

5. 電気・電子回路のシミュレーションの実行(1), (2)

【実験の目的】

2 年, 3 年で学習した電気回路や電子回路およびその応用回路についてパソコンによるシミュレーションを行い, その回路の動作をより深く理解することを目的とする。

具体的には, 回路解析シミュレーション・ソフトウェアの使い方をマスタするとともに, 電気回路のシミュレーションの結果を解析し考察することにある。

【シミュレーション・ソフトウェアについて】

多くの技術者が広く使用している, 電気・電子回路のシミュレーション・ソフトウェア SPICE (Simulation Program with Integrated Circuit Emphasis) の使い方について説明する。今回は米国 OrCAD 社が販売しているソフトウェアで, 回路図入力ツールとして OrCAD Capture CIS Lite Edition, シミュレーション・ツールとして OrCAD PSpice を使用する。(米国製ソフトウェアのため, 英和辞典を持参すると便利である)

I. 電気・電子回路図の入力について

パソコン上で回路図の入力や修正は, ソフトウェア”OrCAD Capture CIS Lite Edition”を用いる。Capture CIS Lite Edition の操作方法について説明する。

(1) [スタート]-[プログラム]-[Orcad Family Release9.2Lite Edition]-[Capture Lite Edition]を起動して, メニューの[File]-[New]-[Project]をクリックする。

[Name]には「example」, [Create New Project Using]は「Analog or Mixed A/D」を選択する。

[Location]には「W:¥circuit」と入力して[OK]ボタンを押す(図 1 参照)。

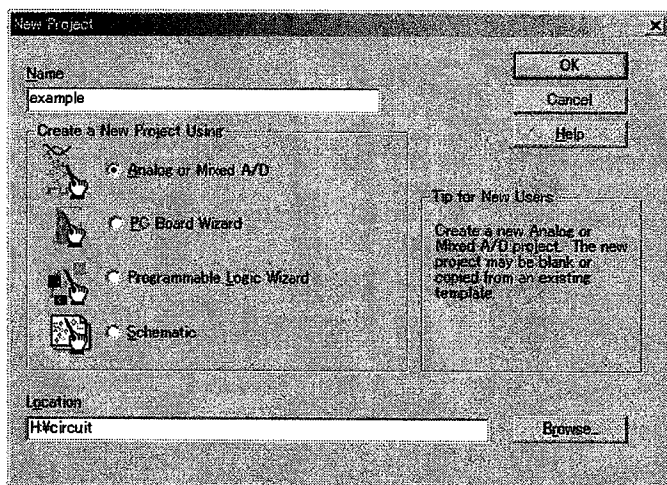


図 1 OrCAD の Project 作成画面その 1

Create PSpice Project ウィンドウでは[Create a blank project]を選択し[OK]ボタンを押す(図 2 参照)。

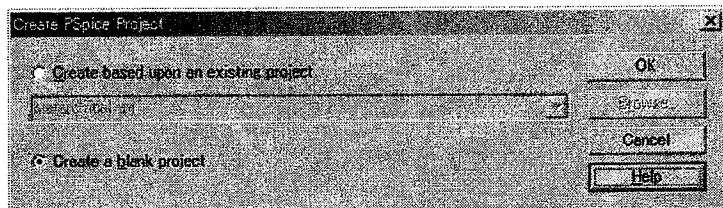



図 2 OrCAD の Project 作成画面その 2

(2) 回路エディタが開き、図 3 のような新しい画面が表示される。この上で回路図を作成する。右辺のツールパレットの上から 2 番目[Place part]ボタンをクリックする。

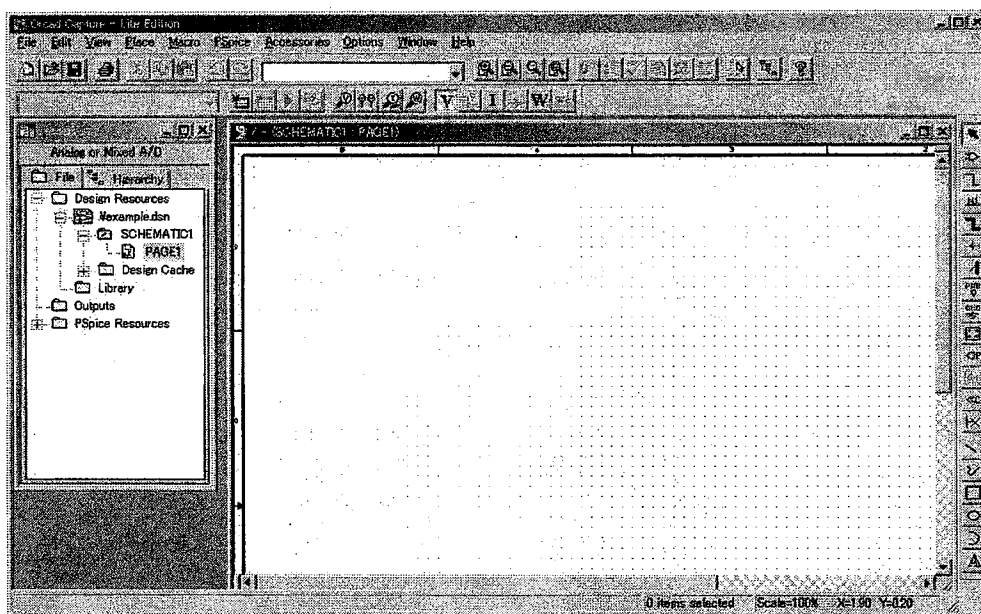


図 3 回路エディタの初期画面

(3) 部品(抵抗 R, コンデンサ C, インダクタンス L, 電源)の呼び出し

最初は[Libraries]に何も読み込んでいないので[Part List]は空白になっている。

[Add Library...]ボタンを押してライブラリを追加する。

抵抗パーツを配置する場合、[Libraries]から「ANAROG_P」を選択し、[Part List]で抵抗「R」を選択して[OK]ボタンを押す(図 4 参照)。コンデンサ C、インダクタンス L も同様に「ANAROG_P」から呼び出すことができる。

