Weighted linear fit

Created using Maple 14.01

Jake Bobowski

```
> restart;
  with(StringTools) :
  with(Statistics) :
  with(plots) :


  FormatTime("%m-%d-%Y, %H:%M");
```

"08-06-2012, 18:08" **(1)**

In this example, *Irms* will be plotted as a function of *Vrms*. Here *Irms* is the current through a resistor *R* across which voltage *Vrms* is applied.

This Maple input enters a list of the measured rms current. The current was measured in milliamps, so divide by 1000. The ac current was measured using a Fluke 8012A DMM. The uncertainty in the measurement is 1% of the reading plus two digits.

$$> \quad Irms := \frac{[9.07, 7.94, 6.42, 4.72, 2.96, 1.911, 0.467, 0.0469]}{1000};$$

$$\Delta Irms := \frac{[.19, .18, .16, .15, .13, .03, .015, .0015]}{1000};$$

Irms := [0.009070000000, 0.007940000000, 0.006420000000, 0.004720000000, 0.002960000000, 0.001911000000, 0.0004670000000, 0.00004690000000]

ΔIrms := [0.0001900000000, 0.0001800000000, 0.0001600000000, 0.0001500000000, 0.0001300000000, 0.00003000000000, 0.00001500000000, 0.000001500000000] **(2)**

The peak-to-peak voltage of the resistor was measured using the TDS1001 oscilloscope (**measured in V**). The uncertainty in the voltage estimate is estimated from the change in voltage due to one click of the cursor position (and also the line thickness of the curve displayed on the oscilloscope screen). (**measured in V**)

```
> Vp2p := [5.62, 4.94, 4.02, 2.96, 1.86, 1.19, 0.302, 0.047];
  ΔVp2p := [0.04, 0.04, 0.04, 0.04, 0.04, 0.03, 0.01, 0.008];
```

Vp2p := [5.62, 4.94, 4.02, 2.96, 1.86, 1.19, 0.302, 0.047]

ΔVp2p := [0.04, 0.04, 0.04, 0.04, 0.04, 0.03, 0.01, 0.008] **(3)**

Since the current was measured as an rms value, convert the peak-to-peak voltage measurements (and ΔVp2p) to rms by dividing by 2 $\sqrt{2}$. (**measured in V**) Note that, the *evalf* function forces Maple to evaluate the expression and display the results as decimal numbers.

$$> \quad Vrms := evalf\left(\frac{Vp2p}{2\,\mathrm{sqrt}(2)}\right);$$

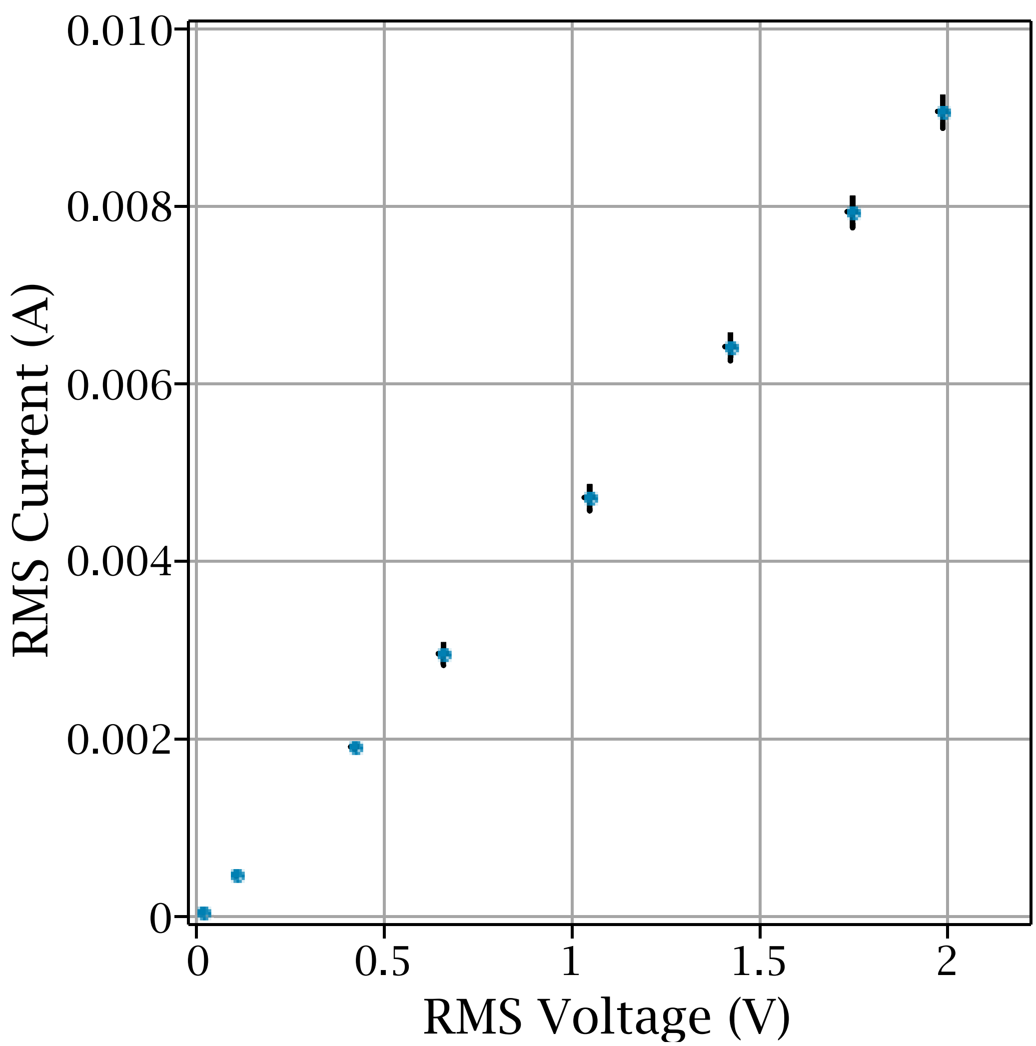
$$\Delta Vrms := evalf\left(\frac{\Delta Vp2p}{2\,\mathrm{sqrt}(2)}\right);$$

