Determination of the constant of gravity

The data

Importing the data

We import the data from a csv-List and display them in a grid:

In[1]:=
```
data = Import[
   "/Users/ihn/Documents/Teaching/VP-Leitung/DataAnalysis/2013NewProgram/Lectures
     given in 2013/7. Lecture/Data_z_t.csv"];
Grid[Prepend[data, {Style["z (cm)", Bold], Style["t (ms)", Bold]}],
 Background → {None, {Gray, {LightGray, White}}},
 Dividers → {Black, {Black, Black}}, Frame → True]
```

Out[2]=

z (cm)	*t* (ms)
10	144.6
10	144.6
10	143.9
20	203.2
20	203.1
20	203.2
30	247.9
30	248.3
30	248.2
40	286.3
40	286.5
40	286.3
50	319.9
50	319.6
60	350.2
60	350.8
70	378.5
70	378.5
70	378.4
80	404.4
80	404.2
90	429.2
90	429.1
100	452.1
100	452.5
110	474.4
110	474.3
120	495.5
120	495.4
124	503.6
124	503.5

Converting the data to SI units

In the data list, the lengths z_j are given in cm and the times are given in ms. We convert to m and s.

In[3]:=
```
data = Table[{1. × 10^-2 data[[j, 1]], 10^-3 data[[j, 2]]}, {j, 1, Length[data]}];
Grid[Prepend[data, {Style["z (m)", Bold], Style["t (s)", Bold]}],
 Background → {None, {Gray, {LightGray, White}}},
 Dividers → {Black, {Black, Black}}, Frame → True]
```

Out[4]=

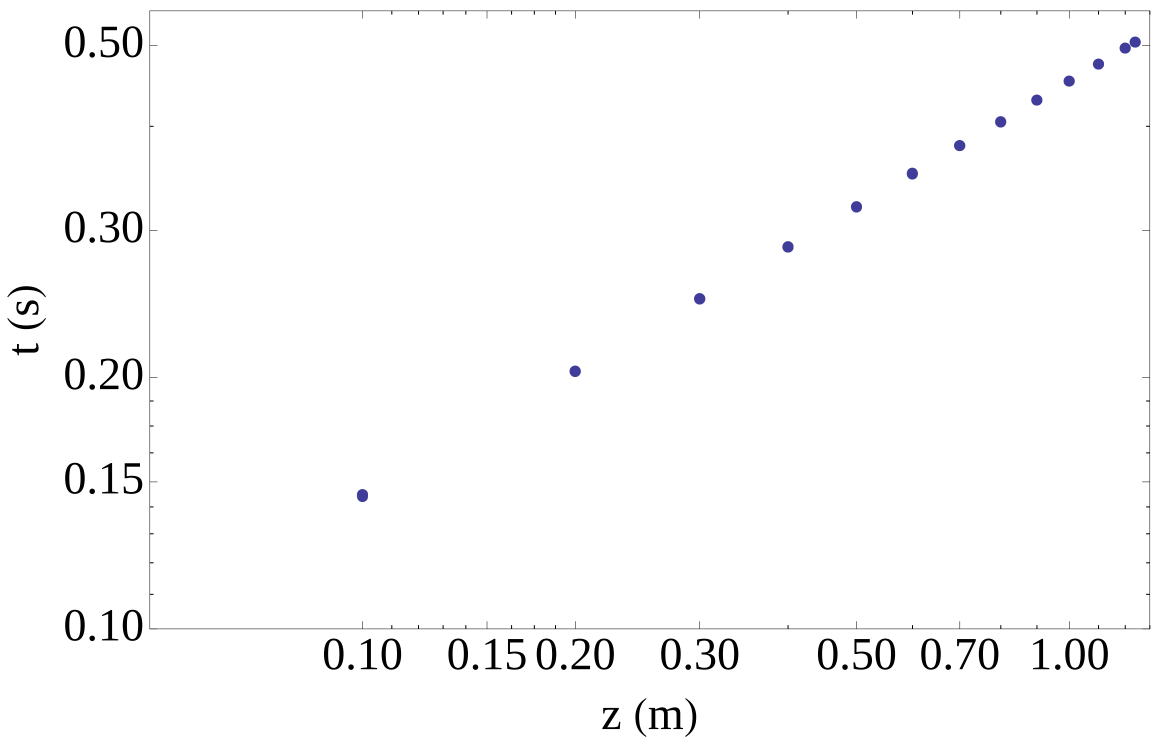
z (m)	*t* (s)
0.1	0.1446
0.1	0.1446
0.1	0.1439
0.2	0.2032
0.2	0.2031
0.2	0.2032
0.3	0.2479
0.3	0.2483
0.3	0.2482
0.4	0.2863
0.4	0.2865
0.4	0.2863
0.5	0.3199
0.5	0.3196
0.6	0.3502
0.6	0.3508
0.7	0.3785
0.7	0.3785
0.7	0.3784
0.8	0.4044
0.8	0.4042
0.9	0.4292
0.9	0.4291
1.	0.4521
1.	0.4525
1.1	0.4744
1.1	0.4743
1.2	0.4955
1.2	0.4954
1.24	0.5036
1.24	0.5035