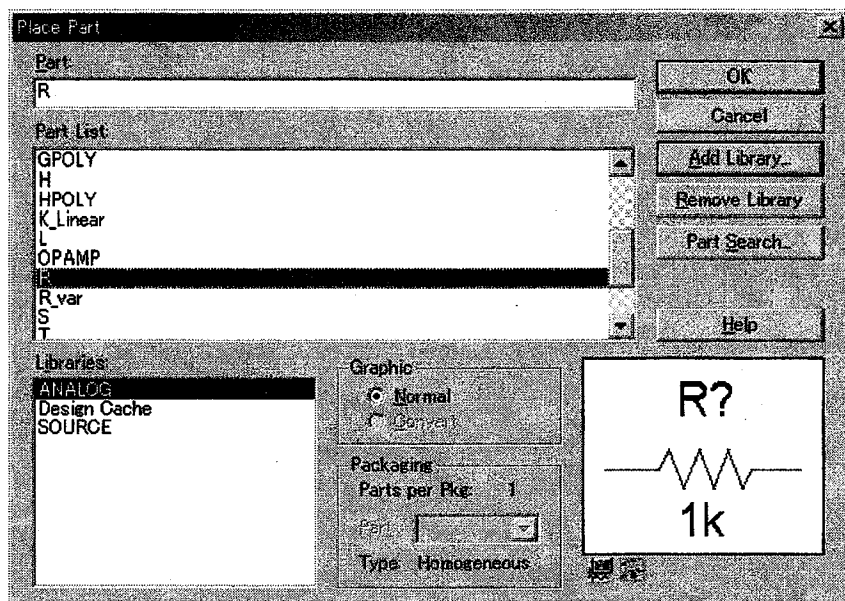


図 4 部品の選択画面例

回路図エディタ上の抵抗を配置したい場所でマウスを左クリックすることにより配置される。配置モードを終了する場合は、マウス右クリックのメニューから[End Mode]を選択する。◎部品の向きを90度変更する場合は、部品全体を選択して右クリックのメニューから[Rotate]をクリックする(図5参照)。

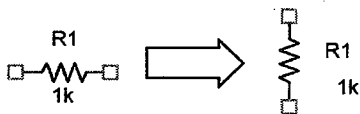




図 5 部品の回転

(4) 入力信号源（電源）の配置と数値の設定

[Place part]ボタンをクリックし、Add library→source→VSIN→OKでエディタ上に正弦波入力信号源“Vsin”を配置する。このシンボルのVOFF欄（オフセット）やVAMPL欄（振幅電圧(最大値)）、FREQ欄（周波数 Hz）をクリックし、値を設定し閉じ終了する。なお、m（ミリ）、u（μ）、n（ナノ）、p（ピコ）、meg（メガ）である。

(5) 各素子間の結線について

右辺のツールパレットのワイヤボタン（記号：）をクリックし、エディタ上で素子間をマウスで移動し結線する（斜線は不可）。マウス右クリックしend wireをクリックすれば結線モードを終了する。

(6) GND の配置

ツールパレットのGNDボタンをクリックする。Place groundが開くので、Add library

→source→0→OK をクリックし、グラウンド・パーツを配置する。マウス右ボタンで終了する。GND は不可欠で、通常電源の下のマイナス側に配置し、電源と結線で接続する。



II. PSpice によるシミュレーションについて

Capture CIS Lite Edition で入力した回路図について、Pspice を用いてシミュレーションを実行する手順を説明する。

(1) シミュレーションの前処理 (過渡解析の場合について説明)

Capture CIS Lite Edition 画面のメニュー PSpice から New Simulation Profile をクリックする。New Simulation ダイアログボックスが表示されるので、シミュレーション名を各自設定する。次に simulation Settings ダイアログボックスが表示されるので、analysis type に TimeDomain を Option に GeneralSetting を入力する。Run To Time に 1.04 を、Start Saving... に 1.0 (解析 1 秒後から 40ms の区間を表示) を入れる。OK クリックで終了する。(最初は過渡状態なので、定常状態になった波形を表示する)

(2) シミュレーション結果の確認

調べたいノードに電圧又は電流探索 Probe ( 又は ) を立てる。これにより、該当するノードのシミュレーションの結果を波形で Probe ウィンドウに表示させ検証することができる。またこの波形を印刷可能である(図 6 参照)。

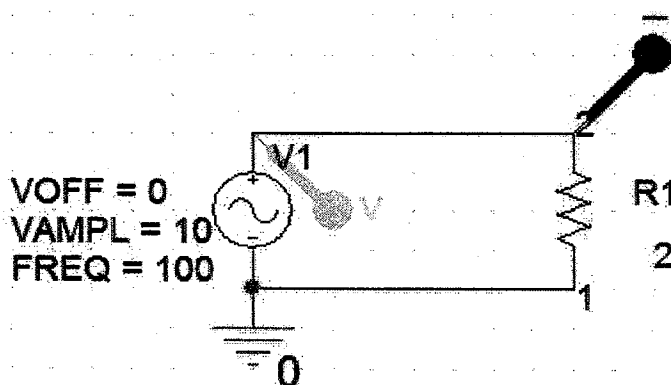


図 6 作成される回路の例

(3) シミュレーションの実行

PSpice メニューから run をクリックする。OrCAD PSpice が自動的に立ち上がり、シミュレーションが開始する。Probe ウィンドウが表示されシミュレーションの Run 状況を確認できる。

[演習問題]

