Inventory(X_t)

Price(S_t)

Forward_Loss vs Batch

Expansion Rate

Generation Rate

Trading Rate

```
target_type='indicator',trick='clamp',loss_type='BCELoss',delta=0.01,w=0.5,lr=0.001)
```
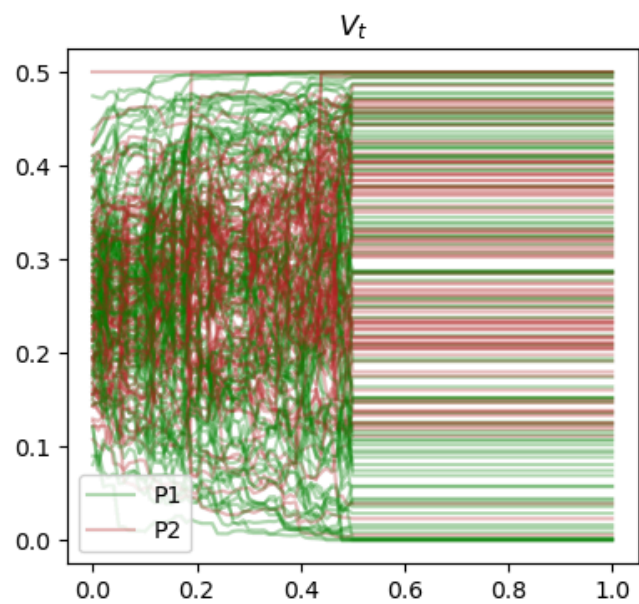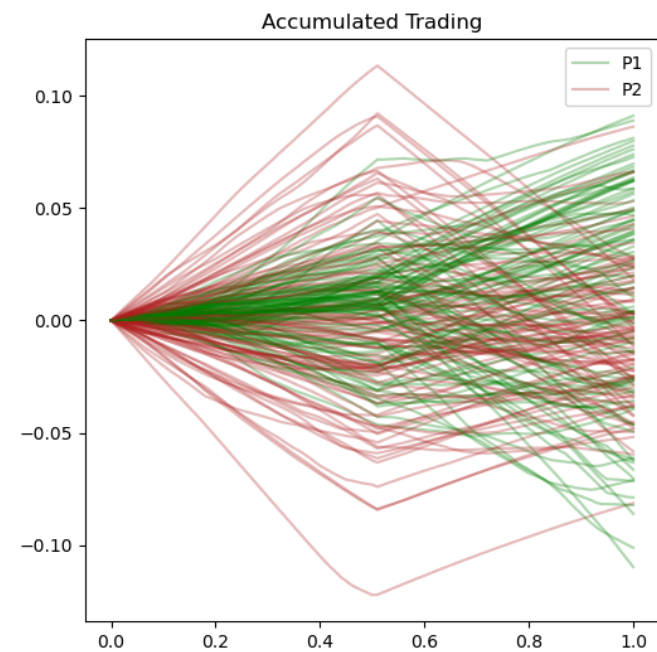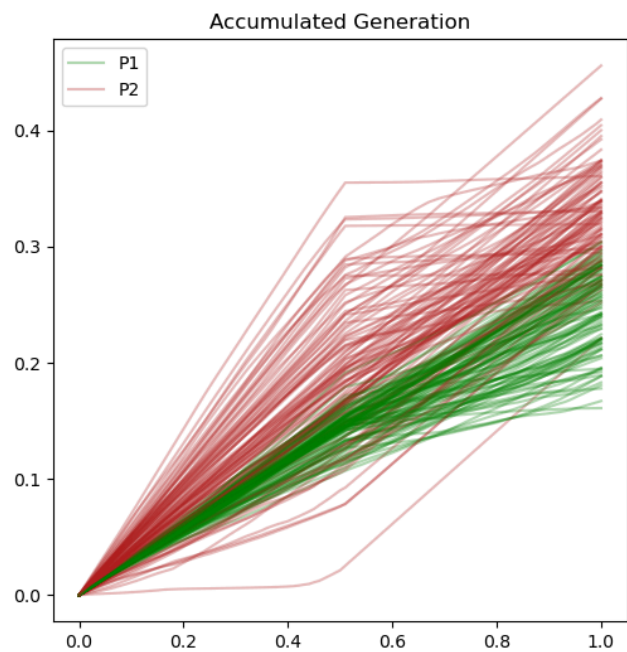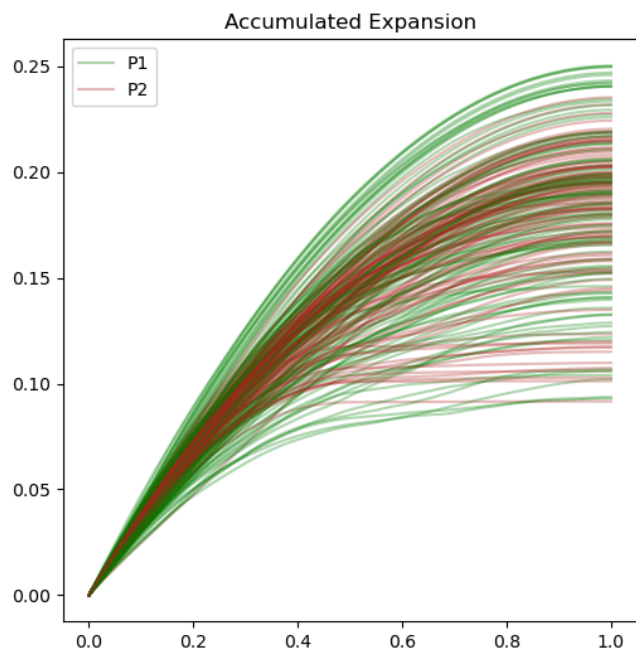
```
scaler_type='minmax'
```

