



PLECS Tutorial

Running PLECS Standalone Simulations from Matlab

MatlabからPLECS Standaloneシミュレーションを実行

- JSON-RPCインターフェースを使用 -

Tutorial Version 1.0

1 はじめに

このチュートリアルでは、Matlab®とJSON-RPCを使用してPLECS シミュレーションを実行する方法について説明します。このチュートリアルに必要なものは以下のとおりです:

- 正常に動作するMatlab®インストール環境
- 正常に動作するPLECS Standaloneインストール環境
- PleximのGitHubページからJSON-RPCクライアントの最新バージョンをダウンロードするためのインターネット接続環境

チュートリアルは3つのセクションに分かれています: JSON-RPCクライアントのインストール、PLECS Standaloneの設定、Matlab®でのシミュレーション スクリプトの記述。

始める前に: 作業ディレクトリにstandalone_from_matlab.plecs、simulation_script_1.m、simulation_script_2.m、simulation_script_3.mのファイルがあることを確認してください。これらのファイルは、このPDFと共に<https://adv-auto.co.jp/products/plexim/tutorial.html>から入手できます。

2 JSON-RPCクライアントのインストール



あなたのタスク:

- 1 <https://github.com/plexim/matlab-jsonrpc>にアクセスし、"Code"をクリックしてから"Download ZIP"をクリックして、JSON-RPCクライアントの最新バージョンをダウンロードしてください。
- 2 matlab-jsonrpc-main.zipファイルを展開します。
- 3 Matlabを開き、JSON-RPCクライアントを含むフォルダmatlab-jsonrpc-mainをMatlab®検索パスに追加します(これを行う方法については、この[リンク](#)を参照してください)。

3 PLECS Standaloneのセットアップ



あなたのタスク:

- 1 PLECS Standalone版を開きます。
- 2 PLECS設定でRPCインターフェースを有効にします: **ファイル -> PLECS設定...**をクリックし、**Genral**タブの**RCPインターフェイスポート番号**を有効にし、1080と入力します。MacOSでは、**PLECS -> PLECS設定**から行います。
- 3 モデルstandalone_from_matlab.plecsを開きます。

このモデルは、アナログ制御を備えた降圧コンバータを構成しています。モデルの詳細については、デモモデル[降圧コンバータのパラメータスweep](#)の説明をご覧ください。

4 シミュレーションスクリプト

4.1 単一シミュレーション



あなたのタスク:

- 1 Matlab®を開き、standalone_from_matlab.plecsファイルと同じフォルダに、simulation_script.mという新しいファイルを作成します。

- 2 mファイルをエディタで、次のコードを記述します:

```
proxy = jsonrpc('http://localhost:1080', 'Timeout', 10);
```

これにより、接続タイムアウトが10 秒の新しいJSON-RPCのProxyオブジェクトが作成されます。

- 3 次のコードでPLECSモデルをロードします:

```
path = pwd;
model_name = 'standalone_from_matlab';
proxy.plecs.load([path '/' model_name '.plecs']);
```

コマンドpwdは、シミュレーション対象のPLECSモデルが格納されている現在の作業ディレクトリのパスを出力します。

- 4 シミュレーション構造体を作成し、シミュレーションを実行します

```
simStruct = struct('ModelVars', struct('varL', 50e-6));
results = proxy.plecs.simulate(model_name, simStruct);
```

- 5 Matlab®でスクリプトを実行します。



この段階では、シミュレーションスクリプトはサンプルファイルのSimulation_script_1.mのようになるはずです。

4.2 並列シミュレーション

このタスクでは、並列シミュレーションを使用してパラメータスイープを設定します。シミュレーションを実行する概念は単一シミュレーションの場合と同じですが、シミュレーションパラメータの1セットのみではなく、個々のシミュレーションごとにパラメータセットを含むセル配列を渡します。



あなたのタスク:

- 1 次の内容を含む、simulation_script_2.mという新しいファイルを作成します。

```
proxy = jsonrpc('http://localhost:1080', 'Timeout', 10);
path = pwd;
model_name = 'standalone_from_matlab';
proxy.plecs.load([path '/' model_name '.plecs']);
simStruct = struct('ModelVars', struct('varL', 50e-6));
```

これは、plecs.simulateコマンドがない以外は、前のタスクと同じコードです。

- 2 後ですべての並列シミュレーション実行のトレースを追加できるように、スコープ内のすべてのトレースをクリアします。

```
proxy.plecs.scope([model_name '/Scope'], 'ClearTraces');
```