R,L,C の素子で構成される回路に交流電圧を印加したとき、電圧と電流の位相関係をシミュレーションにより解析する。

1. 例題

図 7(a)のように交流正弦波起電力 $E[V]$ を抵抗 $R[\Omega]$ の抵抗器に接続した回路を考え、シミュレーションにより抵抗 $R[\Omega]$ の両端に生じる電圧 $V[V]$ と流れる電流 $i[A]$ の波形を求める。この波形から電圧と電流の位相差を読み取り、両者の関係をベクトルで書け。

(指定事項) ①交流正弦波起電力：シンボル V_{\sin} 使用

(周波数 100 と 200[Hz], 振幅値 10[V], $V_{\text{off}}=0[V]$ は各自設定のこと)

②抵抗値は各自設定のこと(例えば、 $2[\Omega]$)。

(例) 理論的考察

交流起電力 $e = \sqrt{2} E \sin \omega t$ は常に抵抗の両端の v として、現れる。したがって抵抗の両端の電圧 v は次のようになる。

$$v = \sqrt{2} E \sin \omega t \quad (\text{但し, } V=E)$$

このとき流れる電流 $I [A]$ はオームの法則から次のようになる。

$$i = v/R = \sqrt{2} E/R \sin \omega t$$

この式より、電圧と電流の関係は同相で、波形とベクトル図は図 7(b)(c)のようになる。

(例) シミュレーション結果と評価

シミュレーションにより電圧と電流の波形をプロットした結果、図 7(b)のようになり、位相関係は同相となり理論とシミュレーションが一致し正しいことが判る。(シミュレーション波形のピーク値、実効値が理論的計算値と一致することを確認すること)

インダクタンスや静電容量は、交流が流れると抵抗とは異なった働きをする。交流回路における R, L, C の働きについてシミュレーションにより動作確認をする。

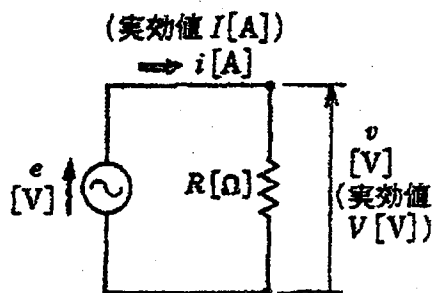


図 7(a) 負荷が抵抗だけの回路

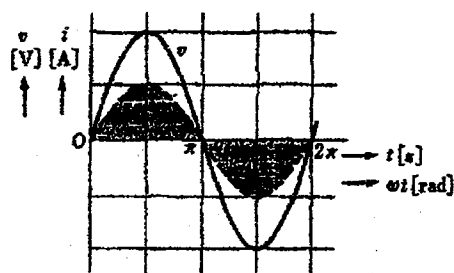


図 7(b) 交流における抵抗の働き



図 7(c) ベクトル図

2. 演習 1

図 8 のように交流正弦波起電力 $e[V]$ の電源に、インダクタンス $L[H]$ のコイルを接続した回路を考え、コイルの両端に生ずる電圧 $V[V]$ と、コイルに流れる電流 $i[A]$ をシミュレーションにより求めなさい。さらに、この波形より両者の大きさ (実効値) と位相の関係を示すベクトル図を書け。

注:インダクタンスの上部に直列に抵抗(0.1[Ω])を入れること

(指定事項) 交流正弦波起電力: シンボル Vsin 使用(周波数と振幅値は各自設定のこと)

注: $V_{off}=0[V]$, $V_{amp}=10[V]$, $Freq=100[Hz]$ および $200[Hz]$ に設定する

インダクタンスの値は各自設定のこと。(例えば、リアクタンス ωL が $2[\Omega]$ 程度)

3. 演習 2

図 9 のように交流起電力 $e[V]$ の電源に、静電容量 $C[F]$ のキャパシタンスを接続した回路を考え、コンデンサの両端に生ずる電圧 $v[V]$ と、コンデンサに流れる電流 $i[A]$ をシミュレーションにより求めなさい。さらに、この波形より両者の大きさ(実効値)と位相の関係を示すベクトル図を書け。

注:コンデンサの上部に直列に抵抗(0.1[Ω])を入れること

(指定事項) 交流正弦波起電力: シンボル Vsin 使用(周波数と振幅値は各自設定のこと)

注: $V_{off}=0[V]$, $V_{amp}=10[V]$, $Freq=100[Hz]$ および $200[Hz]$ に設定する

コンデンサの値は各自設定のこと。(例えば、リアクタンス $1/\omega C$ が $2[\Omega]$ 程度)

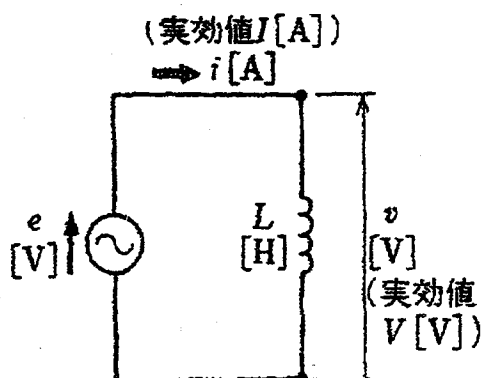


図 8 負荷がインダクタンスだけの回路

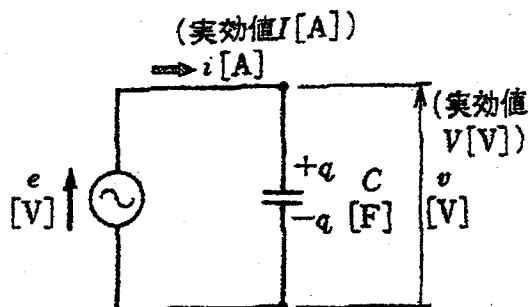


図 9 キャパシタンスを接続した回路

4. 演習 3

- (1) 図 10 に示す R, L, C を直列に接続した回路の周波数特性(周波数可変範囲 $10[Hz]$ から $200[Hz]$ までとする)をシミュレーションする。これにより、横軸を周波数、縦軸を抵抗 R 、インダクタンス L 及びコンデンサ C の、それぞれの両端の電圧 (V_R, V_L, V_C) および図 10 の V_{LC} としたグラフを表示する。また、別に電流也表示させること。

