Elementy kosmologii

ab

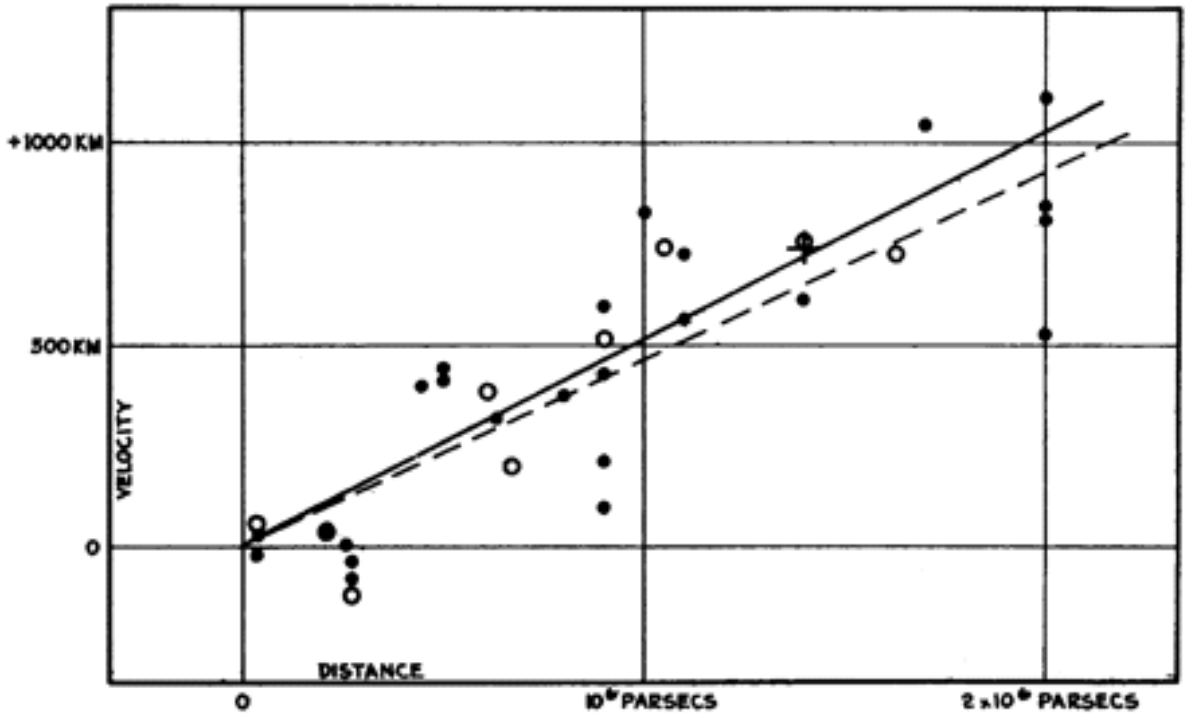
1 czerwca 2019

Tematy

- Regresja liniowa
- Parametr Hubble'a
- CMBR

Parametr Hubble'a

Obserwacje Hubble'a (1929)



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 [?].

Moduł odległości (def):

$$\mu = m - M = 5\log_{10}(r) - 5$$

M - wielkośc absolutna; m - wielkośc pozorna, obserwowana r - odległość od Ziemi w parsekach;

Ze szczególnej teorii wzgledności (STW) wynika, że

$$1 + z = \frac{(1 + v/c)}{\sqrt{1 - v^2/c^2}}$$

Stąd wynika, że predkości v dla większości supernowych są bardzo duże i należy wobec tego korzystać z prawa przesunięć ogolnej teorii wzgledności (OTW)

$$z = \frac{R(t_0)}{R(t)} - 1.$$

gdzie R jest czynnikiem skali.

Dla odległych supernowych otrzymuje się przybliżone wyrażenie

$$\mu = 25 + 5\log_{10}(cz/H_0) + 1.086(1 - q_0)z + ...$$

gdzie c jest w km/s, H_0 w km/s/Mpc. Wyrażenie to można otrzymać z rozwiniecia Taylora czynnika skali $R(t)$:

$$R(t) = R(t_0)\left[1 + (t - t_0)H_0 - 1/2(t - t_0)^2 q_0 H_0^2 + \dots\right]$$

gdzie
t – moment czasu emisji światla przez supernową
$H_0 = \dot{R}(t_0)/R(t_0)$ – parametr Hubble'a
$q_0 = R(t_0)\ddot{R}(t_0)/\dot{R}(t_0)^2$ – parametr spowolnienia.

