4 Least Squares Minimization Problems

So far we dealt with linear systems that have n equations for n unknowns. In many real-life scenarios, this is not the case, and we have more equations than variables. Usually in this case there is no true solution that fulfils all equations. We wish only to get the best vector that approximates all the equations is some "optimal" way. In other words, we wish to find a vector that minimizes the discrepancy of the equations in some norm. In this course we will focus of the ℓ_2 norm, which is the most popular and easy-to-handle norm. In this section, we will consider only real-valued problems.

4.1 Least squares minimization - full rank case

4.1.1 Linear Regression Example

Example 9 (A classic example: best linear approximation from measurements.)**.** *Suppose that you are given with n measurements (x_i, y_i) and wish to find the line that defines $\{y_i\}$ given $\{x_i\}$. That is, find scalars a, b so that $ax_i + b \approx y_i$ for all i in an optimal way. This can be written in matrix form*

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

A common choice for the optimality measure is the ℓ_2 norm. We get a problem

$$\min_{a,b} \sum (ax_i + b - y_i)^2,$$

or

$$\min_{a,b} \left\| \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \right\|_2. \tag{20}$$

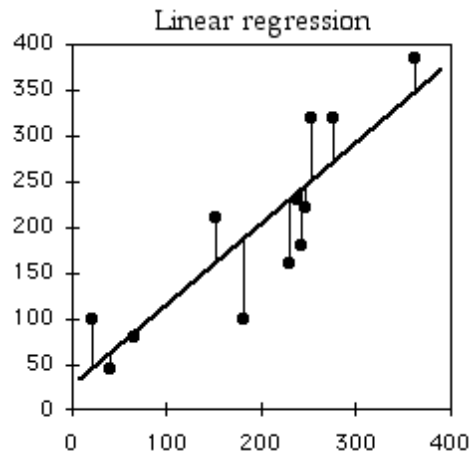

Figure 1: Linear regression - best linear approximation from measurements in a least squares sense.

This problem is also called the linear regression, and boils to the above least squares minimization. Figure 1 illustrates the example. To solve the problem, we can set the derivative of (20) *with respect to* a *and* b *to zero, and find the optimal parameters.*

4.1.2 A general derivation of least squares.

Let $A \in \mathbb{R}^{m\times n}$ be of rank n with $m \geq n$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$. We wish to find $\mathbf{x}$ that minimizes the discrepancy vector $\mathbf{r}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$ in a least squares sense, that is minimize $\|\mathbf{r}(\mathbf{x})\|_2$. Since we are interested in the minimizer $\mathbf{x}$ and not in the minimal value of $\mathbf{r}(\mathbf{x})$, we may minimize the squared objective $\|\mathbf{r}(\mathbf{x})\|_2^2$, since it will lead to the same solution. We get the problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2, \tag{21}$$

where $\arg\min\{f\}$ denotes the argument that minimizes the objective f rather than the minimal value of f as in $\min\{f\}$. If $m = n$ and A is full rank, then there is a solution such that $A\mathbf{x} = \mathbf{b}$ (which is $A^{-1}\mathbf{b}$), and the minimization will turn $\mathbf{r}(\mathbf{x})$ to be 0. So will be the case if $\mathbf{b}$ is in the range of A. In any case, if $\mathbf{r}(\mathbf{x}) = 0$ can be achieved or not, the least squares problem is well-defined and has an optimal solution.

4.1.3 Least squares through derivatives of linear and quadratic functions

Now we will see how to solve the least squares minimization. This will be done by zeroing the first derivative of 21. It is known that an extremum point of function $f(\mathbf{x})$ is a point where the first derivative of f, i.e., its gradient, is equal to the zero vector.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \mathbf{0}.$$

We will see the following examples that will help us to get the gradient of (21).

1. As a simple example, consider the linear function $f(\mathbf{x}) = \mathbf{v}^\top \mathbf{x}$ for two vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$. $f(\mathbf{x}) = \sum_{i=1}^n x_i v_i$, and therefore, $\frac{\partial f}{\partial x_i} = v_i$. This means that $\nabla f = \mathbf{v}$.

2. Consider the quadratic function $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$. $f(\mathbf{x}) = \sum_{i=1}^n x_i^2$, and therefore, $\frac{\partial f}{\partial x_i} = 2x_i$. This means that $\nabla f = 2\mathbf{x}$. We can get the same result using the formula $(uv)' = u'v + v'u$ and applying it on $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{x}$, each time referring to one variable $\mathbf{x}$ as a constant, using the previous linear example.

3. Consider the function $f(\mathbf{x}) = \mathbf{v}^\top A^\top A\mathbf{x}$ for $\mathbf{v}, \mathbf{x} \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m\times n}$. The vector $\mathbf{v}^\top A^\top A$ is just a constant vector, so like the first linear case, we will get $\nabla f = A^\top A\mathbf{v}$.

4. Consider the quadratic function $f(\mathbf{x}) = \mathbf{x}^\top A^\top A\mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m\times n}$. This time, we will twice refer to the vector $\mathbf{x}^\top A^\top A$ as a constant vector, and take the derivative with respect to the other one. Similarly to the previous cases we will get $\nabla f = 2A^\top A\mathbf{x}$.

Rewriting our LS problem in Eq. (21) in a more explicit form we get

$$f(\mathbf{x}) = \|A\mathbf{x} - \mathbf{b}\|_2^2 = (A\mathbf{x} - \mathbf{b})^\top (A\mathbf{x} - \mathbf{b}) = \mathbf{x}^\top A^\top A\mathbf{x} - 2\mathbf{b}^\top A\mathbf{x} + \mathbf{b}^\top \mathbf{b}.$$

By the rules above we get that

$$\nabla f(\mathbf{x}) = 2A^\top A\mathbf{x} - 2A^\top \mathbf{b},$$

and setting $\nabla f(\mathbf{x}) = \mathbf{0}$ will lead to the system

$$A^\top A\mathbf{x} = A^\top \mathbf{b},$$

which is called *the normal equation.* Assuming that the matrix $(A^\top A)$ is non-singular, the minimizer of the LS will be

$$\hat{\mathbf{x}} = (A^\top A)^{-1} A^\top \mathbf{b}.$$

We assumed above that A is full rank and hence the matrix $A^\top A$ is invertible. However, if the matrix is singular, then we may have infinitely many solutions, and each one satisfying the normal equations is a critical point of the objective in the LS problem. As mentioned before, by zeroing the gradient of f we found an extremum point of f. In the case of a convex quadratic f like the one here, any point that zeroes the gradient is a global minimum.

Example 10 (Algebraic Least Squares)**.** *Suppose that we need to minimize* $\|A\mathbf{x} - \mathbf{b}\|_2$ *with*

$$A = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 23 \\ 15 \\ 39 \end{bmatrix}.$$

Using the normal equations, we will form $A^\top A$*, and* $A^\top \mathbf{b}$*, and solve* $A^\top A\mathbf{x} = A^\top \mathbf{b}$*:*

$$A^\top A = \begin{bmatrix} 4 & 8 & 4 \\ 8 & 20 & 24 \\ 4 & 24 & 84 \end{bmatrix}, \quad A^\top \mathbf{b} = \begin{bmatrix} 47 \\ 171 \\ 417 \end{bmatrix}, \quad \mathbf{x} = (A^\top A)^{-1} A^\top \mathbf{b} = \begin{bmatrix} 0.375 \\ 3.75 \\ 3.875 \end{bmatrix}.$$

Note that the residual norm in this example comes out

$$\mathbf{r} = A\mathbf{x} - \mathbf{b} = \begin{bmatrix} -0.25 \\ 0.25 \\ 0.25 \\ -.25 \end{bmatrix}. \tag{22}$$

Here we got an equal error (in magnitue) in all equations, but that will not always be the

case. Also, not surprisingly,

$$A^\top \mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{23}$$

That is, the residual is orthogonal to the columns of A. That is by the construction of the normal equations

$$A^\top (A\mathbf{x} - \mathbf{b}) = 0.$$

Note that on a computer, the vector of zeros there will be vector of small values due to round-off errors. In this example, in double precision, it happened to be on our computer:

$$A^\top \mathbf{r} = \begin{bmatrix} -0.6 \\ -2.0 \\ -5.8 \end{bmatrix} \cdot 1e - 14.$$

This is a "numerical zero".

Example 11 (Linear regression continued)**.** *Assume again that we are given with m measurements (x_i, y_i), $x_i \in [0, 1]$, and wish to find the line that defines $\{y_i\}$ given $\{x_i\}$. We have the following minimization*

$$\arg\min_{a,b} \left\| \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \right\|_2 .$$

Denote this problem as $\arg\min \|A\mathbf{x} - \mathbf{b}\|$*, then the minimizer of this problem will be* $(A^\top A)^{-1} A^\top \mathbf{b}$ *or:*

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & n \end{bmatrix}^{-1} \begin{bmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{bmatrix}.$$

The following code shows an example of linear regression. The output image appears in Fig. 2.

