

# Instructions for using JTF-Net to reconstruct spectra

## 1. 2D NMR spectra

### 1.1 Test

#### 1.1.1 Preprocess

1) Firstly, copy the NUS data and its sampling scheme to the NUS\_data/2D/. Use NMRPipe to automatically read NUS data. Enter “varian” or “bruker” (depending on the data type) in the NMRPipe terminal, as shown in Figure R1. Select the corresponding NUS data and undersampling scheme. Click “Read Parameters” and “Save Script” to automatically generate a script named fid.com. After running fid.com, you can obtain NUS data with unsampled points replaced by 0, which will be saved as “test.fid” in NMRPipe format.

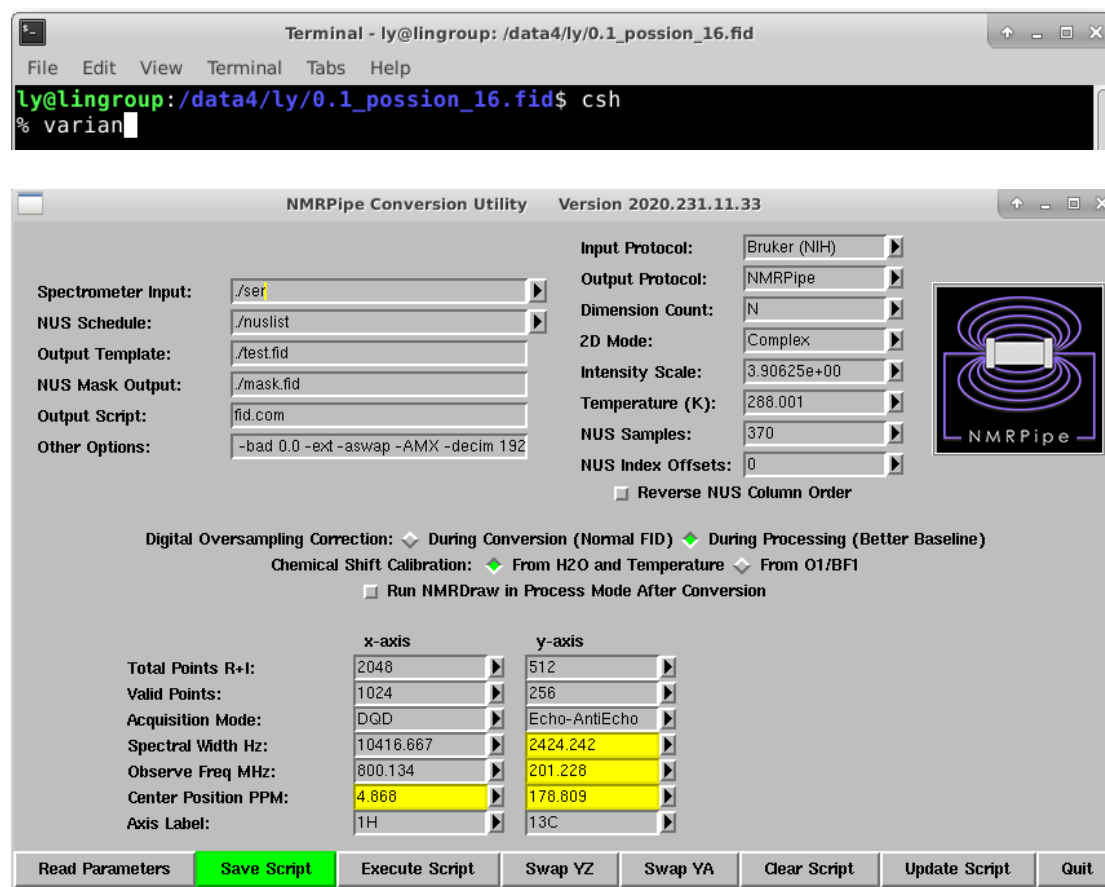


Figure R1. Read NUS data by NMRPipe

2) Copy the NMRPipe\_Code/2D/preprocess.com (available on GitHub) to NUS\_data/2D/, and run preprocess.com to preprocess the NUS data. This includes

processing steps such as direct dimension Fourier transform, using a window function, zero filling, etc (Figure R2). The preprocessed data will be saved as “input.dat” in NMRPipe format.

```
#!/bin/csh
nmrPipe -in test.fid
| nmrPipe -fn SP -off 0.5 -end 0.8 -pow 1 -c 0.5 \
| nmrPipe -fn FT -auto
| nmrPipe -fn PS -p0 -48 -p1 0.0 -verb -di \
| nmrPipe -fn EXT -x1 10.8ppm -xn 5.8ppm -sw \
| nmrPipe -fn POLY -auto \
| nmrPipe -fn TP
| nmrPipe -ov -out full.ft1

nmrPipe -in full.ft1 \
| nmrPipe -fn EM -lb 15 -c 0.5 \
| nmrPipe -fn ZF -size 176
| nmrPipe -ov -out input.dat
```

