Deepayan Bhadra - hmwk7 Solutions

Q1: Downloading FASTA solver (No Code)

Q2) Use the FASTA code to solve the sparse logistic regression problem

```
%% Q2 Solve the sparse logistic regression problem

[D, c] = create_classification_problem(500, 5, 1);
G = randn(500,5);
for i = 1:5
G(:,i) = G(:,i)./norm(G(:,i));
end
mu = 1;
D_hat = [D G];

f = @(x) logreg_objective(x,D_hat,c);
gradf = @(x) logreg_grad(x,D_hat,c);
g = @(x) mu*norm(x,1);
proxg = @(x,t) sign(x).*max(abs(x)-mu*t,0);
%A = diag(c)*D_hat;
A = eye(10);
opts.tol = 1e-8;
x0 = zeros(10,1);
[sol, outs, opts] = fasta(A, A', f, gradf, g, proxg, x0,opts);
plot(outs.residuals)
xlabel('No. of iterations');
ylabel('Residual value');
plot(sol)
title('Solution (Sparse)')
```

Q3: Solving the non-convex non-negative matrix factorization problem

```
n = 75;
A = zeros(50, 3);
A(randperm(numel(A), n)) = 1;
```

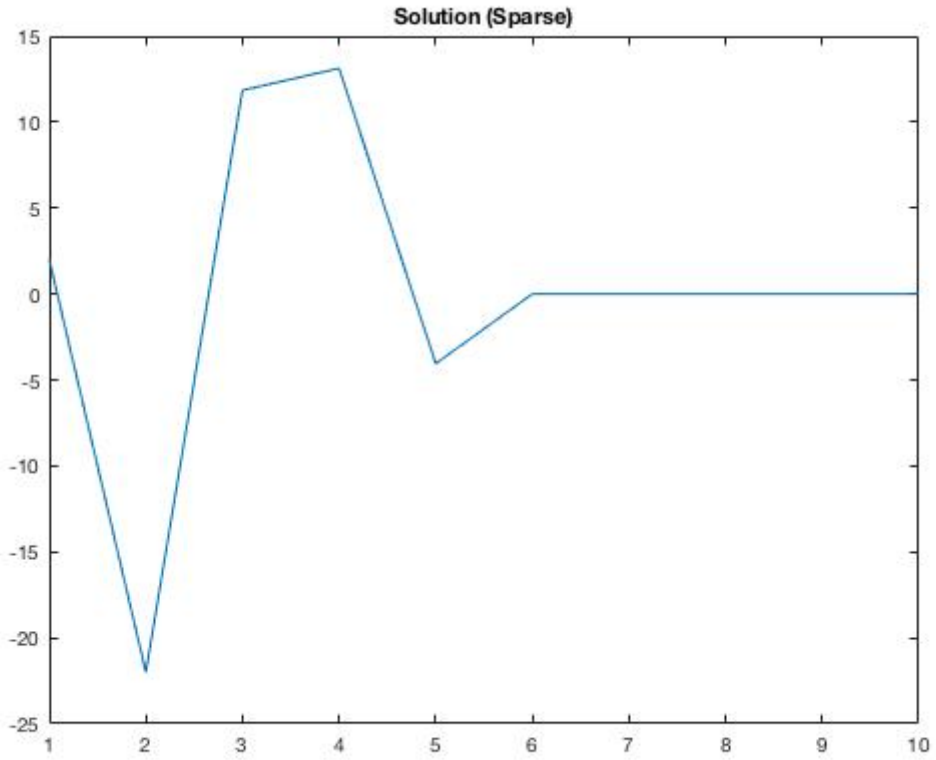

Figure 1: Q2 Sparse Solution

```
B = zeros(50, 3);
B(randperm(numel(B), n)) = 1;
S = A*B';

X0 = randn(50,3);
Y0 = randn(50,3);
mu = 0;
opts.tol = 1e-8;

[Xsol,Ysol, outs] = fasta_nonNegativeFactorization(S, X0, Y0, mu, opts);
plot(outs.residuals)
xlabel('# of iterations');
ylabel('Residual value');
```

The recovery method works because Xsol*Ysol' is almost equal to D

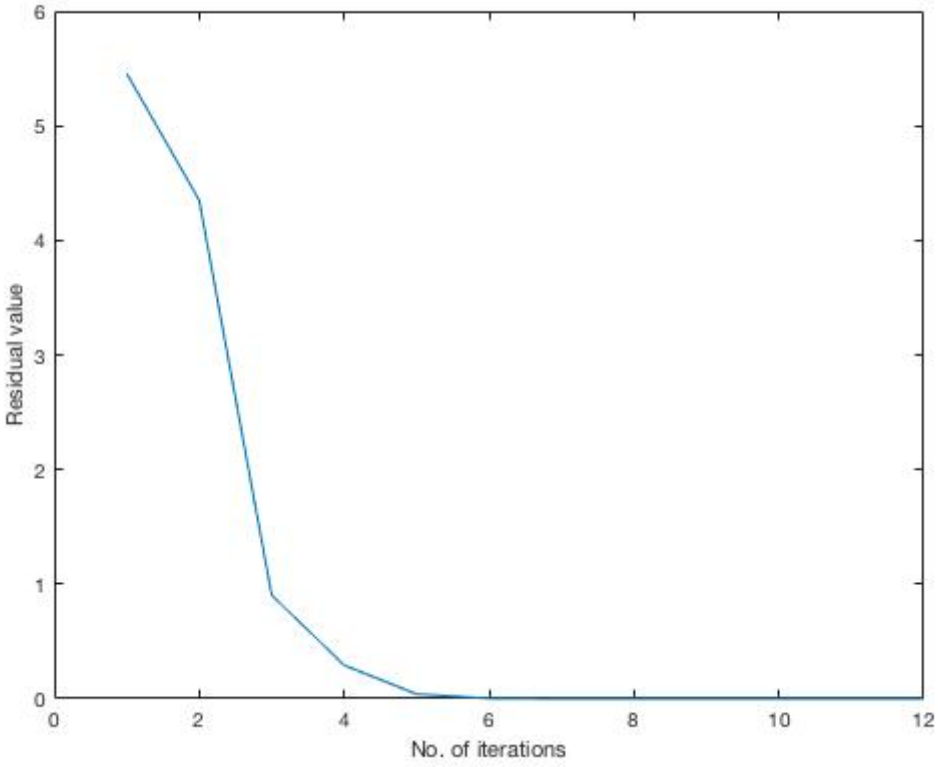

Figure 2: Q2 Residual values

Q4 Deblurring Algorithm with ADMM

```
N = 128;
I = double(imread('lena.jpg'));
figure(1)
imshow(I,[])
blur = fspecial('gaussian',N,3);
blur = fftshift(blur);
D = fft2(blur);
b = ifft2(D.*fft2(I));
figure(2)
imshow(b,[])

yk = randn(N,N,2);
lk = randn(N,N,2);

kernel = zeros(N);
kernel(1,1) = 1; % backward difference
```

