FMAN45 Machine Learning, spring 2020
Assignment 1

Salma Kazemi Rashed

April 2020

Task 1: Verify the first line of (2) by solving (1) for $w_i \neq 0$.

$$\underset{w_i}{\text{minimize}} \ \frac{1}{2}||\mathbf{r}_i - \mathbf{x}_i w_i||_2^2 + \lambda|w_i| \tag{1}$$

$$\hat{w}_i^{(j)} = \begin{cases} \frac{\mathbf{x}_i^T \mathbf{r}_i^{(j-1)}}{\mathbf{x}_i^T \mathbf{x}_i |\mathbf{x}_i^T \mathbf{r}_i^{(j-1)}|}(|\mathbf{x}_i^T \mathbf{r}_i^{(j-1)}| - \lambda), & |\mathbf{x}_i^T \mathbf{r}_i^{(j-1)}| > \lambda \\ 0, & |\mathbf{x}_i^T \mathbf{r}_i^{(j-1)}| < \lambda \end{cases} \tag{2}$$

response: In coordinate descent, we need to optimize one coordinate (w_i) at a time. In order to minimize (1), the gradient of (1) for $w_i \neq 0$ is :

$$-{\mathbf{x}_i}^T \mathbf{r}_i + {\mathbf{x}_i}^T \mathbf{x}_i w_i + \lambda \text{sgn}(w_i) = 0 \tag{3}$$

if we want to decide on the sign of w_i we will have:

$$\hat{w}_i = \begin{cases} \frac{\mathbf{x}_i^T \mathbf{r}_i - \lambda}{\mathbf{x}_i^T \mathbf{x}_i} > 0, & w_i > 0 \\ \frac{\mathbf{x}_i^T \mathbf{r}_i + \lambda}{\mathbf{x}_i^T \mathbf{x}_i} < 0, & w_i < 0 \end{cases} \tag{4}$$

$\mathbf{x}_i^T \mathbf{x}_i > 0$, therefore, from both lines we have $|\mathbf{x}_i^T \mathbf{r}_i| > \lambda$, since λ is positive. The sign of $\mathbf{x}_i^T \mathbf{r}_i$ also follows the sign of w_i. Thus, we will have:

$$\hat{w}_i = \begin{cases} \frac{\mathbf{x}_i^T \mathbf{r}_i}{\mathbf{x}_i^T \mathbf{x}_i |\mathbf{x}_i^T \mathbf{r}_i|}(|\mathbf{x}_i^T \mathbf{r}_i| - \lambda), & |\mathbf{x}_i^T \mathbf{r}_i| > \lambda \\ 0, & |\mathbf{x}_i^T \mathbf{r}_i| < \lambda \end{cases} \tag{5}$$

Now, in each iteration of coordinate descent, each variable is updated as follows:

$$\hat{w}_i^{(j)} = \underset{x}{\text{argmin}} \ f(\hat{w}_1^{(j)}, \hat{w}_2^{(j)}, ...\hat{w}_l^{(j)}, x, \hat{w}_{l+1}^{(j-1)}, .., \hat{w}_M^{(j-1)}) \tag{6}$$

And by substituting $\mathbf{r}_i^{(j-1)}$ as the function of other updated parameters, (2) is achieved from (5).

Task 2: Given the orthogonal regression matrix, show that the coordinate descent solver in (2) will converge in at most 1 full pass over the coordinates in w, i.e., show that $\hat{w}_i^2 - \hat{w}_i^1 = 0, \ \forall i$.

response: By substituting $\mathbf{r}_i^{(j-1)}$ in (2), the term $\mathbf{x}_i^T \mathbf{r}_i^{(j-1)}$ will be equal to 7:

$$\mathbf{x}_i^T \mathbf{r}_i^{(j-1)} = \mathbf{x}_i^T (\mathbf{t} - \sum_{l<i} \mathbf{x}_l \hat{w}_l^{(j)} - \sum_{l>i} \mathbf{x}_l \hat{w}_l^{(j-1)}) \tag{7}$$

The regression matrix is orthogonal, thus both $\mathbf{x}_i^T \sum_{l<i} \mathbf{x}_l \hat{w}_l^{(j)}$ and $\mathbf{x}_i^T \sum_{l>i} x_l \hat{w}_l^{(j)}$ terms will be zero since $\mathbf{x}_i^T \mathbf{x}_l = 0, \ \forall l \neq i$ and $\mathbf{x}_i^T \mathbf{x}_i = 1$. Thus, each $\hat{w}_i^j$ depends on only $\mathbf{x}_i^T, \mathbf{t}$ and λ (not other

estimates) and they are equal for $\forall j$.

