Octave, myDAQ ガイド

2020 年 06 月

目次

第 1 章

はじめに

1.1 このテキストについて

このテキストは電気電子回路演習で使用する,

- Octave (Matlab 互換の数式処理システム)
- NI myDAQ (National Instruments 社製信号発生, 計測装置)
- Analog Discovery 2 (Digilent 社製信号発生, 計測装置)

について, 演習を進める為に必要な事項をまとめている.

各回の実施内容及び課題提出の詳細については, 講義時に配布される資料および PandA の課題ツールを参照すること.

第 2 章

NI-myDAQ のセットアップ, ファンクションジェネレータとオシロスコープ

2.1 NI-myDAQ のセットアップ

2.1.1 内容確認

myDAQ を開封し, 内容物を確認する

- myDAQ 本体
- クイックスタートガイド
- インストール DVD
- テスターケーブル 2 本 (赤・黒)
- オーディオ接続ケーブル
- ターミナルコネクタ (端子台)
- 端子台用ドライバ
- PC 接続用ケーブル

注釈: あらかじめターミナルコネクタが myDAQ 本体にセットされている場合もある.

2.1.2 PC との接続, ソフトウェアのインストール

以下の手順に従い, PC にソフトウェアをインストールする.

> **警告:** ソフトウェアのインストールは, myDAQ を **接続する前** に実施することが勧められる.

> **警告:** インストール時間は 20 分-30 分程度必要. 電源を確保したうえで実施した方がよい.

National Instruments 社の Web サイト "NI myDAQ をセットアップする" (http://www.ni.com/tutorial/11431/ja/) より，[NI ELVISmx ソフトウェア 4.2.2 - NI myDAQ ハードウェアドライバ（1126 MB）] をダウンロードする.

ダウンロードした [ni-elvismx_19.0_online_repack.exe] を実行する. 途中

- ソフトウェアインストール先の選択 (通常は変更しなくてもよい)
- NI からの重要な更新通知の受け取り (チェックを外してもよい)
- Windows Fast Startup の無効化 (チェックを外してもよい)

が尋ねられる. この後, 必要な事項の記入, ライセンス合意事項の項目にチェックを入れ, インストールを行う.

インストール終了後, 再起動が求められる.

2.2 myDAQ の接続, NI ELVISmx Instrument Launcher の起動

再起動後, myDAQ を接続すると, NI ELVISmx Instrument Launcher が起動する. 起動しない場合は, スタートメニューより, *[National Instruments]* → *[NI ELVISmx Instrument Launcher]* を実行する. これは各種の測定を myDAQ を用いて実施するためのアプリケーションランチャーである.

注釈: 以下に説明するファンクションジェネレータ, オシロスコープ以外にも様々な機能が利用できる. 余力があればいろいろ試してみよう.

2.3 ファンクションジェネレータ

最初にファンクションジェネレータ (Function Generator, FG, 信号発生器) を使ってみよう.

ランチャーから [Function Generator] のアイコンを選択する.

注釈: ランチャーを使わず, スタートメニューから, *[National Instruments]* → *[NI ELVISmx Function Generator]* を選択してもよい.

起動時に, ウインドウ下の [Device] の項目が [Dev1 (NI myDAQ)] と表示され, ソフトウェアにより, myDAQ が認識されていることを確認する. そうでない場合, myDAQ を接続, ソフトウェアの起動をやり直す.

2.3.1 端子台の固定, ジャンプワイヤによる接続

myDAQ 側面に端子台を取り付ける. 続いて実験キットにあるミノムシクリップ付きジャンプワイヤ 2 本を

- [AO 0]
- [AO AGND]

の myDQA 端子に, ミノムシクリップ側を実験キットのスピーカに接続する (極性はない).

この時, 以下の点に注意すること.

- myDAQ と端子台との間に隙間ができないようしっかり押し込むこと. ケースには端子台を取り付けたまま収納できるので, 頻繁に取り外す必要はない.
- 端子台のねじを上から見て時計回りに回すと, 接続部にある金属接点が上に移動する. この金属接点と, 接続口の上端でジャンプワイヤを挟み, 固定すること.

図 2.1　myDAQ とスピーカとの接続

2.3.2 信号の出力

よく知られている信号音として, 時報で利用される 440Hz 正弦波がある. これを出力する.

FG ソフトウェアの画面で以下の 2 点を設定, 確認する

- [Waveform Setting] 枠内の [Frequency] を, 440 Hz に設定する
- [Instrumental Control] 枠内の [Signal Route] を, [AO0] に設定する

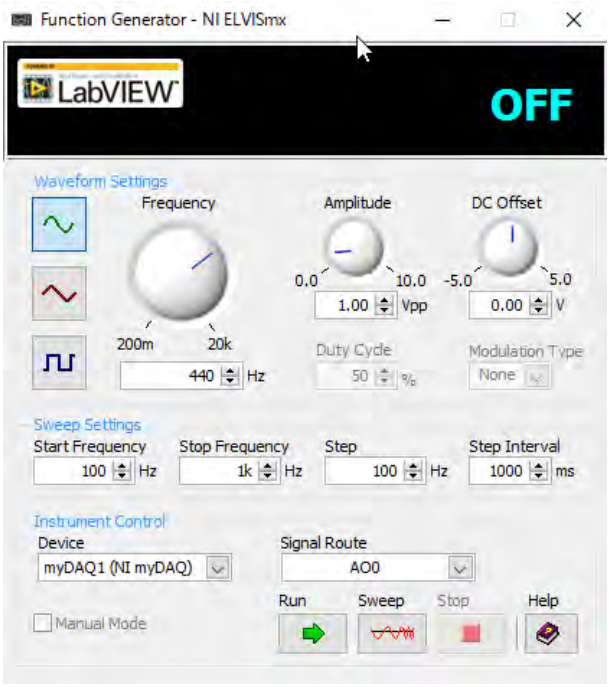


図 2.2　Function Generator ソフトウェアの画面

この状態で画面下部の [Run] ボタンを押すと信号が出力, スピーカから音が出る. 停止するには [Stop] ボタンを押す.

2.3.3 いろいろやってみよう

- 周波数を変えてみる
- 出力波形を三角波, 矩形波に変えてみる
- 出力電圧を変えてみる
  - [Amplitude] は標準で 1.0 Vpp となっている. これは信号の最大値-最小値の幅 (peak-to-peak) が 1V であるという意味.
  - [DC offset] は信号波形の平均値, 標準では 0V である.
- スピーカを実験キットのミニジャックに代え, 自分のイヤホンで聞いてみる. (ブレッドボードを使ってみてもよい).

2.4 オシロスコープ

続いてオシロスコープ (Oscilloscope, OSC) により, FG の出力波形を測定する.

2.4.1 端子台の固定, ジャンプワイヤによる接続

ジャンプワイヤを用い, 以下のように結線する.

- [AO 0] -- [AI 0+]
- [AO AGND] -- [AI 0-]

図 2.3　myDAQ と端子台, ジャンプワイヤの接続

2.4.2 ソフトウェアの起動

ランチャーから [Function Generator] 及び [Oscilloscope] を起動する. いずれのアプリケーションにおいても, 入出力が適切なデバイス, 端子に割り当てられていることを確認する.

- [Function Generator]
  - [Instrumental Control] 枠内の [Device] が [myDAQ1 (NI myDAQ)]
  - [Instrumental Control] 枠内の [Signal Route] が [AO0]
- [Oscilloscope]
  - [Instrumental Control] 枠内の [Device] が [myDAQ1 (NI myDAQ)]
  - [Instrumental Control] 枠内の [Acuqisition Mode] が [Run Continuously]
  - [Channel 0 Settings] 枠内の [Source] が [AI 0]
  - [Channel 0 Settings] 枠内の [Enabled] にチェック

Function Genrator, Oscilloscope アプリケーションの下部にある [Run] ボタンを押すと, 発振, 測定が有効になる.

この状態では, オシロスコープの波形が流れて表示される.

- [Trigger] 枠内の [Type] を [Edge]
- [Trigger] 枠内の [Source] を [Chan 0 Source]

とし, 適切なトリガを設定すると, 波形が固定される.

OSC 画面内の縦軸, 横軸の目盛りはそれぞれ

- [Channel 0 Settings] 枠内の [Scale Volts/Div]
- [Timebase] 枠内の [Time/Div]

により変更できる. 適切に設定し, FG の出力が正しく OSC に反映されていることを確認せよ. なお, OSC 波形画面の下部には測定された信号の

- 2 乗平均値 (Root Mean Square, RMS)
- 周波数
- Vp-p (peak-to-peak, 波形の最大-最小電圧の差)

が表示されるので, これも参考にするとよい.

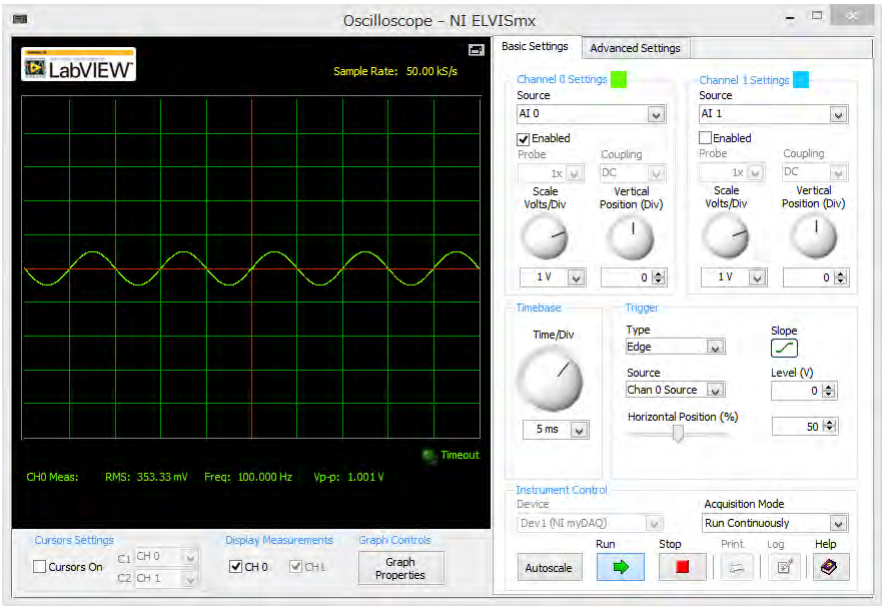


図 2.4　Oscilloscope ソフトウェアの画面 (100Hz, Vpp=1V, 正弦波の測定例)

2.5 myDAQ の電流出力制限, スピーカの定格について

FG の出力によりスピーカを鳴らしているとき, 実際にどのような電圧が加わっているか調べてみよう.

ブレッドボードとジャンプワイヤを用い, FG の出力をスピーカと OSC に同時に入力する回路を構成する.

この状態で, Vpp=1V, 440 Hz の正弦波を FG より出力する. この時, OSC に表示される波形は, スピーカの接続の有無により大きく異なる. スピーカが接続されていない場合, FG で設定した正弦波が観察されるが, スピー

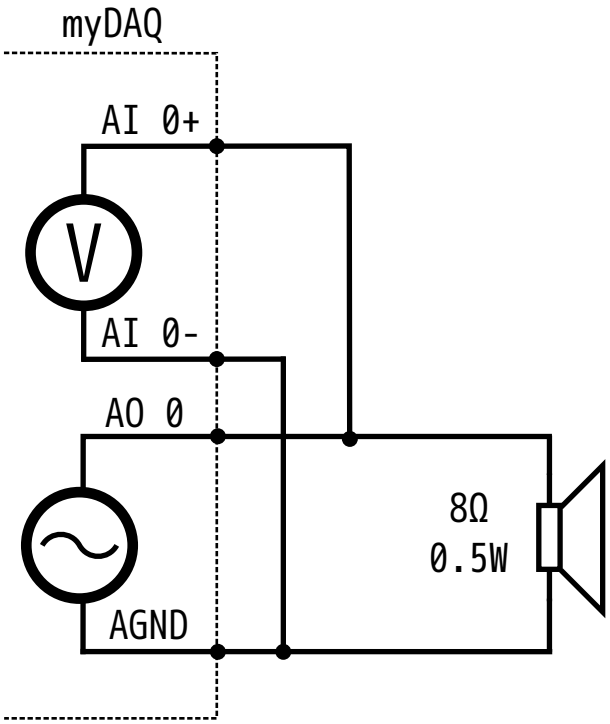


図 2.5　測定の回路図

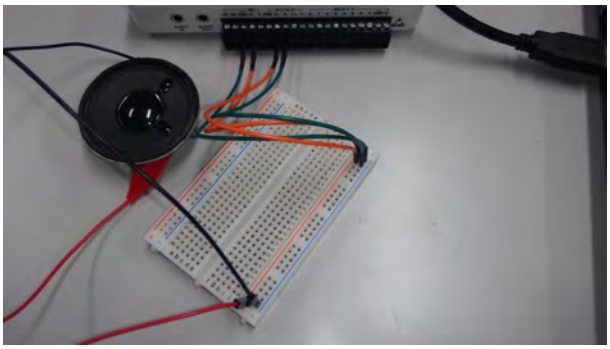


図 2.6　回路の構成例

カを接続することで, 波形はひずみ, また振幅も数十 mV と極めて小さくなる (つまり最初にスピーカから聞こえた音は正弦波ではない!?).

これは, myDAQ のアナログ出力電流の最大値が最大 2mA と極めて小さな値であることによる (myDAQ のスペックシート http://www.ni.com/pdf/manuals/373061f.pdf). 非常に荒っぽい概算をすると, スピーカの内部抵抗は 8 Ω であるため, 理想的な電源だと FG 出力電圧の最大値 0.5V において, 62.5 mA の電流が流れるはずである. しかし, myDAQ の場合それだけの電流を供給することができず, 逆に最大の電流が出せる程度にまで出力電圧が下がってしまう.

FG から出力する信号の振幅を小さくすると, OSC で測定される波形のひずみが減少し, FG が想定している出力に近い波形となる.

このように, 抵抗 (インピーダンス) が極端に低い負荷に対する測定では, 入力電圧の大きさや, (ここでは詳しく触れていないが) 入力回路の内部抵抗の影響を十分考慮する必要がある.

図 2.7　FG より Vpp = 1V, 440 Hz の正弦波を出力した場合の, スピーカにかかる電圧

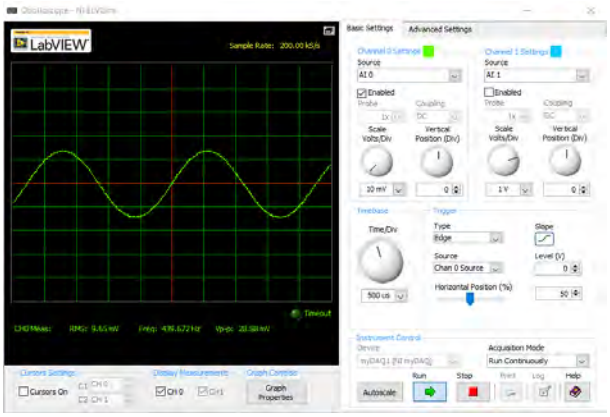


図 2.8　FG より Vpp = 0.03V, 440 Hz の正弦波を出力した場合のスピーカにかかる電圧

2.6 オシロスコープデータの保存, 後処理と表示

オシロスコープに表示された波形は, ファイルに保存することができる.

1. Oscilloscope アプリケーション下部にある [Stop] を押して, 測定を停止する
2. [Log] ボタンを押し, 保存するフォルダ, 並びにファイル名を指定する. 拡張子は .txt が推奨される (ここでは sinecurve.txt とする)

保存したファイルの内容は以下のようになる.

```
waveform      [0]
t0    2015/11/04   18:04:15.527346
delta t       2.000000E-5

time[0]       Y[0]
2015/11/04   18:04:15.527346   -5.577443E-4
2015/11/04   18:04:15.527366   -6.739844E-3
2015/11/04   18:04:15.527386   -1.292194E-2
(中略)
2015/11/04   18:04:15.577326   6.600477E-3
2015/11/04   18:04:15.577346   -2.323706E-4
```

各項目の区切りはタブ記号である. 最初 5 行はヘッダファイルである.

```
1 行目： 保存した波形の数． この場合 [0] のみ
2 行目： chan0 の測定開始時刻． [t0, "chan0 年/月/日 時:分:秒"]
3 行目： データの間隔 (x 軸)． ["delta t", "chan0 dt"]
4 行目： 空白
5 行目： 波形データのヘッダ． この場合 ["time[0]", "Y[0]"] の 1 組
6 行以降： 波形データ ["chan0 年/月/日 時:分:秒, "chan0 Y 値 (V)"]
```

2.6.1 Excel での描画

オシロスコープデータは, タブ区切りデータである． excel2013 や 2016 の場合, まず excel を起動し, sinecurve.txt をドラッグ&ドロップする. これで, 自動的にタブ記号を区切りとしてレイアウトされたデータが読み込まれる.

しかし, 時刻 (time[0]) の精度が悪いため, グラフ作成時には時刻を再計算した方がよい. time[0] の値を上書きしてよいならば, 以下の処理を行う. ヘッダ部に「delta t 2.0000e-5」の記述があることから, このデータの各行は $2.0 \times 10^{-5} = 20\mu$sec の時間間隔での結果である.

1. A6 のセル (波形データの先頭時刻) に **=(ROW()-6)*2e-5** と数式を入力

   (**ROW()** 関数は現在の行数を取得する excel の関数)

2. 以下 A7, A8, のセルにも同様の式を適用する. つまり, 1×10^{-5}, 2×10^{-5} といった計算結果がセルに記入される.

3. この時点で A 列は時刻を表すように設定されているが, これを小数を表現できるように書式設定を変更する.

   a. A 列全体を選択

   b. 右クリックのポップアップメニューより,「セルの書式設定」を選択

   c.「表示形式」タブにおいて「分類」を「指数」,「小数点以下の桁数」を「2」かそれより大きい値とする

4. 必要であれば ch1 の時刻 (time[1]) についても同様の処理を行う.

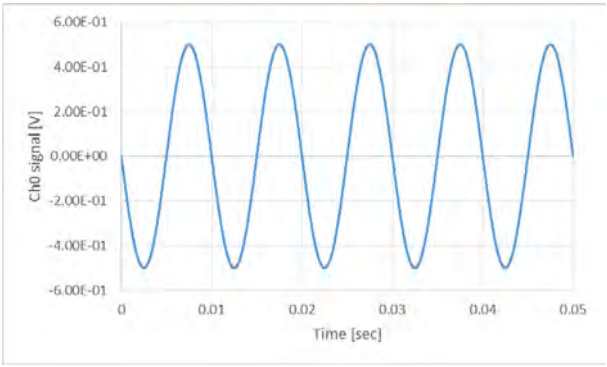


図 2.9　オシロスコープデータを excel によりプロットした図

2.6.2 Octave での描画

Octave でプロットする場合, **dlmread** 関数を用いるとよい.

リスト 2.1　オシロスコープデータの読み込み, プロットを行う Octave スクリプト (Octave におけるスクリプトの使い方は *Octave のセットアップと基本操作* 5.3.6 節を参照)

```
% plot_sinecurve.m Oscilloscope データをプロットする

% Octave ではファイルが function で始まる場合,
% スクリプトではなく関数ファイルとして扱われる.
% そのことを避けるため慣習的に 1; を冒頭に記載する
1;

% 同じフォルダにある sinecurve.txt を読み込む
% 区切り記号はタブ \t
% (0, 0) 始まりで 5 行目, 0 列目から読み込む
% 言い換えれば, 先頭の 5 行, 左から 0 列を読み飛ばす)
%
% これにより
% tbl(:, 1) --- ch0 時刻
% tbl(:, 2) --- ch0 電圧
% tbl(:, 3) --- ch1 時刻 (ch1 が有効な場合)
% tbl(:, 4) --- ch1 電圧 (ch1 が有効な場合)
% が入力される, (ただし時刻のデータは正しく変換されない)
tbl = dlmread('sinecurve.txt', '\t', 5, 0);

% 時間刻みを明示的に与える
dt0 = 2e-5;
% dt1 = 2e-5; % ch1 が有効な場合

% 行数を求める
r = rows(tbl);

% 各行に対する時刻を計算する, tbl(:, 1), tbl(:,3) を上書き
tbl(:,1) = [0:(r-1)] * dt0;
% tbl(:,3) = [0:(r-1)] * dt1; % ch1 が有効な場合

% プロットする, 電圧値は (1 始まりの)2 列目, 青色 ('b') の実線とする.
plot(tbl(:, 1), tbl(:, 2), 'b;ch0;');
% ch1 が有効な場合. 赤色 ('r') の実線とする. また同一のグラフに書くため `hold on` を実行する
% hold on;
% plot(tbl(:, 3), tbl(:, 4), 'r;ch1;');
xlim([0, 0.05]);
ylim([-0.6, 0.6]);
xlabel('Time [sec]');
ylabel('Signal [V]');
legend('location', 'northwest'); % 凡例の表示
grid(); % グリッドの表示
```

注釈: Ch0 と Ch1 の 2 チャンネルを測定した場合については *ch1 が有効の場合* と書かれた部分のコメントを

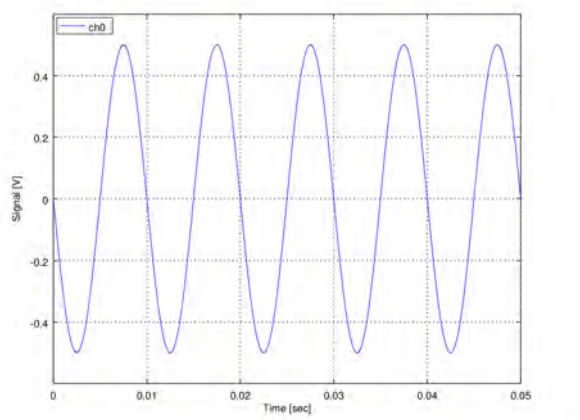


図 2.10　オシロスコープデータを octave によりプロットした図

外す.

第 3 章

Analog Discovery 2 のセットアップ, ファンクションジェネレータとオシロスコープ

Analog Discovery 2 (以下 AD2) は myDAQ と同様の USB 接続可能なオシロスコープ, ファンクションジェネレータである.

3.1 セットアップ

AD2 は, Windows, Mac, Linux で利用可能である. それぞれのセットアップ方法は,

- Windows: https://reference.digilentinc.com/learn/instrumentation/tutorials/analog-discovery-2-getting-started-windows/start
- Mac: https://reference.digilentinc.com/learn/instrumentation/tutorials/digital-discovery-getting-started-mac/start
- Linux: https://reference.digilentinc.com/learn/instrumentation/tutorials/analog-discovery-2-getting-started-linux/start

に記載されている. ここでは Widows, Mac のセットアップについて説明する.

3.1.1 内容物の確認

AD2 の内容物は以下の通りである.

- AD2 本体
- USB ケーブル
- コネクタケーブル
- コンタクトピン (それぞれのピンは 1 本づつバラバラにしてもよい)

3.1.2 ソフトウェアの取得, インストール, 接続確認

WaveForms は, AD2 を PC から操作するためのソフトトウェアである.

WaveForms のダウンロードページ　https://reference.digilentinc.com/reference/software/waveforms/waveforms-3/start

上記 Web ページの右端の "Latest Download" よりそれぞれの OS に対応した WaveForms のインストールプログラムをダウンロード, インストールを行う.

Windows の場合

ダウンロードしたファイル (digilent.waveforms_vxxx.exe) を実行する. ほぼすべての場合で, [Next] ボタンを押してインストールを進めるとよい.

Mac の場合

ダウンロードしたファイル (digilent.waveforms_vxxx.dmg) を開くと, セットアップ画面が表示される.

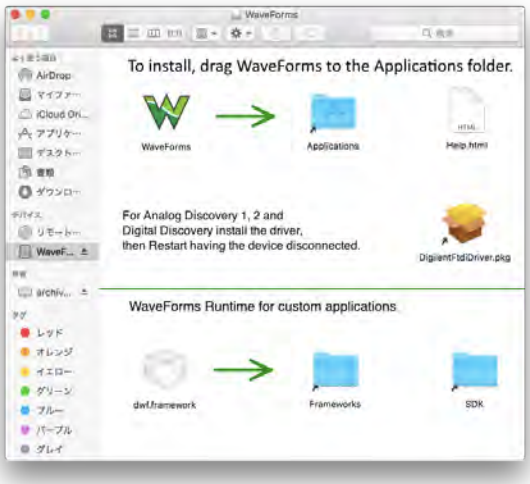


図 3.1　WaveForms インストール画面 (Mac 版)

ここで

1. "WaveForms" のアイコンを "Applications" のアイコンにドラッグ&ドロップする (WaveFormas アプリケーションのコピー).
2. (再び WaveForms セットアップ画面にて) DigilentFtdiDriver.pkg をダブルクリックする. Analog Discovery 2 用ドライバのインストールが開始される.
3. (再び WaveForms セットアップ画面にて) "WaveForms Runtime for custom applications" にある, "dwf.framework" を "Frameworks" のアイコンにドラッグ&ドロップする.
4. 作業終了後, 再起動する.

5. アプリケーションフォルダを開き (Finder の **移動** → **アプリケーション**または, `Shift + Command + A`), "WaverForms" のアイコンを選択, ソフトウェアを起動する.

6. ここで, "WaverForms の開発元を確認できないため, 開けません" のエラーが表示される場合, https://support.apple.com/kb/PH25088?locale=ja_JP に従い, 上記 4 の段階で, "Control キーを押しながらアプリケーションアイコンをクリックし, ショートカットメニューから「開く」を選択"する.

AD2 の接続

AD2 を接続せず, WaveForms を起動した直後は, 有効な (物理) デバイスが存在しない旨のダイアログボックスが表示される. ここで, AD2 を USB ケーブルで接続すると, この機器が認識される. 接続された AD2 機器名 (Discovery2NI) を選択, さらに画面下半分のモードから (Scope と Wavegen が有効な) `1` を選択し *Select* ボタンを押す.

注釈: PC と AD2 との接続に成功すると, AD2 本体中央部の緑色 LED が点滅する.

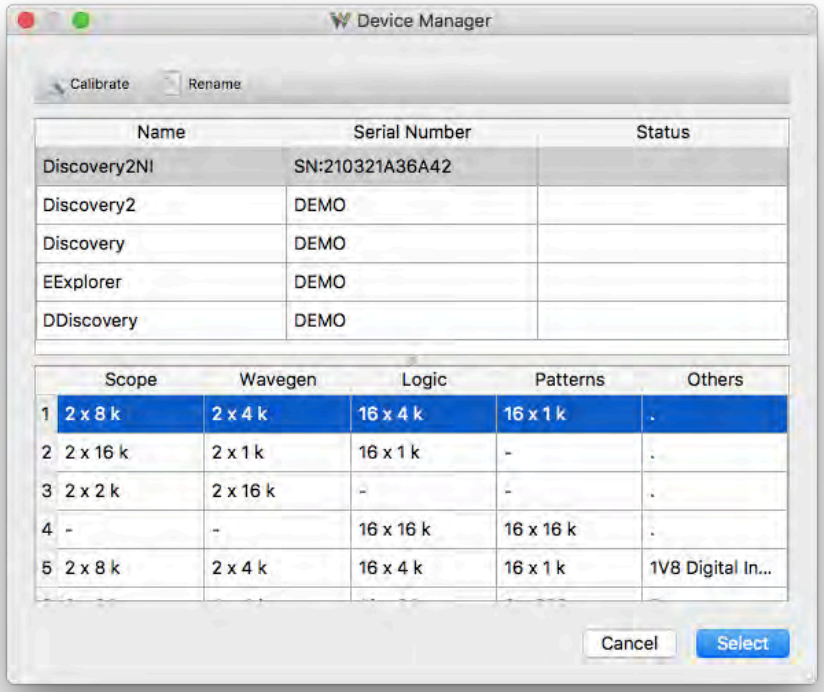


図 3.2　認識された AD2 デバイスおよび, 動作モードの選択

3.2 ファンクションジェネレータ (Wavegen)

最初にファンクションジェネレータ (Function Generator, FG, 信号発生器) を使ってみよう.

3.2.1 スピーカとの接続

AD2 は最大 2 チャンネルのアナログ出力を持ち, それぞれ

- [W1(黄) と [Ground(↓, 黒)]
- [W2(黄白) と [Ground(↓, 黒白)]

より信号が出力される.

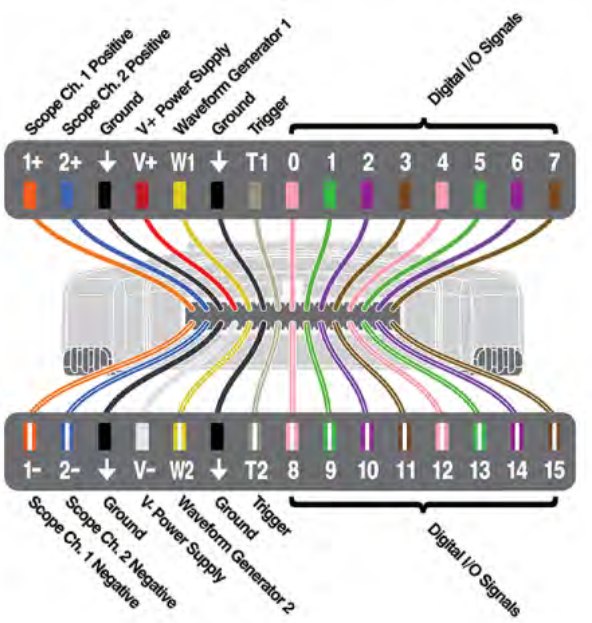


図 3.3　Analog Discovery 2 のピン配置

まず AD2 にコネクタケーブルを接続し, さらに `W1` と `Ground` をそれぞれ実験キットに含まれるスピーカと接続する.

続いて AD2 を PC と接続し, WaveForms を起動する. AD 2 が認識されていれば, 画面下に `Discovery2NI SN:....` の記述が表示される.

ファンクションジェネレータの機能は `Wavegen (Waveform Generator)` により提供される. 画面左の機能一覧から, *Wavegen* をクリックする (またはウィンドウ一番上にあるタブより *Welcome* → *Wavegen* を選ぶ). 標準では `Channle 1` と命名された 1 チャネル分の制御画面が表示される.

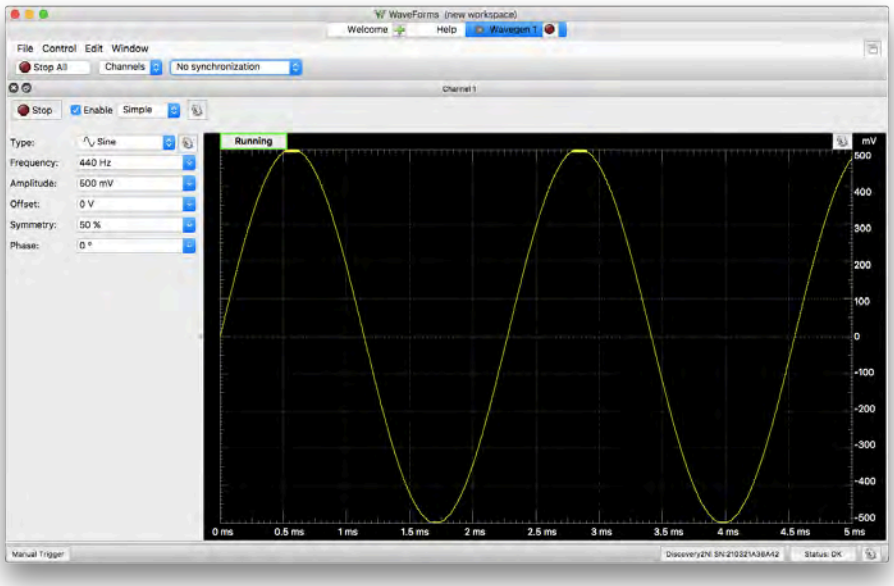


図 3.4　Wavegen の設定画面

3.2.2 信号の出力

信号の出力 と同様に, 440Hz, Vpp=1V の正弦波を出力する.

`Channel 1` のウインドウ内で, 以下の値を設定する.

- *[Frequency]* の値を 440 Hz
- *[Amplitude]* の値を 0.5V (表示は 500mV となる)

`Channel 1` ウインドウ内の *[Run]* をクリックすると信号が出力され (この時 *[Enable]* にチェックが入る), スピーカから音が出る.

注釈: myDAQ の場合と音量を比べてみよう.

3.3 オシロスコープ (Scope)

続いてオシロスコープ (Oscilloscope, OSC) により, FG の出力波形を測定する. AD2 は 2 チャンネルの差動アナログ入力を持つ. すなわち,

- [1+ (橙)] と [1- (橙白)]
- [2+ (青)] と [2- (青白)]

で, それぞれの間の電位差を測定する.

AD2 付属のコンタクトピンを用い,

- [1+(橙)] と [W1 (黄)]
- [1-(橙白)] と [Ground (黒)]

を接続する.

WaveForms において, (`Wavegen 1` ウィンドウが開いた状態のままで), *Welcome* → *Scope* を選択し, `Scope 1` ウィンドウを追加する. これで, 画面上部には `Welcome` , `Help`, `Wevegen 1`, `Scope 1` と 4 つのタブが表示される.

Wevegen 1 画面において, 信号出力のパラメータを確認し, *[Run]* ボタンを押し, 信号を出力する.

続いて, *Scope 1* 画面において, *[Run]* ボタンを押し, 測定を開始する. 標準では, Channel 1(黄色で表示), Channel 2(シアンで表示) ともに波形を表示するので, 今回は *[Channel 2]* のチェックを外す.

オシロスコープ画面の目盛の横軸, 縦軸はそれぞれ, *[Time]* → *[Base]* 及び *[Channel 1]* → *[Range]* で変更できる.

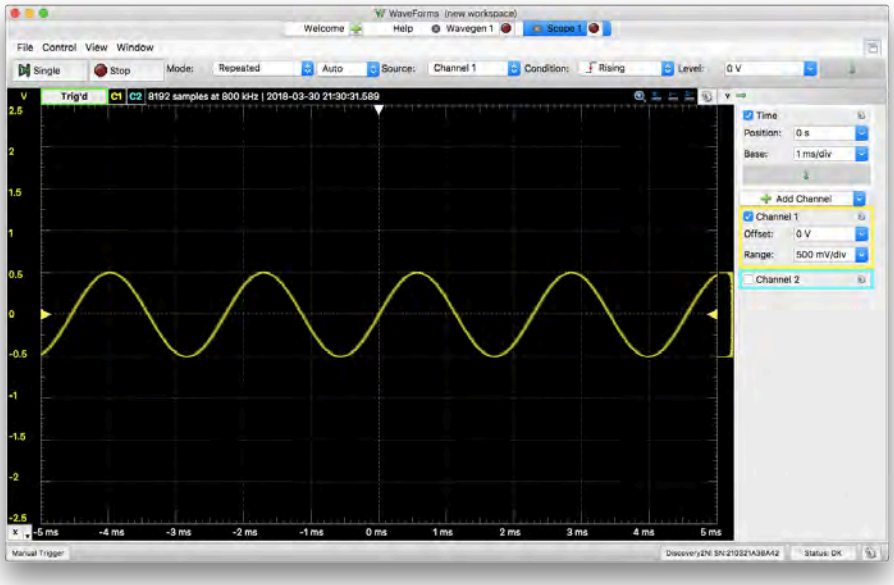


図 3.5　Scope の計測画面

3.3.1 データの保存

Scope で取得したデータをファイルに保存するには, Scope 画面上にて, *File* → *Export* を選択する. Export ダイアログでは, 以下の選択肢がある.

- *CSV* データ *[Data]* または *Scope* 画面のスナップショット *[Image]* が選択できる. ここでは *[Data]* を選択する
- *[Source]* では保存するデータの種類を選択する. 今回は `Oscilloscope` でよい.
- *[Options]* には, データ保存時にコメント (測定条件など), ヘッダ (各カラムの名称), ラベル (時間のカラム) の出力の可否を選択できる.

これらを確認したのち, ダイアログ画面右下の *[Save]* ボタンを押す. ファイル名 (拡張子は .csv が自動的に付加される) 並びに保存場所を指定し, データをファイルに保存する.

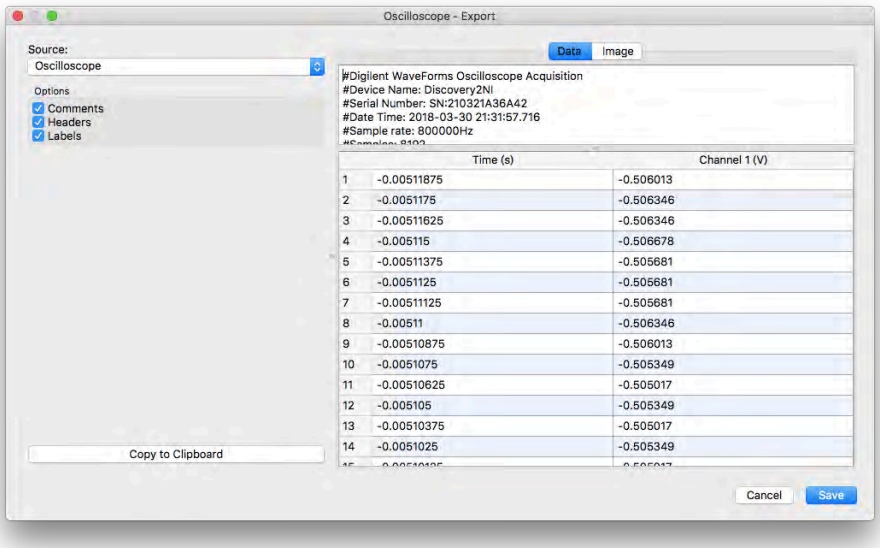


図 3.6　Scope の Export 画面 (CSV 保存)

ファイルの内容は以下の通りである. 今回は Channel 1 だけが有効であったので, Channel 2 に相当するカラムは省略されている.

```
#Digilent WaveForms Oscilloscope Acquisition
#Device Name: Discovery2NI
#Serial Number: SN:210321A36A42
#Date Time: 2018-03-30 21:12:49.885
#Sample rate: 800000Hz
#Samples: 8192
#Trigger: Source: Channel 1 Type: Edge Condition: Rising Level: 0 V Hyst.: Auto␣
↪HoldOff: 0 s
#Channel 1: Range: 500 mV/div Offset: 0 V

Time (s),Channel 1 (V)
-0.00511875,-0.50402030370924233
-0.0051174999999999997,-0.50435248926693266
-0.0051162499999999993,-0.50468467482462298
-0.0051149999999999998,-0.50468467482462298
-0.0051137499999999994,-0.50435248926693266
-0.0051124999999999999,-0.50468467482462298

(後略)
```

excel 等でのデータの処理方法については, *NI-myDAQ のセットアップ, ファンクションジェネレータとオシロスコープ の Excel での描画* 等を参考にするとよい (myDAQ で必要だった時間スケールに関する処理は不要).

第 4 章

オペアンプ回路の作成

ごく簡単なオペアンプ回路を作成し, その動作を確認する.

4.1 用意する物

実験キットの内容 (PandA より 授業資料 → 実験キットの内容) 及び電気電子回路演習パーツキット (授業資料 → 実験キットの内容 → 外観とパーツ名の対応) を参照のこと

ブレッドボード

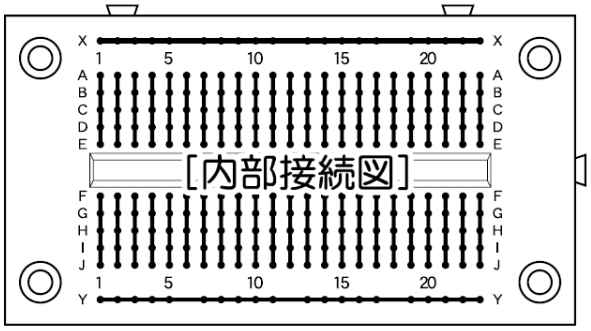


図 4.1　ブレッドボードの結線図

フォトトランジスタ (型番: NJL7502L)・発光ダイオード (LED)

> NJL7502L のデータシートは https://www.njr.co.jp/products/semicon/products/NJL7502L.html
>
> フォトトランジスタ, 発光ダイオード共にリード線の長いほうから短いほうに向かい電流が流れることに注意する.

抵抗器

- 100 k Ω : カラーコードは 茶 黒 黄 (金) である. フォトトランジスタの光応答性を制御する. データシートを参照のこと.

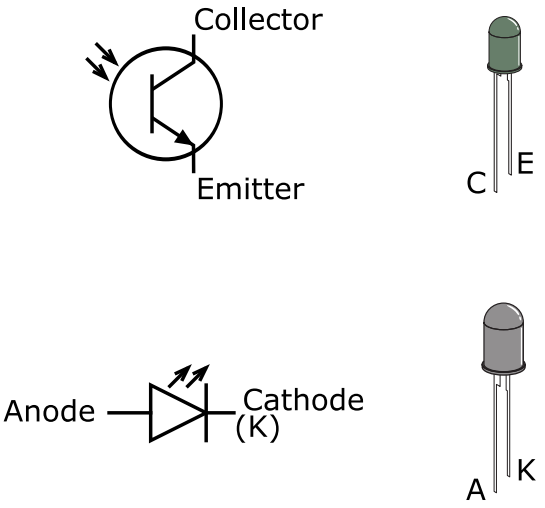


図 4.2　フォトトランジスタと発光ダイオード

- 100 Ω：カラーコードは 茶 黒 茶 (金) である. 発光ダイオードに流れる最大電流を制御する (電流制限抵抗)

オペアンプ (型番: LM324N)

データシートは秋月通商のページ (http://akizukidenshi.com/catalog/g/gI-00060/) を参照のこと. 切り欠きを目印にピンに番号が振られている.

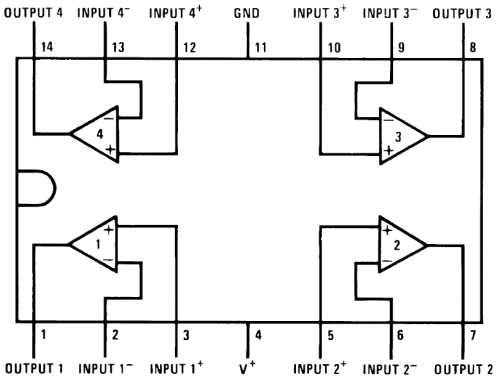


図 4.3　LM324N のピン配置

その他

- 電池ボックス, 電池
- ジャンプワイヤ
- myDAQ または Analog Discovery

4.2 ブレッドボードでの回路作成

テキストの図 1.6(左) に従い, ブレッドボード上に素子を配置する. 以下の点に注意する.

- オペアンプはブレッドボード中央の溝の上に挿す. これにより, 各ピンが縦の列を介して接続できるようになる.
- 最上部, 最下部の列 (図 4.1 の X, Y) は, 通常グランドと, オペアンプやトランジスタを駆動するための電圧源として利用する.
- オペアンプには入出力以外に, 駆動用の電源を接続する必要がある. 記載が省略されることが多いので注意すること.

  LM34N2 の場合は, V+(4 番ピン) に 3V~32V の定電圧を加え (この電圧によりオペアンプの最大出力電圧が決まる), 更に GND(11 番ピン) が 0V の基準となるよう接地する. 従って 1-3 番ピンのオペアンプを利用する場合, 接続が必要なピンは以下の 5 つである.

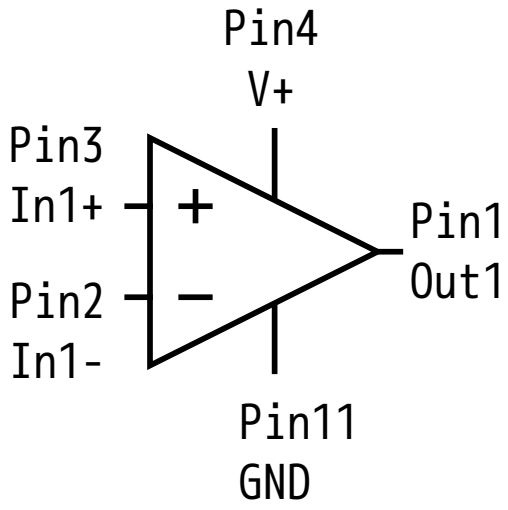


図 4.4　オペアンプ駆動時に接続が必要なピン

作成した回路の一例を示す. これはテキストの図 1.6(左) に相当する. 参考のために, 乾電池から供給される GND(0V) と 3V が加わる部分を, 黒, 赤の点線で記載している.

乾電池を接続すると回路が動作する. フォトトランジスタを指で覆うことで LED がオン・オフとなることが確認できる.

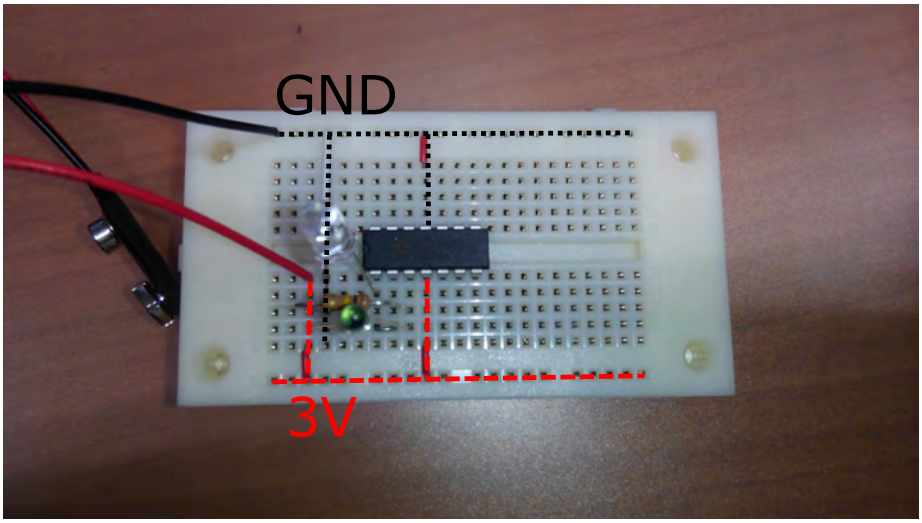


図 4.5　レイアウトの一例 (テキスト図 1.6(左) に相当)

4.3 myDAQ または AD2 からの電源供給

前節では, フォトダイオードとオペアンプ双方を駆動するために 3V の乾電池を用いていた. これを, テキストの図 1.6(右) に示すようにオペアンプへの電源を, myDAQ の 15V, AD2 の 5V に切り替える.

警告: myDAQ, AD2 からオペアンプに電源を共有すると, 乾電池に比べ大きな電力を出力できるようになる. このままでは LED に過度の電流が流れ,

- LED が故障する
- LED が強く発光し, 目を傷める

恐れがある. 必ず電流制限抵抗を LED に直列に挿入すること.

4.3.1 myDAQ の場合

- myDAQ 端子台の [15V] をオペアンプの V+ に接続する. 乾電池の + 出力とは接続しないこと.
- myDAQ 端子台の [AGND] をオペアンプの GND に接続する. これは回路内における共通のグランドである.
- myDAQ を PC に接続すると, 自動的に給電される

4.3.2 AD2 の場合

- AD2 の [V+ (赤)] をオペアンプの V+ に接続する.
- AD2 の [Ground ↓ (黒)] をオペアンプの GND に接続する.
- AD2 の V+ は出力の on/off, 出力電圧 (0-5V) を WaveForms アプリケーションから制御できる. Waveforms アプリケーションを起動する.
- [Supplies] ツールを起動する. 以下の設定により, V+ より 5V の電圧を出力する.
  - [Positive Supply (V+)] の [Voltage] を 5V にセットする
  - [Master Enable] のボタンを ON にする
  - [Positive Supply (V+)] のボタンを ON にする

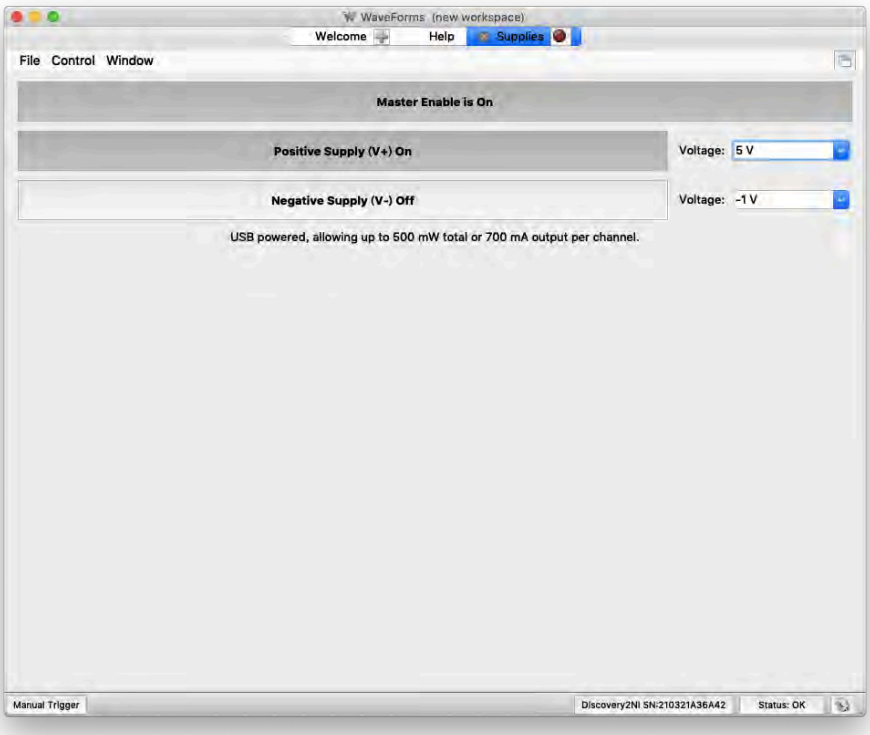


図 4.6　AD2 での定電圧出力設定

4.4 入出力電圧の測定

続いて, myDAQ, AD2 のオシロスコープ機能を用いて, オペアンプの入出力の関係を計測する. 前節の回路に対し, オシロスコープのプローブを追加する.

4.4.1 myDAQ の場合

回路と myDAQ との接続は以下のようになる.

- [+15V] からの給電はそのままにする (一旦接続を解除し, LED が光らないようにしてもよい)
- [AI0+], [AI0-] をそれぞれ, 回路の [オペアンプの Input +], [GND] に接続 (オペアンプの入力電圧の測定用)
- [AI1+], [AI1-] をそれぞれ, [LED のアノード], [LED のカソード (GND に接続)] に接続 (LED に加わる電圧の測定用)

続いて, NI ELVISmx Instrument Launcher より, [Oscilloscope] を起動し, 以下のように測定条件を設定する.

- [Instrumental Control] 枠内
  - [Device] が [myDAQ1 (NI myDAQ)] であることを確認
  - [Acuqisition Mode] が [Run Continuously]
- [Channel 0 Settings] 枠内
  - [Source] が [AI 0]
  - [Enabled] にチェック
  - [Scale] を [1V/div]
- [Channel 1 Settings] 枠内
  - [Source] が [AI 1]
  - [Enabled] にチェック
  - [Scale] を [1V/div]
- [Trigger] 枠内の [Type] が [Immediate]

[Run] ボタンを押すと測定が行われる.

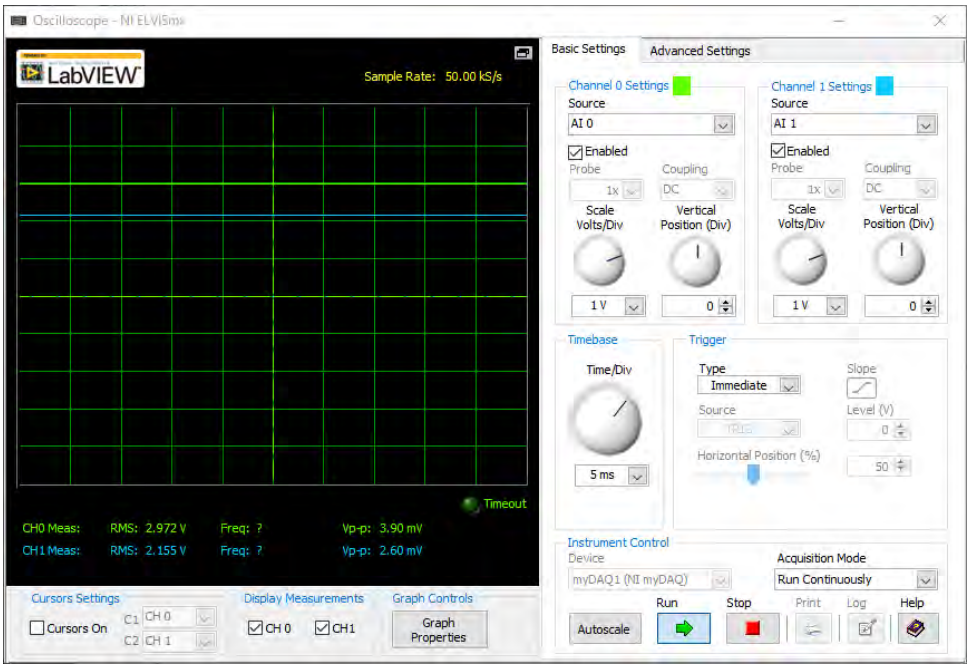


図 4.7　myDAQ Osilloscope での測定

4.4.2 AD2 の場合

回路と AD2 との接続は以下のようになる.

- [V+] からの給電はそのままにする (一旦接続を解除し, LED が光らないようにしてもよい)
- [1+ (橙)], [1- (橙白)] をそれぞれ, 回路の [オペアンプの Input +], [GND] に接続し, オペアンプの入力電圧を測定する.
- [2+ (青)], [2- (青白)] をそれぞれ, [LED のアノード], [LED のカソード (GND に接続)] 接続し, LED に加わる電圧を測定する.

続いて, WaveForms アプリより [Scope] ツールを起動する. すでに [Supplies] ツールが起動しているので, その隣の [Welcome] タブの [+] をクリックし, [Scope] を追加する.

- [Channel 1], [Channel 2] 双方において,
  - チェックボックスにチェックが入っていることを確認
  - [Range] を [1V/div]

[Run] ボタンを押すと測定が行われる.

図 4.8　AD2 Scope ツールでの測定

4.4.3 動作確認

フォトトランジスタに入射する光量の変化にしたがい, オペアンプの入力, LED にかかる電圧が変化することが確認できる. 適当な状態で、入出力電圧のデータを取得し, プロットする.

4.4.4 その他の確認, 考察事項

- オペアンプの入力電圧、出力電圧を同時に測定し, 常に一致することを確認.
- オペアンプの出力電圧と LED に加わる電圧から, 100 Ω 抵抗と LED に流れる電流が概算できる.

第 5 章

Octave のセットアップと基本操作

Gnu Octave は商用の数式処理ソフトウェア MatLAB と同様のインターフェイスを備えた数式処理, プログラミングシステムである. これを利用し, 簡単なデータ処理, 電気回路シミュレーションの演習を行う.

5.1 セットアップ

Gnu Octave の公式サイトは,

http://www.gnu.org/software/octave/

である. ソフトウェアのダウンロードは

http://www.gnu.org/software/octave/download.html

からの指示に従う. 2020 年 4 月現在 Windows 版の最新バージョンは 5.2.0 である (より新しいバージョンが存在する場合, 基本的には最新版を用いればよい). 上記 URL の Windows タブより (または https://ftp.gnu.org/gnu/octave/windows/ より),

octave-5.2.0_1-w64-installer.exe

をダウンロード, 実行する.

注釈: 上記ダウンロードファイルは, 64 ビット対応版である. 32 ビット版が必要な場合, octave-5.2.0_1-w32-installer.exe をダウンロード, 実行する.

注釈: Windows へのインストールでは途中いくつかのメッセージ (ex. Windows のバージョンが新しい, Java Runtime Engine が必要) が表示されるが, 全て「OK」でよい.

5.1.1 ドキュメント

公式ドキュメントは

https://www.gnu.org/software/octave/doc/interpreter/

である. この文書は GUI 版の「ドキュメント」のタブや, スタートメニュの *Gnu Octave 5.2.0 → Octave (HTML)* *Gnu Octave 5.2.0 → Octave (PDF)* からも参照できる.

日本語訳は, バージョンは古いが増田氏による

http://www.obihiro.ac.jp/~suzukim/masuda/octave/html/index.html

がまとまっている (2020 年 6 月現在リンク切れ).

このほか多数のヘルプ, チュートリアルが web 上にて見つかるので, 適宜参考にするとよい.

また, MATLAB との互換性が高いため, MATLAB ドキュメンテーションも参考になる.

http://jp.mathworks.com/help/matlab/

5.2 実行 (GUI 版)

デスクトップのショートカット, またはスタートメニューより *Gnu Octave 5.2.0 → Octave 5.2.0 (GUI)* を検索, これをクリックすると, GUI ウインドウが起動する.

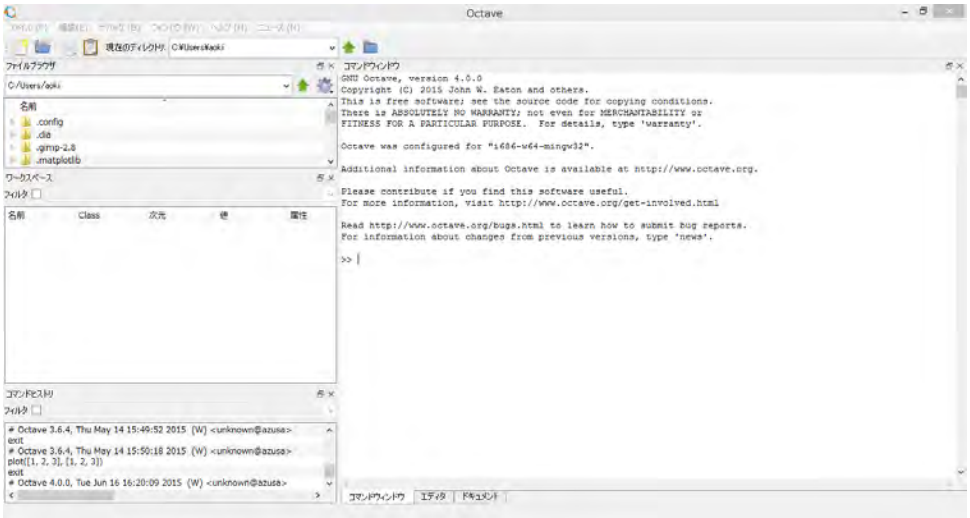


図 5.1　GUI 版 Octave の起動画面 (画面はバージョン 4.0.1 のもの. 以下の実行例等も, バージョン・環境の違いにより多少の差異を生じる可能性がある.)

5.2.1 画面構成

画面は次のパートに分かれている。

- 左半分
  - ファイルブラウザ
  - ワークスペース (変数リストの一覧)
  - コマンドヒストリ (コマンドウィンドウでの入力履歴)
- 右半分

  これらの 3 つのウインドウは画面下部のタブで切り替える.
  - コマンドウィンドウ
  - エディタ
  - ドキュメント

注釈: スタートメニューより *Gnu Octave 5.2.0 → Octave 5.2.0 (CLI)* を選ぶと, コマンドライン版が起動する. 内容は GUI 版におけるコマンドウィンドウの部分と同じである.

5.3 チュートリアル

以下の操作は全てコマンドウィンドウにて入力, 出力結果を得る (なので CLI 版でも同様の結果が得られるはず).

> **警告:** GUI 画面のコマンドウィンドウ, エディタ等は日本語をはじめとする非 ascii 文字への対応が完全ではない. 特に, コマンドウィンドウの入力において, 日本語入力への切り替えは極力行わないこと.

5.3.1 電卓

関数電卓として利用できる.

```
>> 1 + 2
ans =  3
>> ans * 4
ans = 12
```

ここで **ans** は計算結果を暗黙に格納するための変数である. GUI の場合, 生成された変数は ワークスペースにその変数の種類 (Class と次元) その値が一覧として表示される. また, **ans** に格納された値を再利用することもできる (この場合 **ans** は更新される).

入力されたコマンドはコマンドヒストリに記録される.

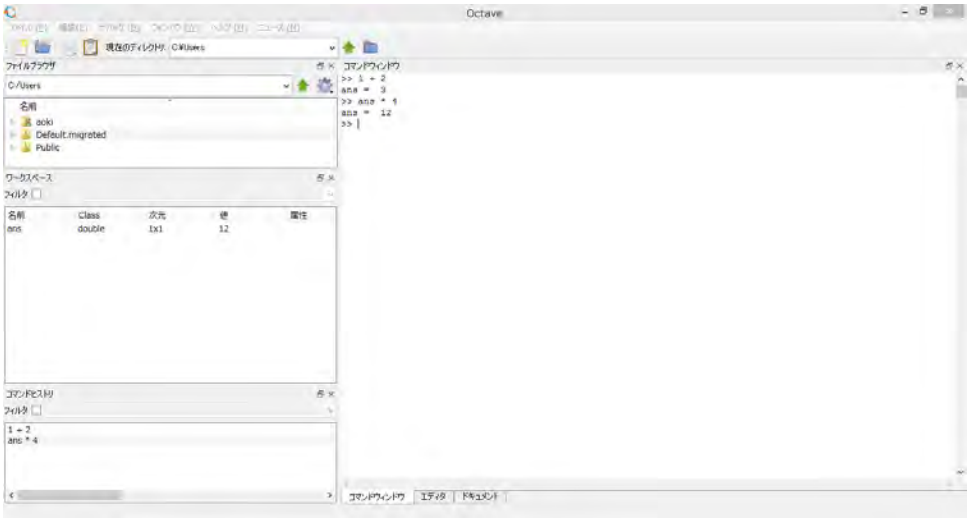


図 5.2　コマンドウィンドウへの入力, 並びにワークスペース, コマンドヒストリの更新の様子

注釈: コマンドウィンドウ, ワークスペース, コマンドヒストリは, メニューの **編集** → **〇〇のクリア** を選ぶことで, それぞれ消去することができる.

四則演算の他, 括弧, 特殊関数なども利用できる. また, 数値は基本的には倍精度小数 (double 型) である. 除算時の整数への丸めを考慮する必要はない. 利用できる主な演算, 関数は, **ドキュメント** タブに表示される *Expressions* → *Arithemtric Operators* または *Arithmetric* の項目で述べられている.

利用できる主な演算子:

```
+ - * / ( ) ^
```

利用できる主な関数:

```
exp log log10 log2 sin cos sqrt ...
```

使用例:

```
>> (1 + 2) * 3 + 4 / 5
ans =  9.8000
>> sqrt(3^2 + 4^2)
ans =  5
```

5.3.2 変数の扱い

前項の例では, 計算結果は暗黙に **ans** 変数に代入され, またこの値を参照することができた. 代入先の変数を明示するには, **=** 演算子を利用する.

```
>> width = 1366
width =  1366
>> height = 768
height =  768
>> area = width * height
area =  1049088
```

各入力の末尾に **; (セミコロン)** を付加すると, 入力した式及び値の結果が表示されない. これは後述するスクリプトファイルの作成, 実行時に便利である.

```
>> width = 1336;
>> height = 768;
>> area = width * height;
>> width
width =  1336
>> height
height =  768
>> area
area =  1026048
```

5.3.3 ベクトル、行列の扱い

初期化

ベクトル (1 次元配列) は, **[] (角括弧)** の中に複数の要素を **, (カンマ)** で区切って記述する. 行列 (2 次元配列) は各列の要素を **,** で区切り, 行要素の境を **; (セミコロン)** で区切る.

```
>> v = [1, 2, 3, 4, 5]
v =

   1   2   3   4   5

>> m = [11, 12, 13; 21, 22, 23]
m =

   11   12   13
   21   22   23

>> w = [6; 7; 8; 9; 10]
w =

   6
   7
   8
```

(次のページに続く)

(前のページからの続き)

```
   9
  10
```

size() 関数はベクトル, 行列の次元を [行 (row), 列 (column)] 形式で得る関数である.

```
>> size(v)
ans =

   1   5

>> size(m)
ans =

   2   3

>> size(w)
ans =

   5   1
```

このように, 行ベクトルは 1 行 n 列, 列ベクトルは n 行 1 列の行列として扱われる.

要素へのアクセス (Indexing)

個々の要素の値を参照するには, 変数名のあと () で要素番号を指定する. 要素番号は 1 から始まる.

注釈: # または % 記号以降はコメントであり無視される. 実際に入力する必要はない.

```
>> v(2)
ans =  2
>> v(2) = 7  # 要素を指定して代入
v =

   1   7   3   4   5

>> v(2) -= 5  # v(2) = v(2) - 5 と同様の操作
v =

   1   2   3   4   5

>> v(end)  # end は最後尾の要素番号を示す.
ans =  5
>> m(1,2)  # 行列要素は (行 (row), 列 (column)) の順で指定する.
ans =  12
```

要素を指定する場所を配列とし, 複数の要素をベクトルや行列として取り出すことができる. この際範囲を示す **:** 演算子も利用できる. 詳細はドキュメントの *Expressions* → *Index Expressions* を参照のこと.

```
>> v([2, 3, 4])  # v(2, 3, 4) はエラー
ans =

   2   3   4

>> 2:4  # 範囲のベクトルを生成する ":" 演算子. 4 が含まれることに注意.
ans =

   2   3   4

>> v(2:4)  # v([2, 3, 4]) と同じ
ans =

   2   3   4

>> v(5:-2:2)  # 5 から始まって-2 づつ加算, 2 を下回るまで
ans =

   5   3

>> m(:,[2,3])  # ":" 一つだけだと"全ての要素 (1:end)"と同じ
ans =

   12   13
   22   23
```

行列, ベクトルに対する演算

加減算, スカラー倍, 乗算, 転置等の演算が可能である.

```
>> v = 1:5  # v = [1, 2, 3, 4, 5] と同じ
v =

   1   2   3   4   5

>> w = 6:10
w =

    6    7    8    9   10

>> v + w  # 加算
ans =

    7    9   11   13   15

>> v - w  # 減算
ans =

  -5  -5  -5  -5  -5

>> 2 * v  # スカラー倍
```

(次のページに続く)

(前のページからの続き)

```
ans =

    2    4    6    8   10

>> v * w   # 内積を求めたいが，ベクトル（行列）の次元が一致しないとエラー
error: operator *: nonconformant arguments (op1 is 1x5, op2 is 1x5)
>> w'   # "'" を付与すると転置．transpose(w) も同様
ans =

    6
    7
    8
    9
   10

>> v * w'   # これだと計算可能
ans =  130
>> dot(v, w)   # 内積を求める関数を利用してもよい．
ans =  130
>>
```

ベクトル, 行列の個々の要素に対する演算も可能である.

- あらかじめ提供されている特殊関数の場合, ベクトルを引数にすることで, 要素ごとに関数を適用したベクトルを生成する
- スカラー a とベクトル b との加減算乗除演算等は, 個々のベクトル要素 b(1), b(2), ... と a との演算結果の配列を生成する.
- .*, ./ 等, ピリオドが前についた演算子は, ベクトル (または行列) の要素ごとの演算となる.

```
>> a = [1:3:10]
a =

    1    4    7   10

>> exp(a)
ans =

  2.7183e+000  5.4598e+001  1.0966e+003  2.2026e+004

>>> 1 - a
ans =

   0  -3  -6  -9

>>> a ./ [2, 4, 6, 8]
ans =

   0.50000   1.00000   1.16667   1.25000
```

5.3.4 大規模データセットとファイル入出力

Octave をデータ処理用ツールとして利用する場合, ベクトル, 行列はその数学的意味合いからやや離れ, 単なる多数のデータの集合として扱うことが多い. たとえば, ある測定やシミュレーションを実施し, 時系列に添って電圧, 電流値を 1000 点にわたり取得したとする. この時得られたデータは行数 1000, 列数 3 の行列ととらえることが可能である.

一定時間間隔の様に, あるルールに従ったデータを多数作成したい場合は **:** (コロン) 演算子や, **linspace**, **logspace** 関数が利用できる.

```
>> ts = 0:0.01:1.0
ts =

 Columns 1 through 9:

    0.00000    0.01000    0.02000    0.03000    0.04000    0.05000    0.06000    0.
↪07000    0.08000

 Columns 10 through 18:

    0.09000    0.10000    0.11000    0.12000    0.13000    0.14000    0.15000    0.
↪16000    0.17000

 # 以下省略 (ページャーによる長大データの表示, 下記注釈を参照のこと)

>> size(ts)  # サイズを確認 0 および 1.0 を含むので要素数は 101
ans =

     1   101
```

注釈: 上記の ts 変数の様に, 対話環境において多数の要素を含むベクトル, 行列を表示する場合, ページ送りのプロンプトが表示されることがある. この時,

- f で次の要素の表示
- b で前の要素の表示
- q で表示を中断

の操作を行う.

単位行列, ゼロ行列等, よく利用する行列, ベクトルについてはこれらを生成する関数が用意されている.

```
>> eye(3)  # 3x3 の単位行列
ans =

Diagonal Matrix

   1   0   0
```

(次のページに続く)

(前のページからの続き)

```
   0   1   0
   0   0   1

>> zeros(2, 3)  # 2x3 のゼロ行列
ans =

   0   0   0
   0   0   0

>> ones(2, 3)  # 要素がすべて 1 の行列
ans =

   1   1   1
   1   1   1
```

行列の要素を csv ファイルに出力するには **csvwrite** 関数を用いる．入力には **csvread** または **dlmread** 関数を用いる．

```
>> x = 0:0.01:1.0;  # 結果の出力を抑制するために 末尾に ";" を付加している
>> y1 = x * 2;
>> y2 = 2 * x .^ 2;  # .^ は要素毎のべき乗
>> tbl = [x; y1; y2]'  # x, y1, y2 を行列にする． ' をつけて転置
tbl =

   0.00000   0.00000   0.00000
   0.01000   0.02000   0.00020
   0.02000   0.04000   0.00080
   0.03000   0.06000   0.00180
   0.04000   0.08000   0.00320
   0.05000   0.10000   0.00500
   0.06000   0.12000   0.00720
   0.07000   0.14000   0.00980
   0.08000   0.16000   0.01280
   0.09000   0.18000   0.01620
   0.10000   0.20000   0.02000
   0.11000   0.22000   0.02420
   # 後略

>> csvwrite("csvout.csv", tbl)  # "csvout.csv" というファイルに出力
>> tbl2 = csvread("csvout.csv")  # "csvout.csv" を読み込み tbl2 変数とする
tbl2 =

   0.00000   0.00000   0.00000
   0.01000   0.02000   0.00020
   0.02000   0.04000   0.00080
   0.03000   0.06000   0.00180
   0.04000   0.08000   0.00320
   0.05000   0.10000   0.00500
   0.06000   0.12000   0.00720
   0.07000   0.14000   0.00980
   0.08000   0.16000   0.01280
```

(次のページに続く)

(前のページからの続き)

```
  0.09000   0.18000   0.01620
  # 後略
```

注釈: ファイルを読み書きする位置は, GUI 版だと "現在のディレクトリ" あるいは "ファイルブラウザ" に表示されているフォルダが基準となる. 必要に応じて移動させること.

なお, **編集** → **設定** ダイアログの **一般** → *Octave Startup* 項目を設定することで, Octave 起動時のディレクトリを変更できる.

CLI 版の場合, pwd 関数で基準となるフォルダを確認し, cd 関数でフォルダの移動を行う.

注釈: GUI 版の場合, csvread 関数に代わりファイルブラウザより該当のファイルを右クリック, 「データのロード」を選ぶことで同様の操作となる. この場合, ファイル名に相当する変数名がつけられる.

5.3.5 グラフの描画

前項で作成した, x, y1, y2 をグラフに表示する. このために **plot** 関数を用いる.

原則として plot 関数では,

1. 1 番目のグラフの x 座標データ (1 次元配列)
2. 1 番目のグラフの y 座標データ (同上)
3. 2 番目のグラフの x 座標データ (同上)
4. 2 番目のグラフの y 座標データ (同上)
5. :

のように描画するデータを指定する.

```
>> plot(x, y1, x, y2)
```

コマンドを実行するとプロットウインドウが表示される (初回実行時は時間がかかるかもしれない).

プロットウインドウは簡単な拡大縮小, 軸・グリッドの表示, 注釈の書き込み機能等を有している. 必要であれば, ここで調整を行うことができる.

グラフを保存するにはプロットウインドウのメニューから **ファイル** → **保存** または **別名で保存** を選ぶ. 保存形式はファイル名に付与する拡張子により決定される. 標準では .pdf の拡張子が与えられており, この場合 PDF 形式 (Adobe Acrobat 等で読むことのできるデータ) となる. この他には,

- .svg : SVG 形式 (ベクターグラフィックデータ.)
- .png : PNG 形式 (ビットマップ)

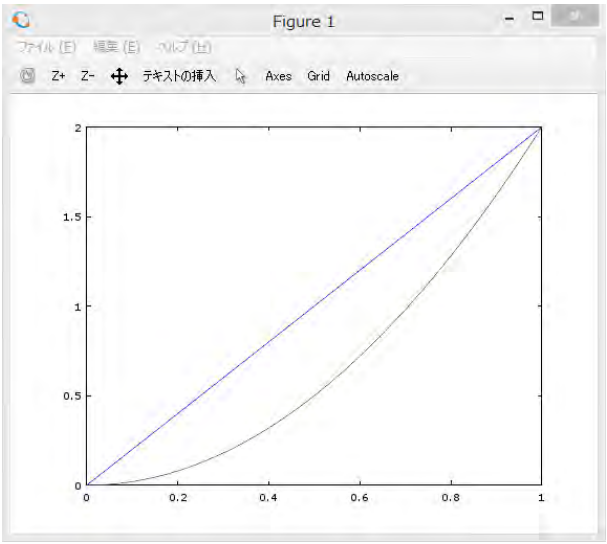


図 5.3　plot 関数の実行結果

- .jpg : JPEG 形式 (ビットマップ)

等の形式で保存できる.

注釈: コマンド上でプロットデータを保存する場合, **print** 関数または **saveas** 関数を利用する.

グラフを作成する場合, 少なくとも

- 凡例 (特にグラフが複数ある場合)
- 縦軸, 横軸のラベル

を備えることが一般的である.

```
>> plot(tbl(:,1), tbl(:,2), ";2*x;", tbl(:,1), tbl(:,3), ";2*x^2;")    # 凡例情報つきでプロット
>> xlabel("x", "fontsize", 18)  # x 軸のラベル. オプションでフォントサイズを変える
>> ylabel("y", "fontsize", 18)
>> legend("location", "northwest")  # 凡例 (legend) を左上隅に移動する
```

5.3.6 一連の手続きをプログラム (スクリプト) にする

一連のコマンドをプログラムファイルとして保存しておくと, 後に再利用することもできる.

右画面下部にある **エディタ** タブをクリックしエディタ画面に切り替える. この状態で, <unnamed>という一時ファイルを編集できる状態になっている. そこで, 以下の内容を記載する (# の行はタイプする必要はない).

```
% Octave ではファイルが function で始まる場合,
% スクリプトではなく関数ファイルとして扱われる.
% そのことを避けるため慣習的に 1; を冒頭に記載する
```

(次のページに続く)

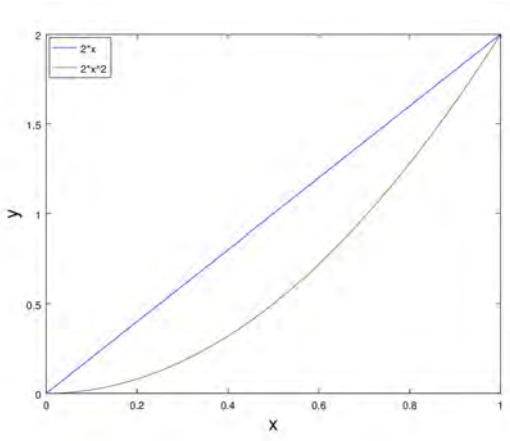


図 5.4　ラベル, 凡例を挿入した結果

(前のページからの続き)

```
1;

# data preperation
x = 0:0.01:1.0;
y1 = 2 * x;
y2 = 2 * x .^ 2;

# save as csv data format
tbl = [x; y1; y2]';
csvwrite("csvout.csv", tbl);

# plot and save
plot(x, y1, ";2*x;", x, y2, ";2*x^2;");
xlabel("x", "fontsize", 18);
ylabel("y", "fontsize", 18);
legend("location", "northwest");
print("plot.pdf");
```

続いてエディタのメニューより **実行 → ファイル保存して実行** を選択する. 適当な名前 (例:myprog.m, 拡張子は .m とする) を指定すると, ファイルが保存され, プログラムが実行される. 上記プログラムの例では, csvout.csv, plot.pdf が保存される.

> **警告:** この時保存するファイル名, 並びに保存する全ての階層のフォルダ名に日本語, 全角記号などを含めないこと. Windows 版 Octave-4.2.1 での不具合で, 今後のバージョンアップで解決される予定.

注釈: スクリプトが長時間応答しない場合等, スクリプトの実行を途中で中止したい場合, コントロールキーを押しながら'c' キーをタイプする (Ctrl+c と表記することが多い).

注釈:

スクリプトファイル冒頭に **1;** を記載する理由については， https://www.gnu.org/software/octave/doc/interpreter/Script-Files.html

(参考対訳は) http://www.obihiro.ac.jp/~suzukim/masuda/octave/html/octave_70.html

を参照のこと.

ここで一旦 octave ソフトウェアを終了, 再起動する. ファイルブラウザを使って myprog.m ファイルを探し, 右クリック →「実行」を選ぶと, このプログラムが実行される.

注釈: CLI 版 または GUI 版のコマンドウィンドウ内だと, myprog.m のあるフォルダに移動 (cd 関数) し,

```
>> run("myprog.m")
```

と, ファイル名指定すると, myprog.m がロード, 実行される.