このグラフより、共振周波数(最も電流が大きくなる周波数)の概略値と、共振時の電流と各電圧 (V_R, V_L, V_C, V_{LC}) の大きさを求めよ。

- (2) 図 10 に示す R, L, C を直列に接続した回路について、演習 1 や 2 と同様に、各素子の電圧、電流の波形を表示させて位相を求め、これを元にベクトル図を示せ。なお、電源

の周波数は、この回路の共振周波数（理論値）に設定する。

（指定事項）演習 3 の(1)は、演習 1～2 と比べ次の設定が異なる。

① 力信号源は、周波数をスイープ出来る **VAC** を使用せよ。

また、property editor により、ACMAG 欄に振幅 100V を入力せよ。

② （シミュレーションの前処理で）回路図エディタ画面のメニュー Pspice から simulation setting をクリックし、analysis type に **AC Sweep/Noise** を選択。AC Sweep Type に start freq. と end freq. を 10Hz 200Hz と入力する。Points/decade に 100 を入力。OK をクリックする。

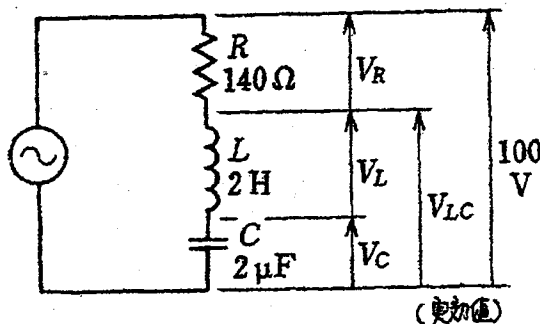


図 10 RLC を直列に接続した回路

[レポートについて]

各演習問題について、次の事柄を書きなさい

1. 理論的考察

・演習 1, 2, 3 について、教科書電気基礎 1（実教出版）第 4 章の交流回路を参考にして、電流と電圧の大きさや位相関係はどうなるかの理論的考察を上記の例題の項を参考にして書きなさい。

ベクトル図など図を用いて説明すること。（説明文章は 100 字を基準にしてください）

2. シミュレーション結果と評価

- ・シミュレーション結果については、各自使った素子の定数を示してください。
- ・シミュレーションの結果をプリントアウトし、波形やグラフから各電圧や電流の大きさと、電流を基準とした場合の位相を読み取り、理論値と一緒に表 1、2 のように表に示しなさい。評価については、表 1、2 から理論的考察した内容とシミュレーション結果を比較して各自評価をしてください。すなわちシミュレーションで求めた位相関係との関係を説明し考察しなさい。違った場合はなぜ違うのか述べてください。
- ・レポートに添付するデータ：演習 1～3 について、シミュレーション結果である波形図 3 枚と定規で書いたベクトル図（報告書中でも可）を添付する。
（この場合も説明文は約 100 字を基準にして説明してください）

インダクタンス $L=$

キャパシタンス $C=$

表 1 演習 1, 2 のまとめ

周波数		理論値	シミュレーション値
100 Hz	IL	$\angle \quad ^\circ$	$\angle 0^\circ$
	VL	\angle	\angle
	Ic	$\angle \quad ^\circ$	$\angle 0^\circ$
	Vc	\angle	\angle
200 Hz	IL	$\angle \quad ^\circ$	$\angle 0^\circ$
	VL	\angle	\angle
	Ic	$\angle \quad ^\circ$	$\angle 0^\circ$
	Vc	\angle	\angle

表 2 演習 3 のまとめ

	理論値	シミュレーション値
I	$\angle 0^\circ$	$\angle 0^\circ$
Vr	\angle	\angle
VC	\angle	\angle
VL	\angle	\angle
VLC	\angle	\angle

参考書： (2 年の教科書) 電気基礎 1, 実教出版

2002.4.8 井上和勇作成, 2004.4.1 修正 (アンダーライン部分)

2002.5.13 日下孝二 追加

2013.12.20 E. Ohira 修正

第2週

ネットワーク演習（2）

2.1 概要

本演習では、インターネットを構築する技術のうち、基礎的な Web サーバに関わる技術を、2回に分けて学ぶ。1週目では、Web サーバの理解と Web アプリケーションのうちクライアントサイドプログラムの基礎を学んだ。2週目では Web アプリケーションのうちサーバサイドプログラムの基礎を学ぶ。

サーバサイドプログラムは PHP、Perl、Java、C# や Ruby などさまざまな言語で作成することができるが、本演習では Python を使ったサーバサイドプログラムを作成する。なお、本演習では Python 3.x の利用を仮定して話を進める。

2.2 Python を使ったサーバサイドプログラムの作成

2.2.1 WSGI の概要

Python で作成されたプログラムを Web アプリケーションとして既存の Web サーバソフトウェアで動かすための統一規格として、**WSGI(Web Server Gateway Interface)** がある。WSGI の要件を満たす Python プログラムであれば、Apache2 などの Web サーバ上で動作させることができる。WSGI を使って動かすサーバプログラムを記述するためには、次の要件に注意するだけである。

- ・サーバがプログラムを起動した際、application という名称の関数が実行される。
 - 第1引数として環境変数オブジェクトが渡され、第2引数として HTTP レスポンスヘッダ(後述)を作成する関数が渡される。

