Table of Contents


```
clc;
clear;
tic;
```

Reading images

```
groundTruth = imread("../data/mri_image_noiseless.png");
lowNoiseImg = imread("../data/mri_image_noise_level_low.png");
medNoiseImg = imread("../data/mri_image_noise_level_medium.png");
highNoiseImg = imread("../data/mri_image_noise_level_high.png");

groundTruth = im2double(groundTruth);
lowNoiseImg = im2double(lowNoiseImg);
medNoiseImg = im2double(medNoiseImg);
highNoiseImg = im2double(highNoiseImg);
```

Initial error calculation

```
RRMSE1 = RRMSE(groundTruth, lowNoiseImg);
RRMSE2 = RRMSE(groundTruth, medNoiseImg);
RRMSE3 = RRMSE(groundTruth, highNoiseImg);
fprintf("Original errors\n------------------\n");
fprintf("RRMSE between noiseless and low noise images = %f\n",
 RRMSE1);
fprintf("RRMSE between noiseless and medium noise images = %f\n",
 RRMSE2);
fprintf("RRMSE between noiseless and high noise images = %f\n\n",
 RRMSE3);
```

```
Original errors
------------------
RRMSE between noiseless and low noise images = 0.051898
RRMSE between noiseless and medium noise images = 0.131247
RRMSE between noiseless and high noise images = 0.155534
```

Denoising using quadratic loss

```
% hyperparams
```

```
alpha = 0.112;
gamma = 0;     % any arbitrary value

[denoisedImg11, loss11] = gradientDescent(lowNoiseImg, alpha, gamma,
 1);
[denoisedImg12, loss12] = gradientDescent(medNoiseImg, alpha, gamma,
 1);
[denoisedImg13, loss13] = gradientDescent(highNoiseImg, alpha, gamma,
 1);

newRRMSE1 = RRMSE(groundTruth, denoisedImg11);
newRRMSE2 = RRMSE(groundTruth, denoisedImg12);
newRRMSE3 = RRMSE(groundTruth, denoisedImg13);

fprintf("Optimal alpha = %f\n", alpha);

fprintf("Quadratic errors\n-----------------\n");

fprintf("RRMSE between noiseless and denoised img (low noise) :- \n");
fprintf("At alpha = %f\n", newRRMSE1);
[temp, ~] = gradientDescent(lowNoiseImg, 1.2 * alpha, gamma, 1);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha = %f\n", error);
[temp, ~] = gradientDescent(lowNoiseImg, 0.8 * alpha, gamma, 1);
error = RRMSE(groundTruth, temp);
fprintf("At 0.8 * alpha = %f\n", error);

fprintf("RRMSE between noiseless and denoised img (med noise) :- \n");
fprintf("At alpha = %f\n", newRRMSE2);
[temp, ~] = gradientDescent(medNoiseImg, 1.2 * alpha, gamma, 1);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha = %f\n", error);
[temp, ~] = gradientDescent(medNoiseImg, 0.8 * alpha, gamma, 1);
error = RRMSE(groundTruth, temp);
fprintf("At 0.8 * alpha = %f\n", error);

fprintf("RRMSE between noiseless and denoised img (high noise) :-
 \n");
fprintf("At alpha = %f\n", newRRMSE3);
[temp, ~] = gradientDescent(highNoiseImg, 1.2 * alpha, gamma, 1);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha = %f\n", error);
[temp, ~] = gradientDescent(highNoiseImg, 0.8 * alpha, gamma, 1);
error = RRMSE(groundTruth, temp);
fprintf("At 0.8 * alpha = %f\n\n", error);
```

```
Optimal alpha = 0.112000
Quadratic errors
------------------
RRMSE between noiseless and denoised img (low noise) :-
At alpha = 0.049174
At 1.2 * alpha = 0.051767
At 0.8 * alpha = 0.047286
RRMSE between noiseless and denoised img (med noise) :-
```

```
At alpha = 0.114592
At 1.2 * alpha = 0.115834
At 0.8 * alpha = 0.115671
RRMSE between noiseless and denoised img (high noise) :-
At alpha = 0.128276
At 1.2 * alpha = 0.129451
At 0.8 * alpha = 0.129354
```

