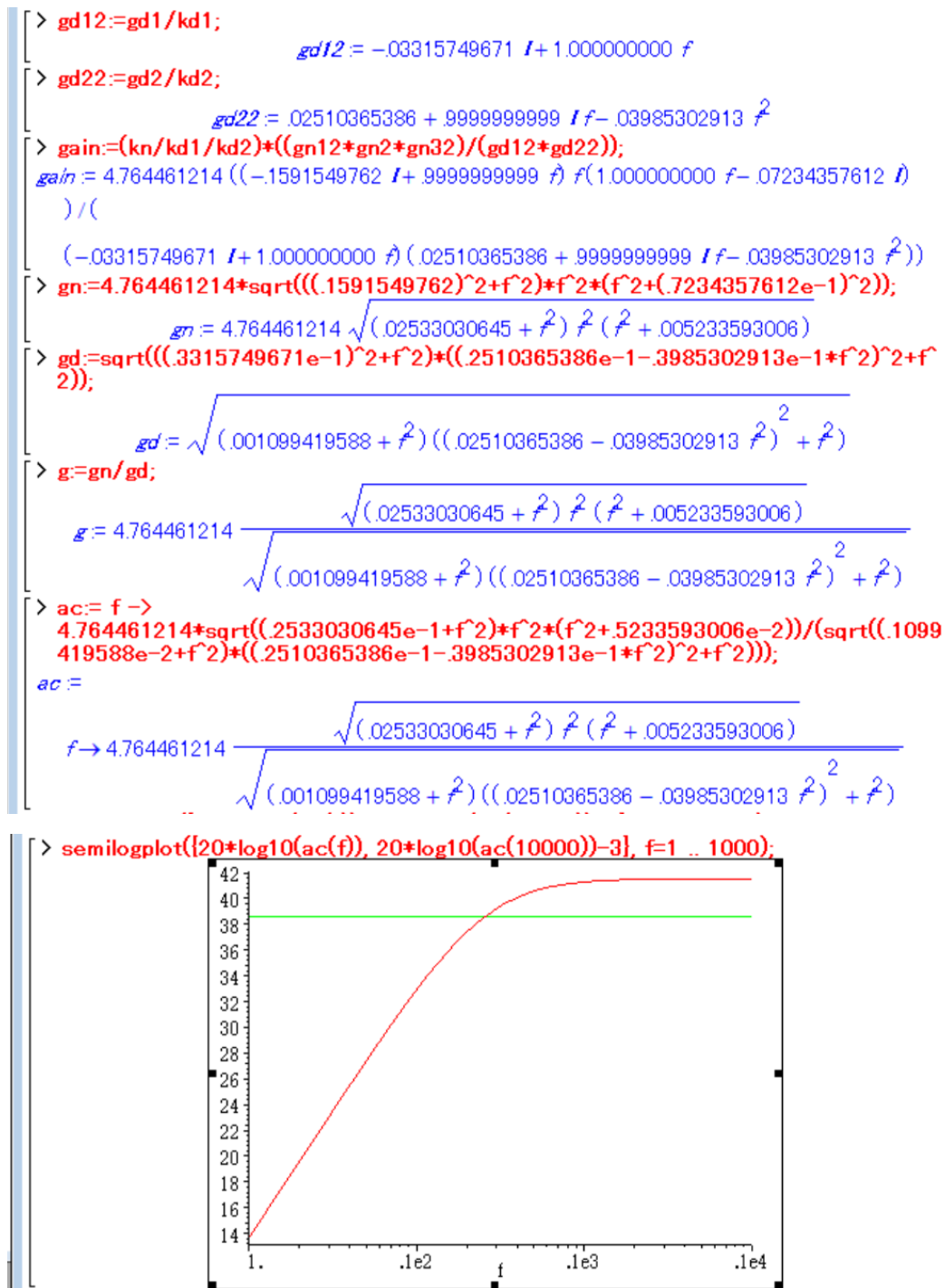

sim.exe の操作練習

```

> x5:=x[5]:g:=x[5]/e1;
g:= -4195502136 1025 (
  (-62500. I+392699. f) f(.1570796000 1017 f-.1136370000 1016 f) / (
  (-1099584229 1035 -.4380176029 1036 I f+.1745632829 1035 f2)
  (-411187500 109 I+.1240104172 1011 f)
)
> gn:=numer(g);
gn:=
  -4195502136 1025 (-62500. I+392699. f) f(.1570796000 1017 f-.1136370000 1016 f)
> kn0:=-.4195502136e25; gn1:=(-62500.*I+392699.*f); gn2:=f;
gn3:=(.1570796000e17*f-.1136370000e16*I);
      kn0:= -4195502136 1025
      gn1:= -62500. I+392699. f
      gn2:= f
      gn3:= .1570796000 1017 f-.1136370000 1016 I
> kn1:=coeff(gn1,f,1);
      kn1:= 392699.
> kn3:=coeff(gn3,f,1);
      kn3:= .1570796000 1017
> kn:=kn0*kn1*kn3;
      kn:= -.2587995569 1047
> gn12:=gn1/kn1;
      gn12:= -.1591549762 I+.9999999999 f
> gn32:=gn3/kn3;
      gn32:= 1.000000000 f-.07234357612 I
> gd:=denom(g);
gd:= (-1099584229 1035 -.4380176029 1036 I f+.1745632829 1035 f2)
  (-411187500 109 I+.1240104172 1011 f)
> gd1:=-411187500.*I+.1240104172e11*f;
      gd1:= -411187500 109 I+.1240104172 1011 f
> kd1:=coeff(gd1,f,1);
      kd1:= .1240104172 1011
> gd2:=-.1099584229e35-.4380176029e36*I*f+.1745632829e35*f^2;
      gd2:= -.1099584229 1035 -.4380176029 1036 I f+.1745632829 1035 f2
> kd2:=coeff(gd2,f,1)/I;
      kd2:= -.4380176029 1036

```

sim.exe の操作練習



f=10kHz のゲインは、41.55 dB で、f=25Hz では 3 dB 低くなります。

```

> 20*log10(ac(10000));
41.55102190
> 20*log10(ac(25));
38.52929890

```

回路ブロックとトランジスタ・FET・IC などの等価回路

「ex-5」の次の「サンプルデータファイルの内容」(ic1.cir と bpf2.cir) 及び「Sim.exe で使用する標準部品として準備してあるブロック素子」(トランジスタ trn.cir, trn2.cir, trp.cir, trp2.cir と fetd.cir, fetdp.cir, fete.cir, fetep.cir の等価回路) を参照してください。

回路ブロックの詳細については「回路シミュレーション.pdf」の[7] 回路のブロック定義 (p40 から p46)を参照してください。

回路を入力する場合に、

ic を入力する時は、

素子名の入力で、「b」に続けてブロック素子の番号(1 など)を入力してから、

ic の名前 (ic1) 「コロン」

ノード番号 0 「カンマ」

マイナス端子の接続先のノード番号(2) 「カンマ」

プラス端子の接続先のノード番号(0) 「カンマ」

出力端子の接続先のノード番号(3)

のように行います。

トランジスタ・FETを入力する時は、

素子名の入力で、「b」に続けてブロック素子の番号(1 など)を入力してから、

トランジスタ名(trn2) 「コロン」の後に、

ベース (ゲート) の接続ノード番号 「カンマ」

エミッタ (ソース) の接続ノード番号 「カンマ」

コレクタ (ドレイン) の接続ノード番号

のように行います。

その他のブロック素子を入力する時は、

素子名の入力で、「b」に続けてブロック素子の番号(1 など)を入力してから、

素子名(bpf2 など) 「コロン」の後に、

素子の外部ノードの接続先のノード番号 「カンマ」

を外部ノードの個数だけ繰り返します。

最後の外部ノードには「カンマ」は不要です。

sim.exe の操作練習

回路ブロックのデータファイルを作成するには

回路データを入力してから、/save コマンドでファイル名を入力して、外部ノードを入力して保存すると name.cir ファイルが出来ます。

例えば、ex-1 の回路をブロックとして保存するなら、@ex1 で入力後/save を実行します。

```
1 番のバッチファイル ex1.sb を終了します。
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[          1 j 0          ]
left[ 0] right[ 2] parts[ r2          ] value[          2 j 0          ]
left[ 1] right[ 2] parts[ r1          ] value[          1 j 0          ]
最大の 素子番号 = 3
最大のノード番号 = 2
? /save
セーブ
ファイル名は ? ex1
外部ノード (1, 2, 3 の様に)
0, 1, 2
```

ex1.cir の内容

```
@ex1:0,1,2
```

```
2                      ノード番号の最大値を表す
```

```
0 1 @e @e1 @ 0 1 0 @
```

```
0 2 @r @r2 @ 0 2 0 @
```

```
1 2 @r @r1 @ 0 1 0 @
```

回路ブロックを部品として利用する場合には、電源を取り除いたほうが利用しやすいと思いますので、エディタで最初の行を削除して、保存すれば良いと思います。

e1 に関する、0 1 @e @e1 @ 0 1 0 @ の行を取り除いた回路ブロック e1.cir は、

```
@ex1:0,1,2
```

```
2
```

```
0 2 @r @r2 @ 0 2 0 @
```

```
1 2 @r @r1 @ 0 1 0 @
```

e1.cir を/load してみる。

```
? /dpart
left[ 0] right[ 2] parts[ r2          ] value[          2 j 0          ]
left[ 1] right[ 2] parts[ r1          ] value[          1 j 0          ]
最大の 素子番号 = 2
最大のノード番号 = 2
```

sim.exe の操作練習

次は、オペアンプを含む回路図を入力します。

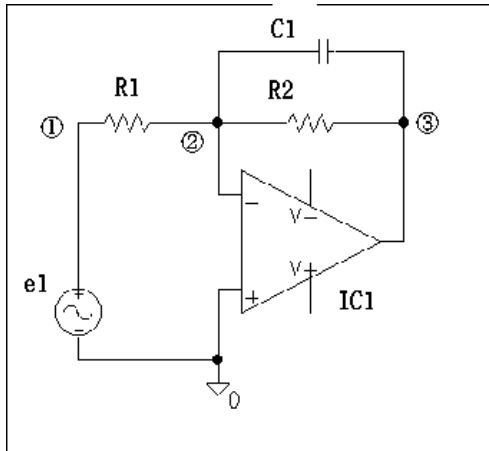


図 ex5

@ex5 を実行します。

```
/ipart
left[ 0] right ? 1
素子名は ? e1
値は ? 1

left[ 0] right ? /

left[ 1] right ? 2
素子名は ? r1
値は ? 1k

left[ 1] right ? /

left[ 2] right ? 3
素子名は ? b1
ファイル名と接続ノードを入力して下さい(ic1:0, 1, 2 の様に)
ic1:0, 2, 0, 3

left[ 2] right ? 3
素子名は ? r2
値は ? 10k

left[ 2] right ? 3
素子名は ? c1
値は ? 0.1u

left[ 2] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
独立電圧源素子の個数 1
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

表示ノード番号 (0 なら全て、-1 ならこのまま) ?
未解凍のブロックがあります。 解凍しますか (y/n)
bname b1 sfnc ic1:0, 2, 0, 3
```

ic の入力は、

ic の名前 (ic1)

「コロン」

ノード番号 0「カンマ」

マイナス端子の接続先のノ

ード番号(2)「カンマ」

プラス端子の接続先のノ

ード番号(0)「カンマ」

出力端子の接続先のノ

ード番号(3)

のように行います。

「サンプルデータファイル

の内容」と「標準部品の等

価回路」などを確認

詳細は「回路シミュレーシ
ョン」の

[7] 回路のブロック定義

(p40 から p46)を参照

sim.exe の操作練習

未解凍のブロックがあります。`/bload` を実行してから、シミュレーションしてください。

```
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0      ]
left[ 0] right[ 3] parts[ v1b1       ] value[ 1e+006 j 0      ]
@ master[r1b1     ] left[ 2] right[ 0]
left[ 1] right[ 2] parts[ r1          ] value[     1000 j 0      ]
left[ 2] right[ 0] parts[ r1b1       ] value[ 1e+006 j 0      ]
left[ 2] right[ 3] parts[ c1          ] value[ 1e-007 j 0      ]
left[ 2] right[ 3] parts[ r2          ] value[ 1e+004 j 0      ]
```

この回路もローパスフィルタ特性を示します。`/range` を使って確認してください。

`r1`, `r2`, `c1` の値を変えた時の周波数特性の変化なども調べてください。

次に示すバッチファイルは、図 ex5 の回路図を入力するまでをバッチファイルに記録して、エディタを使ってそのファイルにコメントと`/wait` を追加したものです。

ファイル名 ex5.sb

`!cls`

`%回路図は ex5 です`

`%これから回路図を入力します`

`/wait`

`/cmode`

`%`

`/ipart`

`1`

`e1`

`1`

`/`

`2`

`r1`

`1k`

`/`

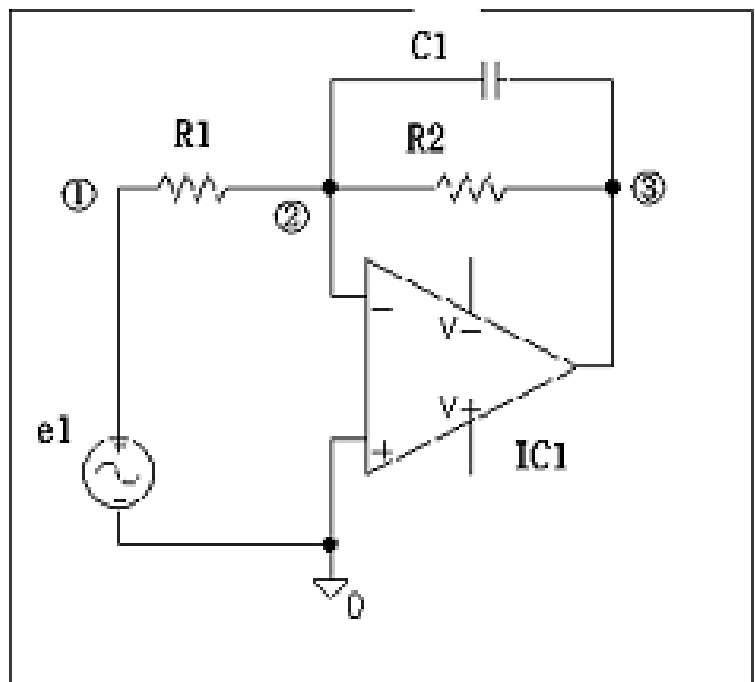
`3`

`b1`

`ic1:0,2,0,3`

`3`

`r2`



sim.exe の操作練習

10k

3

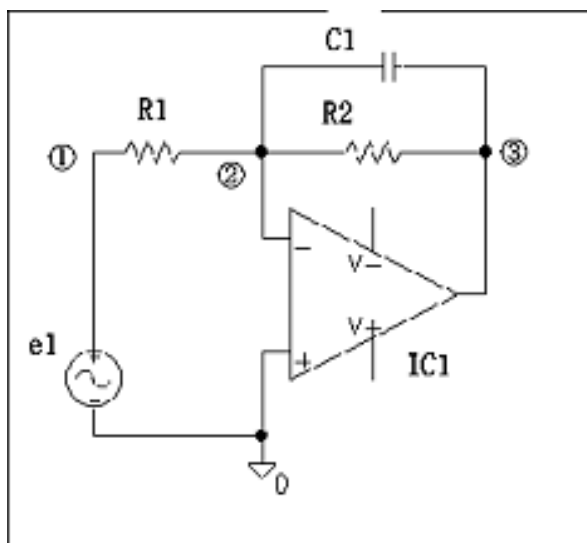
c1

0.1u

*

y

3



回路情報を確認します。

```
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0      ]
left[ 0] right[ 3] parts[ v1b1         ] value[ 1e+006 j 0      ]
@ master[r1b1     ] left[ 2] right[ 0]
left[ 1] right[ 2] parts[ r1          ] value[     1000 j 0      ]
left[ 2] right[ 0] parts[ r1b1         ] value[ 1e+006 j 0      ]
left[ 2] right[ 3] parts[ c1          ] value[ 1e-007 j 0      ]
left[ 2] right[ 3] parts[ r2          ] value[ 1e+004 j 0      ]
最大の 素子番号 = 6
最大のノード番号 = 3
```

sim.exe では、オペアンプを使った回路はオペアンプの電源を入力しなくてもシミュレーションができます。

sim.exe の標準部品の ic1.cir は動作点を調整する必要がないので、電源不要な回路として定義してあるからです。

しかし、標準部品の trn.cir などは動作点を調整する必要があるので、回路に電源を追加しなければなりません。

以上で、sim.exe の基本的な操作練習は終わりです。

サンプルデータファイルの内容

サンプルデータファイルの内容

ic1.cir: sim.exeで使用する、標準オペアンプのブロック素子
0はグラウンド、1はマイナス入力、2はプラス入力、3は出力のノードを表わす

```
@ic1:0,1,2,3
3
0 3 @v @v1 @r1 0 100000 0 @
1 2 @r @r1 @ 0 1000000 0 @
fnc
```

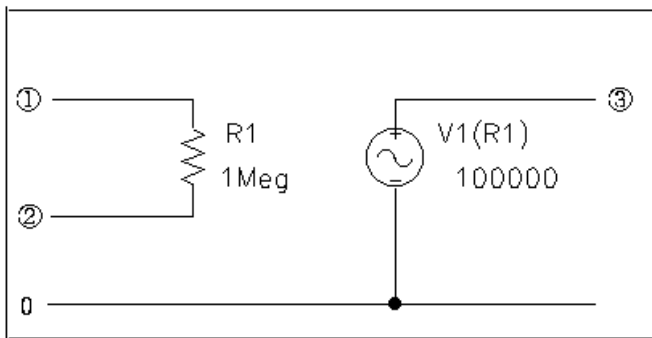


図 ic1.cir の等価回路

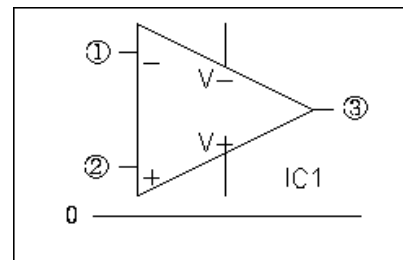


図 一般回路図での表記法

bpf11.cir: ic1を使用して作成した、1次バンドパスフィルターのブロック素子
0はグラウンド、1は信号の入力、4は出力のノードを表わす。
ブロック素子として定義してあるので、入力信号の電圧源は含んでいない。

```
@bpf11:0,1,4
4
0 2 @r @r1b @ 0 357.48 0 @
0 3 @b @b1 @ 1 0 0 @ic1:0,3,0,4
1 2 @r @r1a @ 0 81215 0 @
2 3 @c @c @ 0 1.674e-08 0 @
2 4 @c @c @ 0 1.674e-08 0 @
3 4 @r @r2 @ 0 280968 0 @
fnc
```


サンプルデータファイルの内容

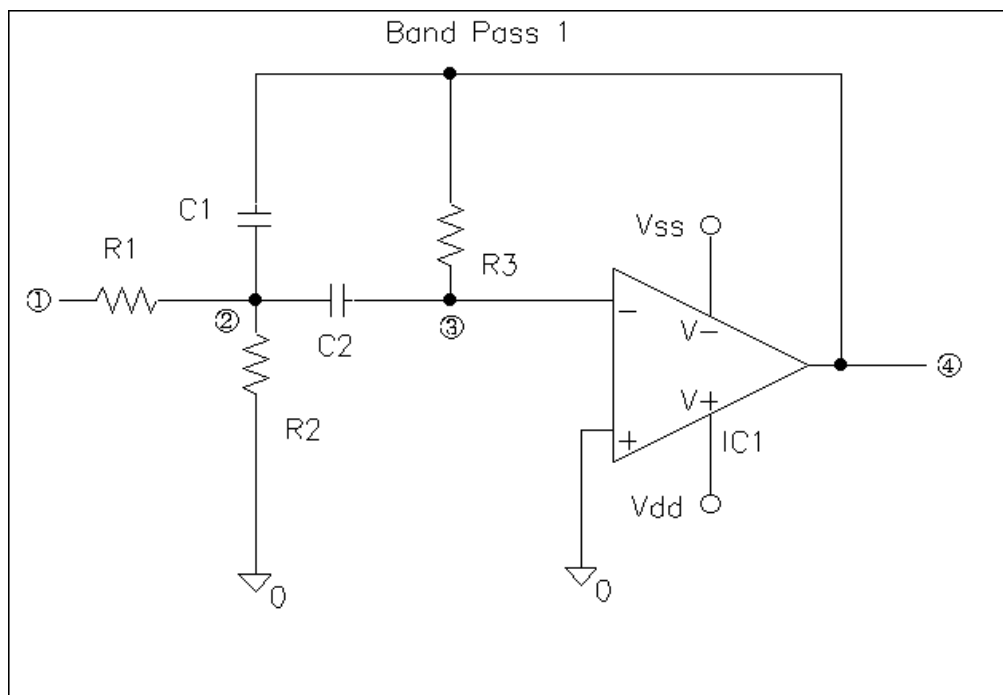
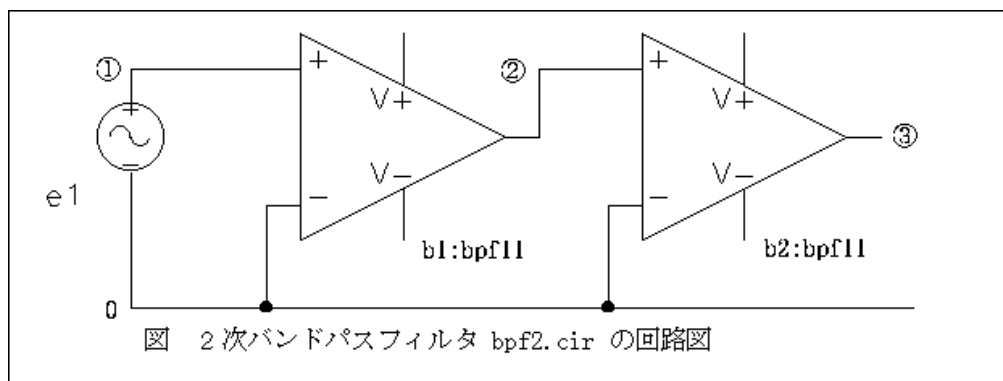


図 バンドパスフィルタの基本となる、1次バンドパスフィルタ回路 bpf11.cir

bpf2.cir: b p f 1 1を使用して作成した、2次バンドパスフィルタのネットリスト
 ブロック素子として定義された、bpf11.cirを2個使用して作成した2次のバンドパス
 フィルタの回路。0はグラウンド、1は信号の入力、3は出力のノードを表わす。
 信号源電圧源 e1 がノード1とノード0に接続されている。

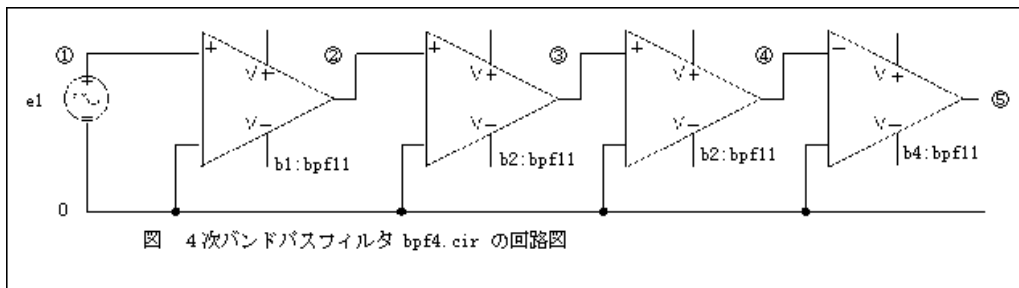
```
@bpf2:0, 1, 3
3
0 1 @e @e1 @ 0 1 0 @
0 2 @b @b1 @ 1 0 0 @bpf11:0, 1, 2
0 3 @b @b2 @ 2 0 0 @bpf11:0, 2, 3
fnc
```



サンプルデータファイルの内容

bpf4.cir: b p f 1 1 を使用して作成した、4 次バンドパスフィルターのネットリスト
 ブロック素子として定義された、bpf11.cir を 4 個使用して作成した 4 次のバンドパス
 フィルタの回路。0 はグラウンド、1 は信号の入力、5 は出力のノードを表わす。
 信号源電圧源 e1 がノード 1 とノード 0 に接続されている。

```
@bpf4:0, 1, 5
5
0 1 @e @e1 @ 0 1 0 @
0 2 @b @b1 @ 1 0 0 @bpf11:0, 1, 2
0 3 @b @b2 @ 2 0 0 @bpf11:0, 2, 3
0 4 @b @b3 @ 3 0 0 @bpf11:0, 3, 4
0 5 @b @b4 @ 4 0 0 @bpf11:0, 4, 5
fnc
```



バッチファイルのサンプル

b a t 1 . s b, b a t 2 . s b, b a t 3 . s b はバッチファイルの
 サンプルである。これらは、s i m . e x e の # コマンドで作られたバッチ
 ファイルをエディタを使用して、コメントを付加したり、/ w a i t コマンド
 を追加したものである。

b a t 1 . s b :

```
% バッチファイルによる自動実行のテスト
% 2 次のバンドパスフィルターの回路データ bpf2
% を読み込んで、解凍せずに load を終了後 bload
% で解凍して、回路リストを表示する。
/load
```

サンプルデータファイルの内容

```
% 回路データのファイル名
bpf2
% 表示ノード指定しない

% 解凍しない

% 解凍しない

% 解凍のために bload を実行する
/bload
% 表示ノード指定しない

% キャパシタ cb2 の値を変更する
cb2=1.514e-8
% 回路リストを表示して、キー入力待ち
/dpart
/wait
% 次のバッチファイルを実行して、キー入力待ち
@bat2
/wait

bat2.sb :
% バッチファイル名 bat2
% 表示ノードを指定して、周波数特性を計算する
%
% 表示ノードを指定
/disp
3

% 周波数特性を計算して、データファイル bpf2b を作る
% モードをセット
/smode
/range
bpf2b
```

サンプルデータファイルの内容

```
% 入力信号源名と値
e1
1
% 最低周波数、最高周波数、周波数ステップ
500
1500
50
% キー入力待ち
/wait
% 次のバッチファイルを実行する
@bat3
% キー入力待ち bat2 終了
/wait

bat3.sb:
% バッチファイル名 bat3
% bpf2 の接続行列を出力する
/save
bpf2-b
0, 1, 3

% 係数行列を出力する
/conv
% 出力先の判断を入力する
f
% ファイル名を bpf2.coe に変更する
! del bpf2.coe
! ren conv.coe bpf2.coe

bpf2 の接続行列 (/save で作られた bpf2-b.cir の内容)
@bpf2-b:0, 1, 3
7
0 1 @e @e1 @ 0 1 0 @
0 2 @* @b1 @ 1 0 0 @bpf11:0, 1, 2
0 2 @v @v1b1b1 @r1b1b1 0 1e+006 0 @
0 3 @* @b2 @ 2 0 0 @bpf11:0, 2, 3
```

サンプルデータファイルの内容

```

0 3 @v @v1b1b2 @r1b1b2 0 1e+006 0 @
0 4 @r @r1bb1 @ 0 357.48 0 @
0 5 @* @b1b1 @ 3 0 0 @ic1:0,5,0,2
0 6 @r @r1bb2 @ 0 357.48 0 @
0 7 @* @b1b2 @ 4 0 0 @ic1:0,7,0,3
1 4 @r @r1ab1 @ 0 81215 0 @
2 6 @r @r1ab2 @ 0 81215 0 @
4 2 @c @cb1 @ 0 1.674e-008 0 @
4 5 @c @cb1 @ 0 1.674e-008 0 @
5 0 @r @r1b1b1 @ 0 1e+006 0 @
5 2 @r @r2b1 @ 0 280968 0 @
6 3 @c @cb2 @ 0 1.514e-008 0 @
6 7 @c @cb2 @ 0 1.514e-008 0 @
7 0 @r @r1b1b2 @ 0 1e+006 0 @
7 3 @r @r2b2 @ 0 280968 0 @
fnc

```

bpf2 の係数行列 (/conv で作られた bpf2.coe の内容)

```

restart: with(linalg):
siki:=matrix(7,7,0):e:=vector(7,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 2] := 1 ;
siki[ 2, 5] := -v1b1b1 ;
siki[ 3, 3] := 1 ;
siki[ 3, 7] := -v1b1b2 ;
siki[ 4, 1] := -1/r1ab1 ;
siki[ 4, 2] := -s*cb1 ;
siki[ 4, 4] := +1/r1bb1+1/r1ab1+s*cb1+s*cb1 ;
siki[ 4, 5] := -s*cb1 ;
siki[ 5, 2] := -1/r2b1 ;
siki[ 5, 4] := -s*cb1 ;
siki[ 5, 5] := +s*cb1+1/r1b1b1+1/r2b1 ;
siki[ 6, 2] := -1/r1ab2 ;
siki[ 6, 3] := -s*cb2 ;
siki[ 6, 6] := +1/r1bb2+1/r1ab2+s*cb2+s*cb2 ;

```

サンプルデータファイルの内容

```
siki[ 6, 7] := -s*cb2 ;  
siki[ 7, 3] := -1/r2b2 ;  
siki[ 7, 6] := -s*cb2 ;  
siki[ 7, 7] := +s*cb2+1/r1b1b2+1/r2b2 ;  
x:=linsolve(siki,e);
```

```
e1 := 1 ;  
r1bb1 := 357.48 ;  
r1ab1 := 81215 ;  
cb1 := 1.674e-008 ;  
r2b1 := 280968 ;  
r1bb2 := 357.48 ;  
r1ab2 := 81215 ;  
cb2 := 1.514e-008 ;  
r2b2 := 280968 ;  
v1b1b1 := 1e+006 ;  
r1b1b1 := 1e+006 ;  
v1b1b2 := 1e+006 ;  
r1b1b2 := 1e+006 ;  
x3 := -0.0216264 ;
```

```
;end;
```

変換終了

使用した式の個数 20、 密度 40.816 %

使用した式のサイズ 360

上記の係数行列のデータを

```
restart: with(linalg):
```

から `x:=linsolve(siki,e);`
の行までを数式処理ソフト Maple に入力して実行すれば
回路方程式を文字式のままで解くことができます。

サンプルデータファイルの内容

@bat1 を実行して、出力された周波数特性のファイル bpf2b.d を開いて、Excel でグラフを作成します。

bpf2b.d の内容

```
1 21
500
3 -43.4018
0 0
550
3 -40.4768
0 0
600
3 -37.4586
0 0
650
3 -34.2651
0 0
700
3 -30.7908
0 0
750
3 -26.8835
0 0
800
3 -22.2964
0 0
850
3 -16.5752
0 0
900
3 -8.81776
0 0
950
3 -0.0784636
0 0
1000
```

サンプルデータファイルの内容

3 -0.00152308

0 0

1050

3 0.0692258

0 0

1100

3 -7.20044

0 0

1150

3 -13.9236

0 0

1200

3 -18.7066

0 0

1250

3 -22.3415

0 0

1300

3 -25.2565

0 0

1350

3 -27.6829

0 0

1400

3 -29.7572

0 0

1450

3 -31.5666

0 0

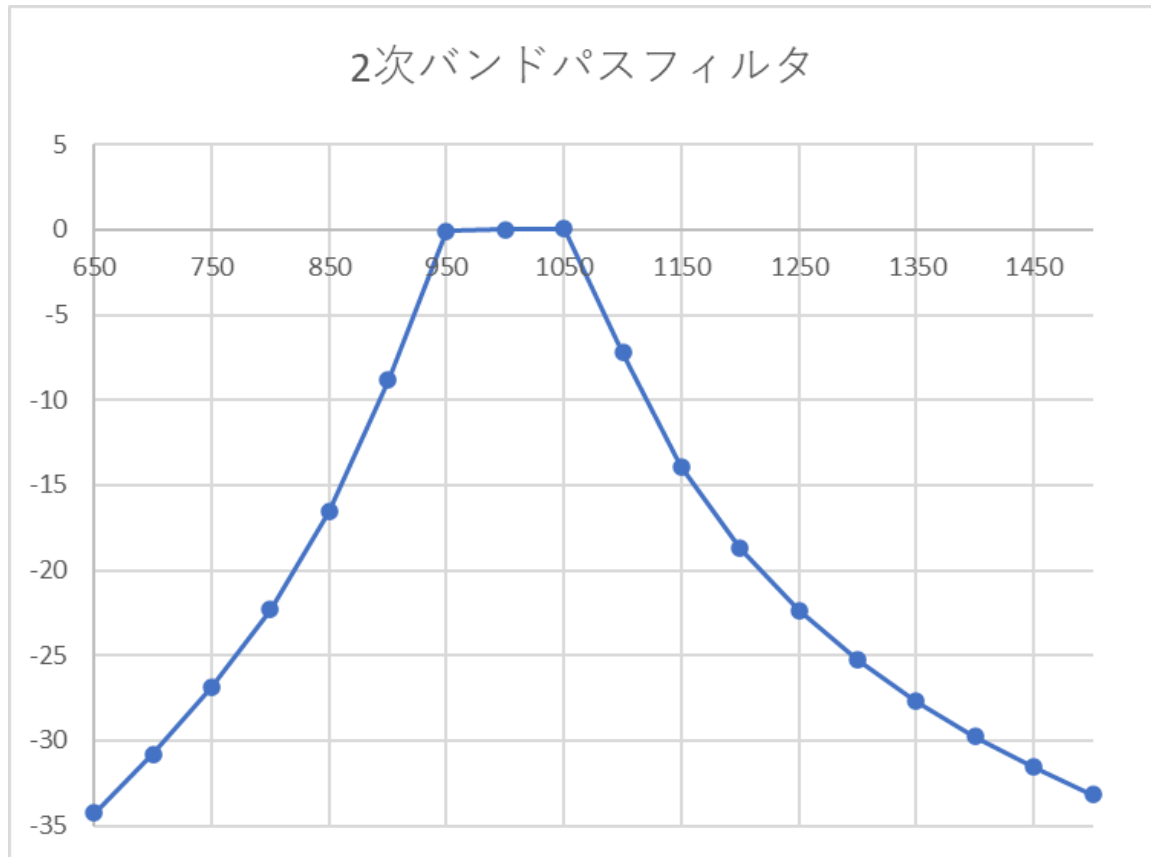
1500

3 -33.1698

0 0

サンプルデータファイルの内容

Excel でグラフを作成すると、



周波数が 950Hz から 1050Hz の信号はそのまま出力されるが、それ以外の周波数の信号は除去される特性を示している。

サンプル回路

サンプル回路

s1.sb、s2.sb などのバッチファイルを実行すると、回路を設計するためにどのようにコマンドを利用すれば良いのかを知ることができます。

バッチファイルを実行すると、計算結果が s1-l.d などのファイルとして保存されます。

トランジスタを使用した回路の設計練習用にサンプル回路を準備してあります。

全てのサンプル回路は、回路リストの形ではなくバッチファイルの形で準備してあります。これは、設計の各ステップごとにどのコマンドをどのように使用すれば回路の設計が出来るかを示すためです。

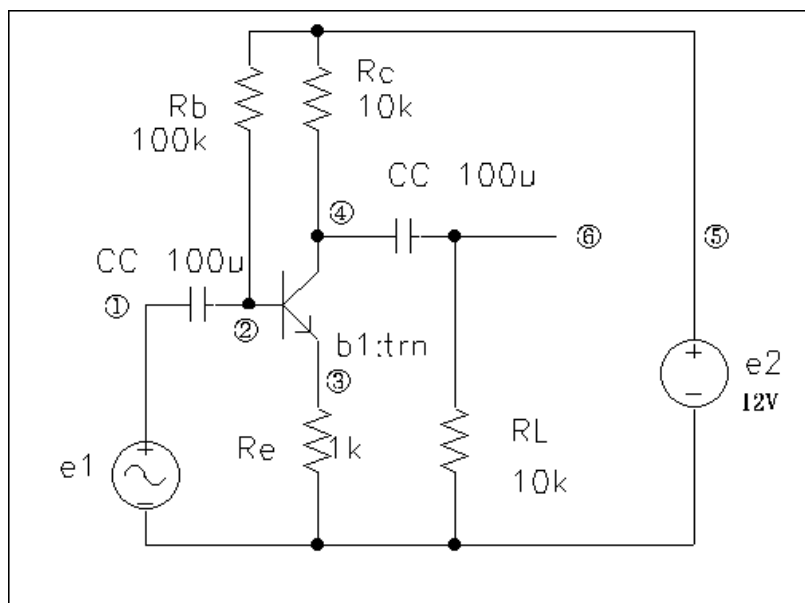


図 s1.sb で設計する回路図

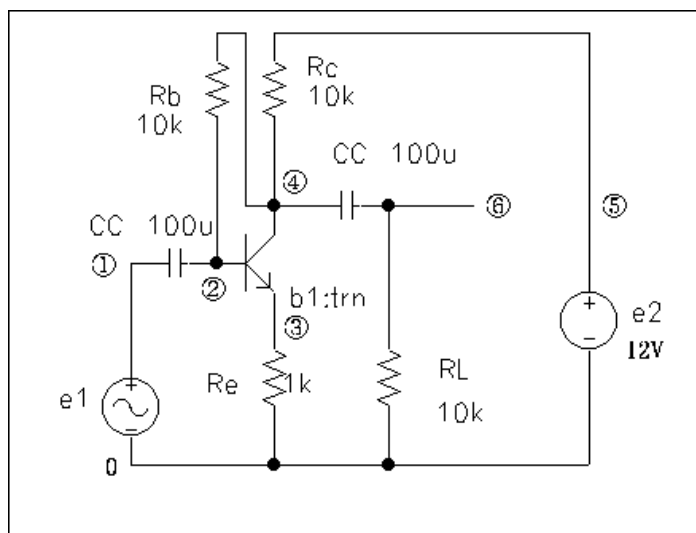


図 s2.sb で設計する回路図

サンプル回路

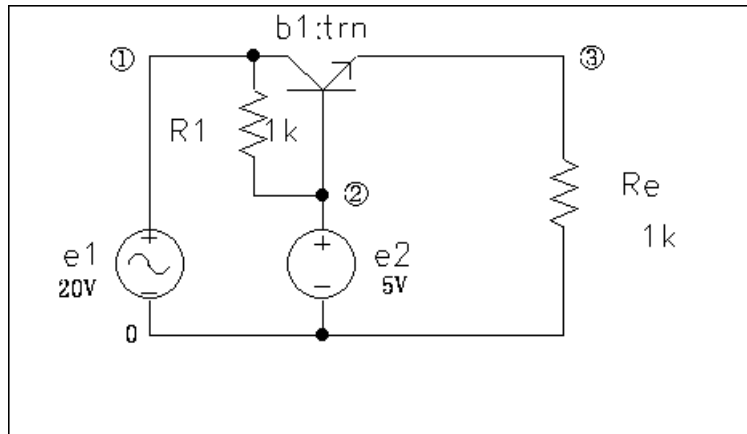


図 s3.sb で設計する回路図

サンプル回路の概要説明

1. サンプル回路の 1 と 2 はエミッタ接地の増幅回路である。サンプル回路の 3 はエミッタホロワの増幅回路である。
2. サンプル回路の 1 と 2 の設計目標は
 1. 無信号時のノード 4 の動作点を電源電圧（12V）の半分の 6V にする。
 2. 周波数 1KHz でのゲインを -10 倍とする。
 3. コレクタ抵抗と負荷抵抗は 10K とする。
 4. カップリングコンデンサは 100uF とする。

従って、上記の目標を達成するために変更可能な素子は R_b と R_e の 2 種類となる。ここで、 R_b は直流動作点を決定する素子であり、 R_e はゲインを決定する素子である。しかし、これらは相互に若干の影響を与えるので数回交互に調整を行なう必要がある。

ここでは、まず 2. のゲインの目標達成のために R_e を変化させてゲインが -10 となる R_e を探す所から始めている。この時、コマンドラインから $f = 1000$ と入力して周波数の設定を行なっている。また、計算結果を表示するノードは 4 を指定している。

ゲインの確認には、コマンド `/ac` を使用する。入力信号源は `e1` であり、値は `1` と設定する。変化させる素子名は `Re` であり、最低値は `100`、最高値は `1000`、ステップは `100` と指定するとシミュレーションが開始される。

サンプル回路

バッチファイル @s1.sb を実行します。

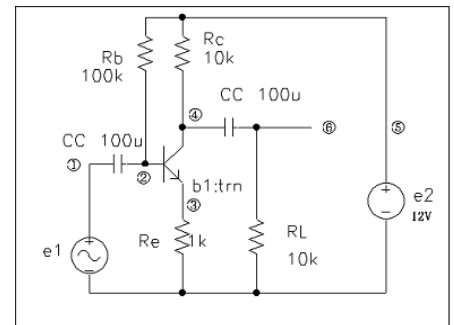
「キー入力待ち？」で、画面をスクロールすると回路データの入力画面を確認することができます。確認したら、「キー入力待ち？」で「Enter」を入力します。

表示される画面を読んだら「Enter」を入力して、先へ進んで下さい。

最初に表示される計算結果は次の画面です。

```
/ac
データファイル名は ? s1-1
ac 入力信号源名は ? e1
値は ? 1

値を変化させる素子名は ? re
ac re 最低値 ? 100
ac re 最高値 ? 1000
ac re ステップ ? 100
[22me1[1 j 0 ]·[0m
re[100 ] f[1,000 ]
x4 [ -49.3282 j -0.0046 ] abs[ 49.3282] arg[ -179.9947]
re[200 ] f[1,000 ]
x4 [ -24.7082 j -3.702932e-004] abs[ 24.7082] arg[ -179.9991]
re[300 ] f[1,000 ]
x4 [ -16.4820 j 1.845695e-004] abs[ 16.4820] arg[ 179.9994]
re[400 ] f[1,000 ]
x4 [ -12.3652 j 3.005037e-004] abs[ 12.3652] arg[ 179.9986]
re[500 ] f[1,000 ]
x4 [ -9.8939 j 3.182741e-004] abs[ 9.8939] arg[ 179.9982]
re[600 ] f[1,000 ]
x4 [ -8.2459 j 3.085159e-004] abs[ 8.2459] arg[ 179.9979]
re[700 ] f[1,000 ]
x4 [ -7.0685 j 2.909557e-004] abs[ 7.0685] arg[ 179.9976]
re[800 ] f[1,000 ]
x4 [ -6.1853 j 2.719908e-004] abs[ 6.1853] arg[ 179.9975]
re[900 ] f[1,000 ]
x4 [ -5.4984 j 2.538051e-004] abs[ 5.4984] arg[ 179.9974]
re[1000] f[1,000 ]
x4 [ -4.9487 j 2.370918e-004] abs[ 4.9487] arg[ 179.9973]
B?
```



ここで、注目するのは -49.3281 の位置である。

これが、 -10 に来るだけ近づく様な R_e を探すことである。

```
値を変化させる素子名は ? re
ac re 最低値 ? 400
ac re 最高値 ? 500
ac re ステップ ? 10
e1[1 j 0 ]
re[400 ] f[1,000 ]
x4 [ -12.3652 j 3.005037e-004] abs[ 12.3652] arg[ 179.9986]
re[410 ] f[1,000 ]
x4 [ -12.0638 j 3.047518e-004] abs[ 12.0638] arg[ 179.9986]
re[420 ] f[1,000 ]
x4 [ -11.7768 j 3.082603e-004] abs[ 11.7768] arg[ 179.9985]
re[430 ] f[1,000 ]
x4 [ -11.5032 j 3.111173e-004] abs[ 11.5032] arg[ 179.9985]
re[440 ] f[1,000 ]
x4 [ -11.2420 j 3.134000e-004] abs[ 11.2420] arg[ 179.9984]
re[450 ] f[1,000 ]
x4 [ -10.9923 j 3.151753e-004] abs[ 10.9923] arg[ 179.9984]
re[460 ] f[1,000 ]
x4 [ -10.7536 j 3.165023e-004] abs[ 10.7536] arg[ 179.9983]
re[470 ] f[1,000 ]
x4 [ -10.5249 j 3.174327e-004] abs[ 10.5249] arg[ 179.9983]
re[480 ] f[1,000 ]
x4 [ -10.3058 j 3.180122e-004] abs[ 10.3058] arg[ 179.9982]
re[490 ] f[1,000 ]
x4 [ -10.0957 j 3.182807e-004] abs[ 10.0957] arg[ 179.9982]
re[500 ] f[1,000 ]
x4 [ -9.8939 j 3.182741e-004] abs[ 9.8939] arg[ 179.9982]
B? とりあえず re=490 として、次に直流動作点を決定する
ノード4 が 6v となるように rb を決定する
キー入力待ち？
```

サンプル回路

最適な値が決定したら、回路の素子値を変更します。

```

B? re=490
  490 j 0
B? /dpart
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 3] parts[ re ] value[ 490 j 0 ]
left[ 0] right[ 5] parts[ e2 ] value[ 12 j 0 ]
left[ 0] right[ 6] parts[ rl ] value[ 1e+004 j 0 ]
left[ 1] right[ 2] parts[ cc ] value[ 0.0001 j 0 ]
left[ 2] right[ 5] parts[ rb ] value[ 1e+005 j 0 ]
left[ 2] right[ 10] parts[ rlb1 ] value[ 0.1 j 0 ]
left[ 3] right[ 7] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 4] right[ 5] parts[ rc ] value[ 1e+004 j 0 ]
left[ 4] right[ 6] parts[ cc ] value[ 0.0001 j 0 ]
left[ 4] right[ 9] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 7] right[ 8] parts[ reb1 ] value[ 26 j 0 ]
left[ 7] right[ 9] parts[ kb1 ] value[ 100 j 0 ]
@ master[reb1 ] left[ 7] right[ 8]
left[ 8] right[ 10] parts[ ebb1 ] value[ 0.6 j 0 ]
最大の素子番号 = 14
最大のノード番号 = 10
B? 周波数を 0 とする
キー入力待ち?

```

また、直流動作点の決定においてはコマンド `/para` を使用して、素子名 `Rb` を変化させるが、このときには先程の位置の数字が出来るだけ 6 に近づく `Rb` を探すことになる。`/para` を使用する時には、`f=0` と設定することにも注意すべき点である。

```

B? /para
データファイル名は ? s1-3
値を変化させる素子名は ? rb
para rb 最低値 ? 10k
para rb 最高値 ? 100k
para rb ステップ ? 10k
rb[1e+004 ] f[0 ]
x4 [ -179.5123 j 0 ] abs[ 179.5123] arg[ 180]
rb[2e+004 ] f[0 ]
x4 [ -151.9670 j 0 ] abs[ 151.9670] arg[ 180]
rb[3e+004 ] f[0 ]
x4 [ -131.3490 j 0 ] abs[ 131.3490] arg[ 180]
rb[4e+004 ] f[0 ]
x4 [ -115.3370 j 0 ] abs[ 115.3370] arg[ 180]
rb[5e+004 ] f[0 ]
x4 [ -102.5427 j 0 ] abs[ 102.5427] arg[ 180]
rb[6e+004 ] f[0 ]
x4 [ -92.0847 j 0 ] abs[ 92.0847] arg[ 180]
rb[7e+004 ] f[0 ]
x4 [ -83.3766 j 0 ] abs[ 83.3766] arg[ 180]
rb[8e+004 ] f[0 ]
x4 [ -76.0131 j 0 ] abs[ 76.0131] arg[ 180]
rb[9e+004 ] f[0 ]
x4 [ -69.7051 j 0 ] abs[ 69.7051] arg[ 180]
rb[1e+005 ] f[0 ]
x4 [ -64.2408 j 0 ] abs[ 64.2408] arg[ 180]
B? ノード4の値がすべて、マイナスなので rb をさらに大きくして調べる
rb 100k から 1M まで調べる
x4の動作点電圧の変化をs1-4.dに出力する
キー入力待ち?

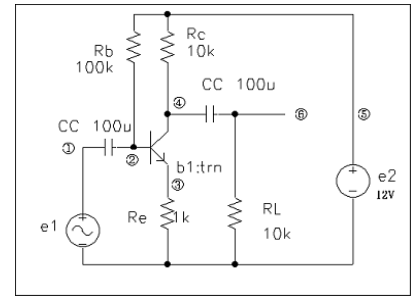
```

サンプル回路

```

データファイル名は ? s1-65
値を変化させる素子名は ? rb
para rb 最低値 ? 1m
para rb 最高値 ? 2m
para rb ステップ ? 100k
rb[1e+006 ] f[0 ]
x4 [ 1.1380 j 0 ] abs[ 1.1380] arg[ 0]
rb[1.1e+006 ] f[0 ]
x4 [ 2.0829 j 0 ] abs[ 2.0829] arg[ 0]
rb[1.2e+006 ] f[0 ]
x4 [ 2.8765 j 0 ] abs[ 2.8765] arg[ 0]
rb[1.3e+006 ] f[0 ]
x4 [ 3.5526 j 0 ] abs[ 3.5526] arg[ 0]
rb[1.4e+006 ] f[0 ]
x4 [ 4.1354 j 0 ] abs[ 4.1354] arg[ 0]
rb[1.5e+006 ] f[0 ]
x4 [ 4.6429 j 0 ] abs[ 4.6429] arg[ 0]
rb[1.6e+006 ] f[0 ]
x4 [ 5.0889 j 0 ] abs[ 5.0889] arg[ 0]
rb[1.7e+006 ] f[0 ]
x4 [ 5.4840 j 0 ] abs[ 5.4840] arg[ 0]
rb[1.8e+006 ] f[0 ]
x4 [ 5.8363 j 0 ] abs[ 5.8363] arg[ 0]
rb[1.9e+006 ] f[0 ]
x4 [ 6.1524 j 0 ] abs[ 6.1524] arg[ 0]
rb[2e+006 ] f[0 ]
x4 [ 6.4377 j 0 ] abs[ 6.4377] arg[ 0]
B? rb が 1.8M と 1.9M の間をさらに調べる
x4の動作点電圧の変化をs1-7.dに出力する
キー入力待ち?

```



```

B? /para
データファイル名は ? s1-7
値を変化させる素子名は ? rb
para rb 最低値 ? 1.8m
para rb 最高値 ? 1.9m
para rb ステップ ? 10k
rb[1.8e+006 ] f[0 ]
x4 [ 5.8363 j 0 ] abs[ 5.8363] arg[ 0]
rb[1.81e+006 ] f[0 ]
x4 [ 5.8694 j 0 ] abs[ 5.8694] arg[ 0]
rb[1.82e+006 ] f[0 ]
x4 [ 5.9022 j 0 ] abs[ 5.9022] arg[ 0]
rb[1.83e+006 ] f[0 ]
x4 [ 5.9346 j 0 ] abs[ 5.9346] arg[ 0]
rb[1.84e+006 ] f[0 ]
x4 [ 5.9667 j 0 ] abs[ 5.9667] arg[ 0]
rb[1.85e+006 ] f[0 ]
x4 [ 5.9985 j 0 ] abs[ 5.9985] arg[ 0]
rb[1.86e+006 ] f[0 ]
x4 [ 6.0299 j 0 ] abs[ 6.0299] arg[ 0]
rb[1.87e+006 ] f[0 ]
x4 [ 6.0610 j 0 ] abs[ 6.0610] arg[ 0]
rb[1.88e+006 ] f[0 ]
x4 [ 6.0918 j 0 ] abs[ 6.0918] arg[ 0]
rb[1.89e+006 ] f[0 ]
x4 [ 6.1223 j 0 ] abs[ 6.1223] arg[ 0]
rb[1.9e+006 ] f[0 ]
x4 [ 6.1524 j 0 ] abs[ 6.1524] arg[ 0]
B? rb=1.85M と決定する
キー入力待ち?

```

s1.sb を実行すると、

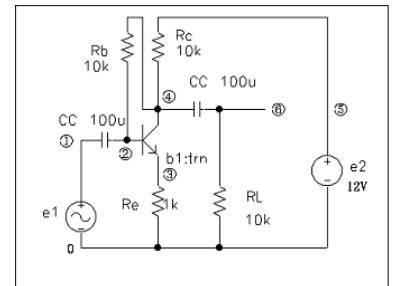
1. 直流動作点は $R_b=1.85M$ によって 6 V になり、
2. 交流ゲインは $R_e=495$ オームによって、 -10 倍となることが分かる。

サンプル回路

同様に、バッチファイル s2. sb を実行すると、

```
ac 入力信号源名は ? e1
値は ? 1

値を変化させる素子名は ? re
ac re 最低値 ? 100
ac re 最高値 ? 1000
ac re ステップ ? 100
e1[1      ] j 0      ]
re[100    ] f[1,000  ]
x4 [ -32.5511 j -0.1772 ] abs[ 32.5516] arg[ -179.6881]
re[200    ] f[1,000  ]
x4 [ -16.1387 j -0.0444 ] abs[ 16.1387] arg[ -179.8422]
re[300    ] f[1,000  ]
x4 [ -10.6546 j -0.0198 ] abs[ 10.6546] arg[ -179.8938]
re[400    ] f[1,000  ]
x4 [ -7.9101 j -0.0111 ] abs[ 7.9101] arg[ -179.9195]
re[500    ] f[1,000  ]
x4 [ -6.2626 j -0.0071 ] abs[ 6.2626] arg[ -179.9350]
re[600    ] f[1,000  ]
x4 [ -5.1639 j -0.0049 ] abs[ 5.1639] arg[ -179.9453]
re[700    ] f[1,000  ]
x4 [ -4.3790 j -0.0036 ] abs[ 4.3790] arg[ -179.9527]
re[800    ] f[1,000  ]
x4 [ -3.7902 j -0.0028 ] abs[ 3.7902] arg[ -179.9582]
re[900    ] f[1,000  ]
x4 [ -3.3322 j -0.0022 ] abs[ 3.3322] arg[ -179.9625]
re[1000   ] f[1,000  ]
x4 [ -2.9658 j -0.0018 ] abs[ 2.9658] arg[ -179.9660]
B? re: 300 から 400 の間でゲインが -10 となる
```



```
ac re 最低値 ? 300
ac re 最高値 ? 400
ac re ステップ ? 10
e1[1      ] j 0      ]
re[300    ] f[1,000  ]
x4 [ -10.6546 j -0.0198 ] abs[ 10.6546] arg[ -179.8938]
re[310    ] f[1,000  ]
x4 [ -10.3006 j -0.0185 ] abs[ 10.3006] arg[ -179.8971]
re[320    ] f[1,000  ]
x4 [ -9.9686 j -0.0174 ] abs[ 9.9686] arg[ -179.9002]
re[330    ] f[1,000  ]
x4 [ -9.6568 j -0.0163 ] abs[ 9.6568] arg[ -179.9031]
re[340    ] f[1,000  ]
x4 [ -9.3633 j -0.0154 ] abs[ 9.3633] arg[ -179.9059]
re[350    ] f[1,000  ]
x4 [ -9.0865 j -0.0145 ] abs[ 9.0865] arg[ -179.9085]
re[360    ] f[1,000  ]
x4 [ -8.8251 j -0.0137 ] abs[ 8.8251] arg[ -179.9109]
re[370    ] f[1,000  ]
x4 [ -8.5778 j -0.0130 ] abs[ 8.5778] arg[ -179.9133]
re[380    ] f[1,000  ]
x4 [ -8.3435 j -0.0123 ] abs[ 8.3436] arg[ -179.9155]
re[390    ] f[1,000  ]
x4 [ -8.1213 j -0.0117 ] abs[ 8.1213] arg[ -179.9176]
re[400    ] f[1,000  ]
x4 [ -7.9101 j -0.0111 ] abs[ 7.9101] arg[ -179.9195]
B? とりあえず re=310 として、次に直流動作点を決定する
ノード4 が 6v となるように rb を決定する
キー入力待ち? -
```

サンプル回路

```

データファイル名は ? s2-3
値を変化させる素子名は ? rb
para rb 最低値 ? 10k
para rb 最高値 ? 100k
para rb ステップ ? 10k
rb[1e+004 ] f[0 ]
x4 [ 1.0483 j 0 ] abs[ 1.0483] arg[ 0]
rb[2e+004 ] f[0 ]
x4 [ 1.1515 j 0 ] abs[ 1.1515] arg[ 0]
rb[3e+004 ] f[0 ]
x4 [ 1.2528 j 0 ] abs[ 1.2528] arg[ 0]
rb[4e+004 ] f[0 ]
x4 [ 1.3522 j 0 ] abs[ 1.3522] arg[ 0]
rb[5e+004 ] f[0 ]
x4 [ 1.4497 j 0 ] abs[ 1.4497] arg[ 0]
rb[6e+004 ] f[0 ]
x4 [ 1.5455 j 0 ] abs[ 1.5455] arg[ 0]
rb[7e+004 ] f[0 ]
x4 [ 1.6396 j 0 ] abs[ 1.6396] arg[ 0]
rb[8e+004 ] f[0 ]
x4 [ 1.7320 j 0 ] abs[ 1.7320] arg[ 0]
rb[9e+004 ] f[0 ]
x4 [ 1.8227 j 0 ] abs[ 1.8227] arg[ 0]
rb[1e+005 ] f[0 ]
x4 [ 1.9119 j 0 ] abs[ 1.9119] arg[ 0]
B? ノード4の値がすべて、6V 以下なので rb をさらに大きくして調べる
rb 100k から 1M まで調べる
s2-4を出力する
キー入力待ち？

```

```

B? /para
データファイル名は ? s2-4
値を変化させる素子名は ? rb
para rb 最低値 ? 100k
para rb 最高値 ? 1m
para rb ステップ ? 100k
rb[1e+005 ] f[0 ]
x4 [ 1.9119 j 0 ] abs[ 1.9119] arg[ 0]
rb[2e+005 ] f[0 ]
x4 [ 2.7246 j 0 ] abs[ 2.7246] arg[ 0]
rb[3e+005 ] f[0 ]
x4 [ 3.4161 j 0 ] abs[ 3.4161] arg[ 0]
rb[4e+005 ] f[0 ]
x4 [ 4.0116 j 0 ] abs[ 4.0116] arg[ 0]
rb[5e+005 ] f[0 ]
x4 [ 4.5299 j 0 ] abs[ 4.5299] arg[ 0]
rb[6e+005 ] f[0 ]
x4 [ 4.9850 j 0 ] abs[ 4.9850] arg[ 0]
rb[7e+005 ] f[0 ]
x4 [ 5.3879 j 0 ] abs[ 5.3879] arg[ 0]
rb[8e+005 ] f[0 ]
x4 [ 5.7470 j 0 ] abs[ 5.7470] arg[ 0]
rb[9e+005 ] f[0 ]
x4 [ 6.0691 j 0 ] abs[ 6.0691] arg[ 0]
rb[1e+006 ] f[0 ]
x4 [ 6.3596 j 0 ] abs[ 6.3596] arg[ 0]
B? rb が 800K と 900K の間で、ノード4が 6v となるので、
とりあえず rb=800K と決定する
キー入力待ち？

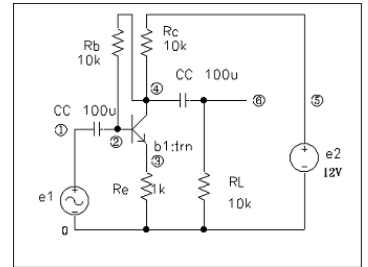
```


サンプル回路

```

値を変化させる素子名は ? re
ac re 最低値 ? 400
ac re 最高値 ? 500
ac re ステップ ? 10
e1[1      ] j 0      ]
re[400    ] f[1,000  ]
x4 [ -12.2821 j 1.633487e-004] abs[ 12.2821] arg[ 179.9992]
re[410    ] f[1,000  ]
x4 [ -11.9827 j 1.779979e-004] abs[ 11.9827] arg[ 179.9991]
re[420    ] f[1,000  ]
x4 [ -11.6975 j 1.910844e-004] abs[ 11.6975] arg[ 179.9991]
re[430    ] f[1,000  ]
x4 [ -11.4255 j 2.027757e-004] abs[ 11.4255] arg[ 179.9990]
re[440    ] f[1,000  ]
x4 [ -11.1659 j 2.132196e-004] abs[ 11.1659] arg[ 179.9989]
re[450    ] f[1,000  ]
x4 [ -10.9179 j 2.225459e-004] abs[ 10.9179] arg[ 179.9988]
re[460    ] f[1,000  ]
x4 [ -10.6806 j 2.308688e-004] abs[ 10.6806] arg[ 179.9988]
re[470    ] f[1,000  ]
x4 [ -10.4534 j 2.382899e-004] abs[ 10.4534] arg[ 179.9987]
re[480    ] f[1,000  ]
x4 [ -10.2356 j 2.448990e-004] abs[ 10.2356] arg[ 179.9986]
re[490    ] f[1,000  ]
x4 [ -10.0267 j 2.507757e-004] abs[ 10.0267] arg[ 179.9986]
re[500    ] f[1,000  ]
x4 [  -9.8262 j 2.559915e-004] abs[  9.8262] arg[ 179.9985]
B? re=490 に決定する
キー入力待ち? _

```



```

値を変化させる素子名は ? re
ac re 最低値 ? 490
ac re 最高値 ? 500
ac re ステップ ? 1
e1[1      ] j 0      ]
re[490    ] f[1,000  ]
x4 [ -10.0267 j 2.507757e-004] abs[ 10.0267] arg[ 179.9986]
re[491    ] f[1,000  ]
x4 [ -10.0063 j 2.513260e-004] abs[ 10.0063] arg[ 179.9986]
re[492    ] f[1,000  ]
x4 [  -9.9860 j 2.518697e-004] abs[  9.9860] arg[ 179.9986]
re[493    ] f[1,000  ]
x4 [  -9.9657 j 2.524069e-004] abs[  9.9657] arg[ 179.9985]
re[494    ] f[1,000  ]
x4 [  -9.9456 j 2.529376e-004] abs[  9.9456] arg[ 179.9985]
re[495    ] f[1,000  ]
x4 [  -9.9255 j 2.534621e-004] abs[  9.9255] arg[ 179.9985]
re[496    ] f[1,000  ]
x4 [  -9.9055 j 2.539802e-004] abs[  9.9055] arg[ 179.9985]
re[497    ] f[1,000  ]
x4 [  -9.8855 j 2.544923e-004] abs[  9.8855] arg[ 179.9985]
re[498    ] f[1,000  ]
x4 [  -9.8657 j 2.549980e-004] abs[  9.8657] arg[ 179.9985]
re[499    ] f[1,000  ]
x4 [  -9.8459 j 2.554977e-004] abs[  9.8459] arg[ 179.9985]
re[500    ] f[1,000  ]
x4 [  -9.8262 j 2.559915e-004] abs[  9.8262] arg[ 179.9985]
B? 最終決定 re=491 とする
キー入力待ち?