Figure R2. Part of NMRPipe\_Code/preprocess.com

1.1.2 Run Python\_code/2D/test\_code/test\_JTF-Net.py and input the necessary details as instructed. Before running the script, note that on lines 9 and 10 (Figure R3), two model paths are provided: “./model\_best” refers to the model we have trained, which you can download from the link provided in README in Github, while “./train\_code/model\_best” refers to the model trained by the reader (training details can be found in 1.2). Additionally, “num\_to\_rec” represents the number of times for reconstructing a single spectrum using JTF-Net (JTF-Net requires multiple reconstructions to get epistemic uncertainty), with the default set to 10.

```
9 #model_path = '../train_code/model_best'
10 model_path = './model_best'
11 num_to_rec = 10
```

Figure R3. Part of test\_JTF-Net.py

Run test\_JTF-Net.py, and these instructions (in black italics) will be displayed in the program running windows:

*Please input the size of the direct dimension:* (The direct dimension size of input.dat)

*Please input the size of the indirect dimension:* (The indirect dimension size of input.dat)

Whether to change the SNR of the original spectrum (Y/N): (Since addition of artificial noise is required to change the SNR when validating REQUIRER in this paper, this option is provided.)

*If you input “N”, it will prompt:	*If you input “Y” for additional artificial noise, it will prompt:
<i>Please input the mask path:</i> (You need to provide the path of the sampling scheme, which should be saved as a *.txt file, and ensure that the index of the first point is 0 and the file does not include blank lines, see Figure R5)	<i>Please input the noise factor:</i> (a float number)
<i>Please input the data path:</i> (Input the path of the “input.dat”, which is generated by NMRPipe_Code/2D/preprocess.com in 1.1.1, and then press Enter to start the reconstruction.)	<i>Please input the mask path:</i> <i>Please input the data path:</i> The noise factor controls the noise level. Once you input the data path, the SNR will be displayed, and you can adjust the noise factor to change the SNR.

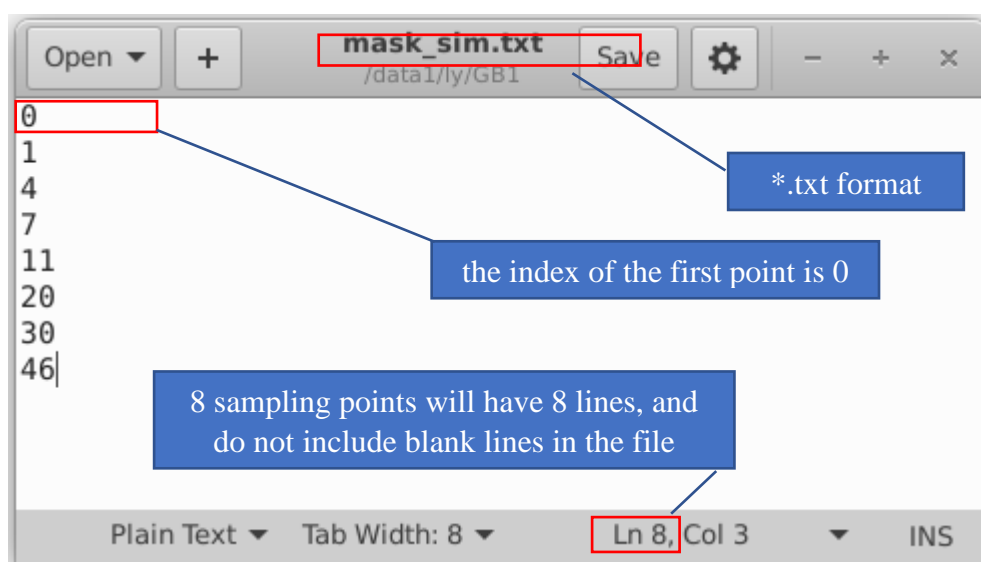


Figure R4. Details of the file of the sampling scheme

1.1.3 When the reconstruction is complete, the reconstructed spectra are automatically saved in the folder Python\_code/2D/test\_code/temp\_data/rec\_temp (this folder will be created automatically). If the “num\_to\_rec” is set to 10, the 10 reconstructed spectra and aleatoric uncertainties will be saved in Python\_code/2D/test\_code/rec\_temp.