Korzystając z danych Tamary Davis [?] można obliczyć stałą Hubble'a H_0 (zakładamy tutaj, że $q_0 = 1$).

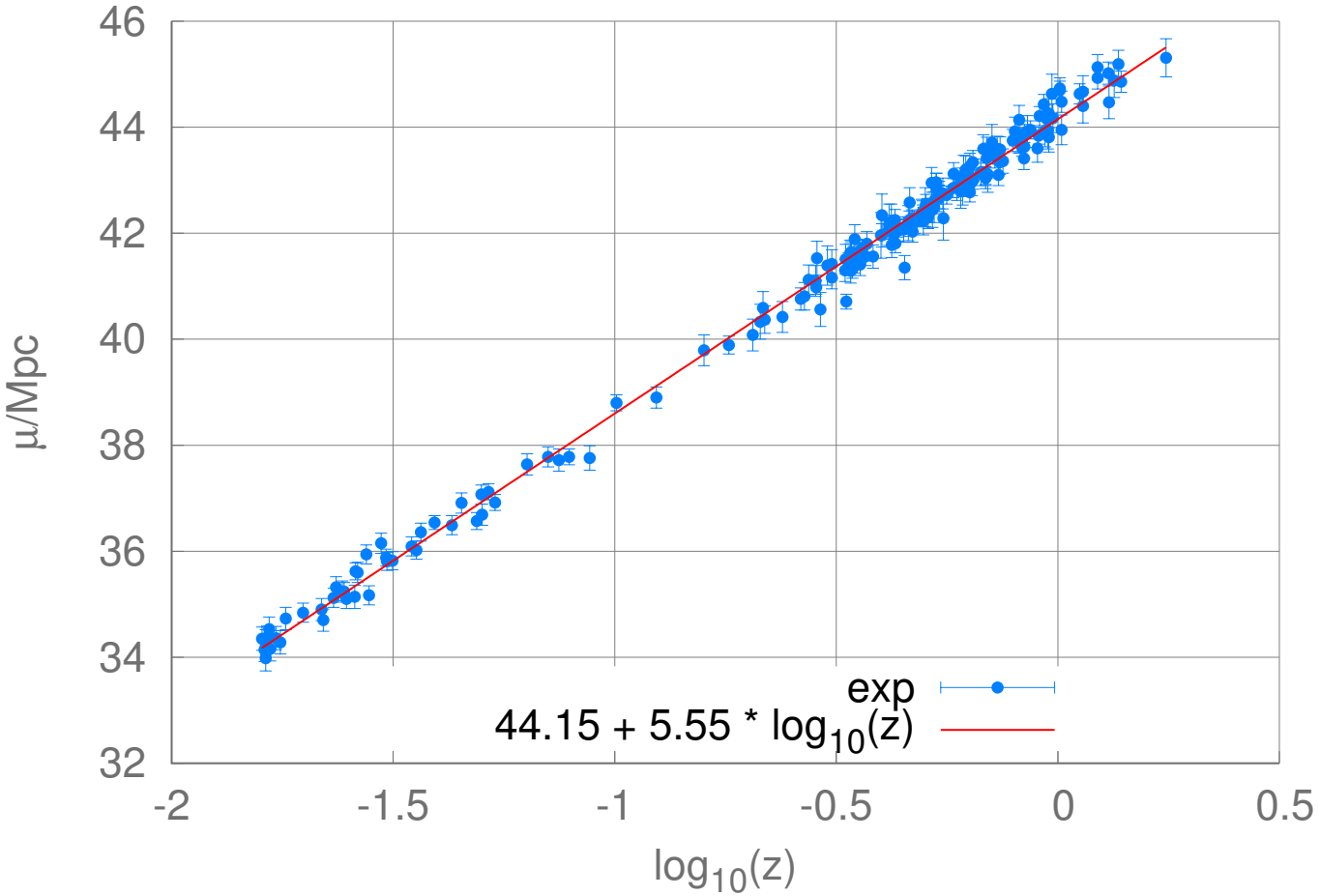
Dopasowanie do danych eksperymentalnych:
$\mu = a + b \log_{10}(z)$, gdzie $a = 25 + 5 \log_{10}(c/H_0)$.
Stąd powinniśmy otrzymać $H_0 = c/10^{(a-25)/5} \approx 44\text{km/s/Mpc}$.

