オペアンプの周波数特性測定用プログラムの開発とその使用方法

Development of a Program for Measuring Frequency Characteristics of an Operational Amplifier and its Usage

袴田　吉朗*　　郡　武治**

Yoshiro　HAKAMATA　　Takeharu　KOHRI

Abstract: In the Information and Electronic Experiment of the Electrical and Electronic Engineering, the experiment on a negative feedback amplifier has been performed more than about ten years. In the experiment, frequency characteristics of the amplifier have been measured to cause much experiment time. To reduce the time for the measurement, a program for measuring frequency characteristics of an operational amplifier has been developed. To develop the program, VC++6.0 integrated development environment was used. The program uses RS232C interface for a PC to connect a generator and an oscilloscope. In the paper, design concept for a program and its usage are described.

1．　はじめに

電気電子工学科では情報・通信コースにおける3年生のコース実験において，PBLを指向した「負帰還増幅器の設計および特性測定」，「マルチバイブレータの設計と試作」と題する2つの回路の設計と，試作を行わせてきた．いずれも6コマ（2日間）の実験である．H23年度までは穴あきプリント基板上に個別トランジスタを用いて回路を製作させてきた．この方法は製作の自由度は非常に大きいものの，逆に回路製作の経験の乏しい学生達にとっては製作時間が非常に長くなる傾向があった．また予期した回路の特性が得にくいこともあり，この意味において必ずしも教育的とは言えず改善すべき余地が残されていた．

PBLの重要性に鑑みて本格的なPBLを行うべくH24年度から「H8マイコンを用いた電子回路の設計と試作」の項目6コマ（2日間）を新設し，実施してきた．この時間を捻出するために，「マルチバイブレータの設計と試作」における素子を個別トランジスタからNANDゲートの使用に変更しこれにより時間を3コマに半減させた．またTV放送がディジタル放送に切り替わったことを契機として，NTSC方式のビデオ信号に関する実験を廃止し，これにより3コマを捻出した．一方「負帰還増幅器の設計および特性測定」についても，H25年度以降素子を個別トランジスタからオペアンプに代え，これにより時間の短縮に努めてきたところである．

ここ2年間の実施状況を見てみると，依然として製作に多くの時間がとられて時間内に終わらないこともあり，冷静に回路の特性を吟味することに時間を割きにくいという状況が続いてきた．これに対処すべく今年度はプリント基板を設計，試作しこれを用いて製作を行わせることにした．これにより時間短縮と，製作した回路の特性が製作した学生によって著しく変わることのないという教育的効果を達成できたものと考えている．

負帰還増幅器については重要な特性である利得の周波数特性を測定させてきたが，測定周波数個数が多いことが時間のかかる一原因になっていた．そこで手持ちの測定器を使用して自動測定を行うこととし，今日ではレガシーインタフェースと呼ばれているRS232Cインタフェースを装備した

・アジレント社のオシロスコープ　54603B

・岩通社の発振器　SG-4105

を使用して自動測定システムを構成することにした．制御プログラムは手持ちのVC++6.0を使用し，MFCを用いて試作することにした．本プログラムの作成を開始したのはH25年における5月の連休明けである．昨年度の実験においては，利得周波数特性の測定およびデータの収集はできていたが，まだ改善点が多数残されていた．今年度は5グループの学生実験を経て，その都度不具合な点を見直して最終的に完成させるに至った．

本プログラムは，オペアンプを用いた非反転増幅器，反転増幅器および開ループ利得の周波数特性を測定できるものであり，比較的汎用性の高いものである．少なくとも実験におけるオペアンプの周波数特性の測定には十分に適用可能である．そこで今後のことを考えてプログラムの考え方およびその使用方法を資料にまとめておくことにした．

2．　オペアンプの周波数特性測定回路

2.1　オペアンプを使用した負帰還増幅器の回路およびプリント基板

図2.1，2.2および図2.3に非反転増幅器，反転増幅器および開ループ利得測定回路の回路図を示す．開ループ利得測定回路は40dBの減衰器を挿入［1］することにより約100dBの開ループ利得の測定を可能にしている．

図2.3においてJ点の電圧をV_J，オペアンプのマイナス端子における電圧をV_{in}とすればR_3およびR_4より構成される減衰器の減衰量を考慮すると式(1)が成立する．

$$V_{in} = -\frac{V_o}{A_0} = \frac{R_4}{R_3 + R_4} V_J \qquad (1)$$

式(1)より式(2)(3)が成り立つ．

$$\frac{V_o}{V_J} = -A_0 \frac{R_4}{R_3 + R_4} \qquad (2)$$

$$A_0 = -\frac{V_o}{V_J} \frac{R_3 + R_4}{R_4} \qquad (3)$$

2015年1月27日受理

*,**理工学部　電気電子工学科

式(3)より V_J, V_O を測定できれば開ループ利得 A_0 を測定でき

図 2.4 は試作したプリント基板を部品面から見た図である．1 枚のプリント基板上に 3 種類の回路を製作できるようにした．入力端子は 2 個の抵抗を選択して並列接続できるようにし，入出力特性測定時に使用する 600Ω 系発振器と周波数特性測定時の 50Ω 系発振器の両方の特性インピーダンスに適合させられるようにした．

