RTR

python matplotlib 画图刻度、图例等字体、字体大小、刻度密度、线条样式设置 - Taylover-Cam 的博客 - CSDN 博客

https://blog.csdn.net/u010358304/article/details/78906768

（原）python 中 matplotlib 的颜色及线条控制 - darkknightzh - 博客园

http://www.cnblogs.com/darkknightzh/p/6117528.html

-----装了 seaborn 扩展的话，在字典 seaborn.xkcd_rgb 中包含所有的 xkcd crowdsourced color names。如下：

plt.plot([1,2], lw=4, c=seaborn.xkcd_rgb['baby poop green'])

```
import pandas  as pd
import numpy as np
from pandas import DataFrame,Series
import os  #检验文件是否存在，避免重复写入
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import re
%pylab
# import matplotlib mpl
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
# matplotlib.pyplot.colors?

#————设置字体样式，分别是字体，颜色，宽度，大小
font1 ={"family" : "TimesNew Roman",

      "weight" : "normal",
      "size"   : "22",
      }
```

```
font2 ={"family" : "TimesNew Roman",
         "color" : "red",
        "weight" : "normal",
        "size"   : "16",
        }
font3 ={"family" : "serif",
         "color" : "black",
        "weight" : "bold",
        "size"   : "24",
       }

#  [固定目录]  (F:_python\python_module)

os.chdir((r"F:\_python\python_module").replace(r"\\",r"\\\\"))
catalog_0 = os.getcwd()
catalog_0

#  [可变目录]  (F:_python\20190516)

temp = input("输入处理文件的 windows 路径:  ")
# temp = r"F:\_python\20190515\M1S170-55-55-55-55"
catalog_1_input = (temp).replace(r"\\",r"\\\\")
os.chdir(catalog_1_input)
catalog_1 = os.getcwd()  #:获得绝对路径
path = catalog_1
catalog_1

# 电机和转子
```

```
mr = catalog_1.split("\\")[-1].split("-")
mr


# python 创建顺序变量名（将字符串转换成变量）
for i in range(0,len(mr)):
    b = 'mr_' + str(i)
    exec(b + '= %r' % mr[i])


mr = [mr_0,mr_1,mr_2,mr_3,mr_4]
type(mr_1),mr_0,mr_4,mr


def Mmr(mr_x):
    if len(mr_x) is 2:
        mr_x = '(' + mr_x[0] + ',' + mr_x[1] + ')'
        return(mr_x)
    elif len(mr_x) is 3 :
        mr_x = '(' + mr_x[:2] + ',' + mr_x[-1] + ')'
        return(mr_x)
    elif len(mr_x) is 4 :
        mr_x = '(' + mr_x[:2] + ',' + mr_x[2:] + ')'
        return(mr_x)
    else :
        print("\033[1;31;43mError\033[0m")


0.746-0.339


mr_1 = "%-7s"%Mmr(mr_1)
mr_2 = "%-7s"%Mmr(mr_2)
```

```
mr_3 = "%-7s"%Mmr(mr_3)
mr_4 = "%-7s"%Mmr(mr_4)



# 定子

temp_0 = input("转子 123 的定子分别为:S--%%--%%--%%")

temp_1 = temp_0.split("-")
temp_1

# python 创建顺序变量名（将字符串转换成变量）
for i in range(1,len(temp_1)):
    b = 'stator_' + str(i)
    exec(b + '= %r' % temp_1[i])

stators = [stator_1,stator_2,stator_3]
type(stator_1),stator_1,stator_3,stators

stator_1 = "%-7s"%Mmr(stator_1)
stator_2 = "%-7s"%Mmr(stator_2)
stator_3 = "%-7s"%Mmr(stator_3)

stator_1

files = os.listdir()
files
```

```
#先编译正则表达式
p = re.compile('omiga')
for e in files :
#     调用_sre.SRE_Pattern 对象的 search ()方法
    if p.search(e) :
        omiga_file = e

# 直接用正则表达式匹配目标字符串
for e in files :
    if re.search('mcenter',e):
        mcenter_file = e



col_num = 3  #  z 列  取所有原子类型的第三列
# atom_type = list(np.r_[1:4,7:9,12])
# 1,2,3,4 电机、转子 1,2,3
atom_type = list(np.r_[1:5])   #vector 文件的第一列是 原子序号  ### 选择需要读取的列

xyz_atom_type = list(range(1,9))

def read_infile(infile):
    with open(infile, encoding='UTF-8') as f:
        rows = f.readlines()
    rows = rows[3:]  #vector 文件  跳过前三行
#     rows[:3]
    return rows

omiga = read_infile(omiga_file)
mcenter =  read_infile(mcenter_file)
```

```
def deal_data(rows):
