nrLDPC.py 및 communication.py 테스트

nrLDPC의 경우, 5G표준으로 nrLDPC의 성능 지표인 다음 논문에서 제시된 결과값과 동일한 결과를 내는 것을 목표로 한다.

Bae, J., Abotabl, A., Lin, H., Song, K., & Lee, J. (2019). An overview of channel coding for 5G NR cellular communications. APSIPA Transactions on Signal and Information Processing, 8, E17. doi:10.1017/ATSIP.2019.10

시뮬레이션 환경은, Zc=384이고, code rate가 각각 1/3, 2/3, 8/9로 동일한 조건을 갖는다.

논문의 결과는 다음과 같다.

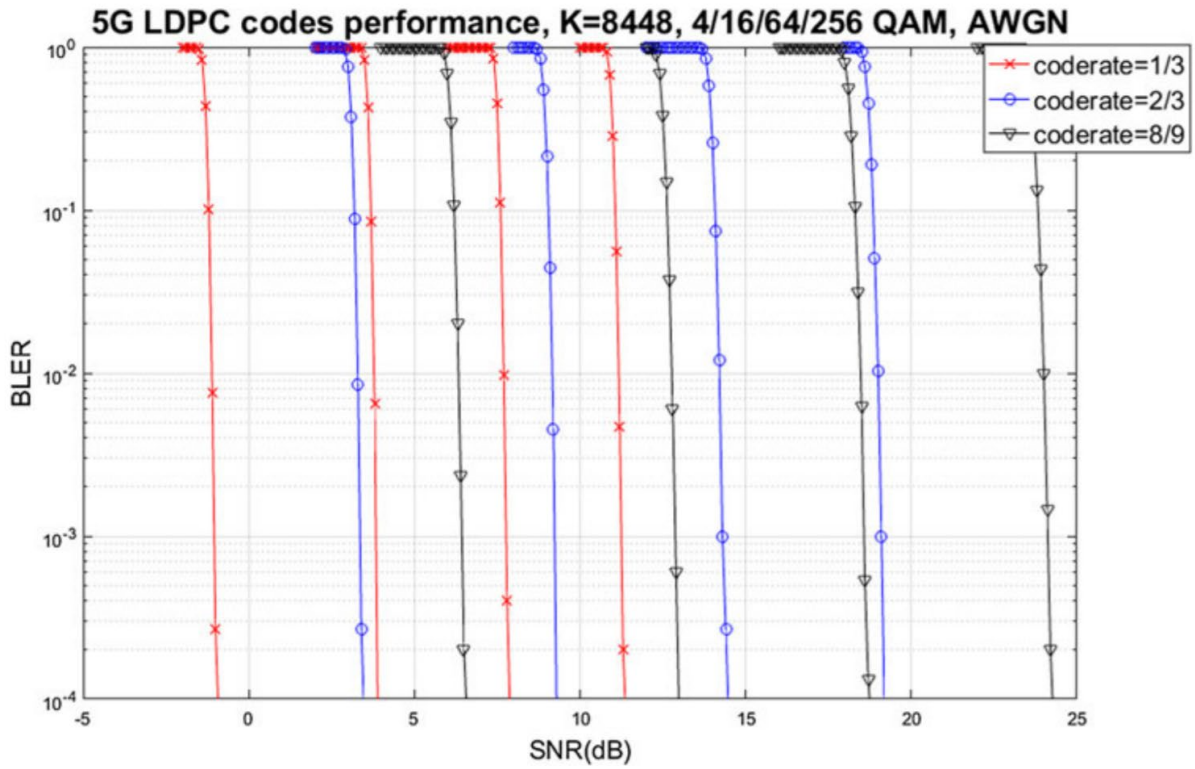


Fig. 13. 5G LDPC codes performance in the AWGN channel for various code rates and modulation orders.

GPU 메모리의 크기 문제로, 한 번에 모든 code rate에 대해 시뮬레이션 할 수 없기 때문에, 각 code rate에 대한 결과값을 저장하고, 마지막에 불러온 뒤, 결과를 확인하는 과정으로 진행한다.

코드의 흐름은 다음과 같다.