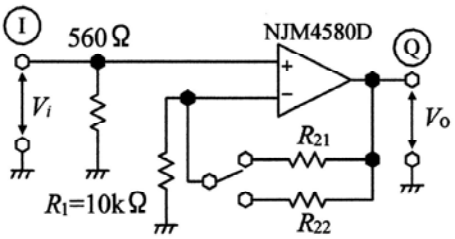


図2.1　非反転増幅器

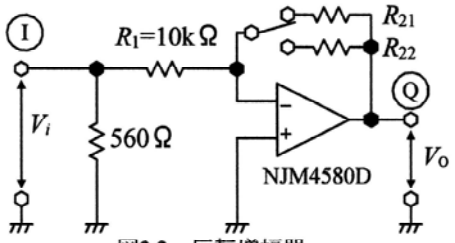


図2.2　反転増幅器

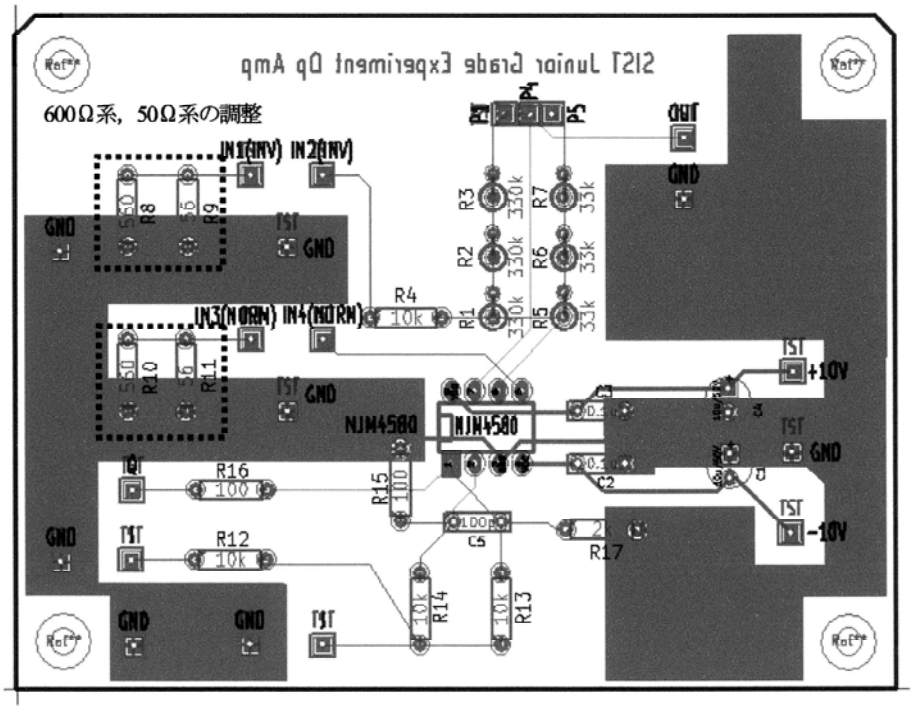


図 2.4　試作プリント基板の部品面（120×94mm²）

2.2　周波数特性測定回路のブロック構成

図 2.5 に周波数特性測定回路のブロック構成を示す．構成要素は以下の通りである．

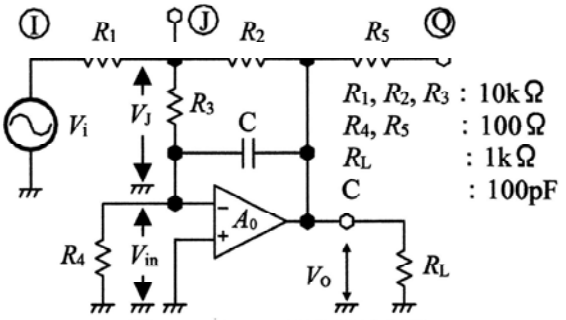


図2.3　開ループ利得測定回路

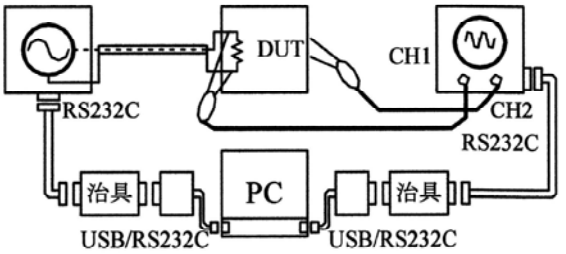


図2.5　周波数特性測定システムのブロック構成

①オシロスコープ　アジレント社の 54603B（外付けの 54659B RS232C インタフェース回路を装備）

②SG-4105 発振器　　岩通社製（RS232C インタフェース回路を標準装備）

③PC　XP マシンに VC++6.0 をインプリメントすればプログラムの開発ができる．XP，Win7 でプログラムの動作を確認

④RS232C ストレートケーブル 2 本

⑤RS232C／USB 変換アダプタ 2 個

⑥RS232C／USB 変換治具　④と⑤を接続するための両端がメスーメス・コネクタになっている手作りの治具，クロス接続

2.3　使用 PC

プログラム開発用の言語として VC++6.0 プロフェッショナルエディションを使用した．MFC を使用しているため使用できる PC は以下に限定される．

・プログラムの開発　WinXP

・プログラムの実行　WinXP,Win7

3．プログラムの構成

FormView を基本クラスとして作成した．図 3.1 にプログラム起動時点における初期画面の様子を示す．特性測定，利得表示および位相表示の 3 個のボタンと，回路の選択およびデータをファイルから読み込むための測定器設定および RS232C を設定するメニューを使用している．

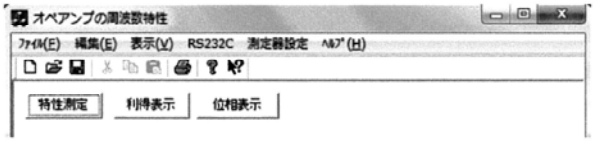


図 3.1 プログラムの初期画面（一部のみ表示）

3.1 グローバル・オブジェクトの宣言

下記のグローバル・オブジェクトを宣言している．またこれ以外にドキュメントクラスに多数の変数を宣言している．

・RS232C クラス rs_oscillo, rs_gene ・・・COM ポートを介したオシロスコープおよび発振器との通信

・CDlg クラス dlg, dlg_Gene ・・・オシロスコープおよび発振器の COM ポート設定用ダイアログの起動

・CDlg_init_new クラス dlg_init_new・・・回路選択（回路種類，オシロスコープの電圧軸，発振器の出力電圧）用ダイアログの起動

・CDlg_Plot クラス プロットデータ選択用ダイアログを起動

上記以外に測定データをリスト構造で保存するために以下の変数を宣言している．

・ampData クラスへのポインタ head=NULL

3.2 測定回路の選択

メニュー／測定器設定／回路の選択をクリックすると図 3.2 のダイアログが起動するので，このダイアログにおいて測定回路，増幅器の利得を選択する．その結果に応じて表 3.1 に示すように，オシロスコープの感度，発振器の出力電圧が設定される．最初のうちは，プルダウンメニューを開き測定者が選択できるようにしていたが，間違って入力すると測定がうまくできず余分な時間がかかるので，自動的に入力されるようにした．