Plotting the data

Since the data are described by a power law, we plot them on double logarithmic scale and expect a straight line.

In[5]:=
```
g1 = ListLogLogPlot[data, PlotRange → {{0.05, 1.30}, {0.1, 0.55}},
  Ticks → {{0.1, 0.2, 0.3, 0.5, 1}, {0.1, 0.2, 0.3, 0.5}},
  FrameLabel → {"z (m)", "t (s)"}, LabelStyle → 24,
  PlotMarkers → Automatic, ImageSize → 600, Frame → True]
```

Out[5]=



Analysis using Model 1 following the lecture

Defining a function for calculating the mean square error

The mean square error is defined as

$$Q(g) = \frac{1}{n} \sum_{j=1}^{n} \left(t_j^m - h(g, z_j)\right)^2 \quad (1)$$

We define this function in *Mathematica* using our specific fitting function

$h(g,z_j) = \sqrt{\frac{2 z_j}{g}}$.

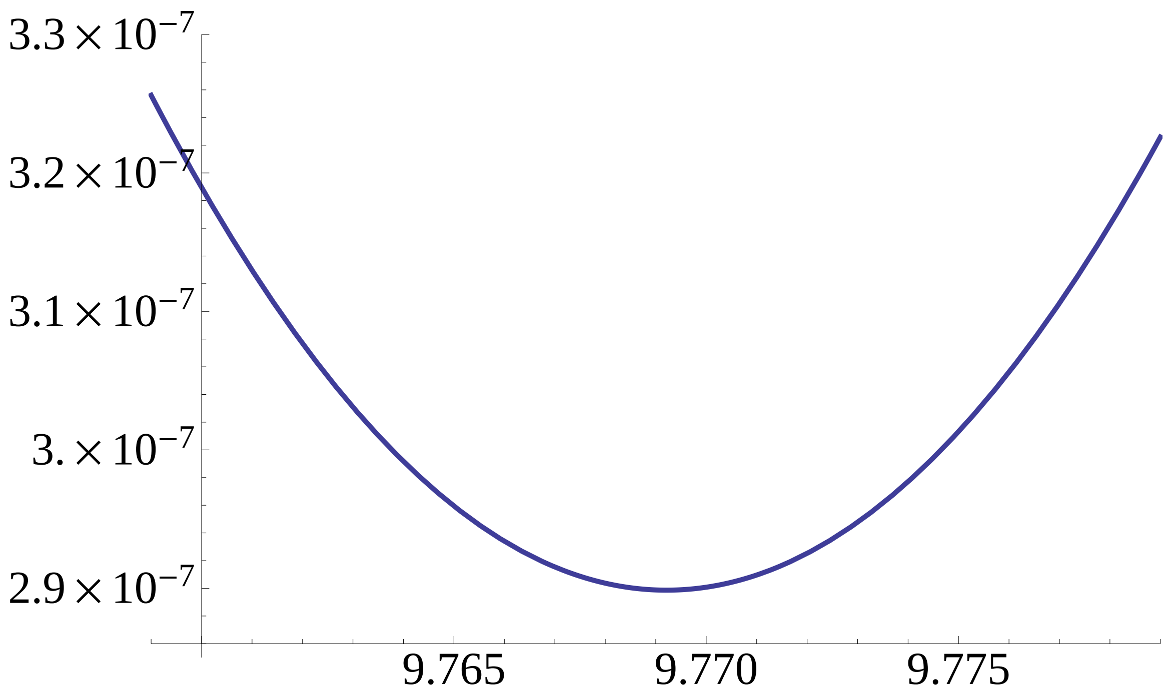
In[6]:=

$$Q[g\_, data\_] := \frac{1}{Length[data]} \sum_{j=1}^{Length[data]} \left(data[[j, 2]] - \sqrt{\frac{2\,data[[j, 1]]}{g}}\right)^2;$$

Plotting the mean square error function

In[7]:=
```
Plot[Q[g, data], {g, 9.759, 9.779},
 PlotRange → {{9.759, 9.779}, {0.285 × 10^-6, 0.33 × 10^-6}},