$$\hat{w}_i^j = \begin{cases} \frac{\mathbf{x}_i^T\mathbf{t}}{\mathbf{x}_i^T\mathbf{x}_i|\mathbf{x}_i^T\mathbf{t}|}(|\mathbf{x}_i^T\mathbf{t}| - \lambda), & |\mathbf{x}_i^T\mathbf{t}| > \lambda \\ 0, & |\mathbf{x}_i^T\mathbf{t}| < \lambda \end{cases} \tag{8}$$

Task 3: Show that the LASSO estimate's bias, for an orthogonal regression matrix and data generated by $\mathbf{t} = \mathbf{X}\mathbf{w}^* + \mathbf{e}$, is given by (9) when $\sigma \to 0$, and discuss how this result relates to the method's acronym, LASSO.

$$\lim_{\sigma\to 0} \mathrm{E}(\hat{w}_i^{(1)} - w_i^*) = \begin{cases} -\lambda, & w_i^* > \lambda \\ -w_i^*, & |w_i^*| \le \lambda \\ \lambda, & w_i^* < -\lambda \end{cases} \tag{9}$$

response: In the task 2, I achieved that for the orthogonal regression matrix the estimated weights equals (8). First I separate (8) to three cases where $\mathbf{x}_i^T\mathbf{t} > \lambda$, $\mathbf{x}_i^T\mathbf{t} < -\lambda$, and $|\mathbf{x}_i^T\mathbf{t}| < \lambda$. Then, I substitute $\mathbf{t} = \mathbf{X}\mathbf{w}^* + \mathbf{e}$ in (8). I will have :

$$E(\mathbf{x}_i^T\mathbf{t}) = E(\mathbf{x}_i^T(\mathbf{X}\mathbf{w}^* + \mathbf{e})) = E(w_i^*) + E(\mathbf{x}_i^T\mathbf{e}) = E(w_i^*) = w_i^* \tag{10}$$

The result of $\mathbf{x}_i^T\mathbf{X}\mathbf{w}^*$ is w_i^* and the expectation of noise equals zero. Thus, we will have:

$$\lim_{\sigma\to 0} \mathrm{E}(\hat{w}_i^{(1)}) = \begin{cases} w_i^* - \lambda, & w_i^* + \mathbf{x}_i^T\mathbf{e} > \lambda \\ 0, & |w_i^* + \mathbf{x}_i^T\mathbf{e}| \le \lambda \\ w_i^* + \lambda, & w_i^* + \mathbf{x}_i^T\mathbf{e} < -\lambda \end{cases} \tag{11}$$

by assuming $\sigma \to 0$, we can assume that noise vector $\mathbf{e} \to \mathbf{0}$ and $\mathbf{x}_i^T\mathbf{e} \to 0$. Thus (9) is achieved.

LASSO adds a penalty for non-zero coefficients to the loss function and penalizes the sum of their absolute values. For high values of λ, many coefficients are exactly set to zero. LASSO is expected to perform better when there are a small number of significant parameters and the others are close to zero. Regarding the coefficients themselves, the lasso shrinkage causes the estimates of the non-zero coefficients to be biased towards zero. When λ is too large (more shrinkage of coefficients), the bias is controlled by the coefficients values. On the other hand, when λ is small (less shrinkage of coefficients) the bias is controlled by λ.

Task 4: Implement a (cyclic) coordinate descent solver for the coordinate-wise LASSO solution, using data t and regression matrix X in A1_data.mat. Do not use Xinterp for estimation, it is provided for vizualization only. Then solve the following two subtasks and discuss your results.

