

Hubble's Law and Least Squares Fitting

Computational Physics

Physics is a natural science which attempts to describe the Universe in terms of a few basic concepts such as time, space, mass, and electric charge. These quantities can be assigned numerical values that every physicist can agree upon. Nature is complex, so physicists choose to study natural phenomena that appear to be as simple as possible and can be reproduced reliably by other physicists.

Most physicists specialize in one of three types of activity. Experimental physicists observe these phenomena, measure their properties using instruments, and attempt to discover simple laws describe these observations: experimental physics is based on technology. Theoretical physicists invent mathematical models that relate measured properties and construct theories which organize and unify these models: theoretical physics is based on mathematics. Computational physicists use digital computers to analyze experimental data and explore the properties of models and theories: computational physics is based on computer science and numerical analysis.

Experimentalists spend most of their time designing and building instruments, and in collecting and analyzing experimental data. Theorists work primarily with mathematical theories and equations, and make predictions that can be verified or falsified by experiment. Computationalists design algorithms to solve equations or analyze data, write and debug computer programs that implement algorithms, and run simulations to explore theories and compare with experimental data.

The Expanding Universe

A galaxy is a collection of several billion stars. Galaxies have characteristic shapes and are separated from one another by empty space. More than 10,000 galaxies have been identified and cataloged, and Earth receives light from more than a hundred billion galaxies. The location and speed of a galaxy are examples of natural phenomena that astronomers have attempted to measure. Fig. 1 shows a typical galaxy studied by Edwin Hubble[1] and his collaborator Milton Humason.

Cepheid Variables as Standard Candles

A Cepheid variable is a type of star whose luminosity varies with a period of a few days. Its absolute luminosity can be predicted from its period. Its apparent luminosity observed on Earth can therefore be used to infer its distance.

The speed of a galaxy can be inferred from the shift in wavelength of spectral lines in the light it emits. Lines from distant galaxies are generally redshifted, which indicates that they are receding from us.

Hubble analyzed the data in Fig. 2 using the relation

$$rK + X \cos \alpha \cos \delta + Y \sin \alpha \cos \delta + Z \sin \delta = v, \quad (1)$$

where r, v are the measured distance and speed of the galaxy located at galactic longitude α and latitude δ , and K, X, Y, Z are constants. Fig. 3 shows the results of his analysis. He concluded that $K \approx 500$ km/s/Mpc, which is considerably larger than the current best value of Hubble's constant $H(t_0) \simeq 72$ km/s/Mpc.



Figure 1: Hubble lists Barnard's Galaxy NGC 6822, see <http://antwrp.gsfc.nasa.gov/apod/ap020123.html>, at a distance $r = 0.214$ Mpc (1 Mega-parsec = 3.086×10^{19} km) moving towards us with speed $v = 130$ km/s.

General-Relativistic Theory of Cosmology

General Relativity is a mathematical theory that relates the properties of time and space to the energy density measured at each point in space and instant of time. The mathematical equations of General Relativity can be written

$$\mathcal{R}_{\mu\nu} - \frac{1}{2}g_{\mu\nu}\mathcal{R} = \frac{8\pi G_{\text{N}}}{c^4}T_{\mu\nu} - g_{\mu\nu}\Lambda \quad (2)$$

where G_{N} is Newton's gravitational constant, c is the speed of light, and Λ is the cosmological constant. The other quantities in the equation are functions of the time-space 4-vector $x^\mu = \{ct, \vec{\mathbf{r}}\}$. The metric tensor $g_{\mu\nu}(x)$ describes the geometry of spacetime, the Ricci tensor $\mathcal{R}_{\mu\nu}$ and curvature scalar \mathcal{R} are functions of the metric tensor and its spacetime derivatives, and the stress-energy tensor $T_{\mu,\nu}$ represents the energy and momentum density of matter and radiation.

This complex set of nonlinear partial differential equations is extremely challenging to solve. Analytic solutions exist only for very simple distributions of matter and radiation. The simplest cosmological model assumes that spacetime is homogeneous and isotropic and described by the Robertson-Walker metric

$$ds^2 \equiv \sum_{\mu=0}^3 \sum_{\nu=0}^3 g_{\mu\nu} dx^\mu dx^\nu = c^2 dt^2 - R^2(t) \left[\frac{dr^2}{1 - kr^2} + r^2(d\theta^2 + \sin^2 \theta d\phi^2) \right], \quad (3)$$

where R is the cosmological scale factor, r, θ, ϕ are spherical-polar coordinates, and the curvature constant $k = 0, \pm 1$ signifies a flat, open, or closed universe.