Denoising using Huber loss

```
% hyperparams
alpha = 0.6;
gamma = 0.02;

[denoisedImg21, loss21] = gradientDescent(lowNoiseImg, alpha, gamma,
 2);
[denoisedImg22, loss22] = gradientDescent(medNoiseImg, alpha, gamma,
 2);
[denoisedImg23, loss23] = gradientDescent(highNoiseImg, alpha, gamma,
 2);

newRRMSE1 = RRMSE(groundTruth, denoisedImg21);
newRRMSE2 = RRMSE(groundTruth, denoisedImg22);
newRRMSE3 = RRMSE(groundTruth, denoisedImg23);

fprintf("Optimal alpha = %f\n", alpha);
fprintf("Optimal gamma = %f\n", gamma);

fprintf("Huber errors\n-----------------\n");

fprintf("RRMSE between noiseless and denoised img (low noise) :- \n");
fprintf("At alpha, gamma = %f\n", newRRMSE1);
[temp, ~] = gradientDescent(lowNoiseImg, 1.2 * alpha, gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(lowNoiseImg, 0.8 * alpha, gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At 0.8 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(lowNoiseImg, alpha, 1.2 * gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 1.2 * gamma = %f\n", error);
[temp, ~] = gradientDescent(lowNoiseImg, alpha, 0.8 * gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 0.8 * gamma = %f\n", error);

fprintf("RRMSE between noiseless and denoised img (med noise) :- \n");
fprintf("At alpha = %f\n", newRRMSE2);
[temp, ~] = gradientDescent(medNoiseImg, 1.2 * alpha, gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(medNoiseImg, 0.8 * alpha, gamma, 2);
error = RRMSE(groundTruth, temp);
```

```
fprintf("At 0.8 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(medNoiseImg, alpha, 1.2 * gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 1.2 * gamma = %f\n", error);
[temp, ~] = gradientDescent(medNoiseImg, alpha, 0.8 * gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 0.8 * gamma = %f\n", error);

fprintf("RRMSE between noiseless and denoised img (high noise) :-
 \n");
fprintf("At alpha = %f\n", newRRMSE3);
[temp, ~] = gradientDescent(highNoiseImg, 1.2 * alpha, gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(highNoiseImg, 0.8 * alpha, gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At 0.8 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(highNoiseImg, alpha, 1.2 * gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 1.2 * gamma = %f\n", error);
[temp, ~] = gradientDescent(highNoiseImg, alpha, 0.8 * gamma, 2);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 0.8 * gamma = %f\n\n", error);
```

```
Optimal alpha = 0.600000
Optimal gamma = 0.020000
Huber errors
------------------
RRMSE between noiseless and denoised img (low noise) :-
At alpha, gamma = 0.045660
At 1.2 * alpha, gamma = 0.052797
At 0.8 * alpha, gamma = 0.043444
At alpha, 1.2 * gamma = 0.047276
At alpha, 0.8 * gamma = 0.044256
RRMSE between noiseless and denoised img (med noise) :-
At alpha = 0.112234
At 1.2 * alpha, gamma = 0.113976
At 0.8 * alpha, gamma = 0.114024
At alpha, 1.2 * gamma = 0.112286
At alpha, 0.8 * gamma = 0.112860
RRMSE between noiseless and denoised img (high noise) :-
At alpha = 0.125533
At 1.2 * alpha, gamma = 0.122617
At 0.8 * alpha, gamma = 0.131138
At alpha, 1.2 * gamma = 0.123993
At alpha, 0.8 * gamma = 0.128150
```