- 3 ここで、個々のシミュレーションのパラメータを含むセル配列を作成します。

```
% Set value for L1 to be swept
inductorValues = 40:20:220; % in uH

% Allocate memory for cell array
simStructs = cell(size(inductorValues));

% Initialize simStruct as cell array with all values for L1
for ix = 1:length(inductorValues)
    simStructs{ix}.ModelVars.varL = inductorValues(ix) * 1e-6;
    simStructs{ix}.Name = ['L=' mat2str(inductorValues(ix)) 'uH'];
end
```

forループでは、セル配列のすべてのメンバーにModelVars構造体と名前が割り当てられます。この名前は、後でシミュレーション結果を識別するのに役立ちます。たとえば、名前はスコープ内のトレースにラベルを付けるために使用します。

- 4 ここで、plecs.simulateコマンドにセル配列を使用します:

```
results = proxy.plecs.simulate(model_name, simStructs);
```

- 5 最後のステップでは、plot関数を使用してシミュレーション結果の1つ(指標5)をプロットします:

```
plot(results(5).Time, results(5).Values); title(simStructs{5}.Name)
```

- 6 Matlab®でスクリプトを実行します。

スクリプトはすべてのシミュレーションを並列に実行しますが、スコープと Matlab®のプロットウィンドウにはシミュレーション結果の1つだけを表示します。この時点では、すべてのシミュレーションが並行して実行され、どのシミュレーションが最初に終了するかが明確ではないため、結果は必ずしも同じではないことに注意してください。

次のステップでは、トレースを保持し、すべてのシミュレーションの結果を後処理するコールバック関数を作成します。

 この段階では、シミュレーションスクリプトはサンプルファイルのsimulation_script_2.mのようになるはずです。

4.3 コールバック関数の追加

自動パラメータスイープのコンテキストで並列シミュレーションの潜在能力を最大限に活用するために、plecs.simulateコマンドの引数にコールバック関数を追加します。

あなたのタスク:

- 1 前のステップsimulation_script_2.mからのシミュレーションの記述を続けます。plecs.simulateコマンドの前に次のコードを追加します:

```
callback = sprintf([ ...
    'if ischar(result)\n' ...
    '    disp(["Simulation errors can be asserted in this message."]); \n' ...
    'else\n' ...
    '    plecs('scope'', './Scope'', 'HoldTrace'', name); \n' ...
    '    result = max(result.Values(1,:)); \n' ...
    'end\n' ...
]);
```

このコードは、シミュレーションの実行ごとに PLECS Standaloneで処理できるOctave 言語のコールバック関数を記述します。これは、各シミュレーション実行からの特定の主要な数値を解析し、スニップの最後にパラメトリックプロットを追加するのに便利なツールです。ここでは、シミュレーション後に最大電流値のみを返します。コールバック関数はスコープ内にトレースも保持するため、スクリプトの終了後にすべての結果をPLECSスコープ内で検査することができます。

- 2 `plecs.simulate` コマンドの呼び出しを次のように変更します:

```
results = proxy.plecs.simulate(model_name, simStructs, callback);
```

- 3 ここで、最大電流とインダクタンス値のパラメトリックプロットを追加します:

```
plot(inductorValues, results, '-*');  
title('Maximum current in A vs. Inductor Values in uH')
```

- 4 Matlab®でスクリプトを実行します。

 この段階では、シミュレーションスクリプトはサンプルファイルのsimulation_script_3.mのようになるはずです。

5 まとめ

このチュートリアルでは、Matlab®から PLECS Standaloneシミュレーションを実行する方法を学習しました。まず、シミュレーションスクリプトにJSON-RPCクライアントをインストールして構成し、その後、JSON-RPCインタフェースを介してStandaloneシミュレーションを呼び出す必要があります。並列シミュレーションの強力な機能については、このチュートリアルの最後で詳しく説明しました。この機能はPLECS Standalone版でのみ利用可能です。

改訂履歴:

Tutorial Version 1.0 初版



Pleximへの連絡方法:

☎ +41 44 533 51 00

Phone

+41 44 533 51 01

Fax

✉ Plexim GmbH

Mail

Technoparkstrasse 1

8005 Zurich

Switzerland

@ info@plexim.com

Email

<http://www.plexim.com>

Web



アドバンオートメーションへの連絡方法:

☎ +81 3 5282 7047

Phone

✉ ADVAN AUTOMATION CO.,LTD Mail

1-9-5 Uchikanda, Chiyoda-ku

Tokyo, 101-0047

Japan

@ info-advan@adv-auto.co.jp

Email

<https://adv-auto.co.jp/>

Web

PLECS Tutorial

© 2002–2023 by Plexim GmbH

このマニュアルで記載されているソフトウェアPLECSは、ライセンス契約に基づいて提供されています。ソフトウェアは、ライセンス契約の条件の下でのみ使用またはコピーできます。Plexim GmbHの事前の書面による同意なしに、このマニュアルのいかなる部分も、いかなる形式でもコピーまたは複製することはできません。

PLECSはPlexim GmbHの登録商標です。MATLAB、Simulink、およびSimulink Coderは、The MathWorks, Inc.の登録商標です。その他の製品名またはブランド名は、それぞれの所有者の商標または登録商標です。