また測定者の苗字を入力し，コメントとして出力するようにして測定したデータを明確に区別できるようにした．

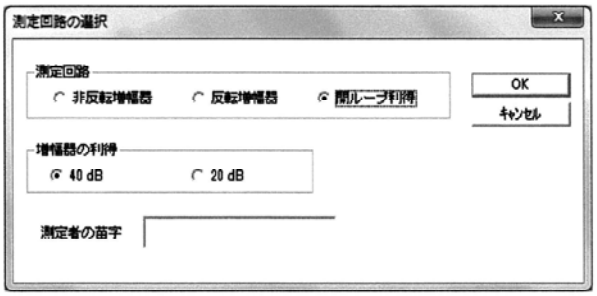


図 3.2 測定回路の選択画面

表 3.1 オシロスコープおよび発振器の設定

	オシロスコープ		発振器
	CH1	CH2	出力電圧
開ループ利得	20mV/div	2V/div	7Vpp
40dB AMP	20mV/div	1V/div	50mVpp
20dB AMP	100mV/div	1V/div	500mVpp

3.3 COM ポートの設定

プログラムが起動すると View クラスにおける OnDraw( )関数が実行される．OnDraw( )関数のフローチャートを図 3.3 に示す．プログラム起動時には メニュー／RS232C／切断 に設定されており，この状態では単にステータスバーに"切断"を表示しているだけである．メニュー／RS232C／送信をクリックすると COM ポートの設定が行われる．まず RS232C クラスの

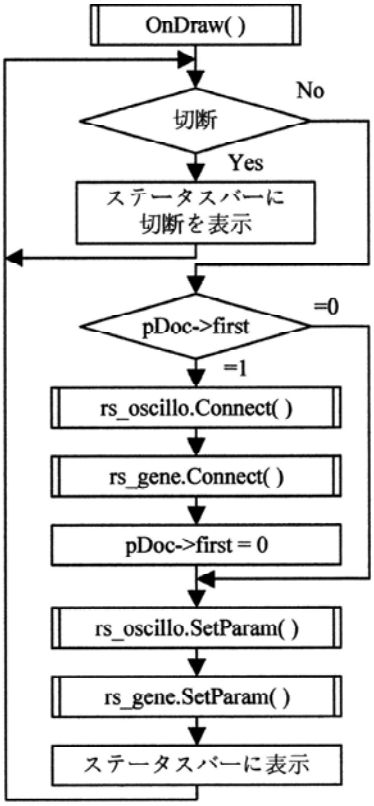


図 3.3 OnDraw()関数のフローチャート

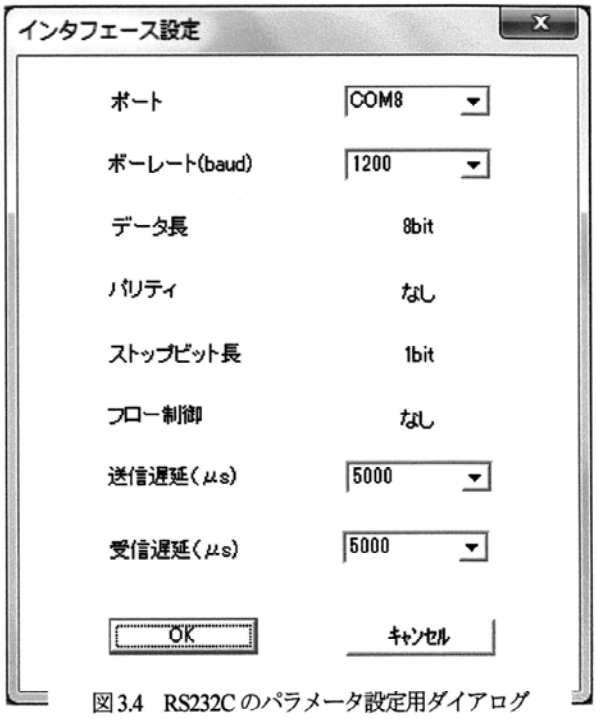


図 3.4 RS232C のパラメータ設定用ダイアログ

Connect(COM_port)関数を実行してオシロスコープおよび発振器のRS232Cインタフェースの設定を行う．引数COM_portは使用するCOMポートの名前である．メニュー／RS232C／通信設定／オシロスコープ（あるいは発振器）を選択すると，ボーレート，ビット長，送信遅延および受信遅延の4パラメータを図 3.4 に示すダイアログから設定できる．この結果はRS232C.INIおよびRS232C_gene.INIファイルに保存し読み出すようになっている．Connect()関数を複数回実行するとエラーになるので，プログラム起動時に最初に1回だけ実行する．

COMポートを接続替えした場合には番号を変更する必要がある．この変更は"切断"時にのみ可能であり，RS232Cクラスの SetParam( )関数を実行することにより変更が反映される．なお，COMポートの未接続，誤接続時にはアラートを発出し警告するようにしている．

3.4 特性測定ボタンのハンドラー関数

3.4.1 オシロスコープおよび発振器との通信

RS232Cクラスのオブジェクト rs_oscillo および rs_gene のメンバ関数である Send( ), Read_CRLF( )を用いて以下のようにオシロスコープあるいは発振器を制御する．

```
・rs_oscillo.Send(cmd);          // コマンド cmd の送信
・rs_oscillo.Read_CRLF(a$);      // アスキー文字の受信
・rs_gene.Send(cmd);             // コマンドの送信（発振器）
```

本プログラムで使用している具体的なコマンドを表 3.2 に示す．PCにデータを取り込むためには，表3.2のコマンドに？を付加したクウェリをまず送信し，その後にRead_CRLF(a$)コマンドを送信して文字列a$を取り込む必要がある．

表 3.2 コマンドの書式

コマンド	意味
*RST	リセット
:AUTOSCALE	オートスケール
:MEAS:SOUR CHAN1	ソースをCH1に設定
:CHAN1:PROB X10	CH1のプローブを10倍
:CHAN1:RANG 0.4	CH1のフルスケールを0.4Vpp
:ACQ:TYPE AVER	アベレージング（8回）
CHAN1:OFFSET 0	CH1のオフセットを0
:CHAN1:COUP AC	CH1をAC結合
:CHAN1:BWL ON	CH1の帯域制限をON
:CHAN1:VIEW CHAN1	CH1を管面に表示
:TRIG:SOUR CHAN2	トリガー源をCH2に
:TIMEBASE:RANG 1.0	フルスケールを1sに設定する
:MEAS:VPP	電圧のpp値を測定
:MEAS:PHAS	位相を測定
:AMPL 0.2	発振器の出力を0.2Vppに設定
:FREQ 1.5E+00	発振器の周波数を1.5Hzに設定