    chunksize = int(rows[0].split()[-1]) + 1    # 取 type 类型  总数 + 1
#     motor_Zvector = float(rows[:3][1].split()[-1])
    data = []
    for i in range(0,len(rows), chunksize):
        chunk = rows[i:i+chunksize]    # 每个 type 总数  取一次
        timestep = i/chunksize    #添加列      timestep
        temp = [chunk[e] for e in atom_type]    #  按 chunksize 放到列表
        entry = [str(int(timestep))] + [row.split()[col_num] for row in temp]   #取所有原子类型的第三列   放入   entry
        data.append(entry)
    return data #,motor_Zvector


# 质心 xy


def deal_mcenter(rows):
    chunksize = int(rows[0].split()[-1]) + 1    # 取 type 类型  总数 + 1
#     motor_Zvector = float(rows[:3][1].split()[-1])
    data = []
    for i in range(0,len(rows), chunksize):
        chunk = rows[i:i+chunksize]    # 每个 type 总数  取一次
        timestep = i/chunksize    #添加列      timestep
        temp = [chunk[e] for e in atom_type]    #  按 chunksize 放到列表
        entry = [str(int(timestep))] + [row.split()[1] for row in temp]+[row.split()[2] for row in temp]  #取所有原子类型的第 1,2 列(x,y)   放入   entry
        data.append(entry)
    return data #,motor_Zvector


xyz_mcenter_df = pd.DataFrame(deal_mcenter(mcenter))
```

```
xyz_mcenter_df.head()


xyz_mcenter_df[xyz_atom_type] = xyz_mcenter_df[xyz_atom_type].astype(float)


xyz_mcenter_df.columns = ["timestep",
                          "x_M_mcenter", "x_R1_mcenter","x_R2_mcenter","x_R3_mcenter",
                          "y_M_mcenter",
                         "y_R1_mcenter",
                          "y_R2_mcenter",
                          "y_R3_mcenter"]


xyz_MC = xyz_mcenter_df.eval("""
                                x_motor = x_M_mcenter/10
                                y_motor = y_M_mcenter/10
                                x_R1    = x_R1_mcenter/10
                                y_R1    = y_R1_mcenter/10
                                x_R2    = x_R2_mcenter/10
                                y_R2    = y_R2_mcenter/10
                                x_R3    = x_R3_mcenter/10
                                y_R3    = y_R3_mcenter/10


                                  """)


MC_xy                                                                                                    =
xyz_MC[["timestep","x_motor","y_motor","x_R1","y_R1","x_R2","y_R2","x_R3","y_R3"]]


# omiga_df to RTR


omiga_df = pd.DataFrame(deal_data(omiga))
```

```
omiga_df[atom_type] =  omiga_df[atom_type].astype(float)

omiga_df.columns = ["timestep","M_omiga","R1_omiga","R2_omiga","R3_omiga"]
# omiga_df.head()

omiga_RTR = omiga_df.eval("""RTR_1 = R1_omiga/M_omiga
                        RTR_2 = R2_omiga/M_omiga
                        RTR_3 = R3_omiga/M_omiga
                        """)  #,inplace = True

RTR = omiga_RTR[["timestep","RTR_1","RTR_2","RTR_3"]]
RTR.tail()

# mcenter_df to mcenter

mcenter_df = pd.DataFrame(deal_data(mcenter))

mcenter_df[atom_type] =  mcenter_df[atom_type].astype(float)

mcenter_df.columns                                                    =
["timestep","M_mcenter","R1_mcenter","R2_mcenter","R3_mcenter"]
# mcenter_df.head()

mcenter_MC = mcenter_df.eval("""MC_1 = R1_mcenter/10
                          MC_2 = R2_mcenter/10
                          MC_3 = R3_mcenter/10

                              """)
```

```