Denoising using DAF loss

```
% hyperparams
alpha = 0.4;
gamma = 0.065;
```

```
[denoisedImg31, loss31] = gradientDescent(lowNoiseImg, alpha, gamma,
 3);
[denoisedImg32, loss32] = gradientDescent(medNoiseImg, alpha, gamma,
 3);
[denoisedImg33, loss33] = gradientDescent(highNoiseImg, alpha, gamma,
 3);

newRRMSE1 = RRMSE(groundTruth, denoisedImg31);
newRRMSE2 = RRMSE(groundTruth, denoisedImg32);
newRRMSE3 = RRMSE(groundTruth, denoisedImg33);

fprintf("Optimal alpha = %f\n", alpha);
fprintf("Optimal gamma = %f\n", gamma);

fprintf("DAF errors\n-----------------\n");

fprintf("RRMSE between noiseless and denoised img (low noise) :- \n");
fprintf("At alpha, gamma = %f\n", newRRMSE1);
[temp, ~] = gradientDescent(lowNoiseImg, 1.2 * alpha, gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(lowNoiseImg, 0.8 * alpha, gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At 0.8 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(lowNoiseImg, alpha, 1.2 * gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 1.2 * gamma = %f\n", error);
[temp, ~] = gradientDescent(lowNoiseImg, alpha, 0.8 * gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 0.8 * gamma = %f\n", error);

fprintf("RRMSE between noiseless and denoised img (med noise) :- \n");
fprintf("At alpha = %f\n", newRRMSE2);
[temp, ~] = gradientDescent(medNoiseImg, 1.2 * alpha, gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(medNoiseImg, 0.8 * alpha, gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At 0.8 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(medNoiseImg, alpha, 1.2 * gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 1.2 * gamma = %f\n", error);
[temp, ~] = gradientDescent(medNoiseImg, alpha, 0.8 * gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 0.8 * gamma = %f\n", error);

fprintf("RRMSE between noiseless and denoised img (high noise) :-
 \n");
fprintf("At alpha = %f\n", newRRMSE3);
[temp, ~] = gradientDescent(highNoiseImg, 1.2 * alpha, gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At 1.2 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(highNoiseImg, 0.8 * alpha, gamma, 3);
```

```
error = RRMSE(groundTruth, temp);
fprintf("At 0.8 * alpha, gamma = %f\n", error);
[temp, ~] = gradientDescent(highNoiseImg, alpha, 1.2 * gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 1.2 * gamma = %f\n", error);
[temp, ~] = gradientDescent(highNoiseImg, alpha, 0.8 * gamma, 3);
error = RRMSE(groundTruth, temp);
fprintf("At alpha, 0.8 * gamma = %f\n\n", error);
```

```
Optimal alpha = 0.400000
Optimal gamma = 0.065000
DAF errors
------------------
RRMSE between noiseless and denoised img (low noise) :-
At alpha, gamma = 0.043564
At 1.2 * alpha, gamma = 0.045085
At 0.8 * alpha, gamma = 0.042841
At alpha, 1.2 * gamma = 0.044675
At alpha, 0.8 * gamma = 0.042884
RRMSE between noiseless and denoised img (med noise) :-
At alpha = 0.111874
At 1.2 * alpha, gamma = 0.111291
At 0.8 * alpha, gamma = 0.112522
At alpha, 1.2 * gamma = 0.111437
At alpha, 0.8 * gamma = 0.112003
RRMSE between noiseless and denoised img (high noise) :-
At alpha = 0.123704
At 1.2 * alpha, gamma = 0.122073
At 0.8 * alpha, gamma = 0.127085
At alpha, 1.2 * gamma = 0.122707
At alpha, 0.8 * gamma = 0.125797
```

