MATLAB-Mini-Project Report

Sampling Signals with Finite Rate of Innovation with an Application to Image Super-Resolution

Endong Sun
Department of Electrical and Electronic Engineering
Imperial College London
Es121@ic.ac.uk

I. INTRODUCTION

This report is to analyze sampling signals with Finite Rate of Innovation (FRI) and apply it to Image Super-Resolution. FRI is that non-bandlimited sample signals can be determined by a finite number of signals. Filters which satisfy Strang-Fix conditions like Daubechies filters, B-Spline filters can reproduce polynomials. Thus, filters can recover a stream of Diracs, differentiated Diracs and piecewise polynomials. Then the noisy signal will affect the reconstruction of the original signal, which can be limited by Total least-squares (TLS) approach and Cadzow's algorithm. Finally, Image Super-Resolution can be achieved combining all the procedures including sampling, kernels or filters setting, denoising to fuse all the low-resolution figure to be one super-resolution image.

This report contains 8 exercises in which exercises 1 and 2 are related to Strang-Fix conditions. Exercises 3, 4, 5 simulate the reconstruction a stream of Dirac to calculate Diracs' amplitudes and locations, sampling Diracs and the whole process of reconstruction by choosing the proper scaling function.

II. METHODOLOGY AND THEORY

A. Strang-Fix conditions

Strang-Fix conditions is the main condition for the sampling kernel, which enables kernel to reproduce polynomials. Formula is:

$$\sum_{n\in\mathbb{Z}} c_{m,n}\varphi(t-n) = t^m \quad (1)$$

Where $c_{m,n}$ is coefficients which is important as it records the kernels' information as well as original polynomials. $c_{m,n}$ can be seen as the length or amplitude after polynomials t^m projecting on the spanned space by $\varphi(t-n)$. $\varphi(t-n)$ can span as a Riesz space which has dual basis $\tilde{\varphi}(t-n)$. The dual basis is the essential basis to reproduce polynomials by applying inner product on both sides of equation (1) thus getting (2).

$$c_{m,n} = < t^m, \tilde{\varphi}(t-n) > \quad (2)$$

Then the most crucial formula that can reproduce polynomials is:

$$\hat{t}^m = \sum_{n\in\mathbb{Z}} < t^m, \tilde{\varphi}(t-n) > \varphi(t-n) \quad (3)$$

This is basically the orthogonal projection taught in lecture 2. The new $\hat{t}^m$ are reproduced polynomials which can be seen as the approximation of original t^m.

B. Exercise 1 and Daubechies Filters

Daubechies filters are orthogonal filters because $< \varphi(t), \varphi(t-n) >= \delta_0$. According to the orthogonality of Daubechies filters, equation (3) becomes:

$$\hat{t}^m = \sum_{n\in\mathbb{Z}} < t^m, \varphi(t-n) > \varphi(t-n) \quad (4)$$

Where dual basis $\tilde{\varphi}(t-n)$ is equal to $\varphi(t-n)$ because we do not need dual basis in the circumstance of orthogonality.

The Daubechies filters are used in the exercise 1. And we use dB4 which is the Daubechies filter with polynomial degree of 4. Because of the vanishing moments property of wavelets, N vanishing moments can vanish the maximum polynomial degree N-1. The maximum polynomial degree is 3 in this exercise. Therefore, dB4 is chosen by applying wavefun function in MATLAB. Then using equation (1), (2) and (4) can reproduce the polynomials as $\hat{t}^m$.

C. Exercise 2 and B-Spline Filters

B-Spline filters are generated by (N+1)-fold convolution of box function $\beta_0(t)$:

$$\beta_N(t) = \beta_0(t) * \beta_0(t) \ldots * \beta_0(t) \quad (5)$$

And box function is very simple which is one within period and zero outside of the period. This zero-order function is actually the Haar scaling function with orthogonality. However, B-Spline function has biorthogonality with increase of the polynomial degree.