MC = mcenter_MC[["timestep","MC_1","MC_2","MC_3"]]
# MC .head()

# 可视化

## RTR

#设置坐标的端点值 lim    设置坐标标签的间隔点 ticks  颜色   字体大小
def lim_ticks():
    y_start = -0.1
    y_end = 1.3
    x_start = -5000
    x_end = 105000
    global labelsize
    labelsize=23
    plt.xticks([])  #去掉横坐标值
    plt.yticks([])  #去掉纵坐标值
    plt.ylim(y_start,y_end)
    plt.xlim(x_start,x_end)
    # 设置坐标轴刻度的字体大小
    matplotlib.axes.Axes.tick_params
    ax.tick_params(axis='y',labelsize=labelsize,colors='k')# y 轴
    ax.tick_params(axis='x',labelsize=labelsize, colors='k')

    # ax.tick_params(axis='x',labelsize=6, colors='b', labeltop=True, labelbottom=False) # x 轴
        # 设置轴记号
    xticks([0,25000 , 50000, 75000, 100000],
```

```
    [r'$0$', r'$5$', r'$10$', r'$15$', r'$20$'])
    # 在指定标记点  标记 y 坐标 的标记（值）
    yticks([0,0.25,0.5,0.75,1],
           [r'$0$',r'$0.25$',r'$0.5$',r'$0.75$',r'$1.0$'])
    ax.set_xticklabels(["0","5","10","15","20"],font1)
    ax.set_yticklabels(["0","","0.5","","1.0"],font1)

    ax.tick_params(left= True, bottom= True)
#     ax.tick_params(right= True,top= True,left= True, bottom= True)

    plt.grid(True)   #  显示标尺
# 添加 文字 text   M1:(5,5)/(5,5)/(5,5)
import matplotlib.pyplot as plt
from matplotlib import transforms
def rainbow_text(x,y,ls,lc,**kw):
    """
    Take a list of strings ``ls`` and colors ``lc`` and place them next to each
    other, with text ls[i] being shown in color lc[i].
    This example shows how to do both vertical and horizontal text, and will
    pass all keyword arguments to plt.text, so you can set the font size,
    family, etc.
    """
    t = plt.gca().transData
    fig = plt.gcf()
    plt.show()
    #horizontal version
    for s,c in zip(ls,lc):
        text = plt.text(x,y," "+s+" ",color=c, transform=t, **kw)
        text.draw(fig.canvas.get_renderer())
```

```
        ex = text.get_window_extent()
        t = transforms.offset_copy(text._transform, x=ex.width-36, units='dots')      # x=ex.width-8 调整字符串间的 x 轴方向距离 16
    #vertical version
#     for s,c in zip(ls,lc):
#         text = plt.text(x,y," "+s+" ",color=c, transform=t,
#                 rotation=90,va='bottom',ha='center',**kw)
#         text.draw(fig.canvas.get_renderer())
#         ex = text.get_window_extent()
#         t = transforms.offset_copy(text._transform, y=ex.height, units='dots')
# plt.figure()
# rainbow_text(0,0.2,"all:unicorns/poop/rainbows/!/!/!".split("p"),
#         ['red', 'orange', 'brown', 'green', 'blue', 'purple', 'black'],
#         size=15)
# rainbow_text(0,0.2,["M1:","(5,5)","/(5,5)/","(5,5)"],
#         ['red', 'orange', 'brown', 'green', 'blue', 'purple', 'black'],
#         size=15)

arange_end  = RTR.shape[0]
figsize=11,9   #设置输出的图片大小
fig = plt.figure(figsize=figsize)
# ax = fig.add_subplot((111), projection='3d')
# ax = fig.add_subplot(3,3,4)

picture  = 1
x = np.arange(0,arange_end)
# add_subplot
nrows = 1  #图形的  行数
ncols = 1   #图形的  列数
```

```
figure_num = 1 #图形  编号
line_width = 0.1  #图像线条宽度

# motor_color = "black"
motor_color = "red"
rotor1_color = "#1F77B4"   #蓝
rotor2_color = "#FF871F"  #黄 orange
rotor3_color = "#2CA02C"   #绿
black_color  = "black"
stator_color = "silver"     # silver   gray
# motor_color = "orange"
# rotor1_color = "red"
# rotor2_color = "black"
# rotor3_color = "blue"
text_x = -4000