5.3.7 関数の定義

単純に 2 つの引数の和を返す関数 (add2values とする) は次のように書ける. コマンドウィンドウに次のコマンドを入力する.

```
>> function ret = add2values(a, b)
  ret = a + b;
endfunction
>> add2values(1, 2)
ans =  3
```

この一連のコマンドで重要な部分は以下の通り.

- 関数は **function ~ endfunction** の間で定義する.
- ここで a, b は関数の引数, ret は戻り値, これらの変数名は任意である.

関数ファイルの作成と利用

有用な関数は関数ファイルとして保存すると, 再利用できる.

エディタを用い, 次の内容 (上記コマンドウィンドウで入力したものと同じ) を add2values.m として保存する. ここでファイル内部で定義している関数名と拡張子を除いたファイル名を一致させることが必要である.

```
# add2values.m ファイルの内容
function ret = add2values(a, b)
  ret = a + b;
endfunction
```

Octave (GUI 版) を一度終了, 再起動したのち, add2values.m を保存したフォルダに移動する. そこで, 他の入力を行わず, add2values 関数を利用するコマンドを入力する. このとき関数名に対応した add2values.m が自動的に読み出され, 実行される.

```
>> add2values(1, 2)
ans =  3
```

複数の値を返す関数

Octave の関数により複数の値を返すことを考える. 例として 2 次元直行座標 (x, y) から極座標 (theta, r) への変換を行う関数を取り上げる. この場合, 次の 2 つの方法が考えられる.

1. 2 つのスカラー変数を受け取り, 2 つのスカラー値を返す

```
# mycart2pol_s2.m ファイルを次の内容で作成
function [theta, r] = mycart2pol_s2(x, y)
  theta = atan2(y, x);
  r = sqrt(x^2 + y^2);
endfunction
```

```
>> [theta, r] = mycart2pol_s2(-1, 1)
theta =  2.3562
r =  1.4142
>> theta / pi * 180  # theta はラジアン表示なので度数表示に. pi は定数として利用できる.
ans =  135
>>
```

2. 1 つの 2 要素ベクトルを受け取り, 1 つの 2 要素ベクトルを返す

```
# mycart2pol_v1.m ファイルを次の内容で作成
function pol = mycart2pol_v1(cart)
  # pol(1) pol(2) とインデックス指定した代入により,
  # pol は自動的にベクトルとして扱われる
  pol(1) = atan2(cart(2), cart(1));
  pol(2) = sqrt(cart(1)^2 + cart(2)^2);
endfunction
```

```
>> mycart2pol_v1([-1, 1])
ans =

   2.3562   1.4142
```

注釈: より汎用性のある cart2pol 関数が標準で定義されている (help cart2pol で表示されるファイルを参照)

第 6 章

Octave による LCR 直列共振回路の解析

LCR 直列共振回路の方程式を Octave を用いて解析する.

注釈: Octave のセットアップと基本的な使い方は, *Octave のセットアップと基本操作* を参照すること.

また補足課題を *Octave 補足課題* にまとめている. 適宜参照すること.

LCR 直列共振回路の方程式は以下で示される.

$$\begin{bmatrix} \frac{\mathrm{d}i}{\mathrm{d}t} \\ \frac{\mathrm{d}v}{\mathrm{d}t} \end{bmatrix} = M \begin{bmatrix} i \\ v \end{bmatrix}, \qquad M = \begin{bmatrix} -\frac{R}{L} & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix} \tag{6.1}$$

6.1 ベクトル場の図示

$-1 \leq i, v \leq 1$ の範囲の (i, v) に対し, $(\mathrm{d}i/\mathrm{d}t, \mathrm{d}v/\mathrm{d}t)$ の向き, 大きさをベクトルとして図示する.

ベクトルの図示には, **quiver** 関数を利用する. 行列の定義, 微係数ベクトルを計算する座標の定義, 微係数ベクトルの算出も含めると, 次のようなプログラムとなる.

リスト 6.1　ex1_1.m ベクトル場の作図

```
% ex1_1.m ベクトル場の作図

% Octave ではファイルが function で始まる場合,
% スクリプトではなく関数ファイルとして扱われる.
% そのことを避けるため慣習的に 1; を冒頭に記載する
1;

% (i, v) -> (di/dt, dv/dt) の写像を定義する行列
R = 3;
L = 1;
C = 1;
M = [-R/L, -1/L; 1/C, 0];

% -1 <= (i, v) <= 1 の空間を (10, 10) 等分する.
i_x = linspace(-1, 1, 11);
```

(次のページに続く)

(前のページからの続き)

```
v_y = linspace(-1, 1, 11);

% meshgrid 関数により, 各グリッドにおける
% x, y 座標の 2 次元配列 (=行列) を作成する.
% x, y はそれぞれ length(i_x) x length(v_y) の行列である.
% サイズ, 内容を確認すること.
[x, y] = meshgrid(i_x, v_y);

% (u(s,t), v(s,t)) = M * (x(s,t), y(s,t))^T を求める.
% (x, y) のそれぞれの要素が 2 次元配列なので, 演算を明示的に展開する必要がある.
u = M(1, 1) * x + M(1, 2) * y;
v = M(2, 1) * x + M(2, 2) * y;

% プロットする. quiver 関数を使用する
quiver(x, y, u, v);

% グラフの描画範囲, ラベルの付与等の調整
xlim([-1.2, 1.2]);
ylim([-1.2, 1.2]);
xlabel("I")
ylabel("V")
```

実行結果は次の通り.

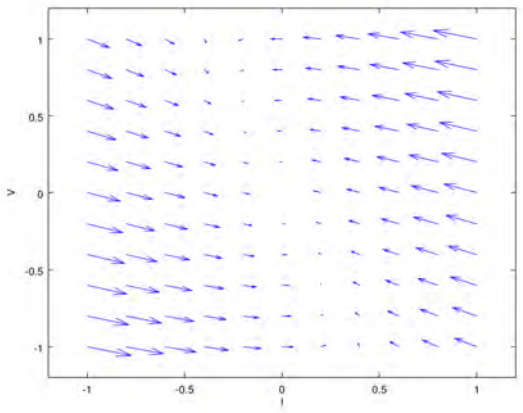


図 6.1　ex1_1.m の実行結果

6.2 微分方程式の数値解を求める

微分方程式を解くための関数, **lsode** を利用し, 初期値 $(i_0, v_0) = (0.8, 0.8)$ からの時間発展を求める. lsode の詳しい解説は *Octave* 補足課題 を参照のこと.

リスト 6.2　ex1_2.m 微分方程式ソルバの利用

```
% ex1_2.m 微分方程式ソルバの利用

1;
```

(次のページに続く)

(前のページからの続き)

```
% (i, v) -> (di/dt, dv/dt) の写像を定義する行列
R = 3;
L = 1;
C = 1;
global M;  % prob 関数から参照するため global 指定が必要.
M = [-R/L, -1/L; 1/C, 0];

% 微分方程式を表す関数
function ret = prob(x, t)
  global M;
  ret = M * x;
endfunction

% シミュレーション結果を出力する間隔
ts = linspace(0, 10, 101);

% ODE を解く
[vcn, stat, msg] = lsode("prob", [0.8, 0.8], ts);

% 相平面にプロット
plot(vcn(:,1), vcn(:,2), "-rx");  % 赤色で描画.
hold on;  % この後の quiver 関数によるプロットも合わせて書くため.

% ベクトル図を描画する. 以下は, ex1_1.m の内容と同じ.
% -1 <= (i, v) <= 1 の空間を (10, 10) 等分する.
i_x = linspace(-1, 1, 11);
v_x = linspace(-1, 1, 11);

% meshgrid 関数により, 各グリッドにおける x, y 座標の 2 次元配列 (=行列) を作成する.
[x, y] = meshgrid(i_x, v_x);

% (u(s,t), v(s,t)) = M * (x(s,t), y(s,t))^T を求める.
u = M(1, 1) * x + M(1, 2) * y;
v = M(2, 1) * x + M(2, 2) * y;

% プロットする.
quiver(x, y, u, v);

% グラフの描画範囲, ラベルの付与等の調整
xlim([-1.2, 1.2]);
ylim([-1.2, 1.2]);
xlabel("I");
ylabel("V");
```

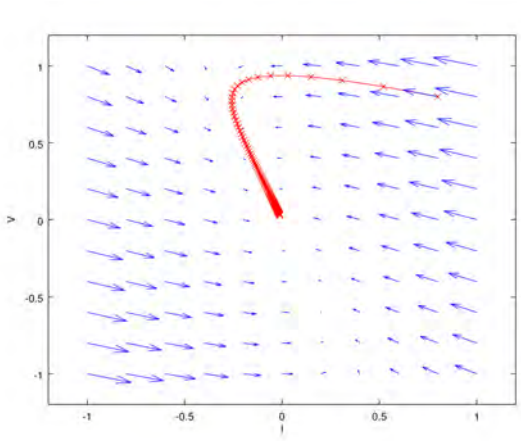


図 6.2　ex1_2.m の実行結果

6.3 固有値・固有ベクトルを求め相平面上に図示する

行列 M に対する固有値, 固有ベクトルを求め, 先の結果と共に図示する. 固有値の計算には **eig** 関数を利用する.

リスト 6.3　ex1_3.m 固有値, 固有ベクトルの導出と図示

```
% ex1_3.m 固有値, 固有ベクトルの導出と図示

1;

% (i, v) -> (di/dt, dv/dt) の写像を定義する行列
R = 3;
L = 1;
C = 1;
M = [-R/L, -1/L; 1/C, 0];

% eig 関数は行列の固有値, 固有ベクトルを求めるための関数.
[ks, lambdas] = eig(M);

% ks は固有ベクトル, lambda の対角成分は固有値が得られ,
%   lambdas(1,1) * ks(:,1) = M * ks(:,1)
%   lambdas(2,2) * ks(:,2) = M * ks(:,2)
% が成立する.

k1 = ks(:,1);
k2 = ks(:,2);

% 値をコンソールに表示する
lambdas(1,1)
k1
lambdas(2,2)
k2

% 描画する
% 1. k1, k2 のベクトルを長さ 2 となるまで延長
```

(次のページに続く)

(前のページからの続き)

```
% 2. (k1 と -k1) (k2 と -k2) を結ぶ直線を引く.
k1 = (2 / norm(k1)) * k1;
k2 = (2 / norm(k2)) * k2;

% k1 は緑の実線 (solid), k2 は緑の破線 (dashed)
plot([k1(1), -k1(1)], [k1(2), -k1(2)], "g",
     [k2(1), -k2(1)], [k2(2), -k2(2)], "--g");
xlim([-1.2, 1.2]);
ylim([-1.2, 1.2]);

hold on;  % 引き続き同じ図にプロットを書く場合指定する.

% 以下, ex1_2.m の図をプロットする
run("ex1_2.m") % 同じフォルダにある ex1_2.m を実行.
```

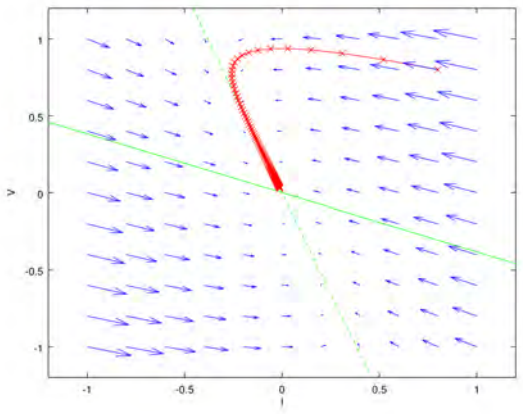


図 6.3　ex1_3.m の実行結果

6.4 LTSpice との比較

微分方程式の数値解を求める に対応する回路を LTSpice で作成し, シミュレーション結果を比較せよ.

対応する回路を LTSpice で作成すると以下のようになる.

$(i_0, v_0) = (0.8, 0.8)$ の初期値を設定するため, .ic ディレクティブを用い

```
.ic V(Vc)=0.8 I(L1)=0.8
```

を設定している. これに対し過渡解析 (.tran 解析) を実施, V(Vc), I(L1) の時間発展は以下のようになる.

プロット結果をエクスポートすると, 各列が (time, V(vc), I(L1)) のタブ区切りテキストが得られる (LCR.txt とする).

```
time V(vc)    I(L1)
0.000000000000000e+000        8.000000e-001    8.000000e-001
9.999999717180685e-010        8.000000e-001    8.000000e-001
```

(次のページに続く)

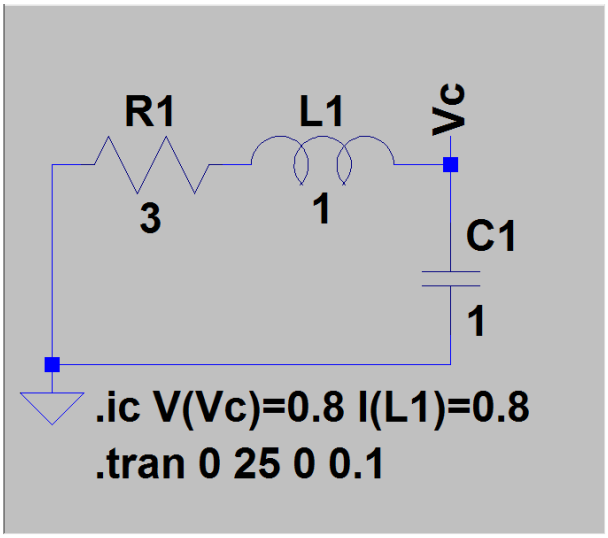


図 6.4　LTSpice による式 (6.1) と同等の回路

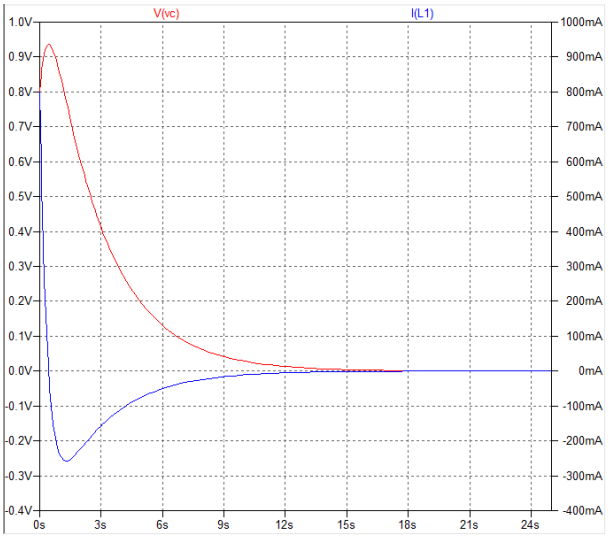


図 6.5　LTSpice による LCR 回路の過渡解析結果 ($(i_0, v_0) = (0.8, 0.8)$)

(前のページからの続き)

```
1.999999943436137e-009        8.000000e-001    8.000000e-001
2.729265542433487e-002        8.203102e-001    7.167977e-001
(後略)
```

このデータを, 微分方程式の数値解を求める の結果と合わせてプロットする. タブやカンマ等で区切られたテキストファイルの読み込みには, **dlmread** または **csvread** 関数を用いる.

リスト 6.4　ex1_4.m LTSpice のデータをプロットする

```
% ex1_4.m
% LTSpice のデータ ("LCR.txt") をプロットする
%   1 行目はタイトルなのでスキップする
%   1 列目:時刻, 2 列目:V(vc), 3 列目:I(L1)
%   各列はタブ ("\t") 区切り
```

(次のページに続く)

(前のページからの続き)

```
1;

% LTSpice のデータ読み込み
% 0 基準で, 1 行目, 0 列目から読み込み
sptbl = dlmread("LCR.txt", "\t", 1, 0);

plot(sptbl(:,3), sptbl(:,2), '-ko'); % データ間を直線で結び、黒丸でプロット
hold on;

% ex1_2.m を続いて呼び出す
run("ex1_2.m");
```

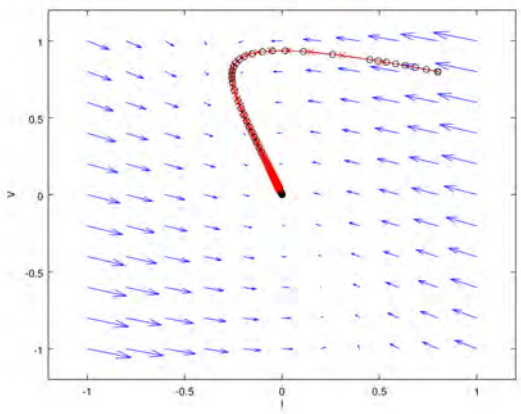


図 6.6　ex1_4.m の実行結果

6.5 減衰パラメータを変えたシミュレーション

上記の R, L, C パラメータはテキスト式 2.11 の条件を満たす. 式 2.12, 2.13 を満たすよう, R を選び,

- 微分方程式の数値解を求める と同様のプロット (ただし, 初期値は $(i_0, v_0) = (0, 1)$ とする)
- v(t) の時間変化

を図示せよ. さらに解析解, LTSpice の結果と比較せよ.

第 7 章

LCR 直列共振回路の過渡応答測定

これまで LCR 直列共振回路の特性を,

- LTSpice によるシミュレーション
- Octave(matlab) による数値計算

の各手法により解析を行ってきた. 今回は実際に回路を作成, 測定することで, 理解を深める.

7.1 準備

事前に *NI-myDAQ のセットアップ, ファンクションジェネレータとオシロスコープ* に従い, ファンクションジェネレータ (FG), オシロスコープ (OSC) の動作確認を行うこと.

7.2 回路図と回路パラメータの導出

測定時の回路は 図 7.1 のようになる.

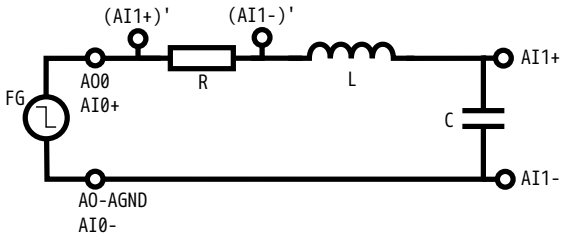


図 7.1　LCR 直列共振回路実験の回路図

過渡応答を見るため, myDAQ の FG からは十分に長い周期の矩形波を出力する. 仮に L = 10 [mH], C = 1 [μ F] とする. 電気電子回路演習テキストの式 2.6 より,

$$\omega_0 = 10^4 [\mathrm{rad/s}] \quad (\simeq 1600 [\mathrm{Hz}])$$

が得られる. NI のウェブサイト情報 (http://www.ni.com/pdf/manuals/373061f.pdf) によると, myDAQ のサンプリング速度は最大毎秒 200k サンプル (200kS/s) である. この速度は 1.6kHz の波形データを取得する為にある

程度許容できる速度である. また, FG から出力する矩形波の周波数は, ω_0 より十分低い必要がある.

続いてダンピング条件 (式 2.12) となる R の条件を考えると,

$$R_0 = 2\sqrt{\frac{L}{C}} = 200[\Omega]$$

を得る. R の値を R_0 を境界に変化させることで, 式 2.11-2.13 の条件が得られることが期待される.

7.3 測定

7.3.1 素子の配置

図 7.1 に従い, ブレッドボード上に LCR 直列共振回路を作成する. ここで, L, C, R は先ほどの議論より,

- L = 10 [mH]
- C = 1 [μ F]
- R = 10 [Ω] またはこれに近い値

とする. これは式 2.13 の条件に相当し, 2 つの時間に関するパラメータ ω_0, α の妥当性を検証することができる.

注釈:

1. コンデンサはセラミックコンデンサ (青色でパッケージされた素子) を使用すること. (電解コンデンサ (円筒型の素子) は極性があるため不向き)
2. ブレッドボードの仕組みについては PandA の **授業資料** → **実験キットの内容** → **ブレッドボード** を参照のこと.

7.3.2 myDAQ との接続

続いて, FG の出力, OSC の入力を接続する. 図 7.1 の (AO0, AO-GND), (AI0+, AI0-), (AI1+, AI1-) に相当する箇所と, myDAQ の端子を接続する. 前 2 者は電気的に同じ場所に接続し, FG の出力信号を AI0 チャネルで取得することになる.

7.3.3 FG の設定

NI ELVISmx の Function Generator ソフトウェアを起動し, 以下のパラメータを設定する.

- [Waveform Settings]
  - [波形] 矩形波
  - [Frequency] 50 Hz
  - [Amplitude] 0.2 Vpp
  - [DC Offset] 0.1 V
  - [Duty Cycle] 50 %
- [Instrumental Control]
  - [Device] myDAQ1 (NI myDAQ) (または有効な myDAQ デバイスであることを確認)
  - [Signal Route] AO0

注釈: ここで FG の出力電圧が大き過ぎると, 出力電流が飽和し, 正しい矩形波が出力できなくなる.

7.3.4 OSC の設定

OSC 側では, FG からの出力を AO0 で測定し, これを Negative トリガとして利用する.

- [Channel 0 Settings] 及び [Channel 1 Settings]
  - [Source] それぞれ (AI 0), (AI 1)
  - [Enabled] にチェックを入れる
  - [Scale Volts/Div] 50 mV (適宜変更してよい)
- [Timebase][Time/Div] 2 ms (適宜変更してよい)
- [Trigger]
  - [Type] Edge
  - [Source] Chan 0 Source
  - [Slope] Negative (標準では Positive なのでクリックして反転する)
  - [Level] 0.1 V
  - [Horizontal Position (%)] 10 (トリガ発生時刻に対応する x 座標を指定する)
- [Instrumental Control]

- [Device] myDAQ1 (NI myDAQ) (または有効な myDAQ デバイスであることを確認)
- [Acquisition Mode] Run Continuously

7.3.5 波形データの表示, データ出力

FG, OSC ソフトウェア双方の [Run] ボタンをクリックし, 信号の出力, 測定を行う. 図 7.2 のような波形が測定できる.

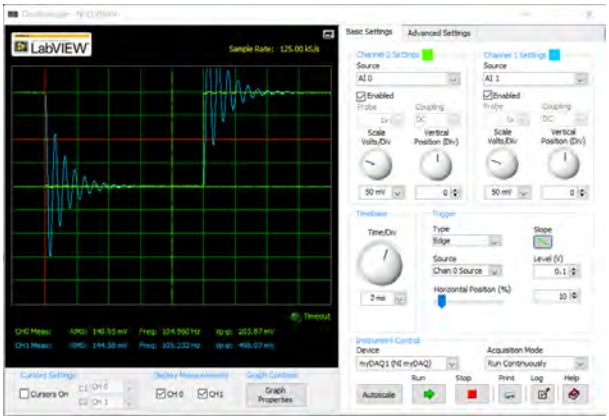


図 7.2　コンデンサ電圧の過渡応答 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz)

FG, OSC の入出力を一旦停止 ([Stop] ボタンをクリック) し, OSC の波形データを保存する ([Log] ボタンをクリック).

続いて, myDAQ の (AI1+, AI1-) 端子を 図 7.1 の ((AI1+)', (AI1-)') に相当する箇所と接続し, 抵抗に加わる電圧の過渡応答を測定, データを保存する.

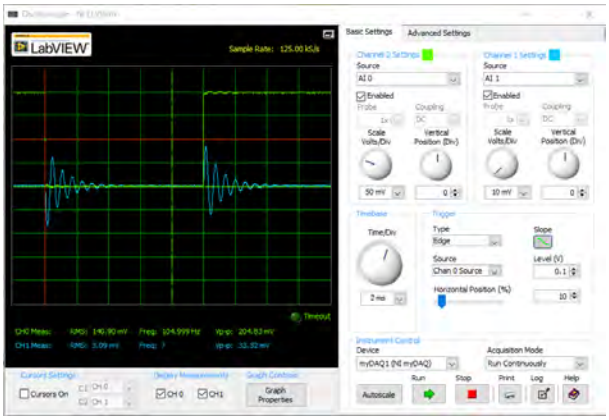


図 7.3　抵抗電圧の過渡応答 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz, Ch1 のレンジを拡大している)

7.4 データの整理, 解析

7.4.1 データの図示

OSC ソフトウェアから保存したデータは, *NI-myDAQ のセットアップ, ファンクションジェネレータとオシロスコープ* でも示したように,

- 先頭 5 行が波形全体に関する情報
- 6 行目以降が波形データ ["chan0 時分秒", "chan0 Y 値 (V)", "chan1 時分秒", "chan1 Y 値 (V)"]. ただし, 時刻部分は精度が不十分なので, 別途計算する必要がある.
- 各列はタブ記号で区切られている

である. excel や octave を用い, 以下の関係を図示する.

1. FG の出力と, コンデンサ電圧の時間変化
2. FG の出力と, 抵抗電圧の時間変化 (FG の出力が 1. とほぼ等しいことを確認すること)
3. 上記データの前半は放電過程に相当する. この部分のみを取り出し, 抵抗電圧-コンデンサ電圧の関係を図示. これは相平面図に相当する.

前項での測定例 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz) は以下の通り.