B-spline filters with high order polynomial degree is one of the biorthogonal filters. Thus, we can choose equation (1), (2) and (3) to implement MATLAB code. The vanishing moment property still affects this filter, so that four-fold convolution of box function are used to generate B-Spline scaling function as equation (5) shown. The most significant point is how to derive the dual basis. The answer is using equation (8) in the matrix interpretation subsection, which can easily find the dual kernel.

D. The matrix interpretation of φ and $\tilde{\varphi}$

This subsection illustrates the relation between φ and $\tilde{\varphi}$. The inner product in equation (2) can be expanded as the equation (6), where each row vector $\tilde{\varphi}$ has n elements. In

this project, n=2048. j=32 means kernel has 32 shift versions.

$$c = \tilde{\Phi}^{-1} t^m = \begin{bmatrix} \tilde{\varphi}_0 \\ \vdots \\ \tilde{\varphi}_j \end{bmatrix} \begin{bmatrix} {t^0}_0 \cdots & {t^m}_0 \\ \vdots \ddots & \vdots \\ {t^0}_n \cdots & {t^m}_n \end{bmatrix} \quad (6)$$

Also equation (7) can be derived by equation (1):

$$t^m = \Phi c = [\varphi_0 \quad \cdots \quad \varphi_j][c_0 \quad \cdots \quad c_0] \quad (7)$$

The column vectors φ_0 has n elements. And c is the matrix of $c_{m,n}$ with column vector c_m of j elements. Therefore, the relation between φ and $\tilde{\varphi}$ can be derived as:

$$\tilde{\Phi} = \Phi(\Phi^H \Phi)^{-1} \quad (8)$$

E. Exercise 3 and Streams of Diracs

The reconstruction of a stream of Dirac algorithms contains 3 steps:

1)

First is to retrieve the first N+1 moments of the signal x(t), which is to calculate tau by:

$$\tau_m = \sum_{k=0}^{K-1} a_k t_k^m \quad m = 0,1, \dots, N \quad (9)$$

Here, K, which represents the number of Diracs, is equal to 2. And the N+1 moments τ is 4. So that we can get N=3.

2)

Second step is to find the locations t_k of x(t). The Yule-Walker system is introduced to calculate the h which is the annihilating filter. In order to solving Yule-Walker system, we need 2K tau value, which is 4 in this exercise. Moreover, the maximum polynomial degree is N, where N=3 in this case according to equation (14).

$$\begin{bmatrix} \tau_1 & \tau_0 \\ \tau_2 & \tau_1 \end{bmatrix} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = -\begin{pmatrix} \tau_2 \\ \tau_3 \end{pmatrix} \quad (10)$$

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} \quad (11)$$

Solving this equation (10) can derive h filter. Then applying z-transform on h as equation (11). And the locations t_k are roots of $H(z)$.

3)

Finally, weights a_k can be figured out by Vandermonde system like equation (12)

$$\begin{bmatrix} 1 & 1 \\ t_0 & t_1 \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \tau_0 \\ \tau_1 \end{pmatrix} \quad (12)$$

F. Exercise 4 and Sampling

Dirac function is the key for solving this exercise. It has the formula:

$$f(x) = \begin{cases} a_k, & x = locations \\ 0, & otherwise \end{cases} \quad (13)$$

So, generating the row vector with 2048 zeros and then assigning amplitudes a_k to specific two locations can give the Dirac function. Then the function should be transferred to time domain as $f(t)$ with 1/64 interval between each sample point. The amount of sample points remains 2048. The locations and amplitudes what we defined in the Diracs function are $x_0 = 1000, x_1 = 2000$, $t_0 = 15.625, t_1 = 31.25$, $a_0 = 10$, $a_1 = 20$