Oryginalne dane z obserwacji Hubble'a opublikowane są w pracy [?].

1. Powtórzyć obliczenia Hubble'a. Znaleźć H_0.
2. Obliczyć H_0 na podstawie danych T. Davis [?].

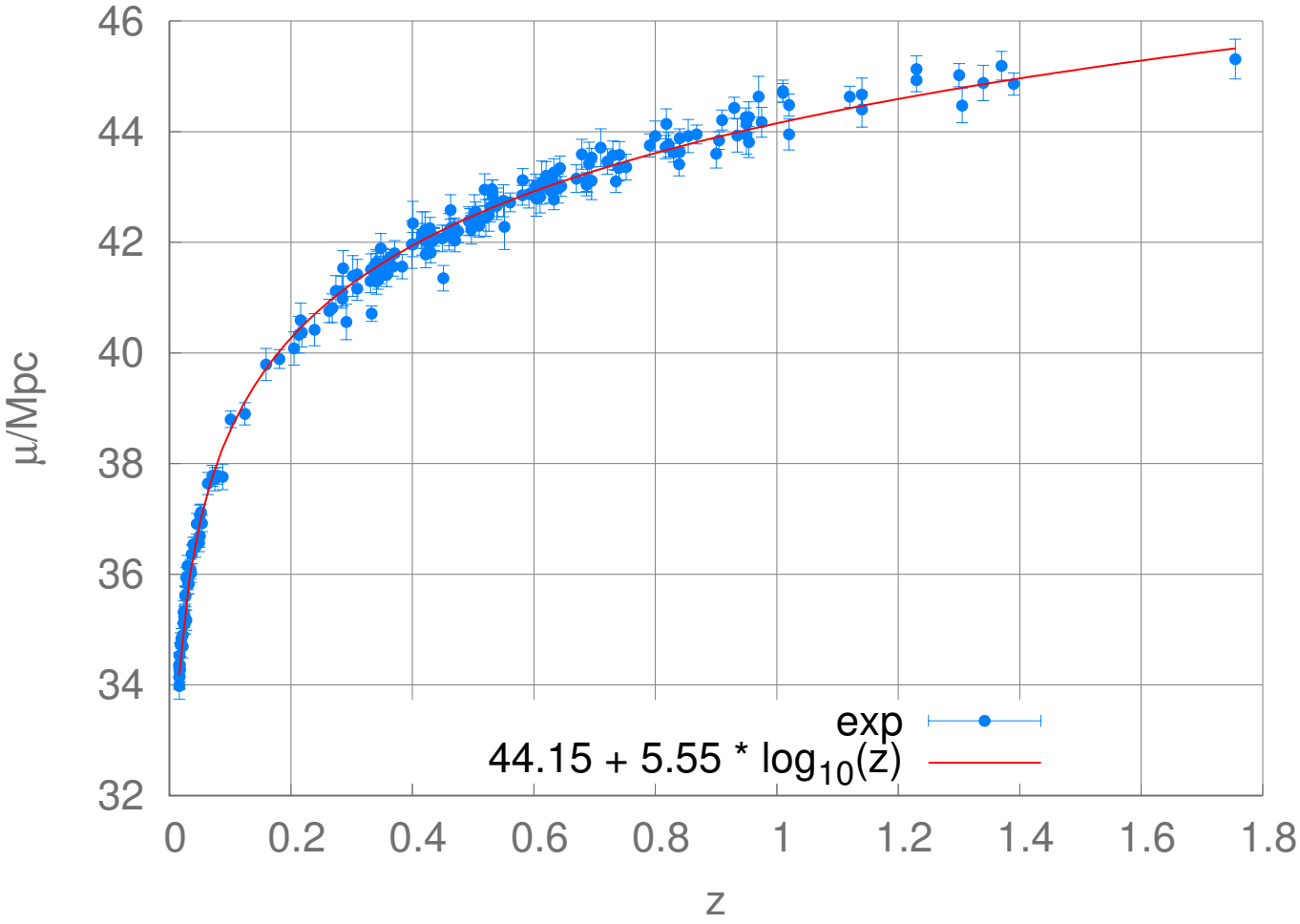
UWAGA techniczna, dotycząca programu…
Należy zwrocić uwagę na metodę czytania danych z plikow. (patrz metoda `czytajDane()` w pliku `Supernova.java`). Metoda jest przystosowana do omawianej sytuacji. W innych przypadkach należy ja zmienic.