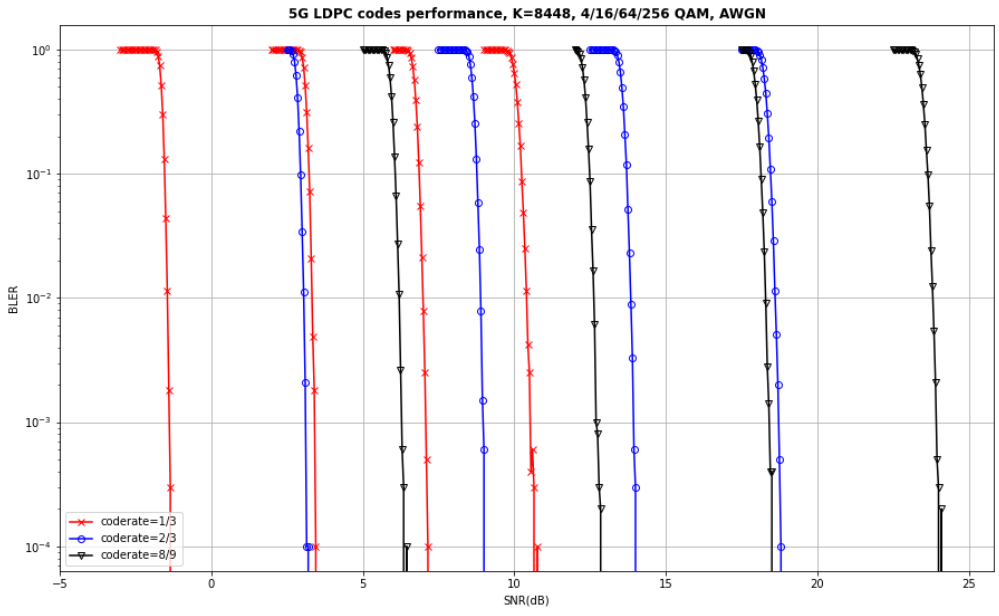
```
code_params={}
comm_params={}

"""
Z_c_list = [2, 4, 8, 16, 32, 64, 128, 256, ...
            3, 6, 12, 24, 48, 96, 192, 384, ...
            5, 10, 20, 40, 80, 160, 320, ...
            7, 14, 28, 56, 112, 224, ...
            9, 18, 36, 72, 144, 288, ...
            11, 22, 44, 88, 176, 352, ...
            13, 26, 52, 104, 208, ...
            15, 30, 60, 120, 240];
"""
code_params['Zc'] = 384
code_params['BG'] = 1
code_params['max_iter'] = 30
code_params['Decode_format'] = 'bits' #'llr'
code_params['channel_code_rate'] = '1/3'

comm_params['mod_type'] = 'qam' # bpsk
comm_params['mod_order'] = 64     # 4, 16, 64, 256



# 1/3 rate
EsdB_list = {4:torch.arange(-3, -1, 0.05),
             16:torch.arange(2, 4, 0.05),
             64:torch.arange(6, 8, 0.05),
             256:torch.arange(9, 11, 0.05)}

# # 2/3 rate
# EsdB_list = {4:torch.arange(2.5, 4.5, 0.05),
#              16:torch.arange(7.5, 9.5, 0.05),
#              64:torch.arange(12.5, 14.5, 0.05),
#              256:torch.arange(17.5, 19.5, 0.05)}

# #8/9 rate
# EsdB_list = {4:torch.arange(5, 7., 0.05),
#              16:torch.arange(12., 14., 0.05),
#              64:torch.arange(17.5, 19.5, 0.05),
#              256:torch.arange(22.5, 24.5, 0.05)}


snr_list = 10.0**(EsdB_list[comm_params['mod_order']]/10.0)
ldpc = nrLDPC(code_params, device)
comm = communication(comm_params, device)

cc_k = ldpc.cc_k
cc_n = ldpc.cc_n
cc_n_punctured = ldpc.cc_n_punctured
cc_rate = cc_k/cc_n_punctured
```

```
o = []
batch_size = 1000
num_batch = 10
q = [4, 16, 64, 256]
for order in tqdm(q):
    snr_list = 10.0**(EsdB_list[order]/10.0)
    BER = np.zeros(len(snr_list))
    comm_params['mod_order'] = order
    ldpc = nrLDPC(code_params, device)
    comm = communication(comm_params, device)

    for i, snr in enumerate(snr_list):
        data = gen_random_bits(batch_size, cc_k).to(device)
        cw = ldpc.nrLDPC_encode(data)
        x = comm.modulate(cw)
        for j, bi in enumerate(np.arange(num_batch)):
            y = comm.AWGN(x, snr)
            llr = comm.demodulate(y, snr)
            xhat = ldpc.LDPC_decode(llr)
            bhat = xhat[:, 0: cc_k]
            BER[i] += torch.count_nonzero(torch.sum((data != bhat).type(torch.float32), axis=1))

        BER[i] = BER[i]/(num_batch*batch_size)
    o.append(BER)
```

단일 code rate에 대하여 4QAM, 16QAM, 64QAM, 256QAM의 modulation에 대한 성능 지표를 구한다.