response: For this part, first I completed the skeleton_lasso_ccd.m file as shown in Fig. (1). The results for three $\lambda = 0.1, 3, 10$ values are depicted in Fig. (2). It is clear that for small λ values, there would be less shrinkage of coefficients and we have lots of non-zero coordinates which leads to overfitting of the model. On the other hand, for large λ values, under fitting occurs where for a middle value we could see a good fit of the model to the data. Besides, by increasing λ, the number of nonzero coordinates decreases. For second part of this task, estimated coefficients are depicted for three values of λ and the number of non-zero coordinates is shown for each plot in Fig. (12a).

```
% sweep over coordinates, in randomized order defined by kInd_random
for ksweep = 1:length(kindvec_random)
   kind = kindvec_random(ksweep); % Pick out current coordinate to modify.

   x = X(:,kind); % ... select current regression vector
   X_temp = X;
   X_temp (:,kind)=[];
   %%%%%%%%%%%
   w_temp = w;
   w_temp(kind) = [];
   %%%%%%%%%%%
   r = t - (X_temp*w_temp);  % ... calculating residual
   if (abs(x'*r)>lambda) & (x'*r > 0)
      w(kind) = (x'*r - lambda)/(x'*x);
   else if (abs(x'*r) > lambda) & (x'*r < 0)
      w(kind) = (x'*r + lambda )/ (x'*x);
      else
         w(kind) = 0; % ... update the lasso estimate at coordinate kind
      end
   end
   wsup(kind) = double(abs(w(kind))>zero_tol); % update whether w(kind) is zero or not.

end
```

Figure 1: Lasso ccd update code.

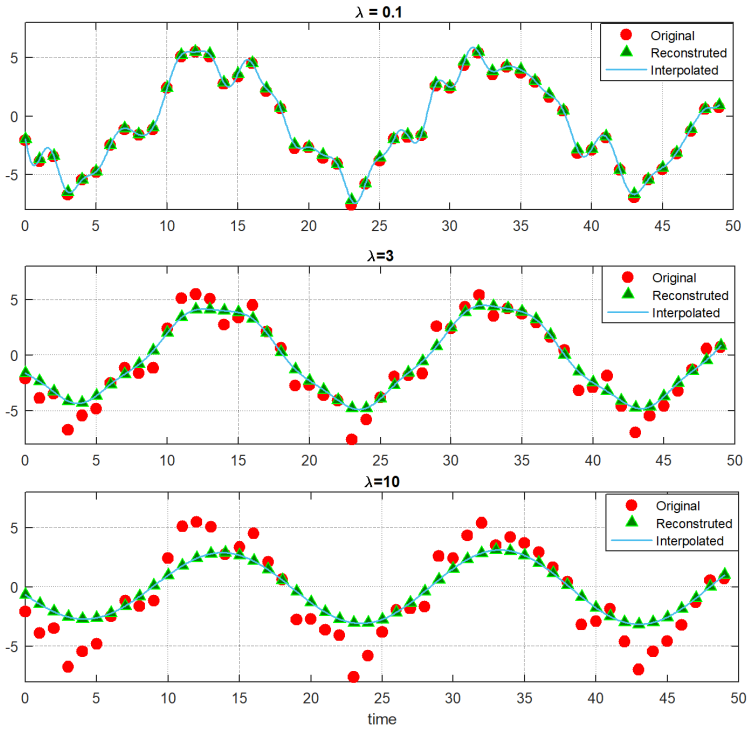
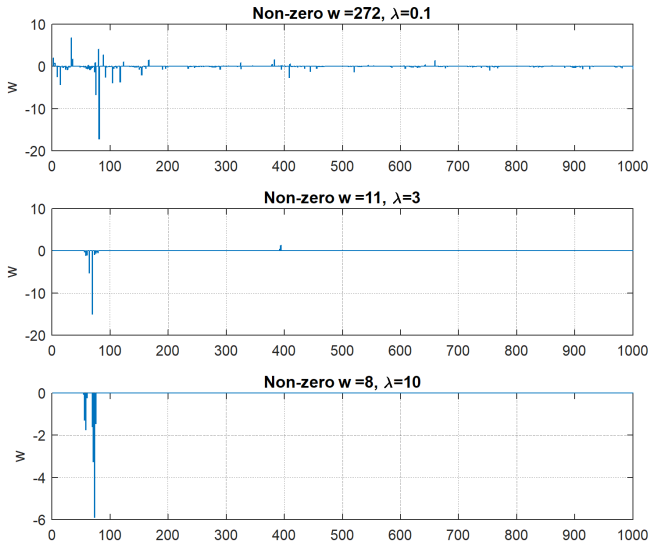
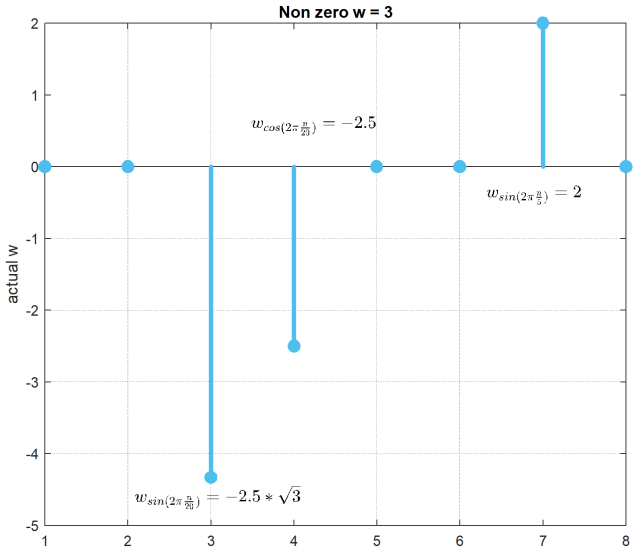