Vrms := [1.986970054, 1.746553749, 1.421284630, 1.046518036, 0.6576093062, 0.4207285348, 0.1067731239, 0.01661700935]

$\Delta Vrms$:= [0.01414213562, 0.01414213562, 0.01414213562, 0.01414213562, 0.01414213562, 0.01060660172, 0.003535533905, 0.002828427125] **(4)**

Give the plot the name *DataPlot*. In this case, the plot won't be displayed. Include a second command *display* to show the plot (this is why the package *plots* was loaded at the beginning). The colon at the end of the first command supresses the output. Note that the *x*-errors are much smaller than the *y*-errors (often smaller than the size of the data points).

```
> DataPlot := ScatterPlot( Vrms, Irms, xerrors = ΔVrms, yerrors = ΔIrms, axes = boxed, view
      = [0 .. 2.2, 0 .. 10e−3], labels = [typeset("RMS Voltage (V)"),
     typeset("RMS Current (A)") ], labeldirections = ["horizontal", "vertical"], tickmarks = [7,
     8 ] , axesfont = [Times, 12 ] , labelfont = [Times, 14 ], axis = [gridlines = [thickness = 1 ]],
      symbolsize = 10, symbol = solidcircle, thickness = 2 ) :
  display(DataPlot);
```



Let's first fit the data to a straight line, and then make incremental improvements to the fit.

Use the *LinearFit* function of Maple to find the best-fit line.