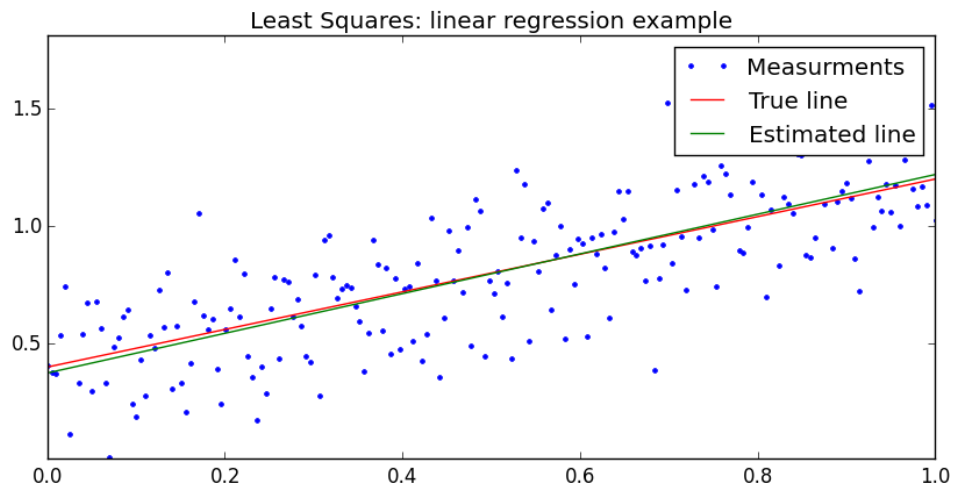

Figure 2: Linear regression code example.

```
x = collect(0.01:0.005:1.0);
a = 0.8;
b = 0.4;
epsilon = 0.2*randn(length(x));
y = a.*x .+ b .+ epsilon;
A = [x ones(length(x))];
B = copy(y);
opt_ab = (A'*A) \ A'*B; # can be obtained using opt_ab = A\B;
using PyPlot
close("all")
plot(x,y,".b");
plot(x,a*x .+ b,"-r");
plot(x,opt_ab[1]*x .+ opt_ab[2],"-g");
title("Least Squares: linear regression example");
legend(("Measurments","True line","Estimated line"));
```

4.2 Weighted least squares

A lot of times we wish to give more emphasis to certain equations in our least squares minimization, because there is a reason to believe that there is less noise there, for example (that is not the only case, though). This can be done with a weighed norm. Let $A \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$. We wish to find $\mathbf{x}$ that minimizes the discrepancy vector $\mathbf{r}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$ in a **weighted** least squares sense, that is minimize

$$\|\mathbf{r}\|_W^2 = \mathbf{r}^\top W \mathbf{r},$$

where $W \in \mathbb{R}^{m\times m}$ is a positive definite matrix, usually a diagonal matrix of positive weights. We get the problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min} \|A\mathbf{x} - \mathbf{b}\|_W^2, \tag{24}$$

and if we do similar steps to the ones we've done before we get

$$\hat{\mathbf{x}} = (A^\top W A)^{-1} A^\top W \mathbf{b}.$$

Example 12 (Algebraic Weighted Least Squares)**.** *Suppose that we need to minimize* $\|A\mathbf{x} - \mathbf{b}\|_2$ *with (exactly as before)*

$$A = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 23 \\ 15 \\ 39 \end{bmatrix}.$$

Suppose now that we want to approximate the first equation in 10 approximately times the accuracy (we are more certain about this equation). We will set

$$W = \begin{bmatrix} 10 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}.$$

Using the normal equations, we will form $A^\top W A$*, and* $A^\top W \mathbf{b}$*, and solve* $A^\top W A\mathbf{x} = A^\top W \mathbf{b}$*:*

$$A^\top W A = \begin{bmatrix} 13 & 17 & -5 \\ 17 & 29 & 15 \\ -5 & 15 & 93 \end{bmatrix}, \quad A^\top W \mathbf{b} = \begin{bmatrix} 47 \\ 171 \\ 417 \end{bmatrix}, \quad \hat{\mathbf{x}} = (A^\top W A)^{-1} A^\top W \mathbf{b} = \begin{bmatrix} -0.097 \\ 3.97 \\ 3.84 \end{bmatrix}.$$

Note that the residual norm in this example comes out

$$\mathbf{r} = A\mathbf{x} - \mathbf{b} = \begin{bmatrix} -0.032 \\ 0.32 \\ 0.32 \\ -0.32 \end{bmatrix}. \tag{25}$$

It is clear that the accuracy in which the first equation is satisfied is 10 times larger than the rest now.