 LabelStyle → 24, PlotStyle → {Thickness[0.005]}, ImageSize → 600]
```

Out[7]=



Numerical minimization of the mean square error

The *Mathematica* function FindMinimum does the numerical minimization:

In[8]:=
```
minQ = FindMinimum[Q[g, data], {g, 9.78}];
```

From the result we extract the least mean square error Q_{min}...

In[9]:=
```
Qmin = minQ[[1]];
```

... and the estimated parameter value $\hat{g}$:

In[10]:=
```
gg = g /. minQ[[2]];
```

The estimate $\hat{\sigma}^2$ is then given by

In[11]:=
$$\sigma est2 = \frac{\text{Length[data]}}{\text{Length[data]} - 1} \text{Qmin};$$

Standard error of the parameters

The uncertainty for $\hat{\sigma}^2$ is given by

In[12]:=
```
σσ2 = (Length[data] Qmin)/(Length[data] - 1) Sqrt[2/(Length[data] - 3)];
```

The uncertainty of $\hat{g}$ is

In[13]:=
```
σg = ((Length[data] - 1)/(2 Qmin) ∂g ∂g Q[g, data] /. {g → gg})^(-1/2);
```

Now we can print our final results, properly rounded:

In[14]:=
```
Print["g = (", Round[gg, 0.0001], " ± ", Round[σg, 0.0001], ") m/s²"]
Print["σ² = (", Round[10⁶ σest2, 0.001], " ± ", Round[10⁶ σσ2, 0.001], ") ms²"]
```

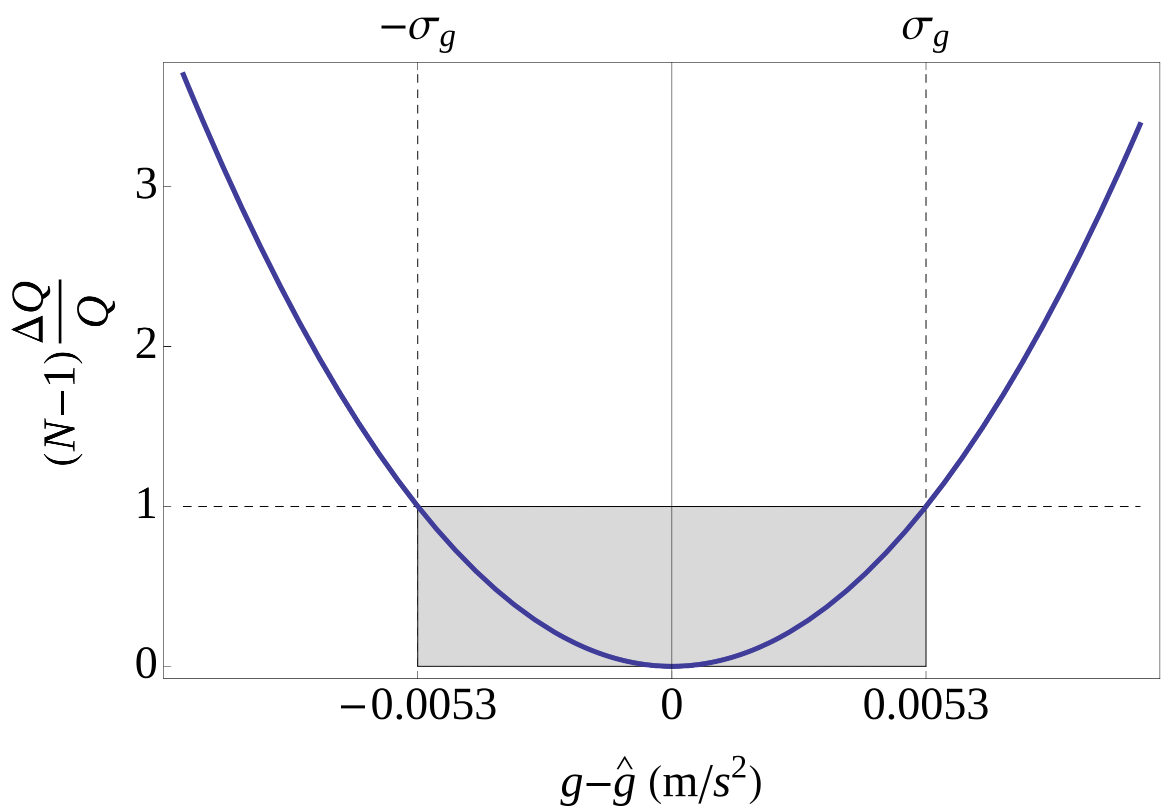
```
g = (9.7692 ± 0.0053) m/s²
σ² = (0.3 ± 0.08) ms²
```

Graphical determination of the standard error of g

Alternatively we can determine σ_g using the graphical method. Here is the plot of the normalized mean square error function (N-1)ΔQ/Q:

In[16]:=
```
Show[{Plot[(Length[data] - 1) (Q[g + gg, data] - Qmin)/Qmin,
   {g, 9.759 - gg, 9.779 - gg}, LabelStyle → 24, PlotStyle → {Thickness[0.005]},
   ImageSize → 600, Frame → True, FrameLabel → {"g-ĝ (m/s^2)", "(N-1) ΔQ/Q"},
   FrameTicks → {{0, {σg, Round[σg, 0.0001]}, {-σg, Round[-σg, 0.0001]}},
     {0, 1, 2, 3}, {{σg, "σg"}, {-σg, "-σg"}}, {}}],
  Graphics[{{Dashed, Line[{{9.759 - gg, 1}, {9.779 - gg, 1}}],
     Line[{{-σg, 0}, {-σg, 3.7}}], Line[{{σg, 0}, {σg, 3.7}}]},
    {LightGray, EdgeForm[Black], Rectangle[{-σg, 0}, {σg, 1}]}}],
  Plot[{(Length[data] - 1) (Q[g + gg, data] - Qmin)/Qmin}, {g, 9.759 - gg, 9.779 - gg},
   Frame -> True, FrameTicks → {{}, {}, {}, {}}, PlotStyle → {Thickness[0.005]}]}]
```

Out[16]=



We may find the two solutions of (N-1)ΔQ/Q = 1 numerically:

In[17]:=
```
σg1 = sg /. FindRoot[(Length[data] - 1) (Q[sg + gg, data] - Qmin)/Qmin == 1, {sg, 0.005, 0, 0.1}]
```

Out[17]= 0.00531603

In[18]:=
```
σg2 =
  sg /. FindRoot[(Length[data] - 1) (Q[sg + gg, data] - Qmin)/Qmin == 1, {sg, -0.005, -0.1, 0}]
```

Out[18]= -0.0053117

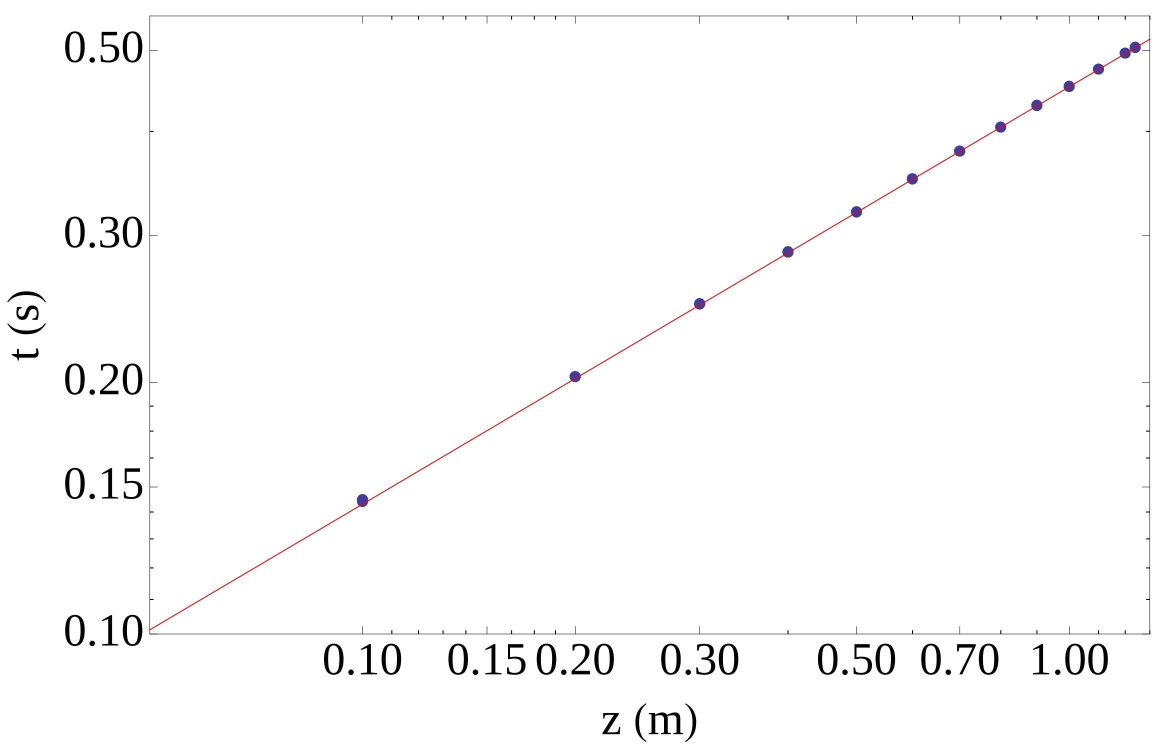
These results agree with our previous error estimate.

Plotting the data with the fitted curve

We plot the resulting fit together with the data: this plot shows us graphically, how well the systematic part of the model describes our data.

In[19]:=
```
Show[g1, LogLogPlot[Sqrt[2 z/gg], {z, 0.05, 1.30}, PlotStyle → {Red}, Frame → True]]
```

Out[19]=



Inspecting the residuals of the fit

We inspect the fitting residuals, in order to see graphically, if the residuals resemble the assumptions that we have made about the noise in our data.

