第六、七周工作总结

张泽宇

2022 年 4 月 10 日

过去两周的主要进行的工作包括有：

- **学习使用 PSCAD，我结合了张正刚师兄给的两个例子，自己搭建了之前的 10kV 接地小电阻模型，并且逐步进行了完善。**
- **学会了使用 Multiple Run 模块进行多重运行，可以自动仿真出多种情况。**
- **基于 python 尝试了 PSCAD 数据接口。**
- **基于 python 实现了从一维时间序列到二维灰度图像程序。**

  以下为详细叙述

---

1 学习使用 PSCAD

相对于 Simulink，PSCAD 更加专注于电气工程方面的仿真，操作起来也更加的方便。由于网络上相关的教程不是很多，我借助张正刚师兄给的两个例子、PSCAD 帮助文档和例程进行了学习，尝试搭建了一个 10kV 小电阻接地电磁暂态模型，并且进行了逐代完善，将故障发生、多重运行等功能加入。最终的模型如下图所示：

主电路部分如下所示：

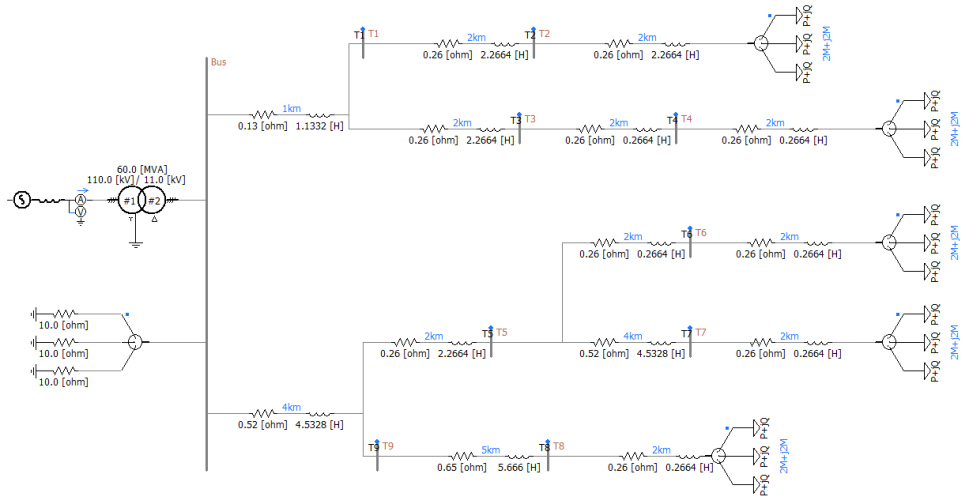

图 1: 主电路部分

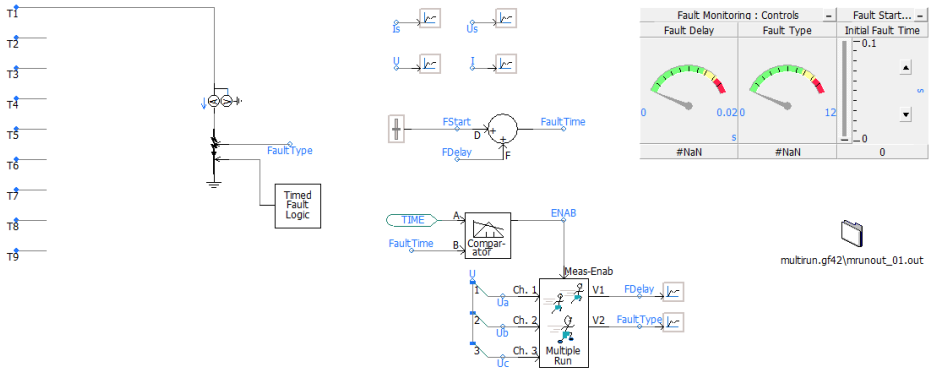

图 2: 控制部分

其中三相电压为 110kV，频率 50Hz 接降压变压器，总线上电压为 11kV，共有两条馈线，五个负载。负载有功功率 2Mw，无功功率 2Mw。输电线路用电阻和电感进行等效，每公里单位电阻为 0.13Ω，单位电感为 1.1332h（这个数据是按照学长