測定器の制御では，一つのコマンドを送信してから次のコマンドを送信するまでの遅延時間の設定が重要である．遅延時間が短いと測定器が応答できない．しかし遅延時間が過大であれば，測定時間が長くなってしまう．本プログラムでは，Sleep(100), Sleep(1000), Sleep(2000)の3種類の遅延（単位 ms）を用いている．これらの値は試行錯誤的に決めたものであり，最適化したものではない．処理系への依存はないものと考えている．

3.4.2 初期設定

以下の順番に従って初期設定を行う．

① 増幅器入力における終端抵抗の確認

測定の順番として非反転増幅器および反転増幅器の入出力特性を600Ω系の測定器を使用して測定する．そのため増幅器の入力を560Ωで終端している．一方引き続いて行う周波数特性の測定時には特性インピーダンス 50Ω の発振器を増幅器に直接接続する．このため56Ωの抵抗を並列接続して50Ωになるように調整する．これを忘れるとうまく測定できない．このため最初に入力端子が 50Ω（あるいは560Ω と 56Ω の並列接続）になっているか確認して下さい。の警告を出している．

② オシロスコープのフルスケール，発振器の出力電圧の設定

表3.1で設定したディビジョン当たりの電圧（感度）を8倍してオシロスコープのフルスケールを求め，文字列に変換する．また発振器の出力電圧は単に文字列に変換する．

その後，発振器およびオシロスコープをリセットし2秒待つ．

③ オシロスコープの初期設定

チャネル1およびチャネル2ともプローブ倍率，フルスケール，カップリング（AC），帯域制限の設定を行う．

④ 発振器の初期設定

表3.1に示した出力電圧を発振器に設定し，出力をONにする．その後1秒待つ．

⑤ オシロスコープのアベレージング

測定回数をデフォルトの8回に設定した．その後1秒待つ．

⑥ オシロスコープのオフセット，カップリングの設定

オフセットを0に，カップリングをACに設定する．その後チャネル1を管面に表示させ，トリガーソースをチャネル2に設定する．

⑦ 測定開始直後の管面の表示

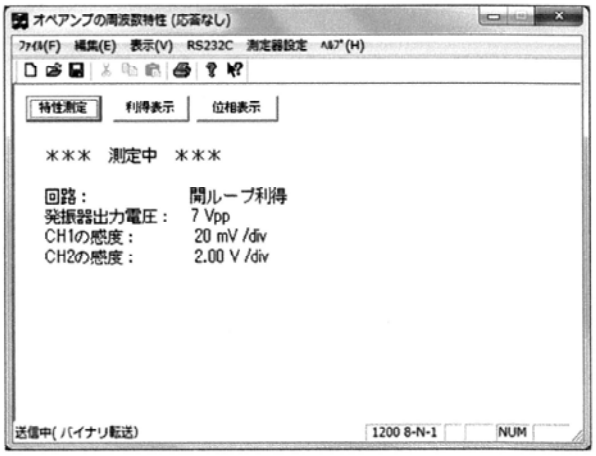


図 3.5 測定開始直後の管面表示

測定中のメッセージ，測定回路名，発振器出力電圧およびオシロスコープの感度（ディビジョン当たり）を図 3.5 のように管面に表示する．その後 1 秒待つ．

3.4.3 測定の流れ

測定ボタンのハンドラー関数のフローチャートを図 3.6 に示す．

① 周波数レンジ（:TIMEBASE:RANGE） の設定

測定周波数範囲は 1.5Hz～20MHz とした．したがって対数周波数軸上に測定データをプロットするとき，8 ディケードになる．対数周波数軸上でほぼ等間隔となるように，各ディケードの 1.5，2，3，5，7 および 10 に相当する周波数のみの測定を行わせるようにし，これを:TIMEBASE:RANGE str ; に反映させた（str は時間軸のフルスケール(s) ）．また予備測定の結果から増幅器の特性を予想できたので，変化の少ない帯域の測定を間引くようにし，測定する回路種類に応じて測定周波数を極力少なくなるようにして測定時間の短縮を図った．

② 電圧レンジの自動切り替え

オシロスコープの感度，フルスケールの初期値を表 3.3 に示す．この状態を変数 flag1 および flag2 で表し，波形がスケールオーバとなった場合には flag1 あるいは flag2 の値を大きくして感度を下げるようにした．また以下の条件を満足すると感度を上げるようにし，同時に flag1 あるいは flag2 の値を小さくするように制御した．

電圧の測定値<0.35*フルスケール

表 3.3 感度，フルスケールと flag の関係

感度（V／div)	フルスケール(V)	flag1, flag2
20×10^{-3}	160×10^{-3}	0
50×10^{-3}	400×10^{-3}	1
100×10^{-3}	800×10^{-3}	2
200×10^{-3}	1.6	3
500×10^{-3}	4	4
1	8	5
2	16	6
5	40	7
10	80	8
20	160	9
50	400	10

③ 電圧および利得の測定

電圧測定の書式を以下に示す．

```
cmd_oscillo(":MEAS:VPP");   // VPP の測定値を管面に表示
Sleep(100);                 // 待ち時間 100ms
cmd_oscillo(":MEAS:VPP?");  // VPP の測定値を問い合わせる
Sleep(100);                 // 待ち時間 100ms
enter_oscillo_asc(Vpp_in$); // 測定値（文字列）を PC に入力
```