```

サンプル回路

```

B? /para
データファイル名は ? s2-8
値を変化させる素子名は ? rb
para rb 最低値 ? 800k
para rb 最高値 ? 900k
para rb ステップ ? 10k
rb[8e+005 ] f[0 ]
x4 [ 5.8084 j 0 ] abs[ 5.8084] arg[ 0]
rb[8.1e+005 ] f[0 ]
x4 [ 5.8416 j 0 ] abs[ 5.8416] arg[ 0]
rb[8.2e+005 ] f[0 ]
x4 [ 5.8743 j 0 ] abs[ 5.8743] arg[ 0]
rb[8.3e+005 ] f[0 ]
x4 [ 5.9067 j 0 ] abs[ 5.9067] arg[ 0]
rb[8.4e+005 ] f[0 ]
x4 [ 5.9388 j 0 ] abs[ 5.9388] arg[ 0]
rb[8.5e+005 ] f[0 ]
x4 [ 5.9706 j 0 ] abs[ 5.9706] arg[ 0]
rb[8.6e+005 ] f[0 ]
x4 [ 6.0020 j 0 ] abs[ 6.0020] arg[ 0]
rb[8.7e+005 ] f[0 ]
x4 [ 6.0330 j 0 ] abs[ 6.0330] arg[ 0]
rb[8.8e+005 ] f[0 ]
x4 [ 6.0638 j 0 ] abs[ 6.0638] arg[ 0]
rb[8.9e+005 ] f[0 ]
x4 [ 6.0943 j 0 ] abs[ 6.0943] arg[ 0]
rb[9e+005 ] f[0 ]
x4 [ 6.1244 j 0 ] abs[ 6.1244] arg[ 0]
B? 最終決定 rb=860K とする
rb=860k

```

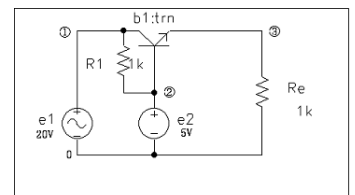
1. 直流動作点は $R_b=860K$ によって 6 V になり、
2. 交流ゲインは $R_e=491$ オームによって、 -10 倍となることが分かる。

バッチファイル s3.sb を実行すると、

```

B? /para
データファイル名は ? s3-1
値を変化させる素子名は ? e1
para e1 最低値 ? 5
para e1 最高値 ? 20
para e1 ステップ ? 5
e1[5 ] f[0 ]
x3 [ 4.3984 j 0 ] abs[ 4.3984] arg[ 0]
e1[10 ] f[0 ]
x3 [ 4.3984 j 0 ] abs[ 4.3984] arg[ 0]
e1[15 ] f[0 ]
x3 [ 4.3984 j 0 ] abs[ 4.3984] arg[ 0]
e1[20 ] f[0 ]
x3 [ 4.3984 j 0 ] abs[ 4.3984] arg[ 0]
B? e1 が変化しても、出力電圧はほとんど変化しないことが分かる
次は、e2 を変化させてみる 1 から 10 V
s3-2を出力する
キー入力待ち? _

```

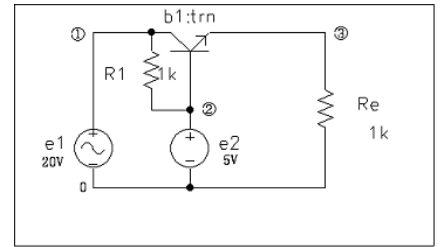


サンプル回路

```

B? /para
データファイル名は ? s3-2
値を変化させる素子名は ? e2
para e2 最低値 ? 1
para e2 最高値 ? 10
para e2 ステップ ? 1
e2[1] ] f[0 ] ] abs[ 0.3999] arg[ 0]
x3 [ 0.3999 j 0 ] ] abs[ 0.3999] arg[ 0]
e2[2] ] f[0 ] ] abs[ 1.3995] arg[ 0]
x3 [ 1.3995 j 0 ] ] abs[ 1.3995] arg[ 0]
e2[3] ] f[0 ] ] abs[ 2.3991] arg[ 0]
x3 [ 2.3991 j 0 ] ] abs[ 2.3991] arg[ 0]
e2[4] ] f[0 ] ] abs[ 3.3988] arg[ 0]
x3 [ 3.3988 j 0 ] ] abs[ 3.3988] arg[ 0]
e2[5] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
e2[6] ] f[0 ] ] abs[ 5.3981] arg[ 0]
x3 [ 5.3981 j 0 ] ] abs[ 5.3981] arg[ 0]
e2[7] ] f[0 ] ] abs[ 6.3977] arg[ 0]
x3 [ 6.3977 j 0 ] ] abs[ 6.3977] arg[ 0]
e2[8] ] f[0 ] ] abs[ 7.3973] arg[ 0]
x3 [ 7.3973 j 0 ] ] abs[ 7.3973] arg[ 0]
e2[9] ] f[0 ] ] abs[ 8.3970] arg[ 0]
x3 [ 8.3970 j 0 ] ] abs[ 8.3970] arg[ 0]
e2[10] ] f[0 ] ] abs[ 9.3966] arg[ 0]
x3 [ 9.3966 j 0 ] ] abs[ 9.3966] arg[ 0]
B? 出力電圧は e2=0.6 V になっていることが分かる
負荷抵抗 re を変化させてみる 100 から 1k
s3-3を出力する
キー入力待ち? _

```



```

/para
データファイル名は ? s3-6
値を変化させる素子名は ? r1
para r1 最低値 ? 1k
para r1 最高値 ? 10k
para r1 ステップ ? 1k
r1[1000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[2000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[3000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[4000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[5000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[6000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[7000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[8000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[9000] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
r1[1e+004] ] f[0 ] ] abs[ 4.3984] arg[ 0]
x3 [ 4.3984 j 0 ] ] abs[ 4.3984] arg[ 0]
B? 以上で、この回路では e2 の値が出力電圧を決定していることが分かる。
この回路形式は、安定化電源の基本形として利用されている
また、この回路は出力インピーダンスが低いのでバッファアンプとして使われる
1 番のバッチファイル s3.sb を終了します。
? _

```

1. エミッタホロワ回路では、e1 の電圧が変化しても、エミッタ電圧はほとんど変化しない事が分かる。
2. エミッタホロワ回路では、エミッタ電圧はベース電圧 - 0.6 V 程度になることが分かる。

Sim.exe で使用する標準部品として準備してあるブロック素子について

Sim.exe で使用する標準部品として準備してあるブロック素子について

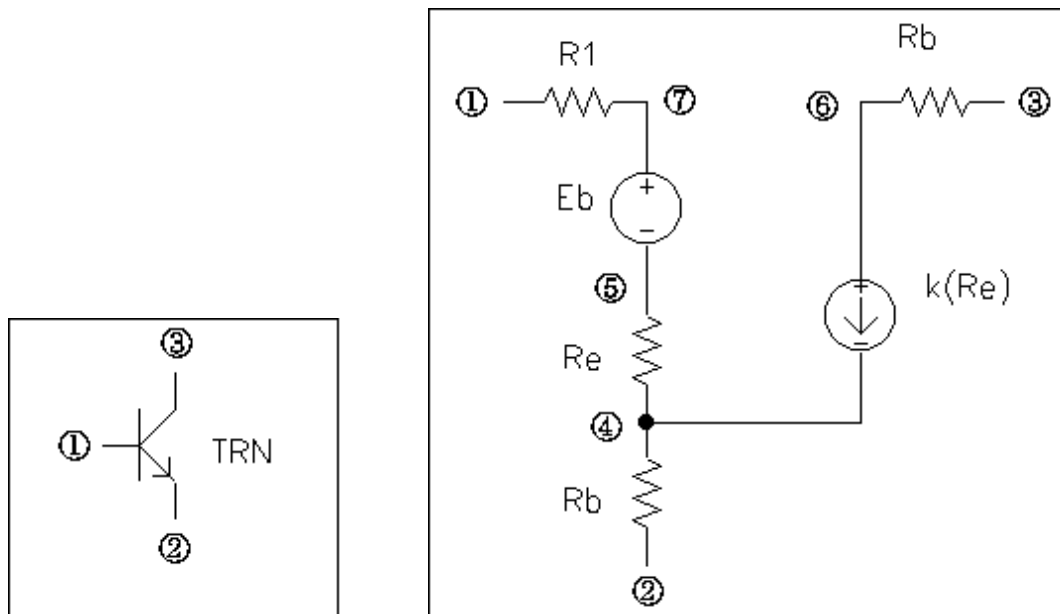


図 npn トランジスタの回路図記号と等価回路 TRN. CIR, TRN2. CIR

npn トランジスタは2種類のブロック素子を準備してある。ブロック素子名（ファイル名）はそれぞれ `trn. cir` と `trn2. cir` である。これらは、等価回路の素子の値が異なるだけである。

素子名	<code>trn. cir</code>	<code>trn2. cir</code>
R 1	0. 1	1 0 0
R e	2 6	2 6
R b	0. 1	0. 1
k (R e)	1 0 0	5 0
E b	0. 6	0. 7

R 1 はトランジスタの規格表では h_{ie} に相当します。k は同様に h_{fe} （または β ）に相当します。 $h_{ie} = 26\text{mV} / (I_c / h_{fe})$ により求められます。

例えば、 $I_c=10\text{mA}$, $h_{fe}=100$ なら $R1 = h_{ie} = 260$ オームとなります。

この等価回路では正確にダイオードの整流機能がシミュレーション出来ないため、実際のトランジスタの動作とは異なります。すなわち、ベース・エミッタ間が逆バイアスとなる入力電圧に対して、通常はベース電流はほとんど流れませんがこの等価回路では逆方向にも入力電圧に比例したベース電流が流れてしまいます。従って、適正な直流バイアス点を決定し小信号の入力に対してのシミュレーションにのみ用いるべきです。

図 npn トランジスタの回路図記号と等価回路 TRN. CIR, TRN2. CIR

Sim.exe で使用する標準部品として準備してあるブロック素子について

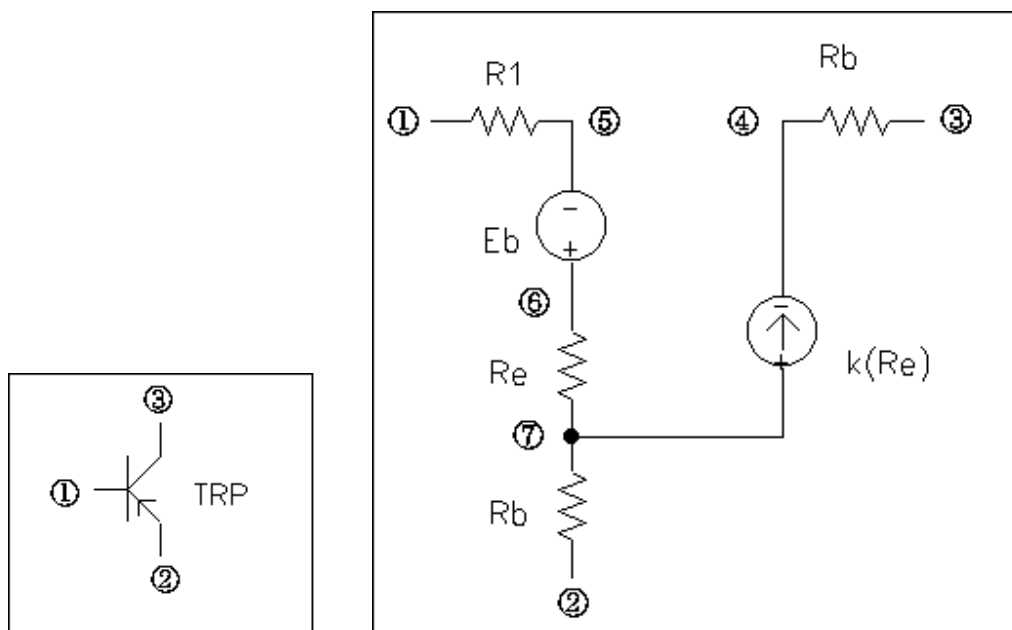


図 pnp トランジスタの回路図記号と等価回路 TRP. CIR, TRP2. CIR

pnp トランジスタは2種類のブロック素子を準備してある。ブロック素子名（ファイル名）はそれぞれ `trp.cir` と `trp2.cir` である。これらは、等価回路の素子の値が異なるだけである。

素子名	<code>trp.cir</code>	<code>trp2.cir</code>
R 1	0. 1	1 0 0
R e	2 6	2 6
R b	0. 1	0. 1
k (R e)	1 0 0	5 0
E b	0. 6	0. 7

R 1 はトランジスタの規格表では h_{ie} に相当します。k は同様に h_{fe} （または β ）に相当します。 $h_{ie} = 26mV / (I_c / h_{fe})$ により求められます。

例えば、 $I_c=10mA$, $h_{fe}=100$ なら $R1 = h_{ie} = 260$ オームとなります。

この等価回路では正確にダイオードの整流機能がシミュレーション出来ないため、実際のトランジスタの動作とは異なります。すなわち、ベース・エミッタ間が逆バイアスとなる入力電圧に対して、通常はベース電流はほとんど流れませんがこの等価回路では逆方向にも入力電圧に比例したベース電流が流れてしまいます。従って、適正な直流バイアス点を決定し小信号の入力に対してのシミュレーションにのみ用いるべきです。

図 pnp トランジスタの回路図記号と等価回路 TRP. CIR, TRP2. CIR

Sim.exe で使用する標準部品として準備してあるブロック素子について

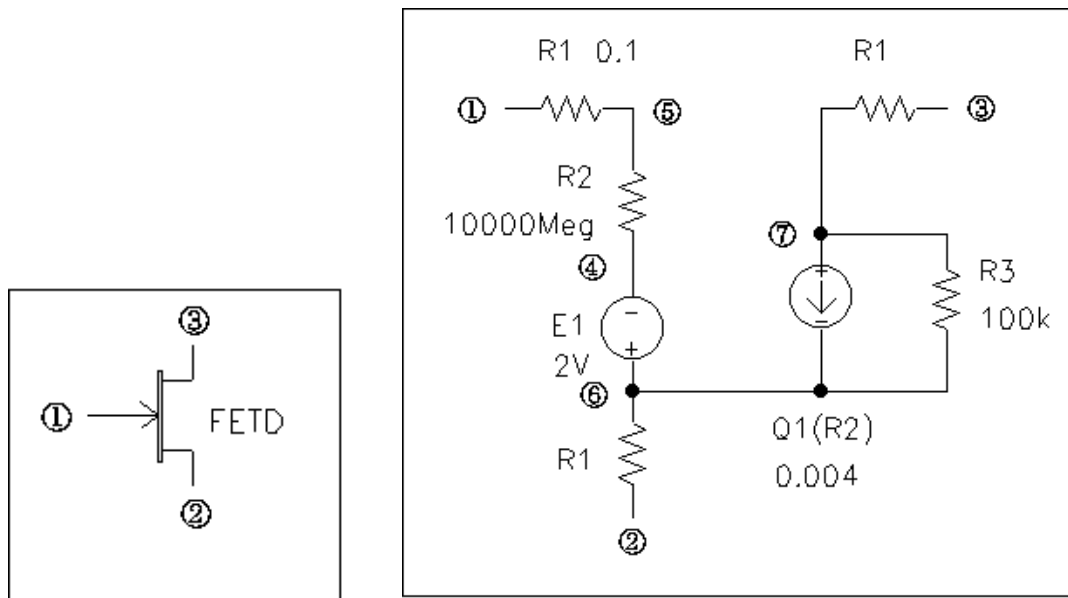


図 n型デプリーションタイプFETの回路図記号と等価回路 FETD.CIR

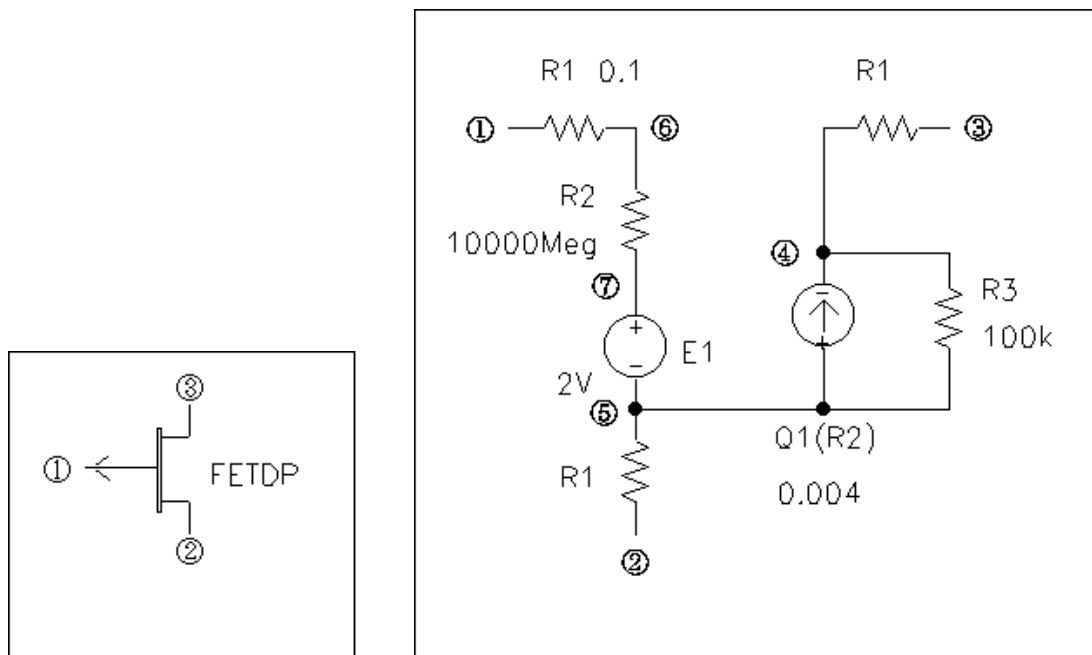


図 p型デプリーションタイプFETの回路図記号と等価回路 FETDP.CIR

実際にシミュレーションを行うときには、素子の規格表を参考にして、E1、R3、Q1の値を適正な値に変更してから使用する。
また、Q1の値については、動作点に合った値に設定する必要がある。

図 n型デプリーションタイプFETの回路図記号と等価回路 FETD.CIR

Sim.exe で使用する標準部品として準備してあるブロック素子について

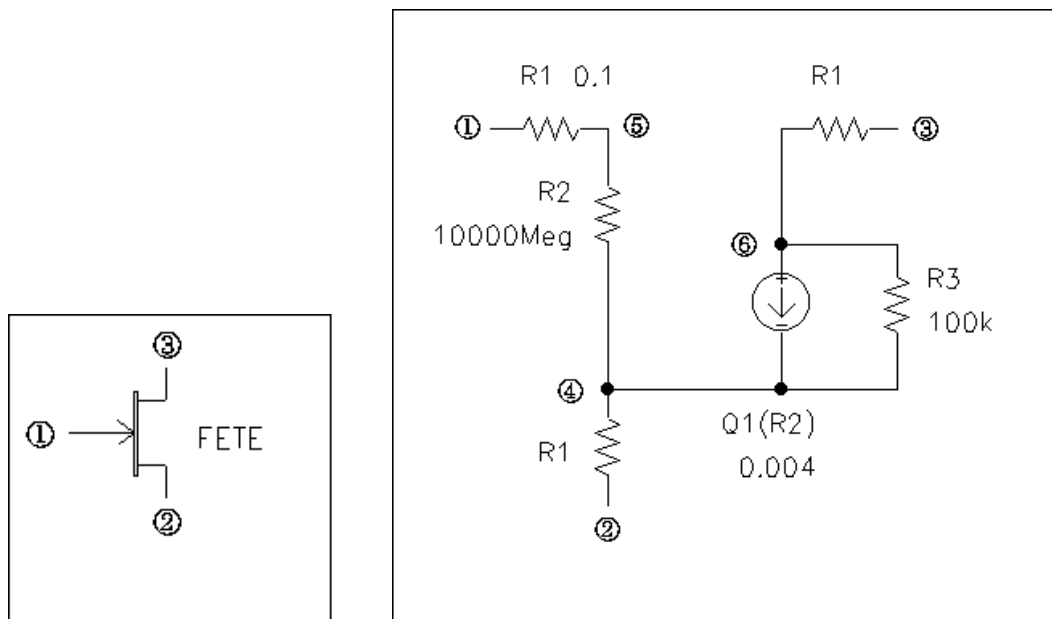


図 n型エンハンスメントタイプFETの回路図記号と等価回路 FETE. CIR

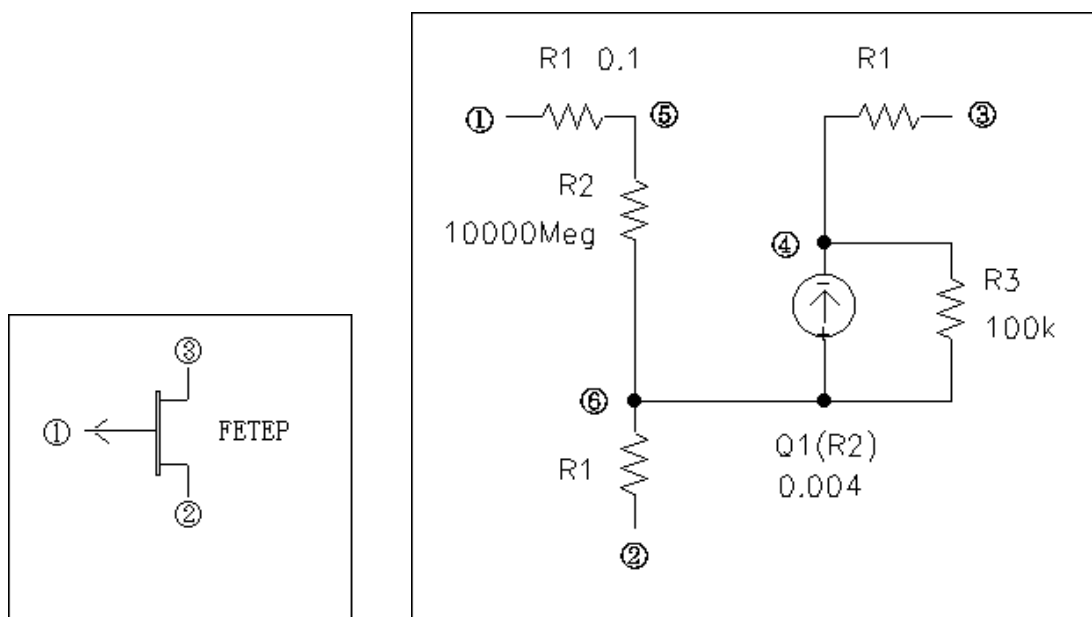


図 p型エンハンスメントタイプFETの回路図記号と等価回路 FETEP. CIR

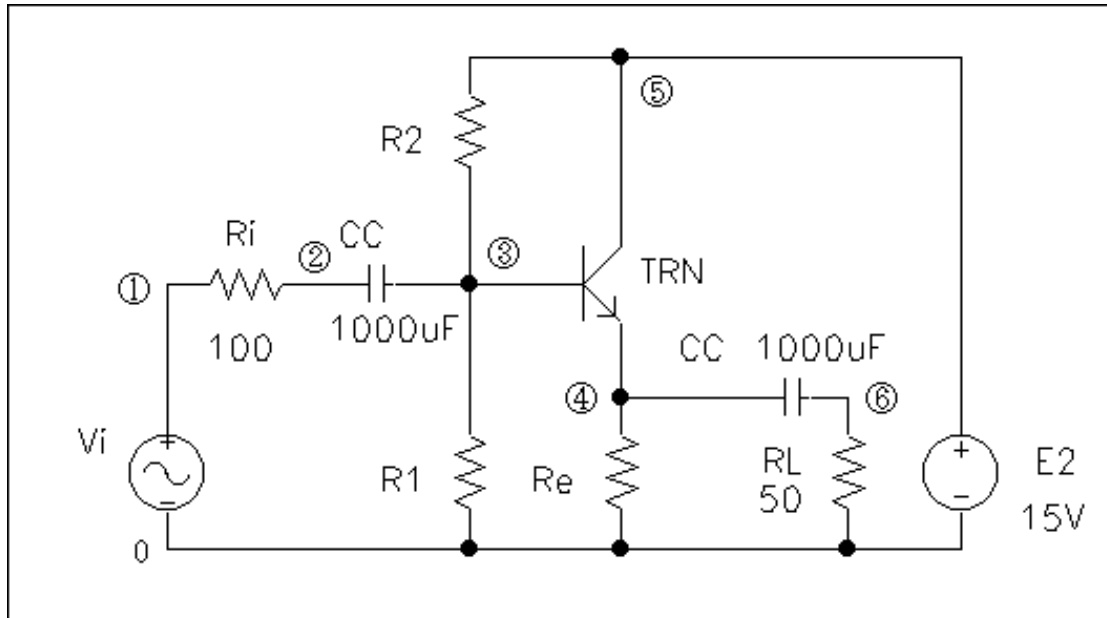
上で示した他に、多数のトランジスタ、FET、オペアンプ、BPF 回路、HPF 回路、LC 回路などの部品回路(*.cir)が準備されている。

図 n型エンハンスメントタイプFETの回路図記号と等価回路 FETE. CIR

サンプルバッチファイルの回路図と説明

(1) バッチファイル名 s 5. s b で扱う回路図

エミッタホロワ増幅回路



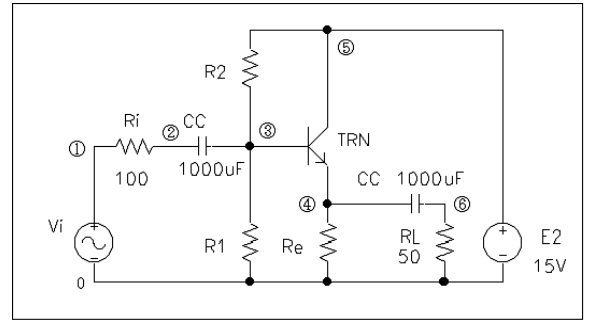
設計目標

1. 負荷抵抗 R_L の両端に発生する出力電圧の入力信号に対する比率 (電圧利得) は最低でも 0.9 を確保する。(エミッタホロワ回路の電圧ゲインは最大でも 1 である)
2. 入力信号の振幅は、 -4 V から $+4\text{ V}$ までとする。
3. 入力信号源抵抗は $R_i = 100$ オームとする。
4. 負荷抵抗は $R_L = 50$ オームとする。
5. npn 型トランジスタ TRN の h_{fe} (等価回路では、 k_{b1} となる) は 100 から 200 までのばらつきがあるものとする。
6. $V_{ce(sat)} = 1\text{ V}$ とする。
7. カップリングコンデンサ CC は入力信号周波数の範囲に対して無視出来るインピーダンスとする。

(1) バッチファイル名 s 5. s b で扱う回路図

サンプルバッチファイルの回路図と説明

s 5. s b の概略説明



1. ノード4の直流動作点を決定する。

$V_{ce}(sat) < V_{ce}$ 及び $0 < x4$ から $x4 = 7$ とする

```

x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]
r1[3000 ] f[0 ] abs [ 5.2906 ] arg [ 0 ]
x4 [ 5.2906 j 0 ] abs [ 5.2906 ] arg [ 0 ]
x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]
r1[4000 ] f[0 ] abs [ 5.9195 ] arg [ 0 ]
x4 [ 5.9195 j 0 ] abs [ 5.9195 ] arg [ 0 ]
x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]
r1[5000 ] f[0 ] abs [ 6.3658 ] arg [ 0 ]
x4 [ 6.3658 j 0 ] abs [ 6.3658 ] arg [ 0 ]
x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]
r1[6000 ] f[0 ] abs [ 6.6989 ] arg [ 0 ]
x4 [ 6.6989 j 0 ] abs [ 6.6989 ] arg [ 0 ]
x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]
r1[7000 ] f[0 ] abs [ 6.9571 ] arg [ 0 ]
x4 [ 6.9571 j 0 ] abs [ 6.9571 ] arg [ 0 ]
x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]
r1[8000 ] f[0 ] abs [ 7.1630 ] arg [ 0 ]
x4 [ 7.1630 j 0 ] abs [ 7.1630 ] arg [ 0 ]
x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]
r1[9000 ] f[0 ] abs [ 7.3311 ] arg [ 0 ]
x4 [ 7.3311 j 0 ] abs [ 7.3311 ] arg [ 0 ]
x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]
r1[1e+004 ] f[0 ] abs [ 7.4709 ] arg [ 0 ]
x4 [ 7.4709 j 0 ] abs [ 7.4709 ] arg [ 0 ]
x6 [ 0 j 0 ] abs [ 0 ] arg [ 0 ]

```

B?

r1 = 8K とします。この時、 $x4 = 7.1630V$
利得を確認します。

2. R 1, R 2 の決定

a. 入力インピーダンスによる条件

b. ノード4の直流動作点による条件

により、とりあえず $R2 = 3K$ とする。次に、/para を使用してノード4が $7V$ となる

$R1$ を求める。 $R1 = 8K$ が得られる。

3. ゲインを確認する

(1) バッチファイル名 s 5. s b で扱う回路図

サンプルバッチファイルの回路図と説明

```

利得を確認します。

キー入力待ち？

B? r1=8k
    8,000 j 0

B? f=1000
    1,000 j 0

B? /ac
ac 入力信号源名は ? e1
値は ? 1

値を変化させる素子名は ? kb1
ac kb1 最低値 ? 100
ac kb1 最高値 ? 200
ac kb1 ステップ ? 50
e1[1      ] j 0      ]
kb1[100    ] f[1,000  ]
x4 [      0.9087 j 4.139658e-005] abs[      0.9087] arg[      0.0026]
x6 [      0.9087 j 0.0029      ] abs[      0.9087] arg[      0.1850]
kb1[150    ] f[1,000  ]
x4 [      0.9227 j 4.731972e-005] abs[      0.9227] arg[      0.0029]
x6 [      0.9227 j 0.0030      ] abs[      0.9227] arg[      0.1853]
kb1[200    ] f[1,000  ]
x4 [      0.9299 j 5.042787e-005] abs[      0.9299] arg[      0.0031]
x6 [      0.9299 j 0.0030      ] abs[      0.9299] arg[      0.1855]
B? キー入力待ち？

```

h f e = 1 0 0 の時に、あまり余裕がないので、ノード4の直流動作点を8 Vにあげてみる。

/para を使用してR 1 を求めると、R 1 = 1 5 Kとなり、今度はh f e = 1 0 0 のときにも余裕がある。

```

B? /para
値を変化させる素子名は ? r1
para r1 最低値 ? 8k
para r1 最高値 ? 15k
para r1 ステップ ? 1k
r1[8000    ] f[0      ]
x4 [      7.1630 j 0      ] abs[      7.1630] arg[      0]
x6 [      0 j 0      ] abs[      0] arg[      0]
r1[9000    ] f[0      ]
x4 [      7.3311 j 0      ] abs[      7.3311] arg[      0]
x6 [      0 j 0      ] abs[      0] arg[      0]
r1[1e+004  ] f[0      ]
x4 [      7.4709 j 0      ] abs[      7.4709] arg[      0]
x6 [      0 j 0      ] abs[      0] arg[      0]
r1[1.1e+004] f[0      ]
x4 [      7.5890 j 0      ] abs[      7.5890] arg[      0]
x6 [      0 j 0      ] abs[      0] arg[      0]
r1[1.2e+004] f[0      ]
x4 [      7.6901 j 0      ] abs[      7.6901] arg[      0]
x6 [      0 j 0      ] abs[      0] arg[      0]
r1[1.3e+004] f[0      ]
x4 [      7.7777 j 0      ] abs[      7.7777] arg[      0]
x6 [      0 j 0      ] abs[      0] arg[      0]
r1[1.4e+004] f[0      ]
x4 [      7.8542 j 0      ] abs[      7.8542] arg[      0]
x6 [      0 j 0      ] abs[      0] arg[      0]
r1[1.5e+004] f[0      ]
x4 [      7.9216 j 0      ] abs[      7.9216] arg[      0]
x6 [      0 j 0      ] abs[      0] arg[      0]
B? キー入力待ち？

```

サンプルバッチファイルの回路図と説明

```

B?
r1 = 15K とします。この時の利得を確認します。

キー入力待ち？

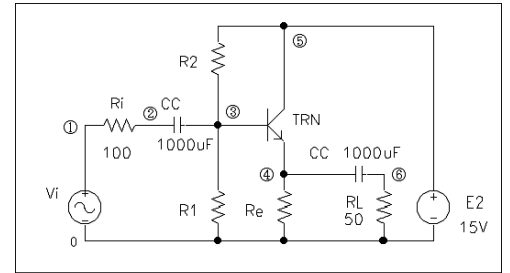
B? r1=15k
15.000 j 0

B? f=1000
1.000 j 0

B? /ac
ac 入力信号源名は ? e1
値は ? 1

値を変化させる素子名は ? kb1
ac kb1 最低値 ? 100
ac kb1 最高値 ? 200
ac kb1 ステップ ? 50
e1[1] j 0
kb1[100] f[1.000]
x4 [ 0.9136 j 3.409550e-005] abs[ 0.9136] arg[ 0.0021]
x6 [ 0.9136 j 0.0029 ] abs[ 0.9136] arg[ 0.1845]
kb1[150] f[1.000]
x4 [ 0.9278 j 3.984743e-005] abs[ 0.9278] arg[ 0.0025]
x6 [ 0.9278 j 0.0030 ] abs[ 0.9278] arg[ 0.1848]
kb1[200] f[1.000]
x4 [ 0.9350 j 4.286738e-005] abs[ 0.9350] arg[ 0.0026]
x6 [ 0.9350 j 0.0030 ] abs[ 0.9350] arg[ 0.1850]
B? キー入力待ち？

```



4. 出力インピーダンスの確認

負荷抵抗 $R_L = 1\text{ M}$ (無負荷の状態) として、ノード 6 の出力電圧が 1 V となる入力信号の電圧を求める。これを a とする。

/ac によって、入力信号の電圧が a の時に、ノード 6 の出力電圧が 0.5 V (半分) になる

R_L の値を求める。これが出力インピーダンスの値である。

$Z_o = 1.3\text{ オーム}$ が得られる。

エミッタホロワの出力インピーダンスはかなり小さいことが分かる。

```

B? /ac
ac 入力信号源名は ? e1
値は ? 1

値を変化させる素子名は ? e1
ac e1 最低値 ? 1
ac e1 最高値 ? 1
ac e1 ステップ ? 1
e1[1] j 0
e1[1] f[1.000]
x4 [ 0.9370 j 8.396168e-005] abs[ 0.9370] arg[ 0.0051]
x6 [ 0.9370 j 8.411081e-005] abs[ 0.9370] arg[ 0.0051]
B? キー入力待ち？

B?
この状態で x6 = 1 となる e1 を求めます

a=1/x6
1.0673 j -9.580561e-005

B? /ac
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? r1
ac r1 最低値 ? 1m
ac r1 最高値 ? 1m
ac r1 ステップ ? 1
e1[1.067] j -9.581e-005
r1[1e+006] f[1.000]
x4 [ 1 j -1.591549e-007] abs[ 1] arg[-9.118907e-006]
x6 [ 1 j 0 ] abs[ 1] arg[ 0]
B?
予定通りに x6 = 1 となりました。
キー入力待ち？

```

サンプルバッチファイルの回路図と説明

```

B?
次に、rl を変化させて x6 = 0.5 となる、rl を求めます
エミッタホロワの出力インピーダンスは re/10 以下と考えられます

キー入力待ち？

B? /ac
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? rl
ac rl 最低値 ? 1
ac rl 最高値 ? 5
ac rl ステップ ? 1
e1[1.067 j -9.581e-005]
rl[1 ] f[1,000 ]
x4 [ 0.4419 j -0.0387 ] abs[ 0.4436] arg[ -5.0101]
x6 [ 0.4370 j 0.0308 ] abs[ 0.4381] arg[ 4.0330]
rl[2 ] f[1,000 ]
x4 [ 0.6112 j -0.0186 ] abs[ 0.6115] arg[ -1.7452]
x6 [ 0.6089 j 0.0298 ] abs[ 0.6096] arg[ 2.8047]
rl[3 ] f[1,000 ]
x4 [ 0.7018 j -0.0109 ] abs[ 0.7019] arg[ -0.8872]
x6 [ 0.7004 j 0.0263 ] abs[ 0.7009] arg[ 2.1496]
rl[4 ] f[1,000 ]
x4 [ 0.7582 j -0.0071 ] abs[ 0.7583] arg[ -0.5360]
x6 [ 0.7573 j 0.0230 ] abs[ 0.7577] arg[ 1.7425]
rl[5 ] f[1,000 ]
x4 [ 0.7967 j -0.0050 ] abs[ 0.7967] arg[ -0.3581]
x6 [ 0.7960 j 0.0204 ] abs[ 0.7963] arg[ 1.4651]
B? キー入力待ち？ _

```

Zo=1.3 オームと推定される。

```

B? /ac
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? rl
ac rl 最低値 ? 1
ac rl 最高値 ? 2
ac rl ステップ ? 0.1
e1[1.067 j -9.581e-005]
rl[1 ] f[1,000 ]
x4 [ 0.4419 j -0.0387 ] abs[ 0.4436] arg[ -5.0101]
x6 [ 0.4370 j 0.0308 ] abs[ 0.4381] arg[ 4.0330]
rl[1.1 ] f[1,000 ]
x4 [ 0.4652 j -0.0355 ] abs[ 0.4665] arg[ -4.3690]
x6 [ 0.4607 j 0.0311 ] abs[ 0.4617] arg[ 3.8638]
rl[1.2 ] f[1,000 ]
x4 [ 0.4866 j -0.0327 ] abs[ 0.4877] arg[ -3.8467]
x6 [ 0.4824 j 0.0313 ] abs[ 0.4834] arg[ 3.7083]
rl[1.3 ] f[1,000 ]
x4 [ 0.5064 j -0.0302 ] abs[ 0.5073] arg[ -3.4150]
x6 [ 0.5025 j 0.0313 ] abs[ 0.5035] arg[ 3.5648]
rl[1.4 ] f[1,000 ]
x4 [ 0.5247 j -0.0280 ] abs[ 0.5254] arg[ -3.0537]
x6 [ 0.5211 j 0.0313 ] abs[ 0.5220] arg[ 3.4319]
rl[1.5 ] f[1,000 ]
x4 [ 0.5417 j -0.0260 ] abs[ 0.5423] arg[ -2.7480]
x6 [ 0.5384 j 0.0311 ] abs[ 0.5393] arg[ 3.3086]
rl[1.6 ] f[1,000 ]
x4 [ 0.5575 j -0.0242 ] abs[ 0.5580] arg[ -2.4868]
x6 [ 0.5544 j 0.0309 ] abs[ 0.5553] arg[ 3.1939]
rl[1.7 ] f[1,000 ]
x4 [ 0.5723 j -0.0226 ] abs[ 0.5727] arg[ -2.2617]
x6 [ 0.5694 j 0.0307 ] abs[ 0.5702] arg[ 3.0868]
rl[1.8 ] f[1,000 ]
x4 [ 0.5861 j -0.0211 ] abs[ 0.5865] arg[ -2.0663]
x6 [ 0.5834 j 0.0304 ] abs[ 0.5842] arg[ 2.9866]
rl[1.9 ] f[1,000 ]
x4 [ 0.5991 j -0.0198 ] abs[ 0.5994] arg[ -1.8954]
x6 [ 0.5965 j 0.0301 ] abs[ 0.5973] arg[ 2.8928]
B? キー入力待ち？ _

```

(1) バッチファイル名 s 5. s b で扱う回路図

サンプルバッチファイルの回路図と説明

5. 入力インピーダンスの確認

$R_i = 0.001$ オームとして、ノード 6 の出力電圧が 1 V となる入力信号の電圧 a を求める。 $/ac$ によって、入力信号の電圧が a の時に、ノード 6 の出力電圧が 0.5 V (半分) になる R_i の値を求める。これが入力インピーダンスの値である。

$Z_i = 1250$ オームが得られる。

```
次に、入力インピーダンスを求めます
ri = 0.001 オームの時の利得が半分になる ri の値が入力インピーダンスです
ri = 50 に戻します
```

```
ri=50
      50 j 0
```

```
B? ri = 0.001 として、x6 = 1V となる、e1 を求めます
```

```
キー入力待ち？
```

```
B? ri=0.001
      0.0010 j 0
```

```
B? /ac
ac 入力信号源名は ? e1
値は ? 1
```

```
値を変化させる素子名は ? e1
```

```
ac e1 最低値 ? 1
```

```
ac e1 最高値 ? 1
```

```
ac e1 ステップ ? 1
```

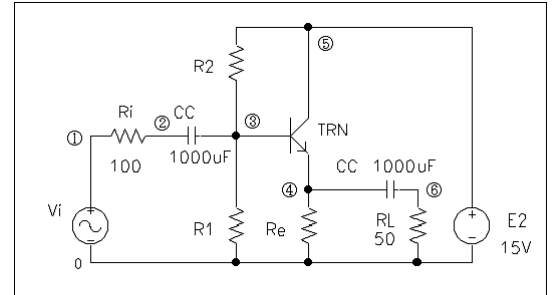
```
e1[1] j 0
```

```
e1[1] f[1,000]
```

```
x4 [ 0.9859 j 1.018468e-004] abs[ 0.9859] arg[ 0.0059]
```

```
x6 [ 0.9859 j 0.0032 ] abs[ 0.9859] arg[ 0.1883]
```

```
B? a=1/x6
      1.0143 j -0.0033
```



```
/ac
```

```
ac 入力信号源名は ? e1
```

```
値は ? a
```

```
値を変化させる素子名は ? ri
```

```
ac ri 最低値 ? 1000
```

```
ac ri 最高値 ? 1500
```

```
ac ri ステップ ? 50
```

```
e1[1.014 j -0.003334]
```

```
ri[1000] f[1,000]
```

```
x4 [ 0.5585 j -0.0020 ] abs[ 0.5585] arg[ -0.2052]
```

```
x6 [ 0.5585 j -2.221239e-004] abs[ 0.5585] arg[ -0.0228]
```

```
ri[1050] f[1,000]
```

```
x4 [ 0.5465 j -0.0020 ] abs[ 0.5465] arg[ -0.2058]
```

```
x6 [ 0.5465 j -2.232647e-004] abs[ 0.5465] arg[ -0.0234]
```

```
ri[1100] f[1,000]
```

```
x4 [ 0.5349 j -0.0019 ] abs[ 0.5349] arg[ -0.2064]
```

```
x6 [ 0.5349 j -2.241114e-004] abs[ 0.5349] arg[ -0.0240]
```

```
ri[1150] f[1,000]
```

```
x4 [ 0.5238 j -0.0019 ] abs[ 0.5238] arg[ -0.2070]
```

```
x6 [ 0.5238 j -2.246974e-004] abs[ 0.5238] arg[ -0.0246]
```

```
ri[1200] f[1,000]
```

```
x4 [ 0.5132 j -0.0019 ] abs[ 0.5132] arg[ -0.2075]
```

```
x6 [ 0.5132 j -2.250520e-004] abs[ 0.5132] arg[ -0.0251]
```

```
ri[1250] f[1,000]
```

```
x4 [ 0.5030 j -0.0018 ] abs[ 0.5030] arg[ -0.2080]
```

```
x6 [ 0.5030 j -2.252010e-004] abs[ 0.5030] arg[ -0.0257]
```

```
ri[1300] f[1,000]
```

```
x4 [ 0.4932 j -0.0018 ] abs[ 0.4932] arg[ -0.2085]
```

```
x6 [ 0.4932 j -2.251675e-004] abs[ 0.4932] arg[ -0.0262]
```

```
ri[1350] f[1,000]
```

```
x4 [ 0.4838 j -0.0018 ] abs[ 0.4838] arg[ -0.2090]
```

```
x6 [ 0.4838 j -2.249719e-004] abs[ 0.4838] arg[ -0.0266]
```

```
ri[1400] f[1,000]
```

```
x4 [ 0.4747 j -0.0017 ] abs[ 0.4747] arg[ -0.2095]
```

```
x6 [ 0.4747 j -2.246323e-004] abs[ 0.4747] arg[ -0.0271]
```

```
ri[1450] f[1,000]
```

```
x4 [ 0.4660 j -0.0017 ] abs[ 0.4660] arg[ -0.2099]
```

```
x6 [ 0.4660 j -2.241649e-004] abs[ 0.4660] arg[ -0.0276]
```

```
ri[1500] f[1,000]
```

```
x4 [ 0.4575 j -0.0017 ] abs[ 0.4575] arg[ -0.2104]
```

```
x6 [ 0.4575 j -2.235841e-004] abs[ 0.4575] arg[ -0.0280]
```

```
B? キー入力待ち？
```

(1) バッチファイル名 s 5. s b で扱う回路図

サンプルバッチファイルの回路図と説明

```
B?
ri = 1250 オームで x6 は大体 0.5V となります
従って、入力インピーダンス Zi = 1250 オーム と求められました。
```

次に、抵抗器とトランジスタの消費電力を計算します

6. 各抵抗器とトランジスタの消費電力を計算し使用する抵抗器の定格を決定する。
 R 1は1 / 1 6 W、R 2は1 / 8 W、R eは5 Wとなる。トランジスタのコレクタ損失は1 .
 2 Wとなり、周囲温度7 0 度まで使用するとしたら、熱抵抗が2 5 度 / W以下の放熱器
 が必要である。R Lには交流信号のみが現れるので、実効値で消費電力を計算して、R
 Lには1 Wの抵抗器を使用する。

```
B?
まず、ri=100 に戻して、直流動作点を再計算します
ri=100
100 j 0

B? f=0
0 j 0
```

```
B? 表示ノードを3、4及び6とします
/disg
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 5] j[ 0] 部品名[e2] 値[15 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.6 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
4 6

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 3
ノード番号 (0 なら終了) ? 4
ノード番号 (0 なら終了) ? 6
ノード番号 (0 なら終了) ?
B? /point
f[0
x3 [ 8.5784 j 0 ] abs[ 8.5784] arg[ 0]
x4 [ 7.9216 j 0 ] abs[ 7.9216] arg[ 0]
x6 [ 0 j 0 ] abs[ 0] arg[ 0]
B? キー入力待ち？
```

サンプルバッチファイルの回路図と説明

```

B?
r2 の両端の電圧は、  $v2 = e2 - x3$ 
r2 の消費電力は、  $p2 = v2 * v2 / r2$ 

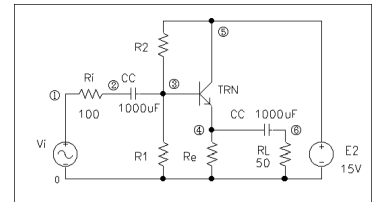
 $v2 = (e2 - x3)$ 
6.4216 j 0

B?  $p2 = v2 * v2 / r2$ 
0.0137 j 0

B?  $1/p2$ 
72.7504 j 0

B?
    従って、約 10 倍の余裕をもって r2 は 1/8W の抵抗器が使用できます

```



```

B?
    同様に、r1 については

 $v1 = x3$ 
8.5784 j 0

B?  $p1 = v1 * v1 / r1$ 
0.0049 j 0

B?  $1/p1$ 
203.8347 j 0

B?
    同様に、10 倍の余裕をもって r1 は 1/16W の抵抗器を使用できます

```

```

B?
    re については
 $ve = x4$ 
7.9216 j 0

B?  $pe = ve * ve / re$ 
1.2550 j 0

B? re については、4 倍の余裕をもって、5W の抵抗器を使用して下さい
    キー入力待ち？

B?
    trn の消費電力は、主にコレクタ損失ですから
 $Vce = (e2 - x4)$ ,  $Ie = x4 / re$ ,  $Pc = Vce * Ie$ 

 $vce = e2 - x4$ 
7.0784 j 0

B?  $ie = x4 / 50$ 
0.1584 j 0

B?  $pc = vce * ie$ 
1.1214 j 0

B? コレクタ損失は、約 1.2W となります

    接合温度の限界を 120 度として、周囲温度 70 度まで使用するには
    コレクタ損失を多めに 2W として計算すると、
     $(120 - 70) / 2 = 25$  度/W 以下の放熱器を使用する必要があります

 $(120 - 70) / 2$ 
25 j 0

```

サンプルバッチファイルの回路図と説明

```

B?
r1 については、-4V から +4V までの振幅の正弦波がかかるとすると、
e1 の実効値は  $e1 = 4/\sqrt{2}$ 
この値の e1 における出力電圧を求める

f=1000
    1,000 j 0

B? /ac
ac 入力信号源名は ? e1
値は ?  $4/\sqrt{2}$ 

値を変化させる素子名は ? r1
ac r1 最低値 ? 50
ac r1 最高値 ? 50
ac r1 ステップ ? 50
e1[2.828      j 0      ]
r1[50         ] f[1,000 ]
x3 [    2.6212 j 1.567832e-004] abs[    2.6212] arg[    0.0034]
x4 [    2.5842 j 9.643665e-005] abs[    2.5842] arg[    0.0021]
x6 [    2.5842 j 0.0083      ] abs[    2.5842] arg[    0.1845]
B? キー入力待ち?

```

```

B?
r1 にかかる電圧は x6 だから、消費電力 PI は  $PI = x6*x6/r1$ 

pl= $x6*x6/r1$ 
    0.1336 j 8.602222e-004

B? 1/pl
    7.4872 j -0.0482

B? 従って、r1 については、約 7 倍の余裕をもって、1W の抵抗器が使用できます

    以上で、エミッタホロワ増幅回路の設計を終わります

1 番のバッチファイル s5.sb を終了します。
? _

```

以上の様に、/para で直流動作点を満足する抵抗値を決定し、/ac によって交流ゲインを満足する抵抗値を決定することが出来る。また、負荷抵抗を開放状態（ $R_L = 1\text{ M}$ 等の非常に大きな値）にしたときの交流ゲインが半分になる負荷抵抗の値が出力インピーダンスであり、入力抵抗 R_i がほとんどゼロのときの交流ゲインが半分になる入力抵抗の値が入力インピーダンスである。この方法は色々な回路の解析に使用出来るので覚えておくと便利である。

サンプルバッチファイルの回路図と説明

このエミッタホロワ増幅回路の出力抵抗は 1.3 オームと求められたので、イヤホンアンプとして利用してみたいと思う。

最近のイヤホンは非常に高感度なので、わずか 1mW の出力があれば 100dB 程度の音量が得られる。85dB を越える音量は耳に悪いほどの激しい音量である。
多少の余裕を考えても、10mW 程度の出力が得られたら十分だと考える。

イヤホンのインピーダンスは商品によって数 10 オームから数 100 オームまで幅があるが、32 オームのイヤホンを対象として検討する。

交流電圧の実効値を V 、負荷抵抗（イヤホンのインピーダンス）を R とすると、電力 P との関係は次式で表される。

$$V = \sqrt{P \cdot R}$$

従って、 $P=10\text{mW}$ 、 $R=32$ を代入すると、 $V=0.5657\text{V}$ が得られる。

電圧の実効値を振幅に変換すると、 $2 \cdot \sqrt{2} \cdot 0.5657 = 1.6 V_{p-p}$ が得られる。

この出力振幅ならば、電源電圧 $e2$ を 3V まで下げても構わないと考えられる。

それから、信号入力が無い時のエミッタ電流を減らすために re を 2K に増加する。

以上の方針で変更した回路の動作を確認することにする。

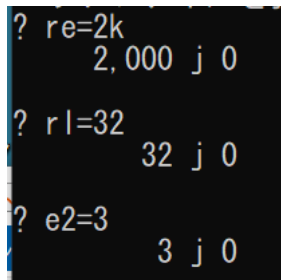
まず、@s5 でバッチファイルを読み込み、「回路リストの入力 キー入力待ち」は「リターンキー」を押し、次の「キー入力待ち」で/exit する。

re=2k

rl=32

e2=3

と素子の設定を変更する。



```
? re=2k
    2,000 j 0
? rl=32
    32 j 0
? e2=3
    3 j 0
```

サンプルバッチファイルの回路図と説明

次に、ノード 3 の電圧を e1 によって直接与えられるようにノード 1 とノード 3 の間に抵抗 $r0=0.1$ を追加する。

```
? /padd
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 3] parts[ r1 ] value[ 1e+004 j 0 ]
left[ 0] right[ 4] parts[ re ] value[ 2000 j 0 ]
left[ 0] right[ 5] parts[ e2 ] value[ 3 j 0 ]
left[ 0] right[ 6] parts[ r1 ] value[ 32 j 0 ]
left[ 1] right[ 2] parts[ ri ] value[ 100 j 0 ]
left[ 2] right[ 3] parts[ cc ] value[ 0.001 j 0 ]
left[ 3] right[ 5] parts[ r2 ] value[ 3000 j 0 ]
left[ 3] right[ 10] parts[ r1b1 ] value[ 0.1 j 0 ]
left[ 4] right[ 6] parts[ cc ] value[ 0.001 j 0 ]
left[ 4] right[ 7] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 5] right[ 9] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 7] right[ 8] parts[ reb1 ] value[ 26 j 0 ]
left[ 7] right[ 9] parts[ kb1 ] value[ 100 j 0 ]
@ master[reb1 ] left[ 7] right[ 8]
left[ 8] right[ 10] parts[ ebb1 ] value[ 0.6 j 0 ]
最大の素子番号 = 15
最大のノード番号 = 10

left[ 0] right ? /

left[ 1] right ? 3
素子名は ? r0
値は ? 0.1

left[ 1] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 5] j[ 0] 部品名[e2] 値[3 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.6 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
4 6

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 4
ノード番号 (0 なら終了) ?
```

表示ノードはノード 4 だけに変更する。

サンプルバッチファイルの回路図と説明

e1 を 0V から 3V まで、0.1V ステップで変えて、ノード電圧を確認する。

```
? /auto
? /gpos
MANU ON または 最高/最低の比が 10 以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 32 です
300
gpos = 300 に設定しました
? /mk
データファイルを作成する..... /mk または /nomk で切り替える
値で計算するモード..... /db または /val で切り替える
計算ステップのモードは /auto (自動切り替え) です
..... /auto または /manu で切り替える
等比級数のポイント数は 300 です..... /gpos で設定する
シミュレーションにおける 表示ノード設定状況
4
ブロック名を回路リストから得るモード..... /bmode で設定する
シミュレーション・コマンドの計算を ノード電圧のみで計算するモード
..... /cal で設定する
```

```
? /para
データファイル名は ? s5-3v
値を変化させる素子名は ? e1
para e1 最低値 ? 0
para e1 最高値 ? 3
para e1 ステップ ? 0.1
```

s5-3v.d を確認する

1 31

0 e1=0V

4 -0.599792 x4=-0.599792V 動作していない

0 0

0.1

4 -0.499815

0 0

0.2

4 -0.399837

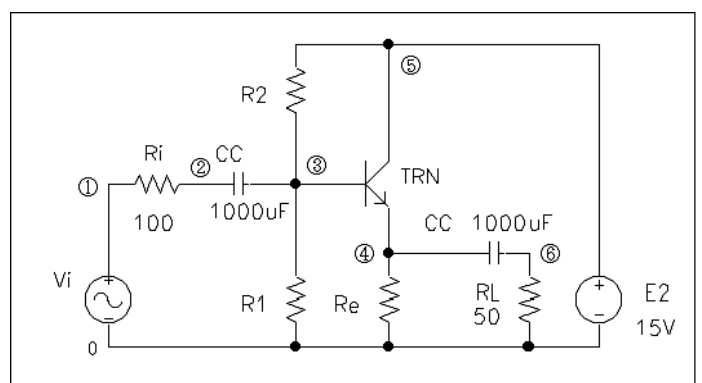
0 0

.

.

.

.