Supernowe - Wynik dopasowania



Dopasowanie do danych T. Davis.

Supernowe - Wynik dopasowania



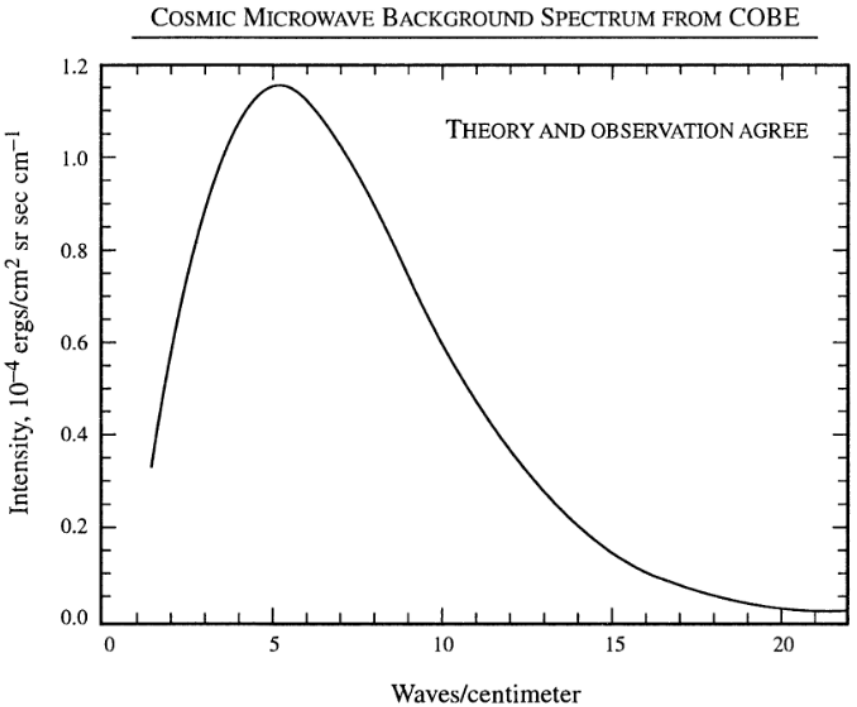
Dopasowanie do danych T. Davis.

CMBR

Problem 4.3
The specific intensity of a gas of photons with blackbody spectrum is

$$I_\nu = \frac{4\pi\hbar\nu^3/c^2}{\exp\{2\pi\hbar\nu\nu/k_B T\} - 1}\,. \tag{1.8}$$

Convert the specific intensity in Eq. (1.8) into an expression for what is plotted in the Figure, the energy per square centimeter per steradian per second. Note that the x-axis is I/λ, the inverse wavelength of the photons. Show that the peak of a 2.73K blackbody spectrum does lie at $1/\lambda = 5.38$ cm. (Dodelson)



Literatura

- Edwin hubble. A relation between distance and radial velocity among extra-galactic nebulae. Proc. N.A.S. Vol. 15, 168 (1929). Patrz e.g., `http://www.pnas.org/content/15/3/168.full.pdf`
- D.J. Fixsen, E.S. Cheng, J.M. Gales, J.C. Mather, R.A. Shafer, and E. L. Wright. The Astrophysical Journal, 473:576-587, 1996 December 20.
- S. Dodelson. Modern Cosmology, Academic Press (2003).
- Particle Data Group. `http://pdg.lbl.gov/2012/reviews/rpp2012-rev-bbang-cosmology.pdf`
- Tamara Davis. `http://dark.dark-cosmology.dk/~tamarad/SN/` Plik `Davis07_R07_WV07.dat`

Thank you!

Metoda najmniejszych kwadratów, χ^2

Dane $(x_i, y_i)\,, i = 1, 2, \ldots, N$ modelujemy za pomocą linii prostej $y(x) = a + bx$ minimalizując funkcję χ^2:

$$\chi^2(a, b) = \sum_{i=1}^{N} \left(\frac{y_i - a - bx}{\sigma_i} \right)^2 \tag{1}$$

gdzie σ_i jest błędem wielkości y_i. Wartości parametrów a i b otrzymamy, minimalizując χ^2. Wprowadzając oznaczenia

$$S = \sum_{i=1}^{N} 1/\sigma_i^2\,, \quad S_x = \sum x_i/\sigma_i^2\,, \quad S_y = \sum y_i/\sigma_i^2\,,$$

$$S_{xx} = \sum x_i^2/\sigma_i^2\,, \quad S_{xy} = \sum x_i y_i/\sigma_i^2\,, \tag{2}$$

otrzymamy

$$\begin{aligned} a &= \frac{S_{xx} S_y - S_x S_{xy}}{\Delta}\,, \\ b &= \frac{S S_{xy} - S_x S_y}{\Delta}\,, \end{aligned} \tag{3}$$

gdzie $\Delta = S S_{xx} - S_x^2$.

