README

檔案名稱: benson_code

檔案內容:

- sparse_SV_channel_RIS
  - Testing data (sparse_SV_channel_RIS.py)
  - Training data (sparse_SV_channel_RIS_slices.py)
- RIS_WMMSE-MO
- RIS_DU (deep-unfolded version)
- Benchmark (GMD-PCA, T-WWMSE-MO, T-SVD)

程式使用方式:

#1. 產 Testing/Training data

Step 1 分別點開以下程式

Testing data (sparse_SV_channel_RIS.py)
Training data (sparse_SV_channel_RIS_slices.py)

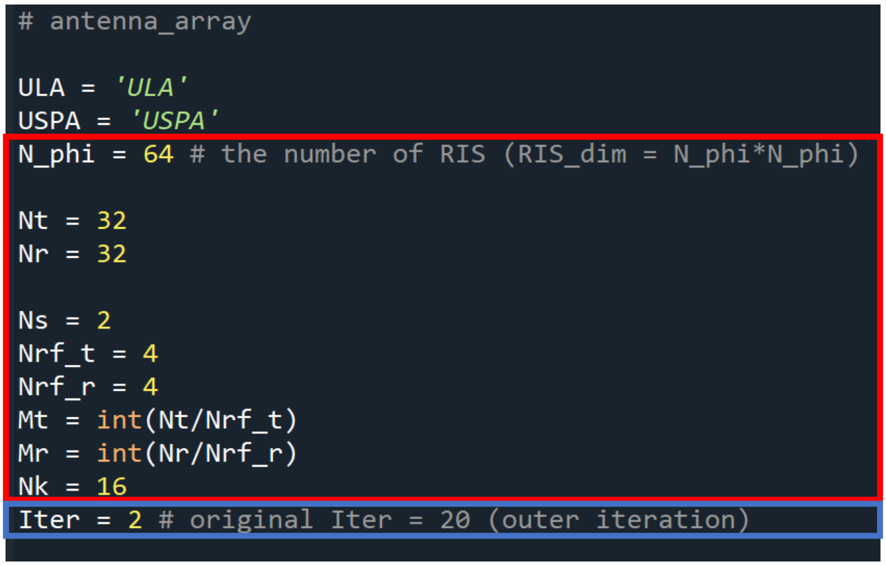
並更改 Testing/Training data 的存處位置以及所需設定的基本參數，即可執行

#2. RIS_WMMSE-MO

Step 1 安裝 python 的環境 (最後一頁有列表)

Step 2 開啟檔案 RIS_WMMSE_MO.py

Step 3 更改 Testing data 的寫入位置以及所需設定的基本參數(紅框)



```
# antenna_array

ULA = 'ULA'
USPA = 'USPA'
N_phi = 64 # the number of RIS (RIS_dim = N_phi*N_phi)

Nt = 32
Nr = 32

Ns = 2
Nrf_t = 4
Nrf_r = 4
Mt = int(Nt/Nrf_t)
Mr = int(Nr/Nrf_r)
Nk = 16
Iter = 2 # original Iter = 20 (outer iteration)
```

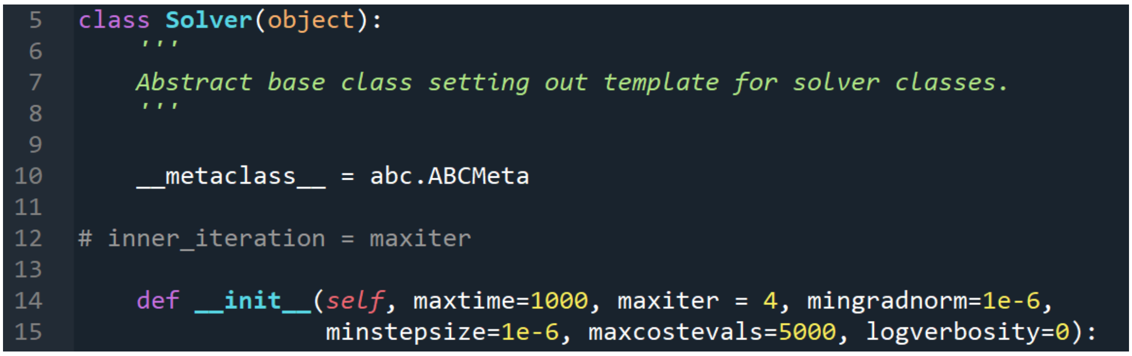
Step 4 更迭代帶次數

Outer iteration: 改藍框的數值

Inner iteration:

點開檔案 solver.py (.\benson_code**RIS_WMMSE_MO**\mypymanopt\solvers)

並更改 maxiter 的值並存檔即可



```
class Solver(object):
    '''
    Abstract base class setting out template for solver classes.
    '''

    __metaclass__ = abc.ABCMeta

# inner_iteration = maxiter

    def __init__(self, maxtime=1000, maxiter = 4, mingradnorm=1e-6,
                 minstepsize=1e-6, maxcostevals=5000, logverbosity=0):
```

Step 4 回到 RIS_WMMSE_MO.py 點選執行(RUN)即可

#3. RIS_DU

Step 1 安裝 python 的環境 (最後一頁有列表)

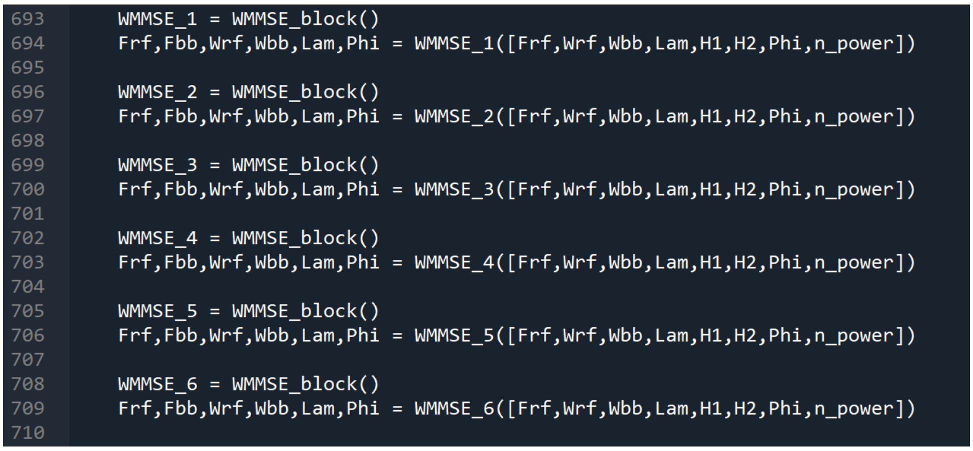
Step 2 開啟檔案 RIS_DU_train_batch.py **(Training phase)**

Step 3 更改 Training data 的讀取位置、訓練權重跟 Loss 圖位置以及所需設定的基本參數 (ex. Io (外層迭代次數)/In (內層迭代次數))

Step 4 更迭代帶次數

Outer iteration:

透過註解(#)來控制外層迭代次數，以下為 6 層外層迭代的例子，若改成 5 層則是將 708 和 709 註解，以此類推

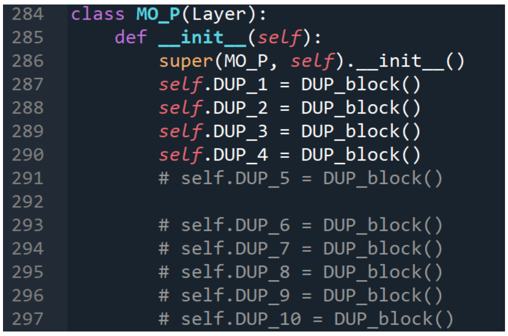


```
693        WMMSE_1 = WMMSE_block()
694        Frf,Fbb,Wrf,Wbb,Lam,Phi = WMMSE_1([Frf,Wrf,Wbb,Lam,H1,H2,Phi,n_power])
695
696        WMMSE_2 = WMMSE_block()
697        Frf,Fbb,Wrf,Wbb,Lam,Phi = WMMSE_2([Frf,Wrf,Wbb,Lam,H1,H2,Phi,n_power])
698
699        WMMSE_3 = WMMSE_block()
700        Frf,Fbb,Wrf,Wbb,Lam,Phi = WMMSE_3([Frf,Wrf,Wbb,Lam,H1,H2,Phi,n_power])
701
702        WMMSE_4 = WMMSE_block()
703        Frf,Fbb,Wrf,Wbb,Lam,Phi = WMMSE_4([Frf,Wrf,Wbb,Lam,H1,H2,Phi,n_power])
704
705        WMMSE_5 = WMMSE_block()
706        Frf,Fbb,Wrf,Wbb,Lam,Phi = WMMSE_5([Frf,Wrf,Wbb,Lam,H1,H2,Phi,n_power])
707
708        WMMSE_6 = WMMSE_block()
709        Frf,Fbb,Wrf,Wbb,Lam,Phi = WMMSE_6([Frf,Wrf,Wbb,Lam,H1,H2,Phi,n_power])
710
```