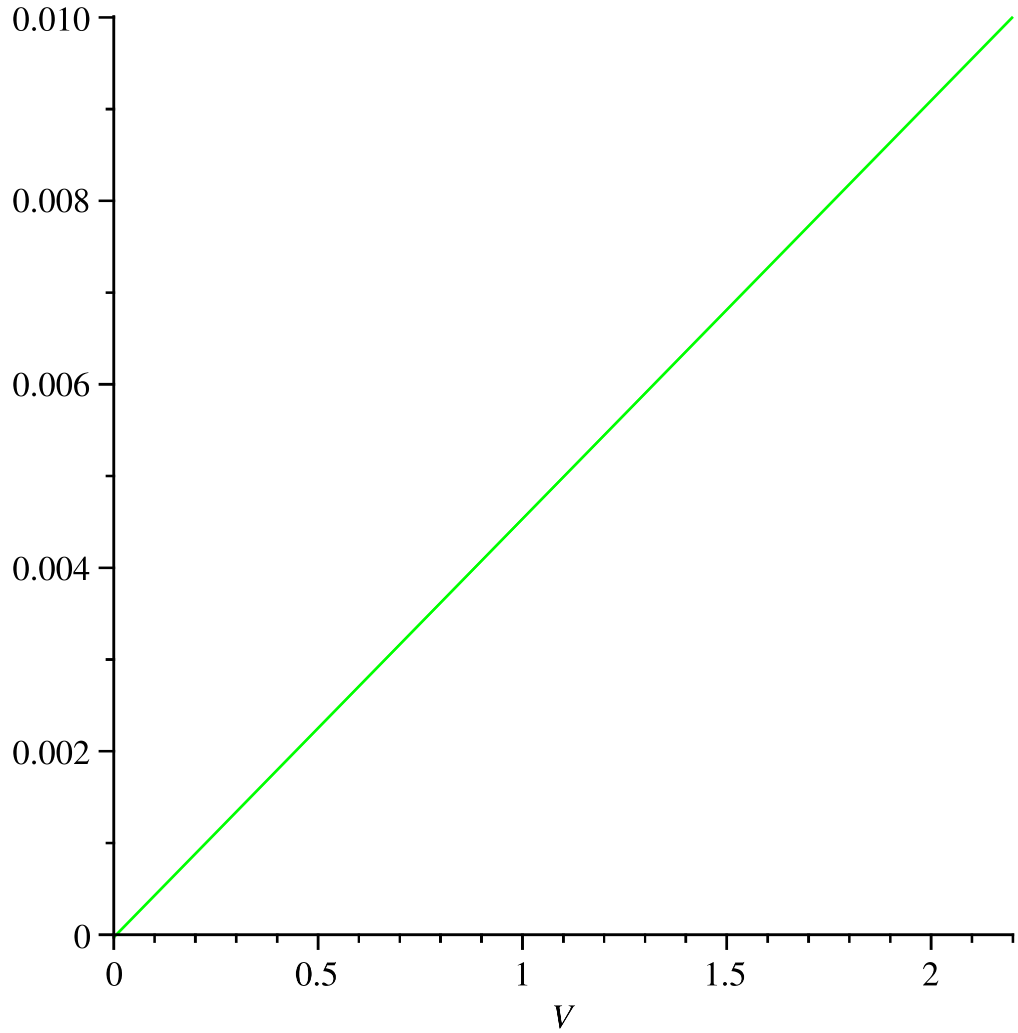
The format of the command below is *LinearFit*(fcn, x data, y data, variable). The notation [1,*V*] means that our function has a single linear variable V and single constant (which will be the y-intercept). The output gives the best fit parameters (slope and y-intercept) extracted from the fit. These parameters are given in the equation of the straight line. The slope is the number in front of the V, and the y-intercept is the other number.