Figure 2: Each plot is depicted for different values of the hyper-parameter $\lambda = 0.1,\ 3,\ 10$. The red circles show the original data points, green triangles show reconstructed data points, while the solid line show the interpolated reconstruction of the data.

In order to compare estimated coefficients with actual non-zero coordinates needed to model the data, I simplified the function to the form of sine and cosine functions to have $f(n) = 5cos(2\pi(\frac{n}{20} + \frac{1}{3})) + 2cos(2\pi(\frac{n}{5} - \frac{1}{4}))$ which leads to three non-zero coefficients in $f_i = 0.05$ and 0.2 for both sine and cosine in $f_i = 0.05$ and sine in $f_i = 0.2$ shown in Fig. (3b). In order to verify this, I created a clean, new set of data points from the main function and estimated the actual coordinates by LS and the code and the results are depicted in Figs. (4, 5), respectively.



(a) Estimated coefficients for different λ values by Lasso.



(b) Actual coefficients estimated by LS.

Figure 3: Non-zero coordinates for the values of the hyper-parameter $\lambda in 0.1, 10, 3$, and compare these to the actual number of nonzero coordinates needed to model the data given the true frequencies.

```
%% actual number of nonzero coordinates task4_b
n = [0:49]';
fi = [0.02, 0.05, 0.1, 0.2] %% we only need 1/20 and 1/5 frequencies

X_reg_new = []
for k=1:length(fi)
    x_temp = [sin(2*pi*fi(k).*n) cos(2*pi*fi(k).*n)];
    X_reg_new = [X_reg_new, x_temp];
end

actual_y =  5*cos(2*pi*(n/20+1/3))+2*cos(2*pi*(n/5-1/4));
w_LS     = (X_reg_new'*X_reg_new)^(-1)*X_reg_new'*actual_y


%% plot
figure;
stem(w_real)
nn = sum(double(w_LS)~=0 )
title(['Number of Non-zero w =',num2str(nn)])

figure;
plot(n,y,'bo')
hold on
plot(n, X_reg_new*w_real,'r')
```

Figure 4: Actual coordinates verification code.

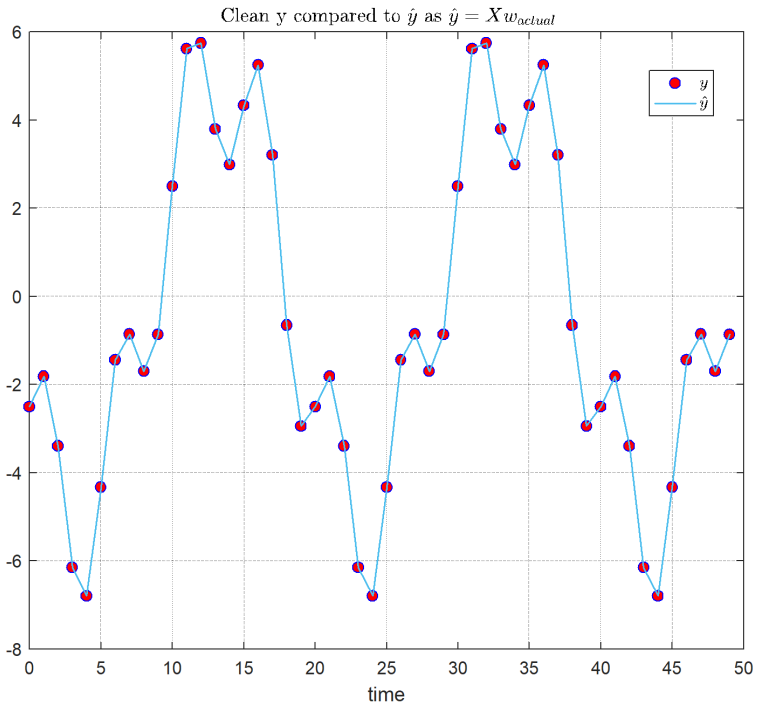


Figure 5: Clean data points and estimated model by actual coordinates.