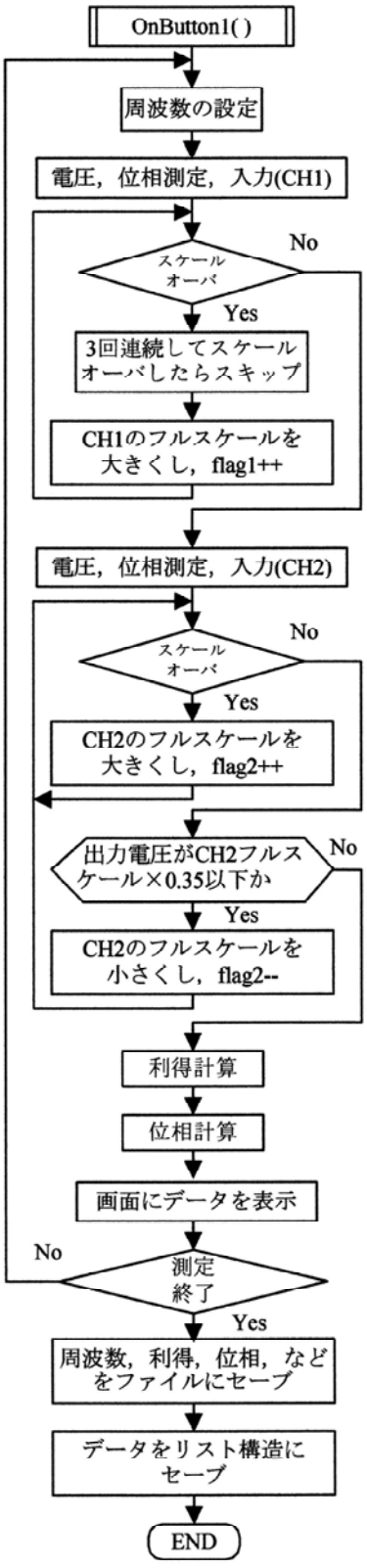


図 3.6 測定ボタンのハンドラー関数のフローチャート

利得は PC に入力した文字列 Vpp_in$および Vpp_out$を実数に変換し，常用対数をとってから 20 倍して求めた．なお開ループ利得は測定値に減衰器の減衰量 40dB を加えて求めた．

④ 位相差の測定

位相差測定の書式を以下に示す.

```
cmd_oscillo(":MEAS:PHAS");  // 位相差の測定値を管面に表示
cmd_oscillo(":MEAS:PHAS?"); // 位相差の測定値を問い合わせ
enter_oscillo_asc(Ph_out$);      // 測定値（文字列）をPCに入力
```

なお, 上記の測定値は「チャネル1の位相ーチャネル2の位相」となっており，実数値に変換するときに符号を反転している．また測定値が1E+37以上になるとスケールオーバとなるが，これが3回連続した場合には測定をスキップするようにした．

測定した位相差θを適切にプロットするために以下の変換を行った．

・非反転増幅器

θ＜－180・・・・・・θ=θ+180 (°)

－180≦θ＜180・・・ θ=θ

θ≧180・・・・・・・θ=θ－180 (°)

・反転増幅器，開ループ利得測定回路

θ＜0 ・・・・・・θ=θ+360 (°)

θ≧0 ・・・・・・・θ=θ (°)

⑤ 周波数，利得および位相差の測定結果を画面に表示

測定結果をCString型の変数に組み立て，図3.7に示すように画面に表示している．学生達はこの値を見て測定を行うことになる．当初エディットボックスに測定値を表示することを試みたが，うまく表示できなかった．

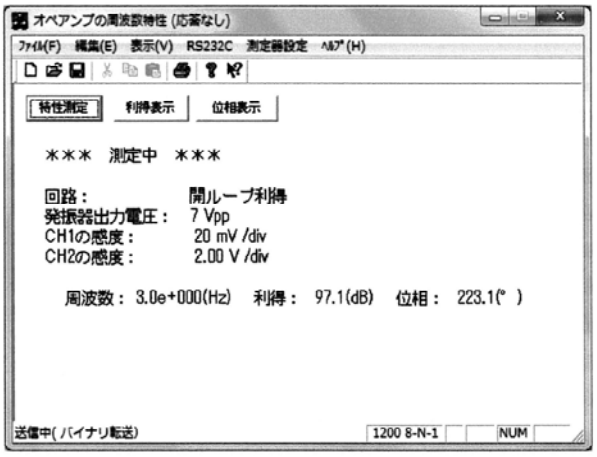


図3.7 測定結果の画面への表示

⑦ 測定結果のファイルへの保存

測定終了後，下記の一連のデータをファイルに保存するようにした．図3.8にセーブしたファイルの一例を示す．

・測定番号，保存年月日，回路種類，発振器出力，AMP利得，測定周波数，入力電圧，出力電圧，利得，位相差

ファイルを書き込むホルダーは，例えば測定年月日が2015年01月10日であれば，以下のようになる．experiment_LOGは自分で作成しなければならないが，測定年月日のホルダーがない場合は自動的に作成される．これはDOSコマンドの_mkdir(path);//pathを用いて行っている．

experiment_LOG¥2015_01_10¥ファイル名

ファイル名は以下のようになる．

exp+測定者の姓+回路+利得+2015_01_10+測定時刻.csv

なお測定が無限ループに陥った場合には，測定を途中で停止させる必要がある．この場合にもデータの消失を避けるために一つの周波数に対する測定の終了後に以下のファイルにデータを上書き保存する．

er+測定者の姓+回路+利得+2015_01_10.csv

情報通信実験におけるオペアンプの周波数特性測定結果

測定番号＝ 1
保存年月日 時刻 2014/11/20 17:43:36

回路種類＝ 反転増幅器
AMP利得＝ 40dB
発振器出力（初期値）＝50mVpp

番号	周波数 (Hz)	入力電圧 (Vpp)	出力電圧 (Vpp)	利得 (dB)	位相差 (°)
0	1.5	0.05	2.89	36.2	182
1	2	0.05	3.39	37.5	179
2	3	0.07	3.97	35.4	179
3	5	0.06	4.47	36.7	175
4	7	0.06	4.66	37.8	179

図3.8 セーブしたファイルの例（一部）

3.4.4 リスト構造によるデータの保存

3つの回路があるので，学生達は分担して回路を製作することになり，増幅器の周波数特性の測定も個々人が行うことになる．しかしながら開ループ利得，40dBの増幅器および20dBの増幅器の特性を一画面上にプロットして学生に見せることが，相互の利得の関係や帰還量の意味を理解させるためには必要である．そのためデータをリスト構造にして保存し，これを選択的に読み出して上記の目的を達成できるように工夫した．なお誤って測定を終了してしまった場合に備えて，図3.10に示すように測定毎にコメント，周波数，利得および位相のデータをリスト構造に保存し，ファイルに保存するようにした．

① 構造体Amp_data型の作成

C言語のテキストに掲載されているリスト構造の例は，単純変数を使用したものが通常である．調べた範囲では，配列を使用した例は見当たらなかったが，調べていくうちに構造体を用いれば良いとのヒントが得られた．そこで下記に示すように配列を含むデータをAmp_data型の構造体として宣言し，これをメンバとするクラスを作成して対処した．

```
typedef   struct ampData_tag1
{
  int num;  // 測定したデータ個数（周波数の個数）
  CString   comment;    // コメント
  float     freq[NO_OF_DATA];   // 周波数
  float     dB[NO_OF_DATA];     // 利得の測定値
  float     ph[NO_OF_DATA];     // 位相の測定値
}Amp_data;
```

