# 1 電力系統モデルの数値シミュレーション

チャプター概要

## 1.1 電力系統モデルの時間応答を計算するためには

### 1.1.1 潮流計算の実装法

### 1.1.2 数値シミュレータの実装法

MATLAB を用いて電力システムの時間応答をシミュレーションするプログラムを実装する方法について説明する。電力系統の時間応答のシミュレーションは，機器の動特性を表す微分方程式と，電力潮流の代数方程式を連立した微分代数方程式を解くことによって行われる。そこで，まずは簡単な微分代数方程式を例に，その解き方を見てみよう。

---

**例 1.1** (簡単な例題)　図図 1.1 の簡単な電気回路において，$R = L = C = E = 1$ のときのシミュレーションを行うことを考える。この回路の動的な要素はコイル $L$ とコンデンサ $C$ であり，その微分方程式は，

$$L\dot{i}_L = v_L \tag{1.1a}$$

$$C\dot{v}_C = i_C \tag{1.1b}$$

である。ただし，初期値は $i_L(0) = 0$，$v_C(0) = 0$ とする。また，オームの法則とキルヒホッフの法則から，

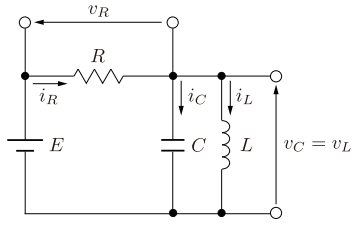



**図 1.1 微分代数方程式の例題：LC 並列回路**

$$v_R = Ri_R \tag{1.2a}$$

$$i_R = i_L + i_C \tag{1.2b}$$

$$v_L = v_C \tag{1.2c}$$

$$E = v_C + v_R \tag{1.2d}$$

という代数方程式が成り立つ。

このシステムは微分方程式と代数方程式が連立された微分代数方程式である。式 1.2 の代数方程式を用いて文字を消去すると，等価な常微分方程式を得ることができる。これはクロン縮約に対応し，得られる常微分方程式は，

$$\dot{i}_L = \frac{1}{L} v_C \tag{1.3a}$$

$$\dot{v}_C = \frac{1}{RC}(E - v_C) - \frac{1}{C} i_L \tag{1.3b}$$

となる。まずは，この常微分方程式を解くプログラムを書いてみよう。常備分方程式に対する MATLAB のソルバとしては ode45 を用いるのが一般的である。ode45 では，常微分方程式 $\dot{x} = f(t, x)$ の関数 $f(t, x)$ を実装することにより常微分方程式を解くことができる。式 (1.3) の右辺を実装すると，プログラム 1-1 のようになる。

**プログラム 1-1** (func_RLC_ode.m)

```
1  function dx = func_RLC_ode(x, R, C, L, E)
2  vC = x(2);
```

```
3
4  diL = vC/L;
5  dvC = (E-vC)/R/C - iL/C;
6
7  dx = [diL; dvC];
8
9  end
```

また，これを用いて ode45 を実行し，常微分方程式を解くプログラムはプログラム 1-2 となる。

**プログラム 1-2** (main_RLC_ode.m)

```
1  R = 1;
2  L = 1;
3  C = 1;
4  E = 1;
5
6  func = @(t, x) func_rlc_ode(x, R, C, L, E);
7  x0 = [1; 1];
8  tspan = [0 30];
9
10 [t, x] = ode45(func, tspan, x0);
```

このプログラムにおいて，出力された変数 x は，$i_L$ と $v_C$ の時系列を並べたベクトルである。したがって，$i_C$ や $i_R$ などの他の変数については式 (1.2) の代数方程式を用いて改めて計算する必要があることに注意しよう。