Results

```
%%%% Low Noise

figure;

subplot(1, 5, 1);
imshow(uint8(255 * groundTruth));
title("Without noise");
subplot(1, 5, 2);
imshow(uint8(255 * lowNoiseImg));
title("Low noise");
subplot(1, 5, 3);
imshow(uint8(255 * denoisedImg11));
title("Denoising using quadratic loss");
subplot(1, 5, 4);
imshow(uint8(255 * denoisedImg21));
title("Denoising using huber loss");
subplot(1, 5, 5);
imshow(uint8(255 * denoisedImg31));
```

```
title("Denoising using DAF loss");

colormap('gray');

%%%% Medium noise

figure;

subplot(1, 5, 1);
imshow(uint8(255 * groundTruth));
title("Without noise");
subplot(1, 5, 2);
imshow(uint8(255 * medNoiseImg));
title("Medium noise");
subplot(1, 5, 3);
imshow(uint8(255 * denoisedImg12));
title("Denoising using quadratic loss");
subplot(1, 5, 4);
imshow(uint8(255 * denoisedImg22));
title("Denoising using huber loss");
subplot(1, 5, 5);
imshow(uint8(255 * denoisedImg32));
title("Denoising using DAF loss");

colormap('gray');

%%%% High noise

figure;

subplot(1, 5, 1);
imshow(uint8(255 * groundTruth));
title("Without noise");
subplot(1, 5, 2);
imshow(uint8(255 * highNoiseImg));
title("High noise");
subplot(1, 5, 3);
imshow(uint8(255 * denoisedImg13));
title("Denoising using quadratic loss");
subplot(1, 5, 4);
imshow(uint8(255 * denoisedImg23));
title("Denoising using huber loss");
subplot(1, 5, 5);
imshow(uint8(255 * denoisedImg33));
title("Denoising using DAF loss");

colormap('gray');
```

