Hubble Law Part 3: Extracting the Hubble Constant H_o

In part 1 of the lab we calculated the radial velocities of galaxies receding from our line of sight using the redshift of the hydrogen alpha line. Whereas in part 2 we filled in the games of the missing distances to these galaxies by measuring its angular size and taking advantage of the average size of a spiral galaxy. The Hubble Law can now be investigated using the galactic information acquired. In the end of this notebook you'll be in a position to quote a final measured Hubble constant from applying regression models onto the data, as well as, goodness-of-fit statistics and comparisons. These final results will include:

- Hubble Constant: $H_o \pm \sigma_H$.
- Residuals of the regression.
- Goodness-of-fit: χ^2, degrees of freedom, p-value.
- Comparison: Student T-test using literature values.

Read in $(V_{radial}, D_{galactic})$:

Read in te distances from the file *distances.txt* and velocities from *radial_vel.txt* into arrays. Both files should be ordered in the same manner (in increasing NGC galaxy designation).

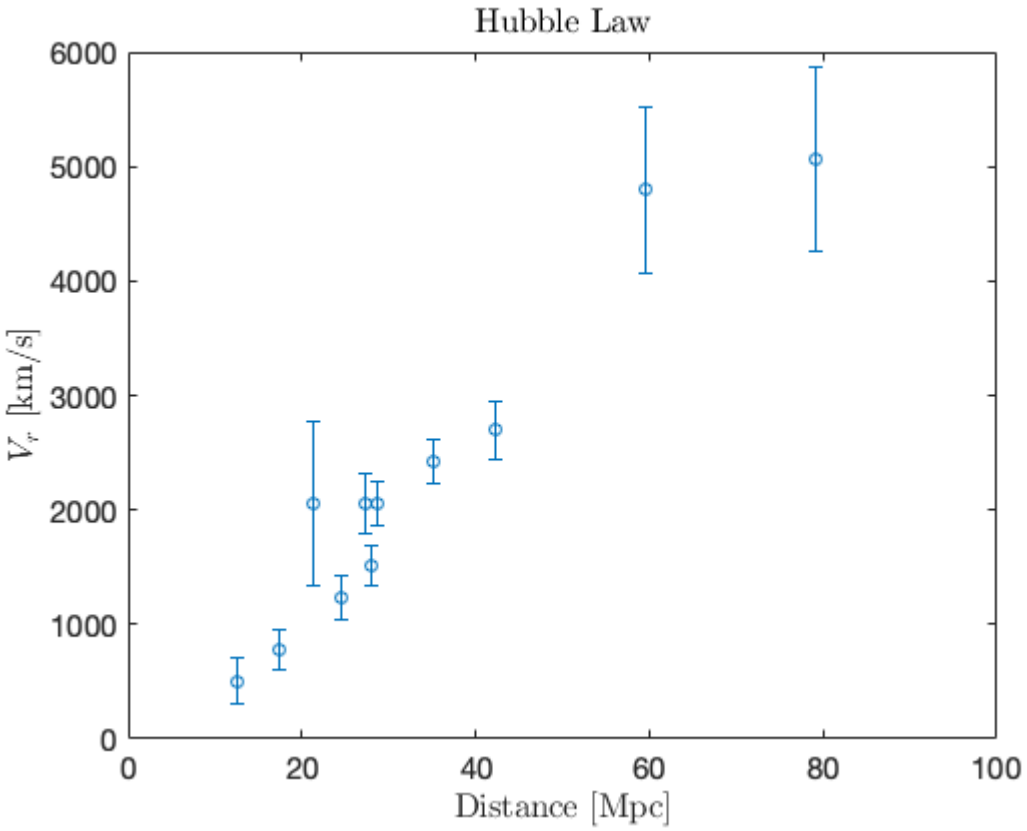
```
directory = '~/tutorialSims/HubbleLaw/NGCData/';

dis_data = dlmread(strcat(directory,"distances.txt"));
vel_data = dlmread(strcat(directory,"radial_vel.txt"));

distances = dis_data(:,2);
velocities = vel_data(:,1);
velocities_std = vel_data(:,2);
```

Visualize the data with error bars to inspect for any possible correlation.

```
p0 = errorbar(distances, velocities, velocities_std,'o');
xlim([0,100])
set(gca,'fontsize',15)
title('Hubble Law','Interpreter','latex')
xlabel('Distance [Mpc]','Interpreter','latex')
ylabel('$V_{r}$ [km/s]','Interpreter','latex')
hold on;
```



Linear Regression:

Hubble's Law is a linear relation, meaning, we can take advantage of MATLAB's χ^2-fitting and linear regression methods. Similar to how we fit a Gaussian to the spectral lines, we'll additionally feed it the algorithm an additional parameter, the 'weight'. The weight is defined as $w = 1/\sigma^2$. We'll be using the two parameter linear function *poly1* as our fit model. Which is defined as,

$$v(d|p_1, p_2) = p_1 \cdot d + p_2$$

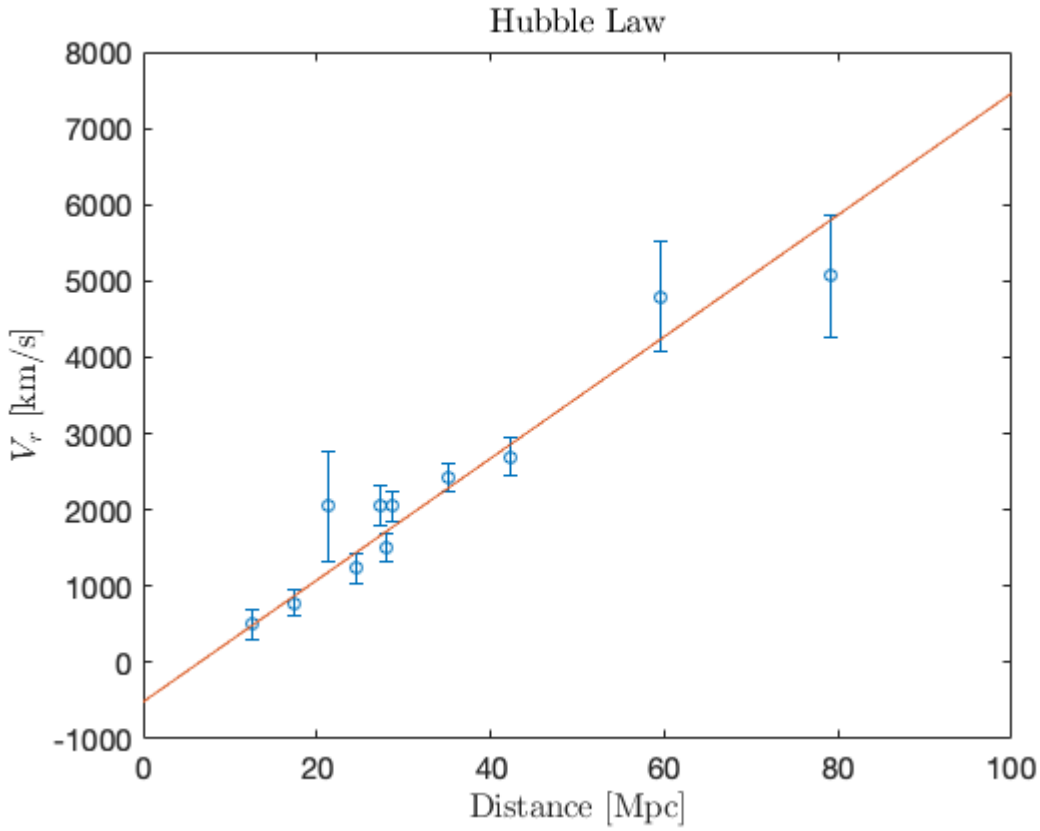
Where v is the radial velocity in km/s, d is the galactic distance in Mpc, and the p_1 is the estimated Hubble Constant H_o.

```
[f, gof, fit_output] = fit(distances,velocities,'poly1','Weights', ...
                                                    velocities_std.^(-2));
f
```

```
f =
     Linear model Poly1:
     f(x) = p1*x + p2
     Coefficients (with 95% confidence bounds):
       p1 =       79.77  (62.46, 97.08)
       p2 =      -514.7  (-1007, -22.35)
```

Similar to part 1, created a linear function from the parameter's estimated from the regression. Create an array of linearly spaced distance values which is used to construct the final function. Plot this on top of our data. How does it look visually?

```
x_dis = linspace(0,100,100);
line = f.p1*x_dis + f.p2;
plot(x_dis,line)
```



Error Analysis, Goodness-of-fit, and P-value:

To extract the estimated errors on the parameters, use the Jocobian matrix to calculate covariance matrix. Keeping in mind that the diagonals on the covariance matrix equal the variance on the parameters $Var[p_i]$, in order $i = (1, 2)$.

```
J = fit_output.Jacobian;           %extract the jocobian matrix
covariance_matrix = inv(J'*J)
```

```
covariance_matrix = 2×2
10^4 ×
    0.0049   -0.1303
   -0.1303    3.9561
```

Quote the estimated Hubble Constant H_o:

```
hubble_const = f.p1
```

```
hubble_const = 79.7690
```

Quote the estimated error σ_H:

```
hubble_const_std = sqrt(covariance_matrix(1,1))
```

```
hubble_const_std = 6.9929
```

Having the final estimated value and associated error $H_o \pm \sigma_H$ isn't enough. To feel confident in these results we need test the goodness of the fit. Specifically, the minimum χ^2, the number of degrees of freedom, and the p-value of the observation. Keeping in mind that a proper statistical inference needs all three of these pieces. A p-value alone doesn't convey enough information to make a proper statistical statement of the data.

Quote the minimum χ^2 value:

```
min_chi_2 = gof.sse
```

```
min_chi_2 = 10.7749
```

Quote the number of degrees of freedom = $N_{data} - N_{parameters}$:

```
ndof = gof.dfe;
```

Last but not least, the p-value of getting H_o given the model, $p = \int_{\chi^2_{\min}}^{\infty} f(x|\nu)dx$:

```
p_value = chi2cdf(min_chi_2, ndof, 'upper')
```

```
p_value = 0.2915
```

```
reduced_chi2 = min_chi_2/ndof
```

```
reduced_chi2 = 1.1972
```

Student's T-test Comparison: