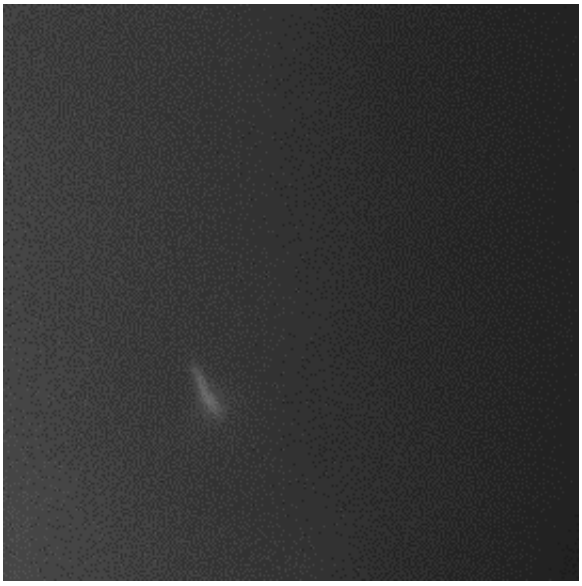
Plane Subtraction and Linear Regression Notes

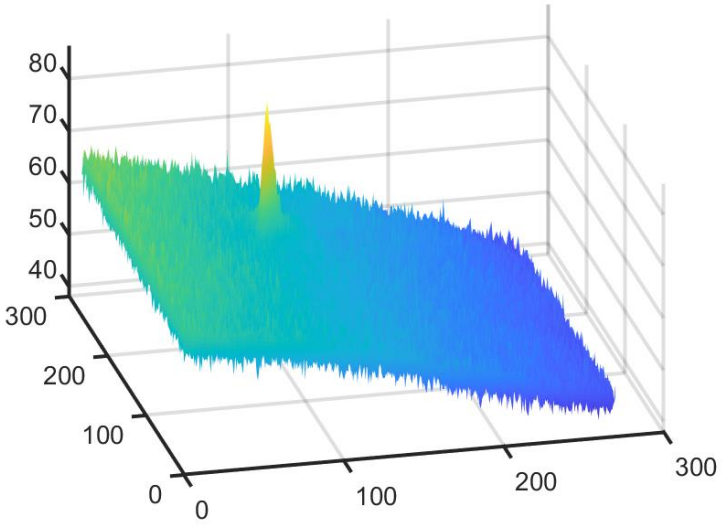
Raghuveer Parthasarathy

October 12, 2022

In Homework 1, we saw an image of a fluorescent bead:



It's interesting to plot our original image as a surface plot:



Clearly, there's a large background gradient!

We want to subtract the **best-fit plane** from the image. This plane has the form

$$z(x, y) = a_2 x + a_1 y + a_0,$$

and we want to determine the best-fit a_0, a_1, a_2.

"Best fit" can have many meanings, and the most principled way to define it requires thinking about the form of the noise that's present. In the absence of other information, though, a good approach is to think of "best fit" as meaning the parameters that minimize the sum of the squared difference between the data (let's call it $I(x,y)$) and the model ($z(x, y) = a_2 x + a_1 y + a_0$), often called χ^2 (chi-squared). In other words, we want to minimize

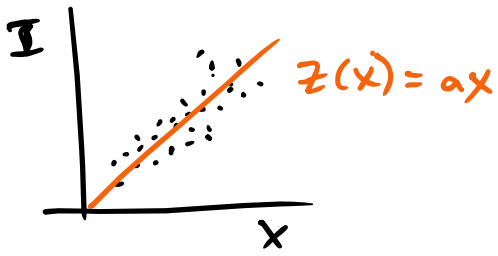
$$\chi^2 = \sum_i (I_i - z_i)^2$$
$$= \sum_i (I_i - (a_2 x + a_1 y + a_0))^2$$

with respect to the parameters, where the sum is over all the datapoints (pixels).

I mentioned in the assignment that you can use fitting tools to do this. Let's elaborate on how this fitting works. You might think we need to search by brute force over all possible values of a_2 , a_1 , a_0, finding whose that minimize χ^2. This isn't the case; there's a method that is both fast and elegant.

A 1 parameter example

Let's first look at a simpler example, a fit of a 1D dataset to a 1D line with zero offset:



The fit function $z(x) = ax$. We want to find the best-fit a.

Now,

$$\chi^2 = \sum_i (I_i - a x_i)^2$$

Let's minimize it:

$$\frac{\partial}{\partial a}\chi^2 = \frac{\partial}{\partial a}\sum_i (I_i - a x_i)^2$$

$$= \sum_i 2(I_i - a x_i)(-x_i)$$

$$= \sum_i (-2)[I_i x_i + a x_i^2]$$

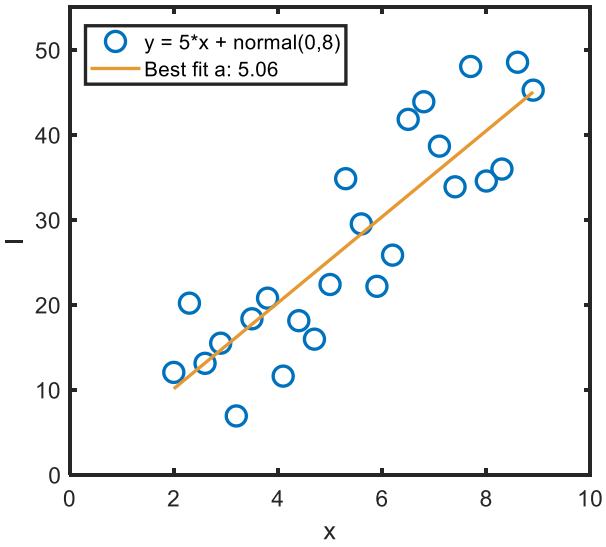
Set this equal to zero:

$$= 0 \Rightarrow \sum_i I_i x_i = \sum_i a x_i^2$$

$$= a = \frac{\sum I_i x_i}{\sum x_i^2}$$

That's the best-fit "a" ! We just add up all the I *times* x values, add up all the x^2 values, and divide. One line, **no searching.**

Here's an illustration:



More parameters, and plane subtraction

You can extend this to linear equations with more parameters.

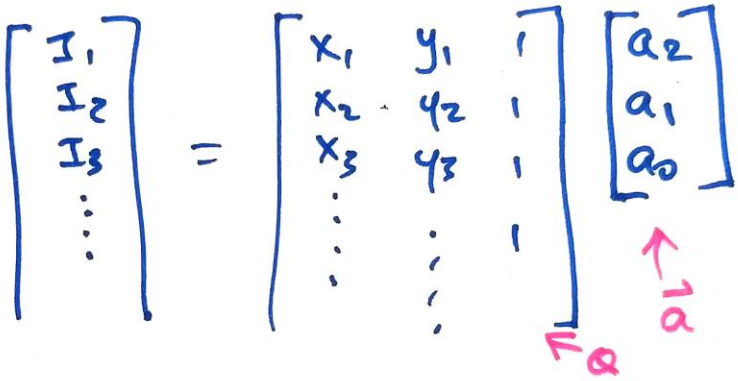
Exercise (strongly recommended): Derive the equation for the best-fit parameters of a general line ($z = a_0x + a_1$). You should again be able to express the result in terms of various sums. Try making up datapoints and see how your best-fit line looks.

There is, however, a more general way to tackle these linear regression problems:

Let's write

$$I(x, y) = a_2x + a_1y + a_0,$$

as a matrix equation:



or

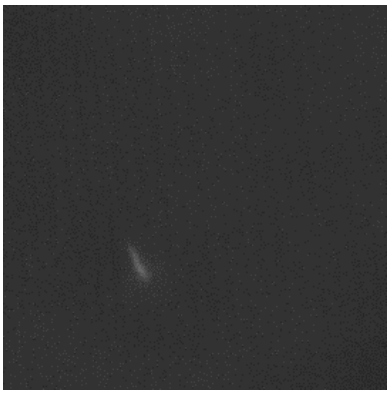
$$I = Q\vec{a}$$

We want to solve this for the best-fit ***a*** vector. You may recall from linear algebra class that the least-squares solution to this overdetermined matrix equation is simply a matrix inversion! (More precisely, we use the Moore–Penrose inverse of matrix Q, denoted Q^+.)

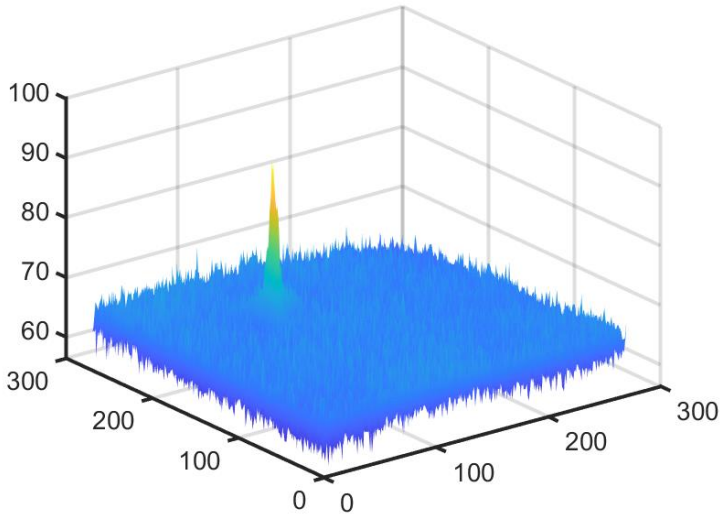
$$\boldsymbol{a = Q^+ I}$$

(Code is given below.)

We can therefore simply get the coefficients ***a***, and then subtract the fit plane (`im_sub = im - a2x - a1y`), keeping or ignoring the constant part as we wish, and get an image that looks like this:



or as a surface plot:



In MATLAB, all this is a few lines:

```
im = imread('.\Images for Homework\GUV24_bead_crop.png');

im = double(im); % make double precision

[X, Y] = meshgrid(1:size(im,2), 1:size(im,1)); % 2D arrays of X, Y
values

Q = [X(:) Y(:) ones(numel(im),1)]; % x and position

A = Q\im(:)  % the coefficients of the best-fit plane

im_sub = im - A(1)*X - A(2)*Y;  % the plane-subtracted image
```

In Python,

```
# first load the image; "im" as above

# x and y from meshgrid, as above

Q = np.column_stack((x.reshape((x.size,1)), y.reshape((y.size,1)),
np.ones((x.size,1))))

a = np.matmul(np.linalg.pinv(Q), im.ravel())

im_sub = im - a[0]x - a[1]*y    # the plane-subtracted image
```