> $fcn := LinearFit([1, V], Vrms, Irms, V);$

$$fcn := -0.0000302937398459901 + 0.00456260995751092\, V \tag{5}$$

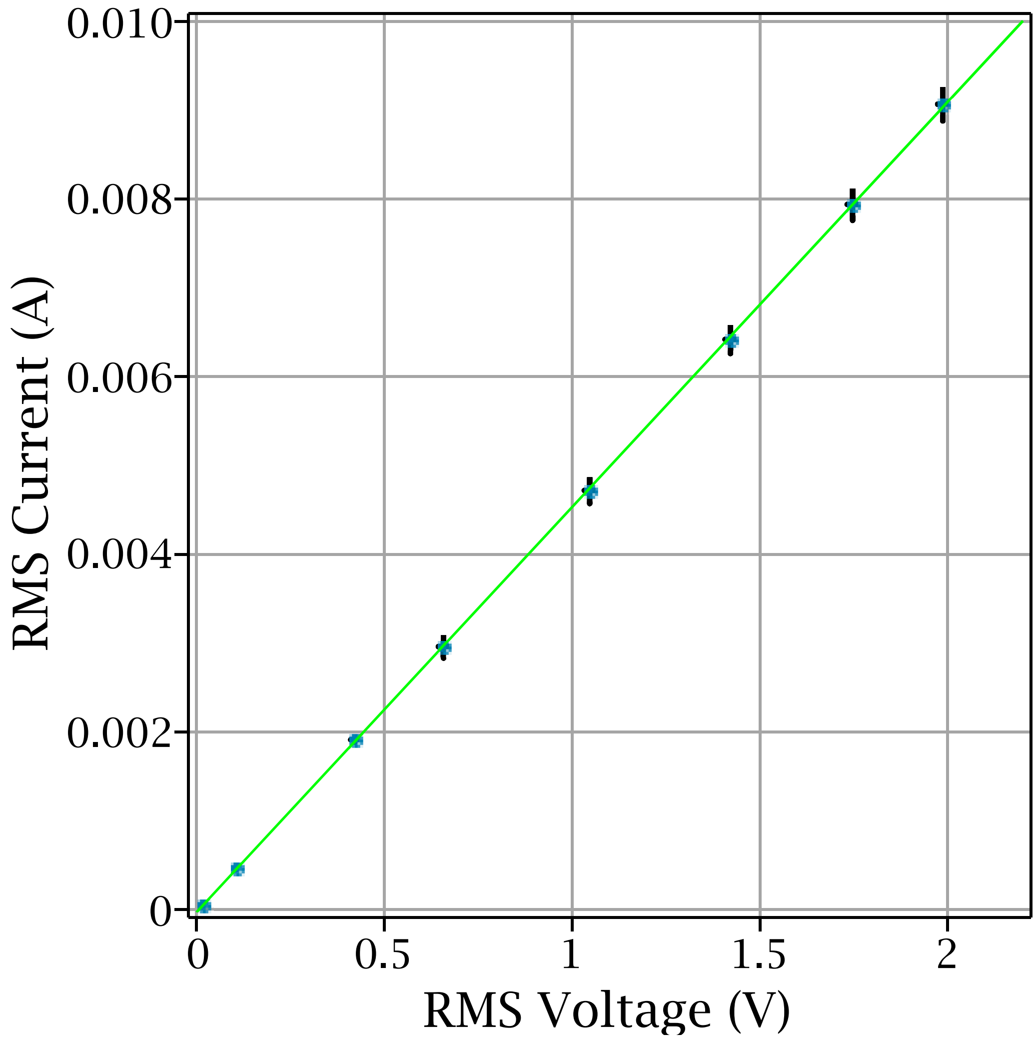
Now plot the fit function defined by fcn over the range $0 < V < 2.2$. Give the plot the name *FitPlot*

> $FitPlot := plot(fcn, V = 0 .. 2.2, colour = green) :$
> $display(FitPlot);$



Use *display* to show both *DataPlot* and *FitPlot*.

> $display(DataPlot, FitPlot);$



The final step will be to improve the linear fit so that it is a *weighted* linear fit. The weighting means that points with larger error bars a given less importance than points with small error bars. Notice that in the plot above the *y*-error bars are much bigger than the *x*-error bars. Therefore, we will use only the *y*-error bars to weight the fit. Other methods are needed to include contributions from the *x*-errors (add slope times *x*-errors to *y*-errors in quadrature, for example).

weights are given by $1/\Delta Irms^2$. In this way, small error bars have large weight ...

> $yWeights := \left[seq\left(\frac{1}{\Delta Irms[i]^2}, i = 1..nops(\Delta Irms)\right)\right];$

$$yWeights := [2.770083102\ 10^7, 3.086419753\ 10^7, 3.906250000\ 10^7, 4.444444444\ 10^7, 5.917159763\ 10^7, 1.111111111\ 10^9, 4.444444444\ 10^9, 4.444444444\ 10^{11}] \quad \textbf{(6)}$$

Use the *weights* option in *LinearFit*. Note that the parameters (slope and *y*-intercept) are slightly changed by the weighting.

> $weightedFCN := LinearFit([1, V], Vrms, Irms, V, weights = yWeights);$

$$weightedFCN := -0.0000291474167787945 + 0.00458160260522137\ V \quad \textbf{(7)}$$

Before plotting the final results, add one more option to the *LinearFit* function - want to display: the fit function, the best fit parameters, and the uncertainties in the determinations of the two fit parameters.