Then we need to sample $f(t)$ to $y[n]$. Its sampling process is similar as the Exercise 1. dB4 is chosen to provide kernels to make inner product with $f(t)$ (equation (15)) according to relationship between number of Diracs (K=2) and the maximum polynomial degree N=3 as equation (14) shown. As same as the use of dB filter in exercise 1, the number of vanishing moments (N+1=4) can vanish the maximum polynomial degree N=3. That is why the Daubechies filter with 4 vanishing moments can be used here.

$$N \geqslant 2K\text{-}1 \quad (14)$$

$$y[n] = < f(t), \varphi(t-n) > \quad (15)$$

Then dB4 generates kernels that makes inner product with f(t) to form the sampled function y[n]. Reconstruction of a stream of Diracs is again applied through 3 steps reconstruction algorithms like Exercise 3 mentioned including tau calculation, solving Yule-Walker system and Vandermonde system. The difference is that moments tau is not given. But equation (16) can handle this problem.

$$s[m] = \tau_m = \sum_n c_{m,n} y[n] \quad (16)$$

Finally, 3 steps reconstruction algorithms find the locations of Dirac, and the amplitudes a_k are also achieved.

G. Exercise 5

This exercise is easier than exercise 4 as the sampled y[n] is given in the samples.mat document. Same as before, 3 steps reconstruction algorithms in Exercise 3 can solve this exercise to find locations and amplitudes of 2 Diracs.

H. Exercise 6

This exercise is similar as before including defining Diracs function with same amplitudes and locations, choosing filter or scaling function to sample f(t) and only the first step of the reconstruction algorithm to compute moments.

The difference is that N>2K, which means the maximum polynomial degree is N-1=2K=4. So according to the vanishing moment property, only dB5 can generate N=5 vanishing moments tau which can vanish the maximum polynomial degree 4.

The other new setting is adding the zero-mean Gaussian noise to the moments by using the MATLAB function randn(1,5) with zero mean and variance 1, which has the same number of vanishing moments (N).

$$\hat{S}[m] = s[m] + \epsilon[m] \quad (17)$$

$$\epsilon[m] = std \times randn \quad (18)$$

The Gaussian noise is defined by equation (18), which is the product of standard variance and zero-mean-one-variance normal distribution random number. And the variance is set as different numbers as 1, 1e3, 1e6, 1e8, 1e12. Then the new moments are produced by equation (17).

I. *Exercise 7 and TLS and Cadzow's algorithm*

This exercise firstly uses Total least-squares (TLS) approach, where constrains $\|H\|_2 = 1$. SH=0 cannot be correct with noise. So, trying to minimize $\|SH\|_2$ is a powerful method to limit the noise. Singular value decomposition (SVD) is used for deriving Matrix V, because H can be achieved by using the last column of V.

Then Cadzow's algorithm combining with TLS can denoise perfectly. The basic procedures are using matrix S and its SVD finds matrix lamda and keeping the K largest diagonal coefficients of lamda and set other elements in the matrix to be zero. Then reconstruct S by SVD and covert it to a Toeplitz matrix. Finally, iterates whole procedures to minimize noise and to get a good reconstruction of a stream of Diracs. The difference from TLS is Cadzow can improve the estimation if SNR is low.

J. *Exercise 8 and image super-resolution*

Sampling of FRI in the image super-resolution is very useful. The low-resolution (LR) images represent different scene of the camera. Image super-resolution has 2 main steps, where 1st step is Image registration that can estimate the shifts (dx, dy) between the LR images. Second step is Image fusion and restoration. LR images can be fused to generate an super-resolution (SR) image by kernels φ like generated from a 2-D cubic B-spline. In this exercise, we only need to accomplish 1st step, Image registration.

At the beginning, we need process each LR image by normalizing and setting the threshold for eliminating noise. Normalization can be divided by 255 as the range of image of each pixel's value is [0, 255]. Using mean of the pixels in each channel as threshold can decrease the noise, which means any value below threshold will be transfers as zero. The other way is median threshold, but we choose mean threshold in this exercise. Normalization and threshold setting are integrated as the function of ImageProcessing.m document.