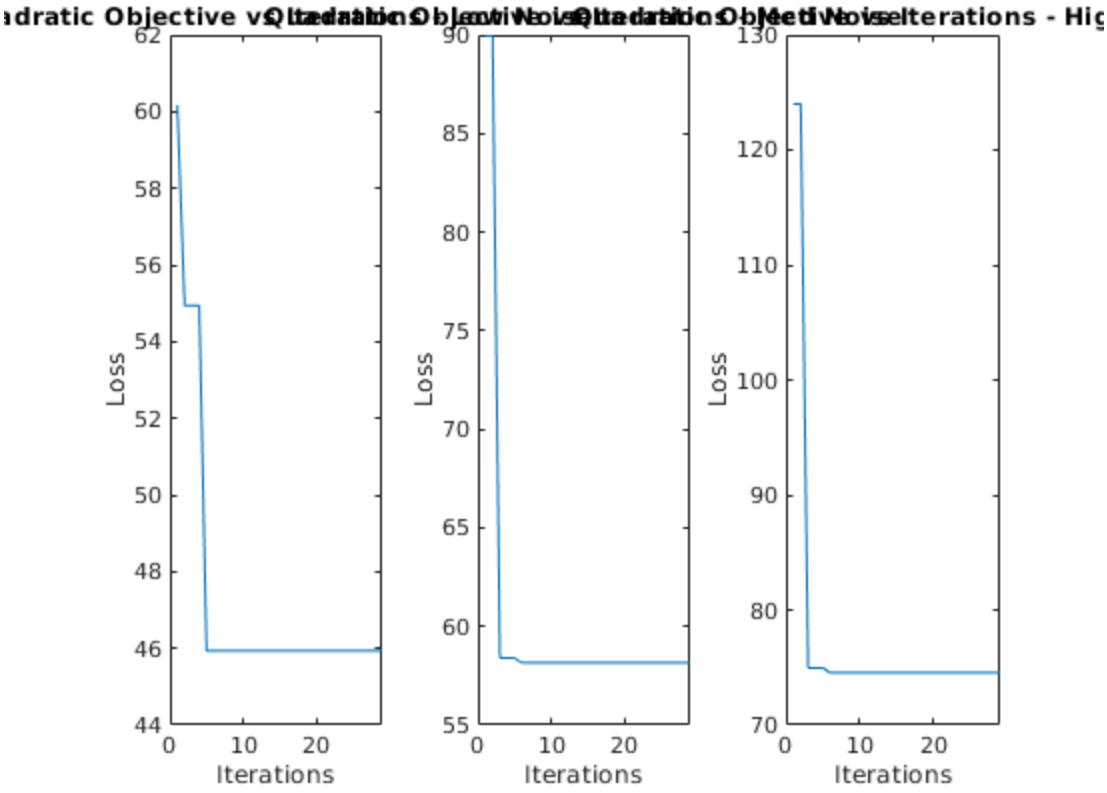

Without noise
Low noise
Denoising using DAF loss

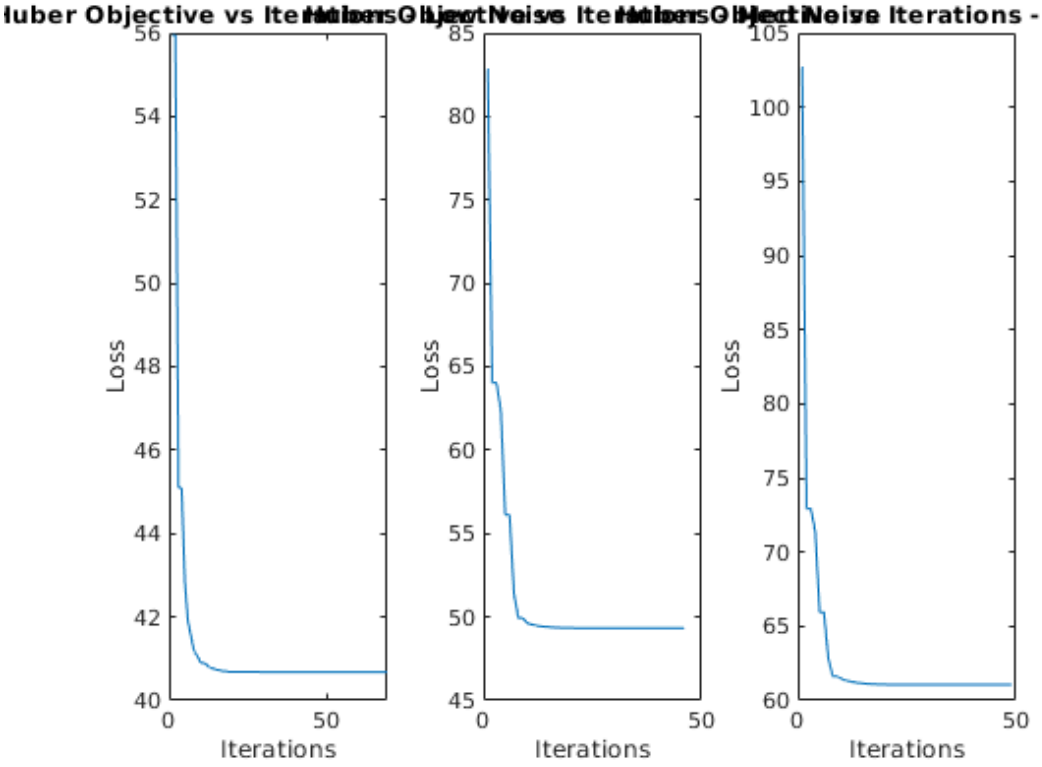

Without noise
Medium noise
Denoising using DAF loss

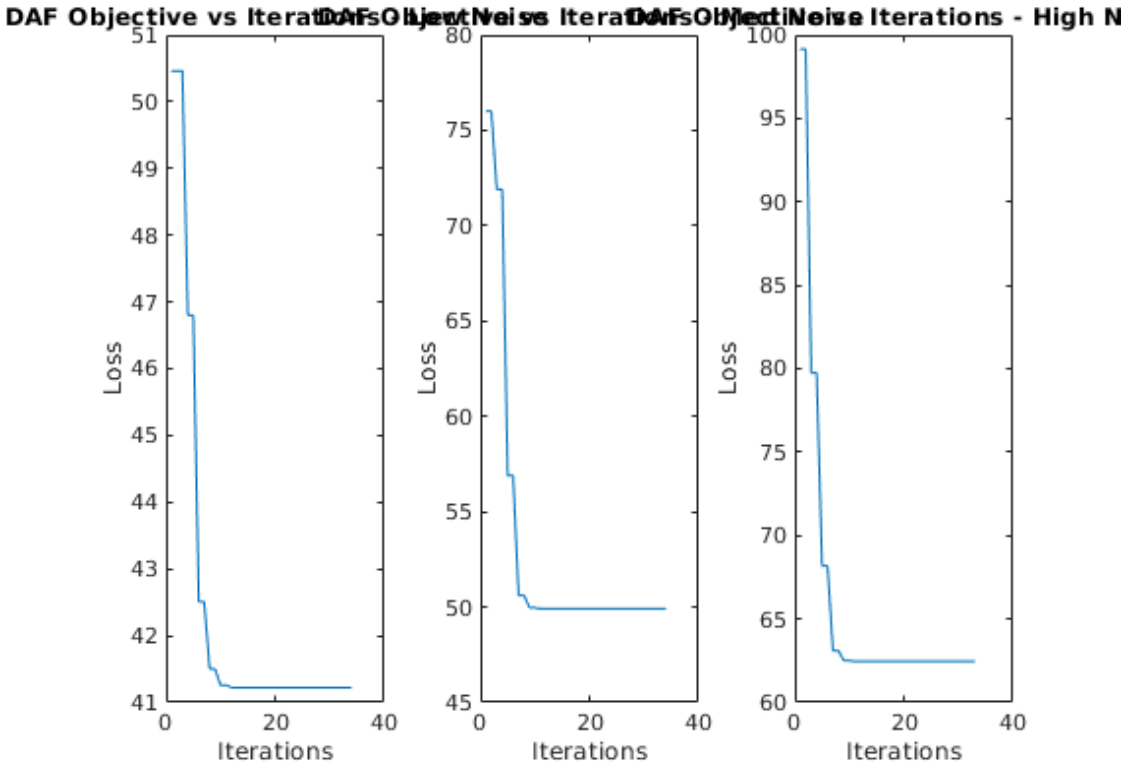
Loss vs Iterations

```
figure;

subplot(1, 3, 1);
plot(loss11);
xlabel("Iterations");
ylabel("Loss");
title("Quadratic Objective vs Iterations - Low Noise");

subplot(1, 3, 2);
plot(loss12);
xlabel("Iterations");
ylabel("Loss");
title("Quadratic Objective vs Iterations - Med Noise");

subplot(1, 3, 3);
plot(loss13);
xlabel("Iterations");
ylabel("Loss");
title("Quadratic Objective vs Iterations - High Noise");

figure;

subplot(1, 3, 1);
plot(loss21);
```

```
xlabel("Iterations");
ylabel("Loss");
title("Huber Objective vs Iterations - Low Noise");

subplot(1, 3, 2);
plot(loss22);
xlabel("Iterations");
ylabel("Loss");
title("Huber Objective vs Iterations - Med Noise");

subplot(1, 3, 3);
plot(loss23);
xlabel("Iterations");
ylabel("Loss");
title("Huber Objective vs Iterations - High Noise");

figure;

subplot(1, 3, 1);
plot(loss31);
xlabel("Iterations");
ylabel("Loss");
title("DAF Objective vs Iterations - Low Noise");

subplot(1, 3, 2);
plot(loss32);
xlabel("Iterations");
ylabel("Loss");
title("DAF Objective vs Iterations - Med Noise");

subplot(1, 3, 3);
plot(loss33);
xlabel("Iterations");
ylabel("Loss");
title("DAF Objective vs Iterations - High Noise");
```


Loss
Iterations
Loss
Iterations
Loss
Iterations


Loss
Iterations
Loss
Iterations
Loss
Iterations




```
toc;
```

```
Elapsed time is 7.050809 seconds.
```

Published with MATLAB® R2019b