Load a sample from MNIST dataset

```
load("mnist.mat")

idx = 23053;

x = reshape (X(: , idx) , [28 28]).' ;
label = labels (idx) ;
```

Visualize the loaded sample

```
figure;
imshow(x,[]);
title(["Digit label: ", num2str(label)]);
```

Blurring by convolution with K matrix

```
n = -2:2;

Z = 0; % normalization constant
for m1 = n
    for m2 = n
        Z = Z + exp(-(m1^2+m2^2)/2);
    end
end

K = zeros(5, 5);

for n1 = n % kernel matrix
    for n2 = n
        K(n1+3, n2+3) = 1/Z*exp(-(n1^2+n2^2)/2);
    end
end

y = conv2(x, K);

figure;
imshow(y,[]);
title(["Blurred image: ", num2str(label)]);
```

Blurred image:
8

Obtain a matrix A that will have the same affect with 2d convolution when multiplied by x using "lazy man's method"

```
x_vec = x(:);

[height_x, width_x] = size(x);
x_vec_size = numel(x_vec);
y_vec_size = numel(y(:));

A = zeros(y_vec_size, x_vec_size);

for i = 1:x_vec_size
    temp_x = zeros(height_x, width_x);
    temp_x(i) = 1;

    temp_y = conv2(temp_x, K);

    A(:, i) = temp_y(:)';
end

y_vec = A * x_vec;
norm(y - reshape(y_vec , [32 32]))
```

```
ans = 1.1060e-13
```

```
imshow(reshape(y_vec , [32 32]),[]);
```

Obtaining original sample by using Moore Penrose pseuode-inverse with vectorized y

```
x_mnls_tilde = pinv(A)*y_vec;
imshow(reshape(x_mnls_tilde,  [height_x width_x]), []);
```

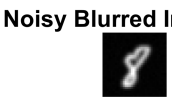
```
diff_image = x - reshape(x_mnls_tilde, [height_x, width_x]);
mse = mean(diff_image(:).^2);
```

Adding gaussian white noise

```
w = randn(size(y));
y_tilde = y + w;

figure;
imshow(y, []);
title('Original Blurred Image (y)');
```


```
figure;
imshow(y_tilde, []);
title('Noisy Blurred Image (ỹ)');
```




```
numerator = norm(y_tilde(:) - y(:));
denominator = norm(y(:));
normalized_difference = numerator / denominator
```

```
normalized_difference = 0.0194
```

Using pseudo-inverse to obtain the x from y with gaussian noie

```
x_mnls_tilde = pinv(A)*y_tilde(:);
imshow(reshape(x_mnls_tilde,  [height_x width_x]), []);
```

```
diff_image = x - reshape(x_mnls_tilde, [height_x, width_x]);
mse = mean(diff_image(:).^2)
```

```
mse = 6.9027e+04
```

Observing singular values

```
[U, S, V] = svd(A);
singular_values = diag(S);
singular_values_A_pseudo = 1 ./ singular_values;
k_A = max(singular_values) / min(singular_values);

fprintf('Condition Number (kappa(A)): %e\n', k_A);
```

```
Condition Number (kappa(A)): 1.775351e+03
```

```
fprintf('Singular Values of A: \n');
```

```
Singular Values of A:
```

```
fprintf("greatest singuler value: ")
```

```
greatest singuler value:
```

```
disp(singular_values(1));
```

```
    0.9897
```

```
fprintf("smallest singuler value: ")
```

```
smallest singuler value:
```

```
disp(singular_values(784));
```

```
   5.5747e-04
```

```
fprintf('Singular Values of A† (Pseudo-Inverse): \n');
```

```
Singular Values of A† (Pseudo-Inverse):
```

```
fprintf("greatest singuler value: ")
```

```
greatest singuler value:
```

```
disp(singular_values_A_pseudo(1));
```

```
    1.0104
```

```
fprintf("smallest singuler value: ")
```

```
smallest singuler value:
```

```
disp(singular_values_A_pseudo(784));
```

```
    1.7938e+03
```

Using regularization to counter noise amplification due too pseudo-inverse

```
lambda = 0.01; % best = 0.0013

x_reg = (A' * A + lambda * eye(size(A, 2))) \ (A' * y_tilde(:));

x_reg_image = reshape(x_reg, [height_x, width_x]);
figure;
imshow(x_reg_image, []);
title(['Regularized Solution (λ = ', num2str(lambda), ')']);
```


```
reg_error = norm(x_reg - x(:)) / norm(x(:))
```

```
reg_error = 0.2392
```

Obtain z by downsampling y

```
z = y(1 : 2 : end, 1 : 2 : end);

figure;
imshow(z,[]);
```

Obtaining a B matrix which is the counterpart of A marix found before for the downsampled z

```
[height_z, width_z] = size(z);
z_vec_size = numel(z);

B = zeros(z_vec_size, x_vec_size);

for i = 1:x_vec_size
    temp_x = zeros(height_x, width_x);
    temp_x(i) = 1;

    temp_z = conv2(temp_x, K);
    temp_z = temp_z(1 : 2 : end, 1 : 2 : end);

    B(:, i) = temp_z(:)';
end

z_vec = B * x_vec;
norm(z - reshape(z_vec , [16 16]))
```

```
ans = 5.8545e-14
```

```
imshow(reshape(z_vec , [16 16]),[]);
```

Obtaining x with pseudo-inverse from z

```
x_mnls_tilde = pinv(B)*z_vec;
imshow(reshape(x_mnls_tilde,  [height_x width_x]), []);
```

```
diff_image = x - reshape(x_mnls_tilde, [height_x, width_x]);
mse = mean(diff_image(:).^2);
```