Example 13 (Linear regression with weights)**.** *Assume again that we are given with m measurements (x_i, y_i) and wish to find the line that defines $\{y_i\}$ given $\{x_i\}$. This time, suppose that we know that the measurements y_i are noisier as x_i is smaller. That is - the magnitude of the errors in y_i are proportional to $\frac{1}{\sqrt{x_i}}$. Obviously, the measurements y_i that correspond to the higher values of x_i are much more informative than the others. How will we treat that numerically? We will use a diagonal weight matrix W, such that $w_{ii} = x_i$. This will give more emphasis to the measurements with high values of x_i. The code below shows this example of (weighted) linear regression. The output image appears in Fig. 3, and it shows that for this example, the reconstructed line with weights is better than without weights.*

Figure 3: Weighted linear regression code example.

```
x = collect(0.01:0.005:1.0);
a = 0.8;
b = 0.4;
epsilon = 0.3.*randn(length(x))./sqrt.(x);
W = diagm(x);
y = a.*x .+ b .+ epsilon;
A = [x ones(length(x))];
B = copy(y);
opt_ab_no_w = (A'*A) \ A'*B;
opt_ab = (A'*(W*A)) \ A'*(W*B);
using PyPlot
close("all")
plot(x,y,".b");
plot(x,a*x .+ b,"-r");
plot(x,opt_ab_no_w[1]*x .+ opt_ab_no_w[2],"-k");
plot(x,opt_ab[1]*x .+ opt_ab[2],"-g");
title("Weighted Least Squares: linear regression example");
legend(("Measurments","True line","Est. line w/o weights","Est. line w weights"));
```

4.3 Regularized least squares

We've seen before that if the matrix A in the least squares problem is not full rank, then the matrix $A^\top A$ may be singular (or ill-conditioned), and in that case, we may have infinitely many solutions satisfying the equation $A^\top A\mathbf{x} = A^\top \mathbf{b}$. In many applications we do not control the matrix A - it's an input of the problem, like a data matrix from an unreliable source. Still, we wish to have a well-defined problem with a single solution. This can be done by adding a regularization term, and each regularization that we choose will influence the solution in its own way. Here, we will demonstrate a pretty common and simple Tikhonov regularization

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min} \|A\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|C\mathbf{x}\|_2^2,$$

where C is a matrix, usually chosen as the identity I, and $\lambda > 0$ is a small parameter that balances between the original LS equation and the regularization. The Tikhonov regularization keeps the problem a standard least squares problem. Indeed, the problem above can be rewritten as

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min} \left\| \begin{bmatrix} A \\ \sqrt{\lambda}C \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2,$$

and after following the above steps, the minimizer is given by

$$\hat{\mathbf{x}} = (A^\top A + \lambda C^\top C)^{-1} A^\top \mathbf{b}.$$

The matrix $A^\top A + \lambda C^\top C$ is guaranteed to be invertible if C is full rank, which is why C is often chosen as the identity I.

Example 14 (Regularized least squares)**.** *Suppose that we need to minimize* $\|A\mathbf{x} - \mathbf{b}\|_2$ *with*

$$A = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 3 & 2 \\ -1 & -1 & -1 \\ 1 & 3 & 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -2.999 \\ 5.999 \\ -3.001 \\ 6.001 \end{bmatrix}.$$

As you may notice, the matrix A is singular - the third column is an average of the first two. The vector $\mathbf{b}$ *is the sum of all columns with a little bit of noise. That is,* $\mathbf{x} = [1, 1, 1]^\top$ *is a reasonable solution. However, the matrix $A^\top A$ is singular, hence we cannot solve the system*