给的例子列出来的)。用不同的总线代表不同的故障位置，从 T1 到 T9 共有 9 个故障发生点。

电路的控制部分如下图 2所示：

控制部分的左边 T1-T9 是从主电路上引出的 Xnode 结点，当模拟不同位置故障发生的时候，只需要在控制部分将 Xnode 结点与故障发生器连接即可，无需在主电路上进行更改，这样可以便捷布线。

故障发生器由两个输入信号：FaultType 控制故障发生的种类，这个数据由 Multiple Run 元件输入；故障定时控制逻辑元件控制故障时间，其中设置用 FaultTime 变量表示故障发生时间，故障持续 0.02s 即一个周期。同时用一个万用表测量故障电流与故障电压，标记为 I 与 U。

中间靠上位置是测量的电源电压、电流和故障电压电流以 Data Label 形式引出，接输出通道。

中间部分是 FaultTime 变量的逻辑设置，由起始时间 FStart 和延迟时间 FDelay 求和，其中 FStart 可有右边的 Control Panel 进行修改，FDelay 由 Multiple Run 元件输入，这样 FaultTime 就可以表示不同时间段开始，不同滞后时间的故障发生时间。

中间靠下是多重运行的设置。Multiple Run 控制两个变量变化，FDelay 从 0 开始，以 0.01 步长增至 0.1；FaultType 为列表类型，从 1 取到 10，其中 123 代表 ABC 三相短路接地、456 代表两相短路接地、7 代表三相短路接地、8910 代表两相短路。Multiple Run 记录三个数据，为故障发生时三相电压。其 Meas-Enab 通过一个两输入比较器输入，将 FaultTime 和仿真时间进行比较，控制启用记录数据。

右边上部是控制面板，可以看出当前的 FaultTime 和 FaultType，并可以对 FStart 进行修改；下部为记录数据。

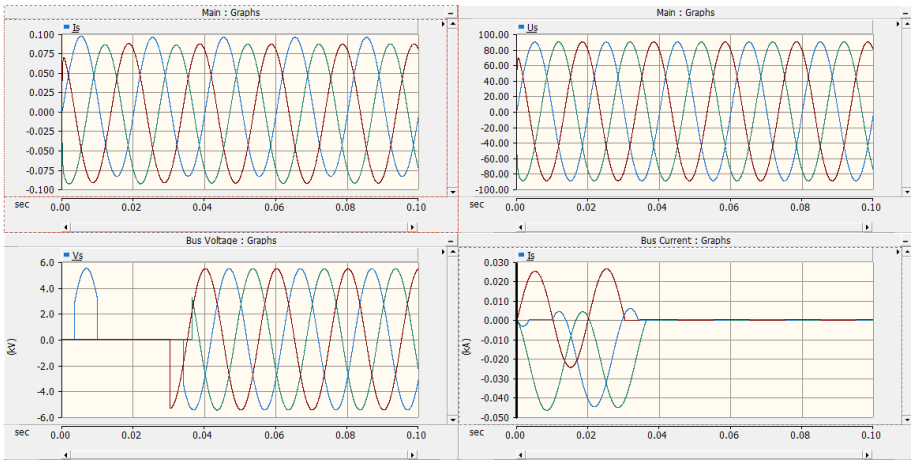

图 3: 控制部分

电路的显示部分如下图所示：

通过 Graph Pane 显示电源电压电流和故障电压电流。

可以看出，增加了 Multiple Run 之后，每次运行都可以自动多次运行仿真，每次具有不同的参数设置。这极大的增加了数据集的数量。例如，上面的设置中，只要运行一次，就可以得到 10*10=100 次仿真数据；理论上讲，增多 FDelay 可以取到的值，是可以将数据数量进一步拓展的，但是否会对电脑性能提出要求，还需要进行尝试。