> $weightedFCN := LinearFit([1, V], Vrms, Irms, V, weights = yWeights, output = [leastsquaresfunction, parametervalues, standarderrors]);$

$$weightedFCN := \left[-0.0000291474167787945 + 0.00458160260522137\ V, \left[\begin{array}{c} -0.0000291474167787945 \\ 0.00458160260522137 \end{array}\right], \left[\begin{array}{cc} 5.80512828584488\ 10^{-7} & 0.0000144484152355823 \end{array}\right]\right] \quad \textbf{(8)}$$

Now *weightedFCN* is a list. To access the actual function, select the first element of the list

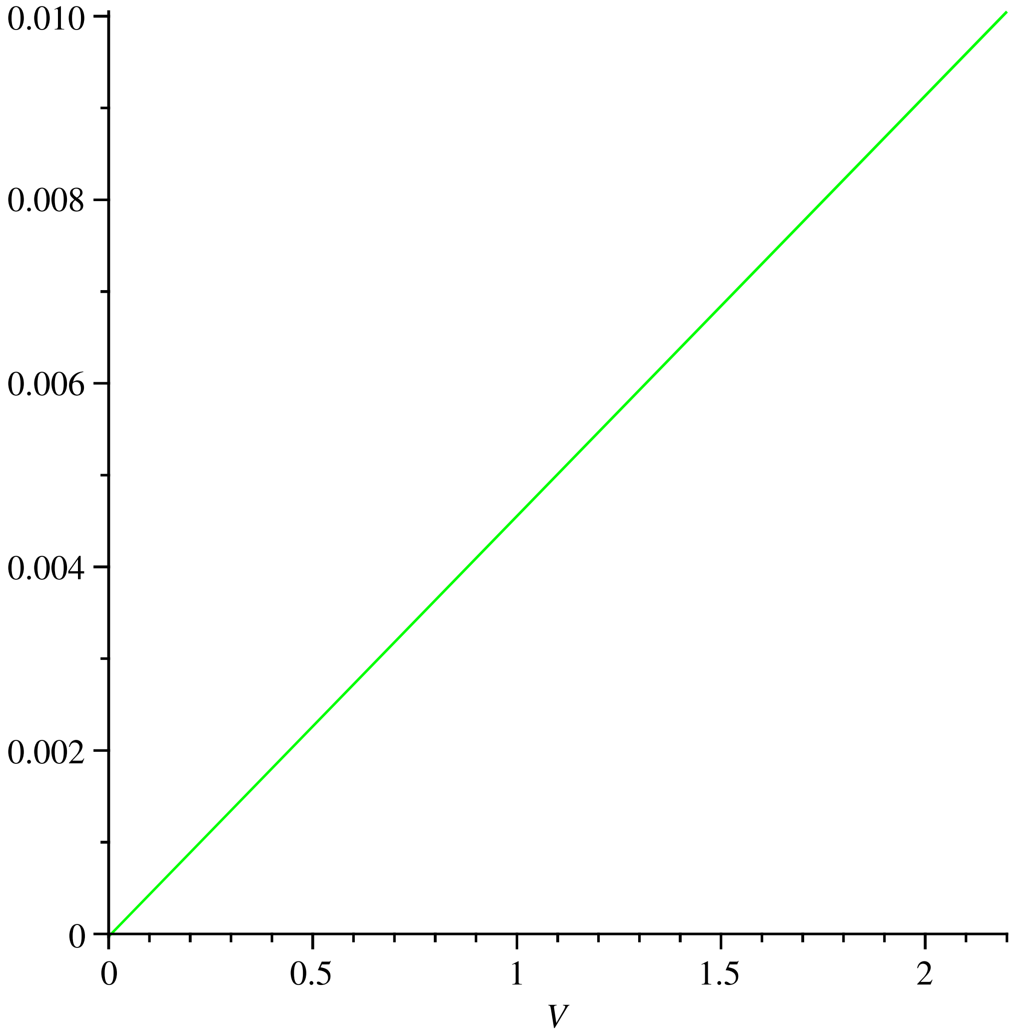
> $weightedFCN[1]$

$$-0.0000291474167787945 + 0.00458160260522137\ V \quad \textbf{(9)}$$

These results tell us that: slope m = 0.004582+/-0.000014 1/ohms, y-intercept b = -0.0000291+/-0.0000006 A