text_y = 1.0
text_fontsize =labelsize = 22

ax = fig.add_subplot(nrows,ncols,figure_num)
ax.scatter(x,RTR.RTR_1,picture,linewidth = line_width,color= rotor1_color )
# ax = fig.add_subplot(nrows,ncols,c)
ax.scatter(x,RTR.RTR_2,picture,linewidth = line_width,color= rotor2_color)
# ax = fig.add_subplot(nrows,ncols,c)
ax.scatter(x,RTR.RTR_3,picture,linewidth = line_width,color= rotor3_color)
ax= plt.gca()

# 调用设置 xy 坐标起终值、坐标值字体大小、轴记号  函数  lim_ticks()
lim_ticks( )
```

```
# plt.table


data_models = [mr+stators]

# columns = (''models'',''motor'',''Rotor1'',''Rotor2'',''Rotor3'',''L-/R-Stator1'',''L-/R-Stator2'',''L-/R-Stator3'')

# rows = [''CNTS'']




rainbow_text(text_x,text_y+0.2,[model_str,mr_1.strip(),space_symbol_1,
                                mr_2.strip(),space_symbol_2,stator_1.strip(),space_symbol_1,
                                mr_3.strip(),space_symbol_2,stator_2.strip(),space_symbol_1,
                                mr_4.strip(),space_symbol_2,stator_3.strip()],
             [black_color,motor_color,black_color,
              rotor1_color,black_color,stator_color,black_color,
              rotor2_color,black_color,stator_color,black_color,
              rotor3_color,black_color,stator_color
              ],size=text_fontsize)



rainbow_text(text_x,text_y+0.3,
             [''%.3f'' % RTR.RTR_1[99999],''%.3f'' % RTR.RTR_2[99999],''%.3f'' % RTR.RTR_3[99999]],
             [rotor1_color,rotor2_color,rotor3_color],size=text_fontsize)
models =  mr+stators



ticks_fontsize = labelsize

# 设置刻度字体大小

plt.xticks(fontsize=ticks_fontsize)
```

```
plt.yticks(fontsize=ticks_fontsize)

label_fontsize = labelsize + 7 # labelsize + 7 = 30
# 设置 x 轴的标签
plt.xlabel('Time(ns)',fontsize=label_fontsize)

# 设置 y 轴的标签
plt.ylabel('Rotation Transmission Ratios',fontsize=label_fontsize)

# ax.scatter(x,RTR.RTR_4,picture )
# 图中图  画质心
#新增区域 ax1,嵌套在 ax 内      add_axes
# left, bottom, width, height = 0.45, 0.2, 0.4, 0.3
# ax1 = fig.add_axes([left, bottom, width, height])
# ax1.plot(x,MC.MC_1)
# ax1.plot(x,MC.MC_2)
# ax1.plot(x,MC.MC_3)
# ax1.plot(x,MC.MC_4)
plt.savefig(catalog_1 + "\\RTR.png")
plt.show()
# plt.close()

mr+stators

motor_str = models[1]


model_str =mr_0 + ":"
space_symbol_1 = "-"
```

```
space_symbol_2 = ''/''

model_str + mr_1.strip() + space_symbol_1 + mr_2.strip() + space_symbol_2 + stator_1.strip() + space_symbol_1 + \

                                mr_3.strip() + space_symbol_2 +stator_2.strip() + space_symbol_1 + \

                                mr_4.strip() + space_symbol_2 +stator_3.strip()


rainbow_text(text_x,text_y-0.2,[model_str,mr_1.strip(),space_symbol_1,

                        mr_2.strip(),space_symbol_2,stator_1.strip(),space_symbol_1,

                        mr_3.strip(),space_symbol_2,stator_2.strip(),space_symbol_1,

                        mr_4.strip(),space_symbol_2,stator_3.strip(),space_symbol_1

                        ],

            [black_color,motor_color,black_color,

             rotor1_color,black_color,stator_color,black_color,

             rotor2_color,black_color,stator_color,black_color,

             rotor3_color,black_color,stator_color,black_color,

             ],size=text_fontsize)


RTR.tail()


''%.3f'' % RTR.RTR_1[99999]


## MC


### 可以画四张图-一张整体的空白提图（按照文献计算好坐标刻度、画 ticks、labels）

        分别画三张质心图在各自所在的尺寸范围内以相同的 0.2 间隔分别保存