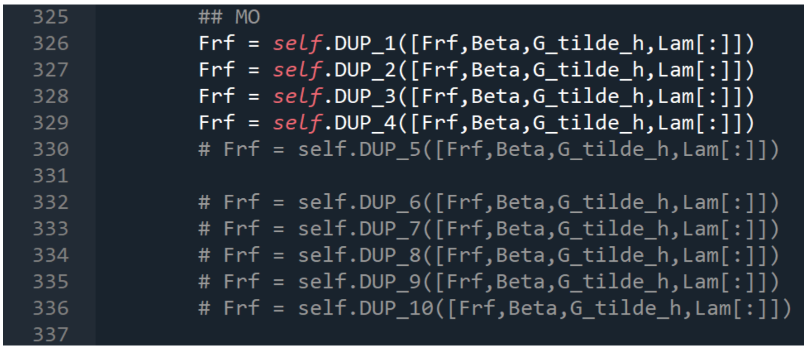
Inner iteration (F_{RF}, RIS, W_{RF}):

透過註解(#)來控制內層迭代次數

F_{RF} (4 層內層迭代)

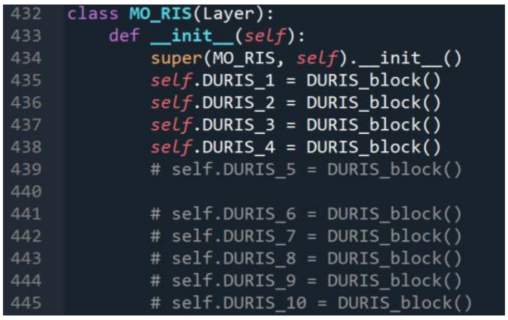


```
class MO_P(Layer):
    def __init__(self):
        super(MO_P, self).__init__()
        self.DUP_1 = DUP_block()
        self.DUP_2 = DUP_block()
        self.DUP_3 = DUP_block()
        self.DUP_4 = DUP_block()
        # self.DUP_5 = DUP_block()

        # self.DUP_6 = DUP_block()
        # self.DUP_7 = DUP_block()
        # self.DUP_8 = DUP_block()
        # self.DUP_9 = DUP_block()
        # self.DUP_10 = DUP_block()
```

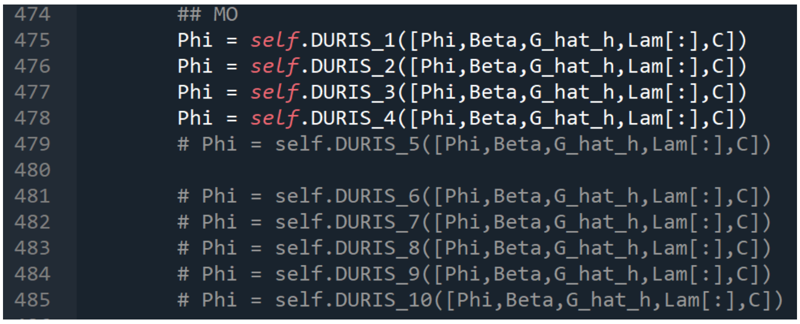


```
        ## MO
        Frf = self.DUP_1([Frf,Beta,G_tilde_h,Lam[:]])
        Frf = self.DUP_2([Frf,Beta,G_tilde_h,Lam[:]])
        Frf = self.DUP_3([Frf,Beta,G_tilde_h,Lam[:]])
        Frf = self.DUP_4([Frf,Beta,G_tilde_h,Lam[:]])
        # Frf = self.DUP_5([Frf,Beta,G_tilde_h,Lam[:]])

        # Frf = self.DUP_6([Frf,Beta,G_tilde_h,Lam[:]])
        # Frf = self.DUP_7([Frf,Beta,G_tilde_h,Lam[:]])
        # Frf = self.DUP_8([Frf,Beta,G_tilde_h,Lam[:]])
        # Frf = self.DUP_9([Frf,Beta,G_tilde_h,Lam[:]])
        # Frf = self.DUP_10([Frf,Beta,G_tilde_h,Lam[:]])
```