The simplest assumption about matter and radiation is that it behaves like uniform perfect fluid with density ρ and pressure p . This leads to the Friedmann-Lamaître equations

$$H^2 \equiv \left(\frac{\dot{R}}{R} \right)^2 = \frac{8\pi G_{\text{N}}\rho}{3} - \frac{kc^2}{R^2} + \frac{\Lambda c^2}{3}, \quad \frac{\ddot{R}}{R} = -\frac{4\pi G_{\text{N}}}{3}(\rho + 3p) + \frac{\Lambda c^2}{3}. \quad (4)$$

$H(t)$ is the Hubble parameter and its value at the present time t_0 is Hubble's constant $H(t_0) = 72$ km/s/Mpc.

TABLE 1
NEBULAE WHOSE DISTANCES HAVE BEEN ESTIMATED FROM STARS INVOLVED OR FROM
MEAN LUMINOSITIES IN A CLUSTER

OBJECT	m_s	r	v	m_t	M_t
S. Mag.	..	0.032	+ 170	1.5	-16.0
L. Mag.	..	0.034	+ 290	0.5	17.2
N. G. C. 6822	..	0.214	- 130	9.0	12.7
598	..	0.263	- 70	7.0	15.1
221	..	0.275	- 185	8.8	13.4
224	..	0.275	- 220	5.0	17.2
5457	17.0	0.45	+ 200	9.9	13.3
4736	17.3	0.5	+ 290	8.4	15.1
5194	17.3	0.5	+ 270	7.4	16.1
4449	17.8	0.63	+ 200	9.5	14.5
4214	18.3	0.8	+ 300	11.3	13.2
3031	18.5	0.9	- 30	8.3	16.4
3627	18.5	0.9	+ 650	9.1	15.7
4826	18.5	0.9	+ 150	9.0	15.7
5236	18.5	0.9	+ 500	10.4	14.4
1068	18.7	1.0	+ 920	9.1	15.9
5055	19.0	1.1	+ 450	9.6	15.6
7331	19.0	1.1	+ 500	10.4	14.8
4258	19.5	1.4	+ 500	8.7	17.0
4151	20.0	1.7	+ 960	12.0	14.2
4382	..	2.0	+ 500	10.0	16.5
4472	..	2.0	+ 850	8.8	17.7
4486	..	2.0	+ 800	9.7	16.8
4649	..	2.0	+1090	9.5	17.0
Mean					-15.5

- m_s = photographic magnitude of brightest stars involved.
 r = distance in units of 10^6 parsecs. The first two are Shapley's values.
 v = measured velocities in km./sec. N. G. C. 6822, 221, 224 and 5457 are recent determinations by Humason.
 m_t = Holetschek's visual magnitude as corrected by Hopmann. The first three objects were not measured by Holetschek, and the values of m_t represent estimates by the author based upon such data as are available.
 M_t = total visual absolute magnitude computed from m_t and r .

Figure 2: Table 1 from Hubble's paper [1].