Then all the pixels representing sample points are processed. And using coefficients provided by PolynomialReproduction_coef.mat can build moments by equation (81) whose theory is very similar as equation (9), where $S_{m,n}$ is the sample points in each 2-D LR image.

$$m_{p,q} = \sum_m \sum_n c_{m,n}^{(p,q)} S_{m,n} \quad (81)$$

$$(dx_i, dy_i) = (\bar{x}_i + \bar{y}_i) - (\bar{x}_1, \bar{y}_1) \quad (82)$$

$$(\bar{x}, \bar{y}) = (\frac{m_{1,0}}{m_{0,0}}, \frac{m_{0,1}}{m_{0,0}}) \quad (83)$$

After achieving moments, we calculate the barycenter of each LR image and then the shifts are the difference between the barycenter of f1 and the barycenter of fi as equation (82). The barycenter of f1 is defined by the 1st LR image as reference image using equation (83). Finally, filling the Tx_RGB and Ty_RGB with moments can finish step 1 of ImageRegistration.m document. The ImageFusion.m can execute and step that fuse each LR images on a single high-resolution grid with 512*512 RGB image. Then set the PSNR code to calculate Peak signal-to-noise ration of the new image and the given high-resolution image.

III. RESULTS AND DISCUSSION

A. *Exercise 1*

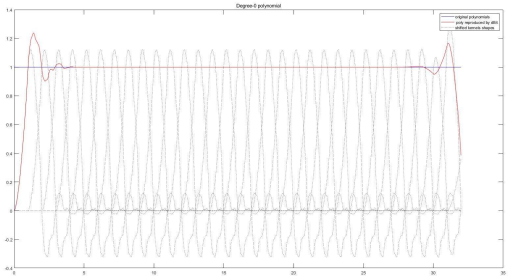

Figure1.1 Reproduced polynomials using dB4 with dgree 0

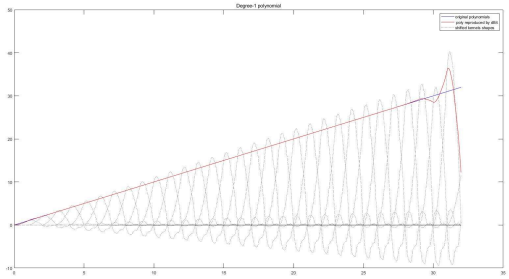

Figure1.2 Reproduced polynomials using dB4 with dgree 1

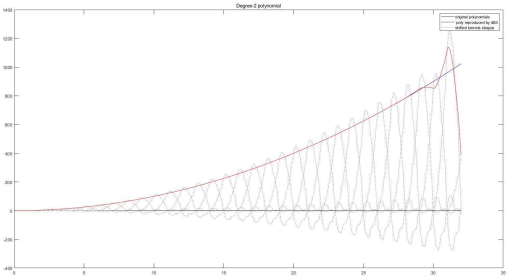

Figure1.3 Reproduced polynomials using dB4 with dgree 2

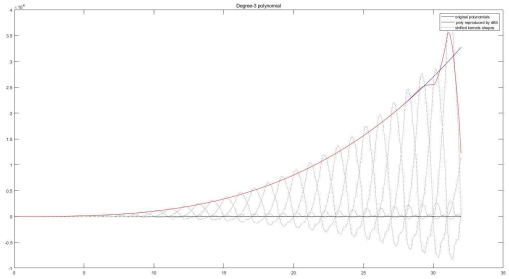

Figure1.4 Reproduced polynomials using dB4 with dgree 3

These four figures are the results of exercise 1 using dB4 filter. They show the polynomials are changing by applying different polynomial degree. The x-axis is the time of the sampling points and the y-axis is the amplitude of signals. There are 32 shifted kernels shapes, and reproduced polynomial by summation of 32 shifted version kernels according to equation (4) overlaps the original one with tiny errors, which shows the correctness of the process of polynomial reconstruction. Each shifted kernel will reproduce the corresponding polynomial.