- 返り値は、HTTP レスポンスボディ (後述) に相当するバイト列オブジェクト (UTF-8 などの文字コードで符号化されたオブジェクト) を含んだリストオブジェクトである。

WSGI の要件を満たした最小構成のプログラム作成例を図2.1示す。

```

1 def application (environ, start_response):
2     """ WSGIで最初に呼び出される関数.
3     """
4     output = 'Hello World!'.encode('utf-8')
5     start_response('200 OK', [('Content-Type', 'text/plain;
        charset=UTF-8')])
6
7     return [output]
```

図2.1: /var/www/html/wsgi_hello.py のソースコード、`__`はタブを示す

この5行のプログラムでは、次の内容の1つの application 関数が定義されている。

- 2行目で、受け取った HTTP レスポンスヘッダ (後述) を作成する start_response 関数を使って、HTTP レスポンス状態コード (後述、ここでは 200 OK) などを指定してレスポンスヘッダを作成
- 3行目で、HTTP レスポンスボディ (後述) として返却する内容を、文字列オブジェクトが持つ encode メソッドを使って UTF-8 文字列 (バイト列) として作成
- 5行目で、3行目で作成したバイト列をリストでくるんで返却

この例を見る限り Web サーバプログラムは、基本的には、ブラウザで表示したい文字列をバイト列オブジェクトのリストにして、返り値として返せば良いことがわかる。

WSGI で Python プログラムを実行するためには WSGI の機能を有効にする必要がある。Apache2 は標準で WSGI の機能がインストールされていないため、Apache2 向け WSGI モジュールを図2.2のコマンドでインストールする。

```

user@hostname ~> sudo apt-get install
    libapache2-mod-wsgi-py3
```

図2.2: Apache2 用の WSGI(Web Server Gateway Interface) モジュールのインストール

WSGI モジュールをインストールすることで、Apache2 の WSGI 設定が有効になる。図2.3に示すように/etc/apache2/sites-enabled/000-default.conf の VirtualHost ブロック内の末尾に図2.4に示す1行で、URL 上のパスと Python プログラムの Web サーバ上でのパスの対応関係を追加する。

```

<VirtualHost *:80>
    # The ServerName directive sets the request scheme,
    # hostname and port that
    # the server uses to identify itself. This is used when
    # creating
    :

    WSGIScriptAlias /wsgi_hello /var/www/html/wsgi_hello.py

</VirtualHost>

```

図2.3: wsgi_hello.py のための Apache2 の 000-default.conf 設定例、_はスペースを表す

```
WSGIScriptAlias [URL 上のパス] [サーバ上のプログラムパス]
```

図2.4: 設定ファイルに追加する行中の各項目の意味、_はスペースを表す

設定後、Apache2 を図2.5のように再起動することで、設定が有効になる。

```
user@hostname ~> sudo systemctl restart apache2
```

図2.5: Apache2 の再起動

問題 2.1.

図2.1のプログラムを、“Hello World! < 自分の名前 >” が返却されるよう改変し、アクセスした際のブラウザのスクリーンショットを報告しなさい。

2.3 HTTP メッセージ

ここからはメッセージボードプログラム board.py を作成しながら、HTTP の仕組みを学習していく。

HTTP メッセージとは、Web サーバとブラウザがデータを交換する手段である。HTTP メッセージは、ブラウザがサーバに何らかの処理 (アクション) を起こさせるために送信する命令 (リクエスト) と、その後、サーバがブラウザに送信する、サーバが行ったアクションに対する回答 (レスポンス) の2種類のメッセージに分けられる。 Web サーバで動作するプログラム、サーバサイドプログラムはそのプログラムの URL に HTTP リクエストを送ることで、実行される。サーバサイドプログラムの動作例を図2.6に示す。

HTTP リクエストを受け取った Web サーバは対応する URL のプログラムを実行し、実行結果を HTTP レスポンスとしてブラウザに送信する。

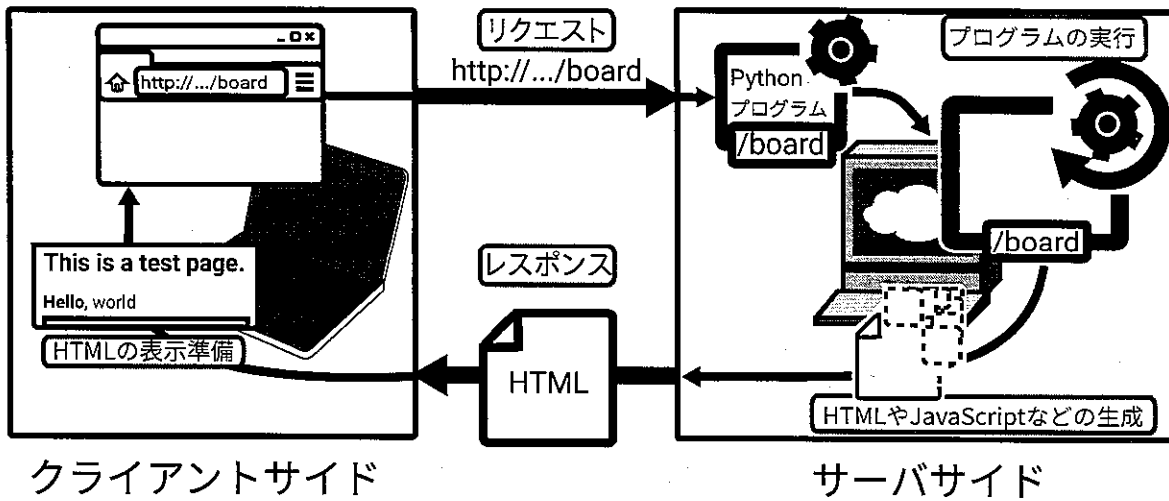


図2.6: サーバサイドプログラムの動作概要

2.3.1 HTTP リクエスト

HTTP リクエストの例を図2.7に示す。

```

1 POST /board HTTP/1.1
2 Host: 192.168.11.28
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko
  /20100101 Firefox/60.0
4 Accept: text/html,application/xhtml+xml,application/xml;q
  =0.9,*/*;q=0.8
5 Accept-Language: ja,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Connection: keep-alive
8 Upgrade-Insecure-Requests: 1
9 Pragma: no-cache
10 Cache-Control: no-cache
11
12 name=Taro&comment=Hoge
  