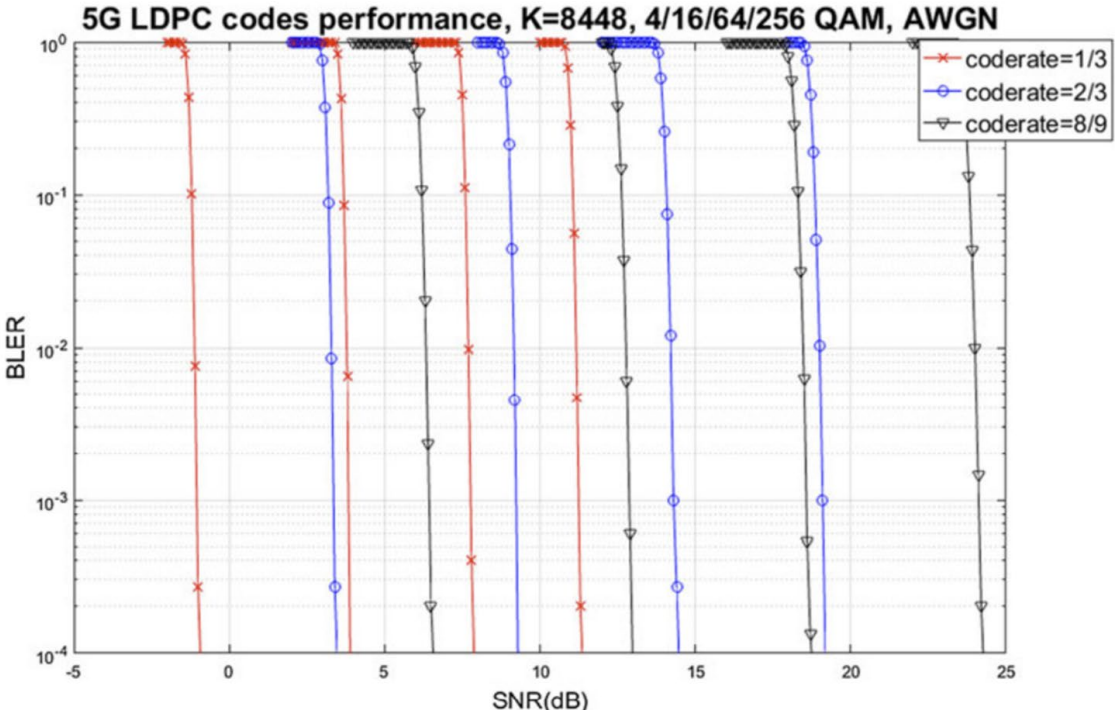
```
cr13 = torch.load('1_3 rate.pt')
cr23 = torch.load('2_3 rate.pt')
cr89 = torch.load('8_9 rate.pt')
```

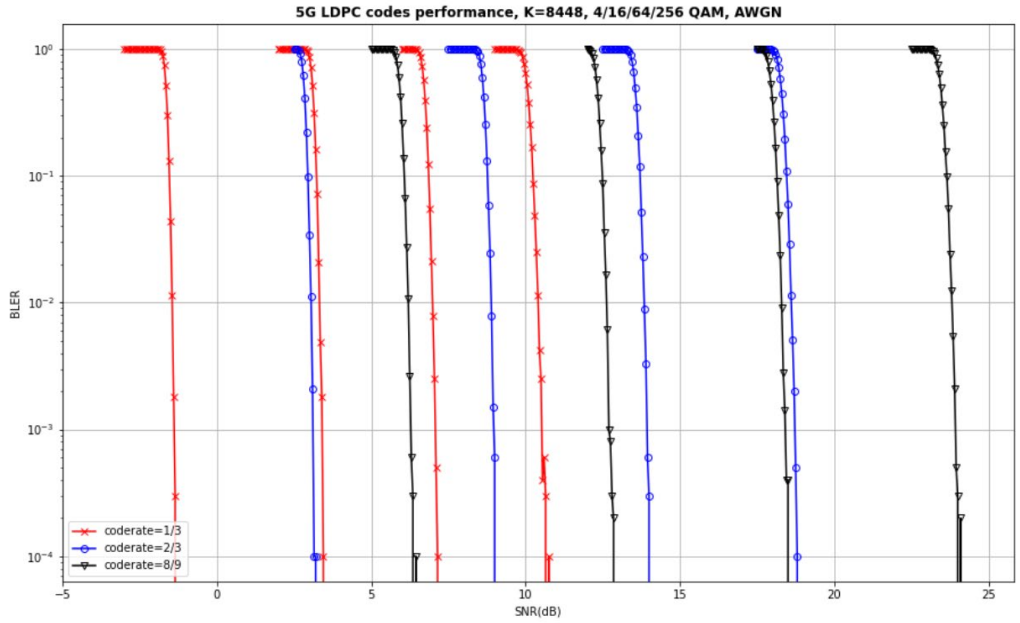
위에서 언급한 모든 code rate에 대하여 성능 성능지표가 저장이 된 후, plot하여 결과값을 비교한다.



[구현한 시뮬레이터로부터 얻은 성능 지표]

두 지표를 비교하면 거의 차이가 없는 것을 확인할 수 있다.





위의 테스트 결과를 통해 본 프로젝트의 시뮬레이션 패키지가 잘 동작함을 확인할 수 있다.