

# 一种基于 Python 的 PsCAD 仿真数据处理方案

## 起因

在使用 PsCad 进行仿真的过程中，常常需要记录数据。PsCad 会默认将仿真数据存储为 .out 文件和 .inf 文件。然而，这样的数据文件并不能直接导入到 Python 或者 Matlab 中，这就对进一步的数据分析处理造成困难。

虽然在 PsCad 中也有一些数学处理的方法，但是终究不能使人满意。因此，尝试通过 Python 脚本将数据文件进行转换。以下介绍一种基于 Python 的 PsCad 仿真数据处理方法。

## 处理思路

### 准备工作

本脚本依赖 PsCad 官方推出的自动化库 `mhrc.automation`，这个自动化库可以实现 Python 脚本自动控制 PsCad 进行仿真的功能。（在本文中对该库的应用只是冰山一角，其余功能有待探究）

安装此库的方法，如果有时间，会再写一篇文章。这里只给出官方介绍，不做过多阐述。[官方文档](#)

### PsCad 数据文件的特点

PsCad 如何设置数据的输出，不是本文重点内容，假设已经得到了 out 和 inf 文件

打开一份 out 文件 `Constant_source_short_circuit_01.out`，如下：

```
Constant_source_short_circuit_01.out - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

0.000000000000000000 -0.50917741029383E-18 -0.12152330920613E-01 0.12152330920613E-01
0.250000000000000000E-03 0.44054691046968E-02 -0.10901293628419 0.10460746717949
0.500000000000000000E-03 0.17591069924353E-01 -0.21203492410052 0.19444385417617
0.750000000000000000E-03 0.39469987669964E-01 -0.31814953759878 0.27867954992882
0.100000000000000000E-02 0.69901813824005E-01 -0.42669978666634 0.35679797284233
0.125000000000000000E-02 0.10869341180347 -0.53701366641645 0.42832025461298
0.150000000000000000E-02 0.15560010770894 -0.64840830032077 0.49280819261183
0.175000000000000000E-02 0.21032719881841 -0.76019415036914 0.54986695155073
```

其中第一列数据为时间，其余列数为数据，对应仿真中的 channel。需要注意，一个 out 文件最多只能存储10列数据，如果记录的 channel 较多，将会分成几个文件，并在文件名中以 `_01`，`_02` 等进行区分。这一点是编程的关键，将在下面进行更为系统的叙述。

显然，我们并不知道每一列数据代表的是哪一个 channel，这就需要 inf 文件。打开对应的 `Constant_source_short_circuit.inf` 文件，如下：

```
Constant_source_short_circuit.inf - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

PGB(1)      Output Desc="Ik:1" Group="Main" Max=2.0 Min=-2.0 Units=""
PGB(2)      Output Desc="Ik:2" Group="Main" Max=2.0 Min=-2.0 Units=""
PGB(3)      Output Desc="Ik:3" Group="Main" Max=2.0 Min=-2.0 Units=""
```

其中，PGB(1)指的是第一列数据（考虑到时间，准确的说是第2列），它代表 Ik:1 这项数据，其他内容是关于这项数据的单位等其他信息。

这样，我们只要从 inf 中读入数据，利用正则提取相关信息，再匹配到 out 文件中，即可实现数据识别，最后写入 csv 文件即可。

这里需要指出，最初我尝试了自己编写程序进行匹配，并最终实现了这一功能，但是了解到已有 PsCad 提供的自动化接口后，还是感觉悔恨不已，这充分说明深入搜集信息对避免重复造轮子是多么的重要~

## PsCad数据文件的命名规则

在进行正式处理前，了解 PsCad 数据文件的命名规则是有必要的。由于仿真时可能是单次仿真，也有可能是采用“Multiple Run”组件进行多次仿真，因此输出的文件命名是不同的。分为如下几种情况：