        最后根据需要将三张图截图放在整体空白图所在的对应位置上


## 空白整体图
```

```
#  Z
# figsize = 22,18
# fig = plt.figure(figsize=figsize)
fig = plt.figure()
x = np.arange(0,arange_end)

rows = 3
columns = 1

y_dy = 0.2
x_start = -5000
x_end = 105000



def xylim():
    #         labelsize=  labelsize
    plt.xticks([])  #去掉横坐标值
#     plt.yticks([])  #去掉纵坐标值
#     plt.ylim( y_start,y_end)
    plt.xlim(x_start,x_end)
    # 设置轴记号
#     yticks([-0.2, -0.1 , 0, 0.1, 0.2],
#         [r'$-0.2$', r'$-0.1$', r'$0$', r'$0.1$', r'$0.2$'])
    # 在指定标记点  标记 y 坐标 的标记（值）
    xticks([0,25000 , 50000, 75000, 100000],
           [r'$0$', r'$5$', r'$10$', r'$15$', r'$20$'])
    plt.grid(True)   #  显示标尺

ax = fig.add_subplot(rows,columns,3)
```

```
ax.scatter(x,MC.MC_1,picture,linewidth = line_width,color= rotor1_color )
xylim()
y_start = min(MC.MC_1) - y_dy
y_end = max(MC.MC_1) + y_dy
plt.ylim( y_start ,y_end ,y_dy/2)


ax.tick_params(axis='y',labelsize=ticks_fontsize)# y 轴
ax.tick_params(axis='x',labelsize=ticks_fontsize)# x 轴
#隐藏图三的 top spine
ax.spines['top'].set_color('none')

ax = fig.add_subplot(rows,columns,2)
ax.scatter(x,MC.MC_2,picture,linewidth = line_width,color= rotor2_color )
xylim()
y_start = min(MC.MC_2) - y_dy
y_end = max(MC.MC_2) + y_dy
plt.ylim( y_start,y_end,y_dy)

# 隐藏坐标轴 ticks
ax.tick_params(axis='y',labelsize=ticks_fontsize)# y 轴
ax.tick_params(axis='x',labelsize=0,colors='w')# x 轴
ax.spines['top'].set_color('none')
ax.spines['bottom'].set_color('none')


ax = fig.add_subplot(rows,columns,1)
ax.scatter(x,MC.MC_3,picture,linewidth = line_width,color= rotor3_color )
xylim()
```

```
y_start = min(MC.MC_3) - y_dy
y_end = max(MC.MC_3) + y_dy
plt.ylim( y_start,y_end,y_dy)

# 隐藏坐标轴 ticks
ax.tick_params(axis='y',labelsize=ticks_fontsize)# y 轴
ax.tick_params(axis='x',labelsize=0,colors='w')# x 轴
# ax.spines['top'].set_color('none')
ax.spines['bottom'].set_color('none')


### 调整 axes 边距 九宫格  紧凑
plt.subplots_adjust(wspace =0, hspace =0)
plt.grid(True)  #  显示标尺

plt.savefig(catalog_1 + "\\MC_MR123Z.png")
plt.close()

# 取 10 个时刻 0,2，4,6，8,10，12,14，16,18,20ns 的 xy 平面内在，xy 质心的变化图示

#  X ----  Y  motor rotor123
# figsize = 22,18
fig = plt.figure(figsize=figsize)
# fig = plt.figure()
x = np.arange(0,arange_end)

rows = 2
columns = 3
```

```
xy_y_dy = 0.025
y_start = min(MC_xy.y_R2) - xy_y_dy
y_end = max(MC_xy.y_R2) + xy_y_dy
x_start = -5000
x_end = 105000


def xylim():
    #         labelsize=  labelsize
    plt.xticks([])  #去掉横坐标值
    plt.yticks([])  #去掉纵坐标值
    plt.ylim( y_start,y_end,xy_y_dy)
    plt.xlim(x_start,x_end)
    # 设置轴记号
    yticks([-0.4, -0.025 , 0, 0.025, 0.4],
        [r'$-0.4$', r'$-0.025$', r'$0$', r'$0.025$', r'$0.4$'])
    # 在指定标记点  标记 y 坐标 的标记（值）
    xticks([0,25000 , 50000, 75000, 100000],
           [r'$0$', r'$5$', r'$10$', r'$15$', r'$20$'])
    plt.grid(True)   #  显示标尺


#电机


# ax = fig.add_subplot(rows,columns,1)
# ax.scatter(x,MC_xy.x_motor,picture,linewidth = line_width*100,color= "red")
# xylim()
# # 隐藏坐标轴 ticks
# # ax.tick_params(axis='y',labelsize=0,colors='w')# y 轴
```

```
# ax.tick_params(axis='x',labelsize=0,colors='w')# x 轴
# # 隐藏坐标轴 spines
# # ax.spines['top'].set_color('none')
# ax.spines['bottom'].set_color('none')

# ax = fig.add_subplot(rows,columns,5)
# ax.scatter(x,MC_xy.y_motor,picture,linewidth = line_width*100,color= "red")
# xylim()



#转子 1