500 steps
Started @ August 04 – 00:00:01
Saved @ August 04 – 04:22:37

QQ-Plot of V_t

QQ-Plot of U_t

QQ-Plot of Y_t

Convergency - P1

Distribution of $V_{T_1}^{(1)}$

$X_{T_1}^{(1)} > 0.9, V_{T_1}^{(1)} \rightarrow 0$

$X_{T_1}^{(1)} < 0.9, V_{T_1}^{(1)} \rightarrow 1$

Distribution of $U_{T_1}^{(1)}$

Distribution of $Y_{T_2}^{(1)}$

$Y_{T_2}^{(1)} \rightarrow 1$

Convergency - P2

Distribution of $V_{T_1}^{(2)}$

$X_{T_1}^{(2)} > 0.9, V_{T_1}^{(2)} \rightarrow 0$

$X_{T_1}^{(2)} < 0.9, V_{T_1}^{(2)} \rightarrow 1$

Distribution of $U_{T_1}^{(2)}$

Distribution of $Y_{T_2}^{(2)}$

$Y_{T_2}^{(2)} \rightarrow 1$
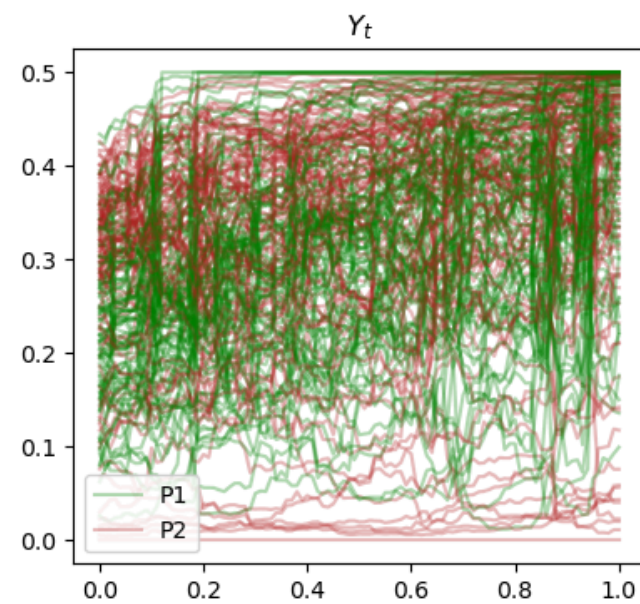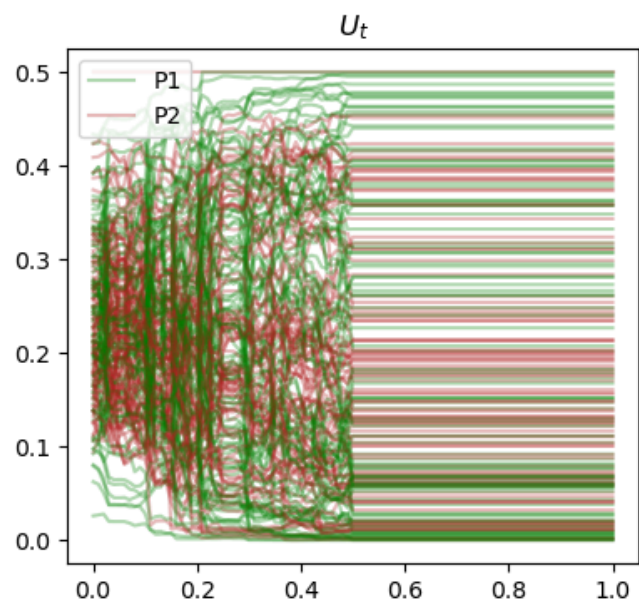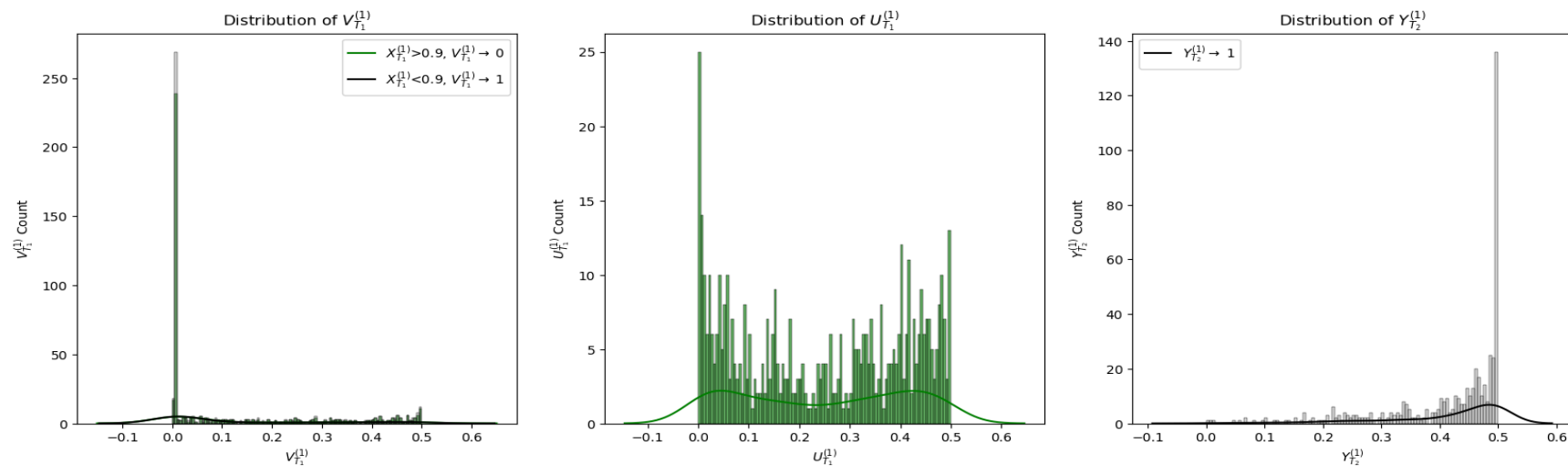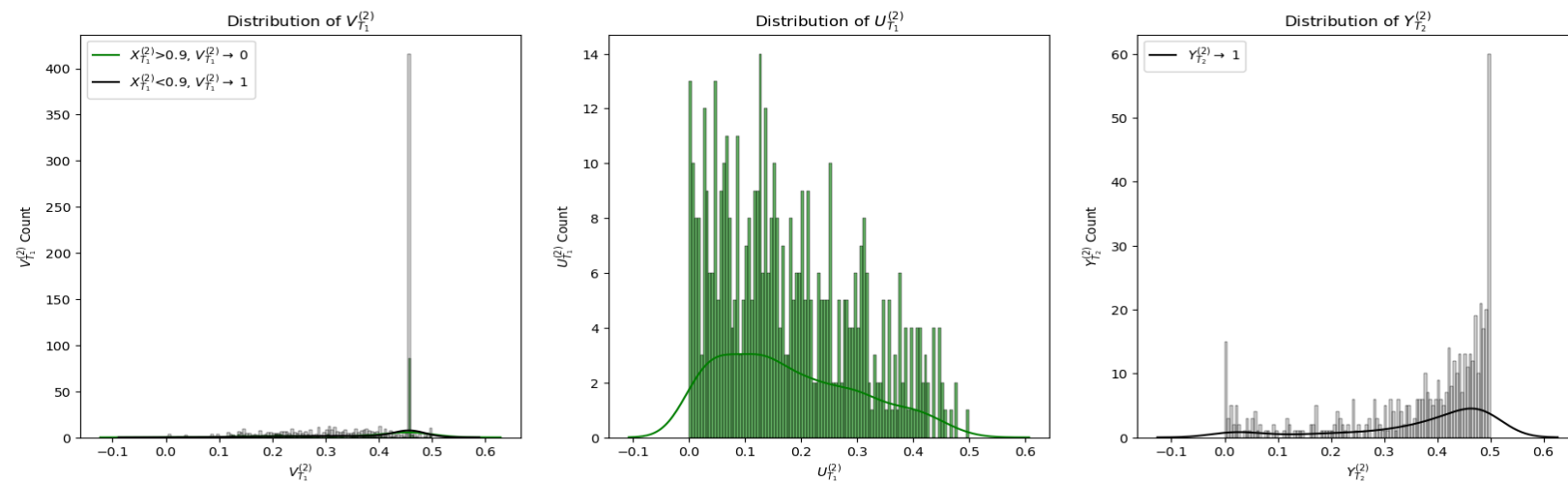
```python
import numpy as np
import torch as torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import random
from scipy.stats import norm

import os
import pathlib

from Model import *
from utils import *

torch.autograd.set_detect_anomaly(True)
start_time=datetime.now().strftime('%B %d - %H:%M:%S')
```

```python
#Global parameters
GlobalParams1=Params(param_type='k1',target_type='indicator',trick='clamp',loss_type='BCELoss',delta=0.01,w=0.5,lr=0.001)
dB1 = SampleBMIncr(GlobalParams=GlobalParams1)
init_x1 =  Sample_Init(GlobalParams=GlobalParams1)