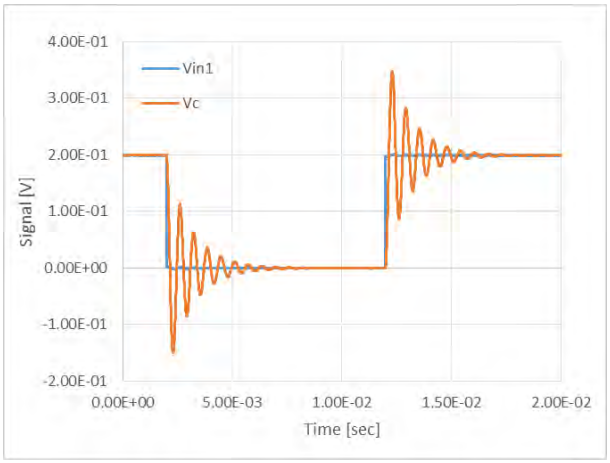


図 7.4　入力電圧とコンデンサ電圧の時間変化 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz)

7.5 パラメータを変えた測定

余力があれば, 上記の測定を, 抵抗 R の値を変え実施せよ. データを保存する際のファイル名にも注意が必要である. 今回の実験では,

- 抵抗値
- 測定箇所

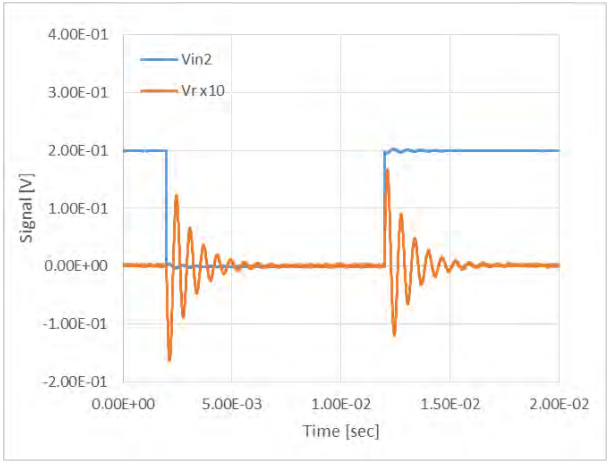


図 7.5　入力電圧と抵抗電圧の時間変化 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz, 抵抗電圧値は 10 倍に拡大している)

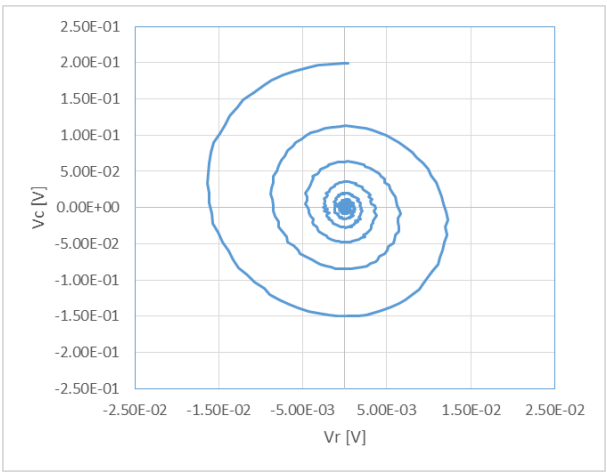


図 7.6　抵抗電圧とコンデンサ電圧の相平面図 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz, 放電過程のみ)

をパラメータとしてデータを取得している. 従ってこれらの情報がファイル名を見るだけで理解できるような命名ルールを適用することが望ましい. また場合によっては,

- 実験実施日
- データ取得の通し番号 (同じような条件で複数回データを取得するかもしれない)

といった情報も必要になるため, 実験ノートやデータ保存を行ったフォルダ内に, これらの情報を残しておくことが重要である.

なお, 上記手順では, 「測定」と「解析」の項を分けて記述しているが, 実際にはデータ取得とそのデータに対する解析はその都度実施し, 結果を確認しながら実験を進めるべきである.

7.6 他手法との比較

余力があれば, 得られた結果を理論, LTSpice, 数値解析等の結果と比較せよ.

第 8 章

LCR 直列共振回路の過渡応答測定 (AD2 利用)

これまで LCR 直列共振回路の特性を,

- LTSpice によるシミュレーション
- Octave(matlab) による数値計算

の各手法により解析を行ってきた. 今回は実際に回路を作成, 測定することで, 理解を深める.

8.1 準備

事前に *Analog Discovery 2 のセットアップ, ファンクションジェネレータとオシロスコープ* に従い, AnalogDiscovery2 のファンクションジェネレータ (FG, Wavegen ツール), オシロスコープ (OSC, Scope ツール) の動作確認を行うこと.

8.2 回路図と回路パラメータの導出

測定時の回路は 図 8.1 のようになる.

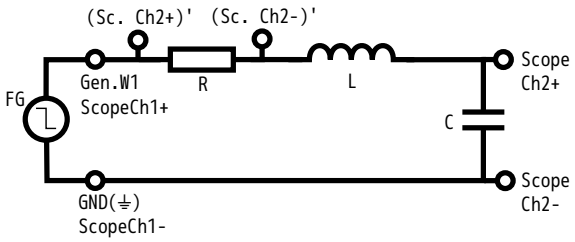


図 8.1　LCR 直列共振回路実験の回路図

過渡応答を見るため, FG からは十分に長い周期の矩形波を出力する. 仮に L = 10 [mH], C = 1 [μ F] とする. 電

気電子回路演習テキストの式 2.6 より,

$$\omega_0 = 10^4[\mathrm{rad/s}] \quad (\simeq 1600[\mathrm{Hz}])$$

が得られる. AD2 の仕様ページ (https://reference.digilentinc.com/reference/instrumentation/analog-discovery-2/specifications) によると, AD2 のサンプリング速度は最大毎秒 100M サンプル (100MS/s) である. これは, 1.6kHz の波形データを取得するには十分な速度である. また, FG から出力する矩形波の周波数は, ω_0 より十分低い必要がある.

続いてダンピング条件 (式 2.12) となる R の条件を考えると,

$$R_0 = 2\sqrt{\frac{L}{C}} = 200[\Omega]$$

を得る. R の値を R_0 を境界に変化させることで, 式 2.11-2.13 の条件が得られることが期待される.

8.3 測定

8.3.1 素子の配置

図 8.1 に従い, ブレッドボード上に LCR 直列共振回路を作成する. ここで, L, C, R は先ほどの議論より,

- L = 10 [mH]
- C = 1 [μ F]
- R = 10 [Ω] またはこれに近い値

とする. これは式 2.13 の条件に相当し, 2 つの時間に関するパラメータ ω_0, α の妥当性を検証することができる.

注釈:

1. コンデンサはセラミックコンデンサ (青色でパッケージされた素子) を使用すること. (電解コンデンサ (円筒型の素子) は極性があるため不向き)
2. ブレッドボードの仕組みについては PandA の **授業資料** → **実験キットの内容** → **ブレッドボード** を参照のこと.

8.3.2 Analog Discovery 2 との接続

続いて, FG の出力, OSC の入力を接続する. 図 8.1 の (Waveform Generator 1, GND), (Scope Ch1+, 1-), (Scope Ch2+, 2-) に相当する箇所にそれぞれ, 対応する AD2 のコネクタに接続する. 前 2 者は電気的に同じ場所に接続し, FG の出力信号を OSC の第 1 チャネルで取得することになる.

8.3.3 FG の設定

Weveforms の Webgen ツールを起動し, 以下のパラメータを設定する.

- [Channel 1]
  - [Type] Square
  - [Frequency] 50 Hz
  - [Amplitude] 100 mV
  - [Offset] 100 mV
  - [Symmetry] 50 %

注釈: ここで FG の出力電圧が大き過ぎると, 出力電流が飽和し, 正しい矩形波が出力できなくなる.

8.3.4 OSC の設定

OSC 側では, FG からの出力を Scope Ch1 で測定し, これを Negative トリガとして利用する.

- [測定方式, トリガ (画面上部メニュー)]
  - [Mode] Repeated, Normal
  - [Source] Channel 1
  - [Condition] Falling
  - [Level] 100 mV
- [Time (画面右)]
  - [Position] 6 ms (適宜変更してよい)
  - [Base] 2 ms/div
  - [Average] None
  - [Samples] Default
  - [Rate]: 400 Hz
- [Channel1 (画面右)]
  - [Channel1] にチェック
  - [Offset] 0 V
  - [Range] 50 mV/div

- [Channel2 (画面右)]
  - [Channel2] にチェック
  - [Offset] -100 mV
  - [Range] 50 mV/div

8.3.5 波形データの表示, データ出力

Wavegen, Scope ツール双方の [Run] ボタンをクリックし, 信号の出力, 測定を行う. 図 8.2 のような波形が測定できる.

図 8.2　コンデンサ電圧の過渡応答 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz)

Scope ツールの *File* → *Export* より波形データを保存する.

続いて, AD2 の (Scope Ch2+, Ch2-) 端子を 図 8.1 の (Sc. Ch2+)′ , (Sc. Ch2-)′ に相当する箇所と接続し , 抵抗 R に加わる電圧の過渡応答を測定, データを保存する.

図 8.3　抵抗電圧の過渡応答 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz, Ch2 のレンジを拡大している)

8.4 データの整理, 解析

8.4.1 データの図示

Scope ツールから保存したデータは, *Analog Discovery 2 のセットアップ, ファンクションジェネレータとオシロスコープ* でも示したように,

- 先頭に全体の測定条件
- 空白行
- カラムのタイトル. 2 チャンネル分のデータの場合, [時間, チャネル 1, チャネル 2]
- 以下数値データ

である. Excel や Octave を使用し,

1. FG の出力と, コンデンサ電圧の時間変化
2. FG の出力と, 抵抗電圧の時間変化 (FG の出力が 1. とほぼ等しいことを確認すること)
3. 上記データの前半は放電過程に相当する. この部分のみを取り出し, 抵抗電圧-コンデンサ電圧の関係を図示. これは相平面図に相当する.

のグラフを作成する. 前項での測定例 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz) は以下の通り.

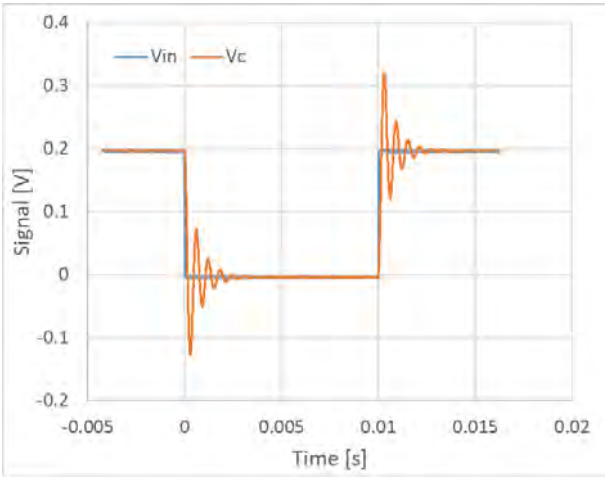


図 8.4　入力電圧とコンデンサ電圧の時間変化 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz)

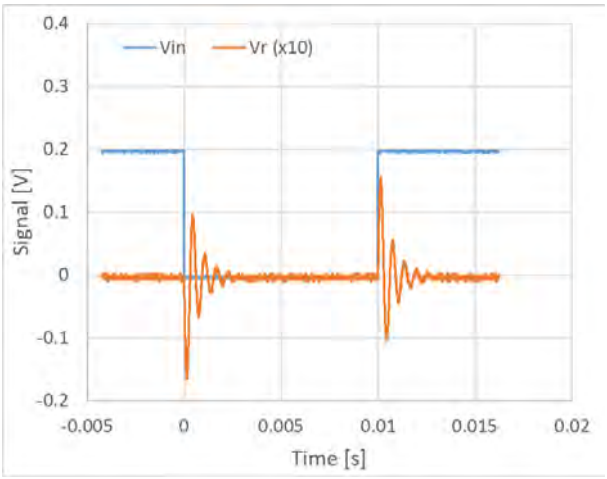


図 8.5　入力電圧と抵抗電圧の時間変化 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz, 抵抗電圧値は 10 倍に拡大している)

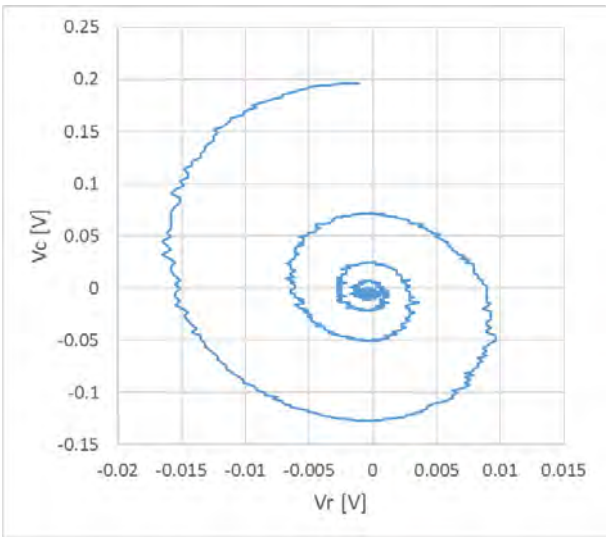


図 8.6　抵抗電圧とコンデンサ電圧の相平面図 (R=10 Ω, L=10mH, C=1 μ F, f=50Hz, 放電過程のみ)

8.5 パラメータを変えた測定

余力があれば, 上記の測定を, 抵抗 R の値を変え実施せよ. データを保存する際のファイル名にも注意が必要である. 今回の実験では,

- 抵抗値
- 測定箇所

をパラメータとしてデータを取得している. 従ってこれらの情報がファイル名を見るだけで理解できるような命名ルールを適用することが望ましい. また場合によっては,

- 実験実施日
- データ取得の通し番号 (同じような条件で複数回データを取得するかもしれない)

といった情報も必要になるため, 実験ノートやデータ保存を行ったフォルダ内に, これらの情報を残しておくことが重要である.

なお, 上記手順では, 「測定」と「解析」の項を分けて記述しているが, 実際にはデータ取得とそのデータに対する解析はその都度実施し, 結果を確認しながら実験を進めるべきである.

8.6 他手法との比較

余力があれば, 得られた結果を理論, LTSpice, 数値解析等の結果と比較せよ.

第 9 章

LCR 直列共振回路の周波数特性

LCR 直列共振回路の周波数特性を測定し, 他手法による解析結果と比較する.

9.1 回路及びパラメータ

測定に用いる回路は *LCR 直列共振回路の過渡応答測定* の図 7.1 と同じである.

L, C, R は以下の値を用いる.

- L = 10 [mH]
- C = 1 [μ F]
- R = 10 [Ω]

この時, テキスト 5.10 で与えられる伝達関数の概形をあらかじめ知っておくほうがよいであろう. Excel, Octave(Matlab) では複素数を扱うことができるため. $G(\mathrm{j}\omega)$ の絶対値, 偏角を計算できる. 以下に Octave での実施例を示す. ただし, あとで作成するボード線図等では, **ゲインを入出力電力 (=電圧の 2 乗) の比として定義し, デシベルを単位とする** 為,

$$20 \log |G(\mathrm{j}\omega)|$$

の値を求めることとなる.

注釈: excel での実施例は http://www.chem.mtu.edu/~tbco/cm416/freqexcel.html が参考になる.

リスト 9.1 LCR 直列回路の伝達関数を計算, プロットする octave(matlab) スクリプト

```
% LCR 回路の伝達関数 G(jw) を計算しプロットする

% Octave ではファイルが function で始まる場合,
% スクリプトではなく関数ファイルとして扱われる.
% そのことを避けるため慣習的に 1; を冒頭に記載する
1;
```

(次のページに続く)

(前のページからの続き)

```
% 素子の定数．gain 関数から参照できるよう global を付している．
global R;
R = 10;
global C;
C = 1e-6;
global L;
L = 10e-3;

% 回路の特性パラメータを計算する
w0 = 1 / sqrt(L*C);  % omega_0 (I の) 共振周波数
a = R / (2*L);   % alpha, この場合 a < w0 / sqrt(2)
w0v = w0 * sqrt(1 - 2 * (a / w0)^2);
% パラメータをコンソールに表示
disp("w0:"), disp(w0);
disp("a:"), disp(a);
disp("w0v:"), disp(w0v);

% 伝達関数．w は角周波数 (虚数単位 i は関数内部で付与)．w は配列でもよい．
function ret = gain(w)
   global R;
   global C;
   global L;
   % i は虚数単位．このように複素数を表現できる．
   ret = 1 ./ ((1 - w.^2 * L * C) + w * C * R * i);
endfunction

% 必要ならば，ここで gain 関数をテストするとよい．
% gain(w0)
% gain(w0v)
% gain([0.1*w0, 0.9*w0, w0v, w0, 1.1*w0, 10*w0])

% w0 を中心に 10^-1 から 10^1 までを対数スケールで計算する
scales = logspace(-1, 1, 1000);
ws = w0 * scales;
gs = gain(ws);
% gs の大きさ (ゲイン) と偏角 (位相差) を求める．
pgain_dbs = 20 * log10(abs(gs)); % 電力デシベル単位に変換
args = arg(gs) * 180 / pi; % ラジアンから度へ

% プロットする，subplot 関数により上段にゲイン，下段に位相差を表示
subplot(2, 1, 1); % 図を 2 行 1 列に配置，そのうちの 1 番目の図に描画
semilogx(scales, pgain_dbs); % x 軸が対数軸のプロット
% xlabel("Frequency [x w0]"); % 下段と共通なので表示しない
ylabel("Power Gain [dB]");
grid();
subplot(2, 1, 2); % 以後 2 番目の図 (下段) に描画
semilogx(scales, args);
xlabel("Frequency [x w0]");
ylabel("Phase [degree]");
ylim([-180, 0]); % y 軸の下限，上限を指定
```

(次のページに続く)

(前のページからの続き)

```
set(gca, 'Ytick', -180:45:0); % 軸目盛の位置を [-180, -135, -90, -45, 0] に
grid();
```

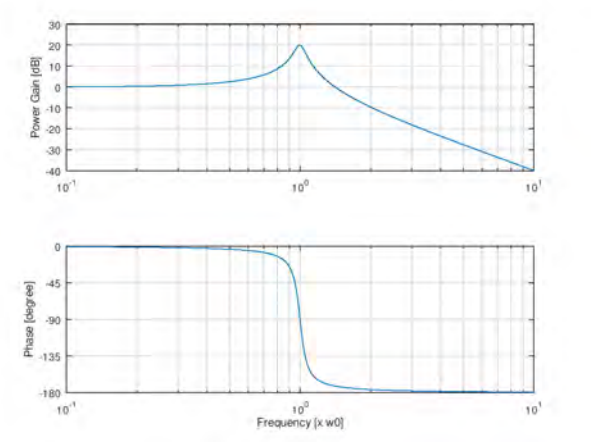


図 9.1　LCR 直列回路の伝達関数 (R=10 Ω, L=10mH, C=1 μ F)

上記の L, C, R 条件では

- $\omega_0 = 10000[\mathrm{rad/s}] \quad (\simeq 1591[\mathrm{Hz}])$
- $\alpha = 500[1/\mathrm{s}]$
- $\omega_{0v} = \omega_0\sqrt{1-2(\alpha/\omega_0)^2} = 9975[\mathrm{rad/s}] \quad (\simeq 1588[\mathrm{Hz}])$

であり, テキストの式 2.13 及び, $\alpha/\omega_0 < 1/\sqrt{2}$ を満たす. また ω_0 と ω_{0v} にほとんど差がない.

9.2 Bode Analyzer を用いた周波数特性の測定

9.2.1 コンデンサ出力特性の測定

ブレッドボード上に *LCR 直列共振回路の過渡応答測定* の 図 7.1 の回路を作成し, myDAQ と接続する. この時,

- AO0 の出力を AI0 で測定する
- AI1 でコンデンサに加わる電圧を測定する

ことが重要である. FC, OSC による動作確認 (後述) も併せて行うとよい.

NI ELVISmx Instrument Launcher より, **Bode Analyzer** のアイコンをクリックしソフトウェアを起動する.

Bode Analyzer は, 以下の手順でボード線図を作成する.

1. AO0 からある周波数の正弦波を発振
2. AI0(入力信号), AI1(出力信号) を測定し, 振幅の倍率, 位相差を算出
3. 結果をボード線図としてプロット

4. 1-3 の手順を周波数を変更しながら実施

Bode Analyzer の設定は以下の通りとする. 先に算出した ω_0 に対応する周波数が (対数軸で) 中央となるよう, 計測する周波数の範囲を決定する.

- [Measurement Setting]
  - [Start Frequency] 50 Hz
  - [Stop Frequency] 20 kHz
  - [Steps] 30 per decade
  - [Peak Amplitude] 0.1

これらの値を設定したのち, [Run] ボタンをクリックすると自動的に計測, プロットが行われる. 測定例を示す. オシロスコープと同様 [Log] ボタンをクリックすると数値データを保存できる.

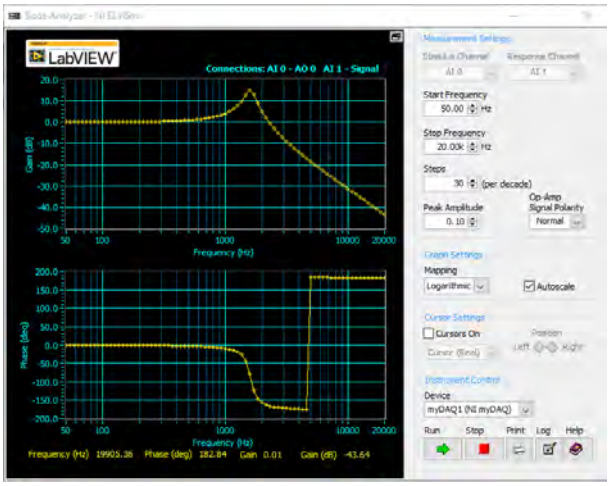


図 9.2　Bode Analyzer による測定結果

ω_0 付近で高いゲインが得られ, 位相が 180 度反転することが確認できる. ここで, 高周波領域において位相が正の方向に折り返されることがある. これは保存したデータファイルを読み込んだ際に,

$$\phi_{\mathrm{mod}} = \begin{cases} \phi, & (\phi <= 0) \\ \phi - 360, & (\phi > 0) \end{cases}$$

と変換すればよい.

9.2.2 抵抗, インダクタ出力特性の測定

コンデンサ電圧の他, 抵抗, インダクタ電圧についても周波数特性を測定せよ. 余力があれば, それぞれの伝達関数を求め, 結果を比較せよ.

9.3 FG, OSC を用いた伝達特性の測定

Bode Analyzer が自動的に行っていた計測を, 一部手動で実施してみよう.

ブレッドボード上の回路及び myDAQ の接続はそのままで, Function Generator (FG) 及び Oscilloscope (OSC) ソフトウェアを起動する.

FG の設定は,

- [Waveform Settings]
  - [波形] 正弦波
  - [Frequency] 1.2kHz (共振周波数からある程度離れた値をとる方がよい)
  - [Amplitude] 0.1 Vpp
  - [DC Offset] 0.0 V

OSC については

- [Trigger]
  - [Type] Edge
  - [Source] Chan 0 Source

と, 入力信号によりトリガがかかるように設定する.

FG, OSC 双方の [Run] ボタンをクリックし, 入出力信号が正弦波として測定出来ていることを確認する.

9.3.1 カーソルツールによるデータの取得

OSC のカーソルツールを使って波形データの値を直接知ることができる. OSC の [Stop] ボタンをクリックし, 連続測定を一時中断する. 画面左下の [Cursor on] にチェックを入れると, 画面左端に「C1」「C2」とラベルの付いた黄色の縦線 (カーソル) が表示される. この線 (ラベルではない) をクリックし, 左右にドラッグすると, カーソルが対応する波形データの電圧値, 並びに二つのカーソル間の時間差が波形表示画面下に表示される.

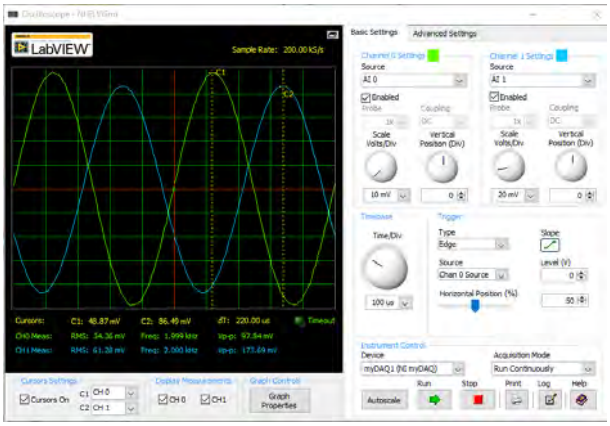


図 9.3　カーソルを用いた波形データの読み取り (f=2kHz の例)

Ch0(FG 入力電圧), Ch1(コンデンサ出力電圧) のピーク位置にそれぞれカーソルを合わせ, それぞれの電圧値, 時間差, 周波数を記録する.

- ピーク値の比よりゲインが得られる. Bode Analyzer と同じく電力デシベル単位とするため, $20 \log V_{FG}/V_{C}$ を計算する. なお, 信号が正弦波であるという仮定の下, RMS(信号の 2 乗平均値), Vp-p(peak-to-peak, 最大-最小値の幅) を利用してもよい.
- FG の入力周波数とピーク位置の時間差より, 信号の位相差が得られる.
- 共振周波数付近では入力電流が飽和し, 入力波形が正しく得られなくなる場合がある. この場合 FG 入力電圧の振幅を小さくする.

いくつかの周波数についてこれらの値を取得, プロットすると, ボード線図 (の一部) が再現できる.

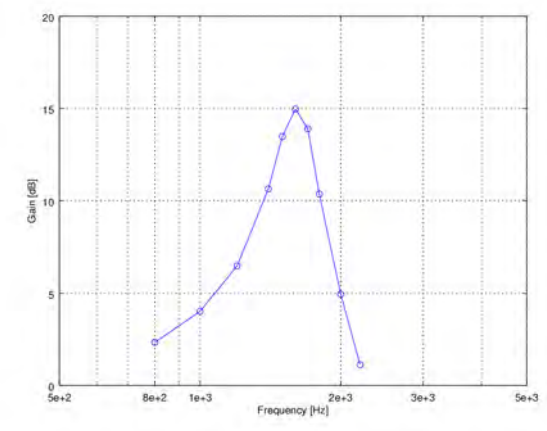


図 9.4　OSC より取得, 算出した周波数特性 (ゲイン)

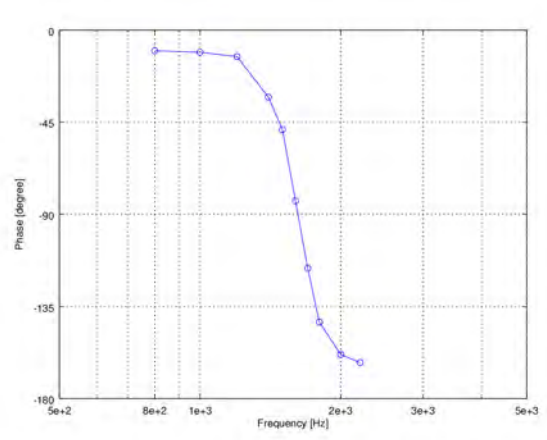


図 9.5　OSC より取得, 算出した周波数特性 (位相差)

第 10 章

LCR 直列共振回路の周波数特性 (AD2 利用)

LCR 直列共振回路の周波数特性を測定し, 他手法による解析結果と比較する.