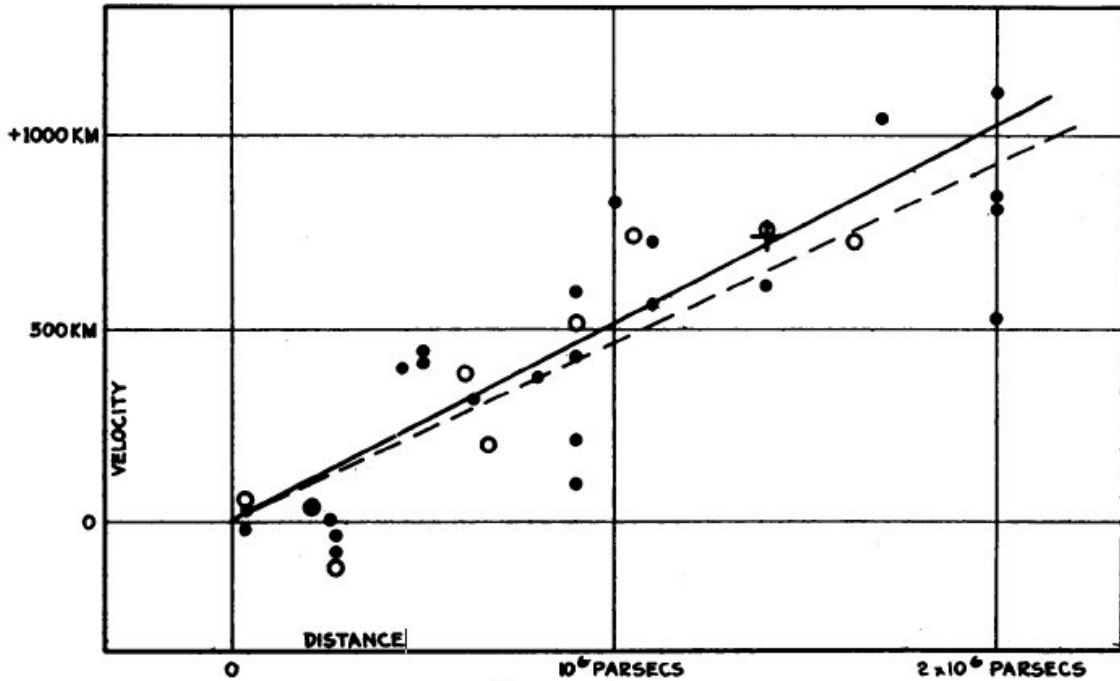


FIGURE 1

Velocity-Distance Relation among Extra-Galactic Nebulae.

Radial velocities, corrected for solar motion, are plotted against distances estimated from involved stars and mean luminosities of nebulae in a cluster. The black discs and full line represent the solution for solar motion using the nebulae individually; the circles and broken line represent the solution combining the nebulae into groups; the cross represents the mean velocity corresponding to the mean distance of 22 nebulae whose distances could not be estimated individually.

Figure 3: A Figure from Hubble's paper [1].

We will study these differential equations later in the course. For this lecture all we need to know is how to relate galaxy redshifts in an expanding universe. Suppose R_1 is the scale factor when light of frequency ν_1 was emitted from a distant galaxy, and R_2 is the scale factor when this light is observed on Earth with frequency ν_2 , then the redshift z is given by

$$z + 1 = \frac{\nu_1}{\nu_2} = \frac{R_2}{R_1} \simeq 1 + \frac{v_{12}}{c}, \quad (5)$$

where v_{12} is the speed of the distant galaxy relative to the observer.

Algorithm for fitting data to a straight line

Consider a data set with n data points labeled by an index $i = 0, 1, \dots, n-1$. Each point consists of two real numbers x_i, y_i . For example, n is the number of observed galaxies, and x_i and y_i are their radial distances and velocities.

We would like to summarize this data set using a model equation that relates the two real variables x, y , for example a linear model

$$y(x) = a + bx, \quad (6)$$

with intercept a and slope b .

Least-squares algorithm

The least-squares algorithm determines the model parameters a, b by minimizing the function

$$f(a, b) \equiv \sum_{i=0}^{n-1} (y_i - a - bx_i)^2. \quad (7)$$

This procedure tends to make the deviation of y_i from the point $y(x_i) = a + bx_i$ on the straight line as small as possible. The derivatives of f vanish at its minimum

$$\frac{\partial f}{\partial a} = -2 \sum_{i=0}^{n-1} (y_i - a - bx_i) = 0, \quad \text{and} \quad \frac{\partial f}{\partial b} = -2 \sum_{i=0}^{n-1} x_i (y_i - a - bx_i) = 0. \quad (8)$$

These two equations can be solved simultaneously for the two unknowns a, b . Define the following sums:

$$s_x \equiv \sum_{i=0}^{n-1} x_i, \quad s_y \equiv \sum_{i=0}^{n-1} y_i, \quad s_{xx} \equiv \sum_{i=0}^{n-1} x_i^2, \quad s_{xy} \equiv \sum_{i=0}^{n-1} x_i y_i. \quad (9)$$

Then the least-squares values of the parameters are given by

$$a = \frac{s_{xx}s_y - s_x s_{xy}}{n s_{xx} - s_x^2}, \quad b = \frac{n s_{xy} - s_x s_y}{n s_{xx} - s_x^2}. \quad (10)$$

Uncertainty estimates

The number of degrees of freedom of the fit is the number of data points minus the number of parameters: $\nu = n - 2$. An estimate of the variance of the data set from the model prediction is

$$\sigma^2 \equiv \frac{f(a, b)}{\nu} = \frac{1}{n-2} \sum_{i=0}^{n-1} (y_i - y(x_i))^2. \quad (11)$$