(1) バッチファイル名 s 5. s b で扱う回路図

サンプルバッチファイルの回路図と説明

```

0.6
4 7.39835e-005
0 0
0.7          e1=0.7V
4 0.100052   x4=0.100052V 動作しているが、0V に近すぎる
0 0
0.8          e1=0.8V
4 0.200029   x4=0.200029 以上で使用する ことにする
0 0
0.9
4 0.300007
0 0.
.
.
.
2.4
4 1.79967
0 0
2.5
4 1.89965
0 0
2.6
4 1.99963
0 0
2.7
4 2.09961
0 0
2.8          e1=2.8V 入力信号の振幅は 2Vp-p 以内と考える
4 2.19958
0 0

```

データよりノード 4 の電圧は、0.2V から 2.2V の範囲で動作することが分かる。

従って、電源電圧を 3V に変更した回路では、インピーダンス 32 オームのイヤホンに対して 2V_{p-p} の出力が可能である。これを実効値に変換すると 0.707 V になるので、出力は $\frac{0.707 \cdot 0.707}{32} \cdot 1000 = 15.6 \text{ mW}$ となる。

サンプルバッチファイルの回路図と説明

この場合、ノード 4 の動作中心電圧は $\frac{0.2+2.2}{2} = 1.2\text{ V}$ となる。

今度は、バッチファイル@s5-3 を実行します。

すでに、回路の素子値がイヤホン用に変更されています。

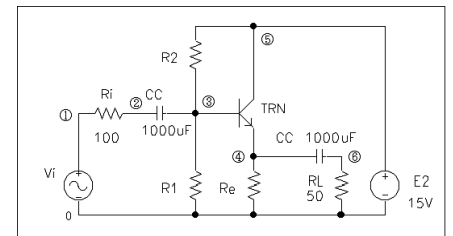
```
/dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0      ]
left[ 0] right[ 3] parts[ r1          ] value[ 1e+004 j 0      ]
left[ 0] right[ 4] parts[ re          ] value[      2000 j 0      ]
left[ 0] right[ 5] parts[ e2          ] value[      3 j 0      ]
left[ 0] right[ 6] parts[ rl          ] value[      32 j 0      ]
left[ 1] right[ 2] parts[ ri          ] value[      0.1 j 0      ]
left[ 2] right[ 3] parts[ cc          ] value[    0.001 j 0      ]
left[ 3] right[ 5] parts[ r2          ] value[     3000 j 0      ]
left[ 3] right[10] parts[ rlb1         ] value[      0.1 j 0      ]
left[ 4] right[ 6] parts[ cc          ] value[    0.001 j 0      ]
left[ 4] right[ 7] parts[ rbb1         ] value[      0.1 j 0      ]
left[ 5] right[ 9] parts[ rbb1         ] value[      0.1 j 0      ]
left[ 7] right[ 8] parts[ reb1         ] value[      26 j 0      ]
left[ 7] right[ 9] parts[ kb1          ] value[     100 j 0      ]
@ master[reb1          ] left[ 7] right[ 8]
left[ 8] right[10] parts[ ebb1         ] value[      0.6 j 0      ]
最大の素子番号 = 15
最大のノード番号 = 10
B?
これからイヤホンアンプを確認します
```

r2 を調整して、ノード 4 の動作点を 1.2 V に設定します。

```
? /point
f[0] [      ]
x4 [ 1.6881 j 0 ] abs[ 1.6881 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
```

現在の動作点が高すぎるので、r2 を増加させてみます。

```
? /para
値を変化させる素子名は ? r2
para r2 最低値 ? 3k
para r2 最高値 ? 10k
para r2 ステップ ? 1k
r2[3000] f[0]
x4 [ 1.6881 j 0 ] abs[ 1.6881 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
r2[4000] f[0]
x4 [ 1.5211 j 0 ] abs[ 1.5211 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
r2[5000] f[0]
x4 [ 1.3770 j 0 ] abs[ 1.3770 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
r2[6000] f[0]
x4 [ 1.2515 j 0 ] abs[ 1.2515 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
r2[7000] f[0]
x4 [ 1.1412 j 0 ] abs[ 1.1412 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
r2[8000] f[0]
x4 [ 1.0435 j 0 ] abs[ 1.0435 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
r2[9000] f[0]
x4 [ 0.9563 j 0 ] abs[ 0.9563 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
r2[1e+004] f[0]
x4 [ 0.8781 j 0 ] abs[ 0.8781 ] arg[ 0 ]
x6 [ 0 j 0 ] abs[ 0 ] arg[ 0 ]
```



(1) バッチファイル名 s 5. s b で扱う回路図

サンプルバッチファイルの回路図と説明

r2=6K から 7K の間で 1.2V になります。

```
r2[6400] f[0]
x4 [ 1.2058 j 0 ] abs[ 1.2058] arg[ 0]
x6 [ 0 j 0 ] abs[ 0] arg[ 0]
```

R2 = 6.4k に設定します。

ライン入力は 2Vp-p なので、正側のピーク電圧 1V を e1 として出力を確認します。

x6 の電圧と同時に電力も確認したいので、コマンド/cal を利用します。

```
? /cal
シミュレーションコマンドの計算は ノード電圧のみで計算するモードです

従来通りノード電圧のみで行なうなら 0
/disg で設定した式を計算するなら 1 1

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 5] j[ 0] 部品名[e2] 値[3 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.6 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0

シミュレーションにおける 計算式設定状況
p1 x4
p2 x4*x4/2/r1

計算式 (0 なら新規設定、-1 ならこのまま、
1 から 3 の数字なら p? の計算式の変更・追加・削除) ? 0

ノード電圧は x1, x2 の様に入力して下さい。
因子として定義されているすべての関数を使用出来ます。
1 番目の計算式を入力して下さい
x6

ノード電圧は x1, x2 の様に入力して下さい。
因子として定義されているすべての関数を使用出来ます。
2 番目の計算式を入力して下さい
x6*x6/2/r1

ノード電圧は x1, x2 の様に入力して下さい。
因子として定義されているすべての関数を使用出来ます。
3 番目の計算式を入力して下さい
```

サンプルバッチファイルの回路図と説明

```
? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x6
p1 [ 0.9887 j 0.0049 ] abs[ 0.9887] arg[ 0.2868]
x6*x6/2/rl
p2 [ 0.0153 j 1.528897e-004] abs[ 0.0153] arg[ 0.5736]
```

1Vの入力に対して、0.9887Vの出力が得られるので、2V p-p に対して 1.9774 Vp-p の出力が得られることになります。

この出力は 32 オームのイヤホンに対して、 $\frac{0.9887 \cdot 0.9887}{2 \cdot 32} \cdot 1000 = 15 \text{ mW}$ となります。

試しに、300 オームのイヤホンに換えて、出力を確認してみます。

```
? rl=300
300 j 0

? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x6
p1 [ 0.9986 j 5.725333e-004] abs[ 0.9986] arg[ 0.0328]
x6*x6/2/rl
p2 [ 0.0017 j 1.905773e-006] abs[ 0.0017] arg[ 0.0657]
```

出力電圧は 0.9986V になり、イヤホンが 32 オームの時とあまり変わりません。

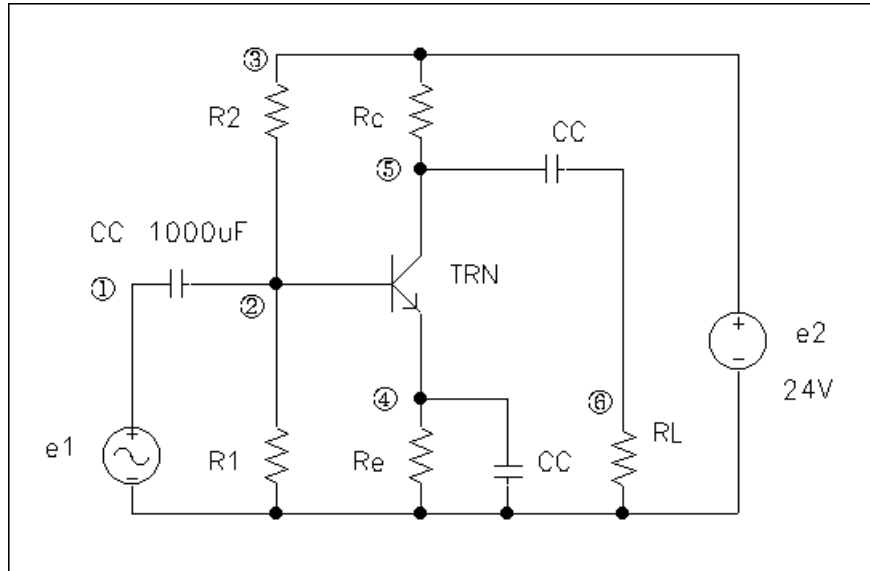
この出力は 300 オームのイヤホンに対して、 $\frac{0.9986 \cdot 0.9986}{2 \cdot 300} \cdot 1000 = 1.7 \text{ mW}$ となります。

出力は 32 オームのイヤホンに比べると 10 分の 1 位に小さくなります。

300 オームのイヤホンでも、何とか利用できると思います。

(2) バッチファイル名 s 6. s b で扱う回路図

エミッタ接地増幅回路



設計の指針

1. ノード5の直流動作点を電源電圧 e 2 の半分に設定すると、出力のダイナミックレンジが最大になるので、12Vに設定する。

/point によって、現在のノード5の電圧を確認すると、18.3965Vになっており12Vまで下げる必要がある。ノード5の電圧を下げるには、TRNのコレクタ電流を増加すればよい。これはノード2の電圧を上げることを意味している。

このためにはR2を小さくするか、R1を大きくすればよい。ここでは、入力インピーダンスが下がりすぎないようにR1を大きくする方法を選ぶ。/para によってR1を変化させて確認すると、ノード5が12VになるのはR1=26Kの時である事が分かる。

2. 入力信号の周波数が1000Hzの時のゲインを調べてみる。現在のゲインは-1820であることがわかる。しかし、この値は現実的な値ではない。trn.cirのr1は0.1オームとなっているが、トランジスタの規格表では通常のトランジスタでは800から1200オームの値が載っているので、r1=800に変更する。このためには、ブロック素子が解凍されたときの素子名であるr1b1を800に設定する。

3. もう一度、直流動作点を確認する。——> あまり変化はない。

4. もう一度、ゲインを確認する。——> -100倍となり、妥当な値である。

エミッタ接地増幅回路のゲインはエミッタホロウに比べると非常に大きく、入力と出力の位相が逆転することがわかる。

5. サンプル s 5. s b の方法を使って、出力インピーダンスを調べる。

(2) バッチファイル名 s 6. s b で扱う回路図

サンプルバッチファイルの回路図と説明

$Z_o = 3.8 \text{ K}$ オームが得られる。

サンプル s 5. s b のエミッタホロワ増幅回路の出力インピーダンスは 1.3 オーム であったことと比較すると、エミッタ接地増幅回路の出力インピーダンスはかなり高いことがわかる。このことは、負荷抵抗が変化すると出力電圧もかなり変化することを意味している。負荷の状態が変動する場合にも、安定した大きなゲインを希望するなら、エミッタ接地増幅回路の後ろにエミッタホロワ増幅回路を接続することが良い方法である。

また、エミッタ接地増幅回路では位相が反転するので、これを 2 段接続すると、もとに戻ることも覚えておくと良い。

```
B? /ipart
```

```
left[ 0] right ? 1
素子名は ? e1
値は ? 1
```

```
left[ 0] right ? 2
素子名は ? r1
値は ? 10k
```

```
left[ 0] right ? 4
素子名は ? re
値は ? 2.2k
```

```
left[ 0] right ? 4
素子名は ? ce
値は ? 1000u
```

```
left[ 0] right ? 6
素子名は ? r1
値は ? 1k
```

```
left[ 0] right ? 3
素子名は ? e2
値は ? 24
```

```
left[ 0] right ? /
```

```
left[ 1] right ? 2
素子名は ? cc
値は ? 1000u
```

```
left[ 1] right ? /
```

```
left[ 2] right ? 3
素子名は ? r2
値は ? 50k
```

```
left[ 2] right ? 5
素子名は ? b1
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn:2,4,5
```

```
left[ 2] right ? /
```

```
left[ 3] right ? 5
素子名は ? rc
値は ? 3.8k
```

```
left[ 3] right ? /
```

```
left[ 4] right ? /
```

```
left[ 5] right ? 6
素子名は ? cc
素子名 cc は既に使用されています。
同じ素子名を使用しますか? (y/n) y
```

```
left[ 5] right ? *
```

```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
独立電圧源素子の個数 2
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
```

```
表示ノード番号 (0 なら全て、-1 ならこのまま) ? キー入力待ち?
```

サンプルバッチファイルの回路図と説明

```

/dpart
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 2] parts[ r1 ] value[ 1e+004 j 0 ]
left[ 0] right[ 3] parts[ e2 ] value[ 24 j 0 ]
left[ 0] right[ 4] parts[ ce ] value[ 0.001 j 0 ]
left[ 0] right[ 4] parts[ re ] value[ 2200 j 0 ]
left[ 0] right[ 6] parts[ rl ] value[ 1000 j 0 ]
left[ 1] right[ 2] parts[ cc ] value[ 0.001 j 0 ]
left[ 2] right[ 3] parts[ r2 ] value[ 5e+004 j 0 ]
left[ 2] right[ 10] parts[ rl1 ] value[ 0.1 j 0 ]
left[ 3] right[ 5] parts[ rc ] value[ 3800 j 0 ]
left[ 4] right[ 7] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 5] right[ 6] parts[ cc ] value[ 0.001 j 0 ]
left[ 5] right[ 9] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 7] right[ 8] parts[ reb1 ] value[ 26 j 0 ]
left[ 7] right[ 9] parts[ kb1 ] value[ 100 j 0 ]
@ master[reb1 ] left[ 7] right[ 8]
left[ 8] right[ 10] parts[ ebb1 ] value[ 0.6 j 0 ]
最大の素子番号 = 16
最大のノード番号 = 10

```

```

B? 動作点確認
/point
f[0 ]
x5 [ 18.3965 j 0 ] abs[ 18.3965] arg[ 0]
B? ノード5の動作点を x5 = 24/2 = 12V に設定する
現在の動作点は x5 = 18.3965 > 12 なので、
R1 を増加させてバイアス電流を増加させれば、x5 を下げることができる
/para を使って調べて見る。

```

```

B? /para
値を変化させる素子名は ? r1
para r1 最低値 ? 10k
para r1 最高値 ? 50k
para r1 ステップ ? 5k
r1[1e+004 ] f[0 ]
x5 [ 18.3965 j 0 ] abs[ 18.3965] arg[ 0]
r1[1.5e+004 ] f[0 ]
x5 [ 15.9725 j 0 ] abs[ 15.9725] arg[ 0]
r1[2e+004 ] f[0 ]
x5 [ 13.9472 j 0 ] abs[ 13.9472] arg[ 0]
r1[2.5e+004 ] f[0 ]
x5 [ 12.2295 j 0 ] abs[ 12.2295] arg[ 0]
r1[3e+004 ] f[0 ]
x5 [ 10.7544 j 0 ] abs[ 10.7544] arg[ 0]
r1[3.5e+004 ] f[0 ]
x5 [ 9.4739 j 0 ] abs[ 9.4739] arg[ 0]
r1[4e+004 ] f[0 ]
x5 [ 8.3518 j 0 ] abs[ 8.3518] arg[ 0]
r1[4.5e+004 ] f[0 ]
x5 [ 7.3605 j 0 ] abs[ 7.3605] arg[ 0]
r1[5e+004 ] f[0 ]
x5 [ 6.4783 j 0 ] abs[ 6.4783] arg[ 0]

```

(2) バッチファイル名 s 6. s b で扱う回路図

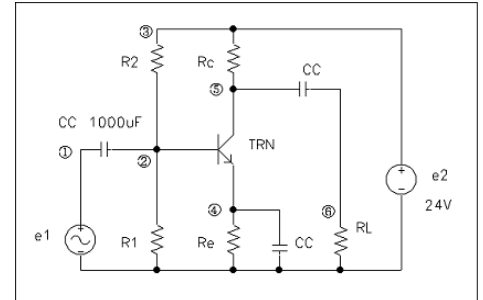
サンプルバッチファイルの回路図と説明

B? R1 が 25K から 30K の間で $x5 = 12$ となるので、詳しく調べる

```

/para
値を変化させる素子名は ? r1
para r1 最低値 ? 25k
para r1 最高値 ? 30k
para r1 ステップ ? 1k
r1[2.5e+004 ] f[0 ]
x5 [ 12.2295 j 0 ] abs[ 12.2295] arg[ 0]
r1[2.6e+004 ] f[0 ]
x5 [ 11.9168 j 0 ] abs[ 11.9168] arg[ 0]
r1[2.7e+004 ] f[0 ]
x5 [ 11.6134 j 0 ] abs[ 11.6134] arg[ 0]
r1[2.8e+004 ] f[0 ]
x5 [ 11.3187 j 0 ] abs[ 11.3187] arg[ 0]
r1[2.9e+004 ] f[0 ]
x5 [ 11.0325 j 0 ] abs[ 11.0325] arg[ 0]
r1[3e+004 ] f[0 ]
x5 [ 10.7544 j 0 ] abs[ 10.7544] arg[ 0]

```



R1=26K に決定します。

```

B? /point
f[0 ]
x5 [ 11.9168 j 0 ] abs[ 11.9168] arg[ 0]
B? f = 1000Hz における、ゲインを調べて見る
表示ノードを 5 から 6 に変更する

キー入力待ち?

B? f=1000
1,000 j 0

B? /disp

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.6 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
5

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 6
ノード番号 (0 なら終了) ?

```

ゲインを確認します

```

B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1 j 0]
f[1,000 ]
x6 [-1,820.6864 j -816.5491] abs[1,995.4076] arg[ -155.8445]
B? 現在のゲインは -1820 となっていますが、これは現実と比べて大き過ぎます。
回路ブロック trn.cir の r1 = 0.1 となっていますが、
トランジスタの規格表には 800 から 1200 の値が hie として記載されています。
ここでは、r1=800 とします
trn.cir は b1 という名称のブロックとして使用されていますから
r1=800 とするには、 r1b1=800 と入力する必要があります
r1b1=800
800 j 0

```

(2) バッチファイル名 s6. sb で扱う回路図

サンプルバッチファイルの回路図と説明

動作点を再確認する。

```
B? 動作点の確認
/point
f[0
x5 [ 11.9571 j 0 ] abs[ 11.9571] arg[ 0]
x6 [ 0 j 0 ] abs[ 0] arg[ 0]
B? 動作点はあまり変化がありません
```

```
f=1000 として、ゲインを調べて見ます。
キー入力待ち？

B? f=1000
1,000 j 0

B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000
x5 [ -94.6500 j -1.8267 ] abs[ 94.6677] arg[ -178.8944]
x6 [ -94.6498 j -1.8417 ] abs[ 94.6677] arg[ -178.8852]
```

e1 の値を出力が 100 となる値に設定して、r1 を変化させて出力インピーダンスを求める。

```
r1 = 1M として、出力が -100 となる e1 を求めます
キー入力待ち？

B? r1=1m
1,000,000 j 0

B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000
x5 [ -452.5992 j -8.7919 ] abs[ 452.6846] arg[ -178.8871]
x6 [ -452.5992 j -8.7920 ] abs[ 452.6846] arg[ -178.8871]
B? a=e1/abs(x6)*100
0.2209 j 0
```

サンプルバッチファイルの回路図と説明

```

B? /range
ac 入力信号源名は ? e1
値は ? a

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[0.2209 j 0 ]
f[1,000 ]
x5 [ -99.9811 j -1.9422 ] abs[ 100] arg[ -178.8871]
x6 [ -99.9811 j -1.9422 ] abs[ 100] arg[ -178.8871]

```

```

B? /ac
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? rl
ac rl 最低値 ? 1k
ac rl 最高値 ? 10k
ac rl ステップ ? 1k

```

```

rl[3000 ] f[1,000 ]
x5 [ -44.2770 j -0.8588 ] abs[ 44.2853] arg[ -178.8888]
x6 [ -44.2769 j -0.8611 ] abs[ 44.2853] arg[ -178.8858]
rl[4000 ] f[1,000 ]
x5 [ -51.4672 j -0.9988 ] abs[ 51.4769] arg[ -178.8883]
x6 [ -51.4672 j -1.0008 ] abs[ 51.4769] arg[ -178.8860]

```

RL=3k から 4k の範囲をさらに調べます。

```

rl[3800 ] f[1,000 ]
x5 [ -50.1806 j -0.9737 ] abs[ 50.1900] arg[ -178.8883]
x6 [ -50.1805 j -0.9758 ] abs[ 50.1900] arg[ -178.8859]

```

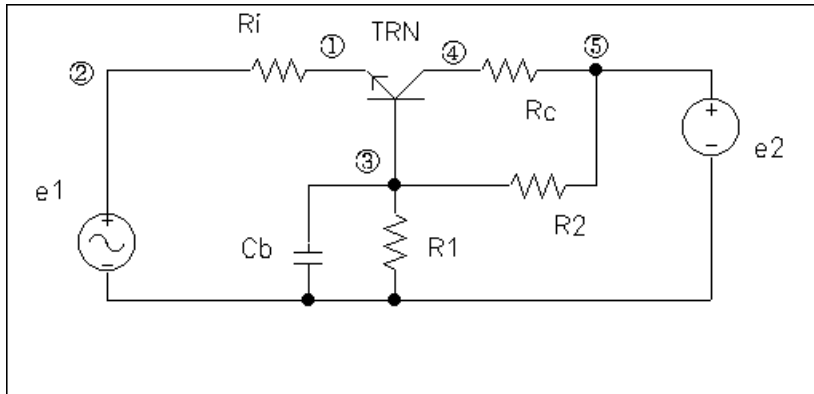
B? RL = 3.8K で出力が -50 となります。出力インピーダンスは $Z_o = 3.8K$ です
 サンプル s5 の出力インピーダンスは $Z_o = 1.3$ オームでしたから、
 エミッタ接地増幅回路の出力インピーダンスはかなり高いことが分かります。
 これは、負荷抵抗の値が出力電圧の値に大きな影響を与えるという事です。
 以上で、エミッタ接地増幅回路の設計を終了します。

1 番のバッチファイル s6.sb を終了します。

このエミッタ接地増幅回路の出力インピーダンスは 3.8K と求められました。

(3) バッチファイル名 s 7. s b で扱う回路図

ベース接地増幅回路



設計の指針

1. ノード4の直流動作点を電源電圧 e_2 の半分に設定する。
/para を使用して、 R_2 を 10 K から 150 K まで変化させてノード4が 12 V になる R_2 の値を求める。
今回は、/mk により計算データをファイルに作成して、 $x_4 = 12$ となる R_2 の値を調べる。
——> $R_2 = 106.35\text{ K}$ が得られる。
2. 入力信号の周波数が 1000 Hz の時のゲインを調べる。
サンプル s 6. s b と同様に $r_{1b1} = 800$ としてゲインを調べる。ゲインは 50 であることが分かる。従って、入力と出力の位相は同じである。ゲインはエミッタ接地よりも小さいが、エミッタホロワよりも大きい事が分かる。
3. $f = 0$ として、直流動作点を再度確認する。 ——> あまり変化なし。
4. 出力インピーダンスを調べる。
/padd を使用して2個の部品を追加する。ノード4からコンデンサ C_C を通して負荷抵抗 $R_L = 1\text{ M}$ を接続する。これは無負荷時のゲインを確認するときの負荷抵抗の値である。
この状態で、出力電圧が 100 (1でも構わない) となる e_1 を求める。
次に、/ac を使用して、入力信号の値を今求めた値を入力して、 R_L を 1 K から 100 K まで変化させて出力電圧が半分になる R_L の値を見つける。
 $Z_o = 4950$ オームが得られる。この値はエミッタ接地の場合と同程度以上の値であり、出力電圧が負荷抵抗により影響されやすいことを示している。
5. 次に、入力インピーダンスを調べる。
まず、 $R_i = 0.001$ オームとしたときの出力電圧が 100 (1でもよい) となる信号入力電圧を求める。
/ac を使用して、この入力信号電圧の時に、 R_i を 0.1 オームから 1 K オームまで変化さ

(3) バッチファイル名 s 7. s b で扱う回路図

サンプルバッチファイルの回路図と説明

せて出力電圧が先程の半分になる R_i の値を見つける。

$Z_i = 8.28$ が得られる。

従って、ベース接地増幅回路では、入力インピーダンスが非常に小さいことが分かる。このことは、信号源インピーダンスの高い入力に接続するためには、まずエミッタホロワを信号に接続してからベース接地増幅回路に接続した方が良いことを意味している。エミッタ接地増幅回路と同様に負荷変動によるゲイン変動を防ぎたいければ出力側にもエミッタ接地増幅回路が必要である。この構成では入力と出力の位相が同じであることが特徴である。

```
B? /dpart
left[ 0] right[ 2] parts[ e1      ] value[      1 j 0      ]
left[ 0] right[ 3] parts[ cb      ] value[    0.001 j 0      ]
left[ 0] right[ 3] parts[ r1      ] value[ 1e+004 j 0      ]
left[ 0] right[ 5] parts[ e2      ] value[      24 j 0      ]
left[ 1] right[ 2] parts[ ri      ] value[     100 j 0      ]
left[ 1] right[ 6] parts[ rbb1    ] value[      0.1 j 0      ]
left[ 3] right[ 5] parts[ r2      ] value[ 3e+004 j 0      ]
left[ 3] right[ 9] parts[ r1b1    ] value[      0.1 j 0      ]
left[ 4] right[ 5] parts[ rc      ] value[     5000 j 0      ]
left[ 4] right[ 8] parts[ rbb1    ] value[      0.1 j 0      ]
left[ 6] right[ 7] parts[ reb1    ] value[      26 j 0      ]
left[ 6] right[ 8] parts[ kb1     ] value[     100 j 0      ]
@ master[reb1      ] left[ 6] right[ 7]
left[ 7] right[ 9] parts[ ebb1    ] value[      0.6 j 0      ]
最大の素子番号 = 13
最大のノード番号 = 9
B? 今入力した、ベース接地増幅回路の回路リストです
```

```
B? /para
データファイル名は ? test
値を変化させる素子名は ? r2
para r2 最低値 ? 20k
para r2 最高値 ? 160k
para r2 ステップ ? 20k
r2[2e+004 ] f[0      ]
x1 [      4.8470 j 0      ] abs[      4.8470] arg[      0]
x4 [ -166.4437 j 0      ] abs[ 166.4437] arg[ 180]
r2[4e+004 ] f[0      ]
x1 [      2.7821 j 0      ] abs[      2.7821] arg[      0]
x4 [ -64.2213 j 0      ] abs[ 64.2213] arg[ 180]
r2[6e+004 ] f[0      ]
x1 [      1.9872 j 0      ] abs[      1.9872] arg[      0]
x4 [ -24.8723 j 0      ] abs[ 24.8723] arg[ 180]
r2[8e+004 ] f[0      ]
x1 [      1.5663 j 0      ] abs[      1.5663] arg[      0]
x4 [ -4.0332 j 0      ] abs[ 4.0332] arg[ 180]
r2[1e+005 ] f[0      ]
x1 [      1.3056 j 0      ] abs[      1.3056] arg[      0]
x4 [ 8.8698 j 0      ] abs[ 8.8698] arg[ 0]
r2[1.2e+005 ] f[0      ]
x1 [      1.1284 j 0      ] abs[      1.1284] arg[      0]
x4 [ 17.6450 j 0      ] abs[ 17.6450] arg[ 0]
r2[1.4e+005 ] f[0      ]
x1 [      1 j 0      ] abs[      1] arg[      0]
x4 [ 24 j 0      ] abs[ 24] arg[ 0]
r2[1.6e+005 ] f[0      ]
x1 [      0.9027 j 0      ] abs[      0.9027] arg[      0]
x4 [ 28.8147 j 0      ] abs[ 28.8147] arg[ 0]
B? R2 が 100K から 120K の間で x4 = 12V となっています
```

(3) バッチファイル名 s7. sb で扱う回路図

サンプルバッチファイルの回路図と説明

```

B?
画面の数字でも分かり易くするには、
/gpos
5
/manu

のようにモードを設定して、計算するポイントを極力減らして
/para, /ac, /range などのコマンドを実行する方法があります。

キー入力待ち？

B? /gpos
MANU ON または 最高/最低の比が10以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 32 です
5
gpos = 5 に設定しました
B? /manu

```

```

B? /para
データファイル名は ? test
値を変化させる素子名は ? r2
para r2 最低値 ? 10k
para r2 最高値 ? 150k
para r2 ステップ ? 10k
r2[1e+004 ] f[0 ]
x1 [ 7.9397 j 0 ] abs[ 7.9397] arg[ 0]
x4 [ -319.5473 j 0 ] abs[ 319.5473] arg[ 180]
r2[1.968e+004] f[0 ]
x1 [ 4.9072 j 0 ] abs[ 4.9072] arg[ 0]
x4 [ -169.4250 j 0 ] abs[ 169.4250] arg[ 180]
r2[3.873e+004] f[0 ]
x1 [ 2.8571 j 0 ] abs[ 2.8571] arg[ 0]
x4 [ -67.9349 j 0 ] abs[ 67.9349] arg[ 180]
r2[7.622e+004] f[0 ]
x1 [ 1.6299 j 0 ] abs[ 1.6299] arg[ 0]
x4 [ -7.1856 j 0 ] abs[ 7.1856] arg[ 180]
r2[1.5e+005 ] f[0 ]
x1 [ 0.9482 j 0 ] abs[ 0.9482] arg[ 0]
x4 [ 26.5626 j 0 ] abs[ 26.5626] arg[ 0]

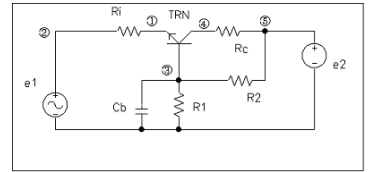
```


サンプルバッチファイルの回路図と説明

```

B? /para
データファイル名は ? test
値を変化させる素子名は ? r2
para r2 最低値 ? 100k
para r2 最高値 ? 110k
para r2 ステップ ? 1k
r2[1e+005 ] f[0 ]
x1 [ 1.3056 j 0 ] abs[ 1.3056] arg[ 0]
x4 [ 8.8698 j 0 ] abs[ 8.8698] arg[ 0]
r2[1.01e+005 ] f[0 ]
x1 [ 1.2952 j 0 ] abs[ 1.2952] arg[ 0]
x4 [ 9.3872 j 0 ] abs[ 9.3872] arg[ 0]
r2[1.02e+005 ] f[0 ]
x1 [ 1.2849 j 0 ] abs[ 1.2849] arg[ 0]
x4 [ 9.8949 j 0 ] abs[ 9.8949] arg[ 0]
r2[1.03e+005 ] f[0 ]
x1 [ 1.2749 j 0 ] abs[ 1.2749] arg[ 0]
x4 [ 10.3932 j 0 ] abs[ 10.3932] arg[ 0]
r2[1.04e+005 ] f[0 ]
x1 [ 1.2650 j 0 ] abs[ 1.2650] arg[ 0]
x4 [ 10.8824 j 0 ] abs[ 10.8824] arg[ 0]
r2[1.05e+005 ] f[0 ]
x1 [ 1.2553 j 0 ] abs[ 1.2553] arg[ 0]
x4 [ 11.3627 j 0 ] abs[ 11.3627] arg[ 0]
r2[1.06e+005 ] f[0 ]
x1 [ 1.2457 j 0 ] abs[ 1.2457] arg[ 0]
x4 [ 11.8343 j 0 ] abs[ 11.8343] arg[ 0]
r2[1.07e+005 ] f[0 ]
x1 [ 1.2364 j 0 ] abs[ 1.2364] arg[ 0]
x4 [ 12.2975 j 0 ] abs[ 12.2975] arg[ 0]
r2[1.08e+005 ] f[0 ]
x1 [ 1.2272 j 0 ] abs[ 1.2272] arg[ 0]
x4 [ 12.7526 j 0 ] abs[ 12.7526] arg[ 0]
r2[1.09e+005 ] f[0 ]
x1 [ 1.2182 j 0 ] abs[ 1.2182] arg[ 0]
x4 [ 13.1996 j 0 ] abs[ 13.1996] arg[ 0]
r2[1.1e+005 ] f[0 ]
x1 [ 1.2093 j 0 ] abs[ 1.2093] arg[ 0]
x4 [ 13.6388 j 0 ] abs[ 13.6388] arg[ 0]
B? R2 が 106K から 107K の間で x4 = 12V となっています

```



```

r2[1.063e+005] f[0 ]
x1 [ 1.2429 j 0 ] abs[ 1.2429] arg[ 0]
x4 [ 11.9742 j 0 ] abs[ 11.9742] arg[ 0]
r2[1.064e+005] f[0 ]
x1 [ 1.2425 j 0 ] abs[ 1.2425] arg[ 0]
x4 [ 11.9974 j 0 ] abs[ 11.9974] arg[ 0]
r2[1.064e+005] f[0 ]
x1 [ 1.2420 j 0 ] abs[ 1.2420] arg[ 0]
x4 [ 12.0206 j 0 ] abs[ 12.0206] arg[ 0]

```

```

B? 大体、R2 = 106.35K で x4 = 12V となっていますから、この値に決定します。
r2=106.35k
106,350 j 0

```

```

B? 周波数 f = 1000Hz における、ゲインを確認します

```

(3) バッチファイル名 s 7. s b で扱う回路図

サンプルバッチファイルの回路図と説明

```

B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1      j 0      ]
f[1,000    ]
x1 [    0.0036 j -1.564556e-005] abs[    0.0036] arg[    -0.2510]
x4 [   49.3282 j 7.745328e-004] abs[   49.3282] arg[8.996376e-004]

```

B? サンプル s6.sb と同様に、hie を 800 とした場合のゲインと動作点も確認します

```

r1b1=800 と入力して、hie=800 に設定します
r1b1=800
      800 j 0

```

```

B? ゲインを確認します
/range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1      j 0      ]
f[1,000    ]
x1 [    0.0765 j -1.344054e-005] abs[    0.0765] arg[    -0.0101]
x4 [   45.7201 j 6.653730e-004] abs[   45.7201] arg[8.338353e-004]
B? ゲインは 1 割程低下しています

```

次は、f=0 としてから、動作点を確認します
キー入力待ち？

```

B? f=0
      0 j 0

```

```

B? /point
f[0      ]
x1 [    1.2328 j 0      ] abs[    1.2328] arg[    0]
x4 [   12.4756 j 0      ] abs[   12.4756] arg[    0]
B? 動作点は 3% 程高くなっていますが、あまり大きな変動ではありません。

```

B? 次は、出力インピーダンスを求めます
現在は無負荷の状態ですから、交流的に負荷抵抗を追加します。
ノード 0 と 10 の間に RL=1M と ノード 4 と 10 の間に cc=1000uF を追加します
キー入力待ち？

```

B? /padd

```

```

B? /point
f[0      ]
x1 [    1.2328 j 0      ] abs[    1.2328] arg[    0]
x4 [   12.4756 j 0      ] abs[   12.4756] arg[    0]
B? 動作点は 3% 程高くなっていますが、あまり大きな変動ではありません。

```

サンプルバッチファイルの回路図と説明

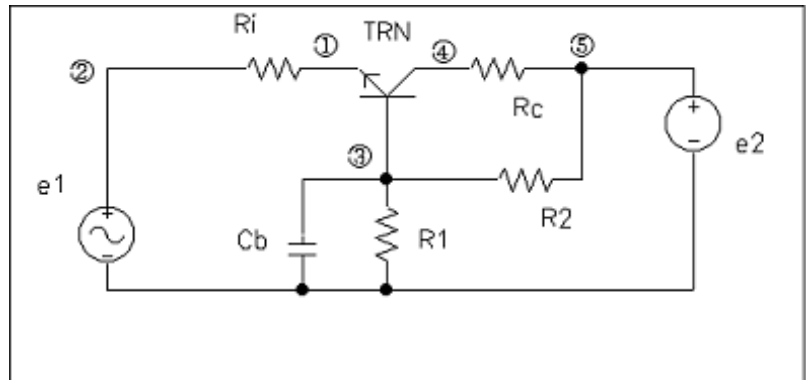
B? 次は、出力インピーダンスを求めます
 現在は無負荷の状態ですから、交流的に負荷抵抗を追加します。
 ノード 0 と 10 の間に $R_L=1M$ と ノード 4 と 10 の間に $C_C=1000\mu F$ を追加します

```
left[ 0] right ? 10
素子名は ? rl
値は ? 1m

left[ 0] right ? /
left[ 1] right ? /
left[ 2] right ? /
left[ 3] right ? /

left[ 4] right ? 10
素子名は ? cc
値は ? 1000u

left[ 4] right ? *
```



B? 現在は負荷抵抗が 1M 追加された状態です。
 この状態で出力電圧が 100 となる、e1を求めます。
 $f=1000$
 $1,000 \text{ j } 0$

```
B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x4 [ 45.4927 j 6.620266e-004] abs[ 45.4927] arg[8.337898e-004]
x10 [ 45.4927 j 6.692670e-004] abs[ 45.4927] arg[8.429087e-004]
B? a=e1/x10*100 の値の e1 で、x10 の電圧が 100 となるはずです。
```

```
B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x4 [ 45.4927 j 6.620266e-004] abs[ 45.4927] arg[8.337898e-004]
x10 [ 45.4927 j 6.692670e-004] abs[ 45.4927] arg[8.429087e-004]
B? a=e1/x10*100 の値の e1 で、x10 の電圧が 100 となるはずです。
```

サンプルバッチファイルの回路図と説明

```

B? /range
ac 入力信号源名は ? e1
値は ? a

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[2.198 j -3.234e-005]
f[1,000 ]
x4 [ 100 j -1.591549e-005] abs[ 100] arg[-9.118907e-006]
x10 [ 100 j 0 ] abs[ 100] arg[ 0]
B? この値の e1 で
/ac で RL を変化させて、x10 が 50 となる値を見つけます。

```

```

B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? rl
ac rl 最低値 ? 1k
ac rl 最高値 ? 100k
ac rl ステップ ? 1k

```

```

rl[4281 ] f[1,000 ]
x4 [ 46.3591 j -9.357424e-004] abs[ 46.3591] arg[ -0.0012]
x10 [ 46.3591 j 7.876171e-004] abs[ 46.3591] arg[9.734263e-004]
rl[5456 ] f[1,000 ]
x4 [ 52.4396 j -7.398792e-004] abs[ 52.4396] arg[-8.083958e-004]
x10 [ 52.4396 j 7.899306e-004] abs[ 52.4396] arg[8.630822e-004]

```

B? RL = 5K の近くで、x10 = 50 となる事が分かります。

```

rl[4900 ] f[1,000 ]
x4 [ 49.7424 j -8.238691e-004] abs[ 49.7424] arg[-9.489731e-004]
x10 [ 49.7424 j 7.917947e-004] abs[ 49.7424] arg[9.120282e-004]
rl[5000 ] f[1,000 ]
x4 [ 50.2500 j -8.077113e-004] abs[ 50.2500] arg[-9.209641e-004]
x10 [ 50.2500 j 7.917959e-004] abs[ 50.2500] arg[9.028172e-004]

```

B? RL = 4950 で、x10 = 50 となります。
出力インピーダンスは Zo = 4950 オームです

```

ri[8.2 ] f[1,000 ]
x4 [ 50.2434 j -0.0048 ] abs[ 50.2434] arg[ -0.0054]
x10 [ 50.2434 j -0.0048 ] abs[ 50.2434] arg[ -0.0054]
ri[8.3 ] f[1,000 ]
x4 [ 49.9403 j -0.0048 ] abs[ 49.9403] arg[ -0.0055]
x10 [ 49.9403 j -0.0048 ] abs[ 49.9403] arg[ -0.0055]

```

サンプルバッチファイルの回路図と説明

B?

次は、入力インピーダンスを求めます。

まず、 $R_i = 0.001$ とした時の出力電圧が 100 となる $e1$ を求めます。

B? /range

データファイル名は ? test

ac 入力信号源名は ? e1

値は ? 1

range 周波数 最低値 ? 1000

range 周波数 最高値 ? 1000

range 周波数 ステップ ? 1

e1[1 j 0]

f[1,000]

x4 [594.9675 j 0.1132] abs[594.9675] arg[0.0109]

x10 [594.9675 j 0.1133] abs[594.9675] arg[0.0109]

B? a=e1/x10*100

0.1681 j -3.201676e-005

B? この値の $e1$ で、 $x10$ の電圧が 100 となるはずです。

B? /ac

データファイル名は ? test

ac 入力信号源名は ? e1

値は ? a

値を変化させる素子名は ? ri

ac ri 最低値 ? 0.1

ac ri 最高値 ? 1k

ac ri ステップ ? 100

B? /ac

データファイル名は ? test

ac 入力信号源名は ? e1

値は ? a

値を変化させる素子名は ? ri

ac ri 最低値 ? 0.1

ac ri 最高値 ? 1k

ac ri ステップ ? 100

```

ri[7.848 ] f[1,000 ]
x4 [ 51.3414 j -0.0048 ] abs[ 51.3414] arg[ -0.0053]
x10 [ 51.3414 j -0.0048 ] abs[ 51.3414] arg[ -0.0053]
ri[12.74 ] f[1,000 ]
x4 [ 39.3855 j -0.0046 ] abs[ 39.3855] arg[ -0.0066]
x10 [ 39.3855 j -0.0045 ] abs[ 39.3855] arg[ -0.0066]

```

サンプルバッチファイルの回路図と説明

```

B? /ac
データファイル名は ? test
ac 入力信号源名は ? el
値は ? a

値を変化させる素子名は ? ri
ac ri 最低値 ? 8
ac ri 最高値 ? 9
ac ri ステップ ? 0.1

```

```

ri[8.2      ] f[1,000      ]
x4 [ 50.2434 j -0.0048 ] abs[ 50.2434] arg[ -0.0054]
x10 [ 50.2434 j -0.0048 ] abs[ 50.2434] arg[ -0.0054]
ri[8.3      ] f[1,000      ]
x4 [ 49.9403 j -0.0048 ] abs[ 49.9403] arg[ -0.0055]
x10 [ 49.9403 j -0.0048 ] abs[ 49.9403] arg[ -0.0055]

```

B? Ri = 8.28 で x10 = 50 となっています

以上のことからベース接地増幅回路の特徴をまとめると、

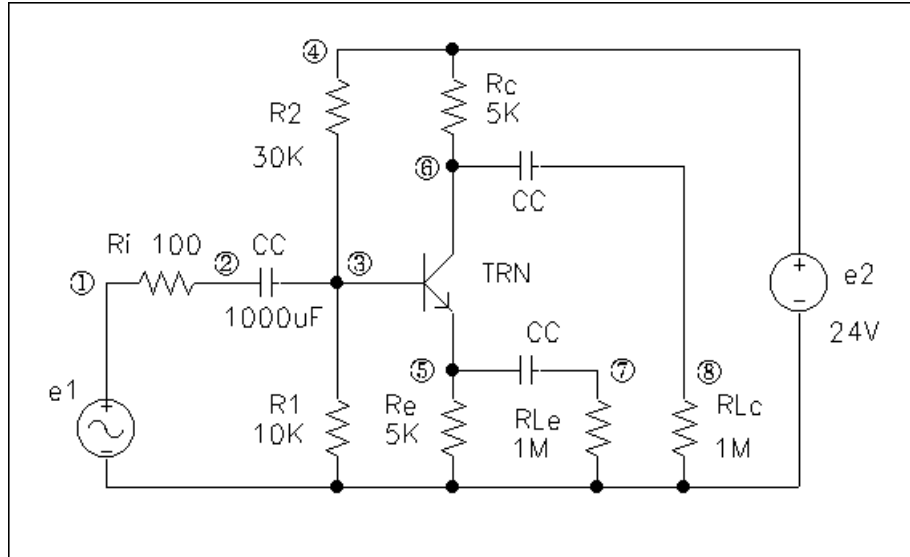
- 1 入力インピーダンスが非常に低い
- 2 出力インピーダンスが非常に高い
- 3 ゲインは中程度
- 4 入力と出力の位相が同じ（エミッタ接地だけが位相が反転する）

ということが分かりました。

以上でベース接地増幅回路の設計を終了します

(4) バッチファイル名 s 8. s b で扱う回路図

反転増幅回路 トランジスタのコレクタとエミッタの両方から出力を得る回路



設計の指針

1. ノード6の直流動作点を電源電圧e2の半分に設定する。

現在の電圧は、18.7321Vなので、/paraを使用してR2を1Kから30Kまで変化させノード6が12VになるR2の値を調べる。

x6 = 12となる時に同時にx5も12となり、トランジスタの $V_{ce} = 0$ になってしまうので、x6 = 14.4VになるR2をさがすことにする。

R2 = 13Kでx6 = 14.4V, x5 = 9.8Vとなる。

従って、 $V_{ce} = x6 - x5 = 4.6 > V_{ce(sat)}$ となり動作が保証される。

2. ノード7及びノード8のそれぞれのゲインを確認する。

入力信号の周波数を1000Hzとして、ゲインを確認する。

ノード7のゲインは0.9824、ノード8のゲインは-0.9726であることが分かる。

したがって、コレクタ側は入力と位相が反転しエミッタ側は入力と同位相の出力が得られ、ゲインはどちらも約1である。しかし、コレクタ側の方が少しだけゲインが小さい。

3. コレクタ側とエミッタ側のゲインの絶対値を等しくする。

/acを使用して、Rcを変化させて、x7とx8の出力が同じになるRcの値を求める。

Rc = 5.075Kで等しくなることが分かる。

(4) バッチファイル名 s 8. s b で扱う回路図

4. 出力インピーダンスを調べる。コレクタ側とエミッタ側のそれぞれの出力インピーダンスを調べる。

コレクタ側の出力インピーダンス $Z_{oc} = 5\text{ K}$

エミッタ側の出力インピーダンス $Z_{oe} = 1.5\text{ オーム}$

負荷抵抗が 1 M の時の出力電圧を 1 として、負荷抵抗を 100 オーム から 1 M まで変化させたときの出力電圧を $/ac$ を使用して確認すると下表が得られる。

負荷抵抗	コレクタ側出力	エミッタ側出力
1	0.0002	0.7311
10	0.002	0.9653
100	0.0194	0.9964
1 K	0.16	0.9996
10 K	0.667	1.0
100 K	0.9565	1.0
1 M	1.0	1.0

従って、エミッタ側では負荷抵抗が 100 オーム 以上の範囲で変動しても出力電圧には影響が現れないが、コレクタ側にはエミッタホロワを接続するなど負荷変動の影響がコレクタ側に伝わらないための工夫が必要であることが分かる。

反転増幅器では直流動作点を適正に選び $V_{ce} > V_{ce(sat)}$ を満足するように設計しなければならない。次に、コレクタ側の出力はエミッタホロワに接続して負荷変動がゲインに影響しないための工夫が必要である。さらに、コレクタ抵抗 R_c を調整してコレクタ側とエミッタ側のゲインが等しくなるように設計する必要がある。

5. エミッタ側の負荷抵抗の変動があっても、エミッタ側のゲインはあまり影響を受けない事が分かった。この時に、コレクタ側のゲインはどうなっているのかを調べる。

エミッタ側の負荷抵抗を 100 オーム から 10 K まで変動させると、エミッタ側のゲインは変化しないが、コレクタ側のゲインは約 -33 から 1 まで大きく変化している。従って、エミッタ側の負荷抵抗も変動しないようにエミッタホロワのようなバッファを接続しておいた方が良いと思われる。

6. 負荷抵抗が固定値であることが保証されている場合には、コレクタ側もエミッタ側もエミッタホロワなどを接続しなくてもよい。

サンプルバッチファイルの回路図と説明

ノード 6 の現在の電圧を確認します。

```

B? ノード 6 の動作点を 12V に設定します。
一度現在の動作点を確認します。
f=0
/point
f=0
      0 j 0

B? /point
f[0
x5 [ 5.3206 j 0 ] abs[ 5.3206] arg[ 0]
x6 [ 18.7321 j 0 ] abs[ 18.7321] arg[ 0]
B? 目標の電圧より x6 が高いので、R2 を減らしてバイアス電流を増加させます。
/para
test
r2
1k
30k
1k

```

/para で R2 を変化させて、x6 が 12 に近づく値を見つけます。

```

B? /para
データファイル名は ? test
値を変化させる素子名は ? r2
para r2 最低値 ? 1k
para r2 最高値 ? 30k
para r2 ステップ ? 1k

```

```

r2[9655 ] f[0
x5 [ 11.4980 j 0 ] abs[ 11.4980] arg[ 0]
x6 [ 12.6158 j 0 ] abs[ 12.6158] arg[ 0]
r2[1.409e+004] f[0
x5 [ 9.2553 j 0 ] abs[ 9.2553] arg[ 0]
x6 [ 14.8363 j 0 ] abs[ 14.8363] arg[ 0]
r2[2.056e+004] f[0
x5 [ 7.1579 j 0 ] abs[ 7.1579] arg[ 0]
x6 [ 16.9130 j 0 ] abs[ 16.9130] arg[ 0]

```

```

x6=12V で x5 も 12V となり Vce=0となるので、x6=14.4V に設定します
= 10K から 20K の間で x6 = 14.4V となります。

```

x6=12 にすると、x5 も同じ位の電圧になって、コレクタとエミッタ間の電圧が小さくなります。x6 の目標値を 14.4 に変更します。

```

B? /para
データファイル名は ? test
値を変化させる素子名は ? r2
para r2 最低値 ? 10k
para r2 最高値 ? 20k
para r2 ステップ ? 1k

```

サンプルバッチファイルの回路図と説明

```

r2[1.3e+004 ] f[0 ]
x5 [ 9.7252 j 0 ] abs[ 9.7252] arg[ 0]
x6 [ 14.3711 j 0 ] abs[ 14.3711] arg[ 0]

```

B? R2 = 13K で x6=14.4V になります、この時 x5=9.8V となっています。
従って、 $V_{ce} = x6 - x5 = 4.6V$ となります。

```

B? R2=13K に決定します。
r2=13k
 13,000 j 0

```

```

B? 次に、ノード7と8の交流利得を計算します。
f=1000
/dis
7
8

```

```

B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

値を変化させる素子名は ? rlc
ac rlc 最低値 ? 1m
ac rlc 最高値 ? 1m
ac rlc ステップ ? 1
e1[1 j 0 ]
rlc[1e+006 ] f[1,000 ]
x7 [ 0.9824 j 2.763706e-005] abs[ 0.9824] arg[ 0.0016]
x8 [ -0.9726 j -2.736343e-005] abs[ 0.9726] arg[ -179.9984]
B? エミッタ、コレクタそれぞれの負荷抵抗が 1M の時
ノード8の利得が、ノード7の利得より 1% 低い

RLc=RLe=10K に変更して、交流利得を確認してみます。

```

負荷抵抗を RLc=RLe=10k に変更して、交流利得を確認します。

```

B? rlc=10k
 10,000 j 0

B? rle=10k
 10,000 j 0

```

```

B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1

```

(4) バッチファイル名 s8. sb で扱う回路図

サンプルバッチファイルの回路図と説明

```

B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1000
range 周波数 最高値 ? 1000
range 周波数 ステップ ? 1
e1[1      j 0      ]
f[1,000    ]
x7 [    0.9822 j 4.325543e-005] abs[    0.9822] arg[    0.0025]
x8 [   -0.9725 j -4.282716e-005] abs[    0.9725] arg[ -179.9975]
B? エミッタ、コレクタそれぞれの負荷抵抗が 10K の時
ノード8の利得が、ノード7の利得より 1% 低い
負荷抵抗 10K までは、利得もあまり変化していない。

```

```

次は、
Rc を増加させて、コレクタ側の利得をエミッタ側の利得と等しくする
/ac
test
e1
1
rc
5k
5.1k
0.01k

```

```

rc[5070    ] f[1,000    ]
x7 [    0.9822 j 4.325543e-005] abs[    0.9822] arg[    0.0025]
x8 [   -0.9815 j -4.317665e-005] abs[    0.9815] arg[ -179.9975]
rc[5080    ] f[1,000    ]
x7 [    0.9822 j 4.325543e-005] abs[    0.9822] arg[    0.0025]
x8 [   -0.9828 j -4.322624e-005] abs[    0.9828] arg[ -179.9975]

```

Rc=5075 で、x7 と x8 の利得が等しくなっている

```

B? rc=5075
    5,075 j 0

B? RLc=RLe=1M に戻して、エミッタ側とコレクタ側の出カインピーダンスを求めます。

```

サンプルバッチファイルの回路図と説明

```

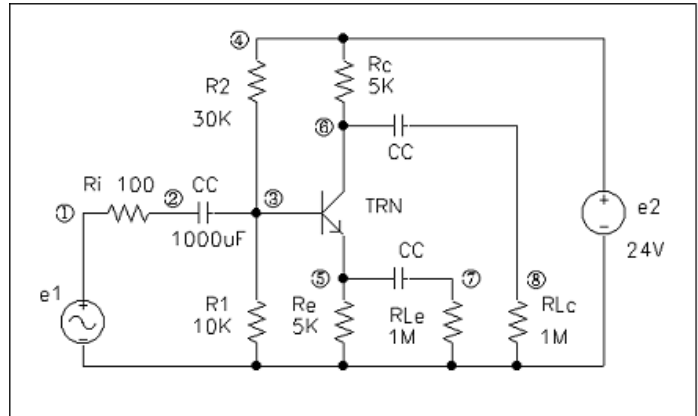
B? rlc=1m
1,000,000 j 0

B? rle=1m
1,000,000 j 0

B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1

```



```

e1[1 j 0]
f[1,000]
x7 [ 0.9824 j 2.763706e-005] abs[ 0.9824] arg[ 0.0016]
x8 [-0.9871 j -2.777180e-005] abs[ 0.9871] arg[-179.9984]
B? まず、コレクタ側の出力インピーダンスを求めます。
x8=1 となる、e1 を求めて、RLc を変化させて x8=0.5 となる RLc を見つけます

```

```

B? e1=1/x8
-1.0130 j 2.850000e-005

```

```

B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? e1

値を変化させる素子名は ? rlc
ac rlc 最低値 ? 1k
ac rlc 最高値 ? 1m
ac rlc ステップ ? 10k

```

```

rlc[4642] f[1,000]
x7 [-0.9951 j -7.349554e-012] abs[ 0.9951] arg[-180]
x8 [ 0.4801 j 7.788229e-006] abs[ 0.4801] arg[9.294158e-004]
rlc[1e+004] f[1,000]
x7 [-0.9951 j -7.349554e-012] abs[ 0.9951] arg[-180]
x8 [ 0.6667 j 6.933314e-006] abs[ 0.6667] arg[5.958300e-004]

```

コレクタ側の出力インピーダンスは $Z_{oc}=5K$ です。

コレクタ側の出力インピーダンスは $Z_{oc}=5K$ と求められました。
 負荷抵抗 RLc が $1K$ から $1M$ まで変化すると、 $x8$ は 0.16 から 1 まで変化します。

B? $x8=1$ を基準に考えると、16% まで利得が下がる訳です。

次は、エミッタ側の出力インピーダンスを求めます。
 $x7=1$ となる、 $e1$ を求めて、 RLe を変化させて、 $x=0.5$ となる RLe を見つけます
 キー入力待ち？

```
B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1
```

```
range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
```

```
e1[1          j 0          ]
f[1,000      ]
x7 [ 0.9824 j 2.763706e-005] abs[ 0.9824] arg[ 0.0016]
x8 [ -0.9871 j -2.777180e-005] abs[ 0.9871] arg[ -179.9984]
B? e1=1/x7
1.0180 j -2.863896e-005
```

```
B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? e1
```

```
値を変化させる素子名は ? rle
ac rle 最低値 ? 1
ac rle 最高値 ? 10k
ac rle ステップ ? 1k
```

```
rle[1          ] f[1,000      ]
x7 [ 0.4270 j 0.0294 ] abs[ 0.4280] arg[ 3.9432]
x8 [-2,135.0760 j -147.1056] abs[2,140.1377] arg[ -176.0586]
rle[2.783      ] f[1,000      ]
x7 [ 0.6754 j 0.0264 ] abs[ 0.6759] arg[ 2.2369]
x8 [-1,214.1842 j -47.3879 ] abs[1,215.1085] arg[ -177.7650]
```

$RLe=1.5$ 位で $x7=0.5$ となっています。

B? エミッタ側の出力インピーダンスは $Zoe=1.5$ です。
 コレクタ側の出力インピーダンスは $Zoc=5K$ でしたから、
 コレクタ側に比較してかなり低い値です。これは、負荷抵抗の大きさで
 出力電圧があまり変化しないことを意味しています。
 逆に、コレクタ側では負荷抵抗の大きさによって出力電圧が大きく変化します。

RLe が 1 から 10K まで変化しても $x7$ は 0.47 から 1 までしか変化していません。

サンプルバッチファイルの回路図と説明

B? 最後に Ri を変化させて、入力インピーダンスを求めます。
 RLc=RLe=10K として、Ri=0.001 の時に、x7=1 となる e1 を求めます。
 次に、Ri を変化させて、x7=0.5 となる Ri を見つけます。

```
B? rlc=10k
    10,000 j 0

B? rle=10k
    10,000 j 0

B? ri=0.001
    0.0010 j 0

B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
```

```
B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1
f[1,000
x7 [ 0.9999 j 4.454108e-005] abs[ 0.9999] arg[ 0.0026]
x8 [-0.9998 j -4.448616e-005] abs[ 0.9998] arg[ -179.9975]
B? e1=1/x7
    1.0001 j -4.455068e-005
```

```
B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? e1

値を変化させる素子名は ? ri
ac ri 最低値 ? 1
ac ri 最高値 ? 10k
ac ri ステップ ? 1k
```

サンプルバッチファイルの回路図と説明

```

ri[3594] f[1,000]
x7 [ 0.6073 j -6.848726e-006] abs[ 0.6073] arg[-6.460924e-004]
x8 [ -0.6073 j 6.880442e-006] abs[ 0.6073] arg[ 179.9994]
ri[1e+004] f[1,000]
x7 [ 0.3573 j -6.594672e-006] abs[ 0.3573] arg[ -0.0011]
x8 [ -0.3573 j 6.613203e-006] abs[ 0.3573] arg[ 179.9989]
B? Ri=5.5K で x7=0.5 となります。入力インピーダンスは Zi=5.5K です。

```

```

ri[5000] f[1,000]
x7 [ 0.5265 j -7.159546e-006] abs[ 0.5265] arg[-7.791829e-004]
x8 [ -0.5264 j 7.186978e-006] abs[ 0.5264] arg[ 179.9992]
ri[5500] f[1,000]
x7 [ 0.5027 j -7.179454e-006] abs[ 0.5027] arg[-8.183491e-004]
x8 [ -0.5026 j 7.205628e-006] abs[ 0.5026] arg[ 179.9992]

```

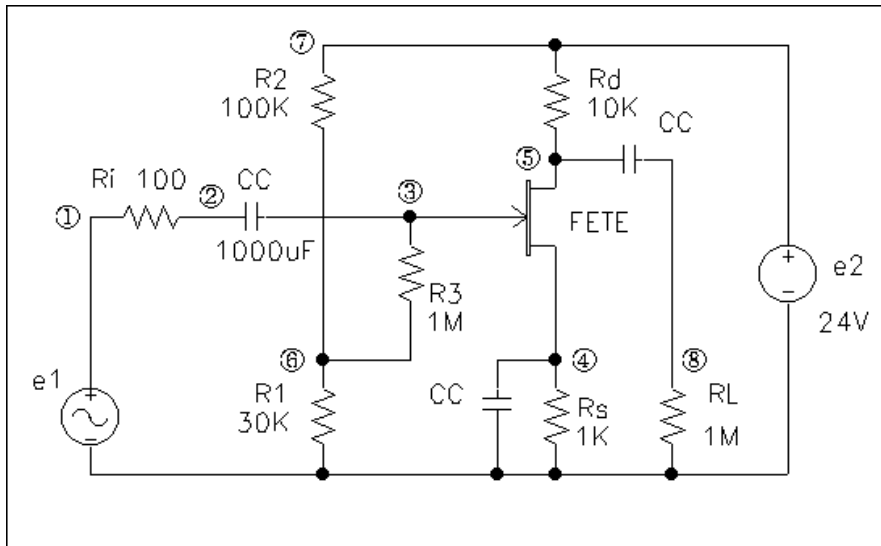
B? Ri=5.5K で x7=0.5 となります。入力インピーダンスは Zi=5.5K です。

反転増幅回路の特性は、
 入力インピーダンスはエミッタホロワ回路の特性、
 出力インピーダンスは、エミッタ側はエミッタホロワ回路の特性
 コレクタ側はエミッタ接地回路の特性となっています。

以上で、反転増幅回路の設計を終了します。

(5) バッチファイル名 s 9. s b で扱う回路図

ソース接地増幅回路 (N型エンハンスメントタイプ F E Tを使用)



設計の指針

1. N型エンハンスメントタイプのF E Tでは、ゲート電圧をソース電圧よりも高くしなければドレイン電流が流れない。
2. この回路形式はトランジスタのエミッタ接地増幅回路に対応するものである。
3. ノード5の直流動作点を電源電圧e 2の半分に設定する。
/point を使用して、現在のノード5の電圧を確認すると、負の値になっているので、/para
を使用して、R 1を1 Kから3 0 Kまで変化させて適当な値を見つける。
R 1 = 6 . 5 Kでノード5が1 2 Vになる。
4. 入力信号の周波数が1 0 0 0 H zの時のゲインを確認する。ゲインは- 1 9 . 0 3 8と分かる。ゲインはエミッタ接地増幅回路よりも小さい。出力の位相が入力と逆になっていることもエミッタ接地増幅回路と同じである。
5. 出力インピーダンスを求める。Z o = 9 Kが得られる。F E Tのソース接地増幅回路では、トランジスタのエミッタ接地増幅回路よりも出力インピーダンスが高いことがわかる。出力にエミッタホロワのようなバッファ回路が必要である。
これには、F E Tのソースホロワ増幅回路に対応する。
6. 入力インピーダンスを求める。Z i = 1 Mが得られる。
この値はR 3の値であり、R 3 = 1 0 Mなら入力インピーダンスも1 0 Mとなる。
入力インピーダンスはエミッタホロワよりも高いので、信号源インピーダンスの高い応用においても使用する事が可能である。

(5) バッチファイル名 s 9. s b で扱う回路図

サンプルバッチファイルの回路図と説明

回路リストは次のようになる。

```

B? /dpart
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 4] parts[ rs ] value[ 1000 j 0 ]
left[ 0] right[ 4] parts[ cc ] value[ 0.001 j 0 ]
left[ 0] right[ 6] parts[ r1 ] value[ 3e+004 j 0 ]
left[ 0] right[ 7] parts[ e2 ] value[ 24 j 0 ]
left[ 0] right[ 8] parts[ r1 ] value[ 1e+004 j 0 ]
left[ 1] right[ 2] parts[ ri ] value[ 100 j 0 ]
left[ 2] right[ 3] parts[ cc ] value[ 0.001 j 0 ]
left[ 3] right[ 6] parts[ r3 ] value[ 1e+006 j 0 ]
left[ 3] right[ 10] parts[ r1b1 ] value[ 0.1 j 0 ]
left[ 4] right[ 9] parts[ r1b1 ] value[ 0.1 j 0 ]
left[ 5] right[ 7] parts[ rd ] value[ 1e+004 j 0 ]
left[ 5] right[ 8] parts[ cc ] value[ 0.001 j 0 ]
left[ 5] right[ 11] parts[ r1b1 ] value[ 0.1 j 0 ]
left[ 6] right[ 7] parts[ r2 ] value[ 1e+005 j 0 ]
left[ 9] right[ 10] parts[ r2b1 ] value[ 1e+010 j 0 ]
left[ 9] right[ 11] parts[ q1b1 ] value[ 0.004 j 0 ]
@ master[r2b1 ] left[ 9] right[ 10]
left[ 9] right[ 11] parts[ r3b1 ] value[ 1e+005 j 0 ]
最大の素子番号 = 18
最大のノード番号 = 11

```

```

B? ノード5の動作点を電源電圧 e2 の 1/2 の 12V に設定します。
現在の動作点を確認します。
/point
f[0 ]
x5 [ -19.8192 j 0 ] abs[ 19.8192] arg[ 180]
B? ノード5が負の電圧になっています。
これは、ドレイン電流が大き過ぎる事を表わしています。
ドレイン電流が大きいということは、ノード6の電圧が高いということです。
/para で R1 を 1K から 30K まで変化させて調べましょう。

```

```

B? /para
データファイル名は ? test
値を変化させる素子名は ? r1
para r1 最低値 ? 1k
para r1 最高値 ? 30k
para r1 ステップ ? 1k

```

```

r1[6135 ] f[0 ]
x5 [ 12.6724 j 0 ] abs[ 12.6724] arg[ 0]
r1[7696 ] f[0 ]
x5 [ 10.1064 j 0 ] abs[ 10.1064] arg[ 0]

```

B? R1=6.5K で動作点が 12V になります。

```

r1=6.5k
6,500 j 0
B? /point
f[0 ]
x5 [ 12.0654 j 0 ] abs[ 12.0654] arg[ 0]

```

サンプルバッチファイルの回路図と説明

```

B? 1000Hz におけるゲインを確認します。
キー入力待ち？

B? f=1000
    1,000 j 0

B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1          j 0          ]
f[1,000       ]
x5 [ -19.0384 j -0.0114   ] abs[  19.0384] arg[ -179.9656]
B? ゲインは -19.0384 と求められました。
- 記号は出力の位相が反転することを表わしています。

```

B? RL=10K を 1M に変更して、無負荷状態でのゲインを求めた後、出力インピーダンスを求めます。

```

B? rl=1m
    1,000,000 j 0

B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1          j 0          ]
f[1,000       ]
x5 [ -36.0193 j -0.0209   ] abs[  36.0193] arg[ -179.9667]
B? a=1/abs(x5)
    0.0278 j 0

```

```

B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? a

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[0.02776     j 0          ]
f[1,000       ]
x5 [          -1 j -5.806591e-004] abs[          1] arg[ -179.9667]

```

(5) バッチファイル名 s 9. s b で扱う回路図

サンプルバッチファイルの回路図と説明

B? RL を 1K から 100K まで変化させて、出力が 0.5 となる RL を見つけます。
キー入力待ち？

B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? rl
ac rl 最低値 ? 1k
ac rl 最高値 ? 100k
ac rl ステップ ? 1k

```
rl[8577      ] f[1,000      ]
x5 [  -0.4899 j -2.941220e-004] abs[   0.4899] arg[ -179.9656]
rl[1.166e+004] f[1,000      ]
x5 [  -0.5670 j -3.399397e-004] abs[   0.5670] arg[ -179.9656]
```

B? RL=9K で出力が 0.5 となるので、出力インピーダンスは 9K です。
キー入力待ち？

B? RL=9K で出力が 0.5 となるので、出力インピーダンスは 9K です。
rl
1,000,000 j 0

B? 次に、入力インピーダンスを求めます。
まず、Ri=0.001、(RL=1M) として、出力が 1V となる e1 を求めます。
キー入力待ち？

B? ri=0.001
0.0010 j 0

B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1
range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1]
f[1,000]
x5 [-36.0228 j -0.0209] abs[36.0229] arg[-179.9667]
B? a=1/abs(x5)
0.0278 j 0