```

図2.7: POST リクエストの例

1行目にはHTTPリクエストメソッド、アクセスURL、そしてHTTPのバージョンを明記する。2行目以降にHTTPリクエストに関わる様々な情報を追加して、HTTPリクエストが作られる。12行目は、**HTTP**リクエストボディと呼ばれ、基本的には空行であるが、次で説明

する **POST** メソッドを使う際は送信データが追加される。この HTTP リクエストを Web サーバに渡すことで、サーバがそれに応じた処理を行い HTTP レスポンスを返却する。

2.3.2 HTTP リクエストメソッド

HTTP リクエストメソッドは、接続する URL 先に対して実行したい処理 (アクション) のことである。最もよく使うメソッドとして、GET と POST がある。

GET: 接続する URL 先を取得する。普段ブラウザから Web ページを見る際には GET メソッドが使われる。これは、URL バーに URL を打ち込み Web ページを表示させる、Web ページ上のハイパーリンクをクリックして別のページへ移動する、などが含まれる。

POST: 接続する URL 先にデータを送信する。POST メソッドは、ブラウザから Web サーバにデータを送信する際に使われることが多い。たとえば、ショッピングサイトの会員登録フォームにキーボードから情報を入力して最後に送信ボタンをクリックしたときのサーバへの情報の送信、などで使われる。

メッセージボードプログラムで、HTTP リクエストメソッドを表示するプログラムを、図2.8に示す。

```

1 BODY_TEMPLATE=""
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <meta charset="utf-8">
6     <title>メッセージボード</title>
7   </head>
8   <body>
9     <h1>メッセージボード</h1>
10    <hr>
11
12    {0}
13
14    <form action="/board" method="post">
15      <div>
16        <input type="text" name="name" value="" placeholder="名前">
17      </div>

```

```

18     <div>
19         <textarea name="comment" placeholder="メッセージを書いてください。">
20             </textarea>
21     </div>
22     <div>
23         <input type="submit" value="書き込み">
24     </div>
25 </form>
26 </body>
27 </html>
28 """
29
30 def application (environ, start_response):
31     """ WSGIで最初に呼び出される関数.
32     """
33     request_method = environ['REQUEST_METHOD']
34
35     raw_response = BODY_TEMPLATE.format(request_method)
36     response = raw_response.encode('utf-8')
37
38     start_response('200 OK', [('Content-Type', 'text/html;
39         charset=UTF-8')]))
40     return [response]

```

図2.8: /var/www/html/board.py のソースコード、_はタブを示す

図2.8の1行目で変数 `BODY_TEMPLATE` に HTTP レスポンスボディとして返す文字列のフォーマットを作成している。この文字列フォーマット `BODY_TEMPLATE` は、35行目で示すように別の文字列オブジェクトを挿入することで、HTML 文章を作成する。作成した HTML 文章は、`response` として UTF-8 文字列に変換した後、40行目でリストオブジェクトに挿入し、返却している。ここで、HTTP リクエストメソッドは、`application` 関数の引数 `environ` から取得できる。`environ` は辞書型オブジェクトであり、キーを使ってさまざまなプログラム実行時の情報を参照できる。HTTP リクエストメソッドは33行目に示すように、`REQUEST_METHOD` キーを使うことで、`environ` から文字列で取得できる。`BODY_TEMPLATE` の中身を見ると、`body` 要

素が form タグを含んでいることが分かる。HTML 文章の form 要素は、内容が Web サーバに送信する情報についてキーボードやマウス入力を受け付ける対象たち(フォーム)を指し示す。form 要素の action 属性は送信先を指し示し、method 属性は HTTP リクエストメソッドを示す。この例では、action 属性が空で、method 属性は POST であるから、この HTML 文章自身と同じ URL に対して POST リクエストを送るフォームを表示することになる。

フォームを形作る要素はいくつか存在するがここではその中でも、input 要素と、textarea 要素を利用している。ここでは input 要素に対して、type 属性が text と submit を使っており、type 属性で text を指定することで入力可能なテキストボックス、submit を指定することでフォーム送信用のボタンに整形される。図2.9で示すような type="submit" の書き込みボタンは、クリックすることでフォーム内容を送信するイベントが割り当てられる。

フォームを使って POST リクエストで送信する際には、フォームを形作る各要素が持つ [name 属性の値(キー)]=[その要素の入力内容(値)] という形で =(イコール) を使ってキーと入力値がまとめられ、さらに &(アンパサンド) を使って複数フォームの入力値がまとめられる。まとめられた各フォームの値は type が submit 属性のフォーム送信ボタンを押すことで、図2.7で示すような HTTP リクエストが form 要素で指定した action 属性で指し示すアドレスに送信される。図2.9で指し示す、図2.8のフォーム要素においては、action 属性で指し示されている自分自身の URL(/board)、に対して送信がなされる。

問題 2.2.

図2.8で示すプログラムを作成し、ソースコードを報告しなさい。また、スクリーンショットも報告しなさい。

レポート課題 2.1.