init_c1= torch.zeros_like(init_x1)

GlobalParams2=Params(param_type='k2',target_type='indicator',trick='clamp',loss_type='BCELoss',delta=0.01,w=0.5,lr=0.001)
dB2 = SampleBMIncr(GlobalParams=GlobalParams2)    ## TODO: same dB?????
init_x2 =  Sample_Init(GlobalParams=GlobalParams2)
init_c2= torch.zeros_like(init_x2)

NT1=GlobalParams1.NT1
NT2=GlobalParams1.NT2
dt=GlobalParams1.dt
device=GlobalParams1.device
learning_rate = GlobalParams1.lr

#Forward Loss
forward_losses = []

#How many batches
MaxBatch= 500
```
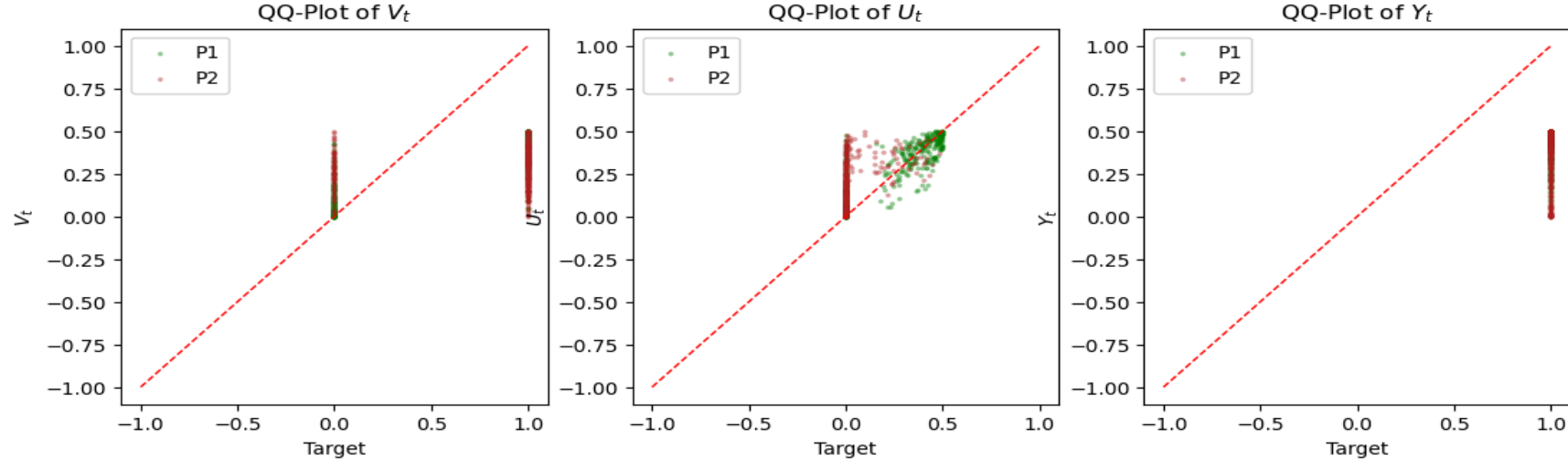
```python
#How many optimization steps per batch
OptimSteps= 25

#Train on a single batch?
single_batch = True

#Set up main models for y0 and z (z will be list of models)
v0_model_main1 = Network(scaler_type='minmax')
u0_model_main1 = Network(scaler_type='minmax')
y0_model_main1 = Network(scaler_type='minmax')

zv_models_main1 = [Network() for i in range(NT1)]
zu_models_main1 = [Network() for i in range(NT1)]
zy_models_main1 = [Network() for i in range(NT2)]
main_models1=Main_Models(GlobalParams=GlobalParams1)
main_models1.create(v0_model=v0_model_main1,
                    u0_model=u0_model_main1,
                    y0_model=y0_model_main1,
                    zv_models=zv_models_main1,
                    zu_models=zu_models_main1,