但是这里还有一个问题，就是电源处三相电流不对称。有一相电流显然较高。这个地方还需要进一步排查原因。

2 Python 数据接口尝试

在 PSCAD 上，记录数据有两种途径，一种是通过仿真设置里勾选存储数据的选项，得到.out 文件；另一种是通过上面的 File Reference 存储。

第一种得到的.out 文件由数据文件和目录文件组成，如下：

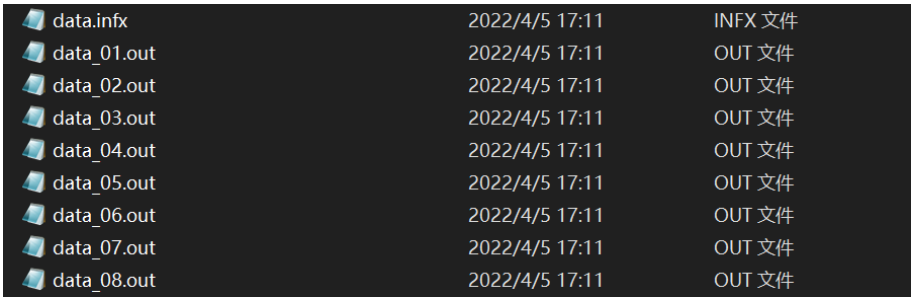

图 4: .out 文件

因此基于 python，用正则表达式从目录文件提取信息，将数据文件中的数据匹配进去，即可得到最后的数据集。

部分目录文件如下所示：

```
<Output device="EMTDC" version="2010" date="2022/04/05" time="17:11:48.000000">
<Domain name="Time" unit="s" mult="0.0" skew="0.0">
<Sample rate="100000.0" end="10000" />
</Domain>
<List classid="Analog">
<Analog name="Main(0):Is" index="0" id="1763691693:0" label="" dim="3" unit="" ...
 min="-2.0" max="2.0" />
<Analog name="Main(0):Us" index="3" id="1026695447:0" label="" dim="3" unit="" ...
 min="-2.0" max="2.0" />
<Analog name="Main(0):I1" index="6" id="862159365:0" label="" dim="3" unit="" ...
 min="-2.0" max="2.0" />
```

```
9
```

代码如下：

```
import numpy as np
import re


def read_variable_name(f_name):
    L = []
    f = open(f_name, 'r', encoding='utf-8')
    for line in f:
        reformat = re.compile(r'Main\(0\):(.*?)"', re.S)
        index = reformat.finditer(line)
        for item in index:
            L.append(item.group()[8:-1])
    f.close()
    return L


def read_variable_time(f_name):
    f = open(f_name, 'r', encoding='utf-8')
    L = []
    for line in f:
        line = line.replace("\n", '')
        line = re.split(r"[ ]+", line)
        if line[1]:
            L.append(line[1])
    else:
        continue
    f.close()
    return L


def read_variable_value(f_name):
    f = open(f_name, 'r', encoding='utf-8')
    L_A, L_B, L_C = [], [], []
    for line in f:
        line = line.replace("\n", '')
        line = re.split(r"[ ]+", line)
        if line[1]:
            L_A.append(line[2])
            L_B.append(line[3])
            L_C.append(line[4])
        else:
            continue
    L_A_mat = np.array(L_A)
    L_B_mat = np.array(L_B)
    L_C_mat = np.array(L_C)
    L = np.dstack([L_A_mat, L_B_mat, L_C_mat])
    f.close()
    return L
```

```
50
    if __name__ == '__main__':
52      path_variable_name = "data\\data.infx"
        path_variable = "data\\data_01.out"
54      variable_list = read_variable_name(path_variable_name)
        variable_time = read_variable_time(path_variable)
56      variable_value = read_variable_value(path_variable)
        print(variable_value)
58
```

这份代码还不能作为做中的版本，因为在一些普适性和代码优化方面的工作还没有进行。待完善后就可以拿来直接使用。

对于第二种数据的记录方式，目前得到的数据结果是这样的：

Multiple Run Output File

Run #	FDelay	FaultType	Out # 1	Out # 2	Out # 3
1	0.000000000	1	-3.026592362	-2.426912868	5.458454291
2	0.1000000000E-02	1	-3.026592362	-2.426912868	5.458454291
3	0.2000000000E-02	1	-3.026592362	-2.426912868	5.458454291
4	0.3000000000E-02	1	-3.026592362	-2.426912868	5.458454291
5	0.4000000000E-02	1	-3.026767868	-2.427088257	5.458278902
6	0.5000000000E-02	1	-3.027221557	-2.427541644	5.457825515
7	0.6000000000E-02	1	-3.026786706	-2.427107031	5.458260128
8	0.7000000000E-02	1	-3.027745127	-2.428064835	5.457302324
9	0.8000000000E-02	1	-3.028631534	-2.428950672	5.456416487
10	0.9000000000E-02	1	-3.029369820	-2.429688483	5.455678676
11	0.1000000000E-01	1	-3.029903270	-2.430221589	5.455145570
12	0.000000000	4	-3.028189067	-2.428510774	5.456857536
13	0.1000000000E-02	4	-3.030086800	-2.430409651	5.454959823
14	0.2000000000E-02	4	-3.030086800	-2.430409651	5.454959823
15	0.3000000000E-02	4	-3.030086800	-2.430409651	5.454959823
16	0.4000000000E-02	4	-3.028492671	-2.428816500	5.456552978
17	0.5000000000E-02	4	-3.029279594	-2.429602890	5.455766576
18	0.6000000000E-02	4	-3.030198652	-2.430521325	5.454848125
19	0.7000000000E-02	4	-3.031146478	-2.431468510	5.453900924
20	0.8000000000E-02	4	-3.032024026	-2.432345465	5.453023956