ax = fig.add_subplot(rows,columns,1)
ax.scatter(x,MC_xy.x_R1,picture,linewidth = line_width,color= rotor1_color )
xylim()
# 隐藏坐标轴 ticks
ax.tick_params(axis='y',labelsize=ticks_fontsize)# y 轴
ax.tick_params(axis='x',labelsize=0,colors='w')# x 轴
# 隐藏坐标轴 spines
# ax.spines['top'].set_color('none')
ax.spines['bottom'].set_color('none')
##################################################################
#########################

ax = fig.add_subplot(rows,columns,4)
ax.scatter(x,MC_xy.y_R1,picture,linewidth = line_width,color= rotor1_color )
xylim()
```

```
# 隐藏坐标轴 ticks
ax.tick_params(axis='y',labelsize=ticks_fontsize)# y 轴
ax.tick_params(axis='x',labelsize=ticks_fontsize)# x 轴
# 隐藏坐标轴 spines
# ax.spines['top'].set_color('none')
# ax.spines['bottom'].set_color('none')
##########################################################################
#########################

#转子 2


ax = fig.add_subplot(rows,columns,2)
ax.scatter(x,MC_xy.x_R2,picture,linewidth = line_width,color= rotor2_color )
xylim()
# 隐藏坐标轴 ticks
ax.tick_params(axis='y',labelsize=0,colors='w')# y 轴
ax.tick_params(axis='x',labelsize=0,colors='w')# x 轴
# 隐藏坐标轴 spines
# ax.spines['top'].set_color('none')
ax.spines['bottom'].set_color('none')
##########################################################################
#########################
ax = fig.add_subplot(rows,columns,5)
ax.scatter(x,MC_xy.y_R2,picture,linewidth = line_width,color= rotor2_color )
xylim()
# 隐藏坐标轴 ticks
ax.tick_params(axis='y',labelsize=0,colors='w')# y 轴
ax.tick_params(axis='x',labelsize=ticks_fontsize)# x 轴
# 隐藏坐标轴 spines
# ax.spines['top'].set_color('none')
```

```
# ax.spines['bottom'].set_color('none')

#########################################################################
#########################

#转子 3


ax = fig.add_subplot(rows,columns,3)

ax.scatter(x,MC_xy.x_R3,picture,linewidth = line_width,color= rotor3_color )

xylim()

# 隐藏坐标轴 ticks

ax.tick_params(axis='y',labelsize=0,colors='w')# y 轴

ax.tick_params(axis='x',labelsize=0,colors='w')# x 轴

# 隐藏坐标轴 spines

# ax.spines['top'].set_color('none')

ax.spines['bottom'].set_color('none')

#########################################################################
#########################

ax = fig.add_subplot(rows,columns,6)

ax.scatter(x,MC_xy.y_R3,picture,linewidth = line_width,color= rotor3_color )

xylim()

# 隐藏坐标轴 ticks

ax.tick_params(axis='y',labelsize=0,colors='w')# y 轴

ax.tick_params(axis='x',labelsize=ticks_fontsize)# x 轴

# 隐藏坐标轴 spines

# ax.spines['top'].set_color('none')

# ax.spines['bottom'].set_color('none')

#########################################################################
#########################
```

```
### 调整 axes 边距 九宫格  紧凑
plt.subplots_adjust(wspace =0, hspace =0)
plt.grid(True)  #  显示标尺

plt.savefig(catalog_1 + "\\MC_upX--downY.png")
# plt.close()

# 输出到 CSV 文件

outfile = "RTR.csv"
if os.path.isfile(outfile):
    print(outfile+"已存在")
else:
    RTR.to_csv(outfile,index=False)

print("#"*15+"  ALL  DONE  "+"#"*15)

## MC

outfile = "MC.csv"
if os.path.isfile(outfile):
    print(outfile+"已存在")
else:
    MC.to_csv(outfile,index=False)

print("#"*15+"  ALL  DONE  "+"#"*15)

os.getcwd()
```

```
os.chdir(catalog_0)

os.getcwd()
```