Task 5: Implement a K-fold cross-validation scheme for the LASSO solver implemented in Task 4. Illustrate your results using the following plots (and make detailed comments for them).

response: For this task, I completed the skeleton_lasso_cv.m as Fig. (6). I assumed $K = 10$ and in each iteration 5 samples are selected randomly without overlapping with other iterations' samples as validation data and the rest of 45 rows used as estimation data set. The optimal λ is 2.1544 in which the minimum validation error is achieved. The results of $RMSE_{val}$ and $RMSE_{est}$ vs. λ values are depicted in Fig. (7) and I have sweeped λ from 0.01 to 10 evenly spaced on log-scale.

```
randomind = randperm(N); % Select random indices for validation and estimation
location = 0; % Index start when moving through the folds
Nval = floor(N/K); % How many samples per fold
hop = Nval; % How many samples to skip when moving to the next fold.
for kfold = 1:K
   valind = randomind((kfold-1)*hop+1:(kfold)*hop) % Select validation indices
   allind = 1:N;
   allind(valind) = [];
   estind = allind; % Select estimation indices
   assert(isempty(intersect(valind,estind)), 'There are overlapping indices in valind and estind!'); % assert empty intersection between valind and es
   wold = zeros(M,1); % Initialize estimate for warm-starting.
   for klam = 1:Nlam
      what = skeleton_lasso_ccd(t(estind),X(estind,:),lambdavec(klam),wold) ;% Calculate LASSO estimate on estimation indices for the current l
      SEval(kfold,klam) = 1/Nval*(t(valind)-X(valind,:)*what)'* (t(valind)-X(valind,:)*what); % Calculate validation error for this estimate
      SEest(kfold,klam) = 1/(N-Nval)*(t(estind)-X(estind,:)*what)'* (t(estind)-X(estind,:)*what); % Calculate estimation error for this estimate
      wold = what; % Set current estimate as old estimate for next lambda-value.
      disp(['Fold: ' num2str(kfold) ', lambda-index: ' num2str(klam)]) % Display current fold and lambda-index.
   end
   location = location+hop; % Hop to location for next fold.
end
MSEval = mean(SEval,1); % Calculate MSE_val as mean of validation error over the folds.
MSEest = mean(SEest,1);% Calculate MSE_est as mean of estimation error over the folds.
[val, ind] = min(MSEval)
lambdaopt = lambdavec(ind); % Select optimal lambda
RMSEval = sqrt(MSEval);
RMSEest = sqrt(MSEest);
wopt =skeleton_lasso_ccd(t,X,lambdaopt,wold); % Calculate LASSO estimate for selected lambda using all data.
```

Figure 6: K-fold cross validation code.

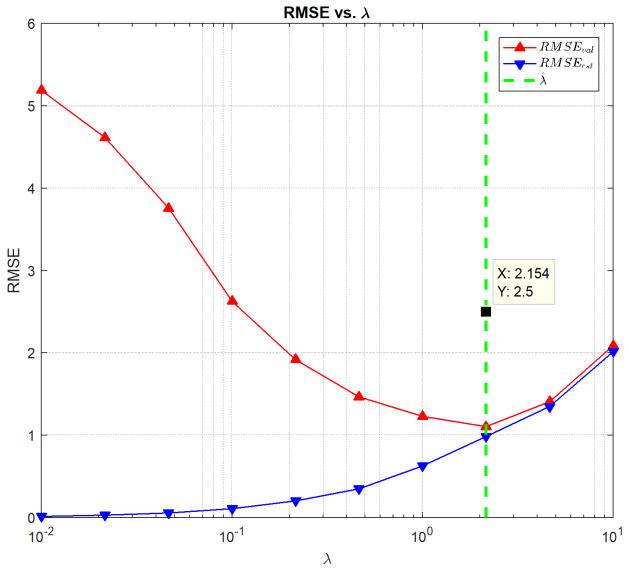


Figure 7: RMSE vs. λ. The estimation RMSE increases by increasing λ parameter while validation RMSE experiences a minimum point in optimum λ. The dashed, green line shows the optimum λ.