NO_OF_DATA は周波数の最大個数である．今回は処理を簡略化するために固定値 42 に設定した．

② クラス ampData の作成

以下のクラスを作成し使用した．

```
class   ampData
{
  Amp_data   a;    // Amp_data 型の変数   a   を定義.
  ampData*  next;    // Amp_data 型のポインタを定義
public:
  ampData( )  // デフォルトコンストラクタ. これがないと関数 add( )  内で   new 演算子   使用時にエラーとなる
{
  a.num= 0;
  a.comment="";
  for(int i=0;i<NO_OF_DATA;i++)
  {         a.freq[i]=0.0;  a.dB[i]=0.0;   a.ph[i]=0.0;          }
            next                    =NULL;
  }
ampData(Amp_data a1, ampData* q)
{         a=a1;       next=q;   }

ampData*   add(ampData g);
void       disp(int* num, CString* comment, float *out_data);
int        save(ampData a);
void       Get_data(void);
int        Get_num(void);
CString    Get_comment(void);
float*     Get_freq(void);
float*     Get_dB(void);
float*     Get_ph(void);
ampData*   Get_next(void);
};
```

実装部分はスペースの関係で一部のみ示す．

```
ampData* ampData::add(ampData g)
{
          int i;
          ampData*   p=new ampData;
          p->a.num                  = g.a.num;
          p->a.comment              = g.a.comment;
          for(i=0;i<NO_OF_DATA;i++)
          p->a.freq[i]  = g.a.freq[i];
          for(i=0;i<NO_OF_DATA;i++)
          p->a.dB[i]                = g.a.dB[i];
          for(i=0;i<NO_OF_DATA;i++)
          p->a.ph[i]                = g.a.ph[i];
          p->next = this;
          return p;
}
・・・・・・
ampData*   ampData::Get_next(void)
{
          ampData* p=this;
          ampData* p1;
          if( p != NULL )           p1 = p->next;
          else                      p1 = NULL;
          return p1;
}
```

③ リスト構造の実装

文献［2］に倣って以下のように実装した．まず ampData 型の先頭を示すポインタ head をグローバル宣言し、測定したデータを周波数毎に Amp_data 型の構造体変数 data に保存する．個数およびコメントは測定周波数には関係ないので，for ループの外で値を代入している．測定個数が count_freq でないのは，for ループの最後に測定を止めるか否かを問い合わせており，通常最後（42 番目の周波数）まで測定することはなく途中で for ループを抜けるためである．

```
ampData*   head=NULL;  //   測定データをリスト構造で保存
void CMeasureView::OnButton1()     // 測定ボタンのハンドラー
{
          Amp_data   data; // 測定データを保存する構造体
          for(count_freq=0; ～)     // 測定を実行する for ループ
          {
                    ～
                data.freq[count_freq]= 周波数の値 ;
                data.dB[count_freq]= 利得の測定値 ;
                data.ph[count_freq]= 位相の測定値 ;
          }
          data.num = count_freq+1;                  // 測定個数
          data.comment = comment+pDoc->student_name;
          // コメント
          ampData a(data, NULL);          // リスト構造に保存
          head = head->add(a);
}
```

3.4.5 リスト構造のファイルへの保存，読み出しおよび表示

① 構造体データのファイルへの保存

関数 save_amp_pahase_data( )において行っている．構造体 data に加えて，ファイルから入力したデータを一覧するときに必要となる回路種類（番号），利得が 40dB か，20dB かあるいは開ループ利得かを表す番号をファイルに書き込んでいる．保存ホルダーは測定データと同じ experiment_LOG であり，ファイル名は listdata+年月日.csv である．

② 構造体データの読み出し

以下の手順で行っている．

・メニュー／測定器設定／データの読み出しをクリック (図3.9) すると関数 OndatafromFILE()が動作する．

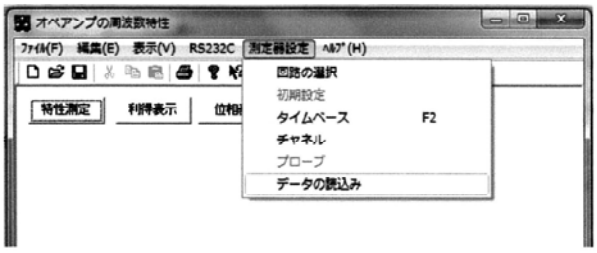


図 3.9 リスト構造のファイルからの読み出し

・関数 OndatafromFILE()においてまず Amp_data 型の構造体 data を宣言する．

・ファイルダイアログを起動し，選択されたファイルから EOF が来るまで（終端まで）データを読み出す．

・測定回路（番号）と増幅器の利得（番号）を読み出す．デリミタは ; である． ，ではうまくいかなかった．

・for ループにおいて NO_OF_DATA 個だけデータを読み出す．

・data.num および data.comment は同一の値が NO_OF_DATA 個だけ書き込まれているので，最初のデータを取り出す．なお，data.comment をグローバル配列に保存し，extern 宣言する．これにより図 3.11 のリストボックスから参照できるようになる．

・リスト構造に保存してから，再度ファイルの読み出しに行き，ファイルの終端に来るまで繰り返す．

なお，pDoc->kairo_tmp_flag=1; とし，利得の表示あるいは位相の表示ハンドラー関数において読み出した特性が，何の測定回路であり，しかも増幅器の利得が何 dB であるかが番号を参照することにより分かるようにした．

③ リスト構造データのファイル構造

測定回路番号;0	増幅器の利得（番号）;1			
21	非反転増幅器 20dB 梶川	1.50E+00	20.4	-2
21	非反転増幅器 20dB 梶川	2.00E+00	20.6	-1
21	非反転増幅器 20dB 梶川	3.00E+00	20.7	0
21	非反転増幅器 20dB 梶川	5.00E+00	20.3	-1
21	非反転増幅器 20dB 梶川	7.00E+00	20.5	-1

図 3.10 リスト構造をファイルにセーブした例（一部）

測定毎に保存したリスト構造のファイルを図 3.10 に示す．最上部は番号により測定回路，利得を示している．1 行目以降は測定周波数の個数，コメント，測定周波数(Hz)，利得(dB)および位相（°）のデータである．コメントは「測定回路+利得+測定者の姓」になっている．