RIS (4 層內層迭代)

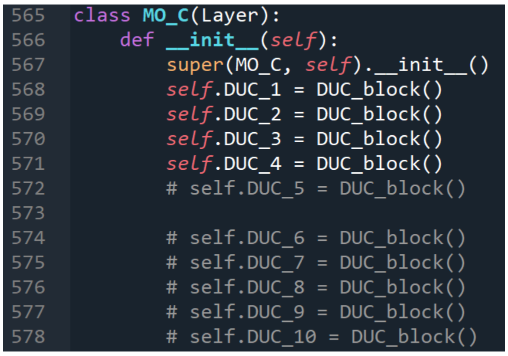


```
class MO_RIS(Layer):
    def __init__(self):
        super(MO_RIS, self).__init__()
        self.DURIS_1 = DURIS_block()
        self.DURIS_2 = DURIS_block()
        self.DURIS_3 = DURIS_block()
        self.DURIS_4 = DURIS_block()
        # self.DURIS_5 = DURIS_block()

        # self.DURIS_6 = DURIS_block()
        # self.DURIS_7 = DURIS_block()
        # self.DURIS_8 = DURIS_block()
        # self.DURIS_9 = DURIS_block()
        # self.DURIS_10 = DURIS_block()
```

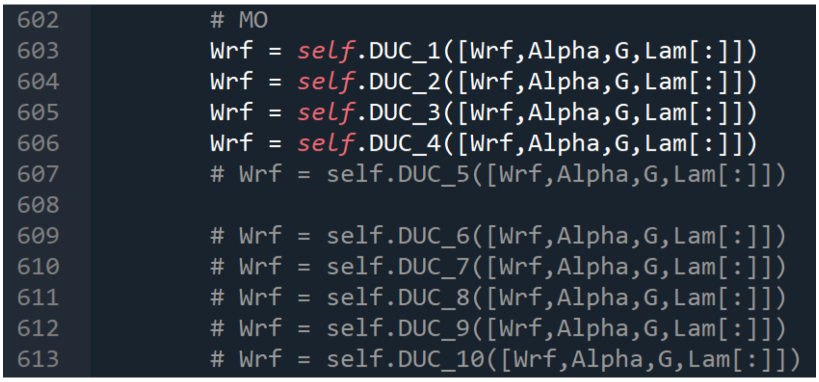


```
        ## MO
        Phi = self.DURIS_1([Phi,Beta,G_hat_h,Lam[:],C])
        Phi = self.DURIS_2([Phi,Beta,G_hat_h,Lam[:],C])
        Phi = self.DURIS_3([Phi,Beta,G_hat_h,Lam[:],C])
        Phi = self.DURIS_4([Phi,Beta,G_hat_h,Lam[:],C])
        # Phi = self.DURIS_5([Phi,Beta,G_hat_h,Lam[:],C])

        # Phi = self.DURIS_6([Phi,Beta,G_hat_h,Lam[:],C])
        # Phi = self.DURIS_7([Phi,Beta,G_hat_h,Lam[:],C])
        # Phi = self.DURIS_8([Phi,Beta,G_hat_h,Lam[:],C])
        # Phi = self.DURIS_9([Phi,Beta,G_hat_h,Lam[:],C])
        # Phi = self.DURIS_10([Phi,Beta,G_hat_h,Lam[:],C])
```

