

Homework #7

The overall goal of this assignment is to fit a star's profile using `mpfit()`, the fitting program that I demonstrated in class. We'll do this in three stages.

1. First, write a function called `gauss2d.pro` which will create a two-dimensional Gaussian, much like the one dimensional version you wrote earlier. Check out http://en.wikipedia.org/wiki/Gaussian_function#Two-dimensional_Gaussian_function for some more information on parameters. You'll want to pass all of the parameters in a single array. The output of the final `shade_surf` command should look like that linked on the website.

```
IDL> x = my_fillarr(0.01, -5, 5)
% Compiled module: MY_FILLARR.
IDL> y = x
IDL> offset = 0
IDL> amplitude = 1
IDL> xCent = 0
IDL> yCent = 0
IDL> xSigma = 1
IDL> ySigma = 0.5
IDL> theta = !pi / 4
IDL> g2d = gauss2d(x, y, [offset, amplitude, xCent, yCent,
xSigma, ySigma, theta])
IDL> shade_surf, g2d, x, y, charsize = 2
```

2. First, take a look at `fitfunc.pro`, which is stashed in `~jmcbride/idl_decal/shared/`, and demonstrates the sort of function that `mpfit` requires. That is, you want to pass in an array of parameters, and then four keyword arguments: an array `x`, an array `y`, a 2d image array, and an array `err` containing error values. Once you've done that, write `gauss2d_diff.pro`, which will do the same thing, but will use the function you wrote in part 1 to return an array of the differences between some input image and a 2d Gaussian computed using your function. Note that `mpfit` requires a 1d input, so the final lines will need to look like

```
diff = (image - fit) / err
diff1d = reform(diff, n_elements(diff))
return, diff1d
```

Since you already did the work of computing a 2d Gaussian in part 1, this function will not be more than a line or two of work beyond what I gave you.

3. Now, put it together in a function called `gauss2dfit.pro`, which will accept as an argument an image and return the best fit (and, as an argument, will pass out the best fit parameters and the `x` and `y` arrays). So, you will call it like this:

```
IDL> fit = gauss2dfit(image, fitpar, x, y)
```

The main challenge of this function is to generate appropriate guesses. As you'll recall, `mpfit`

requires an array of guesses to get started, and if the guesses are way off the actual result, it is unlikely that you will get a good fit. Since the actual calling of `mpfit` is somewhat tricky, I will again provide for you the last few lines of the function which actually involve doing the fitting.

```
; you generate guesses and put them in an array called guessp
functargs = {X: x, Y: y, IMAGE: image, ERR: sqrt(image)}
; the image array is the actual data
; we use the square root for error because we assume that
; the errors are all Poissonian in nature, a good assumption
fitpar = mpfit('gauss2d_diff', guessp, functargs = functargs)
fit = gauss2d(x, y, fitpar)
return, fit
```