Before running Matlab\_code/2D/prob\_predic.m (Figure R5), set the “full\_sampled” to 0 when only the NUS data is available. Then run prob\_predic.m to get REQUIRER and the reconstructed spectrum. In this case, only the REQUIRER will be output.

If the full sampled spectrum is available, the actual RLNE indicator can be output. Before running Matlab\_code/2D/prob\_predic.m, it is necessary to preprocess the fully-sampled spectrum. The same script NMRPipe\_code/2D/preprocess.com can be used to preprocess the fully sampled spectrum. The name of the output file should be set to “label.dat” and this file should be moved to the Python\_code/2D/test\_code/label\_path. Then, run the script Python\_code/2D/test\_code/trans\_label\_to\_mat.py to convert the fully-sampled spectrum to the \*.mat format, and save it in the Python\_code/2D/test\_code/label\_path. Set the “full\_sampled” variable to 1 and run Matlab\_code/2D/prob\_predic.m. In this case, the script will load the fully sampled spectrum in the Python\_code/2D/test\_code/label\_path and output both the REQUIRER and the actual RLNE.

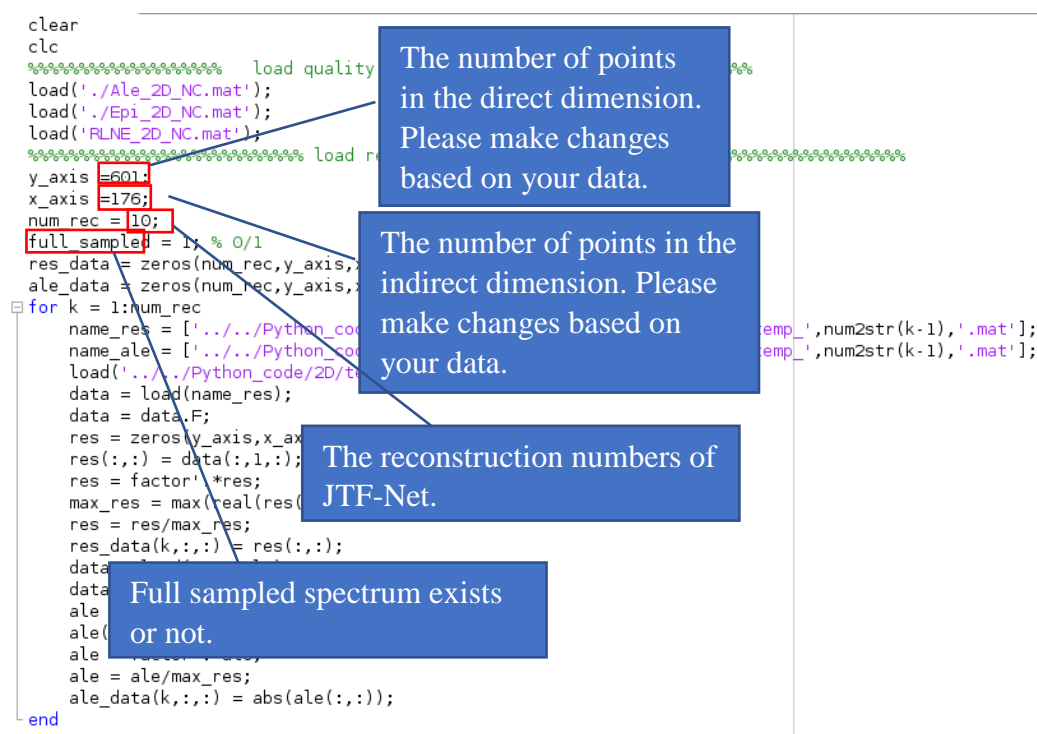


Figure R5. Part of Matlab\_code/2D/prob\_predic.m

## 1.2 Network training (optional)

If you choose to use the trained model we provide, you can skip the step. If you choose to train the model yourself, you can choose the dataset we provide, or you can choose to use the code in Matlab\_code to generate the datasets.

### 1.2.1 Generate training and validation datasets

Matlab\_code/2D/main\_pos.m (available on GitHub) can be used to generate training and validation datasets. Before running it, set the “type” in the first line of main\_pos.m to “train” to generate the training dataset. Then set the type to “val” to generate the validation dataset (Figure R6). The training and validation datasets will be saved in Dataset/2D/ (This folder will be automatically generated when running main\_pos.m for the first time, and there is no need to create it manually).

Or, you can download the training and validation datasets from the link in Github README.

```
1 - type = 'train'; % input "train" or "val" to generate training set or validation set
```

Figure R6. The part of main\_pos.m

### 1.2.2 Train JTF-Net

Run Python\_code/2D/train\_code/run.py to train JTF-Net. The training process is based on an early stopping strategy, which automatically stops when the validation loss does not improve for 10 epochs. The model with the lowest validation loss will be saved in the Python\_code/2D/train\_code/model\_best folder.

## 2. 3D NMR spectra

### 2.1 Test

#### 2.1.1. Preprocess

Firstly, copy the NUS data and its sampling scheme to the NUS\_data/3D/. Use NMRPipe to automatically read NUS data. Enter “varian” or “bruker” (depending on the data type) in the NMRPipe terminal, as shown in Figure R7. Select the NUS data and corresponding undersampling scheme, and click “Read Parameters” and “Save Script” to automatically generate a script named fid.com. After running fid.com, you can obtain NUS data with unsampled points replaced by 0, which will be saved in

NMRPipe format. If the data is in Bruker format, it will be saved at the path `./fid/test%03d.fid`. If the data is in Varian format, it will be saved at the path `./data/test%03d.fid`.

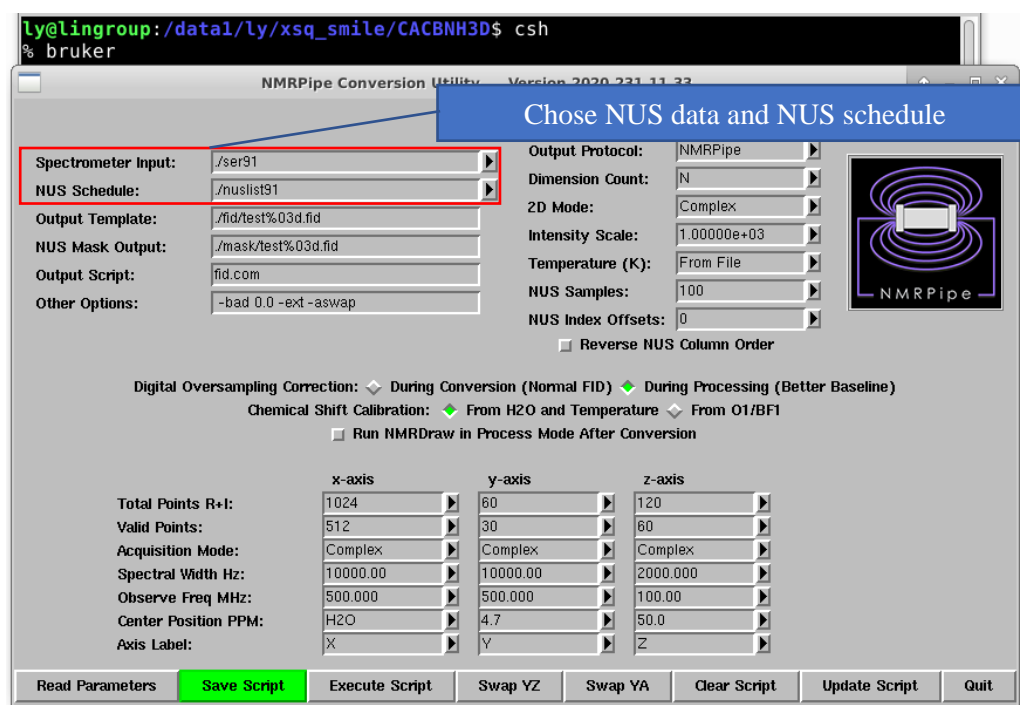


Figure R7. Generate an NMRPipe script for NUS data.

2.1.2 Copy the `3D/NMRPipe_Code/process_direct.com` and `process_indirect.com` (available on GitHub) to `NUS_data/3D/`, set the path in `process_direct.com` (Figure R8) to `fid/test%03d.fid` for Varian data (or `data/test%03d.fid` for Bruker data), and run `process_direct.com` and `process_indirect.com` in turn to preprocess the NUS data. The preprocessed data is saved as “input.dat”.