Błędy parametrów a i b I

Analiza propagacji błędów w danych y_i pokazuje, że błędy w dowolnej funkcji $f = f(x_i, y_i)$ określa wariancja σ_f^2

$$\sigma_f^2 = \sum_{i=1}^{N} \sigma_i^2 \left(\frac{\partial f}{\partial y_i} \right) . \tag{4}$$

W rozważanym przypadku

$$\sigma_a^2 = S_{xx}/\Delta , \quad \sigma_b^2 = S/\Delta . \tag{5}$$

Dodatkowo, tzw. kowariancja a i b jest

$$\mathrm{Cov}(a, b) = -S_x/\Delta . \tag{6}$$

Współczynnik korelacji r_{ab} między nieokreślonością a i nieokreślonością b jest równy

$$r_{ab} = \frac{-S_x}{\sqrt{SS_{xx}}} . \tag{7}$$

$r_{ab} > 0$ – błędy w a i b są prawdopodobnie jednakowego znaku (*korelacja*)
$r_{ab} < 0$ — błędy sa przeciwnego znaku (*antykorelacja*).

Dobroć dopasowania modelu do danych (y_i) pozwala ocenić znaczenie parametrów. Prawdopodobieństwo, że wartość χ^2 (Równanie ??) jest przypadkowa jest dane przez

$$Q(\frac{N-2}{2}, \frac{\chi^2}{2}) \,, \tag{8}$$

gdzie

$$Q(a,x) = \frac{\Gamma(a,x)}{\Gamma(a)} \,. \tag{9}$$

Tutaj $\Gamma(\dots)$ jest funkcją gamma Eulera lub niepełną funkcją gamma:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt \tag{10}$$

$$\Gamma(a,x) = \int_0^x e^{-t} t^{a-1}\, dt \,. \tag{11}$$

Formuły (??) wnoszą do obliczeń błędy numeryczne. Zapiszemy je inaczej (do celów programistycznych), definiując

$$t_i = \frac{1}{\sigma_i}\left(x_i - \frac{S_x}{S}\right), \quad i = 1, 2, \ldots, N, \qquad S_{tt} = \sum_{i=1}^{N} t_i^2 . \tag{12}$$

Wtedy

$$b = \frac{1}{S_{tt}} \sum \frac{t_i y_i}{\sigma_i}, \quad a = \frac{S_y - bS_x}{S} \tag{13}$$

$$\sigma_a^2 = \frac{1}{S}\left(1 + \frac{S_x^2}{SS_{tt}}\right), \quad \sigma_b^2 = \frac{1}{S_{tt}}, \tag{14}$$

$$\mathrm{Cov}(a, b) = -\frac{S_x}{SS_{tt}}, \quad r_{ab} = \frac{\mathrm{Cov}(a, b)}{\sigma_a \sigma_b} \tag{15}$$

Zadania.

1. Sprawdzić formuły (??)–(??).
2. Napisać program `fit(...)`, który oblicza wielkości a, b, σ_a^2, σ_b^2, r_{ab}, oraz $\mathrm{Cov}(a, b)$ dla zbioru danych (x_i, y_i), $i = 1, 2, ..., N$.

Dane (t_i, y_i), $i = 1, 2, \ldots, m$ modelujemy za pomocą funkcji

$$f(t) = c_1\phi_1(t) + c_2\phi_2(t) + \cdots + \phi_n(t)\,, \tag{16}$$

gdzie ϕ_j są zadanymi funkcjami, a c_i parametrami modelu. Oznaczmy

$$a_{ij} = \phi_j(t_i)\,. \tag{17}$$

Jeśli y oznacza wektor o elementach y_i, a c wektor o składowych c_i to równanie

$$\sum_j c_j\phi_j(t_i) \approx y_i\,, \quad i = 1, 2, \ldots, m\,, \tag{18}$$

przybliża dane (t_i, y_i). Można je zapisać w postaci

$$Ac \approx y \tag{19}$$

Równanie to rozwiążemy metodą SVD macierzy A.