B? Ri を 100 から 10M まで変化させて、入力インピーダンスを求めます。
キー入力待ち？

サンプルバッチファイルの回路図と説明

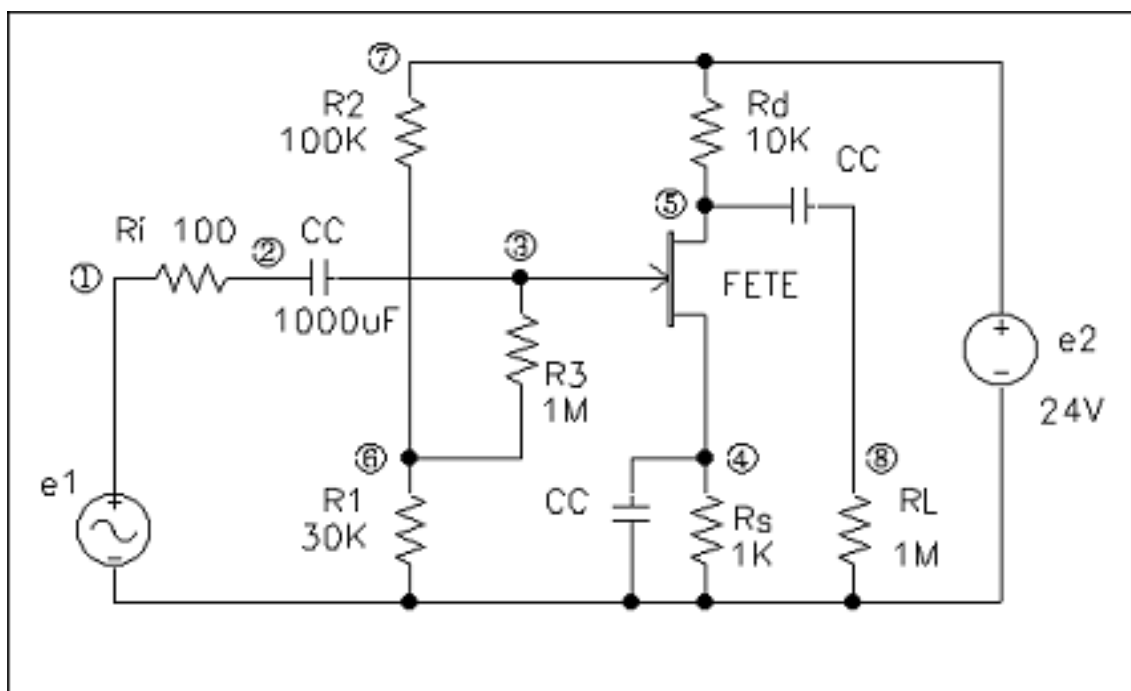
```

ri[1e+006] f[1,000]
x5 [-0.5015 j -2.911442e-004] abs[ 0.5015] arg[ -179.9667]
ri[2.154e+006] f[1,000]
x5 [-0.3183 j -1.847832e-004] abs[ 0.3183] arg[ -179.9667]

```

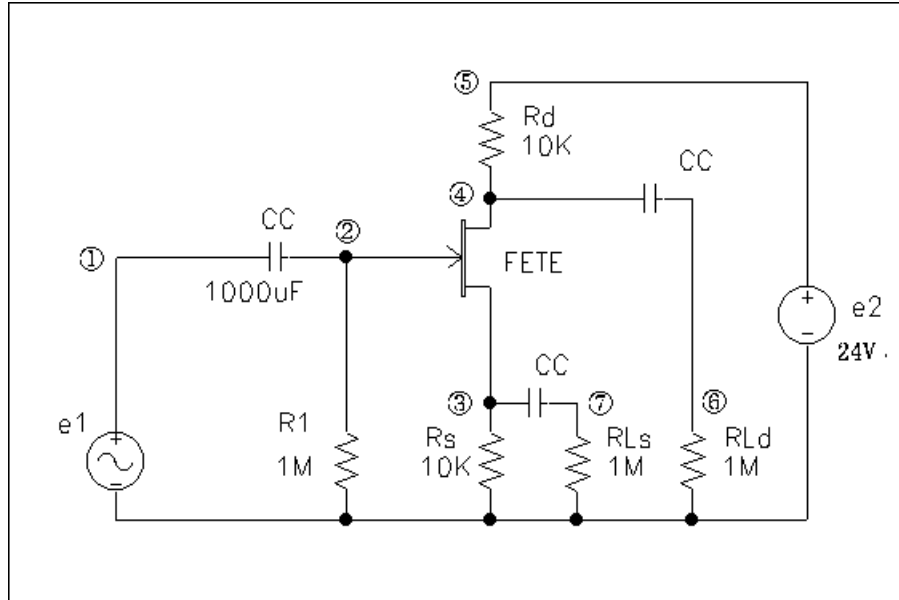
B? 入力インピーダンスは 1M です

FET のソース接地増幅回路は、トランジスタのエミッタ接地増幅回路に似ていますがゲインが低めです。
出力インピーダンスは同等ですが、入力インピーダンスはかなり高い事が分かります
以上で、ソース接地増幅回路の解析を終了します。



(6) バッチファイル名 s10. sb で扱う回路図

反転増幅回路 (N型デプリーションタイプ FET を使用)



設計の指針

1. N型デプリーションタイプのFETでは、 $V_{gs} = 0V$ で最もドレイン電流が流れる。ソース電圧がゲート電圧よりも高くなるに従って、ドレイン電流が減少する。
2. この回路形式はトランジスタの反転増幅回路に対応するものである。
3. ノード4の電圧を電源電圧の半分に設定する。

/point を使用して、現在の電圧を調べると、

ノード3の電圧は1.64Vであり、ノード4の電圧は22.358Vが得られる。

このままでは、出力電圧の振幅があまり得られないので調整をする。

FETのパラメータ $e1b1$ を現在の2Vから6Vに変更して確認すると、

ノード3の電圧は5.8834V、ノード4の電圧は18.117Vとなる。これなら、ソース側は5.8834Vを中心にプラス・マイナス5V程度、ドレイン側は18.117Vを中心にプラス・マイナス5V程度の出力が得られるようになる。

4. ノード3とノード4のゲインを求める。

ノード3のゲインは0.9706、ノード4のゲインは-0.9706となる。

トランジスタとは異なり、ゲート電流が流れないためノード3とノード4のゲインは絶対値が等しい。また、ソース側は非反転、ドレイン側は反転出力になっている。

5. 出力インピーダンスを求める。

ドレイン側の出力インピーダンス $Z_{od} = 9.8K$ が得られる。

ソース側の出力インピーダンス $Z_{os} = 270$ オームが得られる。

(6) バッチファイル名 s10. sb で扱う回路図

サンプルバッチファイルの回路図と説明

ソース側の出力インピーダンスはトランジスタの場合のエミッタ側に比べて、出力インピーダンスは高めの値である。

これは、FETの g_m をトランジスタの h_{fe} に換算すると約10分の1程度しかないためである。出力インピーダンスを低くするには g_m の大きなFETを選ぶこと。

6. トランジスタの反転増幅回路の場合と同様に、ソース側の負荷抵抗の変動がドレイン側に影響を及ぼすかどうかを確認する。

ソース側の負荷抵抗を100オームから10Kまで変動させたとき、負荷抵抗が1K以上であれば、ソース側のゲインはほとんど変化しないが、ドレイン側のゲインは約-15から-1まで大きく変動している。これも、トランジスタの反転増幅回路と同じ傾向である。

7. 負荷抵抗が固定値であればドレイン側もソース側もとくにソースホロワ等のバッファは必要ないが、負荷抵抗の変動が予想される場合にはバッファを考慮すべきである。

回路データは次の通り。

```
B? /dpart
left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 2] parts[ r1 ] value[ 1e+006 j 0 ]
left[ 0] right[ 3] parts[ rs ] value[ 1e+004 j 0 ]
left[ 0] right[ 5] parts[ e2 ] value[ 24 j 0 ]
left[ 0] right[ 6] parts[ rld ] value[ 1e+006 j 0 ]
left[ 0] right[ 7] parts[ rls ] value[ 1e+006 j 0 ]
left[ 1] right[ 2] parts[ cc ] value[ 0.001 j 0 ]
left[ 2] right[ 9] parts[ r1b1 ] value[ 0.1 j 0 ]
left[ 3] right[ 7] parts[ cc ] value[ 0.001 j 0 ]
left[ 3] right[ 10] parts[ r1b1 ] value[ 0.1 j 0 ]
left[ 4] right[ 5] parts[ rd ] value[ 1e+004 j 0 ]
left[ 4] right[ 6] parts[ cc ] value[ 0.001 j 0 ]
left[ 4] right[ 11] parts[ r1b1 ] value[ 0.1 j 0 ]
left[ 8] right[ 9] parts[ r2b1 ] value[ 1e+010 j 0 ]
left[ 8] right[ 10] parts[ e1b1 ] value[ 2 j 0 ]
left[ 10] right[ 11] parts[ q1b1 ] value[ 0.004 j 0 ]
@ master[r2b1 ] left[ 8] right[ 9]
left[ 10] right[ 11] parts[ r3b1 ] value[ 1e+005 j 0 ]
最大の素子番号 = 17
最大のノード番号 = 11
```

```
B? まず、ノード3と4の動作点を確認します。
/point
f[0 ]
x3 [ 2 j 0 ] abs[ 2] arg[ 0]
x4 [ 22 j 0 ] abs[ 22] arg[ 0]
B? ノード3、4の動作点は電源 e2 及びグランドから 2V しか余裕がありません
従って、使用可能な出力電圧の振幅は 2Vp-p 程度となります。
もっと大振幅の出力まで、使用する場合には、
R1 に正のバイアス電圧を加える必要があります。
```

サンプルバッチファイルの回路図と説明

```

B? ノード4の f=1000Hz におけるゲインを求め、
  続けて、出力インピーダンスを求めます。
/range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x3 [ 0.9706 j 1.544435e-007] abs[ 0.9706] arg[9.116734e-006]
x4 [ -0.9706 j -1.544435e-007] abs[ 0.9706] arg[ -180]
B? a=1/abs(x4)
  1.0303 j 0

B? f=1k
  1,000 j 0

```

B? RLd を 1K から 1M まで変化させて出力インピーダンスを求めます。

```

B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? rld
ac rld 最低値 ? 1k
ac rld 最高値 ? 1m
ac rld ステップ ? 1

```

```

rld[6310 ] f[1,000 ]
x3 [ 1.0015 j 1.740584e-007] abs[ 1.0015] arg[9.958068e-006]
x4 [ -0.3913 j 5.983376e-006] abs[ 0.3913] arg[ 179.9991]
rld[1e+004 ] f[1,000 ]
x3 [ 1.0012 j 1.690814e-007] abs[ 1.0012] arg[9.676016e-006]
x4 [ -0.5056 j 3.937318e-006] abs[ 0.5056] arg[ 179.9996]

```

B? 出力インピーダンスは約 9.8K です。
 出力インピーダンスは約 9.8K です。
 次は、ノード3の出力インピーダンスを求めます。

サンプルバッチファイルの回路図と説明

F E T Eのパラメータ e 1 b 1 を現在の 2 V から 6 V に変更して確認します。

```
? e1b1=6
      6 j 0

? /point
f[1,000
x3 [ 6.8521 j 1.541294e-007] abs[ 6.8521] arg[1.288804e-006]
x4 [ 16.9103 j -1.913255e-007] abs[ 16.9103] arg[-6.482519e-007]
```

ゲインを確認します。

```
? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0
f[1,000
x3 [ 0.9706 j 1.544435e-007] abs[ 0.9706] arg[9.116734e-006]
x4 [ -0.9706 j -1.544435e-007] abs[ 0.9706] arg[-180]
```

ソース側もドレイン側もゲインは 0.9706 で、ドレイン側は位相が反転しています。

出力が 1 になるように e1 の値を変更します。

```
? a=1/0.9706
      1.0303 j 0

? e1=a
      1.0303 j 0
```

ドレイン側の負荷抵抗を変えながら、x4=-0.5 となる値を探します。

```
rld[9700 ] f[1,000
x3 [ 1.0013 j 1.693898e-007] abs[ 1.0013] arg[9.693196e-006]
x4 [ -0.4979 j 4.062153e-006] abs[ 0.4979] arg[ 179.9995]
rld[9800 ] f[1,000
x3 [ 1.0012 j 1.692871e-007] abs[ 1.0012] arg[9.687383e-006]
x4 [ -0.5005 j 4.019952e-006] abs[ 0.5005] arg[ 179.9995]
```

B? 出力インピーダンスは約 9.8K です。

サンプルバッチファイルの回路図と説明

次は、ノード3の出力インピーダンスを求めます。
キー入力待ち？

```
B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
```

```
B? /range
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x3 [ 0.9706 j 1.544435e-007] abs[ 0.9706] arg[9.116734e-006]
x4 [ -0.9706 j -1.544435e-007] abs[ 0.9706] arg[-180]
B? a=1/abs(x3)
1.0303 j 0
```

ソース側も負荷抵抗を変えながら、 $x_3=0.5$ となる値を探します。

```
B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? rls
ac rls 最低値 ? 10
ac rls 最高値 ? 10k
ac rls ステップ ? 1
```

```
rls[251.2 ] f[1,000 ]
x3 [ 0.4850 j -1.582220e-004] abs[ 0.4850] arg[-0.0187]
x4 [ -19.5977 j -0.0057 ] abs[ 19.5977] arg[-179.9833]
rls[398.1 ] f[1,000 ]
x3 [ 0.5989 j -9.598095e-005] abs[ 0.5989] arg[-0.0092]
x4 [ -15.4865 j -0.0035 ] abs[ 15.4865] arg[-179.9872]
```

RLs=250 から 400 の範囲を調べます。

サンプルバッチファイルの回路図と説明

```

r1s[260      ] f[1,000      ]
x3 [ 0.4936 j -1.529744e-004] abs[ 0.4936] arg[ -0.0178]
x4 [-19.2871 j -0.0055      ] abs[ 19.2871] arg[ -179.9836]
r1s[270      ] f[1,000      ]
x3 [ 0.5031 j -1.473242e-004] abs[ 0.5031] arg[ -0.0168]
x4 [-18.9463 j -0.0053      ] abs[ 18.9463] arg[ -179.9839]

```

B? ノード3の出カインピーダンスは 270 オームです。

FET による位相反転回路では、
 ドレインとソースにおけるゲインが等しくなる。
 これはトランジスタと違って、ゲート電流が無いためである。
 ドレイン側の出カインピーダンスはトランジスタ回路と同等であるが、
 ソース側の出カインピーダンスはトランジスタ回路よりも高い
 これは FET の g_m をトランジスタの h_{fe} に換算すると
 約 10 分の 1 程度しかないためである。

以上で、FET 位相反転回路の解析を終了します。

追加で、ノード1とノード3の間に抵抗 $rg=0.1$ を追加して、 $e1$ の変化に対するノード電圧
 の変化を確認する。

```

? /padd
left[ 0] right[ 1] parts[ e1      ] value[ 1 j 0      ]
left[ 0] right[ 2] parts[ r1      ] value[ 1e+006 j 0      ]
left[ 0] right[ 3] parts[ rs      ] value[ 1e+004 j 0      ]
left[ 0] right[ 5] parts[ e2      ] value[ 24 j 0      ]
left[ 0] right[ 6] parts[ r1d     ] value[ 1e+006 j 0      ]
left[ 0] right[ 7] parts[ r1s     ] value[ 1e+006 j 0      ]
left[ 1] right[ 2] parts[ cc      ] value[ 0.001 j 0      ]
left[ 2] right[ 9] parts[ r1b1    ] value[ 0.1 j 0      ]
left[ 3] right[ 7] parts[ cc      ] value[ 0.001 j 0      ]
left[ 3] right[ 10] parts[ r1b1   ] value[ 0.1 j 0      ]
left[ 4] right[ 5] parts[ rd      ] value[ 1e+004 j 0      ]
left[ 4] right[ 6] parts[ cc      ] value[ 0.001 j 0      ]
left[ 4] right[ 11] parts[ r1b1    ] value[ 0.1 j 0      ]
left[ 8] right[ 9] parts[ r2b1    ] value[ 1e+010 j 0      ]
left[ 8] right[ 10] parts[ e1b1    ] value[ 2 j 0      ]
left[ 10] right[ 11] parts[ q1b1   ] value[ 0.004 j 0      ]
@ master[r2b1      ] left[ 8] right[ 9]
left[ 10] right[ 11] parts[ r3b1    ] value[ 1e+005 j 0      ]
最大の素子番号 = 17
最大のノード番号 = 11

left[ 0] right ? /
left[ 1] right ? 2
素子名は ? rg
値は ? 0.1

left[ 1] right ? *
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 5] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[e1b1] 値[2 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける表示ノード設定状況
3 4
表示ノード番号 (0 なら全て、-1 ならこのまま) ? -1

```

(6) バッチファイル名 s10. sb で扱う回路図

サンプルバッチファイルの回路図と説明

e1 を変化させて、出力を確認する

```
? /para
データファイル名は ? s10-e1
値を変化させる素子名は ? e1
para e1 最低値 ? 0
para e1 最高値 ? 24
para e1 ステップ ? 1
```

2 25

0 e1=0V を表す

3 1.99998 ノード x3=1.99998V

4 22 ノード x4=22V 動作している

0 0

1

3 2.97084

4 21.0292

0 0

2

3 3.94171

4 20.0583

0 0

3

3 4.91257

4 19.0874

0 0

4

3 5.88344

4 18.1166

0 0

5

3 6.8543

4 17.1457

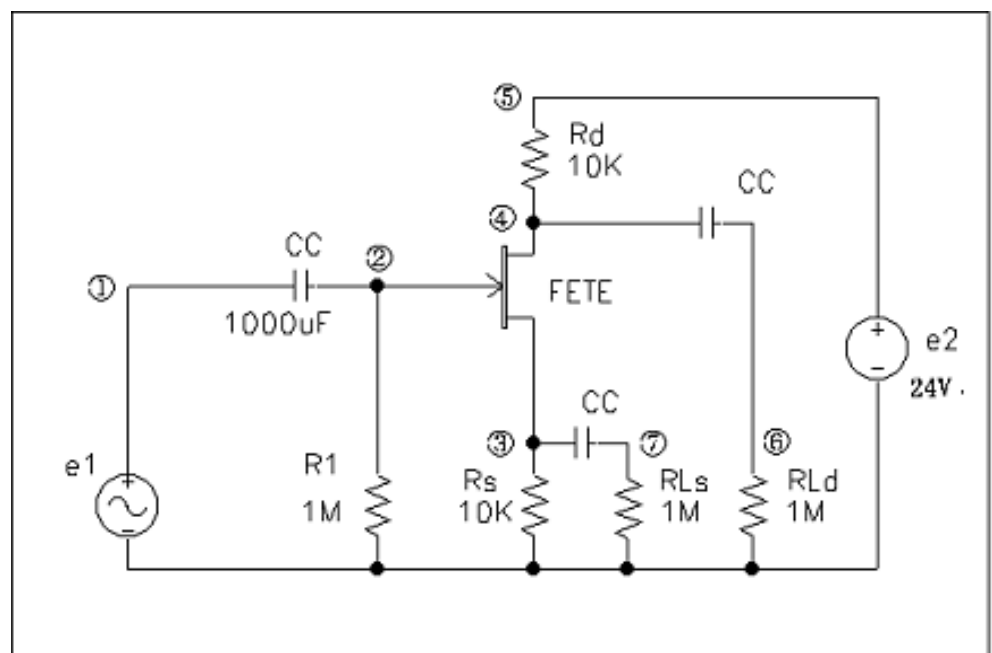
0 0

6

3 7.82517

4 16.1748

0 0



(6) バッチファイル名 s10. sb で扱う回路図

サンプルバッチファイルの回路図と説明

```

7
3 8.79603
4 15.204
0 0
8
3 9.76689
4 14.2331
0 0
9          e1=9V
3 10.7378   ノード x3=10.7378V
4 13.2622   ノード x4=13.2622V
0 0
9.5        e1=9.5V          このデータだけ単独で計算して追加した
3 11.2232   ノード x3=11.2232V
4 12.7768   ノード x4=12.7768V   x4>x3 で差も十分に大きいので動作している
0 0
10         e1=10V
3 11.7086   ノード x3=11.7086V
4 12.2914   ノード x4=12.2914V   x4>x3 だが、差が少ないので動作していない
0 0
11         e1=11V
3 12.6795   ノード x3=12.6795V
4 11.3205   ノード x4=11.3205V   x4<x3 なので、fet が動作していない
0 0

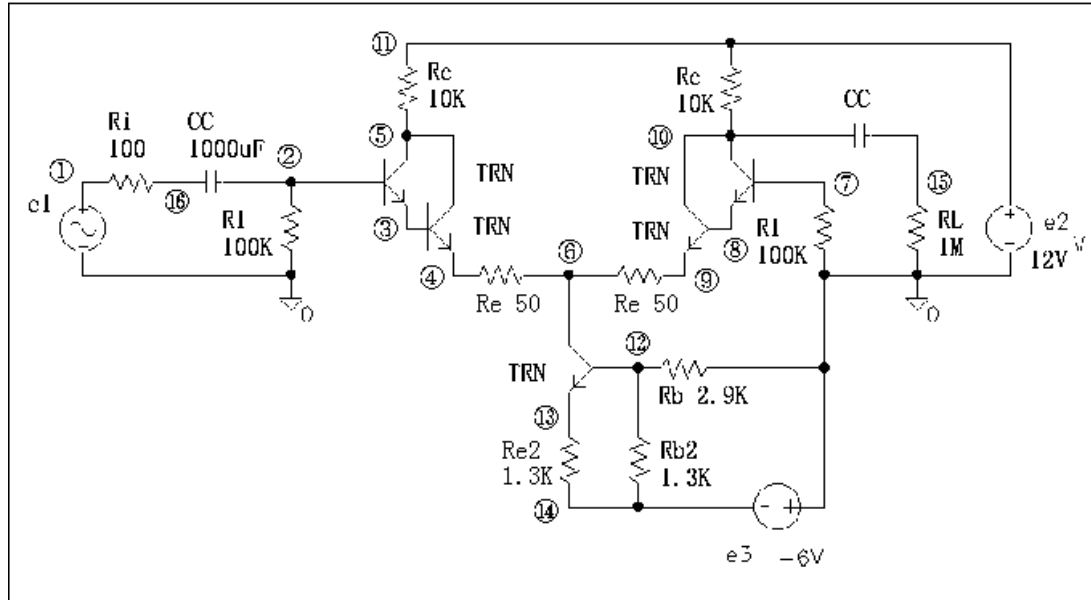
```

以上のデータから、(rg が追加されていない) sb10 の回路は、
 入力信号の振幅が 9.5Vp-p の時に、
 ノード 3 の電圧は、2V から 11.2V まで 9.2Vp-p の出力が得られ、
 ノード 4 の電圧は、22V から 12.8V まで 8.8Vp-p の出力が得られる。
 ドレイン側の方がソース側より少しだけゲインが低い。

反転増幅回路では、ソースの動作点を電源電圧の 1/3 程度に設定すれば、ドレインの動作点
 が電源電圧から 1/3 だけ下がった電圧になって、最大振幅の出力が得られる。

(7) バッチファイル名 s11. sb で扱う回路図

トランジスタのダーリントン接続により入力インピーダンスを高めた差動増幅回路



設計の指針

1. ノード5とノード10の直流動作点を電源電圧e2の半分に設定する。

回路図の下側にあるトランジスタ回路は、一般的には定電流回路といわれており、
Re2によって定電流の値が決定されている。この値を変えればノード5とノード
10の電圧を変更する事が出来る。

/paraを使用して、Re2を1Kから1.5Kまで変化させて、ノード5とノード10の
電圧が6VになるRe2の値を求める。

Re=1030で希望の電圧になることが分かる。

2. Rcに流れる電流を計算する。

(Rcの両端の電圧) / Rcで求められるから、x5を求める。

f=0

/point

(e2 - x5) / Rc = 0.6mAが得られる。

3. 入力信号の周波数が1000Hzの時のゲインを調べる。

ゲインは約90倍である。ノード5は位相反転出力、ノード10は非反転である。

4. 入力インピーダンスを求める。

Zi=95Kが得られる。これは、R1の値とほとんど同じである。

R1の右側のインピーダンスをZiiとしてこの値を計算すると、

$95K = 100K * Zii / (100K + Zii)$ より、

(7) バッチファイル名 s11. sb で扱う回路図

サンプルバッチファイルの回路図と説明

$Z_{ii} = 1.9M$ が得られる。従って、 R_1 を $1M$ に増加すれば入力インピーダンスを $650K$ に増加することが可能であることがわかる。

$$1M * 1.9M / (1M + 1.9M) = 650K$$

しかし、この場合にはトランジスタのベース電流が減少するためにゲインが約 50 に下がってしまう。

回路を入力する操作は次の通りです。

```
B? /ipart
left[ 0] right ? 1
素子名は ? e1
値は ? 1

left[ 0] right ? 2
素子名は ? r1
値は ? 100k

left[ 0] right ? 12
素子名は ? rb
値は ? 2.9k

left[ 0] right ? 14
素子名は ? e3
値は ? -6

left[ 0] right ? 7
素子名は ? r1
素子名 r1 は既に使用されています。
同じ素子名を使用しますか? (y/n) y

left[ 0] right ? 11
素子名は ? e2
値は ? 12

left[ 0] right ? 15
素子名は ? rl
値は ? 1m

left[ 0] right ? /

left[ 1] right ? 16
素子名は ? ri
値は ? 100

left[ 1] right ? /

left[ 2] right ? 16
素子名は ? cc
値は ? 1000u
```

サンプルバッチファイルの回路図と説明

```
left[ 2] right ? 5
素子名は ? b1
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn:2,3,5

left[ 2] right ? /

left[ 3] right ? 5
素子名は ? b2
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn:3,4,5

left[ 3] right ? /

left[ 4] right ? 6
素子名は ? re
値は ? 50

left[ 4] right ? /

left[ 5] right ? 11
素子名は ? rc
値は ? 10k

left[ 5] right ? /

left[ 6] right ? 9
素子名は ? re
素子名 re は既に使用されています。
同じ素子名を使用しますか？ (y/n) y

left[ 6] right ? /

left[ 7] right ? 10
素子名は ? b3
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn:7,8,10

left[ 7] right ? /

left[ 8] right ? 10
素子名は ? b4
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn:8,9,10

left[ 8] right ? /

left[ 9] right ? /

left[ 10] right ? 15
素子名は ? cc
素子名 cc は既に使用されています。
同じ素子名を使用しますか？ (y/n) y

left[ 10] right ? 11
素子名は ? rc
素子名 rc は既に使用されています。
同じ素子名を使用しますか？ (y/n) y
```

(7) バッチファイル名 s11. sb で扱う回路図

```

left[ 10] right ? /
left[ 11] right ? /
left[ 12] right ? 6
素子名は ? b5
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn:12,13,6

left[ 12] right ? 14
素子名は ? rb2
値は ? 1.3k

left[ 12] right ? /
left[ 13] right ? 14
素子名は ? re2
値は ? 1.3k

left[ 13] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[11] j[ 0] 部品名[e2] 値[12 j 0]
i[14] j[ 0] 部品名[e3] 値[-6 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 5
ノード番号 (0 なら終了) ? 10
ノード番号 (0 なら終了) ?
未解凍のブロックがあります。 解凍しますか (y/n) y
bname b1 sfnc trn:2,3,5

func trn:2,3,5 name trn.cir blk b1
@ func trn:1,2,3
bname b2 sfnc trn:3,4,5

func trn:3,4,5 name trn.cir blk b2
@ func trn:1,2,3
bname b3 sfnc trn:7,8,10

func trn:7,8,10 name trn.cir blk b3
@ func trn:1,2,3
bname b4 sfnc trn:8,9,10

func trn:8,9,10 name trn.cir blk b4
@ func trn:1,2,3
bname b5 sfnc trn:12,13,6

func trn:12,13,6 name trn.cir blk b5
@ func trn:1,2,3

```


サンプルバッチファイルの回路図と説明

独立電圧源素子の接続ノード

```

i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[11] j[ 0] 部品名[e2] 値[12 j 0]
i[14] j[ 0] 部品名[e3] 値[-6 j 0]
i[20] j[18] 部品名[ebb1] 値[0.6 j 0]
i[24] j[22] 部品名[ebb2] 値[0.6 j 0]
i[28] j[26] 部品名[ebb3] 値[0.6 j 0]
i[32] j[30] 部品名[ebb4] 値[0.6 j 0]
i[36] j[34] 部品名[ebb5] 値[0.6 j 0]

```

独立電圧源素子の個数 8

独立電流源素子の個数 0

シミュレーションにおける 表示ノード設定状況

5 10

表示ノード番号 (0 なら全て、-1 ならこのまま) ? -1

B? /dpart

```

left[ 0] right[ 1] parts[ e1 ] value[ 1 j 0 ]
left[ 0] right[ 2] parts[ r1 ] value[ 1e+005 j 0 ]
left[ 0] right[ 7] parts[ r1 ] value[ 1e+005 j 0 ]
left[ 0] right[11] parts[ e2 ] value[ 12 j 0 ]
left[ 0] right[12] parts[ rb ] value[ 2900 j 0 ]
left[ 0] right[14] parts[ e3 ] value[ -6 j 0 ]
left[ 0] right[15] parts[ r1 ] value[ 1e+006 j 0 ]
left[ 1] right[16] parts[ ri ] value[ 100 j 0 ]
left[ 2] right[16] parts[ cc ] value[ 0.001 j 0 ]
left[ 2] right[20] parts[ rlb1 ] value[ 0.1 j 0 ]
left[ 3] right[17] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 3] right[24] parts[ rlb2 ] value[ 0.1 j 0 ]
left[ 4] right[ 6] parts[ re ] value[ 50 j 0 ]
left[ 4] right[21] parts[ rbb2 ] value[ 0.1 j 0 ]
left[ 5] right[11] parts[ rc ] value[ 1e+004 j 0 ]
left[ 5] right[19] parts[ rbb1 ] value[ 0.1 j 0 ]
left[ 5] right[23] parts[ rbb2 ] value[ 0.1 j 0 ]
left[ 6] right[ 9] parts[ re ] value[ 50 j 0 ]
left[ 6] right[35] parts[ rbb5 ] value[ 0.1 j 0 ]
left[ 7] right[28] parts[ rlb3 ] value[ 0.1 j 0 ]
left[ 8] right[25] parts[ rbb3 ] value[ 0.1 j 0 ]
left[ 8] right[32] parts[ rlb4 ] value[ 0.1 j 0 ]
left[ 9] right[29] parts[ rbb4 ] value[ 0.1 j 0 ]
left[10] right[11] parts[ rc ] value[ 1e+004 j 0 ]
left[10] right[15] parts[ cc ] value[ 0.001 j 0 ]
left[10] right[27] parts[ rbb3 ] value[ 0.1 j 0 ]
left[10] right[31] parts[ rbb4 ] value[ 0.1 j 0 ]
left[12] right[14] parts[ rb2 ] value[ 1300 j 0 ]
left[12] right[36] parts[ rlb5 ] value[ 0.1 j 0 ]
left[13] right[14] parts[ re2 ] value[ 1300 j 0 ]
left[13] right[33] parts[ rbb5 ] value[ 0.1 j 0 ]
left[17] right[18] parts[ reb1 ] value[ 26 j 0 ]
left[17] right[19] parts[ kb1 ] value[ 100 j 0 ]
@ master[reb1] left[17] right[18]
left[18] right[20] parts[ ebb1 ] value[ 0.6 j 0 ]
left[21] right[22] parts[ reb2 ] value[ 26 j 0 ]
left[21] right[23] parts[ kb2 ] value[ 100 j 0 ]
@ master[reb2] left[21] right[22]
left[22] right[24] parts[ ebb2 ] value[ 0.6 j 0 ]
left[25] right[26] parts[ reb3 ] value[ 26 j 0 ]
left[25] right[27] parts[ kb3 ] value[ 100 j 0 ]
@ master[reb3] left[25] right[26]
left[26] right[28] parts[ ebb3 ] value[ 0.6 j 0 ]
left[29] right[30] parts[ reb4 ] value[ 26 j 0 ]
left[29] right[31] parts[ kb4 ] value[ 100 j 0 ]
@ master[reb4] left[29] right[30]
left[30] right[32] parts[ ebb4 ] value[ 0.6 j 0 ]
left[33] right[34] parts[ reb5 ] value[ 26 j 0 ]
left[33] right[35] parts[ kb5 ] value[ 100 j 0 ]
@ master[reb5] left[33] right[34]
left[34] right[36] parts[ ebb5 ] value[ 0.6 j 0 ]
最大の素子番号 = 46
最大のノード番号 = 36

```

(7) バッチファイル名 s 1 1. s b で扱う回路図

```

B? ノード5、10の直流動作点を確認します。
キー入力待ち？

B? /point
f[0
x5 [ 7.2470 j 0 ] abs[ 7.2470] arg[ 0]
x10 [ 7.2470 j 0 ] abs[ 7.2470] arg[ 0]
B? Re2 を変更して、動作点を 6V に設定します

```

```

B? /para
データファイル名は ? test
値を変化させる素子名は ? re2
para re2 最低値 ? 1k
para re2 最高値 ? 1.5k
para re2 ステップ ? 10
re2[1000 ] f[0
x5 [ 5.8341 j 0 ] abs[ 5.8341] arg[ 0]
x10 [ 5.8341 j 0 ] abs[ 5.8341] arg[ 0]
re2[1010 ] f[0
x5 [ 5.8946 j 0 ] abs[ 5.8946] arg[ 0]
x10 [ 5.8946 j 0 ] abs[ 5.8946] arg[ 0]
re2[1020 ] f[0
x5 [ 5.9540 j 0 ] abs[ 5.9540] arg[ 0]
x10 [ 5.9540 j 0 ] abs[ 5.9540] arg[ 0]
re2[1030 ] f[0
x5 [ 6.0121 j 0 ] abs[ 6.0121] arg[ 0]
x10 [ 6.0121 j 0 ] abs[ 6.0121] arg[ 0]
re2[1040 ] f[0
x5 [ 6.0692 j 0 ] abs[ 6.0692] arg[ 0]
x10 [ 6.0692 j 0 ] abs[ 6.0692] arg[ 0]

```

```

B? Re=1030 とします。
キー入力待ち？

B? re2=1030
1,030 j 0

```

```

B?
今決定した回路定数における、Rc に流れる電流を計算します。
/point
f[0
x5 [ 6.0121 j 0 ] abs[ 6.0121] arg[ 0]
x10 [ 6.0121 j 0 ] abs[ 6.0121] arg[ 0]
B? Rc に流れる電流 = (Rc の両端の電圧)/Rc で求められます。
(e2-x5)/rc
5.987871e-004 j 0
B? 電流は約 0.6mA です。

```

サンプルバッチファイルの回路図と説明

次に、1000Hz におけるゲインを求めます。

```
B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x5 [ -90.3685 j -1.564120e-004] abs[ 90.3685] arg[ -179.9999]
x10 [ 89.4737 j 1.547225e-004] abs[ 89.4737] arg[9.907875e-005]
B? ゲインは約 90 倍です。ノード 5 は位相反転、ノード 10 は非反転です。
```

入力インピーダンスを求めます。

最初に $R_i=0.001$ として、/range で出力を求め、出力が 1 となる E_1 を求めてから/para で R_i を変化させて出力が 0.5 となる値が入力インピーダンスになります。

```
B? Ri=0.001 とします
ri=0.001
0.0010 j 0

B? Ri=0.001 におけるノード 10 の出力電圧が 1 となるような e1 を求めて、
入力インピーダンスを求めます。
/range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x5 [ -90.4669 j -1.567535e-004] abs[ 90.4669] arg[ -179.9999]
x10 [ 89.5711 j 1.550597e-004] abs[ 89.5711] arg[9.918670e-005]
B? a=e1/abs(x10)
0.0112 j 0
```

R_i を 1k から 1M まで変化させて、 $x_{10}=0.5$ となる R_i の値を求めます。

```
B? f=1k
1,000 j 0

B? /ac
データファイル名は ? test
ac 入力信号源名は ? e1
値は ? a

値を変化させる素子名は ? ri
ac ri 最低値 ? 1k
ac ri 最高値 ? 1m
ac ri ステップ ? 1
```

サンプルバッチファイルの回路図と説明

```

ri[4.642e+004] f[1,000]
x5 [-0.6710 j -7.723226e-007] abs[ 0.6710] arg[-179.9999]
x10 [ 0.6643 j 7.636302e-007] abs[ 0.6643] arg[6.586223e-005]
ri[1e+005] f[1,000]
x5 [-0.4836 j -4.011454e-007] abs[ 0.4836] arg[-180]
x10 [ 0.4788 j 3.964200e-007] abs[ 0.4788] arg[4.744088e-005]

```

1 番のバッチファイル s11.sb を終了します。

追加で、ri=46k から 100k の間をさらに調べます。

```

ri[9e+004] f[1,000]
x5 [-0.5101 j -4.464808e-007] abs[ 0.5101] arg[-179.9999]
x10 [ 0.5051 j 4.412651e-007] abs[ 0.5051] arg[5.005514e-005]
ri[9.1e+004] f[1,000]
x5 [-0.5074 j -4.415997e-007] abs[ 0.5074] arg[-180]
x10 [ 0.5023 j 4.364367e-007] abs[ 0.5023] arg[4.977967e-005]
ri[9.2e+004] f[1,000]
x5 [-0.5046 j -4.368325e-007] abs[ 0.5046] arg[-180]
x10 [ 0.4996 j 4.317210e-007] abs[ 0.4996] arg[4.951110e-005]
ri[9.3e+004] f[1,000]
x5 [-0.5019 j -4.321225e-007] abs[ 0.5019] arg[-180]
x10 [ 0.4969 j 4.270619e-007] abs[ 0.4969] arg[4.924317e-005]
ri[9.4e+004] f[1,000]
x5 [-0.4992 j -4.274755e-007] abs[ 0.4992] arg[-180]
x10 [ 0.4942 j 4.224651e-007] abs[ 0.4942] arg[4.897665e-005]
ri[9.5e+004] f[1,000]
x5 [-0.4965 j -4.229187e-007] abs[ 0.4965] arg[-180]
x10 [ 0.4916 j 4.179576e-007] abs[ 0.4916] arg[4.871480e-005]
ri[9.6e+004] f[1,000]
x5 [-0.4939 j -4.184387e-007] abs[ 0.4939] arg[-180]
x10 [ 0.4890 j 4.135261e-007] abs[ 0.4890] arg[4.845624e-005]
ri[9.7e+004] f[1,000]
x5 [-0.4912 j -4.140229e-007] abs[ 0.4912] arg[-180]
x10 [ 0.4864 j 4.091580e-007] abs[ 0.4864] arg[4.819962e-005]
ri[9.8e+004] f[1,000]
x5 [-0.4886 j -4.096591e-007] abs[ 0.4886] arg[-180]
x10 [ 0.4838 j 4.048415e-007] abs[ 0.4838] arg[4.794366e-005]
ri[9.9e+004] f[1,000]
x5 [-0.4861 j -4.053360e-007] abs[ 0.4861] arg[-180]
x10 [ 0.4813 j 4.005652e-007] abs[ 0.4813] arg[4.768709e-005]
ri[1e+005] f[1,000]
x5 [-0.4836 j -4.011454e-007] abs[ 0.4836] arg[-180]
x10 [ 0.4788 j 3.964200e-007] abs[ 0.4788] arg[4.744088e-005]

```

ri=91k から 92k の間をさらに調べます

```

ri[9.18e+004] f[1,000]
x5 [-0.5051 j -4.377916e-007] abs[ 0.5051] arg[-180]
x10 [ 0.5001 j 4.326698e-007] abs[ 0.5001] arg[4.956593e-005]
ri[9.19e+004] f[1,000]
x5 [-0.5049 j -4.373101e-007] abs[ 0.5049] arg[-180]
x10 [ 0.4999 j 4.321935e-007] abs[ 0.4999] arg[4.953832e-005]

```

(7) バッチファイル名 s11.sb で扱う回路図

サンプルバッチファイルの回路図と説明

$R_i = 91.9K$ で、 $X_{10} = 0.5$ になるので、入力インピーダンスは $91.9K$ です。

ノード 5 とノード 10 の電圧が変化する範囲を確認します。

ノード 2 とノード 7 の電圧も同時に確認します。

まず、 $f=0$ に設定して、`/disp` で表示するノードを入力してから `/point` で確認します。

```
? f=0
      0 j 0

? /disp

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[11] j[ 0] 部品名[e2] 値[12 j 0]
i[14] j[ 0] 部品名[e3] 値[-6 j 0]
i[20] j[18] 部品名[ebb1] 値[0.6 j 0]
i[24] j[22] 部品名[ebb2] 値[0.6 j 0]
i[28] j[26] 部品名[ebb3] 値[0.6 j 0]
i[32] j[30] 部品名[ebb4] 値[0.6 j 0]
i[36] j[34] 部品名[ebb5] 値[0.6 j 0]
独立電圧源素子の個数 8
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
5 10

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 2
ノード番号 (0 なら終了) ? 7
ノード番号 (0 なら終了) ? 5
ノード番号 (0 なら終了) ? 10
ノード番号 (0 なら終了) ?
? /point
f[0
x2 [ -0.0059 j 0 ] abs[ 0.0059] arg[ 180]
x5 [ 6.0121 j 0 ] abs[ 6.0121] arg[ 0]
x7 [ -0.0059 j 0 ] abs[ 0.0059] arg[ 180]
x10 [ 6.0121 j 0 ] abs[ 6.0121] arg[ 0]
```

信号入力用のノードであるノード 2 と 7 は同じ電圧です。

また、信号出力用のノードであるノード 5 と 10 も同じ電圧です。

電源電圧 $e_2=12V$ なので、ノード 1 に正弦波の信号が入力されると、ノード 5 の電圧は $0V$ から $12V$ までの振幅 $12V_{p-p}$ 以内の範囲で変化するはずです。

また、この回路のゲインは約 90 と求められているので、入力の振幅は $12/90=0.1333V$ 以内で最大振幅の出力が得られると考えられます。

従って、入力 e_1 の範囲は $-0.07 \sim 0.07V$ の範囲に対する出力を確認します。

サンプルバッチファイルの回路図と説明

e1 を直接ノード 2 に接続するために、/padd でノード 1 とノード 2 の間に r2=0.1 を追加します。表示するノードは 5 と 10 だけに変更します。

```
left[ 0] right ? /
left[ 1] right ? 2
素子名は ? r2
値は ? 0.1
left[ 1] right ? *
```

```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[11] j[ 0] 部品名[e2] 値[12 j 0]
i[14] j[ 0] 部品名[e3] 値[-6 j 0]
i[20] j[18] 部品名[ebb1] 値[0.6 j 0]
i[24] j[22] 部品名[ebb2] 値[0.6 j 0]
i[28] j[26] 部品名[ebb3] 値[0.6 j 0]
i[32] j[30] 部品名[ebb4] 値[0.6 j 0]
i[36] j[34] 部品名[ebb5] 値[0.6 j 0]
独立電圧源素子の個数 8
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
2 5 7 10

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 5
ノード番号 (0 なら終了) ? 10
ノード番号 (0 なら終了) ?
```

```
? /para
データファイル名は ? s11-e1
値を変化させる素子名は ? e1
para e1 最低値 ? -0.07
para e1 最高値 ? 0.07
para e1 ステップ ? 0.001
```

2 141 表示ノード数 2 出力ポイント数 141

-0.07 e1=-0.07V

5 11.8137 x5=11.8137V

10 0.210538 x10=0.210538V 正常に動作している

0 0

-0.069 e1=-0.069V

5 11.7233 x5=11.7233V

10 0.301005 x10=0.301005V

0 0

(7) バッチファイル名 s 1 1. s b で扱う回路図

サンプルバッチファイルの回路図と説明

・
・
・

0.057 e1=0.057V

5 0.324442 x5=0.324442V

10 11.6998 x10=11.6998 V

0 0

0.058 e1=0.058V

5 0.233976 x5=0.233976V

10 11.7903 x10=11.7903 V

0 0

0.059 e1=0.059V

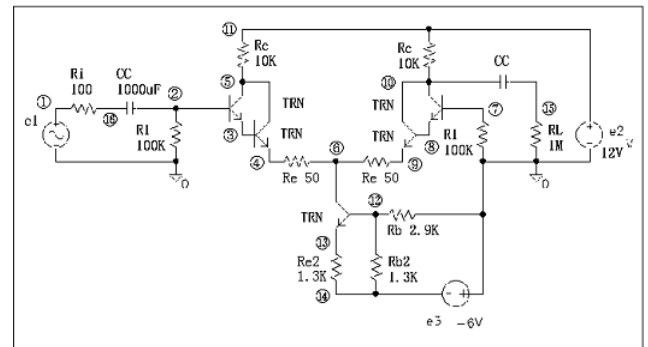
5 0.143509 x5=0.143509V

10 11.8807 x10=11.8807 V

0 0

正常に動作している

余裕がない



以上の電圧データより

e1=-0.07～0.058 V の変化 (0.128Vp-p) に対して、

x5=11.8137～0.233976V の変化 (11.58Vp-p)、ゲインは 90.48 となる。

x10=0.210538～11.7903 V の変化 (-11.58Vp-p)、ゲインは-90.48 となる。

e1 の中心値は $-0.07+0.128/2 = -0.006$ V であることが確認された。

ノード 5 とノード 10 の動作点を決定後に確認した値 -0.0059V に等しいと言える。

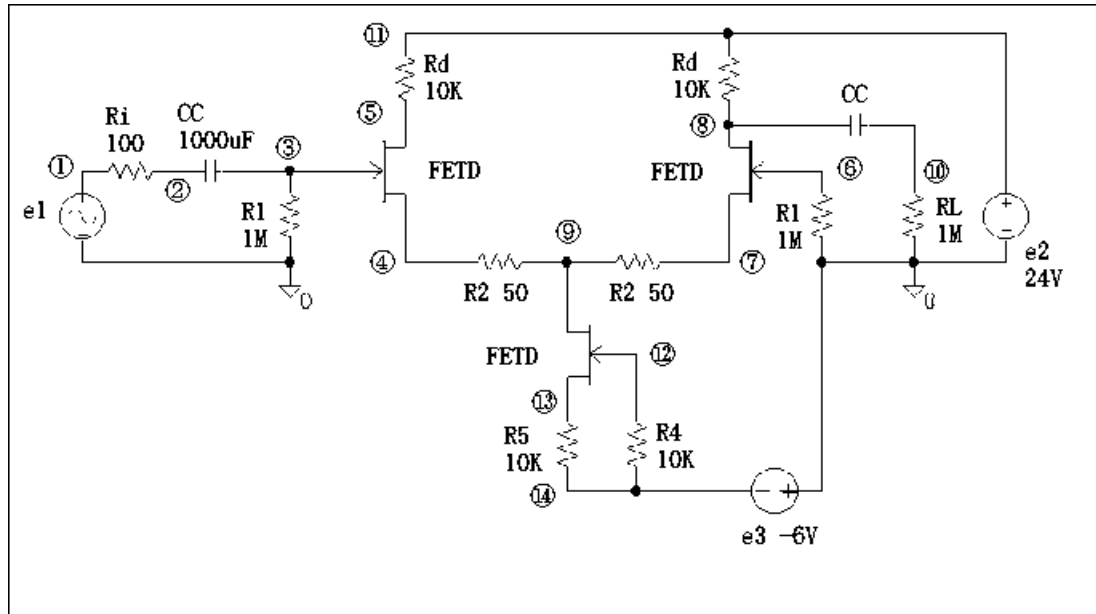
```
? /point
f[0]
x2 [ -0.0059 j 0 ] abs[ 0.0059] arg[ 180]
x5 [ 6.0121 j 0 ] abs[ 6.0121] arg[ 0]
x7 [ -0.0059 j 0 ] abs[ 0.0059] arg[ 180]
x10 [ 6.0121 j 0 ] abs[ 6.0121] arg[ 0]
```

従って、s11 の回路はノード 5 とノード 10 の電圧が -0.0059V を中心として動作しており、振幅 0.128Vp-p の交流入力信号 e1 がキャパシタンス CC を通じて加えられると、最大振幅 (11.58Vp-p) の出力が得られることが分かる。

(7) バッチファイル名 s 1 1. s b で扱う回路図

(8) バッチファイル名 `ss1.sb` で扱う回路図

N型デプリーションタイプのFETを使用した差動増幅回路



設計の指針

1. ノード5とノード8の直流動作点を電源電圧 $e2$ の半分に設定する。

回路図の下側にあるFET回路は、一般的には定電流回路といわれており、
 $R5$ によって定電流の値が決定されている。この値を変えればノード5とノード8
 の電圧を変更する事が出来る。

`/para`を使用して、 $R5$ を100オームから10Kまで変化させて、ノード5と
 ノード8の電圧が12Vになる $R5$ の値を求める。

$R5 = 596$ オームで希望の電圧になることが分かる。

2. 入力信号の周波数が1000Hzの時のゲインを調べる。

ゲインは約15倍である。ノード5は位相反転出力、ノード8は非反転である。
 トランジスタのダーリントン接続の回路に比べると、ゲインはかなり小さい。

3. 入力インピーダンスを求める。

$Z_i = 1M$ が得られる。これは、 $R1$ の値と同じである。

$R1 = 10M$ に増加すれば入力インピーダンスを10Mに増加することが可能であること
 がわかる。この場合にはトランジスタの回路とは異なり、ゲインに影響はない。

4. ノード5側にもノード8側と同じように負荷抵抗を接続すると、ゲインも等しくなる。

(8) バッチファイル名 `ss1.sb` で扱う回路図

サンプルバッチファイルの回路図と説明

回路の入力は次のように行います。

```

B? /ipart

left[ 0] right ? 1
素子名は ? e1
値は ? 1

left[ 0] right ? 3
素子名は ? r1
値は ? 1m

left[ 0] right ? 14
素子名は ? e3
値は ? -12

left[ 0] right ? 6
素子名は ? r1
素子名 r1 は既に使用されています。
同じ素子名を使用しますか? (y/n) y

left[ 0] right ? 10
素子名は ? r1
値は ? 1m

left[ 0] right ? 11
素子名は ? e2
値は ? 24

left[ 0] right ? /

left[ 1] right ? 2
素子名は ? ri
値は ? 100

left[ 1] right ? /

left[ 2] right ? 3
素子名は ? cc
値は ? 1000u

left[ 2] right ? /

left[ 3] right ? 5
素子名は ? b1
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
fetd:3,4,5

left[ 3] right ? /

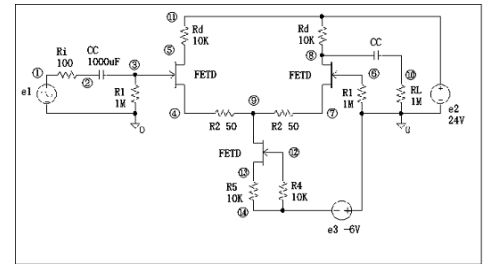
left[ 4] right ? 9
素子名は ? r2
値は ? 50

left[ 4] right ? /

left[ 5] right ? 11
素子名は ? r3
値は ? 10k

left[ 5] right ? /

```



サンプルバッチファイルの回路図と説明

```

left[ 6] right ? 8
素子名は ? b2
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
fetd:6,7,8

left[ 6] right ? /

left[ 7] right ? 9
素子名は ? r2
素子名 r2 は既に使用されています。
同じ素子名を使用しますか? (y/n) y

left[ 7] right ? /

left[ 8] right ? 10
素子名は ? cc
素子名 cc は既に使用されています。
同じ素子名を使用しますか? (y/n) y

left[ 8] right ? 11
素子名は ? r3
素子名 r3 は既に使用されています。
同じ素子名を使用しますか? (y/n) y

left[ 8] right ? /

left[ 9] right ? 12
素子名は ? b3
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
fetd:12,13,9

left[ 9] right ? /

left[ 10] right ? /

left[ 11] right ? /

left[ 12] right ? 14
素子名は ? r4
値は ? 1m

left[ 12] right ? /

left[ 13] right ? 14
素子名は ? r5
値は ? 1k

left[ 13] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[11] j[ 0] 部品名[e2] 値[24 j 0]
i[14] j[ 0] 部品名[e3] 値[-12 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 5
ノード番号 (0 なら終了) ? 8

```

(8) バッチファイル名 s s 1. s b で扱う回路図

サンプルバッチファイルの回路図と説明

```

ノード番号 (0 なら終了) ?
未解凍のブロックがあります。 解凍しますか (y/n)      y
bname b1  sfnc fetd:3,4,5

func fetd:3,4,5 name fetd.cir blk b1
@ func  fetd:1,2,3
bname b2  sfnc fetd:6,7,8

func fetd:6,7,8 name fetd.cir blk b2
@ func  fetd:1,2,3
bname b3  sfnc fetd:12,13,9

func fetd:12,13,9 name fetd.cir blk b3
@ func  fetd:1,2,3

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[11] j[ 0] 部品名[e2] 値[24 j 0]
i[14] j[ 0] 部品名[e3] 値[-12 j 0]
i[17] j[15] 部品名[e1b1] 値[2 j 0]
i[21] j[19] 部品名[e1b2] 値[2 j 0]
i[25] j[23] 部品名[e1b3] 値[2 j 0]
独立電圧源素子の個数 6
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
5 8

表示ノード番号 (0 なら全て、-1 ならこのまま) ? -1

```

```

B? /dpart
left[ 0] right[ 1] parts[ e1      ] value[      1 j 0 ]
left[ 0] right[ 3] parts[ r1      ] value[ 1e+006 j 0 ]
left[ 0] right[ 6] parts[ r1      ] value[ 1e+006 j 0 ]
left[ 0] right[10] parts[ r1      ] value[ 1e+006 j 0 ]
left[ 0] right[11] parts[ e2      ] value[      24 j 0 ]
left[ 0] right[14] parts[ e3      ] value[     -12 j 0 ]
left[ 1] right[ 2] parts[ ri      ] value[     100 j 0 ]
left[ 2] right[ 3] parts[ cc      ] value[ 0.001 j 0 ]
left[ 3] right[16] parts[ r1b1    ] value[      0.1 j 0 ]
left[ 4] right[ 9] parts[ r2      ] value[      50 j 0 ]
left[ 4] right[17] parts[ r1b1    ] value[      0.1 j 0 ]
left[ 5] right[11] parts[ r3      ] value[ 1e+004 j 0 ]
left[ 5] right[18] parts[ r1b1    ] value[      0.1 j 0 ]
left[ 6] right[20] parts[ r1b2    ] value[      0.1 j 0 ]
left[ 7] right[ 9] parts[ r2      ] value[      50 j 0 ]
left[ 7] right[21] parts[ r1b2    ] value[      0.1 j 0 ]
left[ 8] right[10] parts[ cc      ] value[ 0.001 j 0 ]
left[ 8] right[11] parts[ r3      ] value[ 1e+004 j 0 ]
left[ 8] right[22] parts[ r1b2    ] value[      0.1 j 0 ]
left[ 9] right[26] parts[ r1b3    ] value[      0.1 j 0 ]
left[12] right[14] parts[ r4      ] value[ 1e+006 j 0 ]
left[12] right[24] parts[ r1b3    ] value[      0.1 j 0 ]
left[13] right[14] parts[ r5      ] value[     1000 j 0 ]
left[13] right[25] parts[ r1b3    ] value[      0.1 j 0 ]
left[15] right[16] parts[ r2b1    ] value[ 1e+010 j 0 ]
left[15] right[17] parts[ e1b1    ] value[        2 j 0 ]
left[17] right[18] parts[ q1b1    ] value[ 0.004 j 0 ]
@ master[r2b1      ] left[ 15] right[16]
left[17] right[18] parts[ r3b1    ] value[ 1e+005 j 0 ]
left[19] right[20] parts[ r2b2    ] value[ 1e+010 j 0 ]
left[19] right[21] parts[ e1b2    ] value[        2 j 0 ]
left[21] right[22] parts[ r3b2    ] value[ 1e+005 j 0 ]
left[21] right[22] parts[ q1b2    ] value[ 0.004 j 0 ]
@ master[r2b2      ] left[ 19] right[20]
left[23] right[24] parts[ r2b3    ] value[ 1e+010 j 0 ]
left[23] right[25] parts[ e1b3    ] value[        2 j 0 ]
left[25] right[26] parts[ q1b3    ] value[ 0.004 j 0 ]
@ master[r2b3      ] left[ 23] right[24]
left[25] right[26] parts[ r3b3    ] value[ 1e+005 j 0 ]
最大の 素子番号 = 36
最大のノード番号 = 26
B? ノード5と8の現在の動作点を確認します

```

(8) バッチファイル名 s s 1. s b で扱う回路図

サンプルバッチファイルの回路図と説明

```

B? /point
f[0
x5 [ 15.8791 j 0 ] abs[ 15.8791] arg[ 0]
x8 [ 15.8791 j 0 ] abs[ 15.8791] arg[ 0]
B? R5 を変化させて、動作点が 12V となる値を求めます
/para

```

x5, x8=12 となる、R5 の値を求めます。

```

/para
データファイル名は ? test
値を変化させる素子名は ? r5
para r5 最低値 ? 100
para r5 最高値 ? 10k
para r5 ステップ ? 100
r5[464.2 ] f[0
x5 [ 9.7852 j 0 ] abs[ 9.7852] arg[ 0]
x8 [ 9.7852 j 0 ] abs[ 9.7852] arg[ 0]
r5[774.3 ] f[0
x5 [ 14.0892 j 0 ] abs[ 14.0892] arg[ 0]
x8 [ 14.0892 j 0 ] abs[ 14.0892] arg[ 0]

```

追加で、R5=460 から 800 の範囲を調べます。

```

r5[595 ] f[0
x5 [ 11.9864 j 0 ] abs[ 11.9864] arg[ 0]
x8 [ 11.9864 j 0 ] abs[ 11.9864] arg[ 0]
r5[596 ] f[0
x5 [ 12.0007 j 0 ] abs[ 12.0007] arg[ 0]
x8 [ 12.0007 j 0 ] abs[ 12.0007] arg[ 0]

```

B? R5=596 位で目標の動作点となります。

```

B? R5=596 に設定します
r5=596
596 j 0

```

```

B? ゲインを確認します
/range
ac 入力信号源名は ? e1
値は ? 1
range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x5 [ -15.3850 j -2.449368e-006] abs[ 15.3850] arg[ -180]
x8 [ 15.2181 j 2.398980e-006] abs[ 15.2181] arg[9.032112e-006]
B? ゲインは約 15 倍ですから、ダーリントン接続による回路よりかなり低い値です。

```

サンプルバッチファイルの回路図と説明

追加で、f=0 に変更し、表示ノードを 3, 5, 6, 8 に変更して、現在の動作点を確認します。

```
? /point
f[0]
x3 [-2.742966e-005 j 0] abs[2.742966e-005] arg[180]
x5 [12.0007 j 0] abs[12.0007] arg[0]
x6 [-2.742966e-005 j 0] abs[2.742966e-005] arg[180]
x8 [12.0007 j 0] abs[12.0007] arg[0]
```

ノード 3、6 はほとんど 0V です。

ゲインは約 15 なので、e1=-12/15=-0.8V から 12/15=0.8V まで変化(振幅 1.6Vp-p)すると、ノード 5 とノード 8 はおよそ 24Vp-p の変化をすると予想できます。

ノード 1 と 3 の間に、抵抗 rg=0.1 を追加して、e1 の変化に対するノード 5 とノード 8 の電圧変化を確認します。

```
left[ 0] right ? /
left[ 1] right ? 3
素子名は ? rg
値は ? 0.1
left[ 1] right ? *
```

```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[11] j[ 0] 部品名[e2] 値[24 j 0]
i[14] j[ 0] 部品名[e3] 値[-12 j 0]
i[17] j[15] 部品名[e1b1] 値[2 j 0]
i[21] j[19] 部品名[e1b2] 値[2 j 0]
i[25] j[23] 部品名[e1b3] 値[2 j 0]
独立電圧源素子の個数 6
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
3 5 6 8
表示ノード番号 (0 なら全て、-1 ならこのまま) ? 5
ノード番号 (0 なら終了) ? 8
ノード番号 (0 なら終了) ?
```