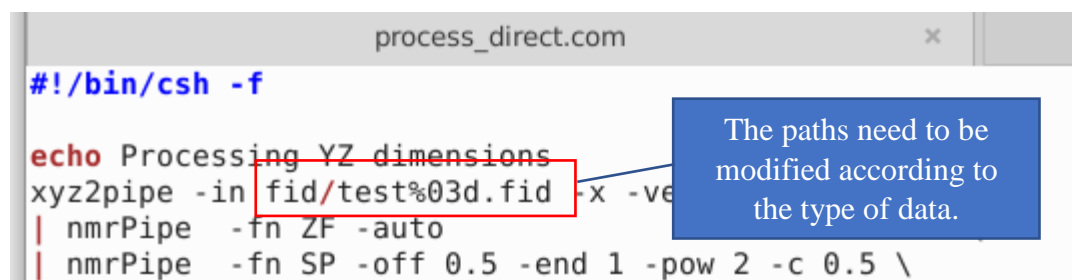


Figure R8. Part of `process_direct.com`

2.1.3 Before run `Python_code/3D/test_code/test_JTF-Net.py`. Like 1.1.2, two model paths are provided: “`./model_best`” refers to the model we have trained, which you can download from the link provided in README, while “`../train_code/model_best`” refers to the model trained by the reader (training details can be found in 2.2).

Run Python\_code/3D/test\_code/test\_JTF-Net.py, and these instructions (in black italics) will be displayed in the program running windows:

*Please input the size of the indirect dimension (z-axis):*

*Please input the size of the indirect dimension (y-axis):*

*Please input the size of the direct dimension:*

*Whether to change the SNR of the original spectrum (Y/N):* (Since the addition of artificial noise is required to change the SNR when validating REQUIRER in this paper, this option is provided.)

*If you input “N”, it will prompt:	*If you input “Y”, it will prompt:
<i>Please input the mask path:</i> (You need to provide the path to the sampling scheme, which should be saved as a *.txt file, and ensure that the index of the first sampling point is 0.) <i>Please input the data path:</i> (Input the path of the “input.dat”, which is generated by NMRPipe_Code/3D/process_indirect.com, and then press Enter to start the reconstruction.)	<i>Please input the noise factor:</i> (a float number) <i>Please input the mask path:</i> <i>Please input the data path:</i> The noise factor controls the noise level. Once you input the data path, the SNR will be displayed, and you can adjust the noise factor to change the SNR.

When the reconstruction is complete, the reconstructed spectra are automatically saved in the folder Python\_code/3D/test\_code/rec\_data (this folder will be created automatically). If the “num\_to\_rec” is set to 10, the 10 reconstructed spectra and aleatoric uncertainties will be saved in Python\_code/3D/test\_code/rec\_data.

2.1.4. Projections are required to observe the 3D spectrum when the reconstruction is completed. NMRPipe can generate projections on different planes for different types of spectra. For example, in the case of an HNCO spectrum, it can be seen from fid.com (Figure R9) that the three axes of this 3D spectrum are HN, C13, and N15. When the spectrum is projected using NMRPipe, three projection plane files are generated, including HN.C13.dat, HN.N15.dat, and C13.N15.dat, as shown in Figure R10.

The script NMRPipe\_code/3D/recFT.com is used to post-process and project the reconstructed spectra saved in Python\_code/3D/test\_code/rec\_data. Since the indirect dimensions of these spectra are in the time domain, the script first applies operations like zero-filling, Fourier transforms, and windows functions before performing the projections. Before running recFT.com, ensure that the projection plane names are correctly set in the script according to your observation needs (Figure R11).

Run recFT.com, and the processed reconstructed spectra will be saved in the NMRPipe\_code/3D/res\_nmrpipe/ directory, named res3D1.dat, res3D2.dat, ..., res3Dn.dat, and the corresponding projections of these spectra will be saved as res\_proj1.dat, res\_proj2.dat, ..., res\_projn.dat.

```
var2pipe -verb -in ./fid \
-noaswap \
-xN          2048 -yN          120 -zN          120 \
-xT          1024 -yT           60 -zT           60 \
-xMODE      Complex -yMODE      Complex -zMODE      Rance-Kay \
-xSW        11204.500 -ySW        3770.087 -zSW        3000.075 \
-xOBS        799.871 -yOBS        201.162 -zOBS        81.059 \
-xCAR        4.773 -yCAR        175.983 -zCAR        118.024 \
-xLAB        HN -yLAB        C13 -zLAB        N15 \
-ndim        3 -aq2D        Complex \
```

Figure R9. Part of a fid.com

```
Reading Projection: HN/C13
Reading Projection: HN/N15
Reading Projection: C13/N15
Writing Projection: HN.C13.dat
Writing Projection: HN.N15.dat
Writing Projection: C13.N15.dat
```

Figure R10. Information output by NMRPipe during projection

```
proj3D.tcl -in ./res_nmrpipe/res3D$I.dat -abs
nmrPipe -in C13.N15.dat
| nmrPipe -ov -out ./re
echo $I
```

Set according to the projection plane you want to observe.