10.1 回路及びパラメータ

測定に用いる回路は *LCR 直列共振回路の過渡応答測定 (AD2 利用)* の図 8.1 と同じである.

L, C, R は以下の値を用いる.

- L = 10 [mH]
- C = 1 [μ F]
- R = 10 [Ω]

この時, テキスト式 5.10 で与えられる伝達関数の概形をあらかじめ知っておくほうがよいであろう. Excel, Octave(Matlab) では複素数を扱うことができるため. $G(\mathrm{j}\omega)$ の絶対値, 偏角を計算できる. 以下に Octave での実施例を示す. ただし, あとで作成するボード線図等では, **ゲインを入出力電力 (=電圧の 2 乗) の比として定義し, デシベルを単位とする** 為,

$$20\log|G(\mathrm{j}\omega)|$$

の値を求めることとなる.

注釈: excel での実施例は http://www.chem.mtu.edu/~tbco/cm416/freqexcel.html が参考になる.

リスト 10.1　LCR 直列回路の伝達関数を計算, プロットする octave(matlab) スクリプト

```
% LCR 回路の伝達関数 G(jw) を計算しプロットする

% Octave ではファイルが function で始まる場合,
% スクリプトではなく関数ファイルとして扱われる.
% そのことを避けるため慣習的に 1; を冒頭に記載する
1;
```

(次のページに続く)

(前のページからの続き)

```
% 素子の定数．gain 関数から参照できるよう global を付している．
global R;
R = 10;
global C;
C = 1e-6;
global L;
L = 10e-3;

% 回路の特性パラメータを計算する
w0 = 1 / sqrt(L*C);  % omega_0 (I の) 共振周波数
a = R / (2*L);   % alpha, この場合 a < w0 / sqrt(2)
w0v = w0 * sqrt(1 - 2 * (a / w0)^2);
% パラメータをコンソールに表示
disp("w0:"), disp(w0);
disp("a:"), disp(a);
disp("w0v:"), disp(w0v);

% 伝達関数．w は角周波数 (虚数単位 i は関数内部で付与)．w は配列でもよい．
function ret = gain(w)
   global R;
   global C;
   global L;
   % i は虚数単位．このように複素数を表現できる．
   ret = 1 ./ ((1 - w.^2 * L * C) + w * C * R * i);
endfunction

% 必要ならば，ここで gain 関数をテストするとよい．
% gain(w0)
% gain(w0v)
% gain([0.1*w0, 0.9*w0, w0v, w0, 1.1*w0, 10*w0])

% w0 を中心に 10^-1 から 10^1 までを対数スケールで計算する
scales = logspace(-1, 1, 1000);
ws = w0 * scales;
gs = gain(ws);
% gs の大きさ (ゲイン) と偏角 (位相差) を求める．
pgain_dbs = 20 * log10(abs(gs)); % 電力デシベル単位に変換
args = arg(gs) * 180 / pi; % ラジアンから度へ

% プロットする，subplot 関数により上段にゲイン，下段に位相差を表示
subplot(2, 1, 1); % 図を 2 行 1 列に配置，そのうちの 1 番目の図に描画
semilogx(scales, pgain_dbs); % x 軸が対数軸のプロット
% xlabel("Frequency [x w0]"); % 下段と共通なので表示しない
ylabel("Power Gain [dB]");
grid();
subplot(2, 1, 2); % 以後 2 番目の図 (下段) に描画
semilogx(scales, args);
xlabel("Frequency [x w0]");
ylabel("Phase [degree]");
ylim([-180, 0]); % y 軸の下限，上限を指定
```

(次のページに続く)

(前のページからの続き)

```
set(gca, 'Ytick', -180:45:0); % 軸目盛の位置を [-180, -135, -90, -45, 0] に
grid();
```

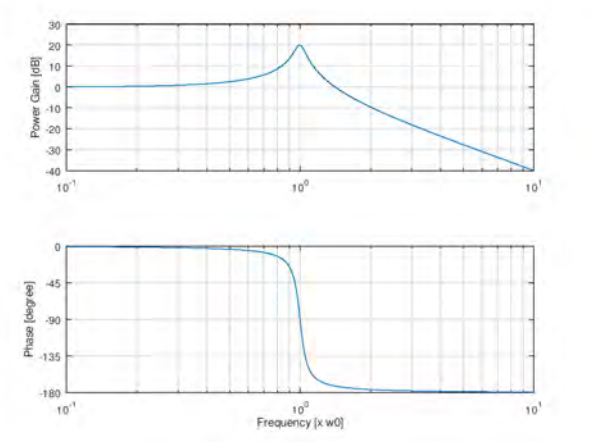


図 10.1　LCR 直列回路の伝達関数 (R=10 Ω, L=10mH, C=1 μ F)

上記の L, C, R 条件では

- $\omega_0 = 10000[\mathrm{rad/s}] \quad (\simeq 1591[\mathrm{Hz}])$
- $\alpha = 500[1/\mathrm{s}]$
- $\omega_{0v} = \omega_0\sqrt{1-2(\alpha/\omega_0)^2} = 9975[\mathrm{rad/s}] \quad (\simeq 1588[\mathrm{Hz}])$

であり, テキストの式 2.13 及び, $\alpha/\omega_0 < 1/\sqrt{2}$ を満たす. また ω_0 と ω_{0v} にほとんど差がない.

10.2 Network Analyzer を用いた周波数特性の測定

10.2.1 コンデンサ出力特性の測定

ブレッドボード上に *LCR 直列共振回路の過渡応答測定 (AD2 利用)* の 図 8.1 の回路を作成し, AD2 と接続する. この時,

- (W1, GND) の出力を (Ch1+, Ch1-) で測定する
- (Ch2+, Ch2-) でコンデンサに加わる電圧を測定する

ことが重要である.

WaveForms より, **Bode Analyzer** のツールを起動する.

Bode Analyzer は, 以下の手順でボード線図を作成する.

1. (W1, GND) からある周波数の正弦波を発振
2. Ch1(入力信号), Ch2(出力信号) を測定し, 振幅の倍率, 位相差を算出
3. 結果をボード線図としてプロット

4. 1-3 の手順を周波数を変更しながら実施

Network Analyzer の設定は以下の通りとする. 先に算出した ω_0 に対応する周波数が (対数軸で) 中央となるよう, 計測する周波数の範囲を決定する.

- [測定方式 (画面上部メニュー)]
  - [Scale] Logarithmic
  - [Start] 50 Hz
  - [Stop] 20 kHz
  - [Samples] 100
- [WaveGen(画面右)]
  - [Offset] 0V
  - [Amplitude] 1V
- [Magnitude(画面右)]
  - [Relative to Channel1] を有効に
  - [Units] dB
  - [Top] 30 dB
  - [Bottom] -70 dB
- [Phase(画面右)]
  - [Offset] 0 $^\circ$
  - [Range] 360 $^\circ$
- [Channel 2(画面右)]
  - Channel2 にチェックを入れ表示を有効にする
  - [Offset] 0 V
  - [Gain] Auto

これらの値を設定したのち, [Run] ボタンをクリックすると自動的に計測, プロットが行われる. 測定例を示す. Scope ツールと同様に, *File* → *Export* メニューより数値データを保存できる.

ω_0 付近で高いゲインが得られ, 位相が 180 度反転することが確認できる. ここで, 高周波領域において位相が正の方向に折り返されることがある. これは保存したデータファイルを読み込んだ際に,

$$\phi_{\mathrm{mod}} = \begin{cases} \phi, & (\phi <= 0) \\ \phi - 360, & (\phi > 0) \end{cases}$$

と変換すればよい.

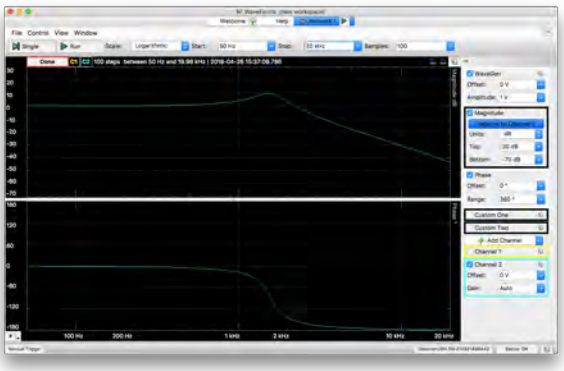


図 10.2　Network Analyzer による周波数特性測定結果

10.2.2 抵抗, インダクタ出力特性の測定

コンデンサ電圧の他, 抵抗, インダクタ電圧についても周波数特性を測定せよ. 余力があれば, それぞれの伝達関数を求め, 結果を比較せよ.

第 11 章

RC フィルタの周波数特性

RC フィルタを作成し, その入出力信号と周波数特性を測定する.

周波数特性の測定 (ボード線図の作成) と図示については, *LCR 直列共振回路の過渡応答測定* を参照のこと.

11.1 1 段フィルタの特性測定

遮断角周波数 ω_c が等しくなる次の RC の組み合わせに対し, RC フィルタの周波数特性を測定し, 比較する.

- $\omega_c = 10^4 [\mathrm{rad/s}]$
  - R = 10k Ω, C = 0.01 μ F
  - R = 1k Ω, C = 0.1 μ F
  - R = 100 Ω, C = 1 μ F
- $\omega_c = 10^3 [\mathrm{rad/s}]$
  - R = 10k Ω, C = 0.1 μ F
  - R = 1k Ω, C = 1 μ F

11.1.1 実施例

実施例を 図 11.1 - 図 11.3 に示す.

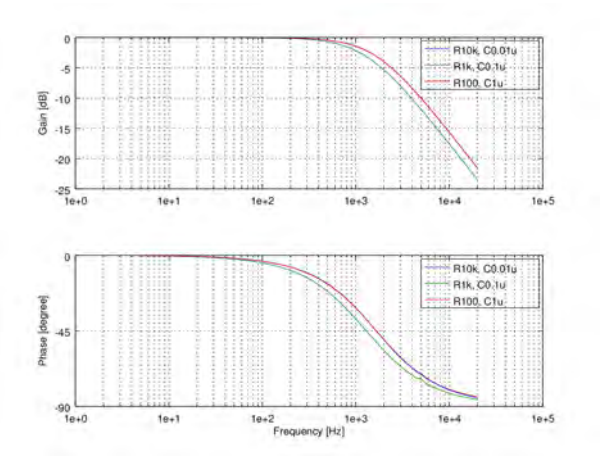


図 11.1　RC フィルタの周波数特性 (wc=10^4[rad/s])

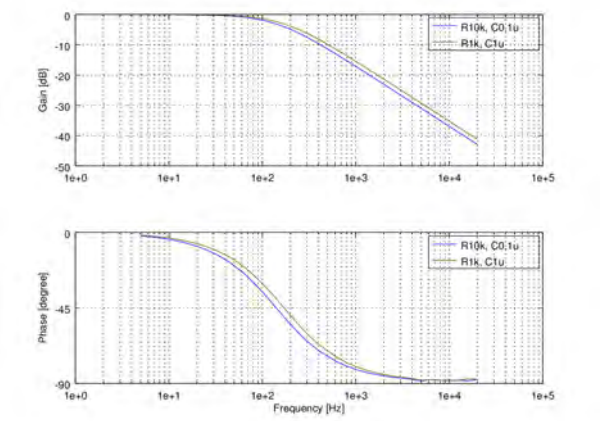


図 11.2　RC フィルタの周波数特性 (wc=10^3[rad/s])

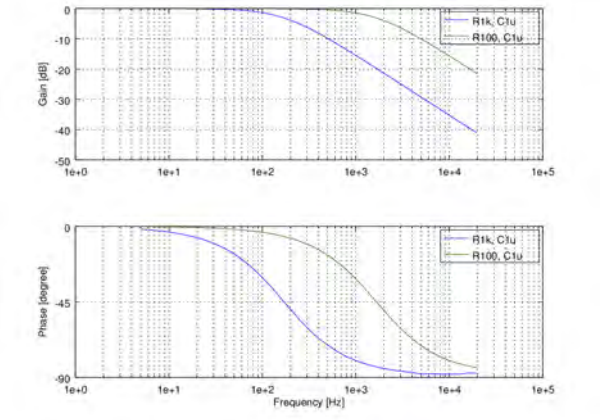


図 11.3　RC フィルタの周波数特性 (C=1 μ F, R=100 or 1k Ω)

11.2 2 段フィルタの特性測定

上記のフィルタ条件から 2 つ (A, B とする) を選び 2 段の LPF を作成し, その周波数特性を測定する. この時,

- 1 段目を条件 A, 2 段目を条件 B
- 1 段目を条件 B, 2 段目を条件 A

の組み合わせに対し, テキストの式 7.10 ではなく 7.11 の近似式が成立する条件を考える.

11.2.1 実施例

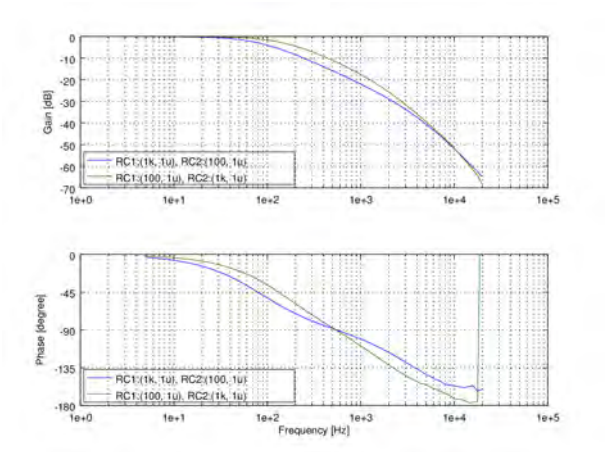


図 11.4　2 段 RC フィルタの周波数特性 (C=1 μ F, R=100 or 1k Ω)

11.3 Octave でのモデル計算

実験を行った 2 段フィルタの条件 A, B に対し,

- 条件 A のみ 1 段, 式 7.1 を適用
- 条件 B のみ 1 段, 式 7.1 を適用
- 1 段目を条件 A, 2 段目を B の 2 段フィルタ, 式 7.10 を適用
- 1 段目を条件 B, 2 段目を A の 2 段フィルタ, 式 7.10 を適用
- 条件 A, B の 2 段フィルタ, 式 7.11 を適用 (この場合, A,B の交換に対して不変)

それぞれの伝達関数を求め, ボード線図を作図する.

11.3.1 実施例

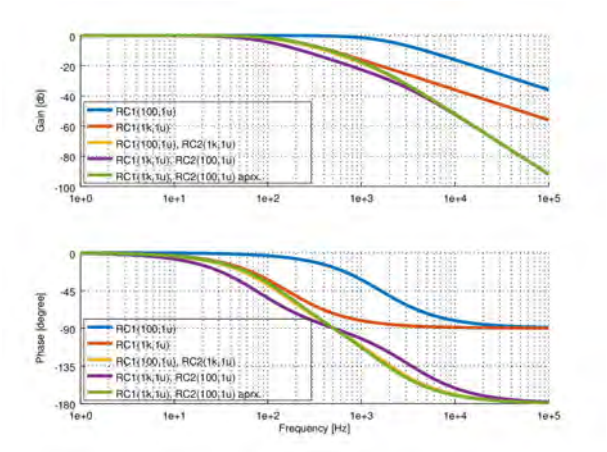


図 11.5　2 段 RC フィルタの周波数特性, Octave によるモデル計算

リスト 11.1　図 11.5 を出力する octave スクリプト

```
% 1 段, 2 段 RC フィルタのボード線図を作成する.

% Octave ではファイルが function で始まる場合,
% スクリプトではなく関数ファイルとして扱われる.
% そのことを避けるため慣習的に 1; を冒頭に記載する
1;

%1 段 RC フィルタの特性関数, w は配列でもよい
function ret = g1(R, C, w)
  ret = 1 ./ (1 + i * C * R * w);
endfunction

%2 段 RC フィルタの特性関数 (jwR1C2 を省略しない), w は配列でもよい
function ret = g2(R1, C1, R2, C2, w)
  ret = 1 ./ ((1 + i * C1 * R1 * w) .* (1 + i * C2 * R2 * w) + i * R1 * C2 * w);
endfunction

%2 段 RC フィルタの特性関数 (jwR1C2 を省略), w は配列でもよい
function ret = g2approx(R1, C1, R2, C2, w)
  ret = 1 ./ ((1 + i * C1 * R1 * w) .* (1 + i * C2 * R2 * w));
endfunction

% 伝達関数の値 (複素) を, ゲイン (電力 dB) と位相差 (度) に変換する
function [gains, phases] = transG(gs)
  gains = 20 * log10(abs(gs));
  phases = arg(gs) * 180 / pi;
endfunction

freqs = logspace(0, 5, 100); % 1 Hz から 100 kHz の間
ws = 2 * pi * freqs; % 周波数から角周波数に変換

% それぞれの伝達関数を計算し gain(), phase() の 2 次元配列に収める
% また凡例として記載する文字列をセル配列という特殊な配列で保持する
```

(次のページに続く)

(前のページからの続き)

```
label = {};
% case1 R=100, C=1u, wc = 10^4
label{1} = ';RC1(100,1u);';
[gain(:, 1), phase(:, 1)] = transG(g1(100, 1e-6, ws));
% case2 R=1k, C=1u, wc = 10^4
label{2} = ';RC1(1k,1u);';
[gain(:, 2), phase(:, 2)] = transG(g1(1000, 1e-6, ws));
% case3 RC1(100,1u), RC2(1k, 1u)
label{3} = ';RC1(100,1u), RC2(1k,1u);';
[gain(:, 3), phase(:, 3)] = transG(g2(100, 1e-6, 1000, 1e-6, ws));
% case4 RC1(1k,1u), RC2(100, 1u)
label{4} = ';RC1(1k,1u), RC2(100,1u);';
[gain(:, 4), phase(:, 4)] = transG(g2(1000, 1e-6, 100, 1e-6, ws));
% case5 RC1(1k,1u), RC2(100, 1u), approx.
label{5} = ';RC1(1k,1u), RC2(100,1u) aprx.;';
[gain(:, 5), phase(:, 5)] = transG(g2approx(1000, 1e-6, 100, 1e-6, ws));

%%% プロットする色をカスタマイズする. RGB 値の 2 次元配列
colors = [
[hex2dec('00'), hex2dec('72'), hex2dec('bd')] ./ 255;
[hex2dec('d9'), hex2dec('53'), hex2dec('19')] ./ 255;
[hex2dec('ed'), hex2dec('b1'), hex2dec('20')] ./ 255;
[hex2dec('7e'), hex2dec('2f'), hex2dec('8e')] ./ 255;
[hex2dec('77'), hex2dec('ac'), hex2dec('30')] ./ 255;];

% プロットする
% 1 画面に縦 2x 横 1 のプロットを配置し、そのうちの 1 番目に対しプロットする
subplot(2, 1, 1);
hold on;
for i = 1:length(label)
  %'Color' や 'linewidth' というオプション引数により, 色や線の太さを調整している
  semilogx(freqs, gain(:,i), label{i}, 'Color', colors(i,:), 'linewidth', 3);
endfor
ylabel('Gain [db]');
legend('location', 'southwest');
grid();
% 2 番目に対するプロット
subplot(2, 1, 2);
hold on;
for i = 1:length(label)
  semilogx(freqs, phase(:,i), label{i}, 'Color', colors(i,:), 'linewidth', 3);
endfor
ylim([-180, 0]);
set(gca, 'Ytick', -180:45:0);
xlabel('Frequency [Hz]');
ylabel('Phase [degree]');
legend('location', 'southwest');
grid();
```

第 12 章

能動回路の実験

12.1 トランジスタの動作特性

NPN トランジスタ (2SC1815) の動作特性を実際に測定する.

図 12.1 の回路を構成し, myDAQ と接続する. 特に, トランジスタのベース, エミッタ, コレクタピンの配置に注意すること. 平たい面を上にし, ピン側から見たとき, 左より

- エミッタ
- コレクタ
- ベース

となる.

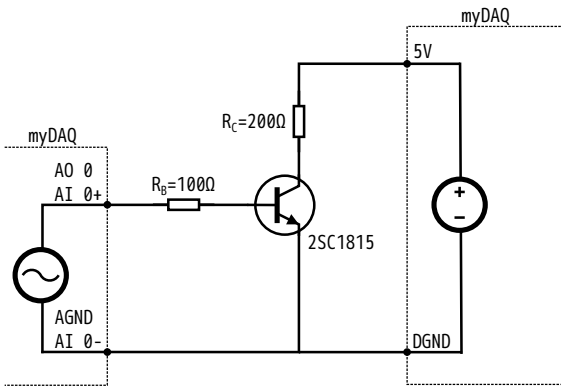


図 12.1　NPN トランジスタ特性測定の回路図

ファンクションジェネレータ (FG), オシロスコープソフトウェア (OSC) を起動する.

FG の出力を次のように設定する.

- 波形: 正弦波
- 周波数 (Frequency): 1Hz (非常に遅い変化とし, 静的な特性を調べる)
- 振幅 (Amplitude): 1 Vpp

- オフセット (DC Offset): 0.5V (この設定により, AI0 の入力範囲は 0-1V となる)

一方 OSC は, 入力信号によりトリガが掛かるよう適切にトリガレベルを設定する (例: 0.5V).

入力電圧 (V_{in}, myDAQ の AI0 端子で測定) に対し, ベース抵抗 R_B, コレクタ抵抗 R_C に加わる電圧 (それぞれ V_{Rb}, V_{Rc} とする) を, myDAQ の AI1 端子で測定する. FG の設定画面 (図 12.2), 並びに V_{Rb}, V_{Rc} の測定結果 (図 12.3, 図 12.4) を示す. また, これらをテキストデータとして保存する.

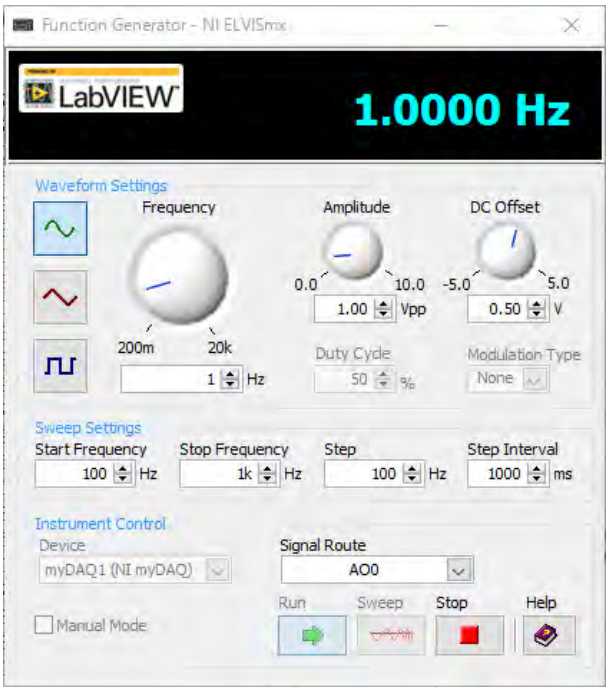


図 12.2　ファンクションジェネレータの設定

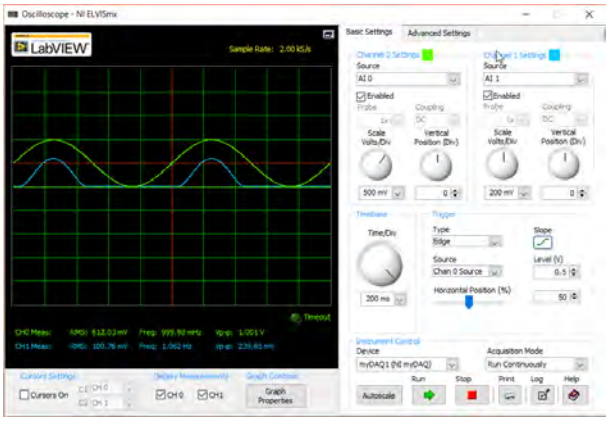


図 12.3　入力電圧とベース抵抗電圧の関係

測定結果より, 以下の値が求められる.

- ベース-エミッタ間電圧: $V_{\mathrm{BE}} = V_{\mathrm{in}} - V_{\mathrm{Rb}}$
- ベース電流: $I_{\mathrm{B}} = V_{\mathrm{Rb}}/R_{\mathrm{B}}$
- コレクタ電流: $I_{\mathrm{C}} = V_{\mathrm{Rc}}/R_{\mathrm{C}}$

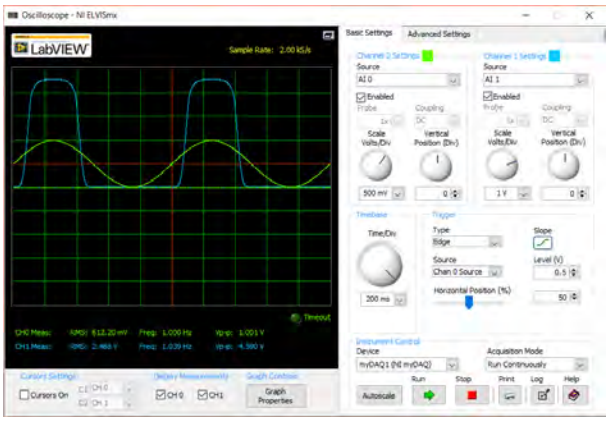


図 12.4　入力電圧とコレクタ抵抗電圧の関係

I_{B} の V_{BE} 依存性を入力特性と呼ぶ (図 12.5). コレクタを無視すると, ベース-エミッタ接合は順方向ダイオードとみなすことができる.