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? -0.8
para e1 最高値 ? 0.8
para e1 ステップ ? 0.01
```

サンプルバッチファイルの回路図と説明

2 161 表示ノード数 2 計算ポイント数 161

-0.8 e1=-0.8 V

5 24.3048 x5=24.3048 V 電源 e2 より高い電圧なので正常に動作していない

8 -0.291684

00

-0.79

5 24.1509

8 -0.138025

00

-0.78

5 23.9971

8 0.0156345

00

-0.77 e1=-0.77 V

5 23.8433 x5=23.8433 V 動作している

8 0.169294 x8=0.169294 V 動作している

00

-0.76

5 23.6895

8 0.322953

00

.

.

0.77 e1=-0.77 V

5 0.157127 x5=0.157127 V 動作している

8 23.8329 x8=23.8329 V 動作している

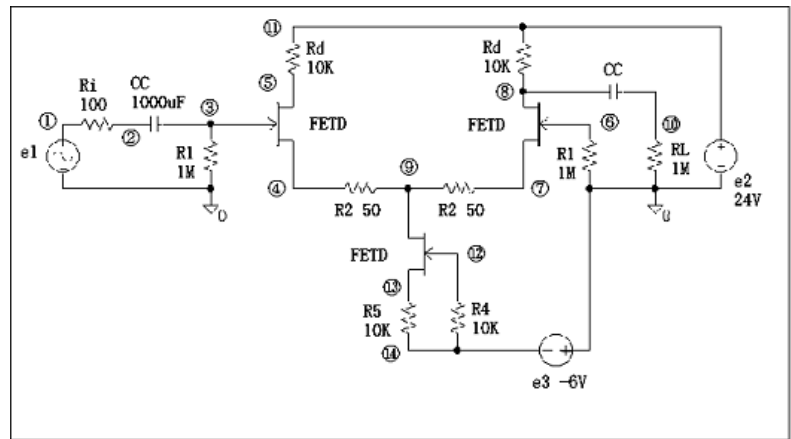
00

0.78 e1=-0.78 V

5 0.00332086 x5=0.00332086 余裕がない

8 23.9865

0 0



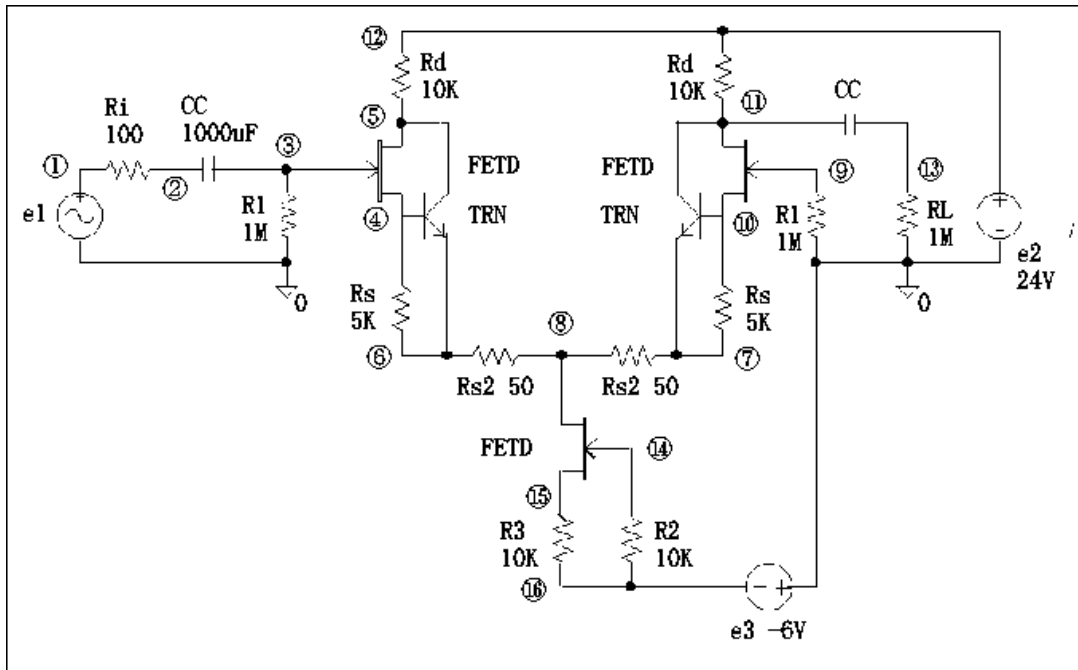
e1 が 0V を中心として-0.78~0.78 V (振幅 1.56 Vp-p) の変化をすると、
x5 は 23.8433~0.157127 V (振幅 23.69) 変化する。ゲインは-15.18。
x10 は 0.169294~23.8329 V (振幅 23.66) 変化する。ゲインは 15.17。

(8) バッチファイル名 `ss1. sb` で扱う回路図

サンプルバッチファイルの回路図と説明

(9) バッチファイル名 `ss2.sb` で扱う回路図

FETとトランジスタを組み合わせた差動増幅回路



設計の指針

1. ノード5とノード11の直流動作点を電源電圧 e_2 の半分に設定する。

回路図の下側にあるFET回路は、一般的には定電流回路といわれており、
 R_3 によって定電流の値が決定されている。この値を変えればノード5とノード11
 の電圧を変更する事が出来る。

/para を使用して、 R_3 を100オームから10Kまで変化させて、ノード5と
 ノード11の電圧が12Vになる R_3 の値を求める。

$R_3 = 600$ オームで希望の電圧になることが分かる。

2. 入力信号の周波数が1000Hzの時のゲインを調べる。

ゲインは約6.3倍である。ノード5は位相反転出力、ノード11は非反転である。
 トランジスタのダーリントン接続の場合とFETのみの場合の中間のゲインが得られる。

トランジスタの場合に $R_1 = 1M$ とした場合よりもゲインは大きい。

3. 入力インピーダンスを求める。

$Z_i = 1M$ が得られる。これは、 R_1 の値と同じである。

$R_1 = 10M$ に増加すれば入力インピーダンスを10Mに増加することが可能であること
 がわかる。この場合にはトランジスタの回路とは異なり、ゲインに影響はない。

4. ノード5側にもノード11側と同じように負荷抵抗を接続すればゲインを等しくする
 事が出来る。

(9) バッチファイル名 `ss2.sb` で扱う回路図

サンプルバッチファイルの回路図と説明

回路の入力は次のように行います。

```

B? /ipart
left[ 0] right ? 1
素子名は ? e1
値は ? 1

left[ 0] right ? 3
素子名は ? r1
値は ? 1m

left[ 0] right ? 9
素子名は ? r1
素子名 r1 は既に使用されています。
同じ素子名を使用しますか? (y/n) y

left[ 0] right ? 13
素子名は ? r1
値は ? 1m

left[ 0] right ? 12
素子名は ? e2
値は ? 24

left[ 0] right ? 16
素子名は ? e3
値は ? -12

left[ 0] right ? /

left[ 1] right ? 2
素子名は ? ri
値は ? 100

left[ 1] right ? /

left[ 2] right ? 3
素子名は ? cc
値は ? 1000u

left[ 2] right ? /

left[ 3] right ? 5
素子名は ? b1
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
fetd:3,4,5

left[ 3] right ? /

left[ 4] right ? 5
素子名は ? b2
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn:4,6,5

left[ 4] right ? 6
素子名は ? rs
値は ? 5k

left[ 4] right ? /

```

(9) バッチファイル名 s s 2. s b で扱う回路図

サンプルバッチファイルの回路図と説明

```

left[ 5] right ? 12
素子名は ? rd
値は ? 10k

left[ 5] right ? /

left[ 6] right ? 8
素子名は ? rs2
値は ? 50

left[ 6] right ? /

left[ 7] right ? 8
素子名は ? rs2
素子名 rs2 は既に使用されています。
同じ素子名を使用しますか？ (y/n) y

left[ 7] right ? 11
素子名は ? b4
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
trn:10,7,11

left[ 7] right ? 10
素子名は ? rs
素子名 rs は既に使用されています。
同じ素子名を使用しますか？ (y/n) y

left[ 7] right ? /

left[ 8] right ? 14
素子名は ? b5
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
fetd:14,15,8

left[ 8] right ? /

left[ 9] right ? 11
素子名は ? b3
ファイル名と接続ノードを入力して下さい(ic1:0,1,2 の様に)
fetd:9,10,11

left[ 9] right ? /

left[ 10] right ? /

left[ 11] right ? 12
素子名は ? rd
素子名 rd は既に使用されています。
同じ素子名を使用しますか？ (y/n) y

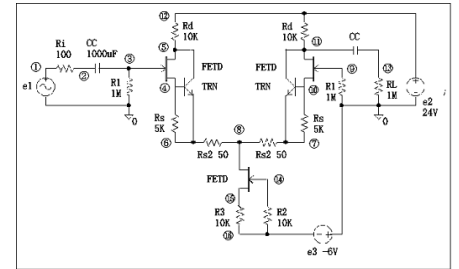
left[ 11] right ? 13
素子名は ? cc
素子名 cc は既に使用されています。
同じ素子名を使用しますか？ (y/n) y

left[ 11] right ? /

left[ 12] right ? /

left[ 13] right ? /

```



サンプルバッチファイルの回路図と説明

```

left[ 14] right ? 16
素子名は ? r2
値は ? 10k

left[ 14] right ? /

left[ 15] right ? 16
素子名は ? r3
値は ? 10k

left[ 15] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[12] j[ 0] 部品名[e2] 値[24 j 0]
i[16] j[ 0] 部品名[e3] 値[-12 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 5
ノード番号 (0 なら終了) ? 11
ノード番号 (0 なら終了) ?
未解凍のブロックがあります。 解凍しますか (y/n) y
bname b1 sfnc fetd:3,4,5

func fetd:3,4,5 name fetd.cir blk b1
@ func fetd:1,2,3
bname b2 sfnc trn:4,6,5

func trn:4,6,5 name trn.cir blk b2
@ func trn:1,2,3
bname b4 sfnc trn:10,7,11

func trn:10,7,11 name trn.cir blk b4
@ func trn:1,2,3
bname b5 sfnc fetd:14,15,8

func fetd:14,15,8 name fetd.cir blk b5
@ func fetd:1,2,3
bname b3 sfnc fetd:9,10,11

func fetd:9,10,11 name fetd.cir blk b3
@ func fetd:1,2,3

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[12] j[ 0] 部品名[e2] 値[24 j 0]
i[16] j[ 0] 部品名[e3] 値[-12 j 0]
i[19] j[17] 部品名[e1b1] 値[2 j 0]
i[24] j[22] 部品名[ebb2] 値[0.6 j 0]
i[28] j[26] 部品名[ebb4] 値[0.6 j 0]
i[31] j[29] 部品名[e1b5] 値[2 j 0]
i[35] j[33] 部品名[e1b3] 値[2 j 0]
独立電圧源素子の個数 8
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
5 11

表示ノード番号 (0 なら全て、-1 ならこのまま) ? -1

```

(9) バッチファイル名 s s 2. s b で扱う回路図

サンプルバッチファイルの回路図と説明

```

? /dpart
left[ 0] right[ 1] parts[ e1      ] value[      1 j 0 ]
left[ 0] right[ 3] parts[  r1     ] value[ 1e+006 j 0 ]
left[ 0] right[ 9] parts[  r1     ] value[ 1e+006 j 0 ]
left[ 0] right[12] parts[  e2     ] value[      24 j 0 ]
left[ 0] right[13] parts[  r1     ] value[ 1e+006 j 0 ]
left[ 0] right[16] parts[  e3     ] value[     -12 j 0 ]
left[ 1] right[ 2] parts[  ri     ] value[     100 j 0 ]
left[ 2] right[ 3] parts[  cc     ] value[     0.001 j 0 ]
left[ 3] right[18] parts[ r1b1    ] value[      0.1 j 0 ]
left[ 4] right[ 6] parts[  rs     ] value[     5000 j 0 ]
left[ 4] right[19] parts[ r1b1    ] value[      0.1 j 0 ]
left[ 4] right[24] parts[ r1b2    ] value[      0.1 j 0 ]
left[ 5] right[12] parts[  rd     ] value[ 1e+004 j 0 ]
left[ 5] right[20] parts[ r1b1    ] value[      0.1 j 0 ]
left[ 5] right[23] parts[ rbb2    ] value[      0.1 j 0 ]
left[ 6] right[ 8] parts[  rs2    ] value[      50 j 0 ]
left[ 6] right[21] parts[ rbb2    ] value[      0.1 j 0 ]
left[ 7] right[ 8] parts[  rs2    ] value[      50 j 0 ]
left[ 7] right[10] parts[  rs     ] value[     5000 j 0 ]
left[ 7] right[25] parts[ rbb4    ] value[      0.1 j 0 ]
left[ 8] right[32] parts[ r1b5    ] value[      0.1 j 0 ]
left[ 9] right[34] parts[ r1b3    ] value[      0.1 j 0 ]
left[10] right[28] parts[ r1b4    ] value[      0.1 j 0 ]
left[10] right[35] parts[ r1b3    ] value[      0.1 j 0 ]
left[11] right[12] parts[  rd     ] value[ 1e+004 j 0 ]
left[11] right[13] parts[  cc     ] value[     0.001 j 0 ]
left[11] right[27] parts[ rbb4    ] value[      0.1 j 0 ]
left[11] right[36] parts[ r1b3    ] value[      0.1 j 0 ]
left[14] right[16] parts[  r2     ] value[ 1e+004 j 0 ]
left[14] right[30] parts[ r1b5    ] value[      0.1 j 0 ]
left[15] right[16] parts[  r3     ] value[      600 j 0 ]
left[15] right[31] parts[ r1b5    ] value[      0.1 j 0 ]
left[17] right[18] parts[ r2b1    ] value[ 1e+010 j 0 ]
left[17] right[19] parts[  e1b1   ] value[       2 j 0 ]
left[19] right[20] parts[ q1b1    ] value[     0.004 j 0 ]
@ master[r2b1      ] left[17] right[18]
left[19] right[20] parts[ r3b1    ] value[ 1e+005 j 0 ]
left[21] right[22] parts[ reb2    ] value[      26 j 0 ]
left[21] right[23] parts[ kb2     ] value[     100 j 0 ]
@ master[reb2      ] left[21] right[22]
left[22] right[24] parts[ ebb2    ] value[      0.6 j 0 ]
left[25] right[26] parts[ reb4    ] value[      26 j 0 ]
left[25] right[27] parts[ kb4     ] value[     100 j 0 ]
@ master[reb4      ] left[25] right[26]
left[26] right[28] parts[ ebb4    ] value[      0.6 j 0 ]
left[29] right[30] parts[ r2b5    ] value[ 1e+010 j 0 ]
left[29] right[31] parts[  e1b5   ] value[       2 j 0 ]
left[31] right[32] parts[ r3b5    ] value[ 1e+005 j 0 ]
left[31] right[32] parts[ q1b5    ] value[     0.004 j 0 ]
@ master[r2b5      ] left[29] right[30]
left[33] right[34] parts[ r2b3    ] value[ 1e+010 j 0 ]
left[33] right[35] parts[  e1b3   ] value[       2 j 0 ]
left[35] right[36] parts[ q1b3    ] value[     0.004 j 0 ]
@ master[r2b3      ] left[33] right[34]
left[35] right[36] parts[ r3b3    ] value[ 1e+005 j 0 ]
最大の 素子番号 = 50
最大のノード番号 = 36

```

サンプルバッチファイルの回路図と説明

```

B? ノード5と11の現在の動作点を確認します
f=0
    0 j 0

B? /point
f[0
x5 [ 23.0105 j 0 ] abs[ 23.0105] arg[ 0]
x11 [ 23.0105 j 0 ] abs[ 23.0105] arg[ 0]
B? R3 を変化させて、動作点が 12V となる値を求めます。

```

```

B? /para
データファイル名は ? test
値を変化させる素子名は ? r3
para r3 最低値 ? 100
para r3 最高値 ? 10k
para r3 ステップ ? 1

```

```

r3[464.2 ] f[0
x5 [ 9.7896 j 0 ] abs[ 9.7896] arg[ 0]
x11 [ 9.7896 j 0 ] abs[ 9.7896] arg[ 0]
r3[774.3 ] f[0
x5 [ 14.0937 j 0 ] abs[ 14.0937] arg[ 0]
x11 [ 14.0937 j 0 ] abs[ 14.0937] arg[ 0]

```

追加で、R3=400 から 800 の範囲を調べます。

```

r3[595 ] f[0
x5 [ 11.9910 j 0 ] abs[ 11.9910] arg[ 0]
x11 [ 11.9910 j 0 ] abs[ 11.9910] arg[ 0]
r3[596 ] f[0
x5 [ 12.0052 j 0 ] abs[ 12.0052] arg[ 0]
x11 [ 12.0052 j 0 ] abs[ 12.0052] arg[ 0]

```

```

B? R3=596 で目標の動作点となります
R3=596 に設定します
r3=596
    596 j 0

```

ゲインを調べます。

```

B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1k
range 周波数 最高値 ? 1k
range 周波数 ステップ ? 1
e1[1 j 0 ]
f[1,000 ]
x5 [ -64.2221 j -1.023650e-005] abs[ 64.2221] arg[ -180]
x11 [ 63.5717 j 1.003277e-005] abs[ 63.5717] arg[ 9.042308e-006]
B? ゲインは、6.3倍となり FET のみより大きく、
トランジスタのダーリントン接続よりも小さい値が得られました。

1 番のバッチファイル ss2.sb を終了します。

```

サンプルバッチファイルの回路図と説明

追加で、f=0 に変更し、表示ノードを 3, 5, 9, 11 に変更して、現在の動作点を確認します。

```
? /point
f[0]
x3 [-7.658912e-007 j 0] abs[7.658912e-007] arg[180]
x5 [12.0052 j 0] abs[12.0052] arg[0]
x9 [-7.658912e-007 j 0] abs[7.658912e-007] arg[180]
x11 [12.0052 j 0] abs[12.0052] arg[0]
```

ss1 の回路と同様に、ノード 3、9 はほとんど 0V です。

ゲインは約 63 なので、 $e1 = -12/63 = -0.19$ V から $12/63 = 0.19$ V まで変化(振幅 0.38 Vp-p)すると、ノード 5 とノード 11 はおよそ 24Vp-p の変化をすると予想できます。

ノード 1 と 3 の間に、抵抗 $rg=0.1$ を追加して、 $e1$ の変化に対するノード 5 とノード 11 の電圧変化を確認します。

```
left[ 0] right ? /
left[ 1] right ? 3
素子名は ? rg
値は ? 0.1
left[ 1] right ? *
```

```
? /para
データファイル名は ? ss2-e1
値を変化させる素子名は ? e1
para e1 最低値 ? -0.19
para e1 最高値 ? 0.19
para e1 ステップ ? 0.002
```

2 191 表示ノード数 2 計算ポイント数 191

-0.19 e1=-0.19 V

5 24.1893 x5=24.1893 V 電源 e2 より高い電圧なので正常に動作していない

11 -0.17598

0 0

.

.

.

-0.184 e1=-0.184 V

5 23.8045 x5=23.8045 V 正常に動作している

11 0.208691 x11=0.208691 V 正常に動作している

0 0

(9) バッチファイル名 ss2.sb で扱う回路図

サンプルバッチファイルの回路図と説明

0.182

5 0.33414

11 23.6737

0 0

0.184 e1=0.184 V

5 0.205886 x5=0.205886 V 正常に動作している

11 23.8019 x11=23.8019 V 正常に動作している

0 0

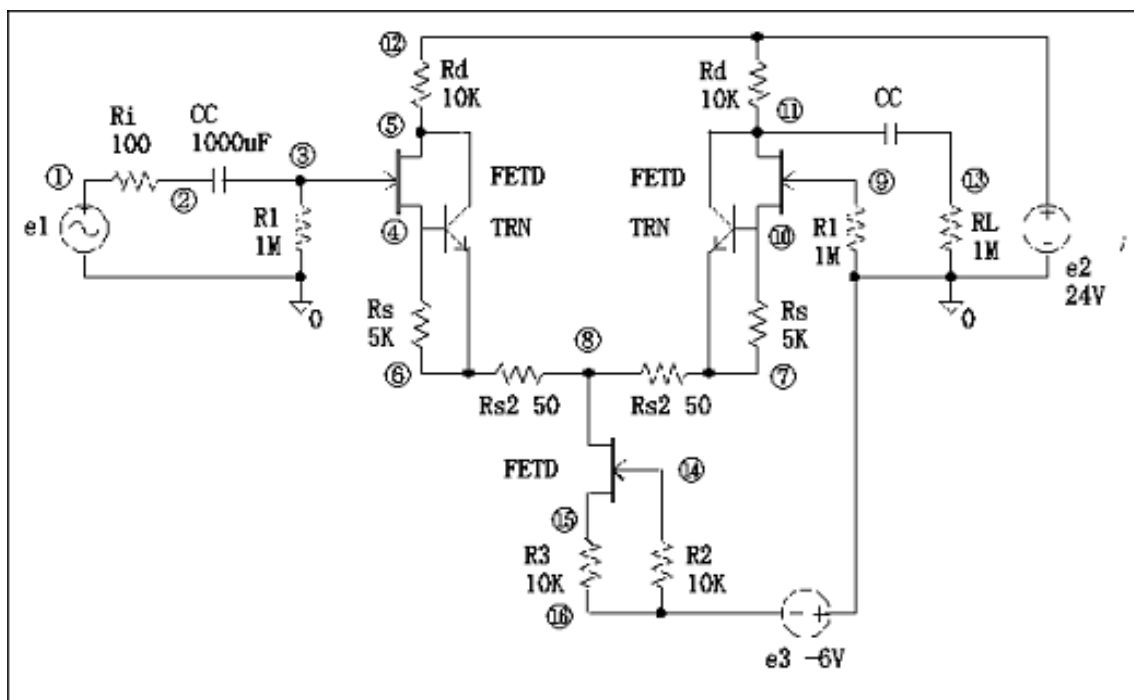
0.186 e1=0.186 V

5 0.0776331 x5=0.0776331 V 余裕が少ない

11 23.9301

0 0

e1 が 0V を中心として -0.184～0.184 V (振幅 0.368 V_{p-p}) の変化をすると、
 x5 は 23.8045～0.205886 V (振幅 23.6) 変化する。ゲインは -64.1。
 x11 は 0.208691～23.8019 V (振幅 23.6) 変化する。ゲインは 64.1。



ss2 の差動増幅回路を回路ブロックにして、部品として利用する。

回路を変更する。

1. 直流から交流までの入力信号に対応する。
2. 出力電圧を $-12\text{V} \sim 12\text{V}$ 程度の範囲に拡大する。
3. $e1=0$ における出力ノード 5 と 11 の電圧を 0V に調整しておく。
4. 素子やノード番号の変更を最小限とする。

変更点：

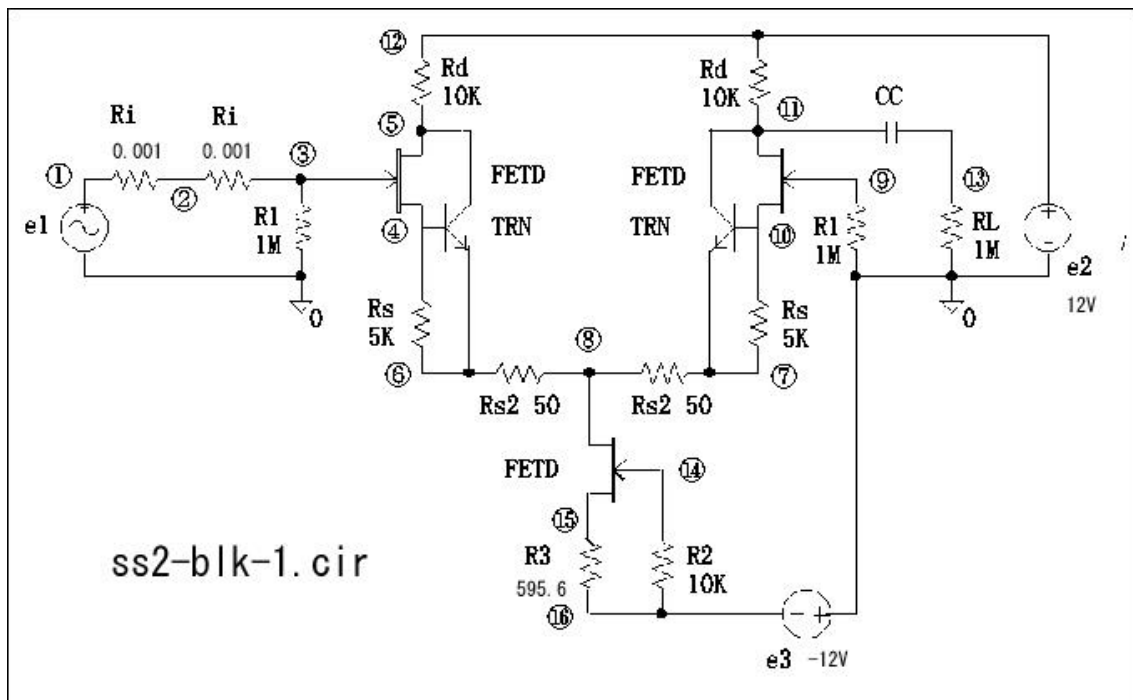
R_i を 100 から 0.001 に変更し、CC を R_i に変更する。

$e3$ を -6V から -12V に変更する。

$R3$ を 10K から 595.6 に変更する。

回路ブロックとしては、 $e1$ は取り除く。

回路ブロックの接続ノードは、0, 1, 9, 5 とする。



回路ブロックファイル ss2-blk-1.cir の作成方法

@ss2 と入力して、バッチファイル ss2.sb を実行します。

表示ノードを入力したところで一時停止しますので、/exit を入力してバッチファイルを終了します。

```

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[12] j[ 0] 部品名[e2] 値[24 j 0]
i[16] j[ 0] 部品名[e3] 値[-12 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 5
ノード番号 (0 なら終了) ? 11
ノード番号 (0 なら終了) ? キー入力待ち? /exit
バッチファイルを強制終了します

```

トランジスタと FET が未解凍ですが、すべて解凍しません。

```

未解凍のブロックがあります。 解凍しますか (y/n)      n
未解凍のブロックがあります。 解凍しますか (y/n)      n
未解凍のブロックがあります。 解凍しますか (y/n)      n
未解凍のブロックがあります。 解凍しますか (y/n)      n
未解凍のブロックがあります。 解凍しますか (y/n)      n

```

```

使用した式の個数 0、密度 0.000 %
使用した式のサイズ 393
最大の 素子番号 = 22
最大のノード番号 = 16
未解凍のブロックを 5 個含んでいます

```

回路情報を保存します。ファイル名は「ss2」、外部ノードは「0,1,9,5」と入力します。

```

? /save
セーブ
ファイル名は ? ss2
外部ノード (1, 2, 3 の様に)
0, 1, 9, 5

```

(9) バッチファイル名 ss2.sb で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 187 / 271

サンプルバッチファイルの回路図と説明

テキストエディタで `ss2.cir` を開きます。

```
@ss2:0,1,9,5
16
0 1 @e @e1 @ 0 1 0 @
0 3 @r @r1 @ 0 1e+006 0 @
0 9 @r @r1 @ 0 1e+006 0 @
0 12 @e @e2 @ 0 24 0 @
0 13 @r @r1 @ 0 1e+006 0 @
0 16 @e @e3 @ 0 -12 0 @
1 2 @r @ri @ 0 100 0 @
2 3 @c @cc @ 0 0.001 0 @
3 5 @b @b1 @ 1 0 0 @fetd:3,4,5
4 5 @b @b2 @ 2 0 0 @trn:4,6,5
4 6 @r @rs @ 0 5000 0 @
5 12 @r @rd @ 0 10000 0 @
6 8 @r @rs2 @ 0 50 0 @
7 8 @r @rs2 @ 0 50 0 @
7 10 @r @rs @ 0 5000 0 @
7 11 @b @b4 @ 3 0 0 @trn:10,7,11
8 14 @b @b5 @ 4 0 0 @fetd:14,15,8
9 11 @b @b3 @ 5 0 0 @fetd:9,10,11
11 12 @r @rd @ 0 10000 0 @
11 13 @c @cc @ 0 0.001 0 @
14 16 @r @r2 @ 0 10000 0 @
15 16 @r @r3 @ 0 10000 0 @
fnc
```

```
@ss2-blk-1:0,1,9,5
16
0 3 @r @r1 @ 0 1e+006 0 @
0 9 @r @r1 @ 0 1e+006 0 @
0 12 @e @e2 @ 0 12 0 @
0 13 @r @r1 @ 0 1e+006 0 @
0 16 @e @e3 @ 0 -12 0 @
1 2 @r @ri @ 0 0.001 0 @
2 3 @r @ri2 @ 0 0.001 0 @
3 5 @b @b1 @ 1 0 0 @fetd:3,4,5
4 5 @b @b2 @ 2 0 0 @trn:4,6,5
4 6 @r @rs @ 0 5000 0 @
5 12 @r @rd @ 0 10000 0 @
6 8 @r @rs2 @ 0 50 0 @
7 8 @r @rs2 @ 0 50 0 @
7 10 @r @rs @ 0 5000 0 @
7 11 @b @b4 @ 3 0 0 @trn:10,7,11
8 14 @b @b5 @ 4 0 0 @fetd:14,15,8
9 11 @b @b3 @ 5 0 0 @fetd:9,10,11
11 12 @r @rd @ 0 10000 0 @
11 13 @c @cc @ 0 0 0 @
14 16 @r @r2 @ 0 10000 0 @
15 16 @r @r3 @ 0 595.6 0 @
fnc
```

まず、上図左側に表示されているファイルで、

```
0 1 @e @e1 @ 0 1 0 @
```

の行を削除します。

次に、

```
15 16 @r @r3 @ 0 10000 0 @
```

を

```
15 16 @r @r3 @ 0 595.6 0 @
```

に変更します。

それから、

```
@ss2:0,1,9,5
```

を

```
@ss2-blk-1:0,1,9,5
```

に変更します。

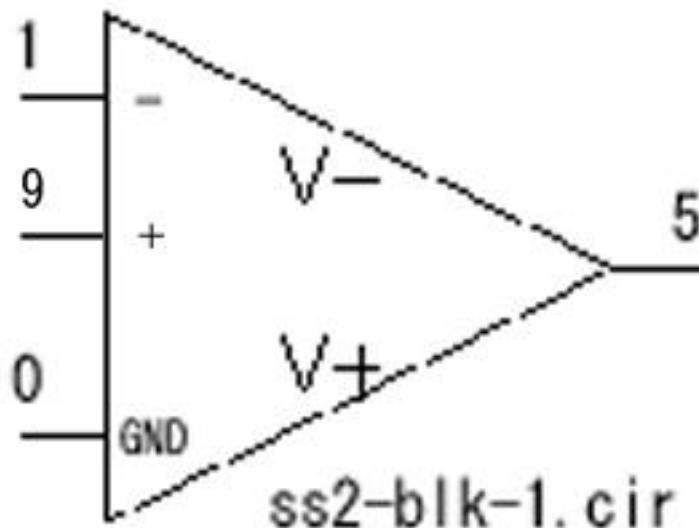
最後に、ファイル名を「`ss2-blk-1.cir`」と名前を付けて保存します。

(9) バッチファイル名 `ss2.s b` で扱う回路図

`ss2` の差動増幅回路を回路ブロックにして、部品として利用する。 188 / 271

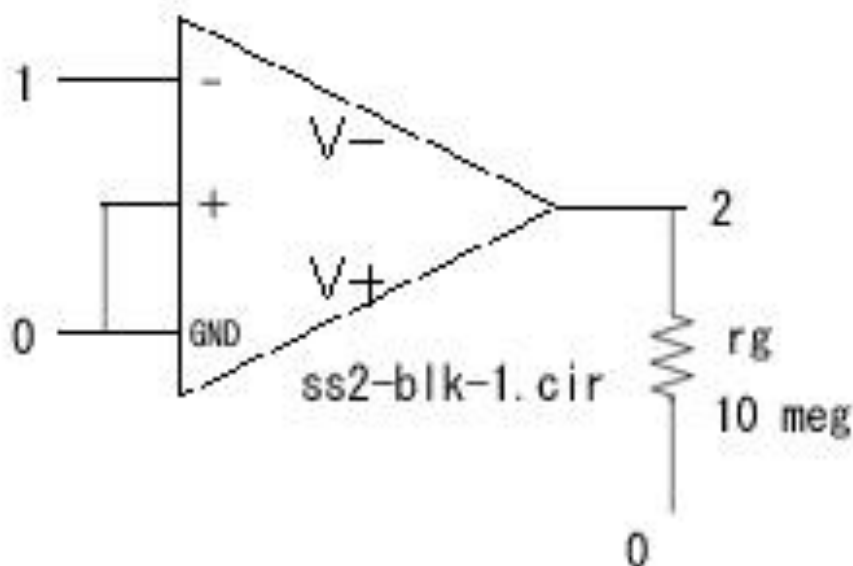
サンプルバッチファイルの回路図と説明

素子 ss2-blk-1.cir として保存したブロック回路。



新しい回路で ss2-blk-1.cir を利用する場合には、ブロック b? の 4 つのノード 0, 1, 9, 5 を新しい回路の対応するノード番号に接続する必要がある。

ss2-blk-1.cir を 1 個だけ使用するなら、次のような回路にする必要がある。



(9) バッチファイル名 ss2.sb で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 189 / 271

サンプルバッチファイルの回路図と説明

この回路を作成するバッチファイル `ss2blk1mk.sb` は次のようになります。

```
/ipart
2
rg
10meg
1
b1
ss2-blk-1:0,1,0,2
*
2

n
```

このように、ノード 2 を入力して、抵抗 `rg=10 meg` を追加する必要があります。

バッチファイル `ss2blk1mk.sb` を実行直後の回路データは次のようになります。

```
? /dpart
left[ 0] right[ 1] parts[ b1          ] value[      0 j 0          ]
@ func ss2-blk-1:0,1,0,2
left[ 0] right[ 2] parts[ rg          ] value[ 1e+007 j 0          ]
最大の 素子番号 = 2
最大のノード番号 = 2
未解凍のブロックを 1 個含んでいます
```

`rg` を追加すると、ノードが 2 個ある回路になります。

バッチファイル `ss2blk1norg.sb` を実行後の回路データは次のようになります。

```
B?
1 番のバッチファイル ss2blk1norg.sb を終了します。
? /dpart
left[ 0] right[ 1] parts[ b1          ] value[      0 j 0          ]
@ func ss2-blk-1:0,1,0,2
最大の 素子番号 = 1
最大のノード番号 = 1
未解凍のブロックを 1 個含んでいます
```

`rg` を追加しないと、ノードが 1 個しかない回路になります。

出力ノードが含まれていません！

(9) バッチファイル名 `ss2.sb` で扱う回路図

`ss2` の差動増幅回路を回路ブロックにして、部品として利用する。 190 / 271

サンプルバッチファイルの回路図と説明

rg を追加した場合（左側）と、追加しない場合の/bload 後の回路は下のようになります。

```
@ss2-blk-1-bload:0,1,2↵
35↵
0 0 @r @r1b1 @ 0 1e+006 0 @↵
0 1 @* @b1 @ 1 0 0 @ss2-blk-1:0,1,0,2↵
0 2 @r @rg @ 0 1e+007 0 @↵
0 4 @r @r1b1 @ 0 1e+006 0 @↵
0 10 @* @b3b1 @ 6 0 0 @fetd:0,9,10↵
0 11 @e @e2b1 @ 0 12 0 @↵
0 12 @r @r1b1 @ 0 1e+006 0 @↵
0 15 @e @e3b1 @ 0 -12 0 @↵
0 33 @r @r1b3b1 @ 0 0.1 0 @↵
1 3 @r @r1b1 @ 0 0.001 0 @↵
2 11 @r @rdb1 @ 0 10000 0 @↵
2 19 @r @r1b1b1 @ 0 0.1 0 @↵
2 22 @r @rbb2b1 @ 0 0.1 0 @↵
3 4 @r @r12b1 @ 0 0.001 0 @↵
4 2 @* @b1b1 @ 2 0 0 @fetd:4,5,2↵
4 17 @r @r1b1b1 @ 0 0.1 0 @↵
5 2 @* @b2b1 @ 3 0 0 @trn:5,6,2↵
5 6 @r @rsb1 @ 0 5000 0 @↵
5 18 @r @r1b1b1 @ 0 0.1 0 @↵
5 23 @r @r1b2b1 @ 0 0.1 0 @↵
6 8 @r @rs2b1 @ 0 50 0 @↵
6 20 @r @rbb2b1 @ 0 0.1 0 @↵
7 8 @r @rs2b1 @ 0 50 0 @↵
7 9 @r @rsb1 @ 0 5000 0 @↵
7 10 @* @b4b1 @ 4 0 0 @trn:9,7,10↵
7 24 @r @rbb4b1 @ 0 0.1 0 @↵
8 13 @* @b5b1 @ 5 0 0 @fetd:13,14,8↵
8 31 @r @r1b5b1 @ 0 0.1 0 @↵
9 27 @r @r1b4b1 @ 0 0.1 0 @↵
9 34 @r @r1b3b1 @ 0 0.1 0 @↵

@ss2-blk-1-norg:0,1,2↵
34↵
0 0 @r @r1b1 @ 0 1e+006 0 @↵
0 1 @* @b1 @ 1 0 0 @ss2-blk-1:0,1,0,2↵
0 3 @r @r1b1 @ 0 1e+006 0 @↵
0 9 @* @b3b1 @ 6 0 0 @fetd:0,8,9↵
0 10 @e @e2b1 @ 0 12 0 @↵
0 11 @r @r1b1 @ 0 1e+006 0 @↵
0 14 @e @e3b1 @ 0 -12 0 @↵
0 32 @r @r1b3b1 @ 0 0.1 0 @↵
1 2 @r @r1b1 @ 0 0.001 0 @↵
2 3 @r @r12b1 @ 0 0.001 0 @↵
2 10 @r @rdb1 @ 0 10000 0 @↵
2 18 @r @r1b1b1 @ 0 0.1 0 @↵
2 21 @r @rbb2b1 @ 0 0.1 0 @↵
3 2 @* @b1b1 @ 2 0 0 @fetd:3,4,2↵
3 16 @r @r1b1b1 @ 0 0.1 0 @↵
4 2 @* @b2b1 @ 3 0 0 @trn:4,5,2↵
4 5 @r @rsb1 @ 0 5000 0 @↵
4 17 @r @r1b1b1 @ 0 0.1 0 @↵
4 22 @r @r1b2b1 @ 0 0.1 0 @↵
5 7 @r @rs2b1 @ 0 50 0 @↵
5 19 @r @rbb2b1 @ 0 0.1 0 @↵
6 7 @r @rs2b1 @ 0 50 0 @↵
6 8 @r @rsb1 @ 0 5000 0 @↵
6 9 @* @b4b1 @ 4 0 0 @trn:8,6,9↵
6 23 @r @rbb4b1 @ 0 0.1 0 @↵
7 12 @* @b5b1 @ 5 0 0 @fetd:12,13,7↵
7 30 @r @r1b5b1 @ 0 0.1 0 @↵
8 26 @r @r1b4b1 @ 0 0.1 0 @↵
8 33 @r @r1b3b1 @ 0 0.1 0 @↵
```

ss2blk1mk.sb のように、rg を追加するためにノード 2 を入力すると、回路ブロックの内部接続ノード 1, 9, 5 はそれぞれ新しい回路のノード 1, 0, 2 に正しく接続されます。

rg を追加しなければ、新しい回路にはノード番号が 1 までしかないので、ss2-blk-1.cir の内部ノード番号 1, 2, 3, 4, は同じ番号が与えられますが、ノード 5 は接続ノードが 2 に指定されているのでノード 2 になります。ノード 6 以降は 1 だけ小さい番号がつけられて、正常に動作しない回路になります。

注意：ブロック素子を使用する場合には、接続素子の接続ノードの指定は、新しい回路の最大のノード番号以内でなければならない。

(9) バッチファイル名 ss2.sb で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 191 / 271

サンプルバッチファイルの回路図と説明

```
? @ss2blk1mk
```

/bload を実行してから、ノード 2 を確認します。

```
? /point
f[0
x2 [2.433055e-004 j 0 ] abs[2.433055e-004] arg[ 0]
```

ノード 2 の電圧 x2 はほとんど 0 に調整されていることが確認できます。

ノード 1 に e1 を追加して、e1 を変化させて x2 の変化を確認します。

```
? /ipart
```

```
left[ 0] right ? 1
素子名は ? e1
値は ? 0

left[ 0] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[0 j 0]
i[12] j[ 0] 部品名[e2b1] 値[12 j 0]
i[16] j[ 0] 部品名[e3b1] 値[-12 j 0]
i[19] j[17] 部品名[e1b1b1] 値[2 j 0]
i[24] j[22] 部品名[ebb2b1] 値[0.6 j 0]
i[28] j[26] 部品名[ebb4b1] 値[0.6 j 0]
i[31] j[29] 部品名[e1b5b1] 値[2 j 0]
i[35] j[33] 部品名[e1b3b1] 値[2 j 0]
独立電圧源素子の個数 8
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
2

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 2
ノード番号 (0 なら終了) ?
```

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? -0.15
para e1 最高値 ? 0.15
para e1 ステップ ? 0.05
e1[-0.15 ] f[0 ]
x2 [ 9.6111 j 0 ] abs[ 9.6111] arg[ 0]
e1[-0.1 ] f[0 ]
x2 [ 6.4074 j 0 ] abs[ 6.4074] arg[ 0]
e1[-0.05 ] f[0 ]
x2 [ 3.2037 j 0 ] abs[ 3.2037] arg[ 0]
e1[1.388e-017] f[0 ]
x2 [2.399767e-006 j 0 ] abs[2.399767e-006] arg[ 0]
e1[0.05 ] f[0 ]
x2 [ -3.2037 j 0 ] abs[ 3.2037] arg[ 180]
e1[0.1 ] f[0 ]
x2 [ -6.4074 j 0 ] abs[ 6.4074] arg[ 180]
```

e1 が-0.15 から 0 まで変化すると、x2 は 9.6111 から 0 まで変化します。従って、ゲインは -64.07 と計算されます。

(9) バッチファイル名 ss2. sb で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 192 / 271

2 個使用する回路を作成するバッチファイル **ss2blk2mk.sb** を示します。

```
/ipart
3
rg
10meg
1
b1
ss2-blk-1:0,1,0,2
2
b2
ss2-blk-1:0,0,2,3
*
2
3

n
n
```

バッチファイル **ss2blk2mk.sb** を実行直後の回路データは次のようになります。

```
? /dpart
left[ 0] right[ 1] parts[ b1          ] value[          0 j 0          ]
@ func ss2-blk-1:0,1,0,2
left[ 0] right[ 2] parts[ b2          ] value[          0 j 0          ]
@ func ss2-blk-1:0,0,2,3
left[ 0] right[ 3] parts[ rg          ] value[      1e+007 j 0          ]
最大の 素子番号 = 3
最大のノード番号 = 3
未解凍のブロックを 2 個含んでいます
```

1 番目の **ss2-blk-1.cir** の出力は反転してます。ノード 3 が反転を維持するために、2 番目の **ss2-blk-1.cir** の+側の入力ノードに接続されています。

ss2blk2mk.sb を実行直後の回路は、**ss2-blk-2.cir** として保存してあります。

サンプルバッチファイルの回路図と説明

/bload を実行してから、ノード 2, 3 を確認します。

```
? /point
f[0]
x2 [2.414489e-004 j 0] abs[2.414489e-004] arg[0]
x3 [0.0155 j 0] abs[0.0155] arg[0]
```

x2 はほとんど 0 ですが、x3 は 0.0155 の出力があります。

ノード 1 に e1 を追加して、e1 を変化させて x2, x3 の変化を確認します。

```
left[ 0] right ? 1
素子名は ? e1
値は ? 0

left[ 0] right ? *

独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[0 j 0]
i[13] j[ 0] 部品名[e2b1] 値[12 j 0]
i[17] j[ 0] 部品名[e3b1] 値[-12 j 0]
i[27] j[ 0] 部品名[e2b2] 値[12 j 0]
i[31] j[ 0] 部品名[e3b2] 値[-12 j 0]
i[34] j[32] 部品名[e1b1b1] 値[2 j 0]
i[39] j[37] 部品名[ebb2b1] 値[0.6 j 0]
i[43] j[41] 部品名[ebb4b1] 値[0.6 j 0]
i[46] j[44] 部品名[e1b5b1] 値[2 j 0]
i[50] j[48] 部品名[e1b3b1] 値[2 j 0]
i[54] j[52] 部品名[e1b1b2] 値[2 j 0]
i[59] j[57] 部品名[ebb2b2] 値[0.6 j 0]
i[63] j[61] 部品名[ebb4b2] 値[0.6 j 0]
i[66] j[64] 部品名[e1b5b2] 値[2 j 0]
i[70] j[68] 部品名[e1b3b2] 値[2 j 0]
独立電圧源素子の個数 15
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
2 3

表示ノード番号 (0 なら全て、-1 ならこのまま) ? 2
ノード番号 (0 なら終了) ? 3
ノード番号 (0 なら終了) ?
```

(9) バッチファイル名 ss2. sb で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 194 / 271

サンプルバッチファイルの回路図と説明

```
? /point
f[0]
x2 [2.350701e-006 j 0] abs[2.350701e-006] arg[0]
x3 [1.529840e-004 j 0] abs[1.529840e-004] arg[0]
```

e1 を追加したら、x3 もほとんど 0 になりました。

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? -0.003
para e1 最高値 ? 0.003
para e1 ステップ ? 0.001
e1[-0.003] f[0]
x2 [0.1908 j 0] abs[0.1908] arg[0]
x3 [12.2215 j 0] abs[12.2215] arg[0]
e1[-0.002] f[0]
x2 [0.1272 j 0] abs[0.1272] arg[0]
x3 [8.1477 j 0] abs[8.1477] arg[0]
e1[-0.001] f[0]
x2 [0.0636 j 0] abs[0.0636] arg[0]
x3 [4.0739 j 0] abs[4.0739] arg[0]
e1[0] f[0]
x2 [2.350701e-006 j 0] abs[2.350701e-006] arg[0]
x3 [1.529840e-004 j 0] abs[1.529840e-004] arg[0]
e1[0.001] f[0]
x2 [-0.0636 j 0] abs[0.0636] arg[180]
x3 [-4.0736 j 0] abs[4.0736] arg[180]
e1[0.002] f[0]
x2 [-0.1272 j 0] abs[0.1272] arg[180]
x3 [-8.1474 j 0] abs[8.1474] arg[180]
e1[0.003] f[0]
x2 [-0.1908 j 0] abs[0.1908] arg[180]
x3 [-12.2212 j 0] abs[12.2212] arg[180]
```

e1 が-0.002 から 0 まで変化すると、x3 は 8.1477 から 0 まで変化します。

従って、ゲインは -4073.85 と計算されます。出力が反転しています。

3 個使用する回路を作成するバッチファイル `ss2blk3mk.sb` を示します。

```
/ipart
4
rg
10meg
1
b1
ss2-blk-1:0,1,0,2
2
b2
ss2-blk-1:0,2,0,3
3
b3
ss2-blk-1:0,3,0,4
*
4
@ss2-blk-3:0,1,4
4
0 1 @b @b1 @ 1 0 0 @ss2-blk-1:0,1,0,2
n 0 2 @b @b2 @ 2 0 0 @ss2-blk-1:0,2,0,3
n 0 3 @b @b3 @ 3 0 0 @ss2-blk-1:0,3,0,4
n 0 4 @r @rg @ 0 1e+007 0 @
n fnc
```

バッチファイル `ss2blk3mk.sb` を実行直後の回路データは次のようになります。

```
B?
1 番のバッチファイル ss2blk3mk.sb を終了します。
? /dpart
left[ 0] right[ 1] parts[ b1          ] value[          0 j 0          ]
@ func ss2-blk-1:0,1,0,2
left[ 0] right[ 2] parts[ b2          ] value[          0 j 0          ]
@ func ss2-blk-1:0,2,0,3
left[ 0] right[ 3] parts[ b3          ] value[          0 j 0          ]
@ func ss2-blk-1:0,3,0,4
left[ 0] right[ 4] parts[ rg          ] value[      1e+007 j 0          ]
最大の素子番号 = 4
最大のノード番号 = 4
未解凍のブロックを 3 個含んでいます
```

`ss2blk3mk.sb` を実行直後の回路は、`ss2-blk-3.cir` として保存してあります。

サンプルバッチファイルの回路図と説明

/bload を実行してから、ノード 4 を確認します。

```
? /point
f[0
x4 [ 0.9837 j 0 ] abs[ 0.9837] arg[ 0]
```

x4=0.9837 です。0 からかなりずれています。

/bload を実行する前の回路に部品を追加して、さらに詳しく動作を調べるためにバッチファイル ss2blk3tst.sb を実行します。

バッチファイルでは、回路にブロックを 3 個使用してから、/bload を実行する前に、ノード 5 を追加して、ノード 5 に e1=0 を接続してから、ノード 1 とノード 5 の間に抵抗 Rin=10k を接続します。

```
/ipart
4
rg
10meg
1
b1
ss2-blk-1:0,1,0,2
2
b2
ss2-blk-1:0,2,0,3
3
b3
ss2-blk-1:0,3,0,4
*
4

n
n
n
/padd
5
e1
0
/
```

(9) バッチファイル名 ss2.sb で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 197 / 271

サンプルバッチファイルの回路図と説明

```
5
rin
10k
*
4

n
n
n
% ブロックをロードします
/wait
/bload
4

!cls
% ノード 4 の電圧を調べます
/point
% 現在 e1=0 ですが、ノード 4 は-0.9V 位の電圧があります
% このまま、e1 を変化させて、ノード 4 の変化を確認します
/wait
/para
e1
0
50u
10u
% e1=0 では x4=-0.9398, e1=10u では x4=-3.5028
% e1 が 10u 変化すると、x4 が  $-3.5028 - (-0.9398) = -2.5911$  変化します
% 従って、ゲインは 259110 と計算できます
% ノード 1 とノード 4 の間に抵抗 rout=100k を追加して、ゲインを-10 倍にします。
/wait
/padd
/
4
rout
100k
```

(9) バッチファイル名 ss2.sb で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 198 / 271

サンプルバッチファイルの回路図と説明

```
*  
4  
  
% ノード 4 の電圧を調べます  
/wait  
/point  
% 現在 e1=0 ですが、ノード 4 はほとんど 0 V になりました  
% e1 を変化させて、ノード 4 の変化を確認します  
/wait  
/para  
e1  
-1  
1  
0.2  
% e1=-1 では x4=9.9995, e1=-0.8 では x4=7.9996  
% e1 が 0.2 変化すると、x4 が  $7.9996 - 9.9995 = -1.9999$  変化します  
% 従って、ゲインは -9.9995 と計算できます  
% 設定したゲインは -10 なので、誤差は -0.00005 (-0.005 %)です  
/wait  
% rout を 1meg に変更して、ゲインを-100 倍にします  
rout=1meg  
/wait  
/para  
e1  
-0.1  
0  
0.01  
% e1=-0.1 では x4=9.9957, e1=-0.09 では x4=8.9961  
% e1 が 0.01 変化すると、x4 が  $8.9961 - 9.9957 = -0.9996$  変化します  
% 従って、ゲインは -99.96 と計算できます  
% 設定したゲインは -100 なので、誤差は -0.0004 (-0.04 %)です
```

(9) バッチファイル名 ss2.sb で扱う回路図

ss2の差動増幅回路を回路ブロックにして、部品として利用する。 199 / 271

サンプルバッチファイルの回路図と説明

作成された回路の x4 を確認します。

```
B? ノード4の電圧を調べます
/point
f[0
]
x4 [ 0.0191 j 0 ] abs[ 0.0191] arg[ 0]
B? 現在 e1=0 ですが、ノード4は0.0191 位の電圧があります
このまま、e1を変化させて、ノード4の変化を確認します
```

e1 を変化させて x4 の変化を確認します。

```
B? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 0
para e1 最高値 ? 50u
para e1 ステップ ? 10u
e1[0 ] f[0 ]
x4 [ 0.0191 j 0 ] abs[ 0.0191] arg[ 0]
e1[1e-005 ] f[0 ]
x4 [ -2.5440 j 0 ] abs[ 2.5440] arg[ 180]
e1[2e-005 ] f[0 ]
x4 [ -5.1071 j 0 ] abs[ 5.1071] arg[ 180]
e1[3e-005 ] f[0 ]
x4 [ -7.6701 j 0 ] abs[ 7.6701] arg[ 180]
e1[4e-005 ] f[0 ]
x4 [ -10.2332 j 0 ] abs[ 10.2332] arg[ 180]
e1[5e-005 ] f[0 ]
x4 [ -12.7963 j 0 ] abs[ 12.7963] arg[ 180]
B? e1=0ではx4=0.0191, e1=10uではx4=-2.5440
e1が10u 変化すると、x4 が -2.5440 - 0.0191 = -2.5911 変化します
従って、ゲインは -259110 と計算できます
ノード1とノード4の間に抵抗rout=100kを追加して、ゲインを-10倍にします。
```

ゲインは -259100 と計算されます。出力は反転しています。

ノード1とノード4の間に抵抗 Rout を接続すると、回路のゲインを $-R_{out} / R_{in}$ に設定することができるので、 $R_{in}=10K$, $R_{out}=100K$ ならゲインは-10 倍になるはずです。

```
left[ 0] right ? /
left[ 1] right ? 4
素子名は ? rout
値は ? 100k
left[ 1] right ? *
```

```
B? /point
f[0
]
x4 [7.801486e-007 j 0 ] abs[7.801486e-007] arg[ 0]
B? 現在 e1=0 ですが、ノード4はほとんど0 V になりました
e1を変化させて、ノード4の変化を確認します
```

x4 がほとんど 0V になりました。

(9) バッチファイル名 ss2. sb で扱う回路図

ss2の差動増幅回路を回路ブロックにして、部品として利用する。 200 / 271

サンプルバッチファイルの回路図と説明

```

B? /para
値を変化させる素子名は ? e1
para e1 最低値 ? -1
para e1 最高値 ? 1
para e1 ステップ ? 0.2
e1[-1] ] f[0 ]
x4 [ 9.9995 j 0 ] abs[ 9.9995] arg[ 0]
e1[-0.8] ] f[0 ]
x4 [ 7.9996 j 0 ] abs[ 7.9996] arg[ 0]
e1[-0.6] ] f[0 ]
x4 [ 5.9997 j 0 ] abs[ 5.9997] arg[ 0]
e1[-0.4] ] f[0 ]
x4 [ 3.9998 j 0 ] abs[ 3.9998] arg[ 0]
e1[-0.2] ] f[0 ]
x4 [ 1.9999 j 0 ] abs[ 1.9999] arg[ 0]
e1[-5.551e-017] ] f[0 ]
x4 [ 7.801486e-007 j 0 ] abs[ 7.801486e-007] arg[ 0]
e1[0.2] ] f[0 ]
x4 [ -1.9999 j 0 ] abs[ 1.9999] arg[ 180]
e1[0.4] ] f[0 ]
x4 [ -3.9998 j 0 ] abs[ 3.9998] arg[ 180]
e1[0.6] ] f[0 ]
x4 [ -5.9997 j 0 ] abs[ 5.9997] arg[ 180]
e1[0.8] ] f[0 ]
x4 [ -7.9996 j 0 ] abs[ 7.9996] arg[ 180]
e1[1] ] f[0 ]
x4 [ -9.9995 j 0 ] abs[ 9.9995] arg[ 180]
B? e1=-1ではx4=9.9995, e1=-0.8ではx4=7.9996
e1が0.2 変化すると、x4 が 7.9996 - 9.9995 = -1.9999 変化します
従って、ゲインは -9.9995 と計算できます
設定したゲインは -10 なので、誤差は -0.00005 (-0.005 %)です

```

e1 が-1 から 0 まで変化すると、x4 は 9.9995 から-0.00000078 まで変化します。従って、ゲインは -9.9995 と計算されます。ゲインの誤差は-0.00005(-0.005%)です。

このように、Rout によって負帰還を掛けると、ゲインを正確に設定できます。
 同時に、Rout が無かった時には、e1=0 でも x4=-0.9398 の出力がありましたが。
 Rout を接続することで x4=0.00000078 に改善されました。

回路ブロック ss2-blk-1.cir を n 段接続すると、ゲインは単独時のゲイン gain の n 乗程度に増加します。同時に、無信号時の出力のずれも少しずつ増加します。

もしも、ss2-blk-1.cir で trn が hfe=800 で、fetd が q=0.02 で Rd=50K、R3=4128.4 に変更できれば、ゲインは 64 倍から 330 倍に増加して、3 段接続した場合のゲインは 3500 万倍程度まで高くなると推測できます。

(9) バッチファイル名 ss2.s b で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 201 / 271

バッチファイル ss2blk3mk.sb

ss2blk3mk.sb を実行直後の回路データを ss2-blk-3.cir として保存してあります。

ss2-blk-3.cir を使ってプリエンファシス回路を作ります。

アナログ TV の時代には、テレビの音声信号が電波ノイズ（特に高周波数域）で音質劣化するのを防ぐために、あらかじめ音声信号にプリエンファシス処理を行っていました。

この処理は音声信号の周波数が **fc2** を超えるほど信号を拡大して、周波数が **fc1** を超えると一定の大きさに落ち着いて行くようなフィルタです。

テレビ受像機側ではこの逆の処理を行うディエンファシス処理を行って、高周波数のノイズは低減しながら、音声信号は元の大きさに戻していました。

アナログ TV 規格では、**fc2 = 2,122 Hz(時定数 tc2=75u)**、**fc1 = 10,610Hz(tc1=15u)**です。

バッチファイル ss2premp.sb によって、プリエンファシス回路を作ります。

/ipart

1

e1

0

5

r2

1k

2

b1

ss2-blk-3:0,2,3

/

2

r1

31.7k

/

4

r3

27k

/

4

r4

4.7k

/

(9) バッチファイル名 `ss2.sb` で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 202 / 271

サンプルバッチファイルの回路図と説明

```

5
c1
0.015u
*
3

n

```

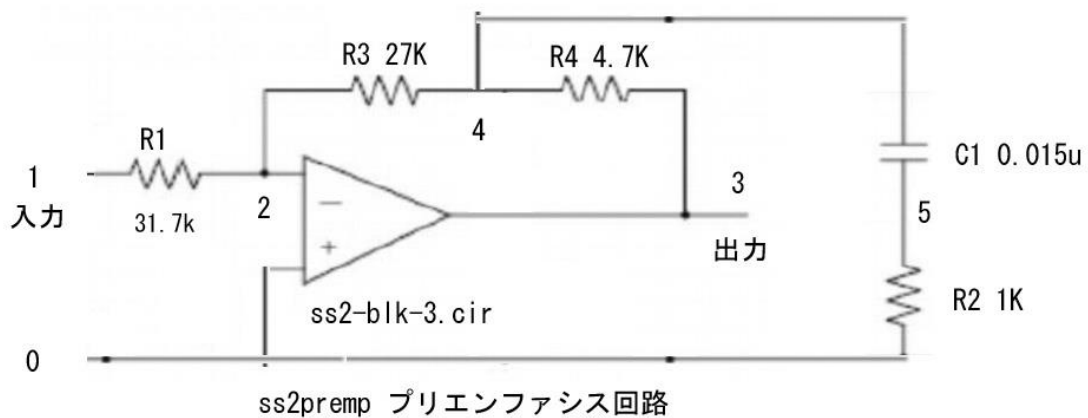
バッチファイル `ss2premp.sb` を実行直後の回路データは次の通りです。

```

B?
1 番のバッチファイル ss2premp.sb を終了します。
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[          0 j 0          ]
left[ 0] right[ 2] parts[ b1          ] value[          0 j 0          ]
@ func ss2-blk-3:0, 2, 3
left[ 0] right[ 5] parts[ r2          ] value[        1000 j 0          ]
left[ 1] right[ 2] parts[ r1          ] value[    3.17e+004 j 0          ]
left[ 2] right[ 4] parts[ r3          ] value[    2.7e+004 j 0          ]
left[ 3] right[ 4] parts[ r4          ] value[         4700 j 0          ]
left[ 4] right[ 5] parts[ c1          ] value[    1.5e-008 j 0          ]
最大の素子番号 = 7
最大のノード番号 = 5
未解凍のブロックを 1 個含んでいます

```

これを回路図で示すと、次の通りです。



`/bload` を実行してから、`/range` によって 10Hz から 100KHz までのノード 3 の値 `x3` をデシベル値で計算してグラフにすると次のようになります。

```

? /point
f[0
x3 [1.931374e-007 j 0          ] abs[1.931374e-007] arg[          0]

```

(9) バッチファイル名 `ss2.sb` で扱う回路図

`ss2` の差動増幅回路を回路ブロックにして、部品として利用する。 203 / 271

サンプルバッチファイルの回路図と説明

```
? /gpos
MANU ON または 最高/最低の比が10以上の時に、
シミュレーションで計算するポイント数を入力して下さい
最低 5 以上、最大 300 以下の数字を入力して下さい
現在の設定は 32 です
16
gpos = 16 に設定しました
? /db
mode = デシベル値で計算するモード
? /mk
```

計算ポイント数

デシベル値

ファイル作成

```
? /manu
? /range
データファイル名は ? ss2premp
ac 入力信号源名は ? e1
値は ? 1

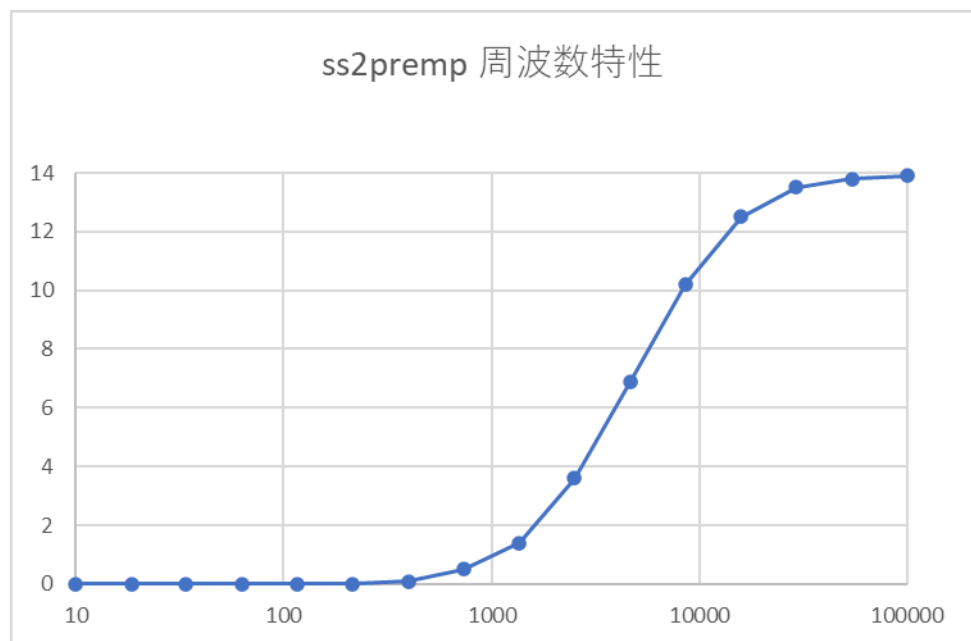
range 周波数 最低値 ? 10
range 周波数 最高値 ? 100k
range 周波数 ステップ ? 100
```

常に等比級数のステップ

周波数特性のグラフは最大と最低の比が大きくなるので、等比級数ステップで計算して、横軸を対数目盛にする。

x3 はデシベル値なので、縦軸は対数目盛にしない。

f Hz	x4 db
10	0
80	0
195	0.03
353	0.11
640	0.36
1160	1.08
2102	2.8
2828	4.14
5125	7.44
9284	10.58
16819	12.6
30470	13.51
55200	13.83
74296	13.9
100000	13.94



1KHz 以下の周波数に対してはゲインは 0db (1 倍) ですが、2KHz 位で 3db (1.4 倍) に上昇します。周波数が高くなるほどゲインが高くなりますが、10KHz を越えるとゲインの上昇が緩やかになり、ゲインは 13.9db 位で最大となります。

(9) バッチファイル名 ss2.sb で扱う回路図

ss2 の差動増幅回路を回路ブロックにして、部品として利用する。 204 / 271

Maple を使って、ss2premp.sb の回路の伝達関数を求めます。

このまま係数行列を求めると(ss2premp.coe を参照)、伝達関数が簡単に理解できない込み入った数式になるので、バッチファイル ss2premp.sb で使用している ss2premp.cir の代わりに、ic1.cir を利用する ic1premp.sb を使って係数行列を作成します。

ic1premp.sb の内容