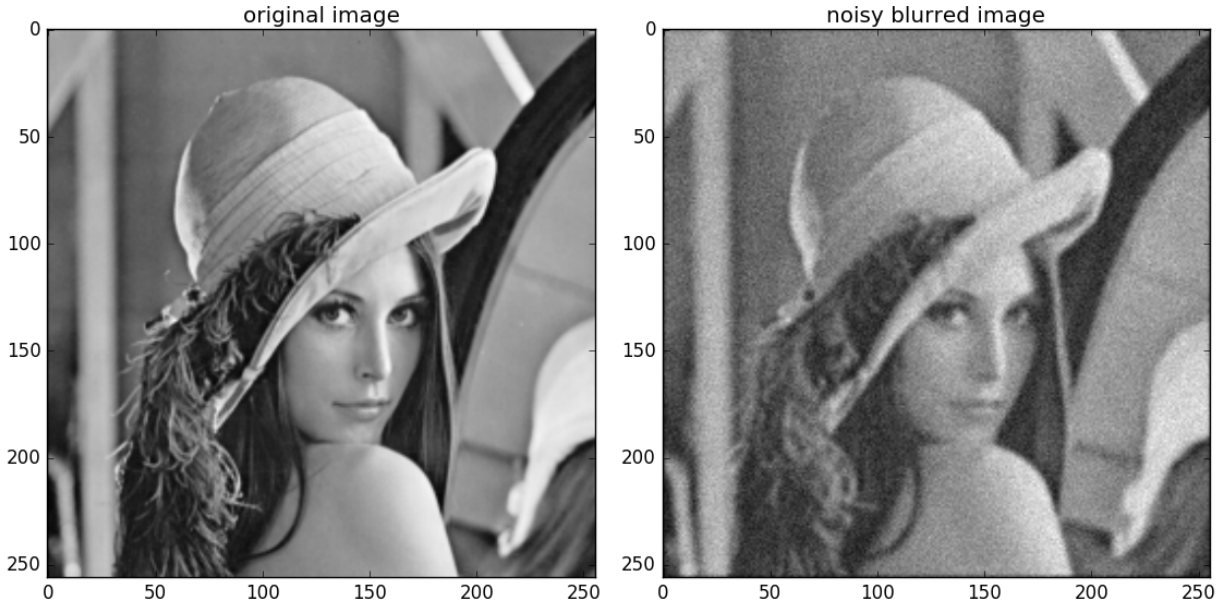

Figure 4: Deblurring example.

with standard LU factorization.

What we'll do is add Tikhonov regularization to solve the problem. That is, we will solve

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min} \|A\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2,$$

with $\lambda = 0.01$*, which is chosen arbitrarily in this example. The solution will be*

$$\hat{\mathbf{x}} = (A^\top A + \lambda I)^{-1} A^\top \mathbf{b}.$$

We get:

$$A^\top A = \begin{bmatrix} 4 & 8 & 6 \\ 8 & 20 & 14 \\ 6 & 14 & 10 \end{bmatrix}, \quad A^\top \mathbf{b} = \begin{bmatrix} 18 \\ 42 \\ 30 \end{bmatrix}, \quad \hat{\mathbf{x}} = (A^\top A + \lambda I)^{-1} A^\top \mathbf{b} \approx \begin{bmatrix} 0.99 \\ 1.003 \\ 0.998 \end{bmatrix}.$$

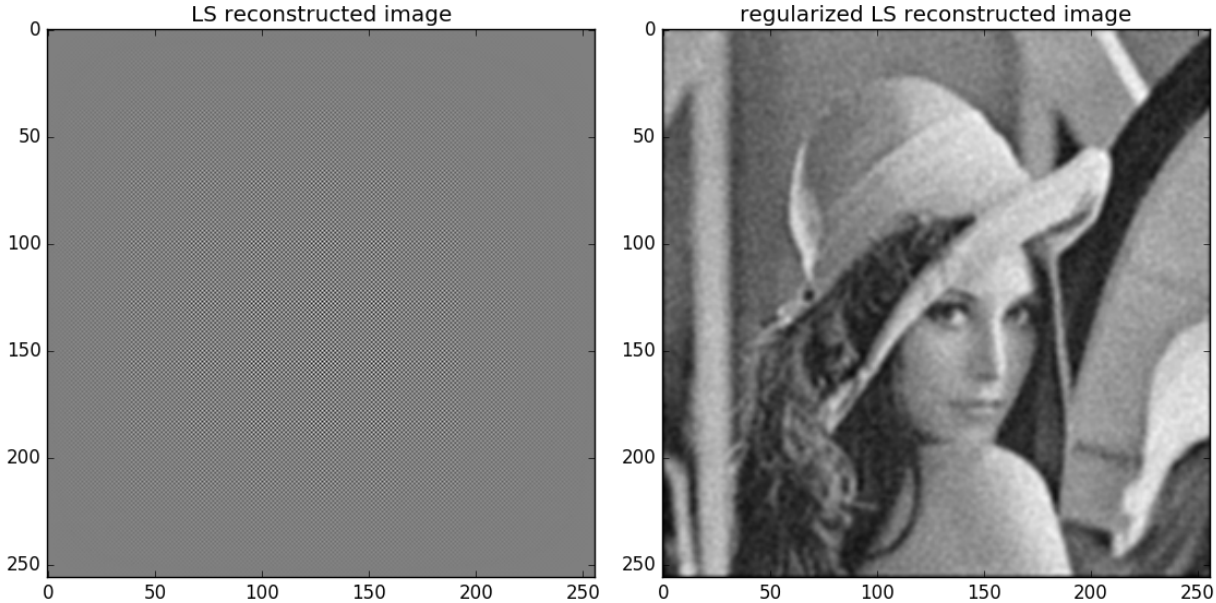

Figure 5: Deblurring example - reconstructions using LS and regularized LS.

4.3.1 Image de-blurring regularized least squares example

Example 15 (Image deblurring using Tikhonov regularization)**.** *The next example is one of the simplest and yet interesting examples for Tikhonov regularization. Assume that we have an image* $\mathbf{x}$ *that undergoes a blurring operation by some matrix* A*, and is corrupted by additive white Gaussian noise. That is, we have the vector* $\mathbf{y} = A\mathbf{x} + \mathbf{n}$*, where* A *is a blurring matrix,* $\mathbf{n}$ *is a noisy image. We wish to recover* $\mathbf{x}$ *from* $\mathbf{y}$*. Figure 4 shows a clean image and a corrupted one.*

In the simple case of deblurring, we assume that we know A*, and we assume that the noise is small. Our first attempt at the reconstruction would be to solve* $\hat{x} = \arg\min \|A\mathbf{x} - \mathbf{y}\|_2^2$*, which in this case would be just* $A^{-1}\mathbf{y}$ *since* A *is square and full rank. Another attempt will be to add regularization, and solve* $\hat{x} = \arg\min \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_2^2$ *for some* λ*. The resulting reconstructions appear in Fig. 5. The first attempt led to essentially nothing, while the second attempt indeed led to a reasonable reconstruction. How did that happen? Apparently, in the first attempt, in addition to sharpen the image we also amplified the noise so much that it ruined the image - this amplification is due to small eigenvalues of* A*. In the second*

attempt we added an identity, which erased the influence of the small eigenvalues and left the large ones more or less as they are. This prevented the amplification of noise. Below is the code of the program.

```
## EXAMPLE - LS with regularization
using MAT
using PyPlot;close("all")
# Read the image and downsize it
a = matread("lena512.mat");
a = a["lena512"];n1 = size(a,1);n2 = size(a,2);
blur1 = spdiagm((0.25*ones(n1-1),0.5*ones(n1),0.25*ones(n1-1)),(-1,0,1),n1,n1);
blur2 = spdiagm((0.25*ones(n1-1),0.5*ones(n1),0.25*ones(n1-1)),(-1,0,1),n2,n2);
A = kron(blur2,blur1);a = reshape(A*a[:],n1,n2);a = a[1:2:end,1:2:end];
n1 = 256;n2 = 256;
# Define the blurring operator A
blur1 = spdiagm((0.25*ones(n1-1),0.5*ones(n1),0.25*ones(n1-1)),(-1,0,1),n1,n1);
blur2 = spdiagm((0.25*ones(n1-1),0.5*ones(n1),0.25*ones(n1-1)),(-1,0,1),n2,n2);
A = kron(blur2,blur1);
A = A^2;
# Define the noisy image
noise = randn(n1,n2);
b = reshape(A*a[:],n1,n2) + 10.0*noise;
# First figure
figure()
subplot(1,2,1)
imshow(a,cmap = "gray")
title("original image")
subplot(1,2,2)
imshow(b,cmap = "gray")
title("noisy blurred image")

# Reconstructions and second figure
a_ls = reshape(A\b[:],n1,n2);
a_rls = reshape((A'*A + 0.5*speye(n1*n2))\(A'*b[:]),n1,n2);
figure()
subplot(1,2,1)
imshow(a_ls,cmap = "gray")
title("LS reconstructed image")
subplot(1,2,2)
imshow(a_rls,cmap = "gray")
title("regularized LS reconstructed image")
```