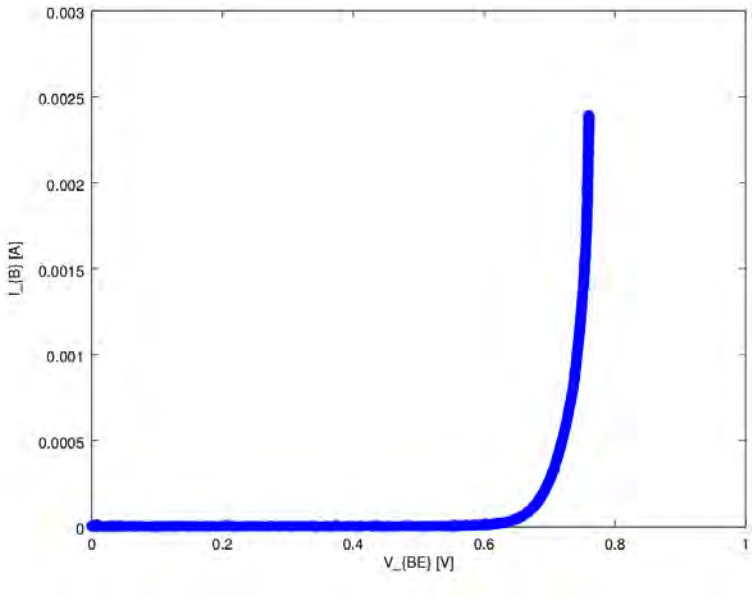


図 12.5　入力特性 (I_{B} は線形軸)

縦軸 (I_{B}) を対数軸にすると, $V_{\mathrm{BE}} > 0.6\,[\mathrm{V}]$ の領域において直線, すなわち,

$$I_{\mathrm{B}} \propto \exp V_{\mathrm{BE}}$$

であることが分かる (図 12.6).

一方, I_{C} の I_{B} 依存性を電流伝達特性という (図 12.7). 両者が比例する領域 (活性領域), そして飽和する領域が観測される.

注釈: このプロットは, V_{in} を共通の変数とし, $V_{\mathrm{in}} - V_{\mathrm{Rb}}$ と $V_{\mathrm{in}} - V_{\mathrm{Rc}}$ の測定結果を利用している. 従って 2 つの測定データの各列において, V_{in} が一致することが前提となる.

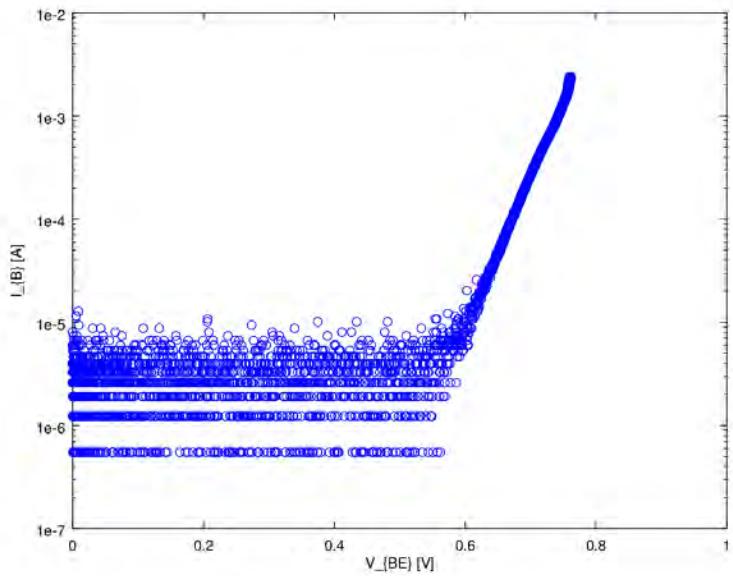


図 12.6　入力特性 (I_{B} は対数軸)

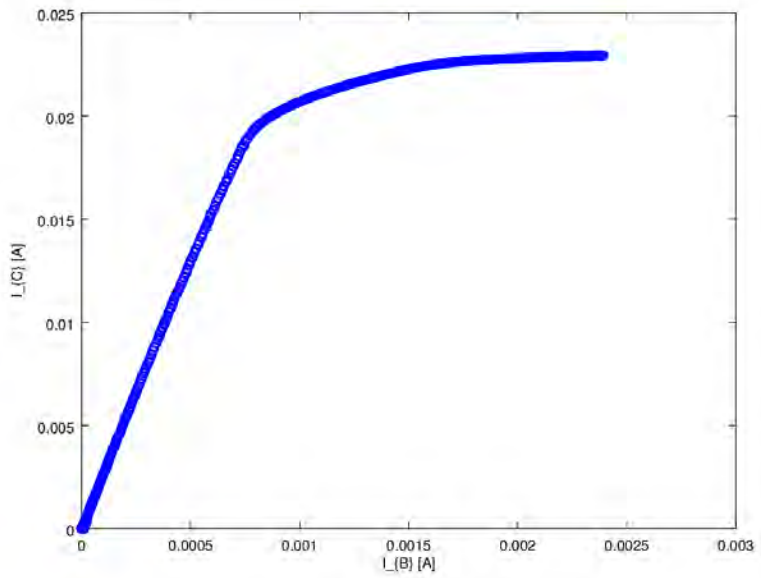


図 12.7　電流伝達特性

12.2 移相発振回路の実験

テキストの図 12.2 の回路を作成する. ただし, 素子は実験キットに含まれる以下のものを利用する.

- C: 0.1 μ F セラミックコンデンサ (104 と刻印のあるもの)
- R: 10k Ω 固定抵抗 (茶黒橙)
- $\mathrm{R_C}$: 10k Ω 固定抵抗 (茶黒橙)

この回路に対し,

- ch0: ベース電圧-GND (Vi に相当)
- ch1: コレクタ電圧-GND (Vo に相当)

を測定する. この時, Vi が極めて微弱な為, 以下のような設定を行うとよい.

- [Channel 0 Settings]:
  - [Scale]: 100mV
  - [Vertical Position]: -5
- [Channel 1 Settings]
  - [Scale]: 500mV
- [Timebase]: 2ms
- [Trigger]:
  - [Type]: Edge
  - [Source]: **Ch1** (測定しやすい Vo を基準にトリガをかける)
  - [Level]: 0.5V

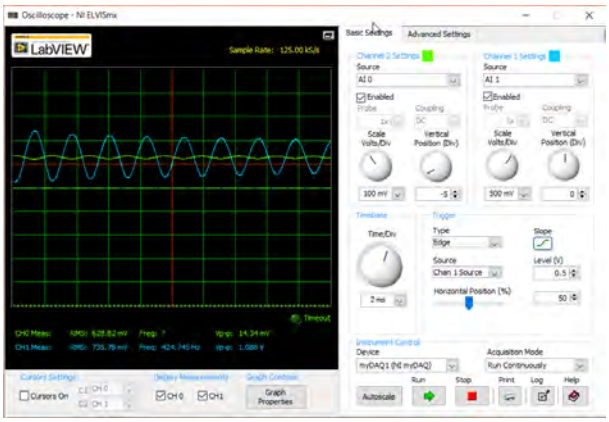


図 12.8　移相発振器の信号測定結果

入力信号が極めて微弱なため, 信号の特徴がとらえにくいが, 波形描画画面の下部の数値を読むことで

- 二乗平均平方根 (RMS, ただの平均ではないことに注意)
- Vp-p (最大・最小値の距離, 振幅の 2 倍に相当)
- 周波数 (Freq)

を知ることができる (図 12.8).

また, 測定データに対し後処理を施すことで, 次のことが確認できる.

- Vi, Vo で位相が反転していること

  例えば, Vi, Vo のそれぞれの平均値, 標準偏差を求め, 波形をシフト, 拡大する.

- 電圧増幅率が十分大きいこと (図 12.9)

  Vi, Vo をプロットし, 近似直線を当てはめることで, おおよその傾向がわかる.

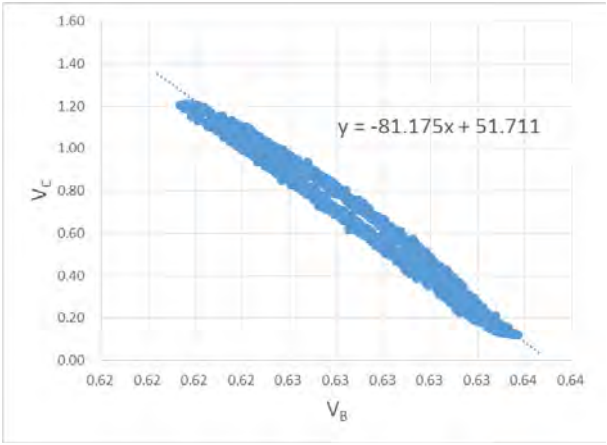


図 12.9　Vi, Vo の対応と近似直線の当てはめ

注釈: 実は, Vi に myDAQ を接続する・しないにより, Vo の波形が大きく異なる. 本来ならば, Vi を接続しない方が正確な発振波形である.

第 13 章

能動回路の実験 (AD2 利用)

13.1 トランジスタの動作特性

NPN トランジスタ (2SC1815) の動作特性を実際に測定する.

図 13.1 の回路を構成し, Analog Discovery 2 (AD2) と接続する. 特に, トランジスタのベース, エミッタ, コレクタピンの配置に注意すること. 平たい面を上にし, ピン側から見たとき, 左より

- エミッタ
- コレクタ
- ベース

となる.

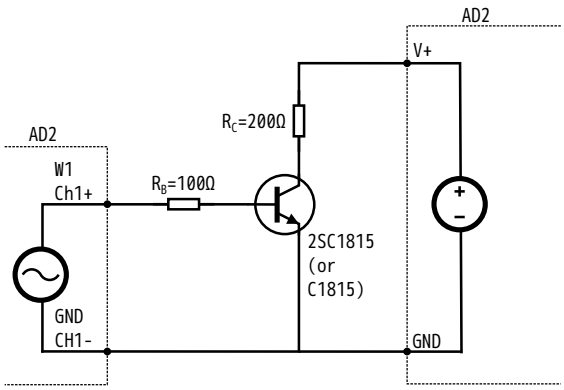


図 13.1　NPN トランジスタ特性測定の回路図 (AD2 利用時)

WaveForms を起動し, Wavegen, Scope, Supplies(定電圧源) ツールを起動する.

Wavegen Channel 1 の出力を次のように設定する (図 13.2).

- 波形 (Type): 正弦波 (Sine)
- 周波数 (Frequency): 1Hz (非常に遅い変化とし, 静的な特性を調べる)
- 振幅 (Amplitude): 0.5 V (1 Vpp である)

- オフセット (DC Offset): 0.5V (この設定により, Ch1 の出力範囲は 0-1V となる)

Supplies ツールは AD2 の V+, V-に定電圧を供給するツールである (図 13.3).

- [Positive Power Supply (V+)] の [Voltage] を 5V にセットする.

Scope は, 入力信号 (Ch1) によりトリガが掛かるよう適切にトリガレベルを設定する (例: 0.5V).

入力電圧 (V_{in}, AD2 の Ch1 端子で測定) に対し, ベース抵抗 $\mathrm{R_B}$, コレクタ抵抗 $\mathrm{R_C}$ に加わる電圧 (それぞれ V_{Rb}, V_{Rc} とする) を, Ch2 端子で測定する.

Wavegen, Supplies を On にする. この時, Scope で信号を取得する場合, 信号は 1Hz と遅いので, [Scan] ではなく [Single] ボタンを押し, 1 回のスイープデータを取得する. また [Single] による取得を実施することで, 正しくトリガレベルが原点となるように波形の位置が調整される.

V_{in} に対する V_{Rb}, V_{Rc} (図 13.4, 図 13.5) を測定し, これらのデータを csv ファイルとして保存する.

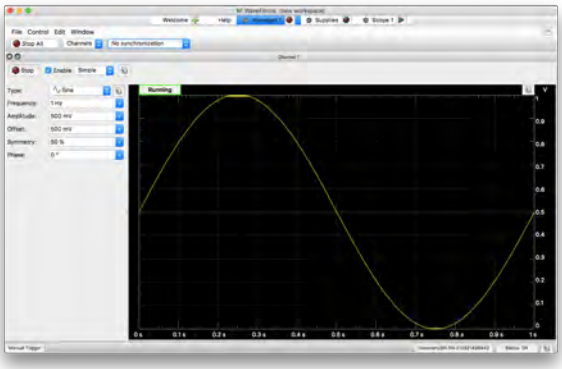


図 13.2　Wavegen ツールの設定

図 13.3　Supplies ツールの設定

測定結果より, 以下の値が求められる.

- ベース-エミッタ間電圧: $V_{\mathrm{BE}} = V_{\mathrm{in}} - V_{\mathrm{Rb}}$
- ベース電流: $I_{\mathrm{B}} = V_{\mathrm{Rb}}/R_{\mathrm{B}}$
- コレクタ電流: $I_{\mathrm{C}} = V_{\mathrm{Rc}}/R_{\mathrm{C}}$

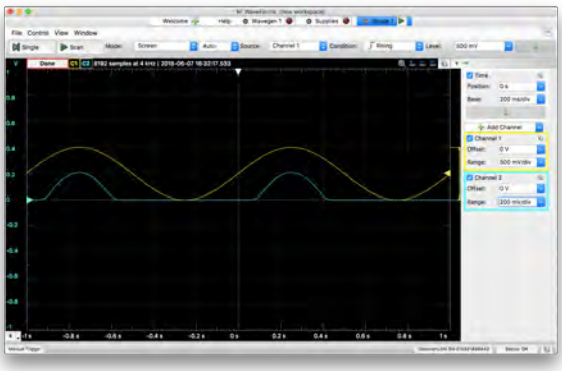


図 13.4　入力電圧とベース抵抗電圧の関係 (Scope ツールでの取得)

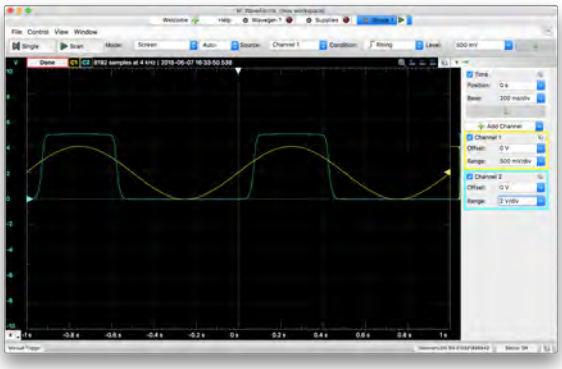


図 13.5　入力電圧とコレクタ抵抗電圧の関係 (Scope ツールでの取得)

I_B の V_BE 依存性を入力特性と呼ぶ (図 13.6). コレクタを無視すると, ベース-エミッタ接合は順方向ダイオードとみなすことができる.

縦軸 (I_B) を対数軸にすると, $V_\mathrm{BE} > 0.6\,[\mathrm{V}]$ の領域において直線, すなわち,

$$I_\mathrm{B} \propto \exp V_\mathrm{BE}$$

であることが分かる (図 13.7).

一方, I_C の I_B 依存性を電流伝達特性という (図 13.8). 両者が比例する領域 (活性領域), そして飽和する領域が観測される.

注釈: このプロットは, V_in を共通の変数とし, $V_\mathrm{in} - V_\mathrm{Rb}$ と $V_\mathrm{in} - V_\mathrm{Rc}$ の測定結果を利用している. 従って 2 つの測定データの各列において, V_in が一致することが前提となる.

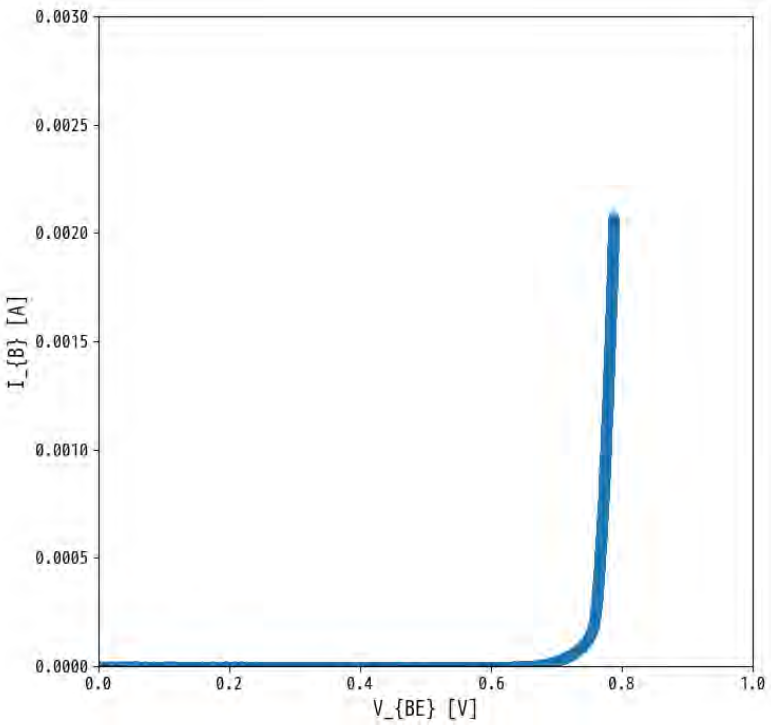


図 13.6　入力特性 (I_{B} は線形軸)

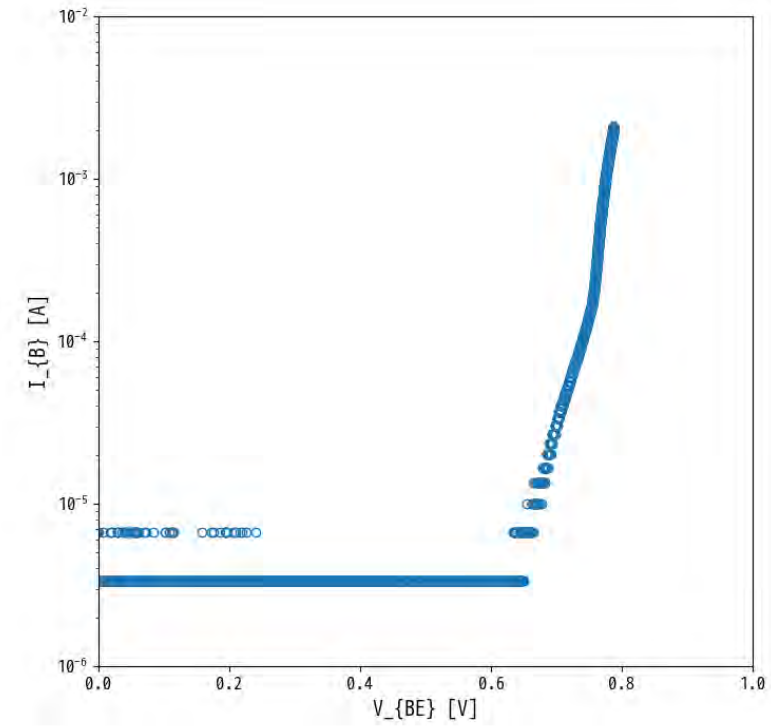


図 13.7　入力特性 (I_{B} は対数軸)

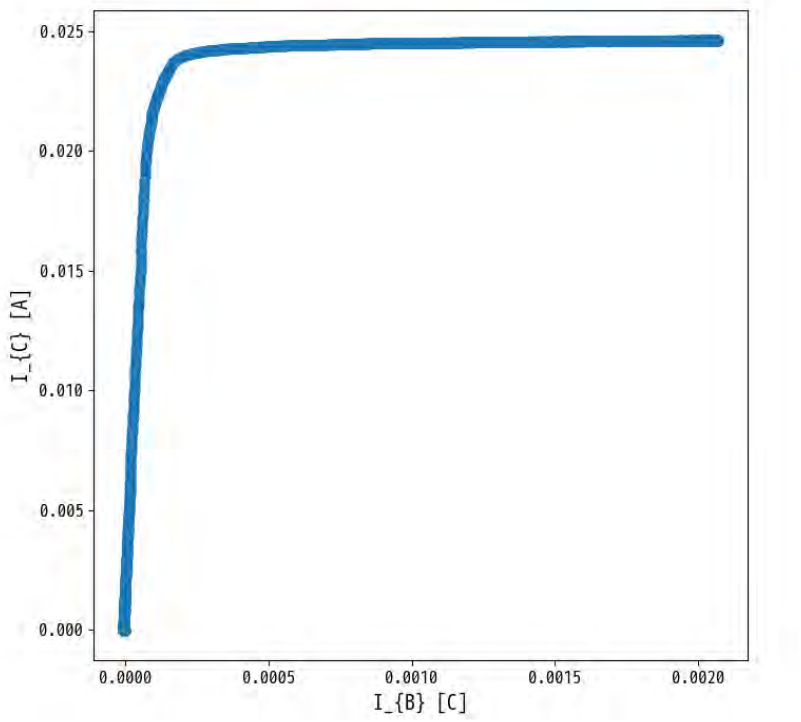


図 13.8　電流伝達特性

13.2 移相発振回路の実験

テキストの図 12.2 の回路を作成する. ただし, 素子は実験キットに含まれる以下のものを利用する.

- C: 0.1 μ F セラミックコンデンサ (104 と刻印のあるもの)
- R: 10k Ω 固定抵抗 (茶黒橙)
- R_C: 10k Ω 固定抵抗 (茶黒橙)

この回路に対し,

- Supplies V+ ツールにより V_{CC} 電圧を供給 (図 13.3 と同様)

さらに

- Scope Ch1 でベース電圧-GND (Vi に相当)
- Scope Ch2 でコレクタ電圧-GND (Vo に相当)

を測定する. この時, Vi が極めて微弱なこと, またトランジスタを駆動するため, ベース-エミッタ間に約 0.6V の電圧が常時加わっていることから, 以下のような設定を行うとよい.

- Scope ツール上部メニューでのトリガー設定:
  - [Source]: **Channel 2** (測定しやすい Vo をトリガのソースとする)

  - [Condition] Rising
  - [Level]: 500 mV
- [Time]:
  - [Base]: 1 ms/div
- [Channel 1]:
  - [Offset]: -600mV
  - [Scale]: 10 mV/div
- [Channel 2]
  - [Scale]: 500 mV/div

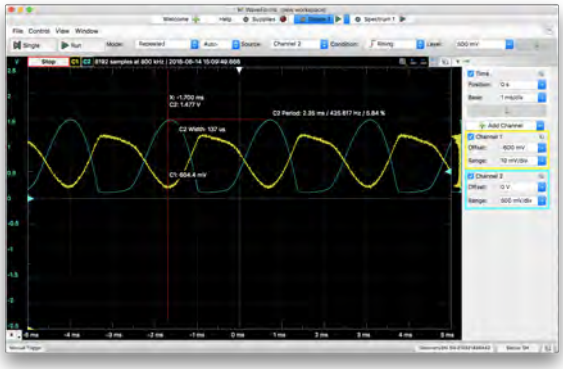


図 13.9　移相発振器の信号測定結果

図 13.9 に Scope ツールでの測定結果を示す. この画面右上にある [Pulse] アイコン (歯車記号の左隣) をクリックすると, 波形の周期を簡便に読み取ることができる. データを読み取ると, 発信周波数は約 430Hz 程度であることがわかる (なおテキストより得られる発信周波数は約 390Hz である).

Ch2 の入力を Spectrum ツールで観察すると, 周波数成分が表示される (図 13.10). 波形が正弦波ではなく歪んでいるため, 基本の周波数 (430Hz 付近) ではなく, この 2 倍, 3 倍, ... の高調波成分が多くみられる.

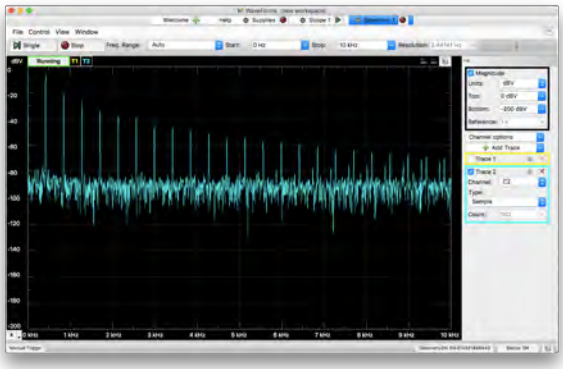


図 13.10　移相発振器のスペクトル

また, 測定データに対し後処理を施すことで, 次のことが確認できる.

- Vi, Vo で位相が反転していること

  例えば, Vi, Vo のそれぞれの平均値, 標準偏差を求め, 波形をシフト, 拡大する.

- 電圧増幅率が十分大きいこと (図 13.11)

  Vi, Vo をプロットし, 近似直線を当てはめることで, おおよその傾向がわかる.

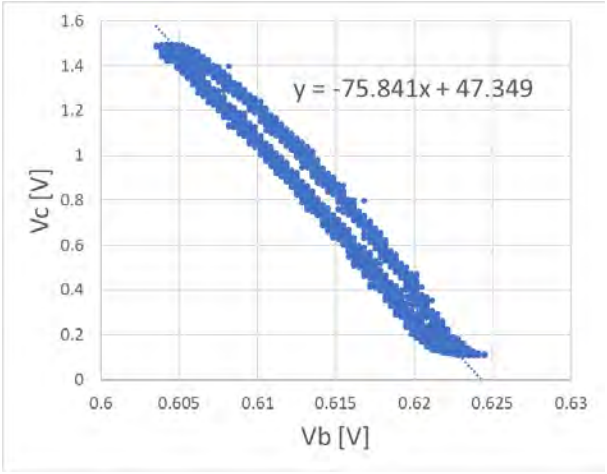


図 13.11　Vi, Vo の対応と近似直線の当てはめ

第 14 章

Octave 補足課題

Octave による *LCR* 直列共振回路の解析 の補足課題して取り組むとよい.

14.1 RC 回路のステップ応答

14.1.1 問題

一定電圧 V_0 の電圧源, 抵抗値 R の抵抗, 容量 C のコンデンサを接続する. $t = 0$ のときコンデンサにかかる電圧 V_C を 0 とする. この後の V_C の時間変化を求めよ.