Problem 4.3 [?] Odwrotnością długości fali jest ν/c. Zastąpienie ν przez c/λ w równaniu (1.8) daje

$$I_\nu = \frac{4\pi\hbar c}{\lambda^3}\frac{1}{\exp\left(2\pi\hbar c/\lambda k_B T\right)-1}$$

Wymiarem jest *energia* na Hz. Potrzebujemy *energii* na cm^{-1}, a więc mnożymy to przez c co daje

$$I_{1/\lambda} = \frac{4\pi\hbar c^2}{\lambda^3}\frac{1}{\exp\left(2\pi\hbar c/\lambda k_B T\right)-1}\,.$$

Wstawiając wartości liczbowe dostajemy

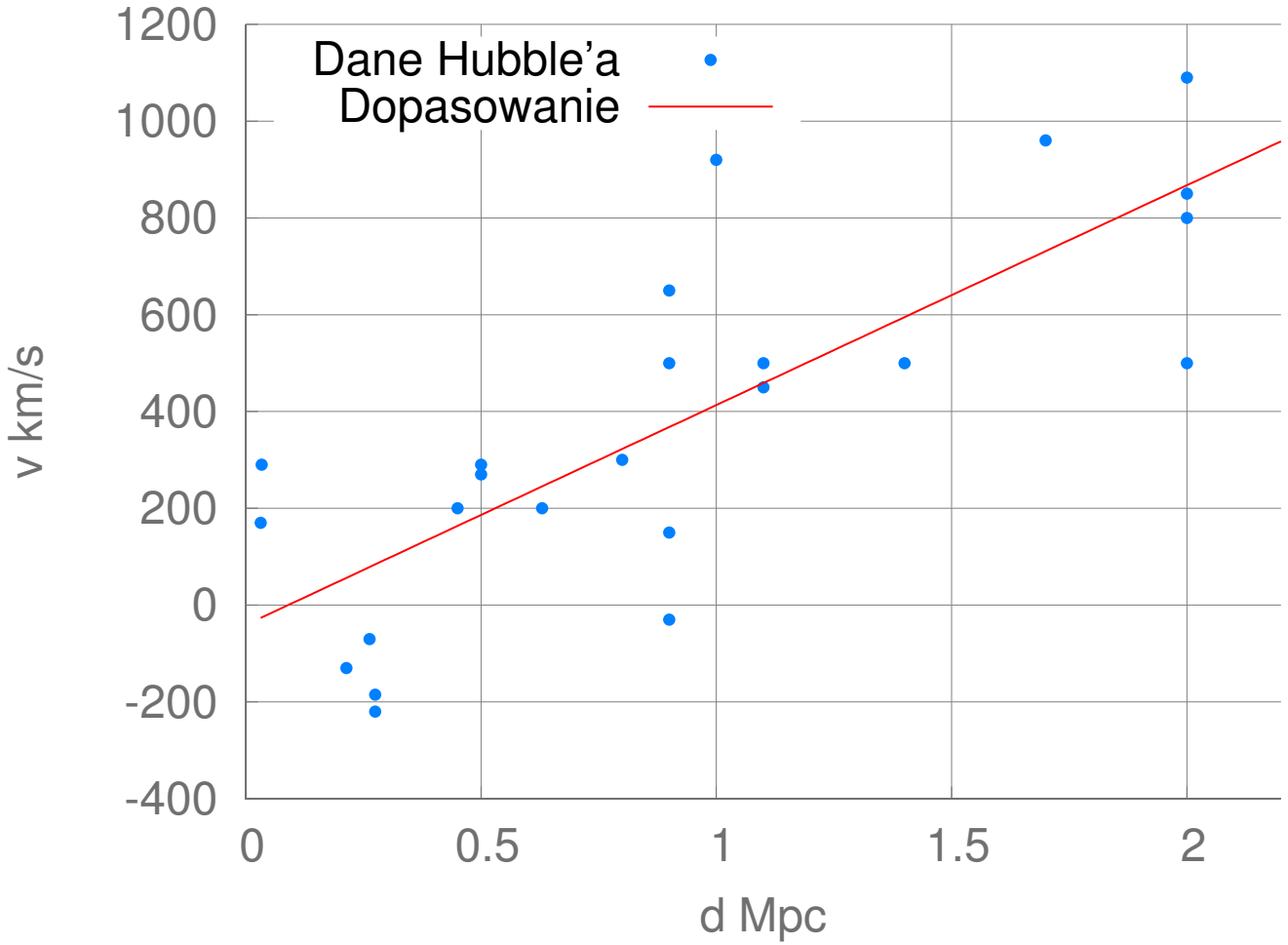
$$I_{1/\lambda} = 1.2\times 10^{-5}\mathrm{ergsec}^{-1}\mathrm{cm}^{-1}\mathrm{sr}^{-1}\left(\frac{cm}{\lambda}\right)^3\frac{1}{\exp\left(0.53\mathrm{cm}/\lambda\right)-1}\,.$$