However, we can observe there are still some errors on edges of 2 sides of the polynomial. This is because the amount of the kernel in both side is not enough to

summarize to the correct signal. We could add more kernels by decreasing the length of shifts.

B. *Exercise 2*

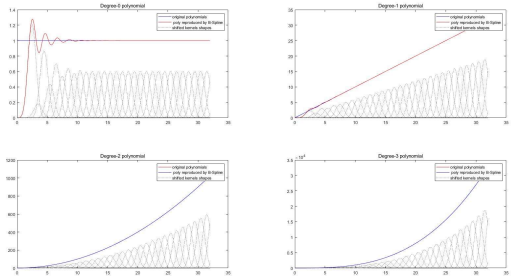

Figure 2: Reproduced polynomials using B-Spline

B-Spline function with high order of polynomial degree has biorthogonality. So, we use equation (3) to reproduce the original polynomials. First figure shows a relatively high error. But errors are decreasing with the increase of polynomial degree. As 3rd and 4th figures shown, we cannot recognize the error between reproduced polynomial and original one as they are fully overlapped. One of the reasons of the big error is as same as exercise 1, which is not enough kernels. The other reason is that B-Spline is non-negative so that we cannot vanish the positive value of the kernels at the beginning of the signal. But as the increase of the polynomial degree, the coefficients' values at the beginning decrease a lot which tends to be zero.

C. *Exercise 3*

After running the code to accomplish three steps to reconstruct 2 impuses of Dirac, we can get the locations t_k of the 2 Diracs and their amplitude a_k. Their values are $t_0 = 133/10 = 13.3;\ t_1 = 81/5 = 16.2;\ a_0 = 3/2 = 1.5;\ a_1 = 3/10 = 0.3.$

D. *Exercise 4*

Defining the Diracs function and carrying out sampling process of functions, and then using 3 steps of reconstruction algorithms can generate results. Because locations and amplitudes what we defined in the Diracs function are $x_0 = 1000, x_1 = 2000$, $t_0 = 15.625, t_1 = 31.25$, $a_0 = 10, a_1 = 20$, where x_0 and x_1 are locations in the f(x), t_0 and t_1 are locations in f(t), a_0 and a_1 are amplitudes of 2 Diracs.

The results of exercise 4 are very closed to the pre-set parameters, where $\widehat{x_0} = 1000$, $\widehat{t_0} = 15.62098$, $\widehat{x_1}$=2007, $\widehat{t_1}$=31.35275 and amplitudes are $\widehat{a_0}$=10.02 and $\widehat{a_1}$=22.48.

This indicates the sampling process works well and 3 steps of algorithm to reconstruct Diracs is successful.

E. *Exercise 5*

We use y_sample as y[n] instead of sampling process on f(t). Then reconstruction algorithm is used one more time to find the results. The results are $\widehat{x_0} = 1263$, $\widehat{t_0} = 19.734375$, $\widehat{x_1}$ =1272, $\widehat{t_1}$ =19.875 and amplitudes are $\widehat{a_0}$=1.26 and $\widehat{a_1}$=2.36

F. *Exercise 6*

31.6627	840.8460	2.3873e+04	7.1029e+05	2.1677e+07
19.7115	860.3512	2.3901e+04	7.1026e+05	2.1677e+07
168.8347	547.9748	2.4176e+04	7.1069e+05	2.1676e+07
-1.7365e+03	-2.0481e+04	3.5328e+04	7.0400e+05	2.1665e+07
-2.5391e+05	-1.4278e+06	3.0171e+03	1.4962e+05	2.3855e+07

Table 1: Moments values with 5 different variances

The results are quite interesting, where each row in Table 1 is corresponding to ever-increasing moments. And columns show the change of each moment by increasing variance δ^2. First moment and Second moment changed a lot with the change of variance. This is because variance affects these two moments a lot. However, 3rd, 4th and the last moments' values only have minor level of variation, which means the influence of the variance can be neglected on specifically the 5th moment.