                    zy_models=zy_models_main1,
                    forward_loss=forward_losses,
                    dB=dB1,
                    init_x=init_x1,
                    init_c=init_c1)

v0_model_main2 = Network(scaler_type='minmax')
u0_model_main2 = Network(scaler_type='minmax')
y0_model_main2 = Network(scaler_type='minmax')

zv_models_main2 = [Network() for i in range(NT1)]
zu_models_main2 = [Network() for i in range(NT1)]
zy_models_main2 = [Network() for i in range(NT2)]
main_models2=Main_Models(GlobalParams=GlobalParams2)
main_models2.create(v0_model=v0_model_main2,
                    u0_model=u0_model_main2,
                    y0_model=y0_model_main2,
                    zv_models=zv_models_main2,
                    zu_models=zu_models_main2,
                    zy_models=zy_models_main2,
                    forward_loss=forward_losses,
                    dB=dB2,
                    init_x=init_x2,
                    init_c=init_c2)

pop1_dict={'dB':dB1,
```

```python
                'init_x':init_x1 ,
                'init_c':init_c1 ,
                'GlobalParams':GlobalParams1,
                'main_models':main_models1}


pop2_dict={'dB':dB2,
                'init_x':init_x2 ,
                'init_c':init_c2 ,
                'GlobalParams':GlobalParams2,
                'main_models':main_models2}
```