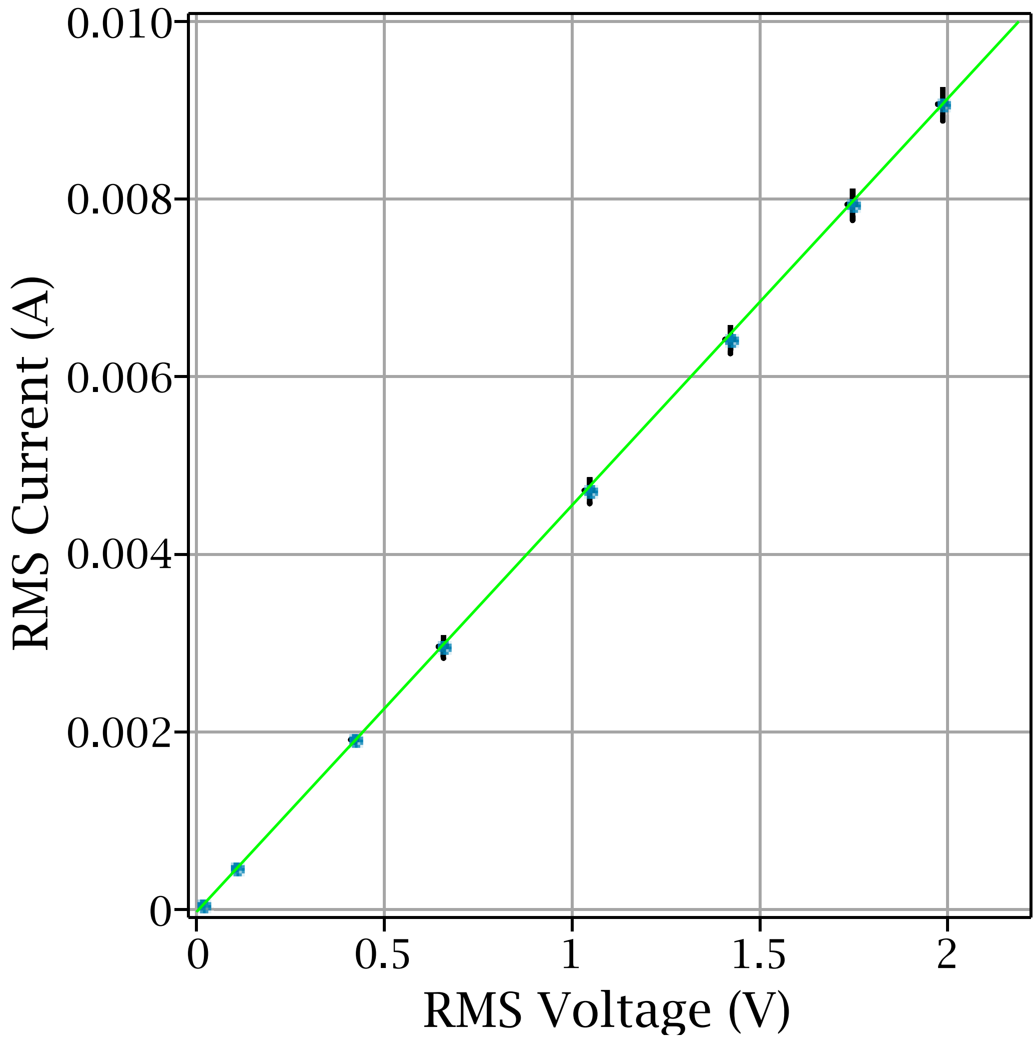
Plot the new fits result

> $weightedFitPlot := plot(weightedFCN[1], V = 0\,..\,2.2, colour = green) :$
$display(weightedFitPlot);$



Display the weighted fit and the data

```
> display(DataPlot, weightedFitPlot);
```



Finally, can calculate the resistance of the resistor used from the slope m of the weighted fit: $R = 1/m$

From propagation of errors:

$\delta R = \left| \frac{\delta n}{m^2} \right|$

> *parameters* := *weightedFCN*[2];

$$parameters := \begin{bmatrix} -0.0000291474167787945 \\ 0.00458160260522137 \end{bmatrix} \quad \textbf{(10)}$$

> *b* := *parameters*[1];
> *m* := *parameters*[2];

$$b := -0.0000291474167787945$$
$$m := 0.00458160260522137 \quad \textbf{(11)}$$

> *errors* := *weightedFCN*[3];

$$errors := \begin{bmatrix} 5.80512828584488\ 10^{-7} & 0.0000144484152355823 \end{bmatrix} \quad \textbf{(12)}$$

> δb := *errors*[1];

$\delta n := errors[2];$

$$\delta b := 5.80512828584488\ 10^{-7}$$

$$\delta n := 0.0000144484152355823 \tag{13}$$

> $R := \dfrac{1}{m};$
>
> $\delta R := \dfrac{\delta n}{m^2};$

$$R := 218.264237684072$$

$$\delta R := 0.688312062146850 \tag{14}$$

So, the final result is: R = 218.3+/-0.7 Ω.

>