图 5: .out 文件

目前的数据形式不是我想要的数据形式。如何能够将每个时刻的三相电压输出值写入文件，还需要进行进一步的探究。

3 从一维时间序列到二维灰度图像转变

在与陈浩泳师兄交流后，因为 CNN 更多的优势是体现在图像识别上，我计划先采用将仿真得到的一维时间序列转换为二维灰度图像作为 CNN 输入的方法。所以这个星期先将时间序列到灰度图的程序进行了编写。

我从网上找到一组一维数据，数据长度为 25 万左右。随机选择 500 个起始点，每个起始点往后 4096 个数据进行堆叠，得到灰度图。

具体代码如下：

```
import numpy as np
import random
import re
import imageio
import matplotlib.pyplot as plt


def load_data(filename):
    data = []
    file = open(filename, 'r', encoding='utf-8')
    for line in file:
        line = line.replace("\n", '')
        line = re.split(r"[ ]+", line)
        if line[data_column]:
            data.append(eval(line[data_column]))
        else:
            continue
    file.close()
    return data


def turn_grayscale(data):
    lens = len(data)
    max_start = lens - dimension_grayscale**2
    starts = []
    gray_scales = []
    for i in range(sampling_value):
        while True:
            start = random.randint(0, max_start)
            if start not in starts:
                starts.append(start)
                break
    temp = data[start: start + dimension_grayscale**2]
    temp = np.array(temp)
    gray_temp = temp.reshape(dimension_grayscale, dimension_grayscale)
    gray_scales.append(gray_temp)
    return gray_scales


def draw_grayscale(graydata):
    np.savez(name_npz, *graydata)


def npz_visualization(filename):
    npz_file = np.load(filename, allow_pickle=True)
    for file in npz_file:
        temp = npz_file['{}.npy'.format(file)]
        plt.imshow(temp)
```

```
                imageio.imwrite("depth.jpg", temp)
                plt.savefig('temp.jpg')
                plt.show()
                break
        return


def divide_dataset(filename):
    npz_file = np.load(filename, allow_pickle=True)
    for file in npz_file:
        temp = npz_file[file]
        imageio.imwrite("DataSet\\{}.jpg".format(file), temp)
    return


if __name__ == '__main__':
    dimension_grayscale = 64
    data_column = 0
    sampling_value = 500
    name_npz = 'grayscales'
    DataSet = load_data('data.txt')
    graydata_temp = turn_grayscale(DataSet)
    draw_grayscale(graydata_temp)
    # npz_visualization('graydatas.npz')
    divide_dataset(name_npz + '.npz')
    print('success!')

```

转化成的灰度图像如下所示：

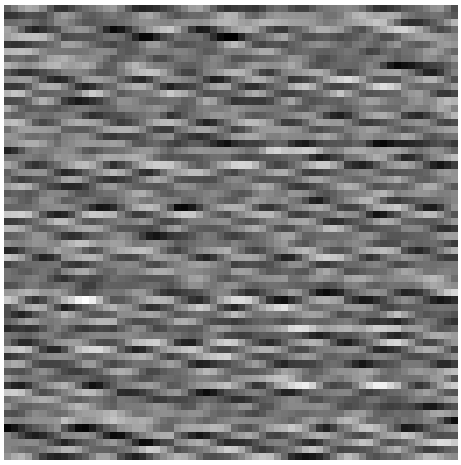

图 6: 灰度图像

我昨天也与陈师兄进行了交流，目前等他将数据仿真出来，先简单的用 CNN 测试一下，建立闭环，验证可行性，然后再考虑完善结果。

以上即为我上两周进行的主要工作。