W_{RF} (4 層內層迭代)



```
class MO_C(Layer):
    def __init__(self):
        super(MO_C, self).__init__()
        self.DUC_1 = DUC_block()
        self.DUC_2 = DUC_block()
        self.DUC_3 = DUC_block()
        self.DUC_4 = DUC_block()
        # self.DUC_5 = DUC_block()

        # self.DUC_6 = DUC_block()
        # self.DUC_7 = DUC_block()
        # self.DUC_8 = DUC_block()
        # self.DUC_9 = DUC_block()
        # self.DUC_10 = DUC_block()
```



```
        # MO
        Wrf = self.DUC_1([Wrf,Alpha,G,Lam[:]])
        Wrf = self.DUC_2([Wrf,Alpha,G,Lam[:]])
        Wrf = self.DUC_3([Wrf,Alpha,G,Lam[:]])
        Wrf = self.DUC_4([Wrf,Alpha,G,Lam[:]])
        # Wrf = self.DUC_5([Wrf,Alpha,G,Lam[:]])

        # Wrf = self.DUC_6([Wrf,Alpha,G,Lam[:]])
        # Wrf = self.DUC_7([Wrf,Alpha,G,Lam[:]])
        # Wrf = self.DUC_8([Wrf,Alpha,G,Lam[:]])
        # Wrf = self.DUC_9([Wrf,Alpha,G,Lam[:]])
        # Wrf = self.DUC_10([Wrf,Alpha,G,Lam[:]])
```

Step 5 執行 RIS_DU_train_batch.py 即可

Step 6 開啟檔案 RIS_DU_x_x_test.py **(Testing phase)**

Step 7 更改 Testing data 的讀取位置、訓練權重的讀取位置以及所需設定的基本參數 (ex. Io (外層迭代次數)/In (內層迭代次數))

Step 8 執行程式即可得到測試結果

#4. Benchmark-GMD_PCA

Step 1 開啟檔案 GMD_PCA.m

Step 2 更改 Testing data 的讀取位置以及所需設定的基本參數

Step 3 執行即可

#5. Benchmark-T_SVD

Step 1 開啟檔案 T_SVD_demo.m

Step 2 更改 Testing data 的讀取位置以及所需設定的基本參數

Step 3 執行即可

P.S. 執行過程可能會出現讀不到.m 檔的問題，去 sub_func_test_ca 資料夾找檔案，並用 set path (MatLab→Home→set path) 新增路徑即可，若資料夾的檔案有缺失，去資料夾內的 T-SVD-BF.zip 找缺失的檔案

#6. Benchmark-T_WMMSE_MO

Step 1 安裝 python 的環境 (最後一頁有列表)

Step 2 開啟檔案 T_SVD_demo.m

Step 3 更改 Testing data 的讀取位置以及所需設定的基本參數

Step 4 更改內外迭代次數的方式如 RIS_WMMSE_MO

P.S. inner iteration 的 solver 檔要從 T_WMMSE_MO 開啟

(.\benson_code\Benchmark\T-WMMSE-MO\mypymanopt\solvers)

Step 5 執行即可

硬體規格:

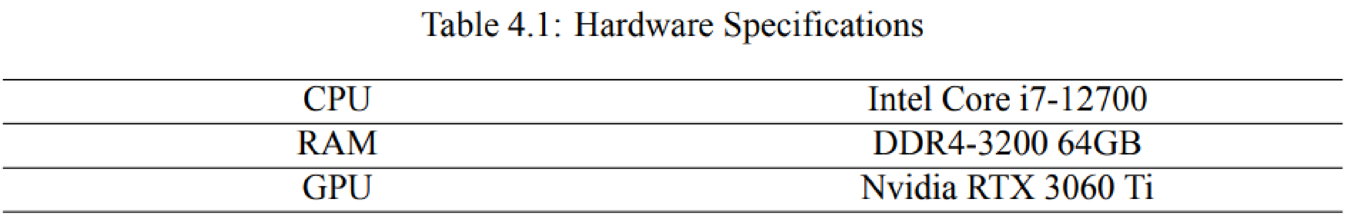


Table 4.1: Hardware Specifications

CPU	Intel Core i7-12700
RAM	DDR4-3200 64GB
GPU	Nvidia RTX 3060 Ti

Python 環境:

Name	Version
python	3.9.17
tensorflow	2.6.0
tensorflow-gpu	2.6.0
keras	2.6.0
numpy	1.25.2
matplotlib	3.5.3
scipy	1.11.1