Readme

I wrote this file to explain what the fucntions do and for some of the functions, I will show how I built the functions in detail.

For any part you find ambiguous, please contact me.

Q2

- visualization
  - This function plots a 3-D graph with a L-1 ball and an affine set (shown as a plane).
- project2l1_ball
  - This function finds the projection of a point on L-1 ball.
  - I implemented three different algorithms for it. The first algorithm is based on sorting and it has a time complexity of $O(nlog(n))$. The second one is based on partitioning with time complexity $O(n)$. The third one is similar to the second one, and it also has time complexity $O(n)$.
  - You may check http://www.optimizationonline.org/DB_FILE/2014/08/4498.pdf and http://stanford.edu/~jduchi/projects/jd_ss_ys_paper.pdf for more details of algorithms.
  - For testing any one of the algorithms, uncomment the block you would like to test and comment the rest block.
- project2affine
  - This function finds the projection of a point to a affine set.
  - The formula is the one introduced in lecture $w \leftarrow v - A^T(AA^T)^{-1}(Av - y)$
- POCS
  - This function implements the POCS algorithm provided in homework1.
  - In each iteration, the point is projected to the L-1 ball and then project back to the affine plane.
  - The way I used to check whether converge is to see if the difference of L-1 norm of the updated point is close to or smaller than the L-1 norm of L-1 ball. If it is, then it converges, else not.
  - **This function will return a point lying the intersection of the L-1 ball and the affine set. When there is no intersection, the function will be locked in an infinite loop.**
- POCS_upt
  - The way I used to check whether converge is to see if the updated point is close enough to the point before this iteration. That is to compare norm of the difference of the point before this iteration and the point after, if the norm is smaller then a threshold, then it converges, else not.
  - This function will return a value whether there is intersection of L-1 ball and affine set of not. It is used in a different situation.
- minimize_L1_proj_subgrad
  - This is a function given in demo.
- Q2_a
  - This script is for homework1, Q2, a.
  - This script defines parameters and calls function ‘visualization’.
- Q2_b_1
  - This script is for homework1, Q2, b, i.
  - This script initializes the affine set with random value and then calls function ‘POCS’ to solve fot the intersection point.
  - The parameters given in this script guarantees there will be intersection.

    - The test result and its target value will be printed at control panel of MATLAB.
- Q2_b_2
    - This script is for homework1, Q2, b, ii.
    - This function finds the point on a randomly generalized affine set with least L-1 norm.
    - This function first initializes the L-1 norm of L-1 ball with a small value. And then it calls function 'POCS_upt' to solve for a potential intersection point. When the result is gained, check whether the L-1 norm of result is the same as current L-1 norm. As in 'POCS_upt', the returned point is always lying on the affine set, if the L-1 norm of this point is the same (or less) than that of L-1 ball, they must have an intersection. And the current L-1 norm of L-1 ball is what we want. If the L-1 norm of the result is larger than the L-1 ball, update the L-1 norm of L-1 ball according to the difference of these two values.
    - When the process it done, this script will call 'POCS' one time to check if there is indeed an intersection. (there might be another way to check whether the algorithm is accurate, but this is what I can come up with)
- Q2_b_3
    - This script is for homework1, Q2, b, iii.
    - This script calls the demo first, measures the running time and plots the result. Then it uses the algorithm in Q2, b, ii and measures its running time and plots the result.
    - The reuslts of the demo and the new algorithm are based on a same set of randomly generated A and x.
- Q2_c
    - This script is for homework1, Q2, c.
    - This functions basically run the algorithm in Q2, b, ii and call a modified version of function 'visualization' to plot the figure, to make it as animation.
    - In order to make the process more obvious, I added a 'pause' function, which makes the expanding process looks better but makes the running time longer. (within an accepted time)
    - When the script running is completed, you can turn the graph and you can actually see the intersection point of the L-1 ball and the affine set.
    - Here, to have better illustration, the parameters of affine set are not randomly taken.

Q3

- wavelet_coeffs
    - This is a function given in demo.
- resconstruct_image
    - This is a function given in demo.
- minimize_L1_proj_subgrad
    - The structure of this function is basically the same as demo, I made some changes to it.
    - Reduced maximal iteration times.
    - Changed the way it maps projects the point back to the constraint area. In the demo code, it just applied the matrix form projection. Here the projection is in function form, by calling mapping function and adjoint mapping function to realize projection.
- resconstruction
    - This function inplements all three kinds of sampling required. With different input parameter, it will utilize different sampling method.
    - After sampling, it uses the sampled data as y, together with the mapping A, which is implemented un another function, it calls the function 'minimize_L1_proj_subgrad', to find a x with minimal L-1 norm within the constraint.
- mapping_A

    - This function implements the mapping from x to y. To be specific, it calls function 'reconstuct_image' first to map the wavelet sapce to image. Then it calls function 'fft2', and finally it takes the entries at specific indice. Thus mapping the $x \in \mathbb{C}^{n\times n}$ to $y \in \mathbb{C}^{m}$.
    - One thing to note here, the return value 'hold_off' is only used in test, it is not used in final scripts.
- adj_mapping_A
    - This function implements the adjoint mapping of function 'mapping_A'. To be specific, it puts the vector members back into the relative entries of the matrix, and then calls function 'ifft2' scales the result by its size. And finally, calls function 'wavelet_coeffs' to map the image into wavelet space.
- test
    - A script used for testing whether the adjoint maps are right.
    - In this test script, to make things easier, I used the whole matrix as samples. Or in other words, I skipped the sampling part.
    - After testing, all the adjoint mappings are right.
- preprocess
    - This script compresses the origin image by taking average over a window.
    - The compressed image is only used in testing part, in the final code, I still used the origin image.
- Q3_b_1
    - This script is for homework1, Q3, b, i.
    - This script calls function 'resconstruction' several times with different choice of 'theta' (the sampling rate), plots and saves the test results.
    - This script only uses bernoulli sampling.
    - **If you would like to run it for test, I would recommand you to set the 'theta' to one single value, otherwise it will take quite a long time.**
- Q3_b_2
    - This script is for homework1, Q3, b, ii.
    - This script calls function 'resconstruction' several times with different choice of 'theta' (the sampling rate), plots and saves the test results.
    - This script only uses variable density sampling.
    - **If you would like to run it for test, I would recommand you to set the 'theta' to one single value, otherwise it will take quite a long time.**
- Q3_b_3
    - This script is for homework1, Q3, b, iii.
    - This script calls function 'resconstruction' several times with different choice of 'theta' (the sampling rate) and 'rotation_angle' (the angle between radial lines), plots and saves the test results.
    - This script only uses radial sampling.
    - **If you would like to run it for test, I would recommand you to set the 'theta' to one single value and 'rotation_angle' to the relevant pair, otherwise it will take quite a long time.**
- Q3
    - This script is for homework1, Q3, c.
    - This script calls function 'resconstruction' several times with different choice of 'theta' (the sampling rate) and 'rotation_angle' (the angle between radial lines), plots and saves the test results.
    - Unlike scripts above, for each pair of 'theta' and 'rotation_angle', all three sampling methods are used once to draw a comparison.
    - **This script's running process is very slow. You might want to just check the results instead of running it yourself.**

Folders

- Bernoulli
  Saves the reuslt of bernoulli sampling.

- Variable density
  Saves the result of variable density sampling.
- Radial
  Saves the result of radial sampling.
- Compare
  Saves the result of all three sampling methods within one figure, good for comparing their performance.