Figure R11. Part of NMRPipe\_code/3D/recFT.com

2.1.5 If the full sampled spectrum is available, the actual RLNE indicator can be output. It is necessary to preprocess the fully-sampled spectrum. Copy the fully sampled data to NMRPipe\_code/3D/. The same script NMRPipe\_code/3D/process\_direct.com and process\_indirect.com can be used to preprocess the fully sampled spectrum. As



recFT.com is used for batch processing of multiple reconstruction outcomes, the script is provided for the post-processing and projection of the fully sampled spectrum. Run labelFT.com and the post-processed fully sampled spectrum and projection file are saved in the NMRPipe\_code/3D/res\_nmrpipe, named full\_spec.ft3.dat and label\_proj.dat, respectively.

Run Python\_code/3D/test\_code/nmrpipe\_to\_mat.py (available on GitHub) to convert the reconstructed spectra and their projection files in NMRPipe\_code/3D/res\_nmrpipe/ to \*.mat for REQUIRER calculation in MATLAB. The script provides the option to convert the full sampled spectrum and its projection (if available) into \*.mat. These instructions will be displayed in the program running windows:

*Convert the full sampled spectrum to mat or not? (Y/N)*

*If you input “Y”	*If you input “N”
<p>It will prompt:</p> <p><i>Please input the path of the projection file:</i></p> <p><i>Please input the path of the full sampled 3D spectrum:</i></p> <p>These *.mat files will be saved in NMRPipe_code/3D/res_mat/.</p>	<p>It will only convert the reconstructed spectra and their projection files to *.mat. These *.mat files will be saved in NMRPipe_code/3D/res_mat/.</p>

2.1.6 Run Matlab\_code/3D/prob\_predic.m (Figure R12). Set the “full\_sampled” to 0 when only the NUS data is available. In this case, only the REQUIRER will be output. Set the full\_sampled variable to 1 when you have the fully sampled spectrum. In this case, the script will output both the REQUIRER and the actual RLNE.

```

rec_path = "../../NMRPipe_code/3D/res_mat/";
direct_dim = 732;
indirect_y = 128;
indirect_z = 128;
full_sampled = 1; %0/1
z_axis = direct_dim;
y_axis = indirect_y;
x_axis = indirect_z;
res_3D = zeros(num_iter,y_axis,x_axis);
res_CN = zeros(num_iter,y_axis,x_axis);
max_res_3D = zeros(1,num_iter);
max_res_CN = zeros(1,num_iter);
for i = 1:num_iter
    name = fullfile(rec_path, ['resCN3D',num2str(i),'.mat']);
    data = load(name);
    data = data.resCN3D;
    max_res_3D(i) = max(max_res_3D, max(max(data,[],2),[],1));
    name = fullfile(rec_path, ['resCN',num2str(i),'.mat']) ;
    data = load(name);
    data = data.resCN;
    max_res_CN(1,i) = max(data(:));
    res_CN(i,:,:)= data/max_res_CN(1,i);
end

```

The number of points in the indirect and direct dimensions. Please make changes based on your data.

Full sampled spectrum exists or not.

Figure R12. Part of Matlab\_code/3D/prob\_predic.m

## 2.2 Network training (optional)

If you choose to use the trained model we provide, you can skip the step. If you choose to train the model yourself, you can choose the dataset we provide, or you can choose to use the code in Matlab\_code to generate the datasets.

### 2.2.1 Generate the training and validation datasets

Matlab\_code/3D/startGeneratingSamples.m (available on GitHub) can be used to generate training and validation datasets. The generated training and validation set will be saved in Dataset/3D/ (This folder will be automatically generated when running for the first time, and there is no need to create it manually).

Or, you can download the training and validation datasets from the link in Github README.

### 2.2.2 Train JTF-Net

Run Python\_code/3D/train\_code/run.py to train JTF-Net. Like 1.1.2, the training is also based on the early stopping strategy. The model with the lowest validation loss also be saved in the Python\_code/3D/train\_code/model\_best folder.