HTTP リクエストメソッドについて他に存在するものについて調査し、どのような役割が説明しなさい。

ソースコード (HTML)

```
14 <form action="/board" method="post">
15   <div>
16     <input type="text" name="name" value=""
17       placeholder="名前">
18   </div>
19   <div>
20     <textarea name="comment" placeholder="メッセージを
21       書いてください。">
22   </div>
23   <div>
24     <input type="submit" value="書き込み">
25   </div>
</form>
```



ブラウザでの描画

入力動作

- (1) name テキストボックスと comment テキストボックスに入力
- (2) 書き込みボタンのクリック



リクエストの生成

```
POST /board HTTP/1.1
Host: 192.168.11.28
:
Cache-Control: no-cache

name=Taro&comment=Hoge
```

↓ /board に送信

図2.9: フォームを使った POST リクエストを作るまでの流れ

新たに利用する Python の機能 - ヒアドキュメント

"""(ダブルクオート 3 つ) もしくは'''(シングルクオート 3 つ) で文字列を囲うことで、入力された文章そのものの文字列オブジェクトを作ることができる。このような、文字列オブジェクト作成方法をヒアドキュメントと呼ぶ。図2.10に例を示す。この例では、6 行目の変数 tanka の表示結果から、囲った文字列をそのまま文字列オブジェクトにできていることが分かる。

```

1 user@hostname:~> python3 -i
2 >>> tanka = '''これやこの
3 ... 往くくもかへるも別れては
4 ... 知るも知らぬも逢坂の関'''
5 >>> tanka
6 'これやこの\n往くくもかへるも別れては\n知るも知らぬも
   逢坂の関'
```

図2.10: ヒアドキュメントの定義例

新たに利用する Python の機能 - 辞書型

Python には、連想配列が辞書型 (**dict** 型) として用意されている。連想配列とは、キーワード (キー) と値が紐付けられたデータ構造であり、キーワードをインデックスとして値を取り出すことができる。C 言語などの構造体と異なり、あらかじめオブジェクトが持つことができる型とその変数の名前を定義しておく必要はなく、自由に~~定義~~追加が可能である。図2.11に利用例を示す。

```
1 user@hostname:~> python3 -i
2 >>> m = {} #辞書型オブジェクトの作成
3 >>> m['name'] = '果物と野菜' #文字列型の追加
4 >>> m['f'] = {'a':'r', 'o':'o'} #辞書型の追加
5 >>> m['野菜'] = ['レタス'] #リスト型オブジェクトの追加
6 >>> m
7 {'name': '果物と野菜', 'f': {'a':'r', 'o':'o'}, '野菜': ['レタス']}
8 >>> m['f'] #fキーに割り当てられた値の取得
9 {'a':'r', 'o':'o'}
10 >>> m['野菜'][0] #'野菜'キーのリストの1番目の要素
11 'レタス'
12 >>> m['f'] = 2 #fキーに新たに整数型オブジェクトの割当
13 >>> m
14 {'name': '果物と野菜', 'f': 2, '野菜': ['レタス']}
15 >>> len(m) # mに含まれているキーの個数を取得
16 3
```

図2.11: 辞書型オブジェクトからの値の取り出し例

2.4 HTTP レスポンス

HTTP レスポンスの例を図2.12に示す。

```
1 HTTP/1.1 200 OK
2 Date: Thu, 27 Sep 2018 13:16:46 GMT
3 Server: Apache/2.2.34 (Unix)
4 Last-Modified: Mon, 14 Jul 2014 06:15:16 GMT
5 ETag: "81d21a-5bc-4fe2136f14500"
6 Accept-Ranges: bytes
7 Content-Length: 1468
8 Keep-Alive: timeout=5, max=95
9 Connection: Keep-Alive
```

```

10 Content-Type: text/html
11
12 <!DOCTYPE html>
13 (more data)

```

図2.12: レスポンスの例

1行目から10行目までを **HTTP レスポンスヘッダ** と呼ぶ。12行目以降を **HTTP レスポンスボディ** と呼ぶ。たとえば、HTTP リクエストメソッドが GET であれば、HTTP レスポンスボディには HTML 文章そのものが埋め込まれる。

2.4.1 HTTP レスポンス状態コード

HTTP レスポンス状態コード (ステータスコード) とは、HTTP リクエストに対してサーバが処理した結果どうなったかを示す3桁の数値のことである。各コードには、説明句と呼ばれるコードを説明する言葉があわせて用意されている。特によく利用する状態コードを以下に示す。

200 OK: リクエストが成功したことを示す。

403 Forbidden: 認証されていないなどの理由で、サーバがレスポンスの返答を拒否していることを示す。

404 Not Found: リクエストされた URL にリソース (ファイル・プログラム) が発見できないことを示す。

500 Internal Server Error: サーバでリクエストの処理方法わからないことを示す。

WSGI では、図2.8中の31行目で示すように、状態コードと説明句をスペースで連結したものをレスポンスヘッダを作成する関数に渡すことで、指定できる。

レポート課題 2.2.

レスポンス状態コードを3つ調べて、どのような番号でどのような時に返却されるのか報告しなさい。

レポート課題 2.3.