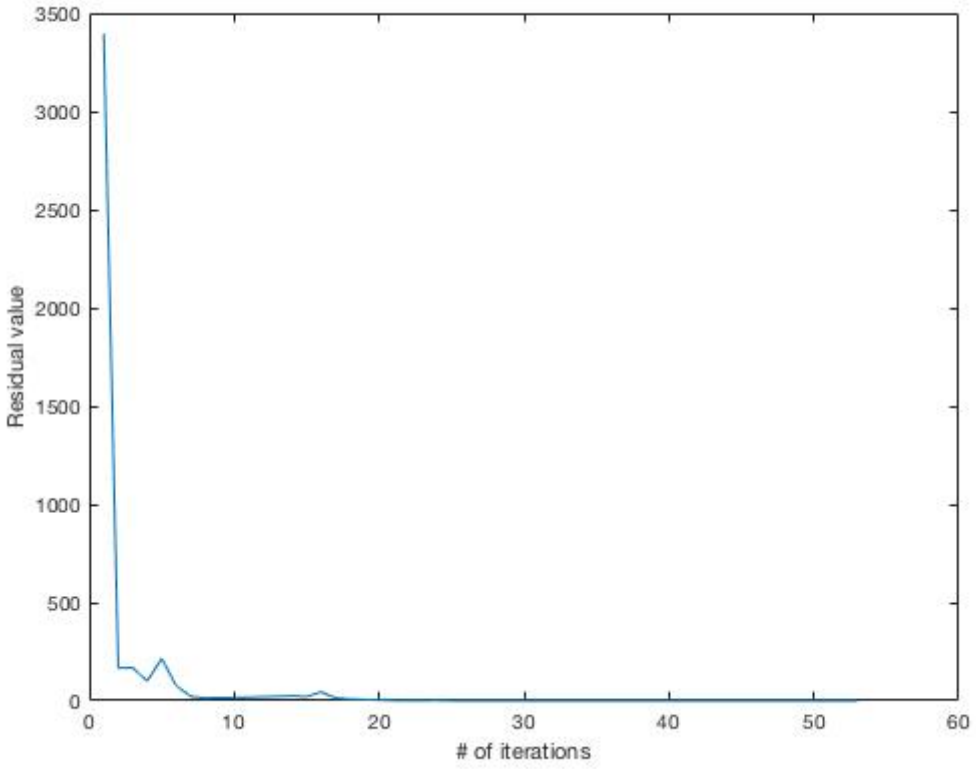

Figure 3: Q3 Sparse Solution

```
kernel(1,2) = -1;
Dx = fft2(kernel);
kernel = zeros(N);
kernel(1,1) = 1;
kernel(2,1) = -1;
Dy = fft2(kernel);

K2 = abs(Dx)^2+abs(Dy)^2;
tau = mean(reshape(abs(D).^2,[],1))/mean(reshape(K2,[],1));
res = 1e8;
mu = 1;
iteration = 1;
prox = @(x,t) sign(x).*max(abs(x)-mu*t,0);

while (res > 1e-6);
xk = real(ifft2((1\(abs(D)^2+tau*K2)))*fft2(ifft2(conj(D))*fft2(b)-div2d(lk)+tau*div2d(yk)));
yk = prox(grad2d(xk) + lk/tau, mu/tau);
lk = lk + tau*(grad2d(xk)-yk);
```

```
res = [res norm(reshape(yk,[],1)-reshape(grad2d(xk),[],1))];
iteration = iteration+1;
end
figure(3)
imshow(xk,[])

semilogy(res)
```

Figure 4: Q4 Original Image

Figure 5: Q4 Blurred Image

Figure 6: Q4 Deblurred Image

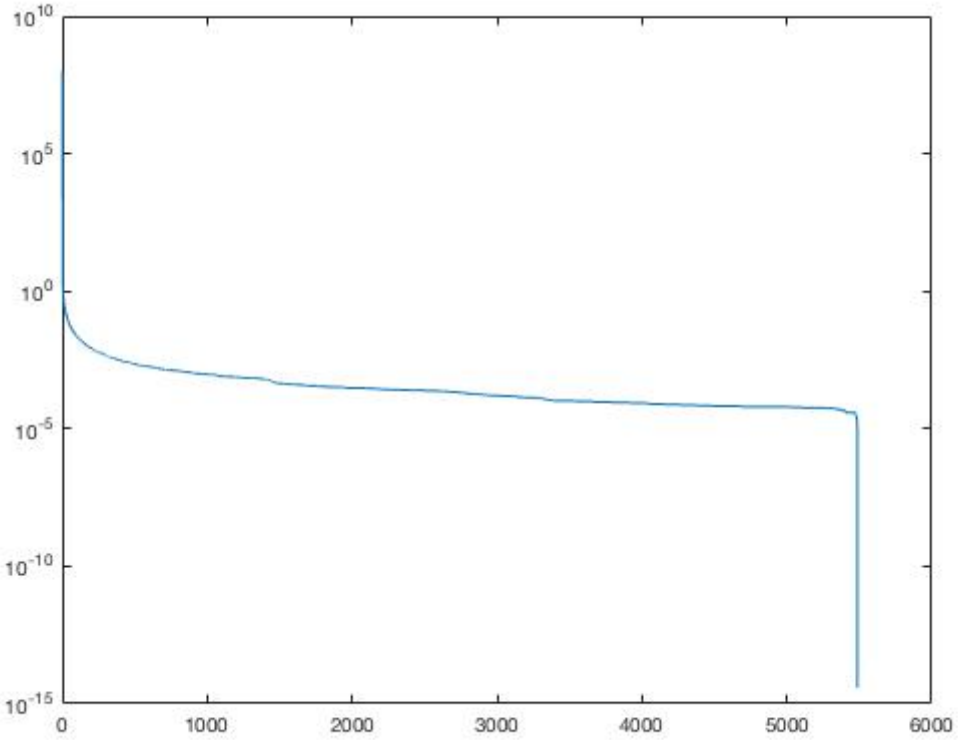

Figure 7: Q4 Residuals