3.5 利得表示ボタンおよび位相表示ボタンのハンドラー関数

両者は基本的に同一のプログラム構造になっているので利得の場合についてのみ示す．

① リスト構造の実装

以下のフローに従っている．

```
void CMeasureView::OnButton2()                    // 利得表示ボタンのハンドラー関数
{
        ampData*  p;    // リスト構造へのポインタを宣言
        p＝head;        // ポインタの初期値を設定
          ～            //  利得周波数特性の表示
        p＝p->Get_next(); // ポインタを更新
}
```

使用しているリスト構造は以下の通りである．

・p->Get_freq()・・・・周波数データを保存している先頭番地

・p->Get_dB() ・・・・利得データを保存している先頭番地

・p->Get_ph() ・・・・位相データを保存している先頭番地

・p->Get_num() ・・・・周波数データの個数

・p->Get_next()・・・・ポインタの更新

② 表示する利得特性の選択

①の～部分の冒頭において図3.11に示すダイアログが起動される．チェックボックスにチェックを入れたデータのみがプロットされる．また pDoc->kairo_tmp_flag=1; の時にも，3.4.5②でファイルから読み出したデータが何の測定回路であり，しかも増幅器の利得が何 dB であるかを正しく表示できるようにした．

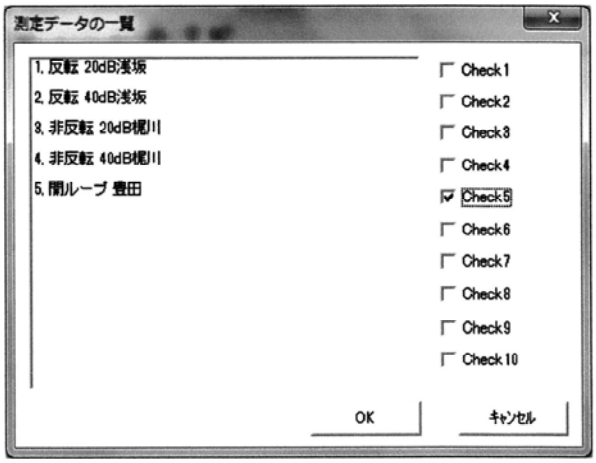


図 3.11 プロットするデータの選択ダイアログ

③ グラフ作成用のクラス CFrame2

関数 OnButton2( )の中でグラフ作成用のクラスのオブジェク成トt1 を宣言し，そのメンバ関数を使用して片対数グラフを作成した．

t1.mySetViewport(pDC, r, 1.0, 1.0, 0.5, 0.5); //ビューポートを管面の矩形領域 r 全体に設定する．pDC はデバイスコンテキストクラスへのポインタである．r は矩形領域を表す変数である．

t1.setFrameData('B', 0.001,-20.0, 20000, 120.0); //周波数レンジが 0.001kHz～20MHz, 利得が-20dB～120dB であることを示す．

t1.setKetaData(0); //y 軸目盛における小数点以下の桁数

t1.getKindofCircuit(0); //0 のとき利得，1 のとき位相を示す

t1.setLabel("周波数", "利得(dB)", "オペアンプの利得周波数特性"); // グラフに印字するラベル

t1.semiLogFrame(pDC,r); //片対数グラフを書く

4．プログラム使用上の注意事項

① Debug ホルダーあるいは Release ホルダー中にある measure.exe をクリックするとプログラムが動作する．RS232C.INI ファイルおよびRS232C_gene.INI ファイルを同じホルダーに作成しておく．

またデータを保存するためのホルダーexperiment_LOG を作成しておく．

② 測定器が全く反応しない場合があった．この場合は一旦発振器およびオシロスコープの電源を切りリセットをかける必要がある．

また USB/RS232C 変換器も PC に再接続する必要がある．

③ 測定可能最大周波数は 20MHz としているが，測定結果からは 2～4MHz が測定限界周波数と思える．測定周波数が 1MHz 程度になると波形が次第に汚くなり，また正弦波とかけ離れてくる場合もある．無限ループに入ってしまう場合もある．これを回避するために 1MHz における測定が終了すると「測定を終了しますか」のメッセージを出し，Yes, NO で選択できるようにしている．適宜波形を見て終了を判断する．

なお，利得が負になった場合にもメッセージを発出する．

④ 処理が無限ループに入ってしまったら cont+alt+delete でプロセスをキルしてプログラムを止める．このときに備えて一つの周波数に対する測定が終了する度にそれまでのデータを上書き保存している．ファイル名に er～と冠している．

⑤ 測定中は図 3.7 のダイアログが管面に表示されているが，このダイアログをマウスでクリックすると「応答なし」となり測定結果が表示されなくなってしまうので要注意である．但し，測定は続行しており，またデータをファイルにセーブしているので，データを消失することはない．最終的にこのファイルを見て測定値を得る．

⑥ 希ではあるが，オシロスコープの波形がクリップしたまま感度の増大がうまく動作しない状況に陥ることがある．恐らくオシロスコープ内にあるリレーの動作不良ではないかと思われる．この場合には，オシロスコープには Local Lockout コマンドは送っていないので，手動にてレンジの増大を行えば良い．

4．学生実験の結果

H26 年度の情報・通信実験は 5 グループ編成で行った．1 グループの人数は 2～4 人であった．最初の頃に実験したグループの測定結果は，まだプログラムに幾分のバグがあったこともあり必ずしも十分に納得のいく結果ではなかった．ここでは 4 回目に実験を行った 3 班の測定結果を示す．

4.1 利得周波数特性の測定結果

図 4.1 はオペアンプ NJM4580D を用いた反転増幅器の利得および開ループ利得の周波数特性である．開ループ利得 A_0 は 10Hz において約 100dB の値が得られた．40dB の減衰器の挿入により大きな利得の測定が可能になった．しかしログから見た入力電圧は低周波数領域では 10mVpp と非常に小さい．また実際に画面を見ても非常に雑音が多かった．このため低周波数領域では入力電圧を大きめに測定している可能性があり，開ループ利得が低めになっているものと思われる．

開ループ利得における 3dB 低下周波数 f_0 は 100Hz であった．またロールオフ特性は周波数 100Hz 以上において 20dB／dec であり，予想通りの結果が得られた．開ループ利得=0dB となる周波数 f_T を直接測定することはできなかったが，外挿により求めた値は 4MHz であった．f_T の計算値は f_0 A_0 より 10MHz となった．いずれも規格値の 15MHz よりはかなり低い値になっていた．

低周波数領域における反転増幅器の利得は設定利得 20dB 時にはほぼ計算値通りの値が得られた．設定利得 40dB 時には計算値よりもやや低めの値になっていた．帰還量は周波数 10Hz において 80dB あるいは 60dB であった．