演習で作成しているプログラムに **POST** と **GET** 以外のメソッドでリクエストした場合、どの **HTTP レスポンス状態コード** を返すべきか、説明しなさい。

2.5 掲示板プログラムの作成

図2.8に追加するサンプルコードを図2.13に示す。

```

1 from urllib.parse import parse_qs
2 import os.path

```



```
3 import json
4
5 MESSAGE_PATH = '/var/www/html/data/messages.json'
6
7 def write_messages(messages):
8     """ 渡された掲示板書き込み情報オブジェクトをJSONファイル
        としてMESSAGE_PATHに書き込む.
9     """
10
11     fp = open(MESSAGE_PATH, 'w')
12     json.dump(messages, fp) #messagesをJSONとして書き込む
13
14     fp.close()
15
16 def read_messages():
17     """ MESSAGE_PATHから掲示板書き込み情報を読み込んで、オ
        ブジェクトに変換し返却する.
18     """
19
20     fp = open(MESSAGE_PATH, 'r')
21     messages = json.load(fp) #messagesとしてJSONを読み込む
22
23     fp.close()
24     return messages
25
26 def parse_get_request():
27     """ GETメソッドを使ったリクエストに対して、掲示板書き込
        み情報を返却する.
28     """
29     if os.path.exists(MESSAGE_PATH) is False:
30         messages = []
31         write_messages(messages)
32
```

```

33  ____else:
34  ____    messages = read_messages()
35
36  ____return messages
37
38  def parse_post_request(envIRON):
39  ____    """ POSTメソッドを使ったリクエストに対して、リクエスト
        ボディから、掲示板書き込み情報を更新して、返却する.
40  ____    """
41  ____    # HTTPリクエストボディを読むためのファイルストリーム
42  ____    body_p = environ['wsgi.input'].read()
43  ____    # HTTPリクエストボディからクエリの取り出し。
44  ____    q = parse_qs(body_p.decode())
45
46  ____    messages = read_messages()
47
48  ____    if len(q) != 0:
49  ____        name = q['name'][0]
50  ____        text = q['comment'][0]
51
52  ____        messages.append({'name': name, 'text': text})
53
54  ____        write_messages(messages)
55
56  ____    return messages
57
58  def create_response_body(messages):
59  ____    """ 掲示板書き込み情報から、レスポンスを作成する.
60  ____    """
61
62  ____    insert_text = ''
63  ____    LINE_FORMAT = '<p>名前: <strong>{0}</strong></p>\n<p>
        >{1}</p><hr>'

```

```
64
65 ____for m in messages:
66 ____    insert_text = insert_text + LINE_FORMAT.format(m['
        name'], m['text'])
67
68 ____return insert_text
69
70 def application (environ, start_response):
71 ____    """ WSGIで最初に呼び出される関数.
72 ____    """
73
74 ____    request_method = environ['REQUEST_METHOD']
75
76 ____    if request_method == 'GET':
77 ____        messages = parse_get_request()
78
79 ____    elif request_method == 'POST':
80 ____        messages = parse_post_request(environ)
81
82 ____    insert_text = create_response_body(messages)
83
84 ____    raw_response = BODY_TEMPLATE.format(insert_text)
85 ____    response = raw_response.encode('utf-8')
86
87 ____    start_response('200 OK', [('Content-Type', 'text/html;
        charset=UTF-8')]))
88
89 ____    return [response]
```

図2.13: 図2.8へ追加するソースコード、なお **application** 関数は書き換えなければならない。__はタブを示す。

プログラムを実行した際のエラーは、Web サーバのログに記録される。今回は、Apache2 の WSGI で実行しているため、`/var/log/apache2/error.log` にエラーが書き込まれる。

2.5.1 JSON

JSON(JavaScript Object Notation) とは、Web アプリケーションでよく使われるファイル形式である。簡易な記述による可読性の高さと、オブジェクトへの相互変換の容易さから、近年よく使われるデータ保存のためのファイルフォーマットである。図2.13に例を示す。

```
1 [{"text": "やあ", "name": "a"}, {"text": "どうも", "name": "b"}]
```

図2.14: 図2.13に送信されたリクエストを保存した **JSON(messages.json)** の例

図2.13のプログラムでは、プログラムが受け取った POST リクエストボディに含まれるフォームの入力値を、`write_messages` 関数で JSON ファイルとして保存している。また、`read_messages` 関数で JSON ファイルを読み込んでいる。

、新たに利用する Python の機能 - ファイル作成と読み込み

Python でファイルの作成・読み込みを行うためには `open` 関数を使う。

open 関数 ファイルパスと、ファイルの開き方を指定することで、ファイルオブジェクトを作り出す。C 言語の `fopen` と `fclose` と同様に、書き込み(読み込み)が終わったファイルオブジェクトは必ず `close` メソッドで破棄する必要がある。

```
1 user@hostname:~> python3 -i
2 >>> fpw = open('./text', 'w') #書き込みモードで開く
3 >>> fpw.write('aiueo') #ファイルに文字列の書込
4 >>> fpw.close() #ファイル書き込みの終了
5 >>> fpr = open('./text', 'r') #読み込みモードで開く
6 >>> fpr.read() #./textからテキストデータの読込
7 'aiueo'
8 >>> fpr.close() #ファイル読込の終了
```

2.5.2 ファイルの書き込み・読み込み・プログラムの実行権限について

Web サーバ上で、外部からのアクセスに対してファイルの書き込みや読み込み、プログラムを実行するためには、ファイル権限を適切に設定する必要がある。図2.13のプログラムでは、`/var/www/html/data` に対して JSON ファイルの読み込み・書き込みを行っている。よって、`data` フォルダに対して、書き込み可能権限を追加しておく必要がある。`data` フォルダの作成と書き込み権限の追加は図2.15で行う。

```
user@hostname ~$ mkdir /var/www/html/data
```

```
user@hostname ~$ chmod 777 /var/www/html/data
```

図2.15: data フォルダの作成と、読み込み権限と書き込み権限実行権限の追加

問題 2.3.

図2.8に、図2.13を追加して、プログラムを作成しソースコードを報告しなさい。またスクリーンショットも報告しなさい。

レポート課題 2.4.

JSON 以外に Web アプリケーションなどで使われるデータフォーマットを、3種類調査して特徴をまとめなさい。

レポート課題 2.5.

今回作成したプログラムでは、受信したフォーム入力値をファイルとして保存している。しかし、一般的な Web アプリケーションでは、データベースを使うことが多い。データベースを Web アプリケーションで使う利点を答えなさい。