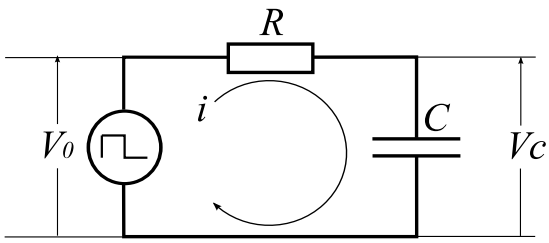


図 14.1　RC 回路

14.1.2 解析解

この問題は解析的に解くことができる. V_C に関する微分方程式は,

$$\frac{dV_C}{dt} = \frac{1}{RC}(V_0 - V_C)$$

これに, $V_C(0) = 0$ の初期値を適用すると,

$$V_C(t) = V_0 \left(1 - \exp(-\frac{t}{RC})\right)$$

である. 時刻 0 から $5RC$ までの区間を 50 等分し, 解析解をプロットするプログラムとその結果は次のようになる.

```
1;

# 適宜パラメータを変化させてグラフの変化を見るとよい．
V0 = 1;
R = 1;
C = 1;

t = linspace(0, 5*C*R, 51);   # 0 から 5RC まで 50 等分，51 点の等分をとる
                              # t はベクトル．

vc = V0 * (1 - exp(-t / (C*R)));    # 多くの数学関数がベクトルの逐次適用に対応しているため
                                    # このようにシンプルに書くことができる．

plot(t, vc, "-x;analytic;");  # 三番目のパラメータは，プロットのスタイルと凡例
xlabel("t", "fontsize", 18);
ylabel("vc", "fontsize", 18);
legend("location", "northwest");

# 必要であればプロット画像，csv データでの保存をこの後行うとよい．
```

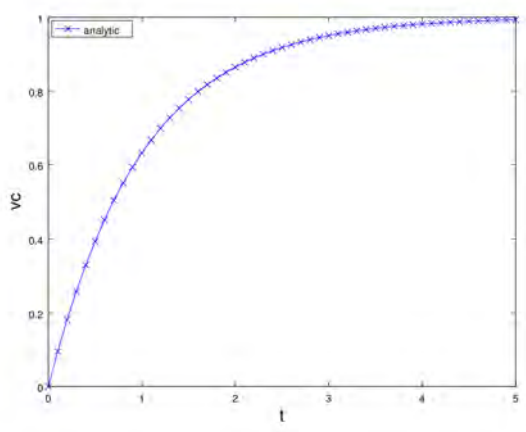


図 14.2　RC 回路のステップ応答 (解析解)

14.1.3 微分方程式ソルバによる数値解

Octave には **lsode** という汎用の常微分方程式ソルバが提供されている．**help lsode** と入力すると，必要な引数，戻り値の詳細な説明が得られる．

```
>> help lsode
'lsode' is a built-in function from the file libinterp/corefcn/lsode.cc

 -- Built-in Function: [X, ISTATE, MSG] = lsode (FCN, X_0, T)
 -- Built-in Function: [X, ISTATE, MSG] = lsode (FCN, X_0, T, T_CRIT)
     Ordinary Differential Equation (ODE) solver.

     The set of differential equations to solve is
```

(次のページに続く)

(前のページからの続き)

```
          dx
          -- = f (x, t)
          dt

     with

        x(t_0) = x_0

   The solution is returned in the matrix X, with each row
   corresponding to an element of the vector T.  The first element of
   T should be t_0 and should correspond to the initial state of the
   system X_0, so that the first row of the output is X_0.

 # (後略)
```

簡単にまとめると, 関数 FCN で定義される常微分方程式を, X_0 を初期値として求める. 出力する時刻は T のベクトルで与えられ, それぞれの時刻に対応する数値解は X として得られる.

関数 FCN は

$$(dx_1/dt, dx_2/dt, \ldots, dx_n/dt) = f((x_1, x_2, x_3, \ldots, x_n), t)$$

というように, ある時刻 t における n 個の状態変数が与えられた場合, それぞれの微係数を求める関数である. 本問題では,

```
# グローバル変数として crprob 関数からも参照できるようにする.
global V0;
global C;
global R;
V0 = 1;
C = 1;
R = 1;

function dxdt = crprob(x, t)
  global V0;  # V0 等はグローバル変数であることを明示する
  global C;
  global R;
  dxdt(1) = (1/RC)*(V0-x(1));  # x, dxdt は要素数 1 のベクトルである
endfunction
```

である.

初期値 X_0 は [0], 数値解を求める (複数の) 時刻 T は解析解と同じにする.

```
1;

# crprob 関数内部からも参照できるように global 変数とする.
global V0;
global C;
global R;
V0 = 1;
```

(次のページに続く)

(前のページからの続き)

```
C = 1;
R = 1;

t = linspace(0, 5*C*R, 51);

function dxdt = crprob (x, t)
  global V0;
  global R;
  global C;
  dxdt(1) = (1 / (C*R))*(V0 - x(1));
endfunction

# stat, msg には計算結果の可否などの情報がはいる
[vcn, stat, msg] = lsode("crprob", [0], t);

plot(t, vcn, "-x;numeric;");  # 3番目の文字列は, プロットのスタイル, 凡例のラベル等
xlabel("t", "fontsize", 18);
ylabel("vc", "fontsize", 18);
legend("location", "northwest");

# 必要であればプロット画像, csv データでの保存をこの後行うとよい.
```

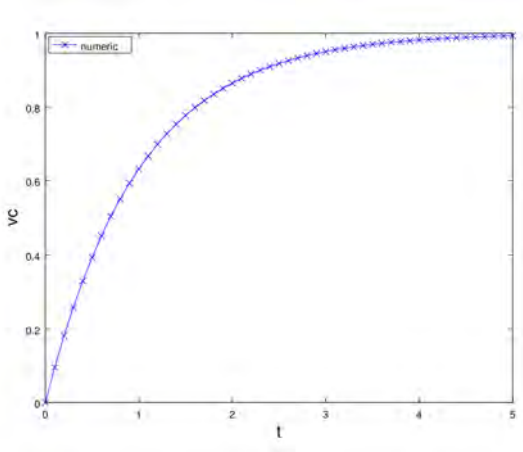


図 14.3　RC 回路のステップ応答 (数値解)

14.1.4 課題

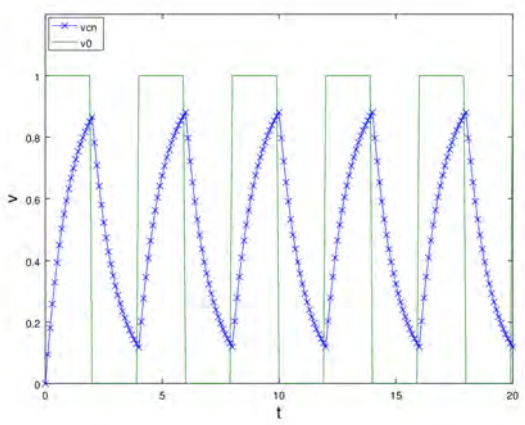
1. 上記の解析解, 数値解の時間変化を同一のグラフに図示せよ. また, 両者の差の時間変化を図示せよ.

2. 上記問題では V_0 を一定としたが, これを,

$$V_0(t) = \begin{cases} 1 & (0 + 4RC \times n \leq t < 2RC + 4RC \times n) \\ 0 & (2RC + 4RC \times n \leq t < 4RC + 4RC \times n) \end{cases}$$

   と $4RC$ の周期で周期的に変化させる. 時間 $20RC$ (5 周期分) にわたり, V_0, v_c を図示せよ.

   解答例




```
1;

global V0;
global R;
global C;
V0 = 1;
R = 1;
C = 1;

% 周期 4CR の矩形波
function ret = v0t(t)
  global C;
  global R;
  global V0;
  % 0 <= phase < 1 で 1 周期となる位相を求める
  phase = mod(t, 4*C*R)/(4*C*R);
  ret = V0;
  if ( phase >= 0.5 )
    ret = 0;
  endif
endfunction

function dxdt = crprob (x, t)
  global R;
  global C;
  dxdt(1) = (1 / (C*R))*(v0t(t) - x(1));
endfunction

t = linspace(0, 20*C*R, 201);
% arrayfun を用い
% (v0(1), v0(2), ... ) = (v0t(t(1)), v0t(t(2)), ... )
% を一度に計算する
v0 = arrayfun("v0t", t);

[vcn, stat, msg] = lsode("crprob", [0], t);

plot(t, vcn, "-x;vcn;", t, v0, ";v0;");
```

(次のページに続く)

(前のページからの続き)

```
xlabel("t", "fontsize", 18);
ylabel("v", "fontsize", 18);
ylim([0, 1.2]);
legend("location", "northwest");
```

3. さらなる発展課題として以下のようなものがある

   - 平均電圧, 平均電力の算出 (一番最後の周期で計算)
   - 波形を変更した場合の挙動 (矩形波, 三角波, 正弦波)
   - 周波数を変化させ, Low Pass Filter (LPF) としての特性を確認する. 周波数を高くするほど振幅が減少し, 波形を正確に再現できなくなる.

14.2 LC 発振回路

容量 C コンデンサとインダクタンス L のインダクタに加わる電圧, 並びに流れる電流をそれぞれ y_1, y_2 とする.

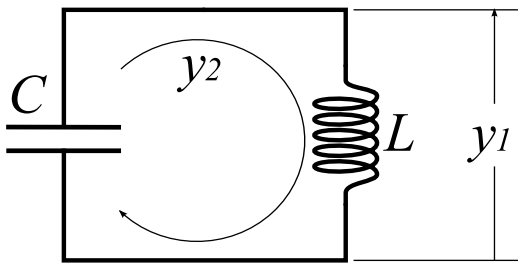


図 14.4　LC 発振回路

この時, y_1, y_2 の満たす微分方程式は,

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = -\dfrac{1}{C}y_2 \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = \dfrac{1}{L}y_1 \end{cases}$$

である.

14.2.1 課題

初期値を $(1, 0)$ とした時の $(y_1(t), y_2(t))$ の軌跡 (相平面図) を解析解, 数値解ともに図示せよ. 数値解導出に必要な関数は, 次のようになる.

```
global L;
global C;
L= ??;   % 適当な値を定める
```

(次のページに続く)

(前のページからの続き)

```
C= ??;   % 適当な値を定める

function dxdt = lcprob(x, t)
  global L;
  global C;
  dxdt(1) = - 1 / C * x(2);
  dxdt(2) = 1 / L * x(1);
endfunction
```

14.3 非線形発振回路

LC 発振回路に, 下記の電圧-電流特性を持った非線形素子を挿入する.

$$i(v) = \gamma v^3 - \alpha v, \quad \gamma, \alpha > 0$$

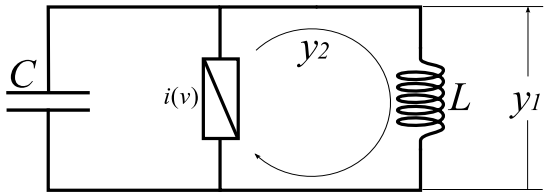


図 14.5　非線形素子を挿入した LC 発振回路

この素子は, $-\sqrt{\alpha/\gamma} < v < \sqrt{\alpha/\gamma}$ の電圧において負性抵抗を示す. この回路の微分方程式は,

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = -\dfrac{1}{C}(y_2 + i(y_1)) \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = \dfrac{1}{L} y_1 \end{cases}$$

14.3.1 課題

1. $i(v)$ を描画し, 負性抵抗となる v の範囲を確認せよ.
2. (L, C, γ, α) の組を適当に定め,
   a. $(\Delta y_1, \Delta y_2)$ の向き, 大きさを相平面に図示し, 解の挙動を予測せよ.
   b. 初期値を変え, (y_1, y_2) の軌跡を求めよ.

この回路は van der Pol 回路と呼ばれるものである.

第 15 章

Python での科学技術計算

Python はシステム管理, web フレームワーク, 科学計算, 画像処理, 自然言語処理, 生命情報等, 様々な分野で広く普及しているプログラミング言語である.

ここでは, 本演習に関連するテーマについて, Python を用いた例を紹介する.

15.1 Python のセットアップ

Python のセットアップ方法については以下のようなものがある. 資料が豊富にあるため, ここでは詳細は述べず, 各自に適した方法をとること.

15.1.1 公式ページが紹介するページを利用する.

https://www.python.org/ から適当なものをダウンロード, インストールする

サードパーティのパッケージは *pip* で管理することが一般的である.

```
pip install matplotlib
```

15.1.2 anaconda をインストール

科学技術計算をはじめとして多様なパッケージを網羅的に取りまとめて提供している. また, 対話的な実行環境として, Jupyter Notebook も利用できる.

https://www.anaconda.com/

追加のパッケージが必要な場合, *conda* コマンドを使う

15.1.3 Google Colaboratory

Google がクラウド上で提供する Jupyter Notebook. Google アカウント (とブラウザ) だけ持っていれば, 特別なセットアップが不要で利用できる.

https://colab.research.google.com

15.2 ベクトル, 行列, 科学計算と描画 (numpy, scipy, matplotlib)

ガイド中では線形代数, 微分方程式をはじめとする高度な科学技術計算, グラフ描画の統合的な環境として Octave での事例を取り上げた. Python においてはこれらの機能は,

- ベクトル, 行列に関する演算 → numpy
- 高度な科学技術計算 → scipy
- グラフ描画 → matplotlib

が事実上の (de facto, デファクト) 標準として用いられる.

Octave による LCR 直列共振回路の解析 にある Octave のスクリプトはそれぞれ以下のようになる.

15.2.1 ベクトル場の図示

numpy, scipy, matplotlib は, Matlab (及び Octave) の影響を強く受けているため, 関数や属性の命名は非常によく似ている.

リスト 15.1　ex1_1.py ベクトル場の作図 (Python + numpy + matplotlib)

```
#!/usr/bin/env python

# ex1_1.py
# 電気電子回路演習
# 演習課題 1 ベクトル場の作図 ex1_1.m とコンパチに

import numpy
import matplotlib.pyplot as plt


def ex1_1(M):
    """ -1 <= (i, v) <= 1 に対する (di, dv)^T = M * (i, v)^T をプロットする.
    """
    # -1 <= (i, v) <= 1 の空間を (10, 10) 等分する.
    i_x = numpy.linspace(-1, 1, 11)
    v_y = numpy.linspace(-1, 1, 11)

    # meshgrid 関数により, 各グリッドにおける x, y 座標の 2 次元配列 (=行列) を作成する.
    # x, y はそれぞれ len(i_x) x len(v_y) の行列である.
    # サイズ, 内容を確認すること.
```

(次のページに続く)

(前のページからの続き)

```
    (x, y) = numpy.meshgrid(i_x, v_y)

    # (u(s,t), v(s,t)) = M * (x(s,t), y(s,t))^T を求める.
    # インデックスが 0 から始まる.
    u = M[0, 0] * x + M[0, 1] * y
    v = M[1, 0] * x + M[1, 1] * y

    # ベクトル場のプロット
    plt.quiver(x, y, u, v, color='blue')


if __name__ == '__main__':
    # (i, v) -> (di/dt, dv/dt) の写像を定義する行列
    R = 3
    L = 1
    C = 1

    M = numpy.array([[-R/L, -1/L], [1/C, 0]])

    ex1_1(M)

    plt.xlim(-1.2, 1.2)
    plt.ylim(-1.2, 1.2)
    plt.xlabel("I")
    plt.ylabel("V")
    plt.show()
```

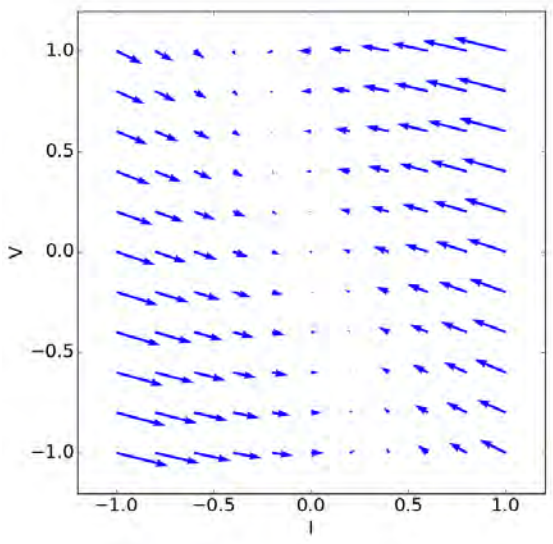


図 15.1　ex1_1.py の実行結果

15.2.2 微分方程式の数値解を求める

リスト 15.2　ex1_2.py 微分方程式ソルバの利用 (Python + scipy)

```
#!/usr/bin/env python

# ex1_2.py
# 電気電子回路演習
# 演習課題 1.2 微分方程式ソルバの利用 ex1_2.m とコンパチに

import numpy
from scipy.integrate import ode
import matplotlib.pyplot as plt


def ex1_2(M, x0, ts):
    """行列 M で与えられる dX = M X の微分方程式の数値解を ts の要素にある時刻毎に求め,
    プロットする.
    M: 行列
    x0: ts[0] における初期値
    ts: 数値解を求める時刻
    """

    def prob(t, xs):
        """dx/dt = f(t, x) を表す関数
        octave とパラメータの並びが異なることに注意.
        """
        return M.dot(xs)

    # 微分方程式ソルバの準備
    solver = ode(prob)
    solver.set_integrator("dopri5")  # Matlab の ode45 と同じ解法
    solver.set_initial_value(x0, ts[0])

    # 解を収める.
    xs = [x0]

    for t in ts[1:]:
        xs.append(solver.integrate(t))

    xs = numpy.array(xs).T

    # プロット
    plt.plot(xs[0], xs[1], "-rx")


if __name__ == '__main__':
    from ex1_1 import ex1_1

    # (i, v) -> (di/dt, dv/dt) の写像を定義する行列
    R = 3
    L = 1
    C = 1
```

(次のページに続く)

(前のページからの続き)

```
M = numpy.array([[-R/L, -1/L], [1/C, 0]])
x0 = [0.8, 0.8]
ts = numpy.linspace(0, 10, 101)

ex1_2(M, x0, ts)

# 課題 1.1 のプロット
ex1_1(M)

plt.xlim(-1.2, 1.2)
plt.ylim(-1.2, 1.2)
plt.xlabel("I")
plt.ylabel("V")
plt.show()
```

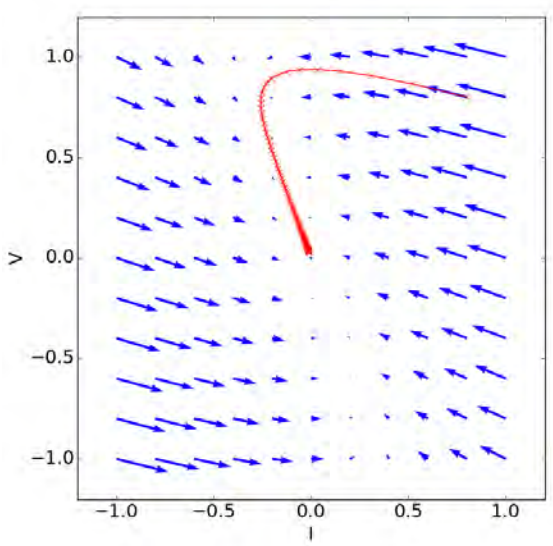


図 15.2　ex1_2.py の実行結果

15.2.3 固有値・固有ベクトルを求め相平面上に図示する

リスト 15.3　ex1_3.py 固有値, 固有ベクトルの導出と図示 (Python)

```
#!/usr/bin/env python

# ex1_3.py
# 電気電子回路演習　演習課題 1.3
# 固有値, 固有ベクトルの導出と図示　ex1_3.m とコンパチに

import numpy
import matplotlib.pylab as plt
```

(次のページに続く)

(前のページからの続き)

```
def ex1_3(M):
    """行列（二次元配列） M の固有値，固有ベクトルを求め，相平面上に直線をプロットする
    """

    # numpy.linlag.eig は固有値を求める関数．
    # ks は固有ベクトル，lams に固有値が得られ（Octave と返り値の順序が逆），
    #   lams(i) * ks(:,i) = M * ks(:,i)
    # が成立する．
    lams, ks = numpy.linalg.eig(M)

    print(lams[0], ks[:, 0])
    print(lams[1], ks[:, 1])

    # 描画する
    # 1. k0, k1 のベクトルを長さ 2 となるまで延長
    # 2. (k0 と -k0) (k1 と -k1) を結ぶ直線を引く．
    k0 = (2 / numpy.linalg.norm(ks[:, 0])) * ks[:, 0]
    k1 = (2 / numpy.linalg.norm(ks[:, 1])) * ks[:, 1]

    # k1 は緑の実線（solid），k2 は緑の破線（dashed）
    plt.plot([k0[0], -k0[0]], [k0[1], -k0[1]], "g",
             [k1[0], -k1[0]], [k1[1], -k1[1]], "--g")


if __name__ == '__main__':
    from ex1_1 import ex1_1
    from ex1_2 import ex1_2

    # (i, v) -> (di/dt, dv/dt) の写像を定義する行列
    R = 3
    L = 1
    C = 1

    M = numpy.array([[-R/L, -1/L], [1/C, 0]])

    ex1_3(M)

    # 課題 1.2 のプロット
    x0 = [0.8, 0.8]
    ts = numpy.linspace(0, 10, 101)
    ex1_2(M, x0, ts)

    # 課題 1.1 のプロット
    ex1_1(M)

    plt.xlim(-1.2, 1.2)
    plt.ylim(-1.2, 1.2)
    plt.xlabel("I")
    plt.ylabel("V")
    plt.show()
```

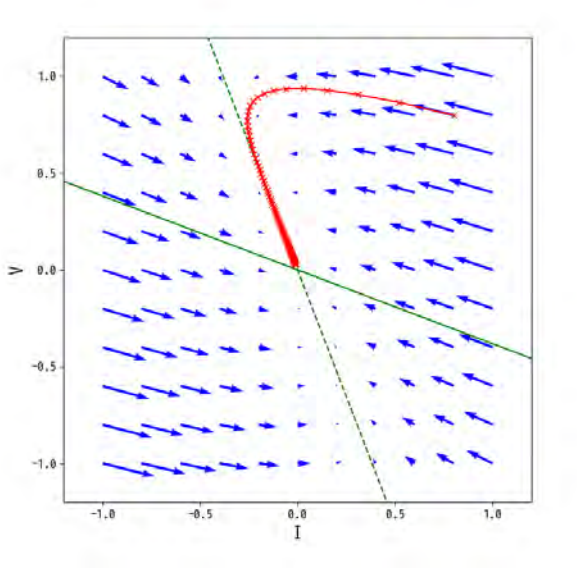


図 15.3　ex1_3.py の実行結果

15.3 数式処理 (SymPy)

能動回路の実験 (及びテキストの図 12.2) では, 3 段の CR フィルタを通じて信号の位相を 180 度変え帰還回路を構成している. この位相が反転する条件は, テキスト式 12.4 以下の議論より, $\omega_0 = \sqrt{6}/CR$ となる. この煩雑な数式の導出に, SymPy という数式処理パッケージを用いる.

注釈: 同様の機能を提供するソフトウェアに, Mathmatica, Maple, Gnu Maxima 等がある.

---

3 段, 4 段の CR フィルタにおいて, 位相が反転する周波数条件をもとめるスクリプトは以下のようになる. 数式内で扱う変数とその条件 (実数, 正数等) を *sympy.Symbol()* により定義し, 以下一般の数式と同様に定義を書き進める. 途中,

- *.expand()* メソッドによる式の展開
- *.solve()* メソッドによる方程式の求解
- *.sub()* メソッドによる代入

といった機能を利用している.

リスト 15.4　SymPy を用いた 3 段, 4 段フィルターの反転周波数とゲインの導出

```
import sympy

C = sympy.Symbol('C', real=True)
R = sympy.Symbol('R', real=True)
om = sympy.Symbol('om', real=True)
```

(次のページに続く)

(前のページからの続き)

```
F = sympy.Matrix([
    [1 + om * C * R * sympy.I, R],
    [om*C*sympy.I, 1]])

F2 = F**2
F3 = F**3
F4 = F**4

def solve_inverse_cond(F):
    '''4 端子行列を元に,
    I_i = 0 として V_i = G V_o を計算
    G 位相が 180 度ずれる周波数 om_0 の条件, その時のゲイン A を求める
    '''

    I_o = sympy.Symbol('I_o')
    V_o = sympy.Symbol('V_o')

    VI_i = F * sympy.Matrix([[V_o], [I_o]])

    V_i = VI_i[0]
    Vg = V_i.coeff(V_o)
    Vg_im = sympy.im(Vg)
    Vg_re = sympy.re(Vg)

    om_0 = sympy.solve(Vg_im, om)
    Vg_re_0 = Vg_re.subs(om, om_0[2])

    return om_0, Vg_re_0

print('F3:')
print('  ', F3.expand())
om_0, Vg_0 = solve_inverse_cond(F3)
print('  om_0:', om_0)
print('  Vg_0:', Vg_0)

print('F4:')
print('  ', F4.expand())
om_0, Vg_0 = solve_inverse_cond(F4)
print('  om_0:', om_0)
print('  Vg_0:', Vg_0)
```

リスト 15.5　前のプログラムの実行結果

```
F3:
   Matrix([[-I*C**3*R**3*om**3 - 5*C**2*R**2*om**2 + 6*I*C*R*om + 1, -
↪C**2*R**3*om**2 + 4*I*C*R**2*om + 3*R], [-I*C**3*R**2*om**3 - 4*C**2*R*om**2 +␣
↪3*I*C*om, -C**2*R**2*om**2 + 3*I*C*R*om + 1]])
  om_0: [0, -sqrt(6)/(C*R), sqrt(6)/(C*R)]
  Vg_0: -29
F4:
   Matrix([[C**4*R**4*om**4 - 7*I*C**3*R**3*om**3 - 15*C**2*R**2*om**2 +␣
↪10*I*C*R*om + 1, -I*C**3*R**4*om**3 - 6*C**2*R**3*om**2 + 10*I*C*R**2*om + 4*R],␣
↪[C**4*R**3*om**4 - 6*I*C**3*R**2*om**3 - 10*C**2*R*om**2 + 4*I*C*om, -
↪I*C**3*R**3*om**3 - 5*C**2*R**2*om**2 + 6*I*C*R*om + 1]])
```

(次のページに続く)

(前のページからの続き)

```
om_0: [0, -sqrt(70)/(7*C*R), sqrt(70)/(7*C*R)]
Vg_0: -901/49
```