図 4.2 は非反転増幅器の利得および開ループ利得の周波数特性である．反転増幅器の場合とほぼ同様な特性が得られている．

4.2 位相周波数特性の測定結果

図 4.4 は非反転増幅器の位相および開ループ利得測定回路の位相周波数特性である．開ループ利得測定回路の位相遅れは約 90°に漸近しており，これは前述したように 20dB／dec の振幅におけるロールオフ特性の結果と符合している．非反転増幅器の位相は 90°よりも大きく回転していた．

図 4.3 は反転増幅器の位相および開ループ測定回路の位相周波数特性である．非反転増幅器の場合とほぼ同様な特性が得られた．

5．むすび

本資料では，オペアンプの周波数特性を自動測定するためのプログラムについて，プログラムの考え方および使用方法について述べた．手持ちのオシロスコープや発振器を流用しており，オペアンプを用いた反転増幅器，非反転増幅器および開ループ利得測定回路の周波数特性をプログラム制御により自動的に測

定し，その特性をファイルにセーブし，表示することができる．

かった不具合を修正し，最終的に完成させるに至った．

従来周波数特性の測定には長い時間がかかり，そのため学生達はひたすら測定に終始し，測定した特性を十分に吟味するだけの時間をなかなか得にくかった．このプログラムの開発を始めたのは，少しでもこれを解消したかったことが発端である．今回，このプログラムを使用して周波数特性の測定を行わせた結果，手動で測定するよりも時間の短縮が図れたので当初の目都合5グループの学生実験において使用したが，その都度見つ的を達成できたものと考えている．

[参考文献]

[1] James M. Bryant, “オペアンプ特性を簡単に測定する方法”, Alalog Dialouge 45-04, April(2011)

[2] 加西朝雄, “標準 Visual C++プログラミングブック”, 技術評論社 (2000)

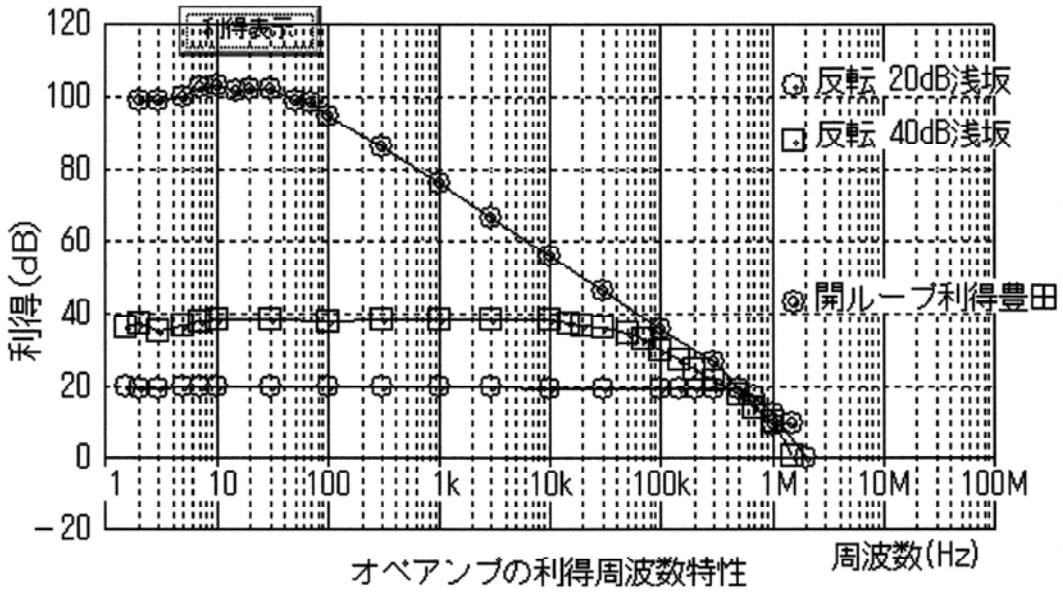
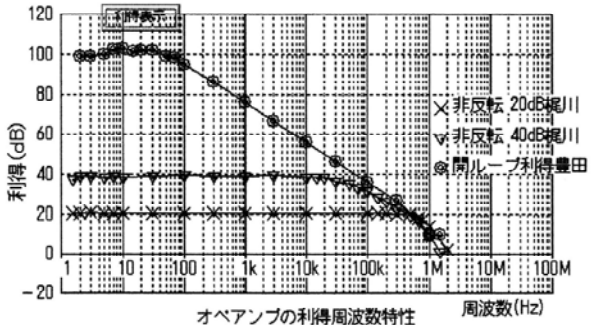
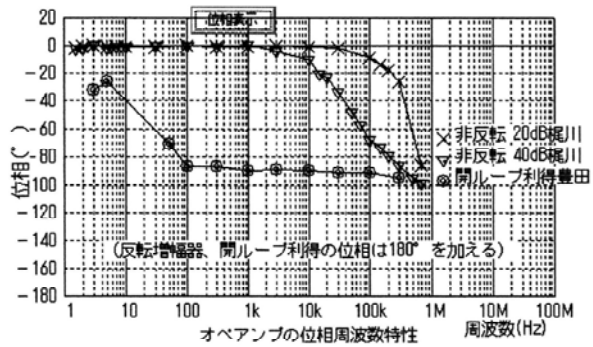
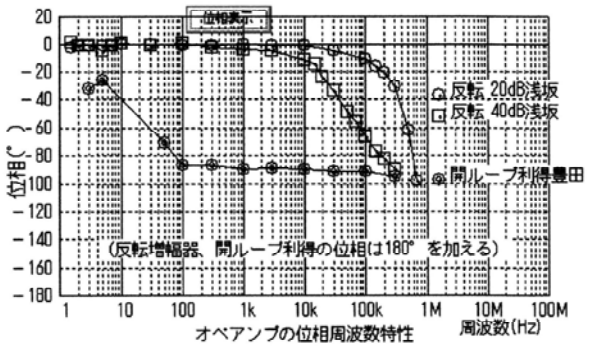


図 4.1 反転増幅器および開ループ利得の測定例







(左上) 図 4.2 非反転増幅器および開ループ利得の測定例

(左下) 図 4.3 非反転増幅器および開ループ利得測定回路の位相の測定例

(右上) 図 4.4 反転増幅器および開ループ利得測定回路の位相の測定例