G. *Exercise 7*

Only TLS: Firstly, setting N=7>2K can get locations of reconstructed Diracs. TLS performs well for locations in which are 997 and 2000. They are very closed to Diracs with locations at 1000 and 2000. What we defined are 150 and 200. The results are 141.65 and 203.69, which is very good performance. Variance of noise is 1000.

Then all the procedures of Cadzow are correct after lots of debugging. And Cadzow can work well combining with TLS approach. The results are 1001 and 2000 for locations and 157.85 and 195.89 for amplitudes. Variance in noise is 1000 as well, because signal to noise ratio (SNR) is small with this value of variance.

Overall, both algorithms work very well to reconstruct Diracs.

H. *Exercise 8*

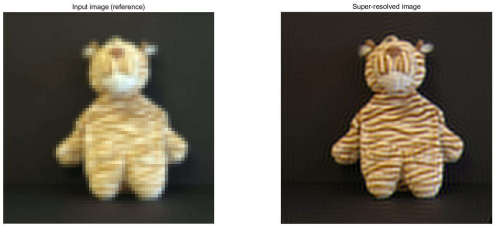

Figure 3: LR image vs SR image

The first image is 1st LR image of 40 LR images with 64*64 pixels in each RGB channel. The right image is generated by fusing with all LR images, estimating the missing data and restoration to remove blue and noise.

PSNR is calculated as 23.01 dB, which indicates the code is successful.

IV. CONCLUSION

These questions practiced how to select the appropriate scaling function for sampling and how to reconstruct originally defined signals. Noise is then applied to the vanishing moments to simulate real signal scenario. TLS and Cadzow can achieve an effect of noise reduction to reconstruct Diracs. Finally, there are two steps to construct super-resolution images, which are image registration and image fusion and restoration.

Two filters, Daubechies and B-Spline, are programmed to sample the original signal respectively. They conform to Strang-Fix conditions and wavelet property, that is N

vanishing moments can vanish the maximum polynomial degree N-1. In the end, the 32 different shifts versions of the kernel sampled the original signal well with only small errors at the edges. It is worth mentioning that the difference is as follows: one is orthogonal filter, and the other is biorthogonal. The former only requires the original φ to carry out sampling as Kernel, while the latter must calculate dual basis $\tilde{\varphi}$ to carry out sampling.

In exercises 3, 4, 5 and 6, a stream of Diracs is constructed and sampled respectively, and then Diracs are reconstructed through 3-step reconstruction algorithm and the effect of noise applied on moments is observed. Exercise 7 uses a combination of two denoising techniques, TLS and Cadzow, which are essentially based on minimizing $\|SH\|_2$ for $\|H\|_2 = 1$. However, Cadzow is more targeted at small SNR and changes S with noise into Toeplitz matrix with rank to K

The final exercise is programmed in the ImageRegistration. The whole process is the preprocessing of sampling points, including normalization, setting thresholds for denoising, and applying the given coefficients to work out moments thus obtaining shifts on each LR image. In this way, by using ImageFusion, all images can be integrated, and the missing points can be estimated to obtain a high-resolution image.

Overall, the essence of the whole signal sampling and reconstruction is based on the inner product space and projection knowledge. This report well applies the knowledge to the field of signal processing, and these achievements can have a good application prospect in the fields of image, communication, and computer vision.

REFERENCES

No references. All the knoweldges are from lecturenotes.

APPENDIX

I do not know how to upload codes in a pdf document. So, I upload them on my repositories on github, you can assess here: https://github.com/Anthony-EEE/MATLAB-Mini-Project-wavelet