The standard deviation σ is an estimate of the error bar in each y_i . These error estimates can be propagated to the parameters a, b considered as functions of y_i :

$$\sigma_a^2 = \sum_{i=0}^{n-1} \left(\sigma \frac{\partial a}{\partial y_i} \right)^2 = \frac{\sigma^2 s_{xx}}{ns_{xx} - s_x^2}, \quad \sigma_b^2 = \sum_{i=0}^{n-1} \left(\sigma \frac{\partial b}{\partial y_i} \right)^2 = \frac{\sigma^2 n}{ns_{xx} - s_x^2}. \quad (12)$$

Programming in C++

The simplest C++ program consists of 12 characters:

_____ Program 1: <http://www.physics.buffalo.edu/phy410-505/topic1/smallest.cpp> _____

```
int main(){}
```

Every C++ program must define a unique `main` function, which can have either no arguments as defined here by empty parentheses `()`, or two arguments usually written `(int argc, char *argv[])`. The return value of the function is of type `int`, which usually represents a 32-bit integer in the range $[-2^{31}..2^{31}-1]$. Note the required space between `int` and `main`. The body of the function in braces `{}` is empty: the function does nothing except to return the integer 0 (zero) by default.

The following program prints a message on the display.

_____ Program 2: <http://www.physics.buffalo.edu/phy410-505/topic1/hello.cpp> _____

```
#include <iostream>    /* standard C++ input output stream header */

int main()            // main function returns 0 on successful completion
{
    std::cout << "Hello, world!" << std::endl;
}
```

The preprocessor directive `#include` is used to import object and function definitions from the C++ standard `iostream` library and make them available to the program. A standard library name is conventionally enclosed in angle brackets `< >`. These objects and functions belong to a `namespace` called `std`.

The main function contains a single executable statement, which is an expression terminated by a semicolon. The expression is built using the binary operator `<<` which takes a left operand and a right operand and associates from left to right. The message is coded as a string literal, which is a sequence of characters enclosed in double quotes `"`. The object `std::cout` is defined in the `iostream` library and handles printing information on the display. The object `std::endl` prints a newline character and then flushes the stream buffer. The expression is interpreted as `((std::cout<<"Hello, world!")<<std::endl)`. The sub-expression `(std::cout<<"Hello, world!")` prints the message string on the display buffer and returns the value `std::cout`. The resulting expression `(std::cout<<std::endl)` then prints a newline on the display buffer and flushes the buffer to the physical display.

To make the program more readable by human beings, comments, spaces and empty lines have been added: all of these are ignored by the C++ compiler. There are two types of comments: C-style comments are

started by the two characters `/*` and ended by the two characters `*/` and can extend over more than one line. C++-style comments are started by the two characters `//` and extend to the end of the line.

Computing with integers and real numbers

Physical quantities are represented real numbers, which include integers, rational fractions, and real numbers like $\sqrt{2}$ and π that require an infinite number of decimal digits to express exactly. Integers that are not too large in magnitude can be represented in C++ by quantities of type `int` and rational fractions and real numbers that are not too large or too small can be represented by quantities of type `double` with approximately 15 decimal digits of precision.

_____ Program 3: http://www.physics.buffalo.edu/phy410-505/topic1/limits_and_pi.cpp _____

```
#include <cmath>          /* defines arc-tangent function atan */
#include <iostream>       /* defines cout, cin and endl */
#include <limits>        /* defines numeric_limits */

using namespace std;    // import included std objects into global namespace

const double pi = 4 * atan(1.0); // pi in radians from arc-tangent function

int main () {

    cout << "\nLimiting values of int and double\n"
         << "max(int)    = " << numeric_limits<int>::max()    << '\t'
         << "min(int)    = " << numeric_limits<int>::min()    << '\n'
         << "max(double) = " << numeric_limits<double>::max() << '\t'
         << "min(double) = " << numeric_limits<double>::min() << endl;

    int digits = 6;      // number of decimal digits printed by default

    do {

        cout << "The double value of pi = " << pi << '\n'
             << "The exact value of pi = 3.1415926535897932384626433832795..."
             << " ..." << endl;

        cout << "\nEnter number of decimal places for pi (or 0 to quit): ";
        cin >> digits;          // read using standard input stream
        cout.precision(digits); // reset number of digits printed by cout

    } while (digits > 0);

}
```

The standard `cmath` header defines various useful mathematical functions such as `atan` which returns the principal arc tangent of its `double` argument in the interval $[-\pi/2, \pi/2]$ radians.

The standard `limits` header defines a parametrized (template) type `numeric_limits` which takes

types like `int` or `double` as parameters. The type has static member functions, such as `min` and `max`, which return limiting values.