The results of part b is shown in Fig. (8) for $\lambda = 0.1,\ 2.1544,\ 10$. For the optimal λ we can see

a good fit of model to the data and number of non-zero coordinates is 14 in this case.

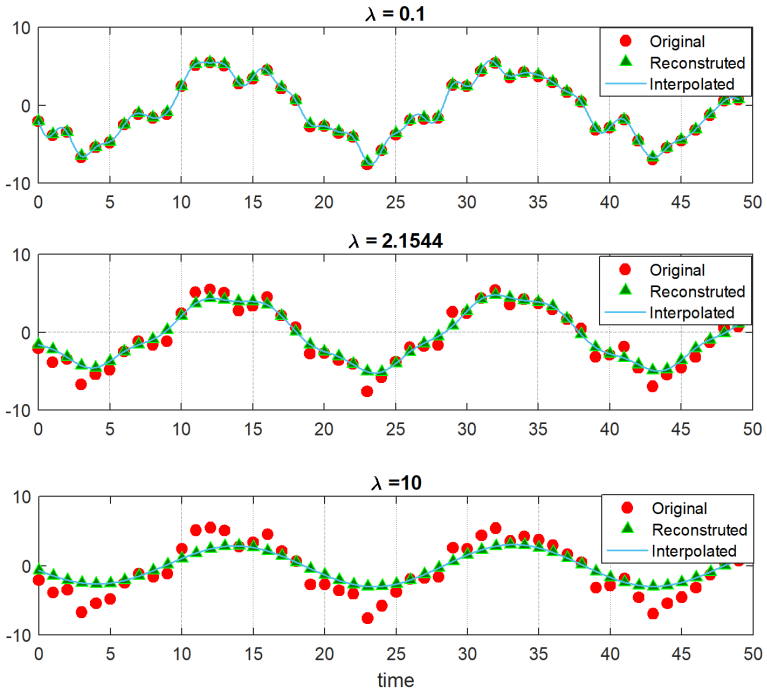


Figure 8: Each plot is depicted for different values of the hyperparameter $\lambda = 0.1,\ 2.1544,\ 10$. The red circles show the original data points, green triangles show reconstructed data points, while the solid line show the interpolated reconstruction of the data.

Task 6: Implement a K-fold cross-validation scheme for the multi-frame audio excerpt Ttrain, similar to Task 5, but modified in order to find one optimal $\hat{\lambda}$ for all frames. Illustrate the results with a plot of $RMSE_{val}$, $RMSE_{est}$, and $\hat{\lambda}$, using the exact same formatting as in the corresponding plot in Task 5, and discuss your findings.

response: For this task, I completed the multi-frame_lasso_code() as Fig. (9). I tried to repeat the K-fold cross-validation for all frames and calculate the validation MSE error in terms of all λ values over all folds for each frame. Then, calculate the MSE over all frames in terms of all λ values. However, running for only one frame, for $K = 4$ and 20 iteration of lasso_ccd() took almost 140 seconds. This was for only one λ value for each fold. Thus, for all frames ($N_{frame} = 55$) and K folds and all λ values I could not run it since it took more than hours. Thus, first I reduce $Xudio$ regression matrix to 352×500 and remove randomly lots of coordinates. I had another problem with the range of λ that for all values I got zero coordinates. I decreased the λ to the range of $0.0001 \rightarrow 0.01$. First, I run it for only one frame and the result is depicted in Fig. (10). Then I sweeped λ from $0.0001 \rightarrow 1$ and let it run for almost 10 hours for all frames and I got the result of Fig. (11).

```
    for klam = 1:Nlam  % Finally loop over the lambda grid
        tic
       what = skeleton_lasso_ccd(t(estind),X(estind,:),lambdavec(klam),wold) ;
       toc
       % Calculate LASSO estimate on estimation indices for the current lambda-value.
       SEval(kfold,klam) = 1/Nval*(t(valind)-X(valind,:)*what)'* (t(valind)-X(valind,:)*what); % Calcula
       SEest(kfold,klam) = 1/(N-Nval)*(t(estind)-X(estind,:)*what)'* (t(estind)-X(estind,:)*what); % Calcu
       wold = what; % Set current estimate as old estimate for next lambda-value.
        test_w  = [test_w , what];
        disp(['Frame: ' num2str(kframe) ', Fold: ' num2str(kfold) ', Hyperparam: ' num2str(klam)]) % Displ
    end
    save('test_w')
   % toc
    cvlocation = cvlocation+cvhop; % Hop to location for next fold.
end

framelocation = framelocation + framehop;% Hop to location for next frame.

MSEval = [MSEval; mean(SEval,1)]; % Average validation error across folds
MSEest = [MSEest ; mean(SEest,1)]; % Average estimation error across folds
end
[val, ind] = min(mean(MSEval,1));
lambdaopt = lambdavec(ind);  % Select optimal lambda
```

Figure 9: Multi frame cross validation code.

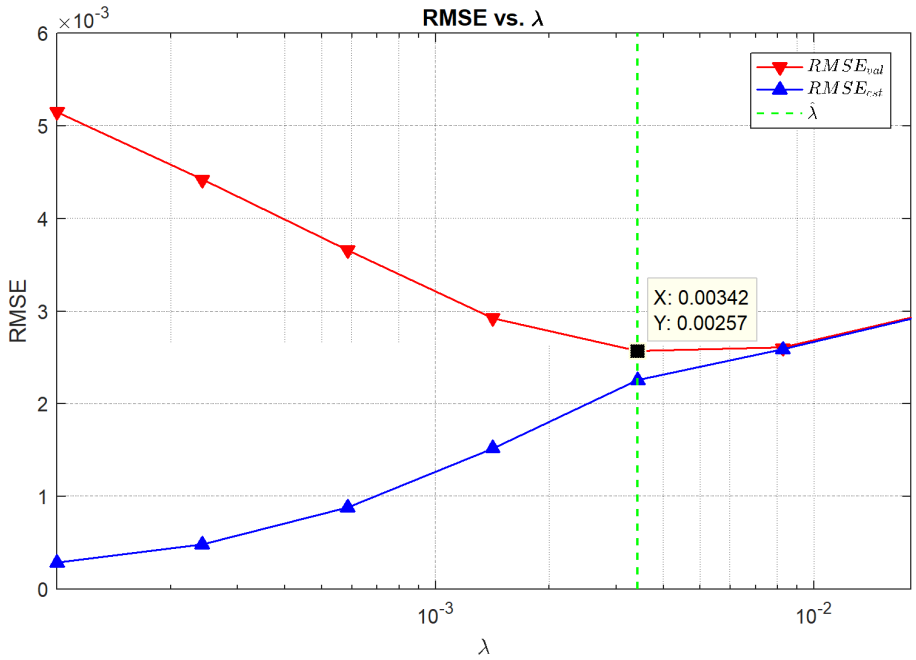


Figure 10: RMSE vs. λ values. The plotted RMSE is only for one frame, 3 folds, and for all λ values sweeped from $0.0001 \rightarrow 0.01$.