Porównanie z Rysunkiem potwierdza prawdziwość wyrażenia. W celu znalezienia maksimum różniczkujemy I względem $1/\lambda$ i przyrównujemy do zera. Otrzymamy

$$\lambda = \frac{1}{3}\frac{\left(2\pi\hbar c/k_B T\right)}{1-\exp\left(-2\pi\hbar c/\lambda k_B T\right)}$$

Wartość $1/\lambda_{\mathrm{peak}} = 3/.53\mathrm{cm}^{-1}$. Dokładna wartość, z uwzględnieniem eksponenty równa się 2.82, czyli $1/\lambda_{\mathrm{peak}} = 5.3\mathrm{cm}^{-1}$ i zgadza się z wartością na rysunku.

Obserwacje Hubble'a (1929)



Program Supernova.java I

```
/**
   File Supernowa.java
   Kosmologia BIG-BANGu
*/
public class Supernova {

    static String url = "http://dark.dark-cosmology.dk/~tamarad/SN/";
    static String data_file_name = "Davis07_R07_WV07.dat";

    static double[]
        dane_z,                  // przesuniecie ku czerwieni
        dane_mu,                 // modul odleglosci supernowej
        dane_muErr;              // blad modulu odleglosci

    static double przeciecie, nachylenie, przeciecieErr, nachylenieErr;

    public static void main(String[] args) {

        System.out.println("# Dopasowanie danych supernowych do linii prostej");
        System.out.println("# " + url);
        System.out.println("# " + data_file_name);
```

```
        czytajDane();

        int n = dane_z.length;
        double[] logZ = new double[n];
        for (int i = 0; i < n; i++) {
              System.out.println(dane_z[i]+" "+dane_mu[i]+" "+dane_muErr[i]);
              logZ[i] = Math.log10(dane_z[i]);  // mu = a + b log_{10}(z)
        }
        double chikw = chi_kwadrat( logZ, dane_mu, dane_muErr );

        System.out.println("# nachylenie: "+ nachylenie+ "+/-"+ nachylenieErr);
        System.out.println("# przeciecie: "+ przeciecie+ "+/-"+ przeciecieErr);
        System.out.println("# chi_square: "+ chikw/(n-2));

        // Obliczenia H0: przeciecie = 25 + log10(c/H0), stąd:
        // c = 299 792 458 m / s ~ 3 x 10^5 km/s;
        // 1 Mpc = 3.08567758 x 10**19 km
        double lh = (przeciecie - 25)/ 5;
        double H0 = 3.0e5 / Math.pow(10, lh);
        System.out.println("# H0: " + H0 + " km/s/Mpc");
    }
```

```
/**
   Czyta dane z pliku Davis07_R07_WV07.dat
   UPROSCIC !!!
 */
static void czytajDane() { // wczytywanie danych z pliku
    String filename   = "Davis07_R07_WV07.dat";
    DataParser parser = new DataParser( filename );
    double[] doubles  = parser.getDoubles();
    int dim = doubles.length/3;
    System.out.println("# " + dim);

    dane_z     = new double[dim];
    dane_mu    = new double[dim];
    dane_muErr = new double[dim];

    for(int i = 0; i < dim; i++) {
        dane_z[i]     = doubles[3*i];
        dane_mu[i]    = doubles[3*i+1];
        dane_muErr[i] = doubles[3*i+2];
        //System.out.println(dane_z[i]+" "+dane_mu[i]+" "+dane_muErr[i]);
    }
}
```

```
/**
 * Oblicza chi-kwadrat
 * @param x[] wektor x
 * @param y[] wektor wartosci funkcji odpowiadajacy x[]
 * @param err[] bledy wartosci y[]
 */
static double chi_kwadrat(                  // wartosc chi_kwadrat
                          double[] x,       // wartosci x
                          double[] y,       // wartosci y
                          double[] err)     // bledy wartosci y
{
    int n = x.length;
    double chi_kw;

    double S = 0, S_x = 0, S_y = 0;
    for (int i = 0; i < n; i++) {
        S += 1 / err[i] / err[i];
        S_x += x[i] / err[i] / err[i];
        S_y += y[i] / err[i] / err[i];
    }

    double[] t = new double[n];
    for (int i = 0; i < n; i++)
```

```
            t[i] = (x[i] - S_x/S) / err[i];

        double S_tt = 0;
        for (int i = 0; i < n; i++)
            S_tt += t[i] * t[i];

        nachylenie = 0;
        for (int i = 0; i < n; i++)
            nachylenie += t[i] * y[i] / err[i];
        nachylenie /= S_tt;

        przeciecie = (S_y - S_x * nachylenie) / S;
        przeciecieErr = Math.sqrt((1 + S_x * S_x / S / S_tt) / S);
        nachylenieErr = Math.sqrt(1 / S_tt);

        chi_kw = 0;
        for (int i = 0; i < n; i++) {
            double roznica = (y[i] - przeciecie - nachylenie * x[i]) / err[i];
            chi_kw += roznica * roznica;
        }
        return chi_kw;
    }
}
```