The program uses the `do { } while (condition);` control structure: the block of statements in braces `{ }` is executed repeatedly while the logical expression *condition* evaluates to `true`.

Computing the Hubble Constant

Hubble used Eq. 1 with 4 parameters to model his data. We will use a simpler linear equation with two parameters

$$v(r) = a + br, \quad (13)$$

to determine Hubble's constant as the slope *b* using the linear least-squares algorithm.

_____ Program 4: <http://www.physics.buffalo.edu/phy410-505/topic1/hubble.cpp> _____

```
#include <cmath>
#include <iostream>
using namespace std;

const int n = 24; // number of galaxies in Table 1

double r[n] = { // distances in Mpc
    0.032, 0.034, 0.214, 0.263, 0.275, 0.275, 0.45, 0.5, 0.5, 0.63, 0.8, 0.9,
    0.9, 0.9, 0.9, 1.0, 1.1, 1.1, 1.4, 1.7, 2.0, 2.0, 2.0, 2.0
};

double v[n] = { // velocities in km/s
    +170, +290, -130, -70, -185, -220, +200, +290, +270, +200, +300, -30,
    +650, +150, +500, +920, +450, +500, +500, +960, +500, +850, +800, +1090
};

int main() {

    // declare and initialize various sums to be computed
    double s_x = 0, s_y = 0, s_xx = 0, s_xy = 0;

    // compute the sums
    for (int i = 0; i < n; i++) {
        s_x += r[i];
        s_y += v[i];
        s_xx += r[i] * r[i];
        s_xy += r[i] * v[i];
    }

    // evaluate least-squares fit formulas
    double denom = n * s_xx - s_x * s_x;
    double a = (s_xx * s_y - s_x * s_xy) / denom;
    double b = (n * s_xy - s_x * s_y) / denom;
```

```

// estimate the variance in the data set
double sum = 0;
for (int i = 0; i < n; i++) {
    double v_of_r_i = a + b * r[i];
    double error = v[i] - v_of_r_i;
    sum += error * error;
}
double sigma = sqrt(sum / (n - 2));    // estimate of error bar in v

// estimate errors in a and b
double sigma_a = sqrt(sigma * sigma * s_xx / denom);
double sigma_b = sqrt(sigma * sigma * n / denom);

// print results
cout.precision(4);
cout << " Least-squares fit of " << n << " data points\n"
    << " -----\n"
    << " Hubble's constant slope   b = " << b
    << " +- " << sigma_b << " km/s/Mpc\n"
    << " Intercept with r axis     a = " << a
    << " +- " << sigma_a << " km/s\n"
    << " Estimated v error bar sigma = " << sigma << " km/s" << endl;
}

```

To store the 24 values of r , an array of `double` values that cannot be changed is declared `const double r[n]` and initialized with a list of 24 real numbers. In C++ an array with n elements is indexed from 0 to $n-1$. Thus `r[0]` is the first element of the array, `r[1]` the second, and `r[23]` the last.

To compute the various sums needed the `for (initializer; condition; increment) { }` control structure is used. The *initializer* expression is evaluated first. The *condition* expression is then tested: if it evaluates to `true` then the sequence of (1) statements in the block `{ }` followed by (2) the *increment* expression, followed by (3) the *condition* expression, is repeatedly executed until the *condition* evaluates to `false`.

Homework Problem

Modify the Hubble program to make a least-squares fit to the 9 open circles in Fig. 3 and compare the slope of the fitted straight line to Hubble's value of K . Estimate the age of the Universe that this value implies.

References

- [1] Edwin Hubble, "A relation between distance and radial velocity among extra-galactic nebulae", Proc. Natl. Acad. Sci. USA **15**, 168 (1929), <http://www.pnas.org/content/15/3/168.full.pdf>.
- [2] K.A. Olive and J.A. Peacock, "Big-bang cosmology", in C. Amsler et al., Phys. Lett. **B667**, 1 (2008), <http://pdg.lbl.gov/2009/reviews/rpp2009-rev-bbang-cosmology.pdf>.

- [3] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, "Numerical Recipes in C" (Cambridge University Press 1992), §15.2 Fitting Data to a Straight Line,
<http://www.nrbook.com/a/bookcpdf/c15-2.pdf>.
- [4] Herbert Schildt, "C++ Beginner's Guide",
<http://msdn.microsoft.com/en-us/beginner/cc305129.aspx>, Chapter 1: C++ Fundamentals,
<http://go.microsoft.com/?linkid=8310946>.