In [ ]:
```python
#Define optimization parameters
params=[]
params = list(main_models1.v0_model.parameters())+\
        list(main_models1.u0_model.parameters())+\
        list(main_models1.y0_model.parameters())+\
        list(main_models2.v0_model.parameters())+\
        list(main_models2.u0_model.parameters())+\
        list(main_models2.y0_model.parameters())
for i in range(NT1):
    params += list(main_models1.zv_models[i].parameters())
    params += list(main_models1.zu_models[i].parameters())
    params += list(main_models2.zv_models[i].parameters())
    params += list(main_models2.zu_models[i].parameters())

for i in range(NT2):
    params += list(main_models1.zy_models[i].parameters())
    params += list(main_models2.zy_models[i].parameters())

#Set up optimizer and scheduler
optimizer = optim.Adamax(params, lr=learning_rate)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=100, gamma=0.95)

for k in range(0,MaxBatch):

    print("Batch Number: ", k+1)
    sloss=0
    #optimize main network wrt the foward loss
    for l in range(0,OptimSteps):
        optimizer.zero_grad()
        loss = get_foward_loss(pop1_dict=pop1_dict, pop2_dict=pop2_dict)
        loss.backward()
        # torch.nn.utils.clip_grad_norm_(parameters=params,max_norm=0.7)
        optimizer.step()
        scheduler.step()
        nloss = loss.detach().numpy()
```

```python
            sloss += nloss
            # print('OptimStep: '+ str(l+1))
            # print('forward_loss: ' + str(nloss))
        avgloss = sloss/OptimSteps
        print("Average Error Est: ", avgloss)
        forward_losses.append(avgloss)

        #Generate a new batch if using multiple batches
        if(not single_batch):

            dB1 = SampleBMIncr(GlobalParams=GlobalParams1)
            init_x1 =  Sample_Init(GlobalParams=GlobalParams1)
            init_c1= torch.zeros_like(init_x1)
            pop1_dict={'dB':dB1,
                       'init_x':init_x1 ,
                       'init_c':init_c1 ,
                       'GlobalParams':GlobalParams1,
                       'main_models':main_models1}

            dB2 = SampleBMIncr(GlobalParams=GlobalParams2)   ## TODO: same dB?????
            init_x2 =  Sample_Init(GlobalParams=GlobalParams2)
            init_c2= torch.zeros_like(init_x2)
            pop2_dict={'dB':dB2,
                       'init_x':init_x2 ,
                       'init_c':init_c2 ,
                       'GlobalParams':GlobalParams2,
                       'main_models':main_models2}
```

```
In [ ]:  print(f"{len(main_models1.loss)} steps\nStarted @ {start_time}\nSaved @ {datetime.now().strftime('%B %d - %H:%M:%S')}")   ## to
         dir_path=pathlib.Path(os.getcwd(),
                               'Results',
                               'Best Models Saved',
                               'minmax_ind_0.001lr_500steps_BCE_0.5w')
         dir_path.mkdir()
         path1=pathlib.Path(dir_path,'pop1.pt')
         path2=pathlib.Path(dir_path,'pop2.pt')
         main_models1.save_entire_models(path=path1)
         main_models2.save_entire_models(path=path2)
```

```
500 steps
Started @ August 04 - 00:00:01
Saved @ August 04 - 04:22:37
```

Load The Models

```
In [ ]:  # # GlobalParams1=Params(param_type='k1',target_type='indicator',trick='clamp',loss_type='MSELoss',delta=0.03,K=0.9,lr=0.01)
         # # GlobalParams2=Params(param_type='k2',target_type='indicator',trick='clamp',loss_type='MSELoss',delta=0.03,K=0.9,lr=0.01)
         # models1=Main_Models(GlobalParams=GlobalParams1)
         # models2=Main_Models(GlobalParams=GlobalParams2)
         # path_dir=pathlib.Path(os.getcwd(),
         #                       "Results",
         #                       "Best Models Saved",
         #                       'minmax_logit_0.01lr_300steps_BCELogits_1_dt')
         # path1=pathlib.Path(path_dir,'pop1.pt')
         # path2=pathlib.Path(path_dir,'pop2.pt')
         # model_dict1=models1.load_entire_models(path=path1,overwrite=True)
         # model_dict2=models2.load_entire_models(path=path2,overwrite=True)

         # dB1=model_dict1['dB']
         # init_x1=model_dict1['init_x']
         # init_c1=model_dict1['init_c']
         # pop1_dict= {'dB':dB1,
         #             'init_x':init_x1,
         #             'init_c':init_c1,
         #             'GlobalParams':GlobalParams1,
         #             'main_models':models1}

         # dB2=model_dict2['dB']
         # init_x2=model_dict2['init_x']
         # init_c2=model_dict2['init_c']

         # pop2_dict= {'dB':dB2,
```