1. 单次仿真且记录 Channel 数较少

`<basename>_01.out` `<basename>.inf`

2. 单次仿真且记录 Channel 数较多

`<basename>_0#.out` `<basename>.inf`，其中，# 代表第几个数据文件，一个数据文件最多存储 10 个 channel。

3. 多次仿真

`<basename>_r0000?_0#.out` `<basename>_0000?.inf`，其中，? 表示第几次仿真，# 表示第几个数据文件。

请注意，上面的 `<basename>` 通常是仿真项目名，这一点可以在 PsCad 的设置中进行设定。

同时，本程序是基于这样的命名规则进行处理的，鉴于本人水平有限，不排除出现其他可能性，如有，欢迎对程序进行完善~ GitHub 地址将在文末给出。

## 程序使用

在编程时，考虑到如下事实：不可能在每个仿真文件夹内都 copy 一份 Python 脚本。因此，利用命令行工具，只提供路径等必要参数就能实现功能，显得十分必要。

因此，使用程序时，打开 cmd 命令行或 Anaconda Prompt（如果你进行了环境管理，推荐使用后者）输入：

```
1 | python <PyOut.py 文件绝对路径> -i=..... -o=..... -b=.....
```

其中，各项参数说明如下：

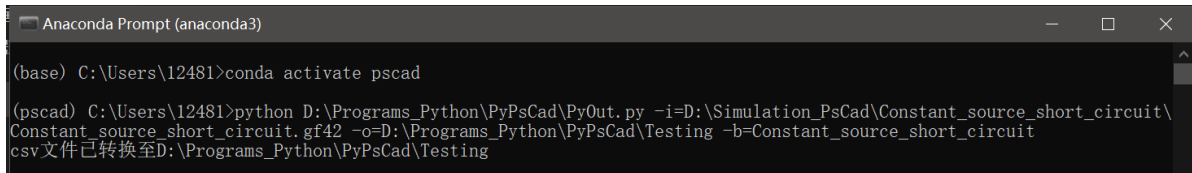
1. `-i` 或 `--input`：仿真数据文件存储路径，例如：  
`D:\Constant_source_short_circuit\Constant_source_short_circuit.gf42`，其注意，务必是以 `.gf42` 或 `.gf46` 等结尾的文件夹。
2. `-o` 或 `--output`：转换后的目标路径，请务必确保这一路径不存在（这是由于 PsCad 的自动库中采用函数无法更改，所以导致的不便）
3. `-b` 或 `--basename`：仿真的项目名，其值在“PsCad数据文件的命名规则”中已有涉及。

例如：

1. 将PyOut.py文件放在D:\Programs\_Python\PyPsCad目录下
2. 仿真数据放在  
`D:\Simulation_PsCad\Constant_source_short_circuit\Constant_source_short_circuit.gf42`  
路径下

3. 期望将转换后的数据放在D:\Programs\_Python\PyPsCad\Testing目录下

输入如下图所示：



```
Anaconda Prompt (anaconda3)

(base) C:\Users\12481>conda activate pscad

(pscad) C:\Users\12481>python D:\Programs_Python\PyPsCad\PyOut.py -i=D:\Simulation_PsCad\Constant_source_short_circuit\Constant_source_short_circuit.gf42 -o=D:\Programs_Python\PyPsCad\Testing -b=Constant_source_short_circuit
csv文件已转换至D:\Programs_Python\PyPsCad\Testing
```

出现 `csv文件已转换至.....`，即说明文件处理完成。

事实上，推荐大家采用环境管理将 `mhrc.automation` 库隔离，因为在实际使用中发现其与一些常用库的版本较难匹配，使用 `conda` 是一个不错的选择。

## 源码链接

附上代码 GitHub 地址：[GitHub](#)

如果无法打开，可以从Gitee中获取：[Gitee](#)

在这里，有必要说明一点：鉴于个人水平以及仿真经验，给出的程序未必能够普适，如果大家在应用中发现有无法使用的情况，请联系我或自行修改，欢迎大家的 fork 与 commit~ 以及关于程序的建议，也希望有大佬指出，不甚感激~