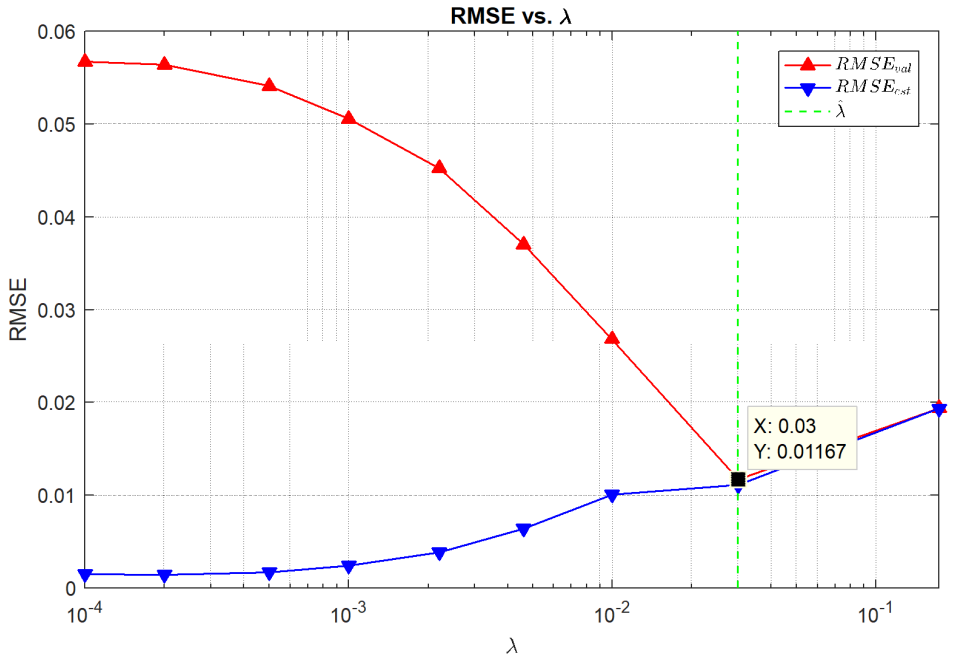
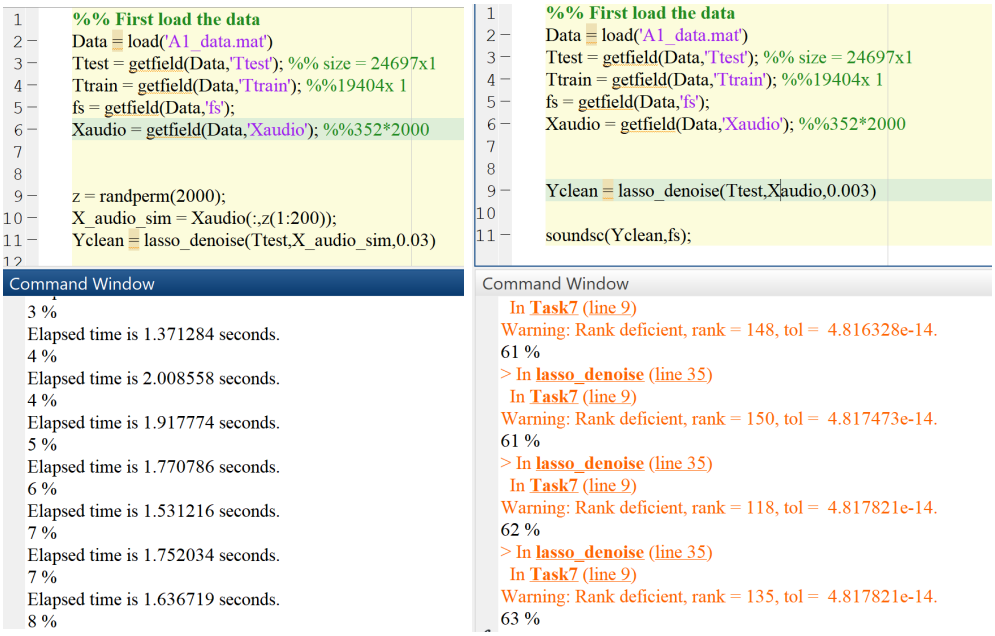


Figure 11: RMSE vs. λ values. The plotted RMSE is averaged over all frames and folds for all λ values. Here, $K = 3$ and λ sweeped from $0.0001 \rightarrow 1$.

Task 7: Using the $\hat{\lambda}$ you've trained in Task 6, use the provided function lasso_denoise() to denoise the test data Ttest. Save your results as save('denoised_audio,'Ytest','fs') and include in your submission. In addition, listen to your output discuss your results. Try whether another choice of λ achieves better noise-cancelling results and detail your findings in the report.

response: The achieved λ from task 6 is 0.003 for only one frame and around 0.03 for all frames. I tried to denoise the Ttest data and I listened to it. For $\lambda = 0.003$ it took near 12 hours for only 60 percents progress. Besides, I got the Rank deficient warning for all steps. For achieving a faster result I used my reduced 352×200 Xaudio regression matrix. The noise was cancelled, but since I removed lots of random coordinates (frequencies) I did not have a nice song after all. By reducing the regression matrix I did not receive the rank deficient warning also. By changing the λ value, I could hear different songs, however, one reason was the random regression matrix that I chose each time and I could hear higher or lower frequencies each time.



(a) The program is running for reduced regression matrix and it is so fast.

(b) The program is running for original regression matrix after 12 hours.

Figure 12: I could only listen to the clean output of the left run since the right one is still running :).