つぎに，クロン縮約を行わず，微分代数方程式を直接解くことを考える。これには，物理的な代数方程式をそのまま用いることとができ，複雑なシステムになっても記述が簡単である利点がある。MATLAB で微分代数方程式を解くことができるコマンドのひとつに ode15s がある。ode15s は

$$M\dot{x} = f(x)$$

という形で記述された微分代数方程式を対象とする。式 1.1，1.2 をこの形にあてはめると，

$$\begin{bmatrix} 0 & L & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & C \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{i}_R \\ \dot{i}_L \\ \dot{i}_C \\ \dot{v}_R \\ \dot{v}_L \\ \dot{v}_C \end{bmatrix} = \begin{bmatrix} v_L \\ i_C \\ v_R - Ri_R \\ i_R - i_L - i_C \\ v_L - v_C \\ E - v_C - v_R \end{bmatrix}$$

となる。ここで，右辺の 3 番目以降の要素の順番は任意であることに注意しよう。この右辺を実装するとプログラム 1-4 となる。

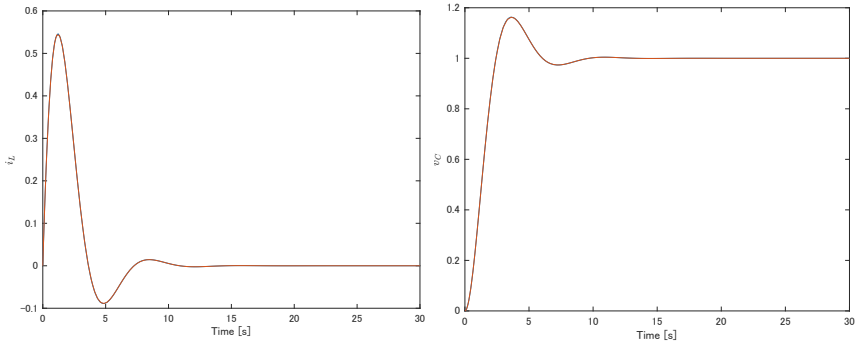
**プログラム 1-3** (func_RLC_DAE.m)

```
function dx = func_rlc_dae(x, R, C, L, E)

vR = x(1);
vL = x(2);
vC = x(3);
iR = x(4);
iL = x(5);
iC = x(6);

dvC = iC;
diL = vL;

con1 = vC-vL;
con2 = E-vC-vR;
con3 = iR-(iC+iL);
con4 = vR-iR*R;

dx = [dvC; diL; con1; con2; con3; con4];
end
```

この関数を用いると，プログラム 1-4 のように微分代数方程式を解くことができる。

**プログラム 1-4** (main_RLC_DAE.m)

```
R = 1;
C = 1;
L = 1;
E = 1;

```




(a) $i_L$ の時間応答　　(b) $v_C$ の時間応答

**図 1.2　LC 並列回路の時間応答**

```
 6  M = zeros(6, 6);
 7  M(1, 2) = L;
 8  M(2, 6) = C;
 9
10  x0 = zeros(6, 1);
11  tspan = [0 30];
12
13  options = odeset('Mass', M);
14  func = @(t, x) func_RLC_DAE(x, R, C, L, E);
15  [t, y] = ode15s(func, tspan, x0, options);
```

このプログラムにおいて，10 行目で初期状態をすべて 0 ととっているが，意味を持つのは動的な要素の値のみ，すなわち，2 番目と 6 番目の要素のみであり，他の値については代数方程式を満たす値が ode15s によって探索される。また，解 x は，$v_R, v_L, v_C, i_R, i_L, i_C$ の時系列を並べた行列であり，常微分方程式に変換した場合と異なり，すべての物理量を含んでいる。

常微分方程式として解いた場合と微分代数方程式として解いた場合の $i_L$ と $v_C$ を図図 1.2 に示す。それぞれの図には ode45 と ode15s の解の 2 本の線が表示してあるが，当然ながらこれらは完全に重なっている。

**例 1.2** (電力系統のシミュレーション) 図**??**の電力系統のシミュレーションを行うことを考える。