Program DataParser.java I

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class DataParser {
    private final List<Double> doublelist = new ArrayList<>();
    private String filename;
    private List<String> inputlines;

    public DataParser(String filename) {
        this.filename = filename;
        parseData();
    }

    private void parseData() {
        Path inputpath = Paths.get(filename);
```

Program DataParser.java II

```
        try{inputlines = Files.readAllLines(inputpath,StandardCharsets.UTF_8);
        }catch(IOException e){ };
        // set up a loop over the input data
        for (String line : inputlines) {
            int charpos = line.indexOf(";");
            if (charpos < 0) { // just data, not a comment line ";"
                line = line.trim().replaceAll(" +", " ");
                String[] parts = line.split(" ");
                // skip name at index 0
                for(int i=1;i<parts.length;i++){
                    //System.out.print(parts[i]+" ");
                    doublelist.add(Double.parseDouble(parts[i]));
                }
                //System.out.println();
            }
        }
    }

    public double[] getDoubles() {
        double[] ret = new double[doublelist.size()];
        int cnt = 0;
        for (Double d : doublelist) {
            ret[cnt++] = d;
```

```
        }
        return ret;
    }

    // Przykład użycia DataParser
    public static void main(String[] args)  {
        String filename   = "Davis07_R07_WV07.dat";
        DataParser parser = new DataParser(filename);
        double[] doubles  = parser.getDoubles();

        double[] z      = new double[doubles.length/3];
        double[] mu     = new double[doubles.length/3];
        double[] mu_err = new double[doubles.length/3];

        for(int i=0; i<doubles.length/3; i++) {
            z[i]      = doubles[3*i];
            mu[i]     = doubles[3*i+1];
            mu_err[i] = doubles[3*i+2];
            //System.out.println(z[i]+" "+mu[i]+" "+mu_err[i]);
        }
    }
}
```

Program DavisRead.java I

Program DavisRead.java czyta plik Davis.dat otrzymany z pliku Davis07....dat poprzez usunięcie z niego początkowych linii komentarzy.

```
import java.io.*;
import java.util.Scanner;
import java.util.Locale;
import java.util.ArrayList;

public class DavisRead {

    public static void main(String[] args)  throws IOException {

        File myFile = new File("Davis.dat");
        Scanner in = new Scanner(myFile);
        in.useLocale(Locale.US);
        String s;

        ArrayList<Double> z  = new ArrayList<Double>();
        ArrayList<Double> mu = new ArrayList<Double>();
        ArrayList<Double> er = new ArrayList<Double>();
```

```
        while ( in.hasNext() ) {
            s  = in.next();
            z.add(in.nextDouble());
            mu.add(in.nextDouble());
            er.add(in.nextDouble());
        }

        for(int i=0;i<z.size();i++){
            System.out.println(z.get(i)+" "+mu.get(i)+" "+er.get(i));
        }
    }
}
```