```
/ipart
1
e1
0
5
r2
1k
2
b1
ic1:0,2,0,3
/
2
r1
31.7k
/
4
r3
27k
/
4
r4
4.7k
/
5
c1
0.015u
*
3

n
```

(9) バッチファイル名 ss2.sb で扱う回路図

Maple を使って、ss2premp.sb の回路の伝達関数を求めます。

/bload を実行してから、/conv で係数行列 ic1premp.coe を作成します。

これを、Maple にコピーアンドペーストして方程式を解きます。

```
ic1-premp.mws
>
restart: with(linalg):
siki:=matrix(5,5,0):e:=vector(5,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 1] := -1/r1 ;
siki[ 2, 2] := +1/r1+1/r1b1+1/r3 ;
siki[ 2, 4] := -1/r3 ;
siki[ 3, 2] := -v1b1 ;
siki[ 3, 3] := 1 ;
siki[ 4, 2] := -1/r3 ;
siki[ 4, 3] := -1/r4 ;
siki[ 4, 4] := +1/r3+1/r4+s*c1 ;
siki[ 4, 5] := -s*c1 ;
siki[ 5, 4] := -s*c1 ;
siki[ 5, 5] := +1/r2+s*c1 ;
x:=linsolve(siki,e);
```

ノード 3 の電圧 x3 を入力電圧 e1 で割って、ゲイン g を計算します。

```
> x3:=x[3]:g:=x3/e1;
g:= v1b1 r1b1 (r4+r4 s c1 r2+r3+s c1 r2 r3+s c1 r3 r4) / (s c1 r2 r1b1 r3
+ r1b1 s c1 r3 r4+r1b1 r3+r1 r3+s c1 r2 r1 r3+r3 r1 s c1 r4+r1 r1b1
- v1b1 r1 r1b1- v1b1 r1 r1b1 s c1 r2+r1b1 r4+r1b1 r4 s c1 r2+r1 r4 s c1 r2
+ s c1 r2 r1 r1b1+r1 r1b1 s c1 r4+r1 r4)
```

ic1 がゲインが無限大の理想アンプであると仮定してゲインを近似します。

ゲインの数式が分数なので、分子 gn と分母 gd に分割し、それぞれを ic1 のゲイン v1b1 で割って、v1b1 を無限大にします。その後、g=gn/gd によってゲインを計算します。

```
> gn:=collect(limit(numer(g/v1b1), v1b1=infinity),s);
gn:= r1b1 (r4 c1 r2+ c1 r2 r3+ c1 r3 r4) s+ r1b1 (r4+ r3)
> gd:=limit(denom(g/v1b1),v1b1=infinity);
gd:= -signum(r1 r1b1+ s c1 r2 r1 r1b1) ∞
> gd:=factor(r1*r1b1+s*c1*r2*r1*r1b1);
gd:= r1 r1b1 (1+ s c1 r2)
> g:=simplify(gn/gd);
g:= 
$$\frac{r4+r4 s c1 r2+r3+s c1 r2 r3+s c1 r3 r4}{r1 (1+ s c1 r2)}$$

```

(9) バッチファイル名 ss2. sb で扱う回路図

Maple を使って、ss2premp.sb の回路の伝達関数を求めます。

$r1, r2$ などに回路図の素子値を代入します。また、 $s := j \cdot 2 \cdot \pi \cdot f$ 、 $\pi := 3.141592$ および $j = \sqrt{-1}$ を代入します。Maple では、 j ではなく I を使用します。

```
[> g;
```

$$\frac{1}{31700} \frac{31700 + 14.94769473 / f}{1 + .00009424776000 / f}$$

g を分子 gn と分母 gd に分割します。

```
[> gd:=denom(g);
      gd:= 31700 + 2.987653992 / f
[> gn:=numer(g);
      gn:= 31700 + 14.94769473 / f
```

分子と分母の両方に 31700 があるので、それぞれ 31700 で割ります。

```
[> gn:=gn/31700;
      gn:= 1 + .0004715361114 / f
[> gd:=gd/31700;
      gd:= 1 + .00009424776000 / f
```

分子と分母を、 $1 + i \cdot \frac{f}{f_x}$ の形に変形します。

```
[> tn:=coeff(gn,f)/i;fn:=1/tn;
      fn:= 2120.728351
[> td:=coeff(gd,f)/i;fd:=1/td;
      fd:= 10610.33175
```

$$gn := 1 + i \cdot \frac{f}{2120.728}, gd := 1 + i \cdot \frac{f}{10610.33}$$

複素数 $a = x + i \cdot y$ の絶対値は $|a| = \sqrt{x^2 + y^2}$ で計算できるので、

ゲインは $g = gn / gd$ より、その絶対値は $g(f) = \sqrt{\frac{1 + \left(\frac{f}{2120.728}\right)^2}{1 + \left(\frac{f}{10610.33}\right)^2}}$ と表すことが出来ます。

この数式から、 $f=0$ ではゲインは 1 倍、 $f=2120.728\text{Hz}$ で約 $\sqrt{2}$ (3 dB) にゲインが上昇して、その後ゲインは増加し続けて、 $f=10610.33\text{Hz}$ から上昇が緩慢になります。

サンプルバッチファイルの回路図と説明

```

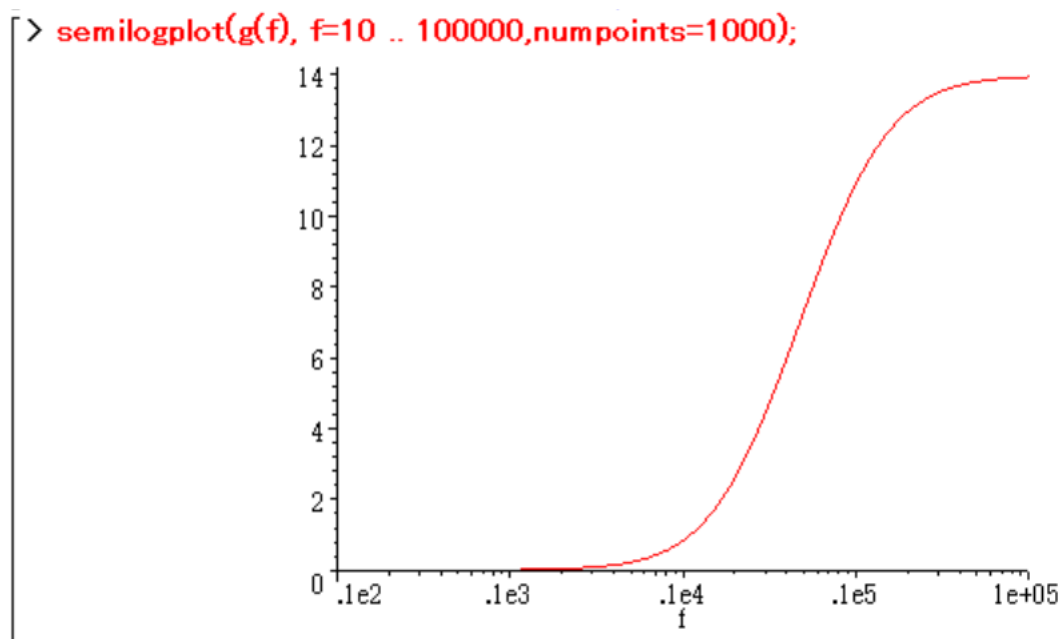
> g:=(1+I*f/2120.728351)/(1+I*f/10610.33175);
      
$$g := \frac{1 + .0004715361114 \, I f}{1 + .00009424775997 \, I f}$$

> gain:=20*log10(sqrt((1+(.4715361114e-3*f)^2)/(1+(.9424775997e-4*f)^2)));
      
$$gain := 20 \log_{10} \left( \sqrt{\frac{1 + 2223463044 \, 10^{-6} f^2}{1 + .8882640259 \, 10^{-8} f^2}} \right)$$

> g:= f -> 20*log10(sqrt((1+.2223463044e-6*f^2)/(1+.8882640259e-8*f^2)));
      
$$g := f \rightarrow 20 \log_{10} \left( \sqrt{\frac{1 + 2223463044 \, 10^{-6} f^2}{1 + .8882640259 \, 10^{-8} f^2}} \right)$$


```

f=10Hz から 100KHz の範囲のゲインの変化をグラフにします。



f=10 Hz, 2102 Hz, 100000 Hz のゲインは次の通り、sim.exe の結果と等しくなります。

```

> g(10);
      .00009270400540
> g(2102);
      2.804759324
> g(100000);
      13.93821188

```

(9) バッチファイル名 ss2.sb で扱う回路図

Maple を使って、ss2premp.sb の回路の伝達関数を求めます。

差動増幅回路について

バッチファイル s11、ss1、ss2 で紹介した回路は差動増幅回路と呼ばれており、直流から交流までの幅広い周波数帯域の信号を増幅できます。

入力ノードが 2 つと対応する出力ノードも 2 つあり、各出力ノードには入力ノードの電圧の差が増幅されて出力されます。2 つの出力ノードは位相が反転しています。

オペアンプと言われる IC チップには差動増幅回路を多段接続して、ゲインを数百万倍程度まで高くした物が製造されています。

このようなオペアンプはゲインが非常に高いので、負帰還回路を利用することで正確なゲインを設定することが出来ます。

また、2 つの入力ノードの信号に平等にノイズが加えられても、信号の差だけが増幅されるので、雑音の影響を受けずらいという長所があります。

1 つのシリコンチップ上にすべてのトランジスタが作られるので、温度変化によるゲインなどへの影響も少なくなっています。

このような特徴を生かして、電気・電子・化学・物理・医用などの分野で、雑音が比較的多い環境でも正確な測定ができる精密な測定装置の信号増幅回路として利用されています。

例えば、微小信号を増幅する心電計、ひずみゲージ等の精密計測分野や通信分野で使用されます。

オペアンプの回路構成や歴史など、さらに詳しい解説は下記を参照してください。

https://www.rohm.co.jp/electronics-basics/opamps/op_what2

<http://www.tij.co.jp/jp/lit/an/jaja460/jaja460.pdf>

<http://www.ti.com/lit/ds/symlink/lm1458.pdf>

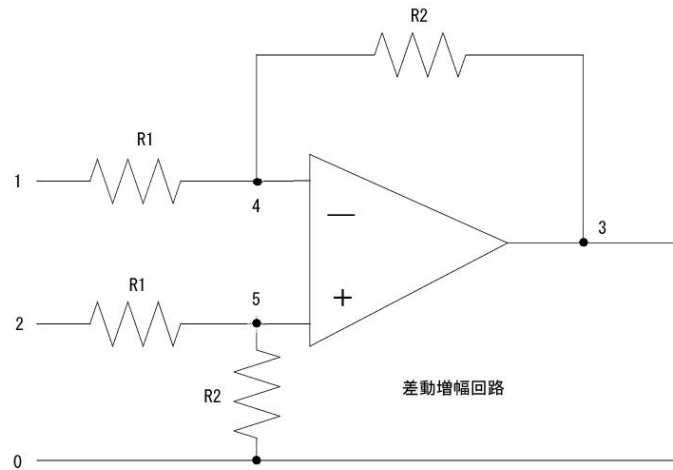
<https://www.njr.co.jp/products/semicon/products/NJM074.html>

<https://www.renesas.com/jp/ja/doc/DocumentServer/006/G13257JJ5V0AN00.pdf>

https://www.marutsu.co.jp/pc/static/large_order/1104opamp_sub1

サンプルバッチファイルの回路図と説明

差動増幅回路



上の回路において、

ノード 1 に e_1 、ノード 2 に e_2 の入力が増えらると、ノード 3 には次の出力 x_3 が得られます。

$$x_3 = \frac{R_2}{R_1} \cdot (e_2 - e_1)$$

出力 e_3 は 2 つの入力電圧の差($e_2 - e_1$)を R_2/R_1 倍した値になります。

回路を入力して、 e_1 と e_2 の変化させて出力 x_3 の変化を確認します。

バッチファイル DifAmp.sb

```
/ipart
```

```
1
```

```
e1
```

```
1
```

```
2
```

```
e2
```

```
1
```

```
5
```

```
r2
```

```
100k
```

```
4
```

```
b1
```

(9) バッチファイル名 `ss2.sb` で扱う回路図
差動増幅回路

サンプルバッチファイルの回路図と説明

```
ic1:0,4,5,3
```

```
/
```

```
4
```

```
r1
```

```
50k
```

```
/
```

```
5
```

```
r1
```

```
y
```

```
/
```

```
4
```

```
r2
```

```
y
```

```
*
```

```
3
```

```
y
```

```
-1
```

```
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[          1 j 0          ]
left[ 0] right[ 2] parts[ e2          ] value[          1 j 0          ]
left[ 0] right[ 3] parts[ v1b1        ] value[        1e+006 j 0          ]
@ master[r1b1          ] left[ 4] right[ 5]
left[ 0] right[ 5] parts[ r2          ] value[        1e+005 j 0          ]
left[ 1] right[ 4] parts[ r1          ] value[        5e+004 j 0          ]
left[ 2] right[ 5] parts[ r1          ] value[        5e+004 j 0          ]
left[ 3] right[ 4] parts[ r2          ] value[        1e+005 j 0          ]
left[ 4] right[ 5] parts[ r1b1        ] value[        1e+006 j 0          ]
最大の素子番号 = 8
最大のノード番号 = 5
```

現在の e1,e2 はどちらも 1V です。e3 を確認します。

```
? /point
f[0          ]
x3 [          0 j 0          ] abs[          0] arg[          0]
? /para
```

e1=e2 なので x3=0 となります。

$R2/R1=2$ なので、 $x3=2 \cdot (e2 - e1)$ となります。

サンプルバッチファイルの回路図と説明

e2 を 1V から 5V まで変化させてみます。

```
? /para
値を変化させる素子名は ? e2
para e2 最低値 ? 1
para e2 最高値 ? 5
para e2 ステップ ? 1
e2[1] [ ] f[0] [ ] ] abs[ 0] arg[ 0]
x3 [ 0 j 0 ] ]
e2[2] [ ] f[0] [ ] ] abs[ 2] arg[ 0]
x3 [ 2 j 0 ] ]
e2[3] [ ] f[0] [ ] ] abs[ 4] arg[ 0]
x3 [ 4 j 0 ] ]
e2[4] [ ] f[0] [ ] ] abs[ 6] arg[ 0]
x3 [ 6 j 0 ] ]
e2[5] [ ] f[0] [ ] ] abs[ 8] arg[ 0]
x3 [ 8 j 0 ] ]
```

今度は、e1 を 1V から 5V まで変化させてみます。

```
? /para
値を変化させる素子名は ? e1
para e1 最低値 ? 1
para e1 最高値 ? 5
para e1 ステップ ? 1
e1[1] [ ] f[0] [ ] ] abs[ 0] arg[ 0]
x3 [ 0 j 0 ] ]
e1[2] [ ] f[0] [ ] ] abs[ 2] arg[ 180]
x3 [ -2 j 0 ] ]
e1[3] [ ] f[0] [ ] ] abs[ 4] arg[ 180]
x3 [ -4 j 0 ] ]
e1[4] [ ] f[0] [ ] ] abs[ 6] arg[ 180]
x3 [ -6 j 0 ] ]
e1[5] [ ] f[0] [ ] ] abs[ 8] arg[ 180]
x3 [ -8 j 0 ] ]
```

オペアンプによる差動増幅回路では 2 つの抵抗の比だけでゲインを正確に決定できます。

ゲインの誤差は、差動増幅回路のゲインとオペアンプのゲインの比で決まります。

オペアンプでは 1000 万倍程度のゲインが期待できますから、差動増幅回路のゲインを 1000 倍に設定しても、ゲインの誤差は 1 万分の 1(0.01%)程度になります。

微小な信号を正確に増幅して分析することができます。

e1=0, e2=0.001V, r1=1k, r2=1meg に設定しゲインを 1000 倍として、ic1 のゲイン v1b1 を 10meg に変更してゲインの誤差を確認します。

サンプルバッチファイルの回路図と説明

```
? /point
f[0
x3 [ 0.9999 j 0 ] abs[ 0.9999] arg[ 0]
? a=x3/e2
999.8997 j 0
? err=(a-1000)/1000
-1.002899e-004 j 0
```

ゲインの誤差が-0.01%であることが確認できます。

ic1 のゲインが v1b1=1meg の時の誤差は、-0.1%に増加します。

```
? v1b1=1meg
1,000,000 j 0
? /point
f[0
x3 [ 0.9990 j 0 ] abs[ 0.9990] arg[ 0]
? a=x3/e2
998.9980 j 0
? err=(a-1000)/1000
-0.0010 j 0
```

Maple を利用する

/conv で係数行列 difamp.coe を作成して、Maple にコピーアンドペーストして伝達関数を計算します。

```
restart: with(linalg): with(plots):↓
siki:=matrix(5,5,0):e:=vector(5,0):↓
siki[ 1, 1] := 1 ;↓
e[1] := +e1 ;↓
siki[ 2, 2] := 1 ;↓
e[2] := +e2 ;↓
siki[ 3, 3] := 1 ;↓
siki[ 3, 4] := +v1b1 ;↓
siki[ 3, 5] := -v1b1 ;↓
siki[ 4, 1] := -1/r1 ;↓
siki[ 4, 3] := -1/r2 ;↓
siki[ 4, 4] := +1/r1+1/r2+1/r1b1 ;↓
siki[ 4, 5] := -1/r1b1 ;↓
siki[ 5, 2] := -1/r1 ;↓
siki[ 5, 4] := -1/r1b1 ;↓
siki[ 5, 5] := +1/r2+1/r1+1/r1b1 ;↓
x:=linsolve(siki,e):↓
↓
e1 := 1 ;↓
e2 := 1 ;↓
r2 := 100000 ;↓
r1 := 50000 ;↓
v1b1 := 1e+006 ;↓
r1b1 := 1e+006 ;↓
↓
s:=I*2*Pi*f;↓
↓
↓
;end;↓
```

(9) バッチファイル名 s s 2. s b で扱う回路図
Maple を利用する

Maple にコピーアンドペーストして、方程式を解きます。

```

difamp.mws
> restart: with(linalg): with(plots):
siki:=matrix(5,5,0):e:=vector(5,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 2] := 1 ;
e[2] := +e2 ;
siki[ 3, 3] := 1 ;
siki[ 3, 4] := +v1b1 ;
siki[ 3, 5] := -v1b1 ;
siki[ 4, 1] := -1/r1 ;
siki[ 4, 3] := -1/r2 ;
siki[ 4, 4] := +1/r1+1/r2+1/r1b1 ;
siki[ 4, 5] := -1/r1b1 ;
siki[ 5, 2] := -1/r1 ;
siki[ 5, 4] := -1/r1b1 ;
siki[ 5, 5] := +1/r2+1/r1+1/r1b1 ;
x:=linsolve(siki,e);

> x3:=x[3];

$$x3 := - \frac{(e1 - e2) r1b1 v1b1 r2}{2 r1 r2 + r1 r1b1 + r1 r1b1 v1b1 + r2 r1b1}$$

> x3:=limit(x3, v1b1=infinity);

$$x3 := - \frac{(e1 - e2) r2}{r1}$$


```

「差動増幅回路」の冒頭で唐突に出力 $x3$ の数式を示しましたが、係数行列から回路方程式を解くことで、出力 $x3$ が次式で表されることが確認できました。

$$x3 = \frac{R2}{R1} \cdot (e2 - e1)$$

出力 $x3$ は 2 つの入力信号 $e1$ と $e2$ の差($e2 - e1$) に 2 つの抵抗の比 $\frac{R2}{R1}$ を掛けた値になります。2 つの信号の差を増幅するので、「差動増幅回路」と呼ばれています。

(10) バッチファイル s20. sb で扱う3つの回路図

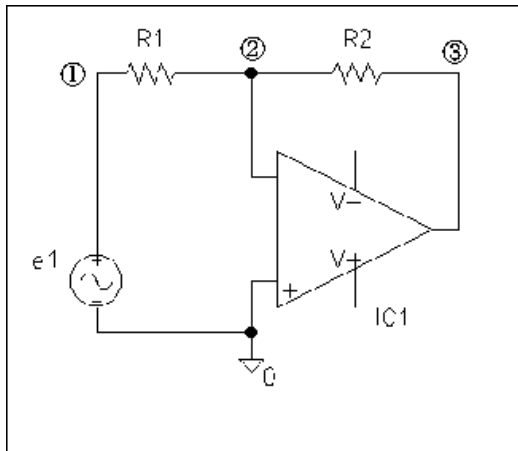


図 s20_1 反転増幅回路

設計の指針

1. 理想オペアンプでは、

- a. 入力インピーダンスは無限大
- b. ゲインも無限大
- c. 周波数特性も無限大

の特性を仮定している。

しかし、実際に市販されているオペアンプではそれらはすべて有限の値となっているため理想の性能からは誤差が生じる。

2. 理想オペアンプでは、a. と b. の仮定からノード2とノード0の電位は等しくなる。

このためノード2はグラウンドのレベルと等しくなるため、仮想グラウンドと呼ばれる。

3. また、この仮定からR1に流れる電流とR2に流れる電流が等しくなる。従って、ノード1とノード3の位相は逆になり、ゲインは $-R2/R1$ となる。

4. 試しに、 $e1 = 2$ とすれば、ノード3の電圧は $-20V$ になるはずである。

5. この等価回路で使用しているオペアンプの入力インピーダンスが $1M$ であり、ゲインが 1000000 のために、ノード2は $0V$ にならず、 $-10\mu V$ 程度になっている。

また、ゲインも抵抗の比率に比べて $1/1000000$ の誤差を含んでいる。

6. 試しに、オペアンプのゲインを 1000 と 1000000 に変更してみると、

ゲイン 1000 の時、ノード2は $-0.0101V$ 、ゲインは $1/88.5$ の誤差、

ゲイン 1000000 の時は、ノード2は $-0.1mV$ 、ゲインは $1/9091$ の誤差となる。

ゲインを 1000000 に戻して、

7. 試しに、オペアンプの入力インピーダンスを $100K$ と $10M$ に変更してみると、

(10) バッチファイル s20. sb で扱う3つの回路図

サンプルバッチファイルの回路図と説明

100Kの時も10Mの時も、1Mの場合と変わらないので、誤差はゲインによって大きく左右されるといっても良い。

8. 反転回路の入力インピーダンスはR1の値そのものである。

```
B? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0          ]
left[ 0] right[ 3] parts[ v1b1        ] value[    1e+006 j 0          ]
@ master[r1b1      ] left[ 2] right[ 0]
left[ 1] right[ 2] parts[ r1          ] value[     1000 j 0          ]
left[ 2] right[ 0] parts[ r1b1        ] value[    1e+006 j 0          ]
left[ 2] right[ 3] parts[ r2          ] value[    1e+004 j 0          ]
最大の素子番号 = 5
最大のノード番号 = 3
B? 最初に直流におけるゲインを確認します。
```

```
B? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 0
range 周波数 最高値 ? 0
range 周波数 ステップ ? 1
e1[1          ] j 0          ]
f[0          ]
x1 [      1 j 0          ] abs[      1] arg[      0]
x2 [-1.000011e-005 j 0          ] abs[1.000011e-005] arg[      180]
x3 [-10.0001 j 0          ] abs[ 10.0001] arg[     -180]
B? ノード2の電圧はほとんど 0V となっています。
オペアンプの非反転入力ノードがグランドレベルの時に、
オペアンプの反転入力ノードのことを、仮想グランドと呼ぶことがあります。

また、ノード3の電圧はほとんど -10V となっています。
この値は、-R2/R1 の比率に等しくなっています。

試しに、e1=2 とすれば、ノード3は -20 となるはずです。
```

```
e1[2          ] j 0          ]
f[0          ]
x1 [      2 j 0          ] abs[      2] arg[      0]
x2 [-2.000022e-005 j 0          ] abs[2.000022e-005] arg[      180]
x3 [-20.0002 j 0          ] abs[ 20.0002] arg[     -180]
B? 予想どおり、-20V となっています。
```

サンプルバッチファイルの回路図と説明

B? 入力インピーダンスは、 $Z_{in} = e1 / (R1 \text{ に流れる電流})$ で求められます。
e1=2

2 j 0

B? $i1 = (x1 - x2) / r1$
0.0020 j 0

B? $z_{in} = e1 / i1$
999.9900 j 0

B? 入力インピーダンスはほとんど、 $R1$ と等しくなっています。

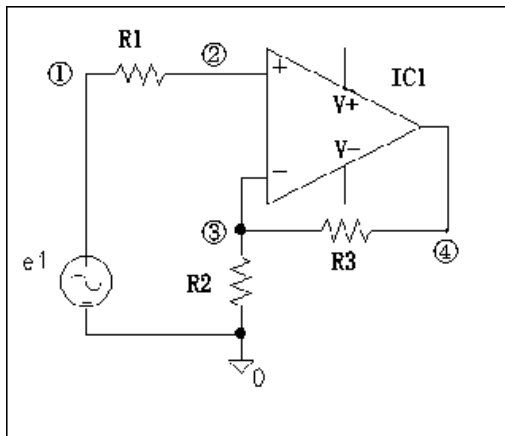


図 s20_2 非反転増幅回路

設計の指針

1. 非反転回路では、理想オペアンプの仮定 a. と b. からノード2とノード3及びノード1の電位は等しくなる。
2. 従って、ノード4の電圧はノード1の電圧に $R3 / R2$ を掛けた値にノード1の電圧を加えた値に等しくなる。よって非反転回路のゲインは、 $(1 + R3 / R2)$ となる。
3. 非反転回路の入力インピーダンス Z_i を求める。

$Z_i = e1 / (R1 \text{ に流れる電流値})$

$R1 \text{ に流れる電流値} = (\text{ノード1の電位} - \text{ノード2の電位}) / R1 = 1 \text{ pA}$

より、 $Z_i = 90000 \text{ M}$ となり、非常に入力インピーダンスが高いことが分かる。

サンプルバッチファイルの回路図と説明

```

B? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0      ]
left[ 0] right[ 3] parts[ r2          ] value[     1000 j 0      ]
left[ 0] right[ 4] parts[ v1b1        ] value[    1e+006 j 0      ]
@ master[r1b1      ] left[ 3] right[ 2]
left[ 1] right[ 2] parts[ r1          ] value[    1e+006 j 0      ]
left[ 3] right[ 2] parts[ r1b1        ] value[    1e+006 j 0      ]
left[ 3] right[ 4] parts[ r3          ] value[    1e+004 j 0      ]
最大の素子番号 = 6
最大のノード番号 = 4
B? 先程と同様に、直流におけるゲインを調べます。

```

```

e1[1          j 0          ]
f[0          ]
x1 [          1 j 0          ] abs[          1] arg[          0]
x2 [          1 j 0          ] abs[          1] arg[          0]
x3 [          1 j 0          ] abs[          1] arg[          0]
x4 [ 11.0002 j 0          ] abs[ 11.0002] arg[          0]
B? 今度は、ノード1、2及び3が全て同じ電圧となりました。
これは、入力インピーダンスが十分高くて R1 にはほとんど
電流が流れないことを表わしています。

```

B? また、オペアンプの電圧増幅率が非常に高いために
ノード2と3の電圧がほとんど同じ電圧になっています。

ノード4の電圧は 11V となっていますがこれは
(R2+R3)/R2 の比率と等しくなっています。

入力インピーダンスは $Z_{in} = e1 / (R1 \text{ に流れる電流})$ で求められます。
 $i1 = \text{abs}(x1 - x2) / r1$
 1.100024e-011 j 0

B? $z_{in} = e1 / i1$
 90,907,089,999.2307 j 0

B? 入力インピーダンスは約90,000M オームと求められました。

s20_1 の R1 または R2 にコンデンサやインダクタを組み合わせると
周波数によって、ゲインが変化する回路を構成することができます。
 s20_2 では、R2 または R3 にコンデンサやインダクタを組み合わせる
事で、周波数によって、ゲインが変化する回路を構成することができます。

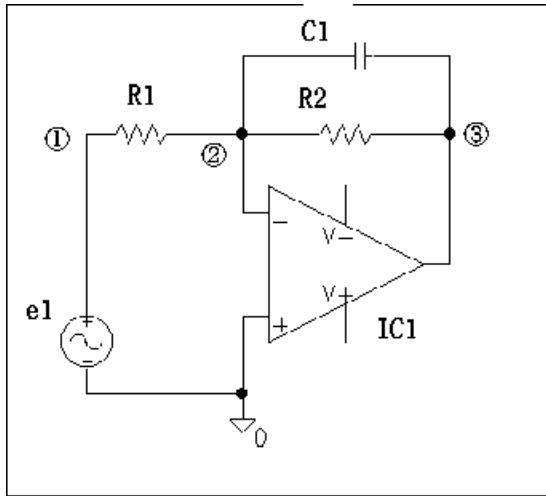


図 s20_3 ローパスフィルタ

設計の指針

1. 図 20-1 の R 1 または R 2 にコンデンサやインダクタを組み合わせるとゲインが周波数によって変化ようになる。
2. 図 20-2 では、R 2 または R 3 にコンデンサやインダクタを組み合わせるとゲインが周波数によって変化ようになる。
3. 図 20-3 は一つの例として、R 2 と並列にコンデンサを接続した回路である。この回路ではゲインは $-(C 1 \text{ と } R 2 \text{ の並列インピーダンス}) / R 1$ と表される。従って、周波数が高くなるに従ってゲインは小さくなり、一般的にはローパスフィルタと呼ばれる特性となる。

周波数特性を確認するには、/mk の出力ファイルを確認する。

4. 角周波数を w とすると、ゲイン G は、

$$G = -\frac{R_2}{R_1} \frac{1}{1 + jwC_1R_2}, \quad \therefore |G|_w = \frac{R_2}{R_1} \frac{1}{\sqrt{1 + (wC_1R_2)^2}}$$

と表される。周波数が 0 (直流) のときのゲインを $G_0 = \frac{R_2}{R_1}$ とすると、

$w_c C_1 R_2 = 1$ の時に、 $|G|_{w_c} = \frac{G_0}{\sqrt{2}} \rightarrow -3dB$ となる。

このことから、 $w_c = \frac{1}{C_1 R_2} \rightarrow f_c = \frac{1}{2\pi C_1 R_2}$ をカットオフ周波数と呼ぶ。

サンプルバッチファイルの回路図と説明

```

B? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0      ]
left[ 0] right[ 3] parts[ v1b1        ] value[ 1e+006 j 0      ]
@ master[rlb1      ] left[ 2] right[ 0]
left[ 1] right[ 2] parts[ r1          ] value[     1000 j 0      ]
left[ 2] right[ 0] parts[ r1b1        ] value[ 1e+006 j 0      ]
left[ 2] right[ 3] parts[ c1          ] value[ 1e-007 j 0      ]
left[ 2] right[ 3] parts[ r2          ] value[ 1e+004 j 0      ]
最大の素子番号 = 6
最大のノード番号 = 3

```

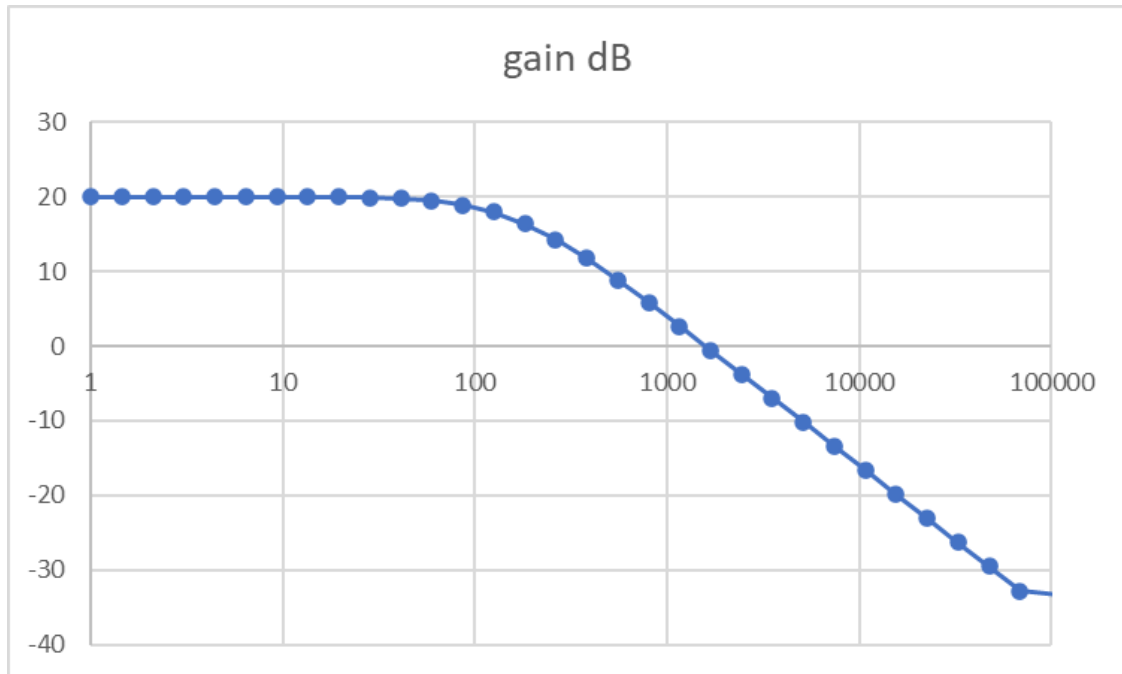
```

range 周波数 最低値 ? 1
range 周波数 最高値 ? 100k
range 周波数 ステップ ? 1
e1[1          ]
f[1          ]
x3 [ -9.9997 j 0.0628 ] DB[ 19.9999] arg[ 179.6400]
f[1.4497      ]
x3 [ -9.9993 j 0.0911 ] DB[ 19.9997] arg[ 179.4781]
f[2.1017      ]
x3 [ -9.9984 j 0.1320 ] DB[ 19.9993] arg[ 179.2434]
f[3.0470      ]
x3 [ -9.9964 j 0.1914 ] DB[ 19.9985] arg[ 178.9032]
f[4.4173      ]
x3 [ -9.9924 j 0.2773 ] DB[ 19.9968] arg[ 178.4101]
f[6.4040      ]
x3 [ -9.9839 j 0.4017 ] DB[ 19.9931] arg[ 177.6958]
f[9.2841      ]
x3 [ -9.9662 j 0.5814 ] DB[ 19.9853] arg[ 176.6615]
f[13.4596     ]
x3 [ -9.9291 j 0.8397 ] DB[ 19.9691] arg[ 175.1660]
f[19.5129     ]
x3 [ -9.8520 j 1.2079 ] DB[ 19.9353] arg[ 173.0102]
f[28.2887     ]
x3 [ -9.6938 j 1.7230 ] DB[ 19.8650] arg[ 169.9212]
f[41.0113     ]
x3 [ -9.3774 j 2.4164 ] DB[ 19.7209] arg[ 165.5501]
f[59.4557     ]
x3 [ -8.7754 j 3.2783 ] DB[ 19.4327] arg[ 159.5156]
f[86.1954     ]
x3 [ -7.7321 j 4.1876 ] DB[ 18.8830] arg[ 151.5606]
f[124.9609    ]
x3 [ -6.1864 j 4.8573 ] DB[ 17.9144] arg[ 141.8624]
f[181.1609    ]
x3 [ -4.3561 j 4.9584 ] DB[ 16.3910] arg[ 131.2999]
f[262.6364    ]
x3 [ -2.6859 j 4.4323 ] DB[ 14.2909] arg[ 121.2152]
f[380.7546    ]
x3 [ -1.4873 j 3.5583 ] DB[ 11.7242] arg[ 112.6847]
f[551.9954    ]
x3 [ -0.7675 j 2.6620 ] DB[ 8.8509] arg[ 106.0835]
f[800.2502    ]
x3 [ -0.3805 j 1.9131 ] DB[ 5.8034] arg[ 101.2482]
f[1,160.1553  ]
x3 [ -0.1847 j 1.3465 ] DB[ 2.6651] arg[ 97.8112]
f[1,681.9243  ]
x3 [ -0.0887 j 0.9379 ] DB[ -0.5184] arg[ 95.4056]
f[2,438.3541  ]
x3 [ -0.0424 j 0.6499 ] DB[ -3.7240] arg[ 93.7344]
f[3,534.9811  ]
x3 [ -0.0202 j 0.4493 ] DB[ -6.9401] arg[ 92.5779]
f[5,124.8059  ]
x3 [ -0.0096 j 0.3103 ] DB[ -10.1613] arg[ 91.7788]
f[7,429.6395  ]
x3 [ -0.0046 j 0.2141 ] DB[ -13.3849] arg[ 91.2272]
f[10,771.0506 ]
x3 [ -0.0022 j 0.1477 ] DB[ -16.6097] arg[ 90.8465]
f[15,615.2301 ]
x3 [ -0.0010 j 0.1019 ] DB[ -19.8350] arg[ 90.5839]
f[22,638.0341 ]
x3 [ -4.942395e-004 j 0.0703 ] DB[ -23.0606] arg[ 90.4028]
f[32,819.2787 ]
x3 [ -2.351625e-004 j 0.0485 ] DB[ -26.2863] arg[ 90.2778]
f[47,579.4431 ]
x3 [ -1.118904e-004 j 0.0335 ] DB[ -29.5120] arg[ 90.1917]
f[68,977.8538 ]
x3 [ -5.323714e-005 j 0.0231 ] DB[ -32.7378] arg[ 90.1322]
f[100,000     ]
x3 [ -2.533000e-005 j 0.0159 ] DB[ -35.9636] arg[ 90.0912]
B? ある周波数を起点にゲインが減衰するローパスフィルタ
の一般的な周波数特性を示しています。

```

サンプルバッチファイルの回路図と説明

上の周波数とゲインのデータを excel を使ってグラフを作成した。

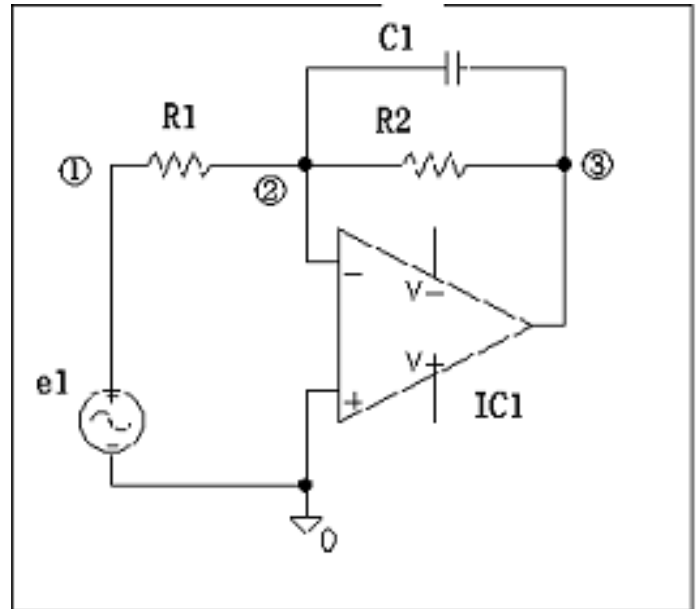


低い周波数でのゲインは 20dB 程度で一定だが、100Hz を越えるあたりからゲインが低下してローパスフィルタ特性を示している。

Maple を使って、s20.sb の係数方程式 s20.coe を解く。

s20.coe の内容。

```
restart: with(linalg):
siki:=matrix(3,3,0):e:=vector(3,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 1] := -1/r1 ;
siki[ 2, 2] := +1/r1+1/r1b1+s*c1+1/r2 ;
siki[ 2, 3] := -s*c1-1/r2 ;
siki[ 3, 2] := -v1b1 ;
siki[ 3, 3] := 1 ;
x:=linsolve(siki,e);
```



```
e1 := 1 ;
r1 := 1000 ;
r2 := 10000 ;
c1 := 1e-007 ;
v1b1 := 1e+006 ;
r1b1 := 1e+006 ;
x3 := -2.533e-005 ;
```

```
;end;
```

変換終了

使用した式の個数 7、 密度 77.778 %

使用した式のサイズ 133

Maple にコピーアンドペーストして、方程式を解きます。

```

ワークシート(1)
restart: with(linalg):
siki:=matrix(3,3,0):e:=vector(3,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 1] := -1/r1 ;
siki[ 2, 2] := +1/r1+1/r1b1+s*c1+1/r2 ;
siki[ 2, 3] := -s*c1-1/r2 ;
siki[ 3, 2] := -v1b1 ;
siki[ 3, 3] := 1 ;
x:=linsolve(siki,e);
Warning, new definition for norm
Warning, new definition for trace


$$siki_{1,1} := 1$$


$$e_1 := e1$$


$$siki_{2,1} := -\frac{1}{r1}$$


$$siki_{2,2} := \frac{1}{r1} + \frac{1}{r1b1} + s \cdot c1 + \frac{1}{r2}$$


$$siki_{2,3} := -s \cdot c1 - \frac{1}{r2}$$


$$siki_{3,2} := -v1b1$$


$$siki_{3,3} := 1$$


$$x := \left[ e1, \right.$$


$$- \frac{e1 \cdot r1b1 \cdot r2}{-r1b1 \cdot r2 - r1 \cdot r2 - s \cdot c1 \cdot r1 \cdot r1b1 \cdot r2 - r1 \cdot r1b1 + v1b1 \cdot r1 \cdot r1b1 \cdot s \cdot c1 \cdot r2 + v1b1 \cdot r1 \cdot r1b1},$$


$$\left. - \frac{v1b1 \cdot e1 \cdot r1b1 \cdot r2}{-r1b1 \cdot r2 - r1 \cdot r2 - s \cdot c1 \cdot r1 \cdot r1b1 \cdot r2 - r1 \cdot r1b1 + v1b1 \cdot r1 \cdot r1b1 \cdot s \cdot c1 \cdot r2 + v1b1 \cdot r1 \cdot r1b1} \right]$$


```

ノード 3 の値を e1 で割って、ゲイン g を求めます。

```

> x3:=x[3]:g:=x3/e1;
g:=

$$- \frac{v1b1 \cdot r1b1 \cdot r2}{-r1b1 \cdot r2 - r1 \cdot r2 - s \cdot c1 \cdot r1 \cdot r1b1 \cdot r2 - r1 \cdot r1b1 + v1b1 \cdot r1 \cdot r1b1 \cdot s \cdot c1 \cdot r2 + v1b1 \cdot r1 \cdot r1b1}$$


```

(10) バッチファイル s20. sb で扱う 3つの回路図

Maple を使って、s20.sb の係数方程式 s20.coe を解く。

サンプルバッチファイルの回路図と説明

ic1 のゲイン v1b1 を無限大に近付けて、ゲイン g を近似します。

g の分子 gn と分母 gd をそれぞれ v1b1 で割って、v1b1 を無限大に近付けます。

割った結果の分子を gn、分母を gd とします。

最終的なゲイン gain を gn / gd によって求めます。

```
> gn:=limit(numer(g)/v1b1, v1b1=infinity);
                                gn := -r1b1 r2
> gd:=limit(denom(g)/v1b1, v1b1=infinity);
                                gd := s c1 r1 r1b1 r2 + r1 r1b1
> gain:=simplify(gn/gd);
                                gain := -  $\frac{r2}{r1(s c1 r2 + 1)}$ 
```

ゲイン gain を s を含む因子と含まない因子に分けると、次のように表すことができます。

$$gain = -\frac{r2}{r1} \cdot \frac{1}{1 + s \cdot c1 \cdot r2}$$

s はラプラス演算子と呼ばれる記号で、周波数を f とすると次のように表されます。

$$s = j \cdot 2 \cdot \pi \cdot f$$

ここで、j は虚数単位で $j = \sqrt{-1}$ です。

e1, r1, r2, c1 に回路図の素子値を代入します。

```
> e1 := 1 ;
  r1 := 1000 ;
  r2 := 10000 ;
  c1 := 1e-007 ;
  v1b1 := 1e+006 ;
  r1b1 := 1e+006 ;
  s:=i*2*pi*f;
  pi:=3.141592;

> gain;
                                -10  $\frac{1}{.006283184000 / f + 1}$ 
```

gain の分子は -10 となり、直流に対するゲインを表します。

gain の分母は交流に対するゲインの変化を表します。

gain の分母を gd として取り出します。

(10) バッチファイル s20. sb で扱う3つの回路図

Maple を使って、s20.sb の係数方程式 s20.coe を解く。

```
[> gd:=denom(gain);                                gd:=.006283184000 / f+1
```

$$gd = 1 + j 0.006283 \cdot f = 1 + j \frac{f}{159.16}$$

回路のゲインは **gain** の絶対値で次のように表されます。

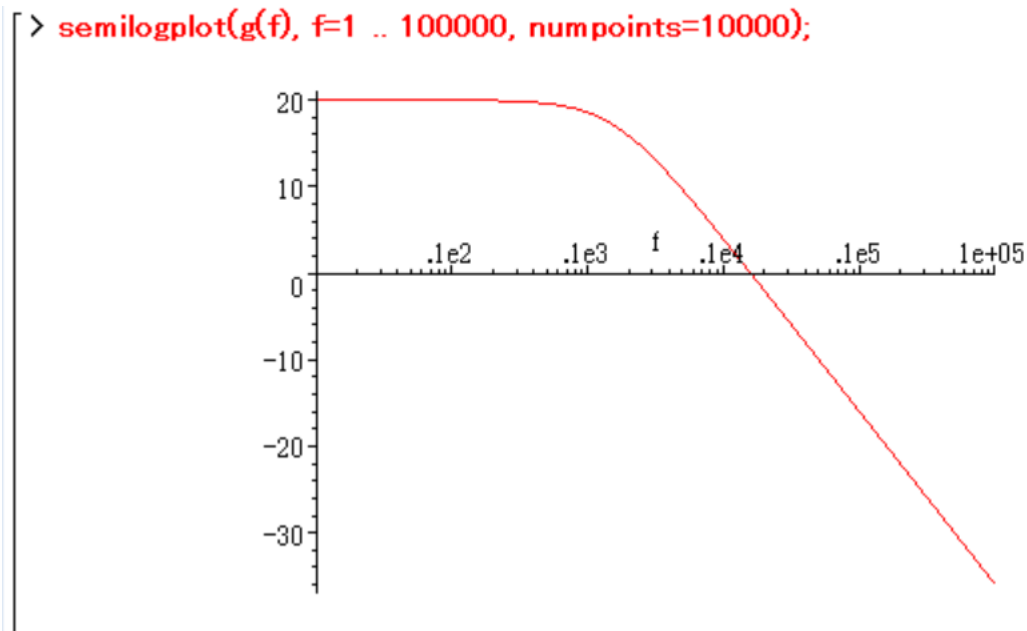
$$G(f) = |gain| = -\frac{10}{\sqrt{1 + \left(\frac{f}{159.16}\right)^2}}$$

この数式から、 $f=0$ ではゲインは-10（位相反転で 10 倍）、 $f=159.16$ で $-10/\sqrt{2}$ に低下して、その後ゲインが低下し続けることが分かります。

ゲインの関数を作成します。

```
[> g:= f -> 20*log10(10*1/(sqrt(1+(f/159.1549762)^2)));
    g:= f -> 20 log10(10 * 1 / sqrt(1 + .00003947840118 f^2))
```

ゲインの関数が求められたので、 $f=1$ Hz から 100 KHz までのグラフを作成します。



サンプル回路全体を順に実行するバッチファイル ALL.sb

コマンドラインから `@all` を入力すると、
多数のバッチファイルが順に呼び出されて、
様々な回路のシミュレーションが実行されます。

`@all` は`@all1`、`@all2`、`@all3` を順に実行します。

`@all1` は、`@s1`、`@s2`、`@s3`、`@s4`、`@s5`、`@s6`、`@s7`、`@s8` を実行します。

`@all2` は、`@s9`、`@s10`、`@s11`、`@ss1`、`@ss2` を実行します。

`@all3` は、`@s20`、`@s21` を実行します。

シミュレーションでキー入力待ちの時に、`/skip` と入力するとその演習を終了して次に進みます。

シミュレーションでキー入力待ちの時に、`/rewind` と入力するとそのシミュレーションを最初からやり直しが出来ます。

シミュレーションでキー入力待ちの時に、`/exit` と入力するとバッチファイル `ALL.sb` を強制終了します。

バッチファイル `ALL1.sb`、`ALL2.sb`、`ALL3.sb` を単独で実行することもできます。

サンプルバッチファイルの回路図と説明

Maple の利用方法

/conv によって保存した係数行列を Maple にコピーアンドペーストすると、文字式のままで回路の伝達関数を求めたり、その伝達関数に回路の素子値を代入して周波数特性の数式に変えて、グラフを作成することなどが出来ます。

```
# Maple用 係数行列・素子値リスト↓
restart: with(linalg): with(plots):↓
siki:=matrix(3,3,0):e:=vector(3,0):↓
siki[ 1, 1] := 1 ;↓
e[1] := +e1 ;↓
siki[ 2, 1] := -1/r1 ;↓
siki[ 2, 2] := +1/r1+1/r1b1+s*c1+1/r2 ;↓
siki[ 2, 3] := -s*c1-1/r2 ;↓
siki[ 3, 2] := -v1b1 ;↓
siki[ 3, 3] := 1 ;↓
x:=linsolve(siki,e);↓
↓
e1 := 1 ;↓
r1 := 1000 ;↓
r2 := 10000 ;↓
c1 := 1e-007 ;↓
v1b1 := 1e+006 ;↓
r1b1 := 1e+006 ;↓
x3 := -10.0001 ;↓
↓
s:=I*2*Pi*f;↓
#/point, /para 用↓
#f:=0;↓
#/ac 用↓
#f:=1000;↓
#/range, /ac では、↓
#e1以外のすべての電源を 0 に設定すること↓
```

s20.coe (2020/06/04) の内容

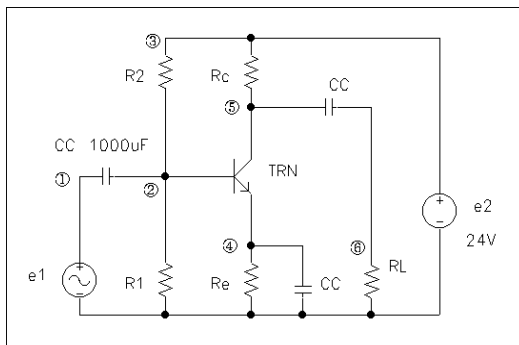


図 ex4

```
# Maple用 係数行列・素子値リスト↓
restart: with(linalg): with(plots):↓
siki:=matrix(10,10,0):e:=vector(10,0):↓
siki[ 1, 1] := 1 ;↓
e[1] := +e1 ;↓
siki[ 2, 1] := -s*cc ;↓
siki[ 2, 2] := +1/r1+s*cc+1/r2+1/r1b1 ;↓
siki[ 2, 3] := -1/r2 ;↓
siki[ 2, 10] := -1/r1b1 ;↓
siki[ 3, 3] := 1 ;↓
e[3] := +e2 ;↓
siki[ 4, 4] := +s*ce+1/re+1/rbb1 ;↓
siki[ 4, 7] := -1/rbb1 ;↓
siki[ 5, 3] := -1/rc ;↓
siki[ 5, 5] := +1/rc+s*cc+1/rbb1 ;↓
siki[ 5, 6] := -s*cc ;↓
siki[ 5, 9] := -1/rbb1 ;↓
siki[ 6, 5] := -s*cc ;↓
siki[ 6, 6] := +1/r1+s*cc ;↓
siki[ 7, 4] := -1/rbb1 ;↓
siki[ 7, 7] := +1/rbb1+1/reb1+kb1*1/reb1 ;↓
siki[ 7, 8] := -1/reb1-kb1*1/reb1 ;↓
siki[ 8, 2] := -1/r1b1 ;↓
siki[ 8, 7] := -1/reb1 ;↓
siki[ 8, 8] := +1/reb1 ;↓
siki[ 8, 10] := +1/r1b1 ;↓
siki[ 9, 5] := -1/rbb1 ;↓
siki[ 9, 7] := -kb1*1/reb1 ;↓
siki[ 9, 8] := +kb1*1/reb1 ;↓
siki[ 9, 9] := +1/rbb1 ;↓
siki[ 10, 8] := -1 ;↓
siki[ 10, 10] := 1 ;↓
e[10] := +ebb1 ;↓
x:=linsolve(siki,e);↓
↓
e1 := 1 ;↓
r1 := 10000 ;↓
re := 2200 ;↓
ce := 0.001 ;↓
r1 := 1000 ;↓
e2 := 24 ;↓
cc := 0.001 ;↓
r2 := 19200 ;↓
rc := 3800 ;↓
r1b1 := 300 ;↓
rbb1 := 0.1 ;↓
reb1 := 26 ;↓
kb1 := 50 ;↓
ebb1 := 0.7 ;↓
x5 := 12.0053 ;↓
↓
s:=I*2*Pi*f;↓
#/point, /para 用↓
#f:=0;↓
#/ac 用↓
#f:=1000;↓
#/range, /ac では、↓
#e1以外のすべての電源を 0 に設定すること↓
```

ex4.coe (2020/06/04) の内容

サンプルバッチファイルの回路図と説明

係数行列のファイルには回路の出力ノードの番号が示されないので、/disp で出力ノードを指定して、/point で一度表示してから、/conv を実行すると係数行列の最後に出力ノードが追加されます。上記、ex4.coe の例では、x5 が出力ノードだと分かります。

文字式のままで伝達関数を求める場合は、

restart: with(linalg): with(plots): の行から、

x:=linsolve(siki,e); の行までを Maple にコピーアンドペーストして、

「エンター」キーを押すと、すべてのノードの値が配列 x に求められます。

ノード 5 の値やゲインは、x5:=x[5]; g:=x5/e1; と入力すれば、数式が示されます。

```
> x5:=x[5];
x5 := -((1 + s cc r1) (-r2 re e2 - e2 r1 r1 b1 - re e2 r1 - r2 e2 rbb1 - r2 kb1 rbb1 e2
- r2 re kb1 e2 - rc kb1 r2 ebb1 - re rbb1 kb1 ce s e2 r1 - re s ce rbb1 e2 r1 - re s ce r1 b1 e2 r1
- reb1 re s ce e2 r1 - kb1 rbb1 e2 r1 - re kb1 e2 r1 - s r1 r2 cc kb1 rbb1 e2 - s r1 r2 cc e2 r1 b1
- s^2 r1 r2 cc re ce r1 b1 e2 - s r1 r2 cc re e2 - s^2 r1 r2 cc re rbb1 kb1 ce e2 - s r1 r2 cc e2 reb1
- s^2 r1 r2 cc re ce rbb1 e2 - s r1 r2 cc e2 rbb1 - s^2 r1 r2 cc reb1 re ce e2 - s r1 r2 cc re kb1 e2
- r2 re s ce r1 b1 e2 - r2 re rbb1 kb1 ce s e2 - r2 re s ce rbb1 e2 - r2 reb1 re s ce e2
- rc kb1 s^2 r1 r2 cc re ce ebb1 - rc kb1 r1 ebb1 - rc kb1 re s ce r1 ebb1 - rc kb1 re s ce r2 ebb1
- rc kb1 s r1 r2 cc ebb1 + rc cc kb1 s^2 e1 r2 re ce r1 + rc cc kb1 s e1 r2 r1 - r2 e2 re s ce r1
+ rc kb1 e2 re s ce r1 + rc kb1 e2 r1 - r2 e2 reb1 - r2 e2 r1 - rbb1 e2 r1 - r2 e2 r1 b1
- reb1 e2 r1)) / ((s cc r1 + rc s cc + 1) (r1 r2 re s cc + r2 s ce re rbb1 + re r2 + re r1
+ kb1 re r2 + kb1 re r1 + rbb1 r2 + r1 rbb1 + reb1 r2 + r1 reb1 + r2 rbb1 kb1 ce re s
+ r2 reb1 s ce re + kb1 rbb1 r1 + re rbb1 kb1 ce s^2 cc r1 r2 + re s ce rbb1 r1 + re s ce r1 b1 r2
+ re s ce r1 r2 + re kb1 s cc r1 r2 + re s^2 ce rbb1 cc r1 r2 + re s ce r1 b1 r1
+ reb1 re s^2 ce cc r1 r2 + reb1 re s ce r1 + reb1 s cc r1 r2 + re s^2 ce r1 b1 cc r1 r2
+ re rbb1 kb1 ce s r1 + rbb1 s cc r1 r2 + kb1 rbb1 s cc r1 r2 + r2 rbb1 kb1 + s cc r1 r2 r1 b1
+ r2 r1 b1 + r1 r1 b1 + r1 r2))
```

この回路の場合は文字式では複雑すぎて、良く分かりません。しかし、シンプルな回路の場合には、文字式の方が回路の特徴が分かりやすいことがあります。

回路定数を代入して、数値でノードの値やゲインを求めたいなら、

e1 := 1; の行から、s:=I*2*Pi*f; の行までを追加でコピーアンドペーストして、

x5 := 12.0053; の行の先頭に「#」を追加してから、

x5; または g;

と入力すれば、数値で示されます。

```
> x5;
- (1+2.000 I π f) (-4175339488 1011 - .828809626 1011 π2 f2 - .8529426119 1012 I π f)
- (9.600 I π f+1) (-5594265600 109 π2 f2 + .3477908120 1010 + .4409928213 1011 I π f)
```

`g:=x5/e1;` ですが、`e1:=1;` なので、ゲイン `g` も同じ式で表されます。

素子の値を代入すると、数式が大分簡略化されます。

Maple で行の先頭に「#」を入力すると、その行は「コメント行」になります。

この回路のように、キャパシターやインダクターが使用されている場合には、ノード値の数式などには I (虚数単位) を含む数式が示されます。

係数行列を Maple にコピーアンドペーストしてから、内容を少し変更して実行すると、`sim.exe` の `/para`, `/point`, `/range`, `/ac` の処理を文字式や数値で確認できます。

例えば、`ex4.coe` で、`sim.exe` の `/para` のように、抵抗 `R2` を変化させた時の回路の直流動作点の変化を確認したいならば、

`restart: with(linalg): with(plots):` の行から、

`#f:=0;` の行までをコピーアンドペーストしてから、

`#f:=0;` の 「#」 を削除します。

`r2 := 19200;` の行の先頭に 「#」 を追加します。→ `#r2 := 19200;`

さらに、`x5 := 12.0053;` の行の先頭にも 「#」 を追加します。

それから、「エンター」を入力すると、全ノードの値が配列 `x` に計算されます。

`x5:=x[5];`

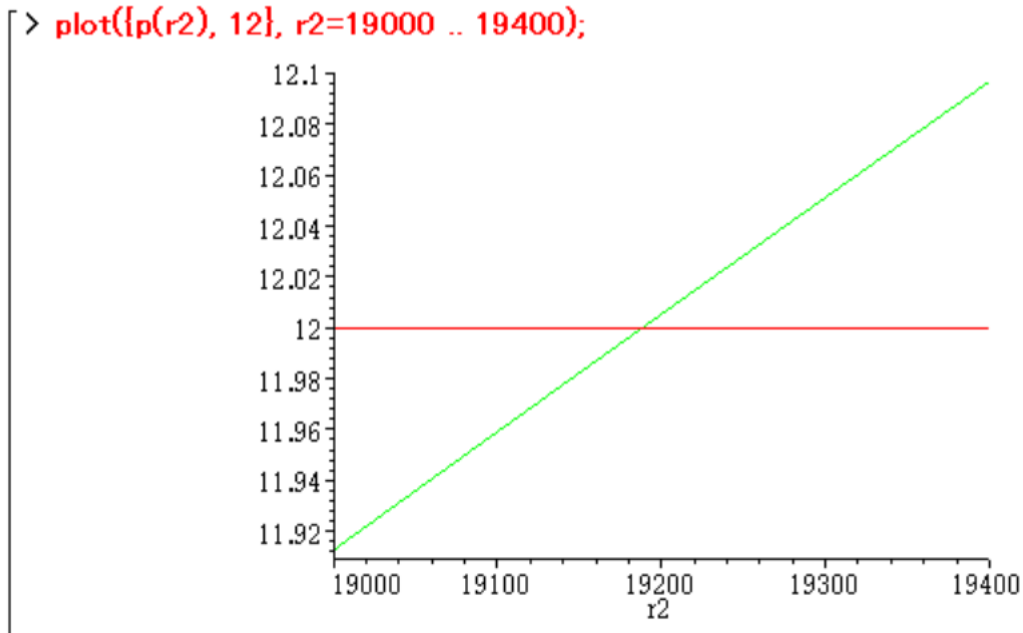
と入力すると、

```
> x5:=x[5];
x5 := (-.1726253600 1011 + .30737464 107 r2) / (.1125311000 1010 + 122531.1 r2)
```

これから、電圧変化を示す関数 `p` を作ります。

```
> p:= r2 -> x5;
p := r2 -> x5
```

`x5` を `r2=19K` から `19.4K` の範囲で確認します。



R2=19.2K で x5=12 になるようです。関数 p で確認します。

```
> subs(r2=19200, p(r2));
```

12.00531855

間違いなく、R2=19.2K で、x5=12 になります。

ある素子の値を変化させて、/para のように直流動作点の変化を確認するには、変化させる素子値の行の先頭に「#」を入力し、f:=0; に変更して「エンター」を入力します。

素子の値を変化せず、/point のように現在の直流動作点を確認するには、ex4.coe から restart: with(linalg): with(plots): の行から、#f:=0; の行までをコピーしてから、#x5 := 12.0053; と f:=0; に変更するだけで「エンター」を入力します。

```
> x5:=x[5];
```

x5:= 12.00531855

サンプルバッチファイルの回路図と説明

交流のゲインを sim.exe の **/range** コマンドのように計算するためには、ex4.coe から
 restart: with(linalg): with(plots): の行から、s:=I*2*Pi*f; の行までをコピーアンドペーストしてから、

素子値のリストで入力信号 e1 以外の独立電源の値を 0 に変更します。

ex4.coe では、e2 := 24; を e2 := 0; に変更し、ebb1 := 0.7; を ebb1 := 0; に変更します。

さらに、x5 := 12.0053; の行の先頭に「#」を入力します。

それから「エンター」を入力します。

次に、出力ノードの値とゲインを求めます。

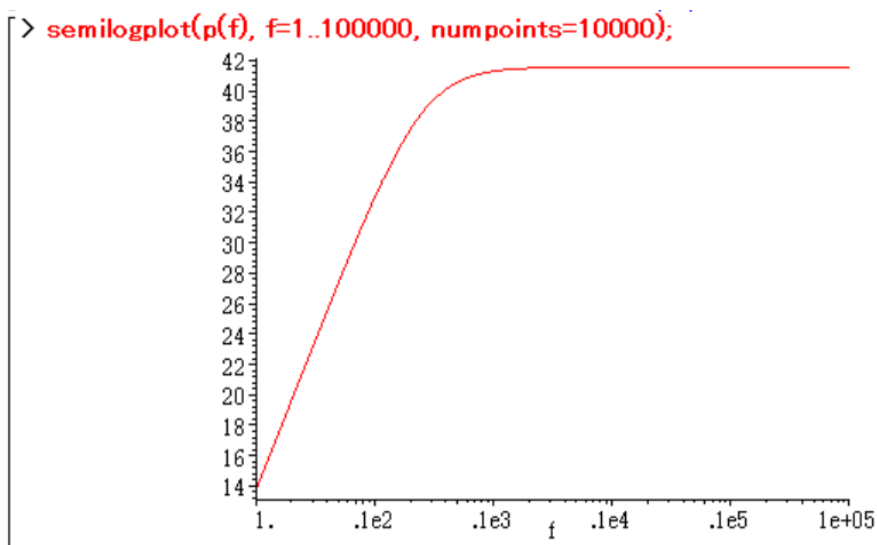
```
> x5:=x[5]; g:=x5/e1;
x5:=((1+2.000 I π f) (3210240000 1012 π2 f2 - .7296000000 1011 I π f)) / (
-4239125349 1012 π2 f2 + .3477908120 1010 - .5370494976 1010 I π3 f3
+.7748720008 1011 I π f)
g:=((1+2.000 I π f) (3210240000 1012 π2 f2 - .7296000000 1011 I π f)) / (
-4239125349 1012 π2 f2 + .3477908120 1010 - .5370494976 1010 I π3 f3
+.7748720008 1011 I π f)
```

g は複素数の数式の分数なので、ゲインは絶対値 abs(g) で計算する必要があります。

ゲインを dB 値で表示するには、gain = 20 · log10(abs(g)) とします。

```
> p:= f -> 20*log10(evalf(abs(g)));
p:= f -> 20 log10(evalf(|g|))
```

f=1 Hz から 100KHz までのゲインをグラフ表示します。



独立電源の表示

- /ipart で回路の入力を終了した時に表示されます。

```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
独立電圧源素子の個数 2
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
```

未解凍の回路ブロックがあると、ブロック内の電源は表示されません。

- /bload で回路ブロックをロードした時に表示されます。

```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.7 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
5
```

- /padd で何も追加しないで、すぐに「*」を入力して終了した時に表示されます。

```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.7 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
5
```

- バッチ @gen を実行した時に表示されます。

```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
i[ 3] j[ 0] 部品名[e2] 値[24 j 0]
i[10] j[ 8] 部品名[ebb1] 値[0.7 j 0]
独立電圧源素子の個数 3
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
5
```

Maple で、/range や/ac の計算をする時には、e1 以外の電源の値を 0 に変更します。

サンプルバッチファイルの回路図と説明

回路の素子値を変化させたときの交流のゲインの変化を sim.exe の **/ac** コマンドのように計算するためには、ex4.coe から

restart: with(linalg): with(plots): の行から、s:=I*2*Pi*f; の行までをコピーアンドペーストしてから、

素子値のリストで入力信号 e1 以外の独立電源の値を 0 に変更します。

ex4.coe では、e2 := 24; を e2 := 0; に変更し、ebb1 := 0.7; を ebb1 := 0; に変更します。さらに、x5 := 12.0053; の行の先頭に「#」を入力します。

それから、変化したい素子値の行の先頭に「#」を入力します。

次に、#f:=1000; の行を、f:=1000; のように変更します。

それから「エンター」を入力します。次に、出力ノードの値とゲインを求めます。

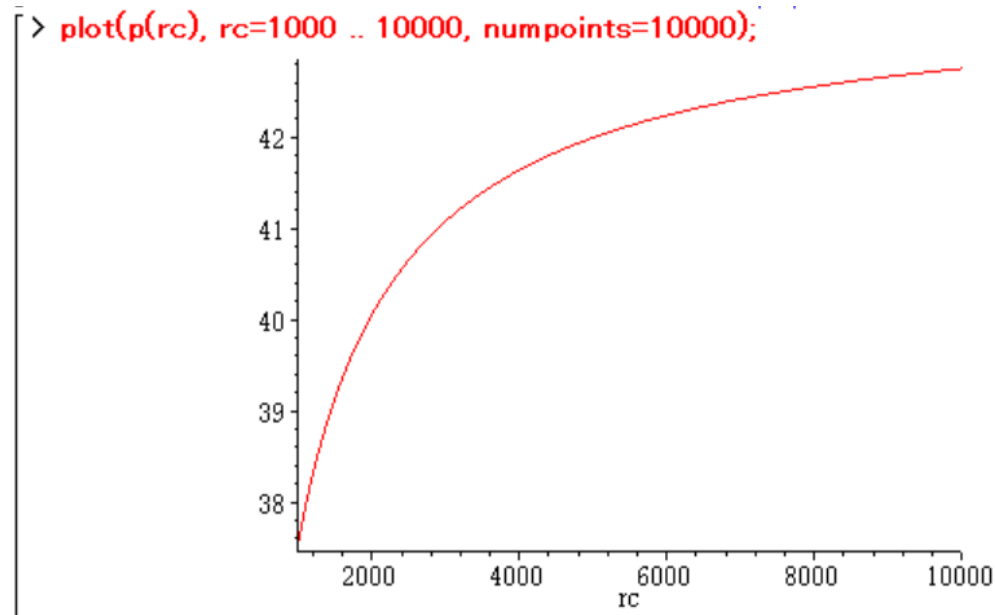
例えば、f=1000 Hz において、Rc を変化させてゲインの変化を確認します。

素子値のリストから、rc := 3800; の行の先頭に「#」を入力します。

それから「エンター」を入力します。次に、出力ノードの値とゲインを求めます。

```
> x5:=x[5]:g:=evalf(x5/e1);
g:=((1.+6283.185308 I)(-6031857896 1011 I rc+ 8337841800 1015 rc))/(-3469144756 1017 I rc-3469130902 1020 I-8760062536 1018
-8704849382 1015 rc)
> p := rc -> 20*log10(evalf(abs(g)));
p := rc -> 20 log10(evalf(|g|))
```

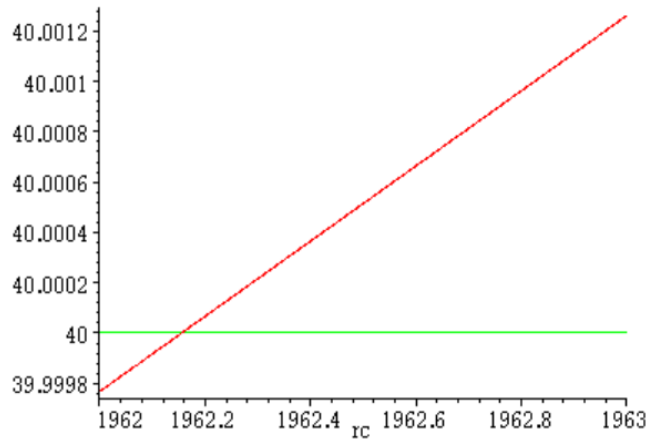
Rc=1K から 10K の範囲でゲインの変化をグラフで確認します。



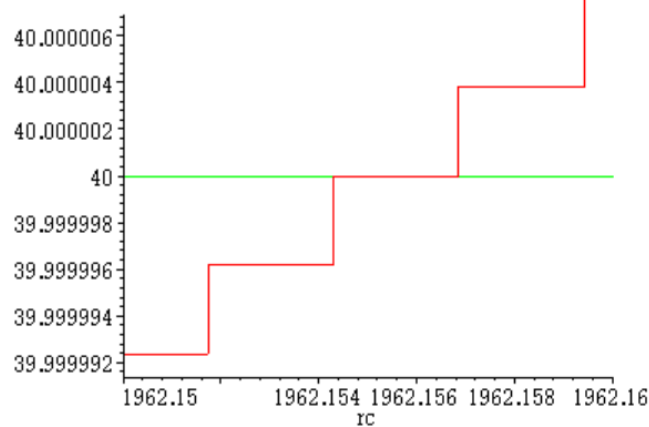
サンプルバッチファイルの回路図と説明

目標のゲインが 40dB なら、 $rc=2000$ の付近を数回絞り込めば、抵抗値を高精度で決定できると思います。

```
> plot({p(rc), 40}, rc=1962 .. 1963, numpoints=10000);
```



```
> plot({p(rc), 40}, rc=1962.15 .. 1962.16, numpoints=1000);
```



$rc=1962.154$ から 1962.157 の平均値、 $rc=1962.1555$ でゲインが 40 dB になります。

最後のグラフは、計算ポイント数を 1000 に減らして見やすくしました。

sim.exe の/ac コマンドでも、 $rc=1962.1555$ でゲインが 40 dB になります。

```
rc[1962] f[1,000]
rc*1000
p1 [ 1,962,155 j 0 ] abs[ 1,962,155] arg[ 0]
db(x5)*1000
p2 [39,999.9992 j 0 ] abs[39,999.9992] arg[ 0]
rc[1962] f[1,000]
rc*1000
p1 [ 1,962,156 j 0 ] abs[ 1,962,156] arg[ 0]
db(x5)*1000
p2 [40,000.0006 j 0 ] abs[40,000.0006] arg[ 0]
```

結果の数値は同じでも、グラフを見て即座に計算範囲を判断できるのは楽だと思います。

サンプルバッチファイルの回路図と説明

LC 回路 グラフが利用できると分かりやすい

信号源 E1 にインダクターL1 とキャパシターC1 を直列接続してグラウンドに接続した回路の出力を/range で確認します。

バッチファイルは、lc.sb

```
/ipart
```

```
1
```

```
e1
```

```
1
```

```
2
```

```
c1
```

```
0.1u
```

```
/
```

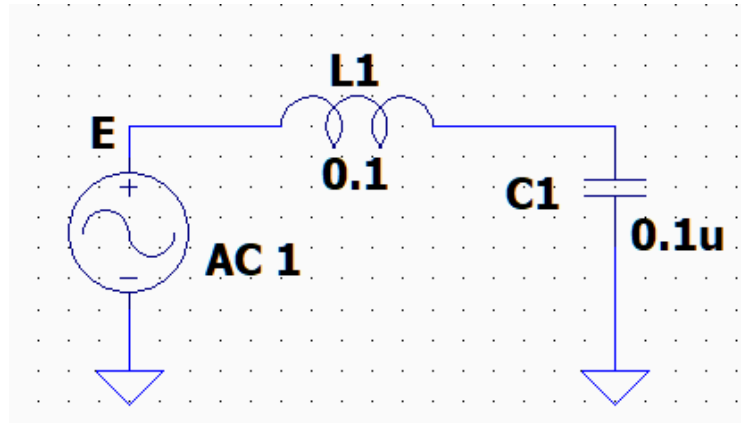
```
2
```

```
l1
```

```
0.1
```

```
*
```

```
2
```



```
独立電圧源素子の接続ノード
i[ 1] j[ 0] 部品名[e1] 値[1 j 0]
独立電圧源素子の個数 1
独立電流源素子の個数 0
シミュレーションにおける 表示ノード設定状況
```

独立電源は、信号源 e1 だけです。

```
? /dpart
left[ 0] right[ 1] parts[ e1          ] value[      1 j 0          ]
left[ 0] right[ 2] parts[ c1          ] value[ 1e-007 j 0          ]
left[ 1] right[ 2] parts[ l1          ] value[      0.1 j 0          ]
最大の 素子番号 = 3
最大のノード番号 = 2
```

この LC 回路の共振周波数は、 $f = \frac{1}{2\pi\sqrt{L1 \cdot C1}}$ で表されます。素子の値を代入すると、

$f = 1591.55 \text{ Hz}$ になります。

サンプルバッチファイルの回路図と説明

/range で、共振周波数付近のノード 2 の値を調べます。

```
? /range
ac 入力信号源名は ? e1
値は ? 1

range 周波数 最低値 ? 1591.5
range 周波数 最高値 ? 1591.6
range 周波数 ステップ ? 0.05
e1[1] j 0 ]
f[1, 591.5000]
x2 [16, 098. 9739 j 0 ] abs[16, 098. 9739] arg[ 0]
f[1, 591.5500]
x2 [-1, 398, 350. 1608 j 0 ] abs[1, 398, 350. 1608] arg[ -180]
f[1, 591.6000]
x2 [-15, 736. 1387 j 0 ] abs[15, 736. 1387] arg[ -180]
```

数値をよく見ると、f=1591.55 で、前後の数値とは桁違いに大きな数値になっていることが分かります。これが、共振現象です。

Maple で確認してみます。

係数行列ファイル lc.coe の restart: with(linalg): with(plots): の行から、s:=I*2*Pi*f; の行までをコピーアンドペーストして、x2 := 1; の行の先頭に # を追加します。

「エンター」を入力して、続けて入力します。

```
> x2:=x[2];

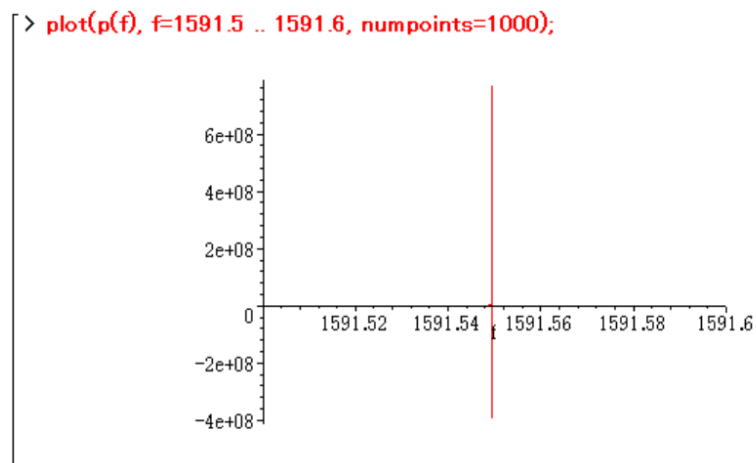
$$x2 := \frac{1}{-4 \cdot 10^{-7} \pi^2 f^2 + 1}$$

> p:= f-> x2;

$$p := f \rightarrow x2$$

```

グラフを作成します。



グラフでは、x2 が 0 から土の方向に大きく変化して、0 に戻るのが一目で分かります。

RLC 回路 3 個の素子の配置を変えるだけで 4 通りの特性

R1, L1, C1 を直列に接続しただけの回路ですから、接続ファイルを見て回路図を自分で作成して下さい。接続ファイルから回路図を作ることが簡単だと分かります。

3 個の素子の配置を変えると、4 通りの周波数特性が確認できます。

しかし、直列回路に流れる電流は同じなので、伝達関数の分母は同じ式になります。

$$s^2 + \frac{R1}{L1} \cdot s + \frac{1}{L1 \cdot C1} = s^2 + a \cdot s + b$$

$s^2 + a \cdot s + b = 0$ を s についての 2 次方程式と考えて、判別式が $D = \left(\frac{R1}{L1}\right)^2 - 4 \cdot \frac{1}{L1 \cdot C1} = 0$

となる R1 を求めます。

$$R1_0 = 2 \cdot \sqrt{\frac{L1}{C1}}$$

R1 が $R1_0$ より大きい、等しい、小さいかによって、判別式の符号が変わります。

この回路の特徴的な周波数を ω_0 とすると、 $\omega_0 = 2\pi \cdot f_0 = \frac{1}{\sqrt{L1 \cdot C1}}$ で表すことが出来ます。

R1 を $R1_0$ の前後で変化させると、RLC 回路の周波数特性がどのように変化するかを確認します。

伝達関数 $s^2 + \frac{R1}{L1} \cdot s + \frac{1}{L1 \cdot C1}$ を $s^2 + \frac{\omega_0}{Q} \cdot s + \omega_0^2$ と変形すると、 Q はフィルタのクオリティファクタと呼ばれます。

$$Q = \frac{1}{R1} \cdot \sqrt{\frac{L1}{C1}}$$

$f_0 = \frac{1}{2\pi \cdot \sqrt{L1 \cdot C1}}$ と $Q = \frac{1}{R1} \cdot \sqrt{\frac{L1}{C1}}$ より、 $f_0 \cdot Q = \frac{1}{2\pi \cdot C1 \cdot R1} = f_{cr}$ と表されます。従って、C1 と R1

で決まる周波数 f_{cr} と L1 と C1 で決まる周波数 f_0 の比が $Q = \frac{f_{cr}}{f_0}$ と表されます。

$Q > \frac{1}{\sqrt{2}}$ ならば、周波数特性にオーバーシュートが生じ、 $Q < \frac{1}{\sqrt{2}}$ ならば、ローパスフィルタ

では低い周波数からゲインが低下し始めます。

$f_0 = 1000$ とするために、 $L1=0.001$ H, $C1=25.33$ uF に設定します。

従って、 $\omega^2 = 39478878.79984$, $\omega = 6283.222$ となります。

サンプルバッチファイルの回路図と説明

ローパスフィルタ特性

バッチファイル rlcLPF.sb
 接続ファイル rlcLPF.cir

係数行列ファイル rlcLPF.coe

@rlclpf:0,1,3

入力ノードは 1、出力ノードは 3

3

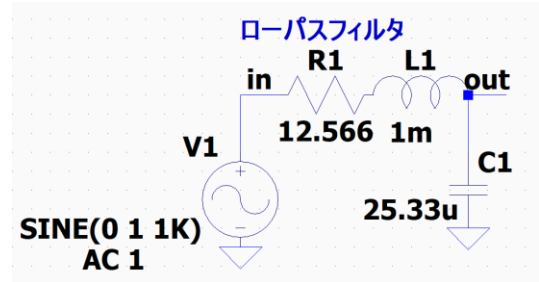
0 1 @e @e1 @ 0 1 0 @

0 3 @c @c1 @ 0 1 0 @

1 2 @r @r1 @ 0 1 0 @

2 3 @l @l1 @ 0 1 0 @

fnc



```
rlcLPF.mws
> restart: with(linalg): with(plots):
  siki:=matrix(3,3,0):e:=vector(3,0):
  siki[ 1, 1] := 1 ;
  e[1] := +e1 ;
  siki[ 2, 1] := -1/r1 ;
  siki[ 2, 2] := +1/r1+1/s/l1 ;
  siki[ 2, 3] := -1/s/l1 ;
  siki[ 3, 2] := -1/s/l1 ;
  siki[ 3, 3] := +s*c1+1/s/l1 ;
  x:=linsolve(siki,e);
```

```
> x3:=x[3]:g:=x3/e1;
```

$$g := \frac{1}{s^2 c l l1 + 1 + r l s c l}$$

ローパスフィルタの伝達関数は $\frac{1}{s^2 + \frac{R1}{L1} \cdot s + \frac{1}{L1 \cdot C1}}$ となります。

```
> l1:=0.001; c1:=25.33*10^(-6);s:=I*2*Pi*f;
  // := .001
  cl := .000025330000000
  s := 2 I pi f
> r10:=2*sqrt(l1/c1);
  r10 := 12.56644402
```

サンプルバッチファイルの回路図と説明

伝達関数 g を R1 の関数として表します。

```
> g:
      1
-----
-1.013200000 10-6 π2 f2 + 1 + .00005066000000 l r f π f
```

R1 に値を設定して、周波数特性のグラフを作成します。同時に Q を求めます。

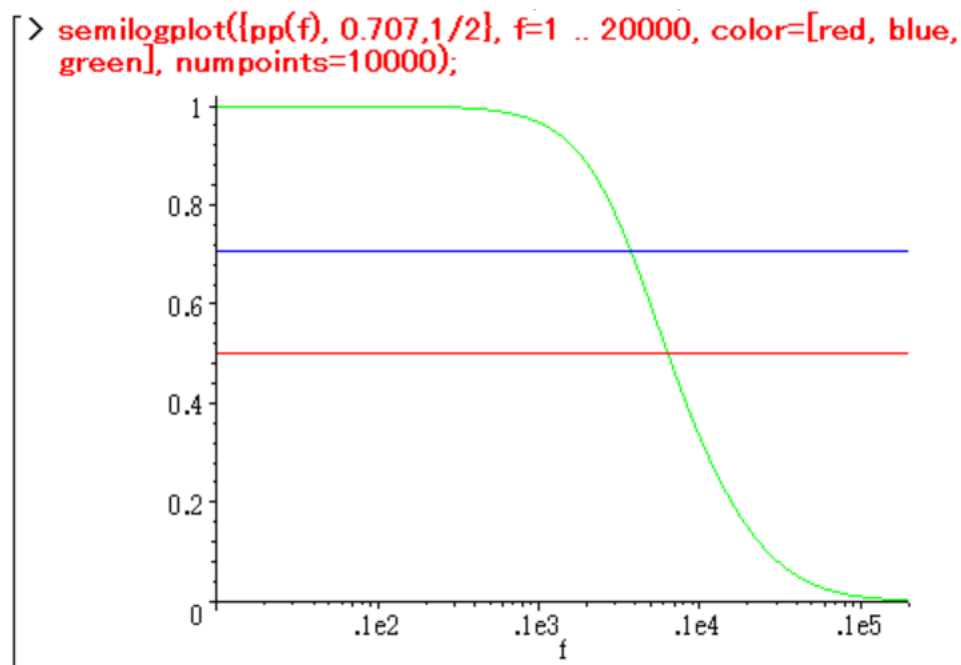
$D > 0$ となる $R1 = 1.5 * r10 = 18.849666$ による、周波数特性を確認する。

```
> gp:=subs(r1=1.5*r10,g); q:=1/r1*sqrt(l1/c1); qx:=subs(r1=1.5*r10,q);
      1
      gp := -----
      -1.013200000 10-6 π2 f2 + 1 + .0009549240811 l π f
      qx := .3333333332
> pp:= f -> abs(evalf(gp));
      pp := f -> |evalf(gp)|
```

$R1 = 1.5 * r10 = 18.84966$ の場合は、 $Q = 0.333$ になります。

伝達関数は、 $\frac{39478878.7998421}{s^2 + 18849.66603 \cdot s + 39478878.7998421}$ になります。

ここで、 $R1$ と $C1$ で決まる周波数 f_{cr} は、 $f_{cr} = \frac{1}{2 \cdot \pi \cdot C1 \cdot R1} = 333 = f_0 \cdot Q$ となります。



2 次ローパスフィルタなので、 $f = 1000$ で -6 dB(0.5)の予想でしたが -10dB 程度に達し、 $f_{cr} = f_0 \cdot Q = 333$ で -3dB(0.7)程度になっています。 $D > 0$ ではゲインの低下が早まります。

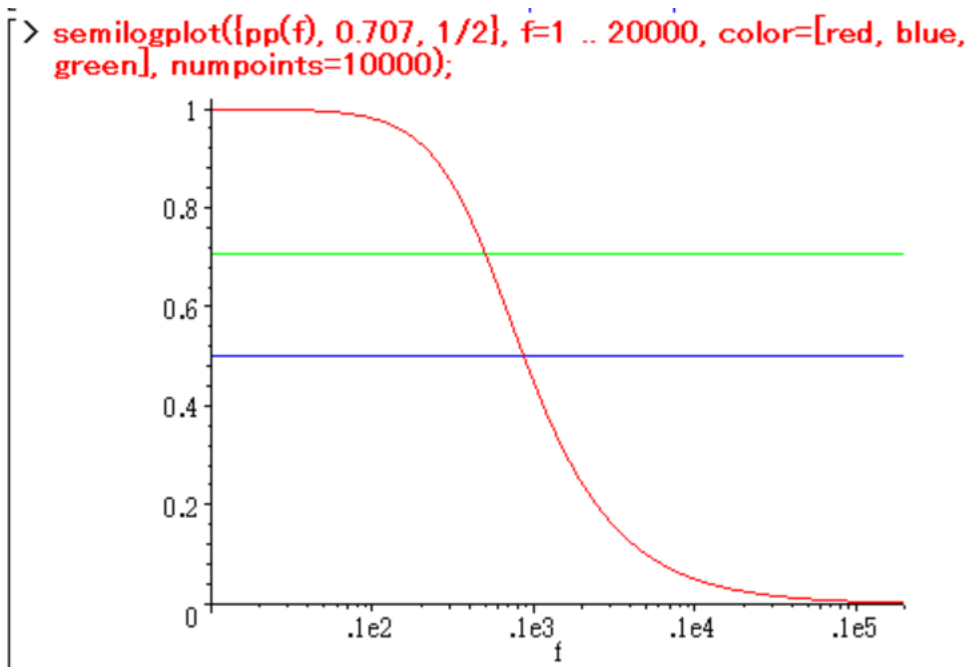
D>0 となる $R1=10*r10=125.66444$ による、周波数特性を確認する。

```
> gp:=subs(r1=10*r10,g); qx:=subs(r1=10*r10,q);
      gp := 
$$\frac{1}{-.1013200000 \cdot 10^{-6} \pi^2 f^2 + 1 + .006366160541 i \pi f}$$

      qx := .049999999998
> pp:= f -> abs(evalf(gp));
      pp :=  $f \rightarrow |evalf(gp)|$ 
```

$R1=10*r10=125.66444$ の場合には、 $Q=0.05$ になります。

伝達関数は、 $\frac{39478878.7998421}{s^2+125664.4402 \cdot s+39478878.7998421}$ になります。



$f=100$ 位からゲインが低下し始めています。

$R1$ が大きくなり、 $Q < \frac{1}{\sqrt{2}}$ が小さくなると、低い周波数 $f_{cr}=f_0*Q=50$ でゲインが 3dB 低下しています。 Q が小さくなるほど、低い周波数からゲインが低下し始めます。

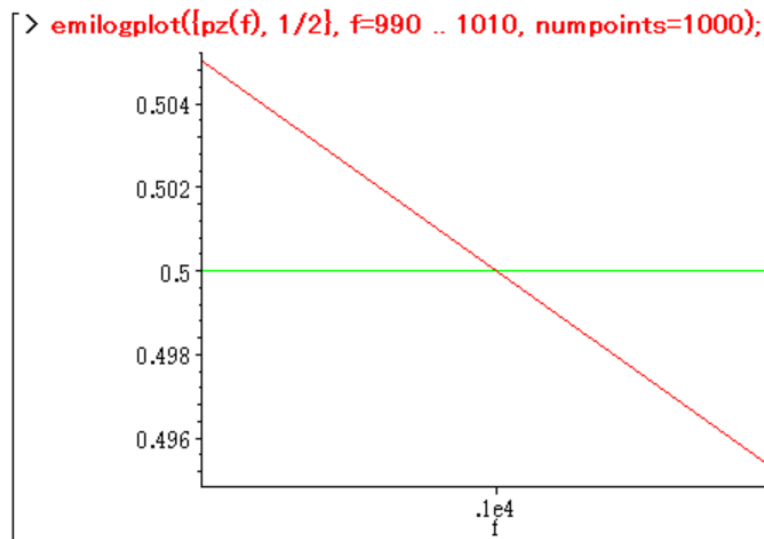
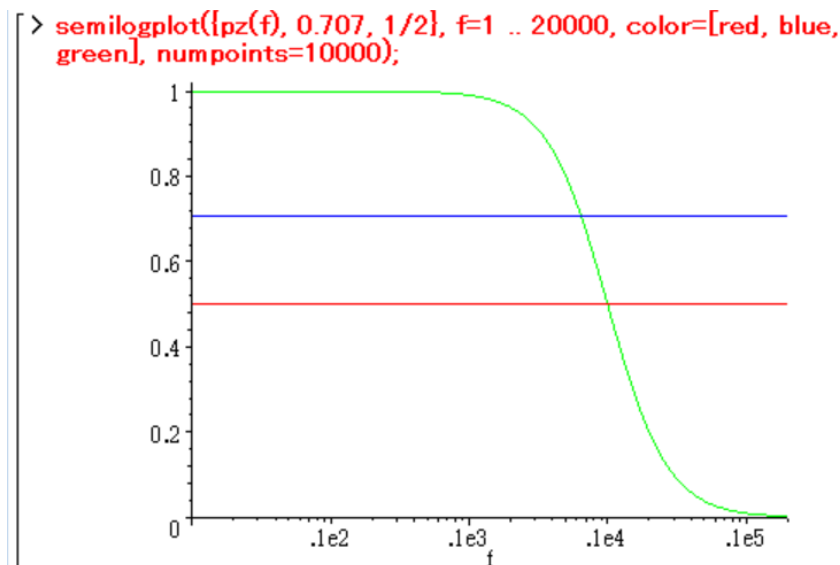
サンプルバッチファイルの回路図と説明

D=0 となる R1=r10=12.566444 による、周波数特性を確認する。

```
> gz:=subs(r1=r10,g); qx:=subs(r1=r10,q);
      1
      -
      - .1013200000 10-6 π2 f2 + 1 + .0006366160541 i π f
      qx := .49999999998
> pz:= f -> abs(evalf(gz));
      pz := f -> |evalf(gz)|
```

R1=r10=12.566444 の場合には、Q=0.5 になります。

伝達関数は、 $\frac{39478878.7998421}{s^2 + 12566.44402 \cdot s + 39478878.7998421}$ になります。



ちょうど f=1000 でゲインが 6 dB 低下しています。D=0 では f0 で -6dB になります。

サンプルバッチファイルの回路図と説明

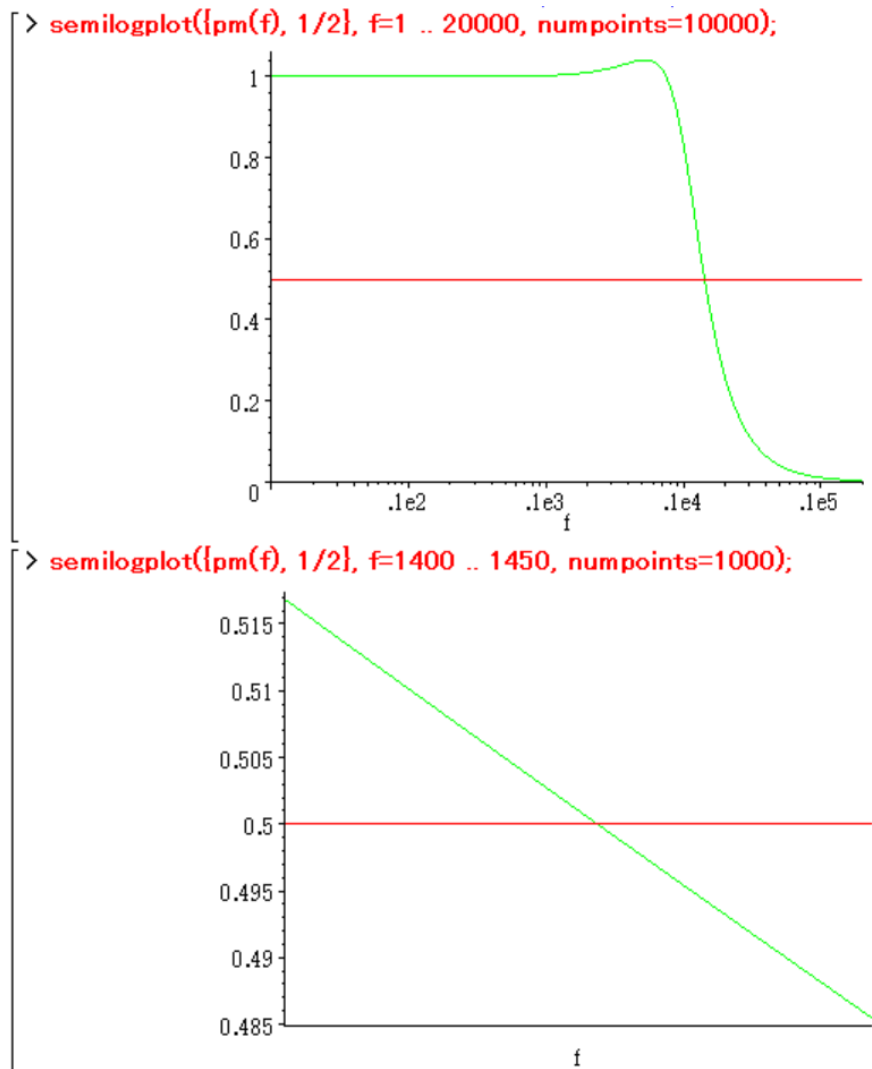
D<0 となる R1=0.6*r10=7.5398664 による、周波数特性を確認する。

```
> gm:=subs(r1=0.6*r10,g); qx:=subs(r1=0.6*r10,q);
gm := 
$$\frac{1}{-1.013200000 \cdot 10^{-6} \pi^2 f^2 + 1 + .0003819696324 i \pi f}$$

qx := .8333333330
> pm:= f -> abs(evalf(gm));
pm := f -> |evalf(gm)|
```

R1=0.6*r10=7.5398664 の場合には、Q=0.833 になります。

伝達関数は、 $\frac{39478878.7998421}{s^2 + 7539.866412 \cdot s + 39478878.7998421}$ になります。



f=1425Hz 位で、ゲインが 6 dB 減衰して、オーバシュートも発生しています。

D<0 では、ゲインが低下する周波数が高い周波数に移動しています。

サンプルバッチファイルの回路図と説明

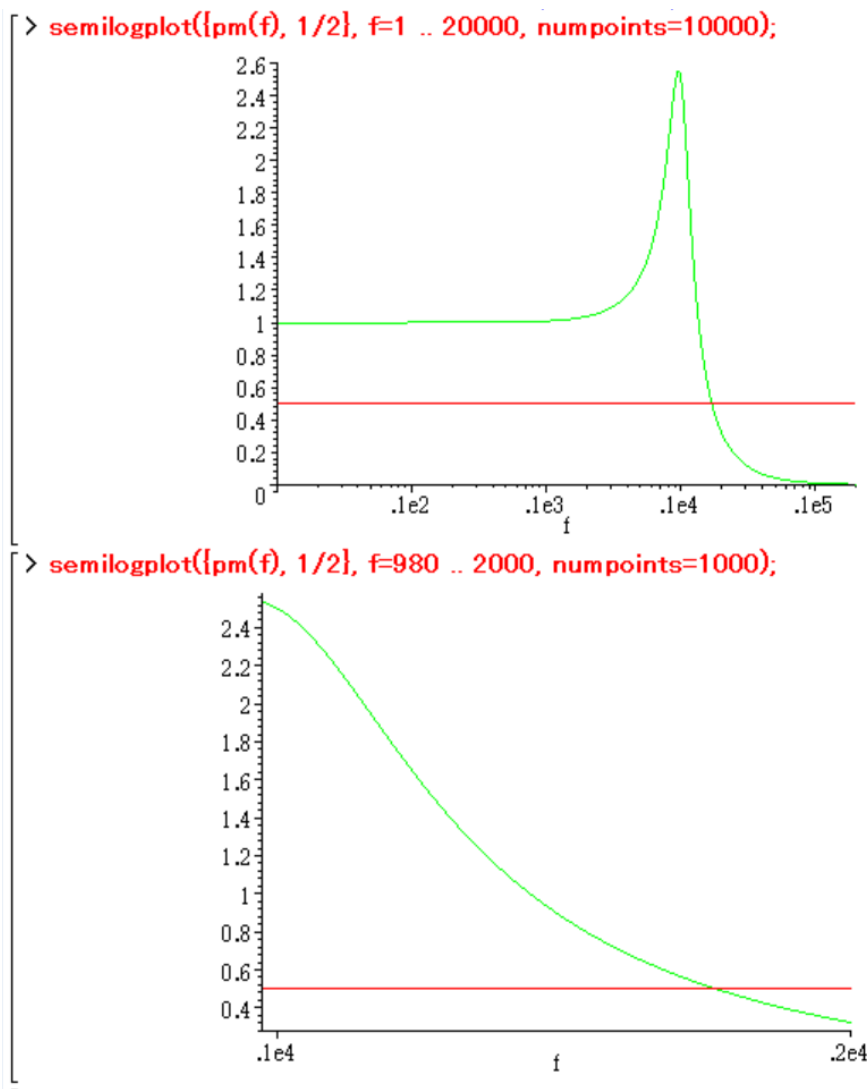
D<0 となる R1=0.2*r10=2.5132888 による、周波数特性を確認する。

```
> gm:=subs(r1=0.2*r10,g); qx:=subs(r1=0.2*r10,q);
gm := 
$$\frac{1}{-1.013200000 \cdot 10^{-6} \pi^2 f^2 + 1 + .0001273232108 f \pi f}$$

qx := 2.499999999
> pm:= f -> abs(evalf(gm));
pm := f -> |evalf(gm)|
```

R1=0.2*r10=2.5132888 の場合には、Q=2.5 になります。

伝達関数は、 $\frac{39478878.7998421}{s^2 + 2513.288804 \cdot s + 39478878.7998421}$ になります。



ゲインが 6dB 低下する周波数は、2 KHz 位まで上昇しており、f=1000 Hz 付近に大きなピークが現れます。

Q が大きくなるほど、ローパスフィルタのピークが高くなります。

サンプルバッチファイルの回路図と説明

ハイパスフィルタ特性

バッチファイル	rlcHPF.sb	係数行列ファイル	rlcHPF.coe
接続ファイル	rlcHPF.cir		

@rlchpf0,1,3

入力ノードは 1、出力ノードは 3

3

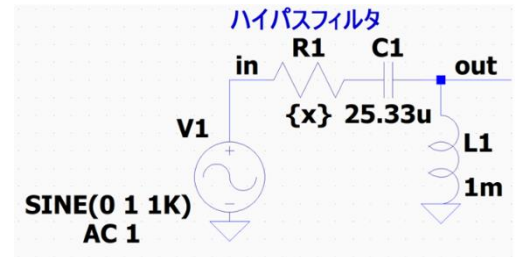
0 1 @e @e1 @ 0 1 0 @

0 3 @l @l1 @ 0 1 0 @

1 2 @r @r1 @ 0 1 0 @

2 3 @c @c1 @ 0 1 0 @

fnc



```

> restart: with(linalg): with(plots):
siki:=matrix(3,3,0):e:=vector(3,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 1] := -1/r1 ;
siki[ 2, 2] := +1/r1+s*c1 ;
siki[ 2, 3] := -s*c1 ;
siki[ 3, 2] := -s*c1 ;
siki[ 3, 3] := +1/s/l1+s*c1 ;
x:=linsolve(siki,e);

```

```

> x3:=x[3]:g:=x3/e1;

```

$$g := \frac{s^2 \text{ c1 l1}}{1 + s^2 \text{ c1 l1} + s \text{ c1 r1}}$$

ハイパスフィルタの伝達関数は $\frac{s^2}{s^2 + \frac{R1}{L1} \cdot s + \frac{1}{L1 \cdot C1}}$ となります。

```

> l1:=0.001; c1:=25.33*10^(-6);s:=I*2*Pi*f;
// := .001
c1 := .00002533000000
s := 2 I pi f
> r10:=2*sqrt(l1/c1);
r10 := 12.56644402
> g;
-.1013200000 10^-6 \frac{\pi^2 f^2}{1 - .1013200000 10^-6 \pi^2 f^2 + .00005066000000 I \pi f r1}

```

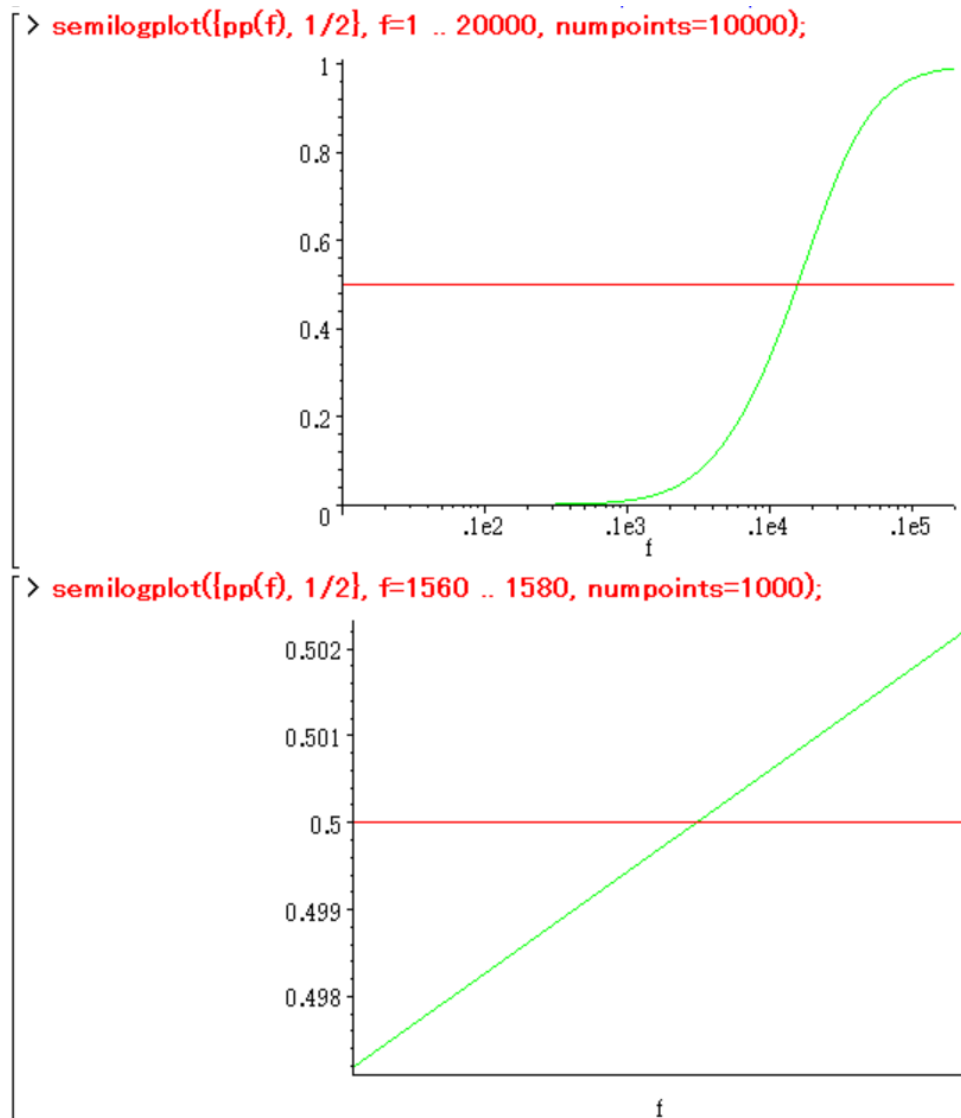
サンプルバッチファイルの回路図と説明

D>0 となる $R1=1.5*r10=18.849666$ による、周波数特性を確認する。

```
> gp:=subs(r1=1.5*r10,g); q:=1/r1*sqrt(l1/c1); qx:=subs(r1=1.5*r10,q);
      gp := -1.013200000 10-6  $\frac{\pi^2 f^2}{1 - 1.013200000 10^{-6} \pi^2 f^2 + .0009549240811 i \pi f}$ 
      qx := .333333332
> pp:=f -> abs(evalf(gp));
      pp := f -> |evalf(gp)|
```

$R1=1.5*r10=18.849666$ の場合には、 $Q=0.333$ になります。

伝達関数は、 $\frac{s^2}{s^2+18849.66603 \cdot s+39478878.7998421}$ になります。



$f=1570$ Hz 付近で、+6 dB ゲインが上昇しています。

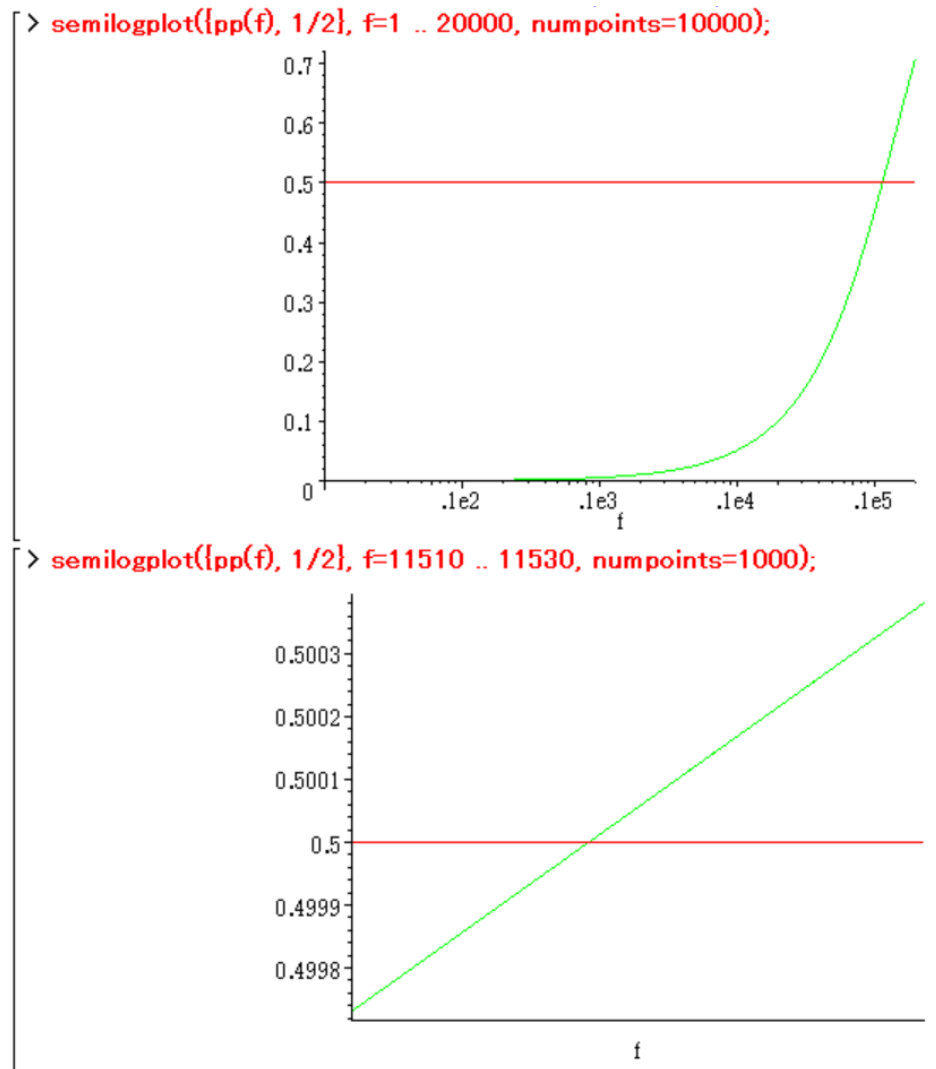
サンプルバッチファイルの回路図と説明

D>0 となる $R1=10*r10=125.66444$ による、周波数特性を確認する。

```
> gp:=subs(r1=10*r10,g); qx:=subs(r1=10*r10,q);
      gp := -1.013200000 10-6  $\frac{\pi^2 f^2}{1 - .1013200000 10^{-6} \pi^2 f^2 + .006366160541 i \pi f}$ 
      qx := .04999999998
> pp:= f -> abs(evalf(gp));
      pp := f -> |evalf(gp)|
```

$R1=10*r10=125.66444$ の場合には、 $Q=0.05$ になります。

伝達関数は、 $\frac{s^2}{s^2 + 125664.4402 \cdot s + 39478878.7998421}$ になります。



$f=11.52$ KHz 程度で、ゲインが 6 dB 上昇している。

$R1$ が $r10$ よりも大きくなるほど、ゲインが上昇し始める周波数は高くなります。

サンプルバッチファイルの回路図と説明

D=0 となる R1=r10=12.566444 による、周波数特性を確認する。

```
> gz:=subs(r1=r10,g); qx:=subs(r1=r10,q);
      
$$gz := -1.013200000 \cdot 10^{-6} \frac{\pi^2 f^2}{1 - 1.013200000 \cdot 10^{-6} \pi^2 f^2 + .0006366160541 i \pi f}$$

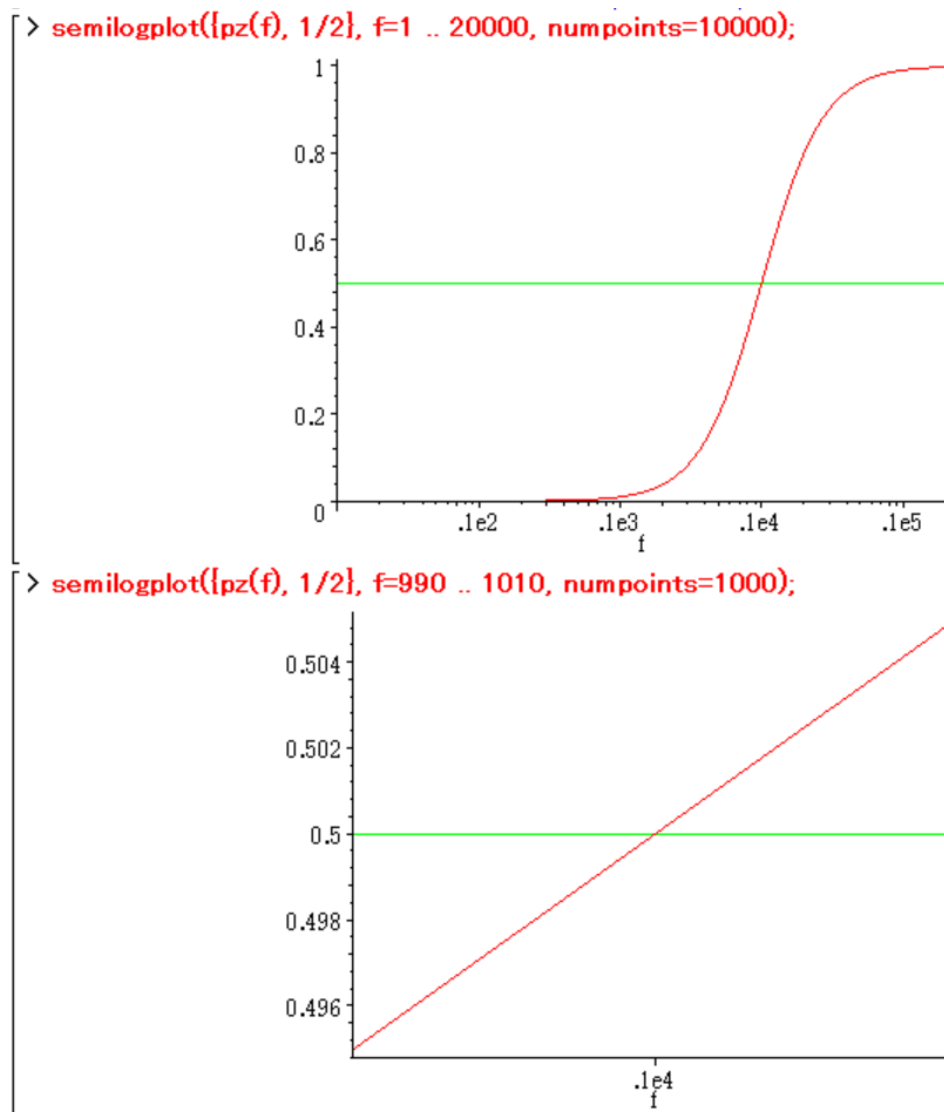
      
$$qx := .4999999998$$

> pz:= f -> abs(evalf(gz));
      
$$pz := f \rightarrow |\text{evalf}(gz)|$$

```

R1=r10=12.566444 の場合には、Q=0.5 になります。

伝達関数は、 $\frac{s^2}{s^2 + 12566.44402 \cdot s + 39478878.7998421}$ になります。



D=0 では、ちょうど f=1000 Hz でゲインが 6 dB 上昇しています。

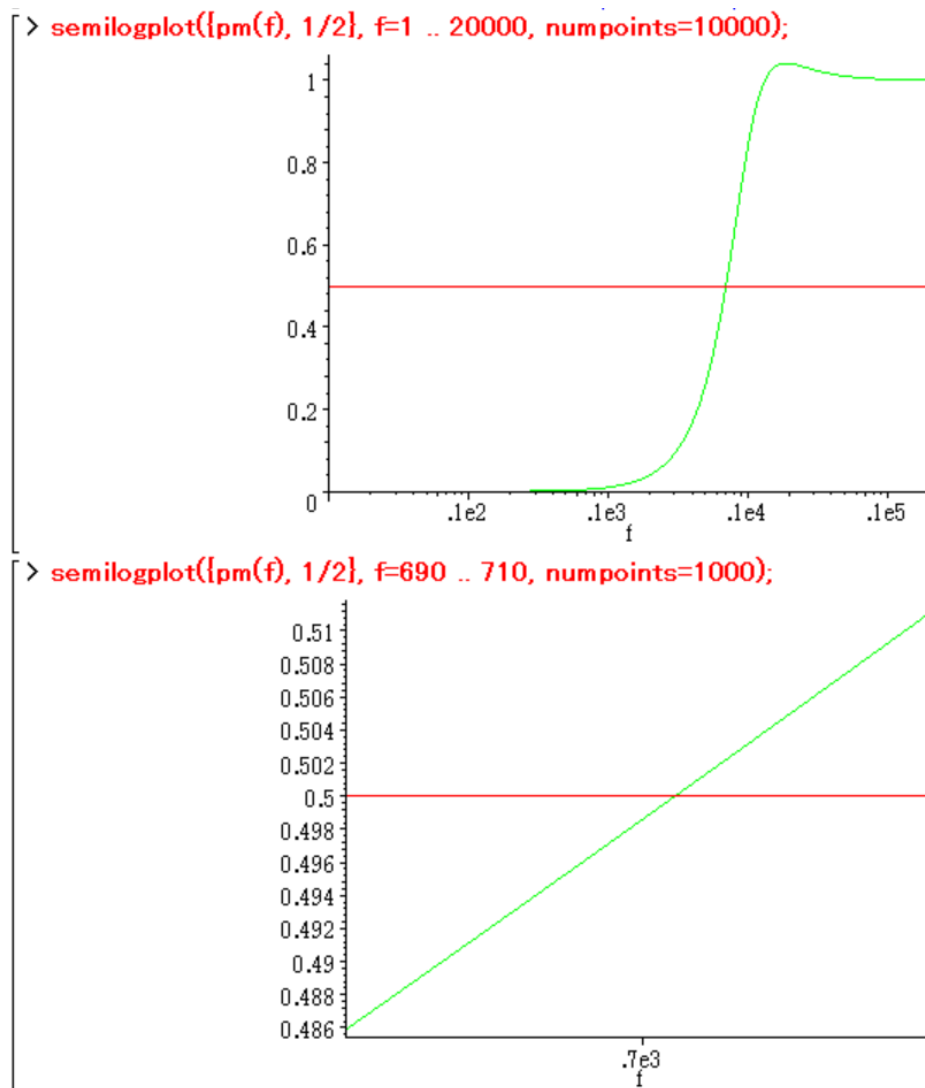
サンプルバッチファイルの回路図と説明

D<0 となる R1=0.6*r10=7.5398664 による、周波数特性を確認する。

```
> gm:=subs(r1=0.6*r10,g); qx:=subs(r1=0.6*r10,q);
      gm := - .1013200000 10-6  $\frac{\pi^2 f^2}{1 - .1013200000 10^{-6} \pi^2 f^2 + .0003819696324 i \pi f}$ 
      qx := .8333333330
> pm:=f -> abs(evalf(gm));
      pm := f -> |evalf(gm)|
```

R1=0.6*r10=7.5398664 の場合には、Q=0.833 になります。

伝達関数は、 $\frac{s^2}{s^2 + 7539.866412 \cdot s + 39478878.7998421}$ になります。



f=1000 Hz 付近にオーバシュートが現れました。

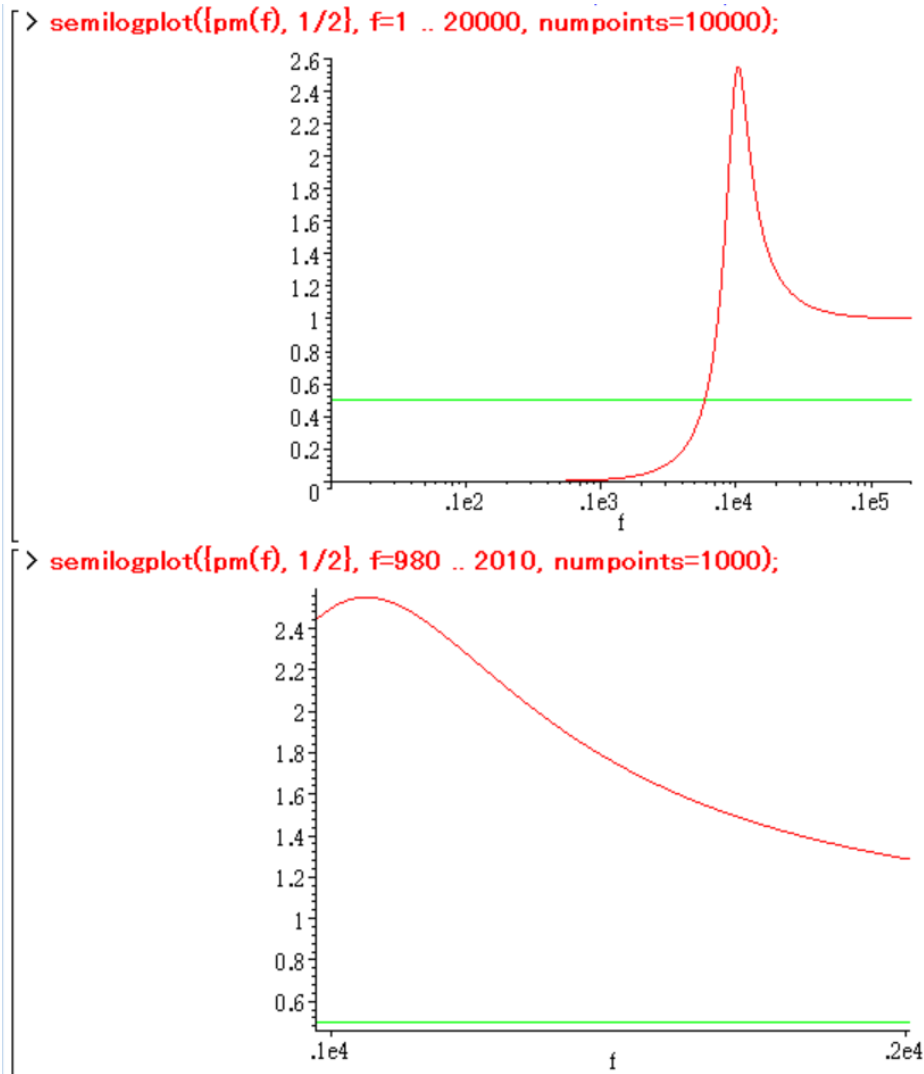
f=700 Hz 付近でゲインが 6 dB 上昇しています。

D<0 となる R1=0.2*r10=2.5132888 による、周波数特性を確認する。

```
> gm:=subs(r1=0.2*r10,g); qx:=subs(r1=0.2*r10,q);
      gm := - .1013200000 10-6  $\frac{\pi^2 f^2}{1 - .1013200000 10^{-6} \pi^2 f^2 + .0001273232108 i \pi f}$ 
      qx := 2.499999999
> pm:=f->abs(evalf(gm));
      pm := f->|evalf(gm)|
```

R1=0.2*r10=2.5132888 の場合には、Q=2.5 になります。

伝達関数は、 $\frac{s^2}{s^2 + 2513.288804 \cdot s + 39478878.7998421}$ になります。



f=1000 Hz 付近に大きなピークが見られます。

Q が大きくなるほど、ハイパスフィルタのピークが大きくなります。

サンプルバッチファイルの回路図と説明

バンドパスフィルタ特性

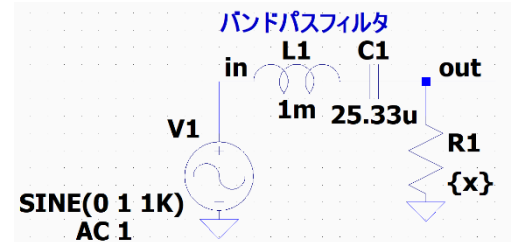
バッチファイル rlcBPF.sb
 接続ファイル rlcBPF.cir

係数行列ファイル rlcBPF.coe

@rlcbpf:0,1,3

入力ノードは 1、出力ノードは 3

```
3
0 1 @e @e1 @ 0 1 0 @
0 3 @r @r1 @ 0 1 0 @
1 2 @l @l1 @ 0 1 0 @
2 3 @c @c1 @ 0 1 0 @
fnc
```



```
> x3:=x[3]:g:=x3/e1;
```

$$g := \frac{s \, c1 \, r1}{1 + s \, c1 \, r1 + s^2 \, c1 \, l1}$$

バンドパスフィルタの伝達関数は $\frac{\frac{R1}{L1} \cdot s}{s^2 + \frac{R1}{L1} \cdot s + \frac{1}{L1 \cdot C1}}$ となります。

```
> l1:=0.001; c1:=25.33*10^(-6);s:=I*2*Pi*f;
// := .001
cf := .00002533000000
s := 2 I pi f
> r10:=2*sqrt(l1/c1);
r10 := 12.56644402
```

伝達関数 g を R1 の関数として表します。

```
> g;
.00005066000000  $\frac{I \pi f r1}{1 + .00005066000000 I \pi f r1 - .1013200000 10^{-6} \pi^2 f^2}$ 
> q:=1/r1*sqrt(l1/c1);
q := 6.283222008  $\frac{1}{r1}$ 
```

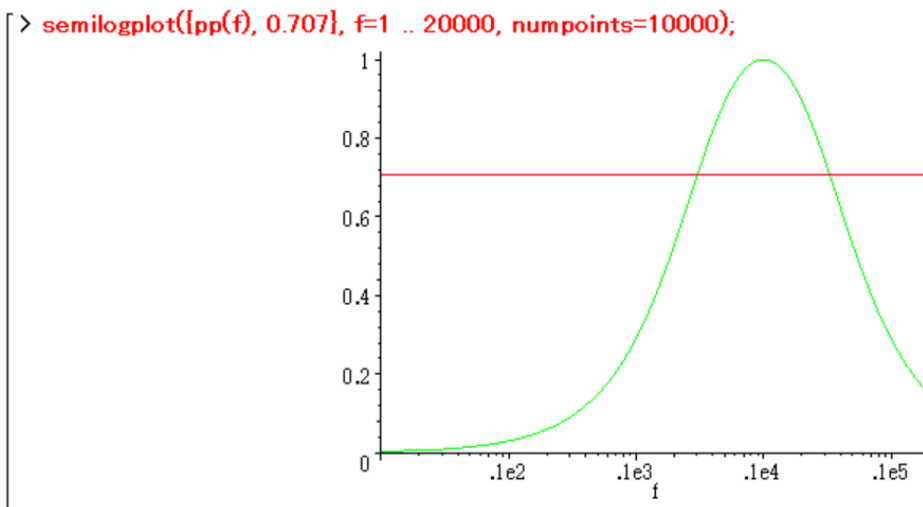
R1 に値を設定して、周波数特性のグラフを作成します。同時に Q を求めます。

D>0 となる $R1=1.5*r10=18.849666$ による、周波数特性を確認する。

```
[> gp:=subs(r1=1.5*r10,g); qx:=subs(r1=1.5*r10,q);
      gp := .0009549240811  $\frac{I \pi f}{1 + .0009549240811 I \pi f - .1013200000 10^{-6} \pi^2 f^2}$ 
      qx := .333333332
> pp:= f -> abs(evalf(gp));
      pp := f -> |evalf(gp)|
```

$R1=1.5*r10=18.849666$ の場合には、 $Q=0.333$ になります。

伝達関数は、 $\frac{18849.66603 \cdot s}{s^2 + 18849.66603 \cdot s + 39478878.7998421}$ になります。



$f=300$ Hz から 3 KHz 付近までのゲインが-3dB 以上の通過域になっています。

通過帯域幅はおよそ、 $\frac{f_0}{Q} = \frac{1000}{0.333} = 3000 = 3KHz$ です。

D>0 となる $R1=10*r10$ による、周波数特性を確認する。

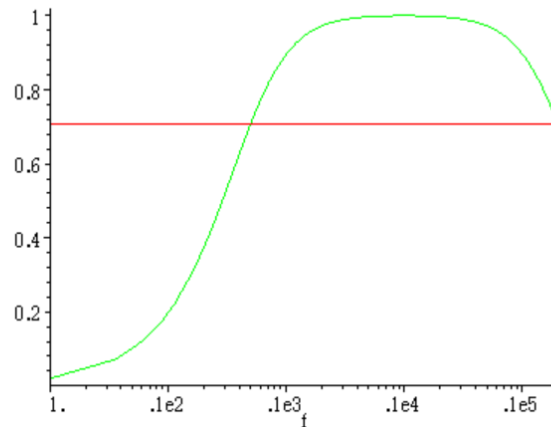
```
[> gp:=subs(r1=10*r10,g); qx:=subs(r1=10*r10,q);
      gp := .006366160541  $\frac{I \pi f}{1 + .006366160541 I \pi f - .1013200000 10^{-6} \pi^2 f^2}$ 
      qx := .04999999998
> pp:= f -> abs(evalf(gp));
      pp := f -> |evalf(gp)|
```

$R1=10*r10$ の場合には、 $Q=0.05$ になります。

伝達関数は、 $\frac{125664.4402 \cdot s}{s^2 + 125664.4402 \cdot s + 39478878.7998421}$ になります。

サンプルバッチファイルの回路図と説明

```
> semilogplot([pp(f), 0.707], f=1 .. 20000, numpoints=1000);
```



通過域は、 $f=30$ Hz から 20 KHz まで広がり、ゲインがフラットな -1dB~0dB の範囲も広くなりました。通過帯域幅はおよそ、 $\frac{f_0}{Q} = \frac{1000}{0.05} = 20000 = 20\text{KHz}$ です。

Q が小さくなるほど、バンドパスフィルタの通過域が広がります。

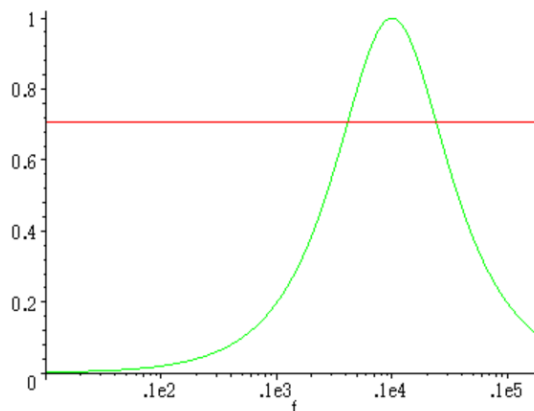
$D=0$ となる $R1=r10=12.566444$ による、周波数特性を確認する。

```
> gz:=subs(r1=r10,g); qx:=subs(r1=r10,q);
      gz = .0006366160541  $\frac{f \pi}{1 + .0006366160541 f \pi - .1013200000 \cdot 10^{-6} \pi^2 f^2}$ 
      qx = .4999999998
> pz:= f -> abs(evalf(gz));
      pz = f -> |evalf(gz)|
```

$R1=r10=12.566444$ の場合には、 $Q=0.5$ になります。

伝達関数は、 $\frac{12566.44402 \cdot s}{s^2 + 12566.44402 \cdot s + 39478878.7998421}$ になります。

```
> semilogplot([pz(f), 0.707], f=1 .. 20000, numpoints=10000);
```



通過域は、 $f=400$ Hz から 2.4 KHz 程度です。フラットな帯域は狭いです。

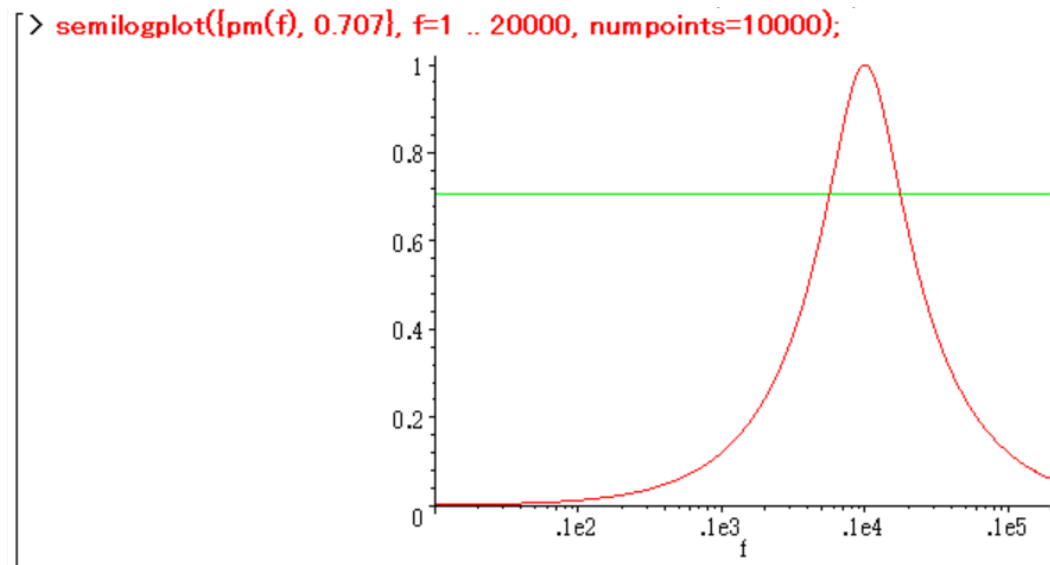
サンプルバッチファイルの回路図と説明

D<0 となる R1=0.6*r10=7.5398664 による、周波数特性を確認する。

```
> gm:=subs(r1=0.6*r10,g); qx:=subs(r1=0.6*r10,q);
gm := .0003819696324  $\frac{I \pi f}{1 + .0003819696324 I \pi f - .1013200000 10^{-6} \pi^2 f^2}$ 
qx := .8333333330
> pm:= f -> abs(evalf(gm));
pm := f -> |evalf(gm)|
```

R1=0.6*r10=7.5398664 の場合には、Q=0.833 になります。

伝達関数は、 $\frac{7539.866412 \cdot s}{s^2 + 7539.866412 \cdot s + 39478878.7998421}$ になります。



通過域は f=400Hz~2KHz に狭くなりました。

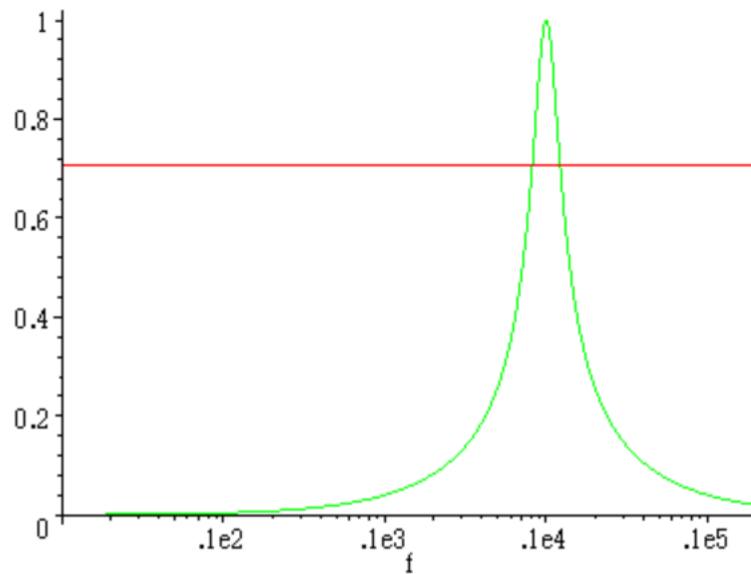
D<0 となる R1=0.2*r10=2.5132888 による、周波数特性を確認する。

```
> gm:=subs(r1=0.2*r10,g); qx:=subs(r1=0.2*r10,q);
gm := .0001273232108  $\frac{I \pi f}{1 + .0001273232108 I \pi f - .1013200000 10^{-6} \pi^2 f^2}$ 
qx := 2.499999999
> pm:= f -> abs(evalf(gm));
pm := f -> |evalf(gm)|
```

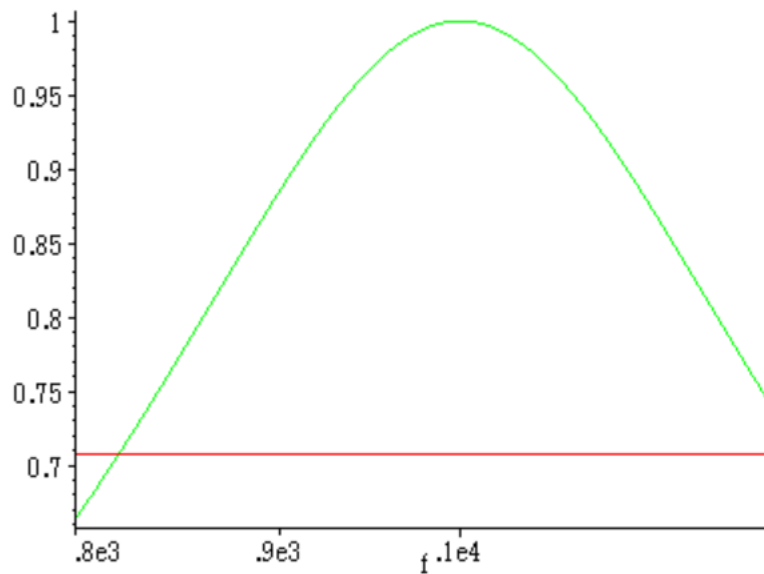
R1=0.2*r10=2.5132888 の場合には、Q=2.5 になります。

伝達関数は、 $\frac{2513.288804 \cdot s}{s^2 + 2513.288804 \cdot s + 39478878.7998421}$ になります。

```
> semilogplot({pm(f), 0.707}, f=1 .. 20000, numpoints=10000);
```



```
> semilogplot({pm(f), 0.707}, f=800 .. 1200, numpoints=1000);
```



通過域は、 $f=800\text{ Hz} \sim 1200\text{ Hz}$ ($400 = 1000 / 2.5$) に狭くなりました。

$R1=0.01*r10=0.12566$, $Q=50$ にすると、通過域は $f=990\text{ Hz} \sim 1010\text{ Hz}$ まで狭くなります。通過域は 20 Hz ですが、これは $f=1000\text{ Hz}$ に対して、 f/Q に相当する値です。試しに、 $R1=0.001*r10=0.012566$, $Q=500$ にすると、通過域は $f=999\text{ Hz} \sim 1001\text{ Hz}$ になり、通過域が $2 = 1000 / 500$ になります。

Q が大きくなるほど、バンドパスフィルタの通過域は $\frac{f_0}{Q}$ に従って狭くなります。

サンプルバッチファイルの回路図と説明

バンドエリミネーションフィルタ特性

バッチファイル	rlcBEF.sb	係数行列ファイル	rlcBEF.coe
接続ファイル	rlcBEF.cir		

@rlcbef:0,1,2

入力ノードは 1、出力ノードは 2

3

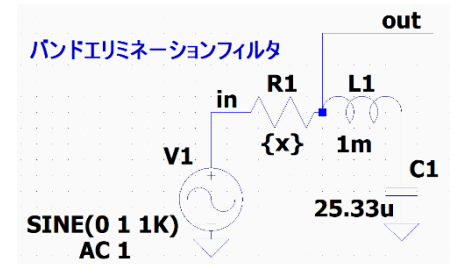
0 1 @e @e1 @ 0 1 0 @

0 3 @c @c1 @ 0 1 0 @

1 2 @r @r1 @ 0 1 0 @

2 3 @l @l1 @ 0 1 0 @

fnc



```

rlcBEF.mws
> restart: with(linalg): with(plots):
siki:=matrix(3,3,0):e:=vector(3,0):
siki[ 1, 1] := 1 ;
e[1] := +e1 ;
siki[ 2, 1] := -1/r1 ;
siki[ 2, 2] := +1/r1+1/s/l1 ;
siki[ 2, 3] := -1/s/l1 ;
siki[ 3, 2] := -1/s/l1 ;
siki[ 3, 3] := +s*c1+1/s/l1 ;
x:=linsolve(siki,e);

```

```

> x2:=x[2]:g:=x2/e1;

```

$$g = \frac{s^2 c1 l1 + 1}{s^2 c1 l1 + 1 + r1 s c1}$$

バンドエリミネーションフィルタの伝達関数は $\frac{s^2 + \frac{1}{L1 \cdot C1}}{s^2 + \frac{R1}{L1} \cdot s + \frac{1}{L1 \cdot C1}}$ となります。

```

> l1:=0.001; c1:=25.33*10^(-6);s:=I*2*Pi*f;

```

$$l1 := .001$$

$$c1 := .00002533000000$$

$$s := 2 \pi f$$

```

> r10:=2*sqrt(l1/c1);

```

$$r10 := 12.56644402$$

サンプルバッチファイルの回路図と説明

伝達関数 g を R1 の関数として表します。

```
> g;
      -1.013200000 10-6 π2 f2 + 1
      -----
      -1.013200000 10-6 π2 f2 + 1 + .000050660000000 1/rf π f
> q:=1/r1*sqrt(l1/c1);
      q := 6.283222008 1/rf
```

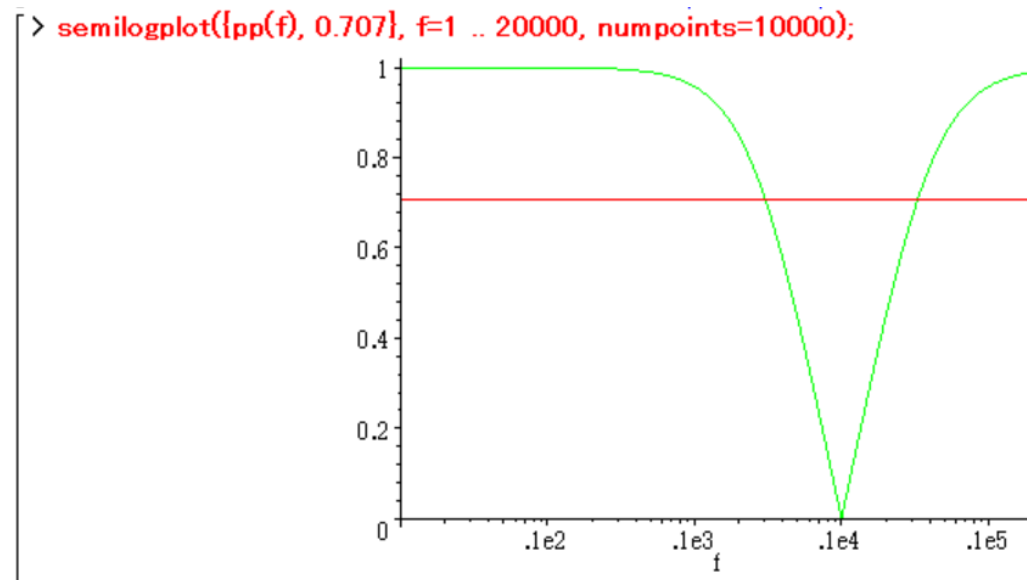
R1 に値を設定して、周波数特性のグラフを作成します。同時に Q を求めます。

$D > 0$ となる $R1 = 1.5 * r10 = 18.849666$ による、周波数特性を確認する。

```
> gp:=subs(r1=1.5*r10,g); qx:=subs(r1=1.5*r10,q);
      -1.013200000 10-6 π2 f2 + 1
      -----
      -1.013200000 10-6 π2 f2 + 1 + .0009549240811 1/π f
      qx := .3333333332
> pp:= f -> abs(evalf(gp));
      pp := f -> |evalf(gp)|
```

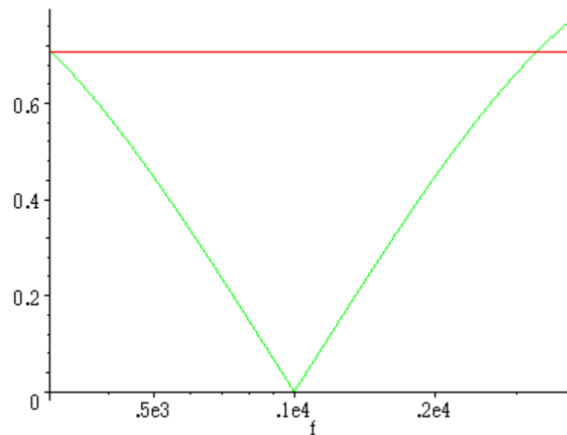
$R1 = 1.5 * r10 = 18.849666$ の場合には、 $Q = 0.333$ になります。

伝達関数は、 $\frac{s^2 + 39478878.7998421}{s^2 + 18849.66603 \cdot s + 39478878.7998421}$ になります。



サンプルバッチファイルの回路図と説明

```
> semilogplot({pp(f), 0.707}, f=300 .. 4000, numpoints=1000);
```



f=300 Hz～4 KHz の周波数帯はゲインが -3dB 以下の阻止域になります。

D>0 となる R1=10*r10=125.66444 による、周波数特性を確認する。

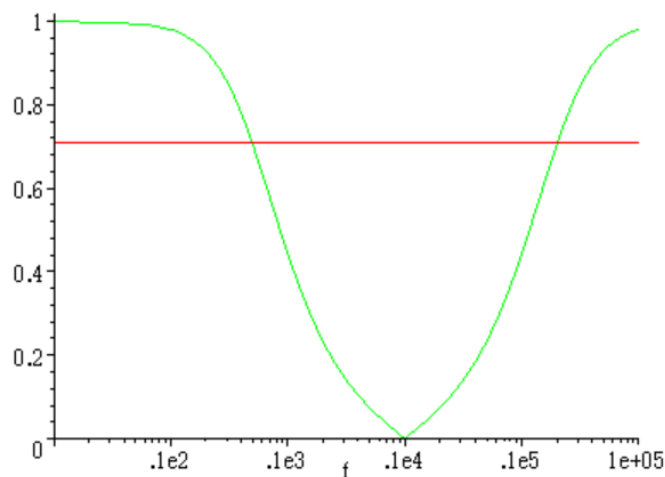
```
> gp:=subs(r1=10*r10,g); qx:=subs(r1=10*r10,q);
gp := 
$$\frac{-1.013200000 \cdot 10^{-6} \pi^2 f^2 + 1}{-1.013200000 \cdot 10^{-6} \pi^2 f^2 + 1 + .006366160541 i \pi f}$$

qx := .04999999998
> pp:= f -> abs(evalf(gp));
pp := f -> |evalf(gp)|
```

R1=10*r10=125.66444 の場合には、Q=0.05 になります。

伝達関数は、 $\frac{s^2 + 39478878.7998421}{s^2 + 125664.4402 \cdot s + 39478878.7998421}$ になります。

```
> semilogplot({pp(f), 0.707}, f=1 .. 100000, numpoints=20000);
```



阻止域が f=50 Hz～20 KHz の範囲に広がっています。

Q が小さくなるほど、バンドエリミネーションフィルタの阻止域は広がります。

サンプルバッチファイルの回路図と説明

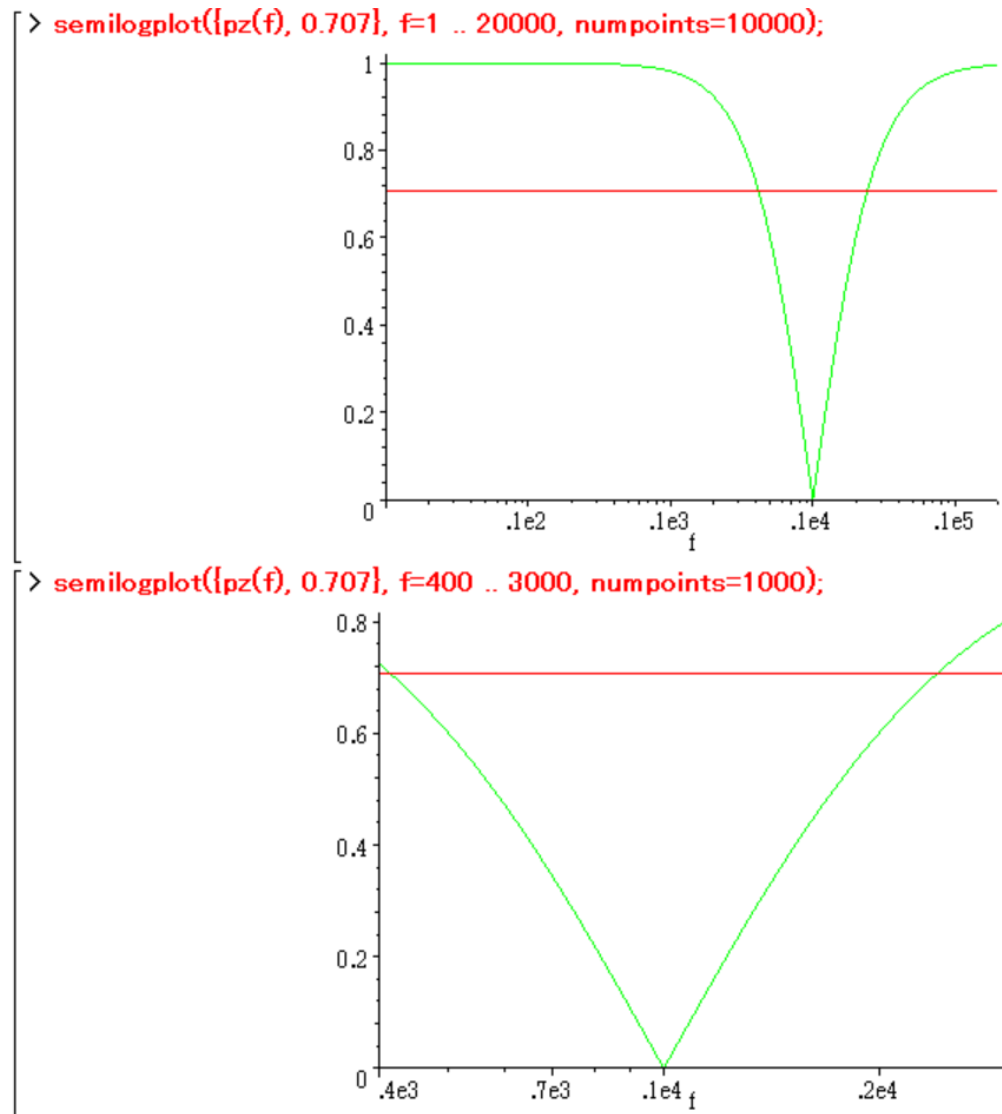
D=0 となる R1=r10=12.566444 による、周波数特性を確認する。

```
> gz:=subs(r1=r10,g); qx:=subs(r1=r10,q);
      
$$gz := \frac{-1.013200000 \cdot 10^{-6} \pi^2 f^2 + 1}{-1.013200000 \cdot 10^{-6} \pi^2 f^2 + 1 + .0006366160541 f \pi}$$

      qx := .4999999998
> pz:= f -> abs(evalf(gz));
      pz := f -> |evalf(gz)|
```

R1=r10=12.566444 の場合には、Q=0.5 になります。

伝達関数は、 $\frac{s^2+39478878.7998421}{s^2+12566.44402 \cdot s+39478878.7998421}$ になります。



阻止域は f=400Hz～3KHz 程度に狭くなっています。

サンプルバッチファイルの回路図と説明

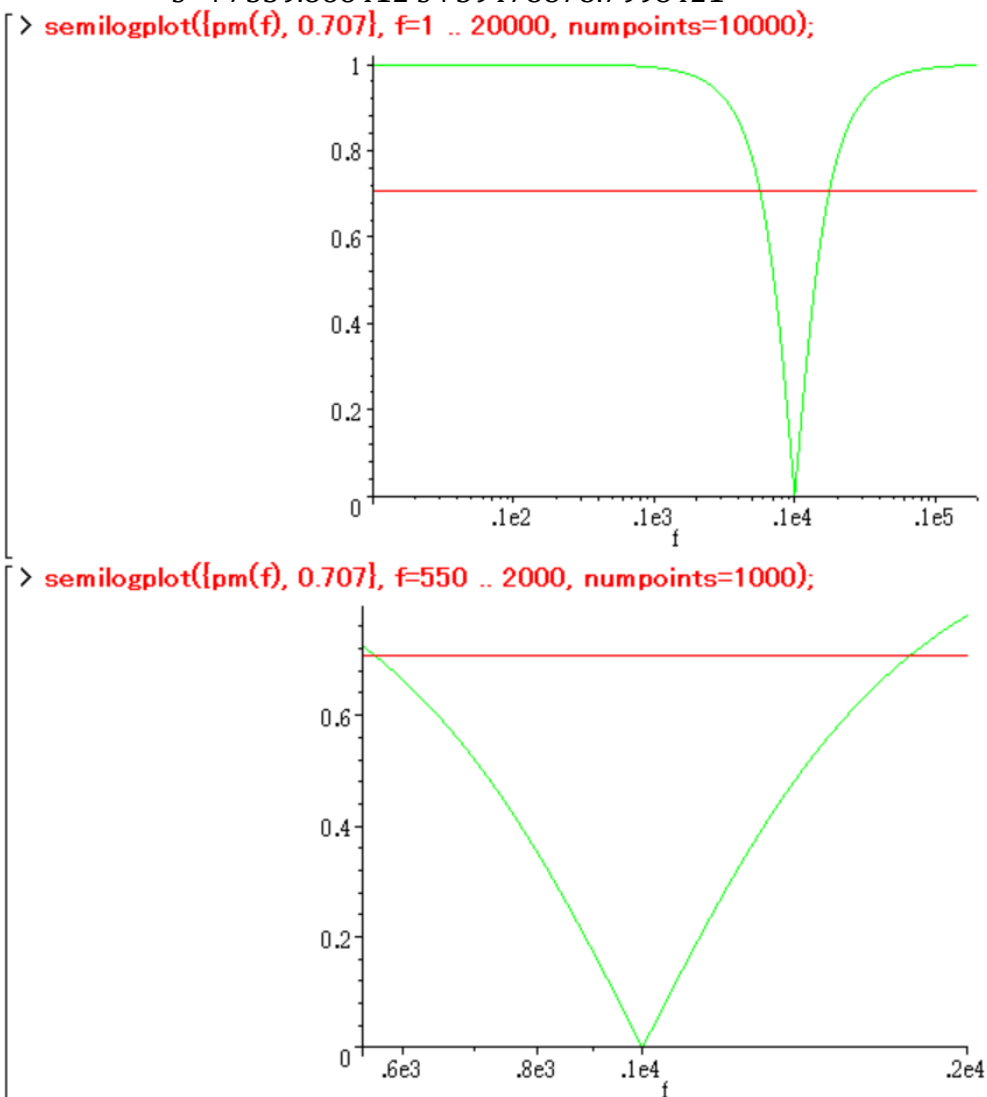
D<0 となる $R1=0.6*r10=7.5398664$ による、周波数特性を確認する。

```
> gm:=subs(r1=0.6*r10,g); qx:=subs(r1=0.6*r10,q);
gm := 
$$\frac{-1.013200000 \cdot 10^{-6} \pi^2 f^2 + 1}{-1.013200000 \cdot 10^{-6} \pi^2 f^2 + 1 + .0003819696324 i \pi f}$$

qx := .8333333330
> pm:= f -> abs(evalf(gm));
pm := f -> |evalf(gm)|
```

$R1=0.6*r10=7.5398664$ の場合には、 $Q=0.5$ になります。

伝達関数は、 $\frac{s^2+39478878.7998421}{s^2+7539.866412 \cdot s+39478878.7998421}$ になります。



阻止域は $f=550 \sim 2\text{KHz}$ に狭くなりました。

サンプルバッチファイルの回路図と説明

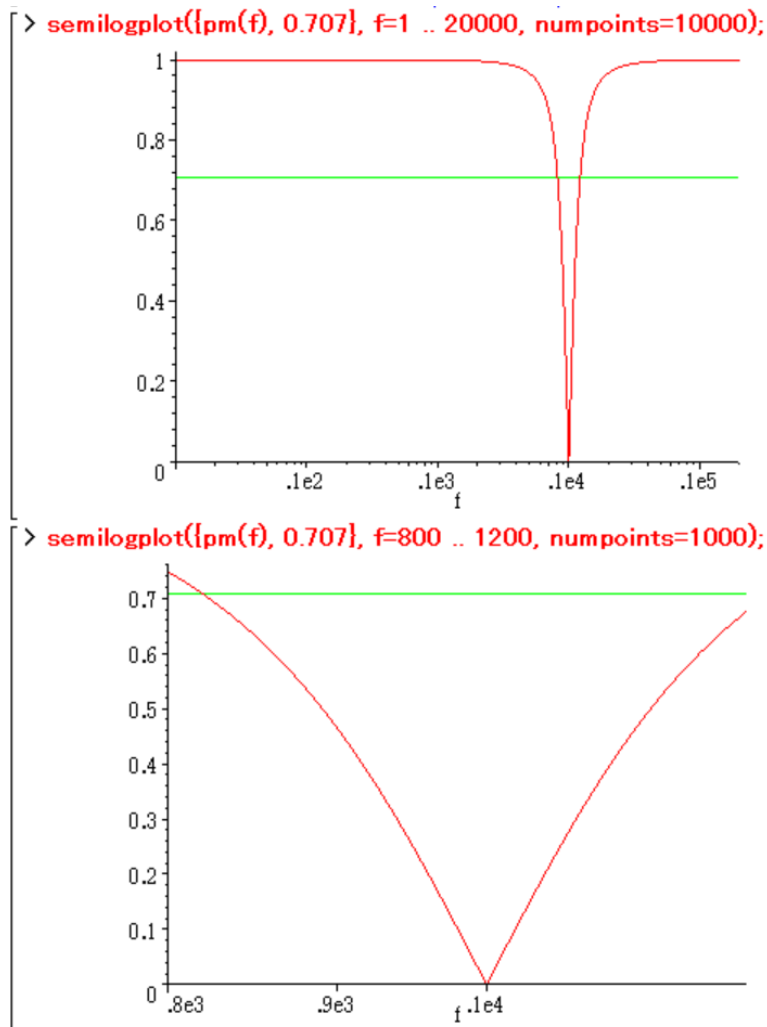
D<0 となる $R1=0.2*r10=2.5132888$ による、周波数特性を確認する。

```
> gm:=subs(r1=0.2*r10,g); qx:=subs(r1=0.2*r10,q);
      gm := 
$$\frac{-1013200000 \cdot 10^{-6} \pi^2 f^2 + 1}{-1013200000 \cdot 10^{-6} \pi^2 f^2 + 1 + .0001273232108 i \pi f}$$

      qx := 2.499999999
> pm:=f->abs(evalf(gm));
      pm := f->|evalf(gm)|
```

$R1=0.2*r10=2.5132888$ の場合には、 $Q=2.5$ になります。

伝達関数は、 $\frac{s^2+39478878.7998421}{s^2+2513.288804 \cdot s+39478878.7998421}$ になります。



阻止域は $f=800\text{Hz} \sim 1200\text{Hz}$ ($400\text{Hz} = 1000 / 2.5$) に狭くなりました。

$R1=0.001*r10=0.012566$, $Q=500$ にすると、阻止域は 2Hz ($1000 / 500$) になります。

バンドエリミネーションフィルタの阻止域は $\frac{f_0}{Q}$ に従って狭くなります。

数式処理ソフトを sim.exe と併用することを推奨します

Maple のような数式処理ソフトを利用すると、sim.exe で作成した係数行列を利用して、回路の動作をさらに詳しく調べることが出来ます。

しかし、Maple や Mathematica は非常に高価（数十万円）なので、利用できない場合にはフリーソフトを利用することをお勧めします。

48,000 円で購入できる、パーソナルライセンスという購入方法があります。

<https://www.cybernet.co.jp/maple/purchase/license/personal.html>

Maxima というフリーソフトが下記で紹介されています。

<http://maxima.osdn.jp/maxima.html>

<http://maxima.zuisei.net/>

<http://maxima.osdn.jp/maxima.html>

<http://www.math.kobe-u.ac.jp/HOME/taka/2007/knx/maxima.pdf>

<http://www.ac.cyberhome.ne.jp/~konoha/Math/books/ManualBook.pdf>

<https://kougaku-navi.net/maxima/index.html>

R というフリーソフトについて下記で紹介されています。

ライセンスは GPL で、Windows/Mac/Linux で動作します。

<http://rikeizine.hatenablog.com/entry/2014/09/14/235706>

他にもあるようなので、インターネットで検索してみてください。

私は、30 年以上前に、

アナログフィルタの伝達関数を回路の素子値を決定する数式に変形するために、Reduce という数式処理ソフトを使っていました。コマンドラインベースのソフトでした。

その後、Windows 用の Mathematica をしばらく利用してから、Mathcad というソフトを少しの間利用して、最終的に Maple に変更して現在も使い続けています。

Mathematica を利用する場合

/conv で作成した係数行列を Maple ではなく、Mathematica で利用したい場合には conv.coe の数式を次のように変更すれば利用することが出来ます。

```
sk=Array[siki,{5,5}]↓
ee=Array[e,{5}]↓
xx=Array[x,{5}]↓
Do[siki[i,j]=0,{i,5},{j,5}]↓
Do[e[i]=0,{i,5}]↓
port=2↓
siki[ 1, 1] = +1/r1 ;↓
siki[ 1, 3] = -1/r1 ;↓
e[1] = -i1 ;↓
siki[ 2, 2] = +1/r3+1/r2 ;↓
siki[ 2, 3] = -1/r2 ;↓
siki[ 2, 4] = +1/r4 ;↓
siki[ 2, 5] = -1/r4 ;↓
siki[ 3, 1] = -1/r1 ;↓
siki[ 3, 2] = -1/r2 ;↓
siki[ 3, 3] = +1/r1+1/r2 ;↓
siki[ 4, 2] = -1+w1*1/r2 ;↓
siki[ 4, 3] = -w1*1/r2 ;↓
siki[ 4, 4] = 1 ;↓
siki[ 5, 5] = 1 ;↓
e[5] = +e1 ;↓

restart: with(linalg):↓
siki:=matrix(3,3,0):e:=vector(3,0):↓
siki[ 1, 1] := 1 ;↓
e[1] := +e1 ;↓
siki[ 2, 1] := -1/r1 ;↓
siki[ 2, 2] := +1/r1+1/r1b1+s*c1+1/r2 ;↓
siki[ 2, 3] := -s*c1-1/r2 ;↓
siki[ 3, 2] := -v1b1 ;↓
siki[ 3, 3] := 1 ;↓
x:=linsolve(siki,e);↓
↓
e1 := 1 ;↓
r1 := 1000 ;↓
r2 := 10000 ;↓
c1 := 1e-007 ;↓
v1b1 := 1e+006 ;↓
r1b1 := 1e+006 ;↓
```

Mathematica 用係数行列の例
(24 年前に作成したファイル)

Maple 用係数行列の例
(2020 年に作成したファイル)

2つの例を見比べると、

配列作成の部分は、相違があるので Mathematica のマニュアルなどを参考にして下さい。
siki[1, 1] 以降は、Maple の「:=」を「=」に変更すれば良いと思います。

元々、/conv の出力は Mathematica 用のファイルを出力していましたが、20 年以上前から Maple を使用するようになって、Maple 用のファイルを出力するように変更しました。

Mathematica で使うためには、何かの solve コマンドを入力する必要があると思いますが、忘れてしまいました。Mathematica のマニュアルを参考にして下さい。

サンプルバッチファイルの回路図と説明

フィルタ回路の参考資料

各種フィルタ回路の設計・製作に興味がある方は、下記より「LtAct」をダウンロードしてください。 <https://www.vector.co.jp/vpack/browse/person/an008575.html>

フィルタの種類は、

ローパス、ハイパス、バンドパス、バンドエリミネーション

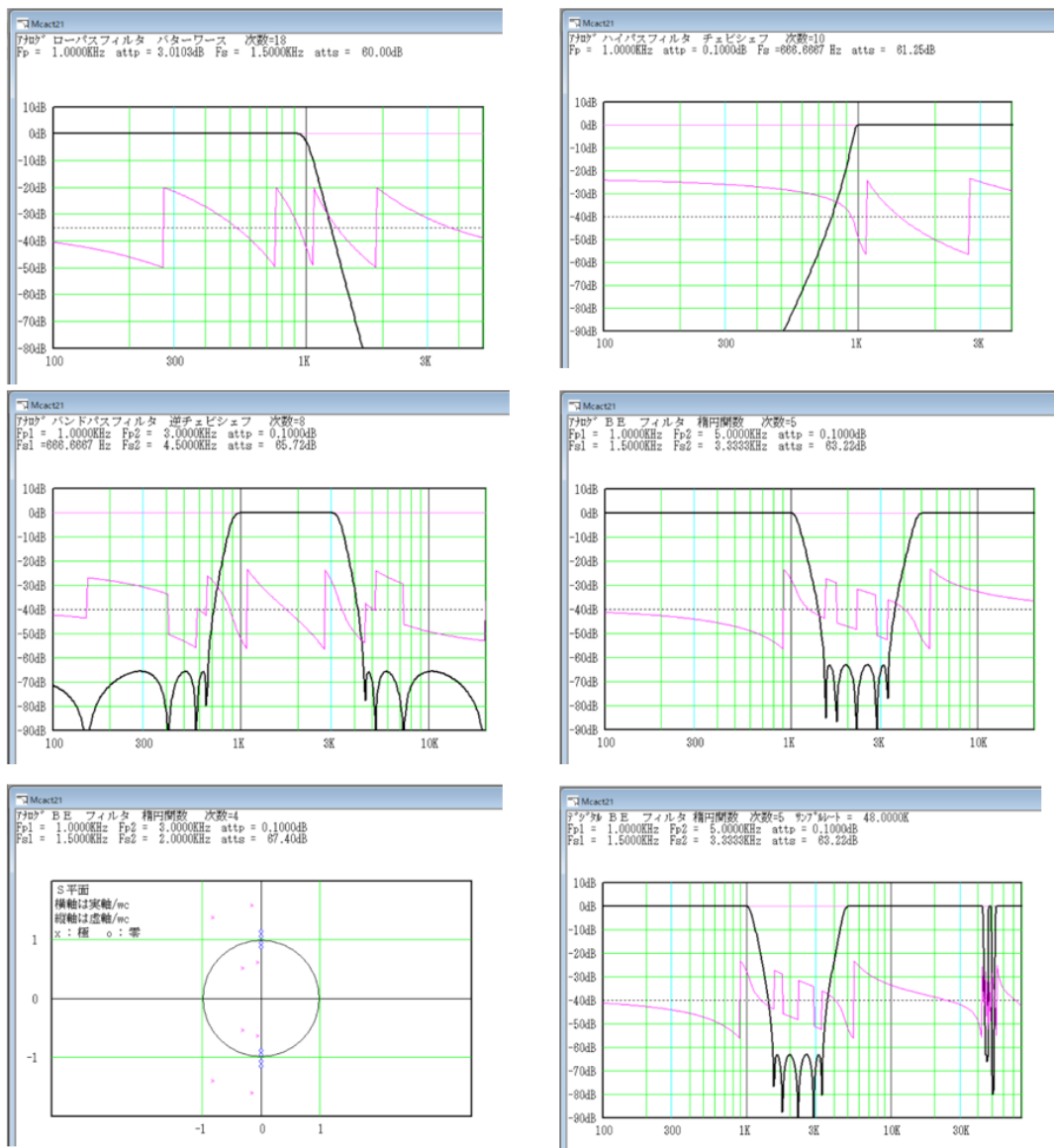
遮断特性は、

バターワース、チェビシェフ、逆チェビシェフ、楕円関数

計算機能は、

周波数特性、極・ゼロ、アナログ回路へ変換、デジタルフィルタへ変換、

回路図出力、伝達関数の係数表示・出力



サンプルバッチファイルの回路図と説明

5 次の楕円関数 BE フィルタの設計例

(阻止域：1KHz～5KHz 通過域のリプル 0.01dB)

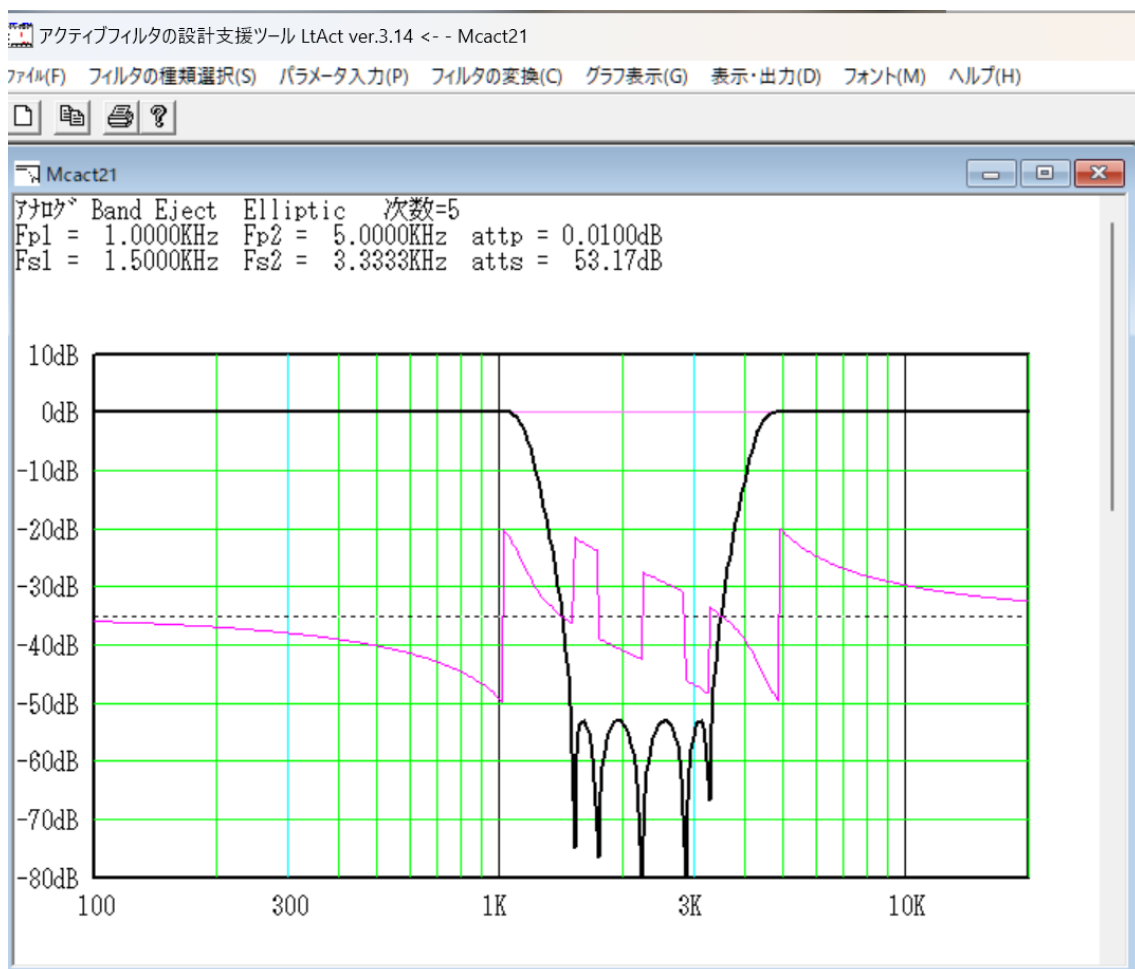
パラメータ入力画面

設計パラメータの入力

フィルタの種類	BE フィルタ			遮断特性	Elliptic
設計するフィルタの次数 $m(<=58)$	5				
阻止帯域 下端の周波数 F_{p1} : ($F_{s1} = F_{p1} \times x_s$)	1	KHz			
阻止帯域 上端の周波数 F_{p2} : ($F_{s2} = F_{p2} / x_s$)	5	KHz			
周波数 F_{p1}, F_{p2} における減衰量又はリプル att_p	0.01	dB			
最低減衰量に達する周波数を F_{s1} として、 $x_s = F_{s1} / F_{p1}$ を次の範囲で					
入力して下さい	1 < x_s < 2.2361	1.5	倍		

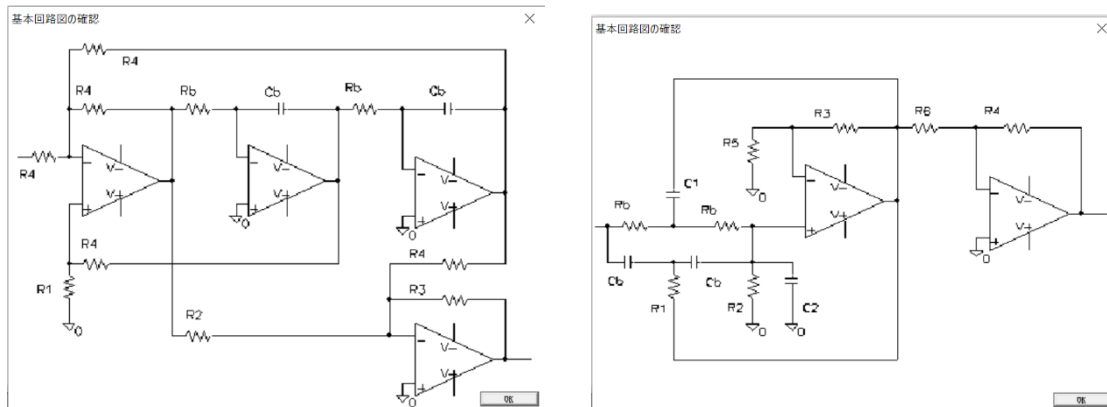
OK キャンセル

入力パラメータにおける周波数特性



サンプルバッチファイルの回路図と説明

回路図を出力する時に基本回路を選択する画面



入力パラメータに対する 5 次の楕円関数 BE フィルタの伝達関数の係数

+++++++ 伝達関数の係数、カットオフ周波数、Q 値、GB 値 +++++++

アナログ Band Eject Elliptic 次数=5

Fp1 = 1.0000KHz Fp2 = 5.0000KHz attp = 0.0100dB

Fs1 = 1.5000KHz Fs2 = 3.3333KHz atts = 53.17dB

2 次式の形式

$$Pn_2 * s^2 + Pn_3 * s + Pn_4$$

$$Hn = \frac{Pn_2 * s^2 + Pn_3 * s + Pn_4}{s^2 + Pn_0 * s + Pn_1}$$

$$s^2 + Pn_0 * s + Pn_1$$

2 次式

n	Pn_0	Pn_1	Pn_2	Pn_3	Pn_4
1	27.8606K	197.3921Meg	1.0000	0	197.3921Meg

Fc= 2.2361KHz Fc ゲイン= - 1.#INFdB Q = 0.5043 GB 積=112.7612KHz

2	5.4774K	758.2396Meg	1.7799	0	576.5538Meg
---	---------	-------------	--------	---	-------------

Fc= 4.3825KHz Fc ゲイン= 14.1945 dB Q = 5.0272 GB 積=2.2032MegHz

3	1.4259K	51.3870Meg	0.7604	0	91.4639Meg
---	---------	------------	--------	---	------------

Fc= 1.1409KHz Fc ゲイン= 14.1945 dB Q = 5.0272 GB 積=573.5568KHz

4	23.1635K	773.9904Meg	1.1610	0	492.5773Meg
---	----------	-------------	--------	---	-------------

Fc= 4.4278KHz Fc ゲイン= - 4.0124 dB Q = 1.2011 GB 積=531.8048KHz

5	5.9074K	50.3412Meg	0.6364	0	58.4460Meg
---	---------	------------	--------	---	------------

サンプルバッチファイルの回路図と説明

Fc= 1.1292KHz Fc ゲイン= - 4.0124 dB Q = 1.2011 GB 積
=135.6271KHz

決定された抵抗とコンデンサの素子値

***** 回路の構成と素子値 *****

回路図ファイル名 E:\LT test\BE2_5-ellip-1KHz-5KHz-atp-0.01.asc 作成日時 Tue
Apr 23 08:18:28 2024

アナログ Band Eject Elliptic 次数=5

参照モード=0

Fp1 = 1.0000KHz Fp2 = 5.0000KHz attp = 0.0100dB

Fs1 = 1.5000KHz Fs2 = 3.3333KHz atts = 53.17dB

1 (et2) 「HP4-0-2」 Rb_1(2 個)= 15.1439K Cb_1(2 個)= 4.7000n R1_1 = 7.5719K
C1_1 = 9.4000n 誤差=3.46 %

1 R2_1 = 237.2542 C2_1 = 0.3000u 誤差 = 1.16 %

1 R3_1 = 464.2243 R5_1 = 10.0000K 誤差 = 1.24 %

1 R4_1 = 1.2295Meg R6_1 = 10.0000K 誤差 = 2.40 %

2 (et2) 「HP4-0-0」 Rb_2(2 個)= 16.8370K Cb_2(2 個)= 3.3000n R1_2 = 8.4185K
C1_2 = 6.6000n 誤差=6.37 %

2 R2_2 = 15.0000K C2_2 = 0.6373n 誤差 = 2.72 %

2 R3_2 = 5.5232K R5_2 = 10.0000K 誤差 = 1.39 %

2 R4_2 = 15.8948K R6_2 = 10.0000K 誤差 = 0.66 %

3 (et2) 「LP4-0-0」 Rb_3(2 個)= 9.1178K Cb_3(2 個)= 10.0000n R1_3 = 4.5589K
C1_3 = 20.0000n 誤差=3.10 %

3 R2_3 = 3.4820K C2_3 = 68.0000n 誤差 = 3.39 %

3 R3_3 = 42.3475K R5_3 = 10.0000K 誤差 = 1.54 %

3 R4_3 = 21.2075K R6_3 = 10.0000K 誤差 = 3.74 %

4 (et2) 「HP4-0-2」 Rb_4(2 個)= 17.9810K Cb_4(2 個)= 2.7000n R1_4 = 8.9905K
C1_4 = 5.4000n 誤差=3.90 %

4 R2_4 = 30.0000K C2_4 = 0.2771n 誤差 = 2.56 %

4 R3_4 = 1.2153K R5_4 = 100.0000K 誤差 = 1.26 %

4 R4_4 = 13.8250K R6_4 = 10.0000K 誤差 = 5.97 %

5 (et2) 「HP4-0-2」 Rb_5(2 個)= 13.9133K Cb_5(2 個)= 7.5000n R1_5 = 6.9567K
C1_5 = 15.0000n 誤差=6.94 %

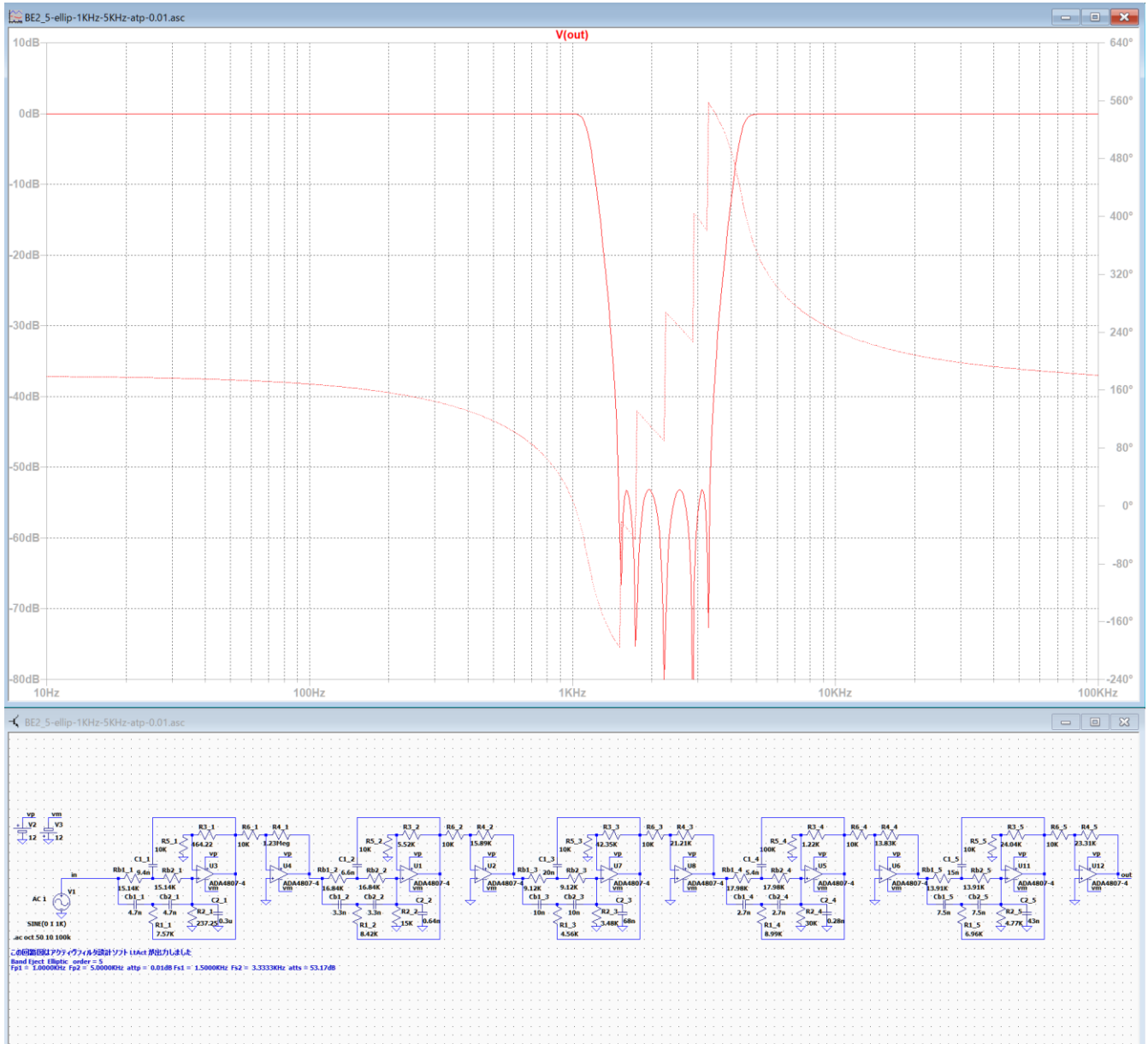
5 R2_5 = 4.7700K C2_5 = 43.0000n 誤差 = 1.47 %

5 R3_5 = 24.0387K R5_5 = 10.0000K 誤差 = 0.16 %

5 R4_5 = 23.3086K R6_5 = 10.0000K 誤差 = 2.97 %

サンプルバッチファイルの回路図と説明

LtAct を使用して、5 次の楕円関数 BE フィルタ（阻止域：1KHz～5KHz 通過域のリプル 0.01dB）を設計して、フリーで利用可能な Spice シミュレーション・ソフトウェア「LTSpice」用の回路図ファイル(BE2_5-ellip-1KHz-5KHz-atp-0.01.asc)を出力して、「LtAct」で作成された回路図と、その周波数特性を「LTSpice」で確認しました。



Vector で公開中のソフトとデータ

Vector で公開中のソフトとデータ

作者: 三浦 高志 (vector.co.jp)

汎用データ/画像&サウンド

- [キャプチャ NX の使用方法](#)

ニコンキャプチャ NC と NX の使用方法の説明

- [キャプチャ NX の画像調整データ](#)

「キャプチャ NX の使用方法」の本文中で使用した画像と調整データ

- [ニコンキャプチャ NX2-調整手順](#)

ニコンキャプチャ NX2 の操作に慣れるための調整手順を説明

- [ニコンキャプチャ NX2-撮影練習](#)

ニコンデジタル一眼レフカメラの撮影練習 — 露出設定を重点的に説明

汎用データ/学習&教育

- [LTSpice 操作入門](#)

アナログ電子回路のシミュレーション操作入門

WindowsMe/98/95 用ソフト/学習&教育

- [Sim for DOS](#)

アナログ回路の DC 及び AC 解析の出来るシミュレーションソフト

- [SimPack](#)

フリーソフトとして公開中の Sim.exe の開発資料とプログラムソースを公開する

Windows10/8/7/Vista/XP/2000/NT/画像&サウンド

- [ぬり絵ビルダー](#)

画像データ(BMP または JPG)を自動的にぬりえに変換する

Vector で公開中のソフトとデータ

Windows10/8/7/Vista/XP/2000/NT/学習&教育

- [McAct2W](#)

アナログフィルタ(回路図出力付き)及びデジタルフィルタの設計支援ソフト

- [ActiveFilter-Design-Schematics](#)

LtAct の名称を変更し、取扱説明書の一部を英文に翻訳しました

- [ActDoc](#)

フリーソフトとして公開中の McAct2W.exe のプログラム開発資料を公開する

- [LtAct](#)

アクティブフィルタの設計と LTSpice 用の回路図作成

Windows10/8/7/Vista/XP/2000/NT/パーソナル

- [電卓プログラム Dt_.exe](#)

複素数の計算が出来て、関数も自作できる 組み込み関数は複素数処理に対応

Windows10/8/7/Vista/XP/2000/NT/画像&サウンド

- [カラーコーディネイタ](#)

色彩調整における「色と補色の確認」ツール

- [TIFF 圧縮](#)

RGB 各 16 ビットの TIFF データを高画質に圧縮/伸張するプログラム

- [EV 計算](#)

デジタル一眼レフの露出情報を入力すると EV 値(撮影環境の明るさ)を計算する

Vector で公開中のソフトとデータ

汎用データ/学習&教育

- [英語-地球の歴史](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

- [星の王子さま-スペイン語](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

- [星の王子さま-ドイツ語](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

- [異邦人-スペイン語](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

- [星の王子さま-英語](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

- [スペイン語学習](#)

構文解析によるスペイン語学習

- [星の王子様-フランス語](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

- [フランス語学習](#)

構文解析によるフランス語学習

- [星の王子様-イタリア語](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

- [英語-アル・ゴアからのメッセージ](#)

気候危機に関するアル・ゴア元副大統領の講演

- [異邦人-フランス語](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

- [異邦人-英語](#)

辞書なしでも、何度も読むだけで単語と文法が分かってくると思います

Vector で公開中のソフトとデータ

Windows10/8/7/Vista/XP/2000/NT/画像&サウンド

●[HDPPhotoTool \(HD フォトツール\)](#)

TIF(RGB48bit または RGB24bit)または BMP ファイルを HDP 形式に圧縮して TIF 形式に伸張

汎用データ/家庭&趣味

●[北海道の家庭菜園](#)

寒冷地における苗の植付と栽培方法および野菜の育苗を体験に基づいて説明