In[20]:=
```
residuals =
  Table[{data[[j, 1]], data[[j, 2]] - Sqrt[(2 data[[j, 1]])/gg]}, {j, 1, Length[data]}];
Grid[Prepend[residuals, {Style["z (m)", Bold], Style["r (s)", Bold]}],
 Background → {None, {Gray, {LightGray, White}}},
 Dividers → {Black, {Black, Black}}, Frame → True]
```

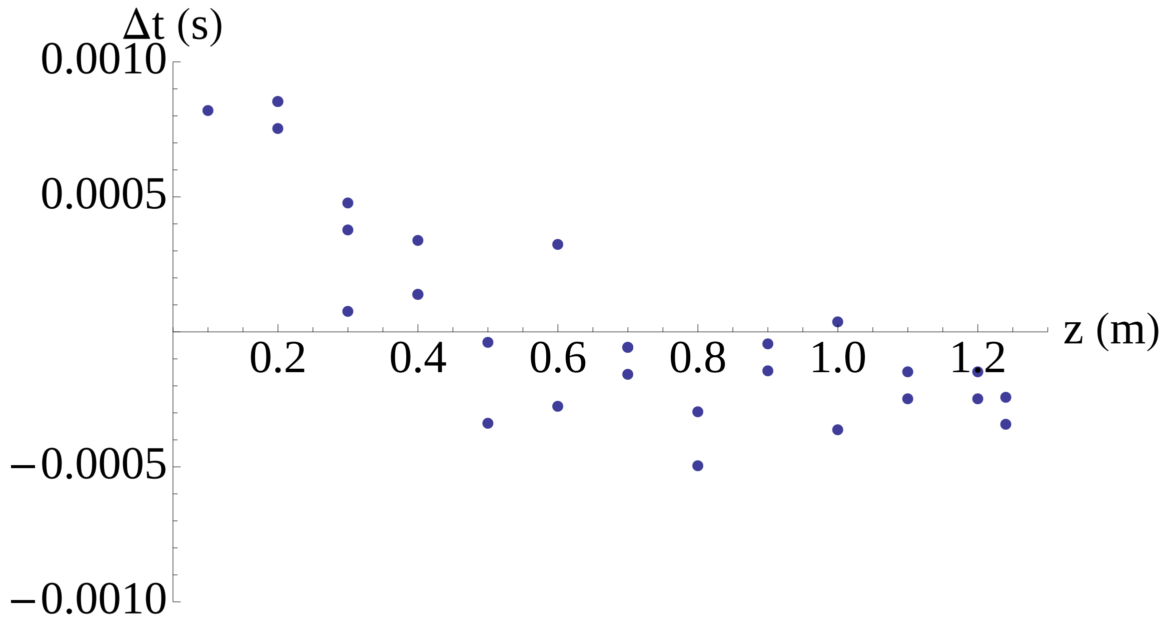
Out[21]=

z (m)	*r* (s)
0.1	0.00151793
0.1	0.00151793
0.1	0.00081793
0.2	0.000851396
0.2	0.000751396
0.2	0.000851396
0.3	0.0000745852
0.3	0.000474585
0.3	0.000374585
0.4	0.00013586
0.4	0.00033586
0.4	0.00013586
0.5	−0.0000412347
0.5	−0.000341235
0.6	−0.000278063
0.6	0.000321937
0.7	−0.0000595741
0.7	−0.0000595741
0.7	−0.000159574
0.8	−0.000297208
0.8	−0.000497208
0.9	−0.0000462098
0.9	−0.00014621
1.	−0.000365233
1.	0.0000347667
1.1	−0.00014954
1.1	−0.00024954
1.2	−0.00015083
1.2	−0.00025083
1.24	−0.00024396
1.24	−0.00034396

Systematic deviations of the fitted curve from the data can be detected by plotting the residuals versus the z_j.

In[22]:=
```
ListPlot[residuals, PlotRange → {{0.05, 1.30}, {-0.001, 0.001}},
 AxesLabel → {"z (m)", "Δt (s)"}, LabelStyle → 24,
 PlotMarkers → Automatic, ImageSize → 600, AxesOrigin → {0.05, 0}]
```

Out[22]=



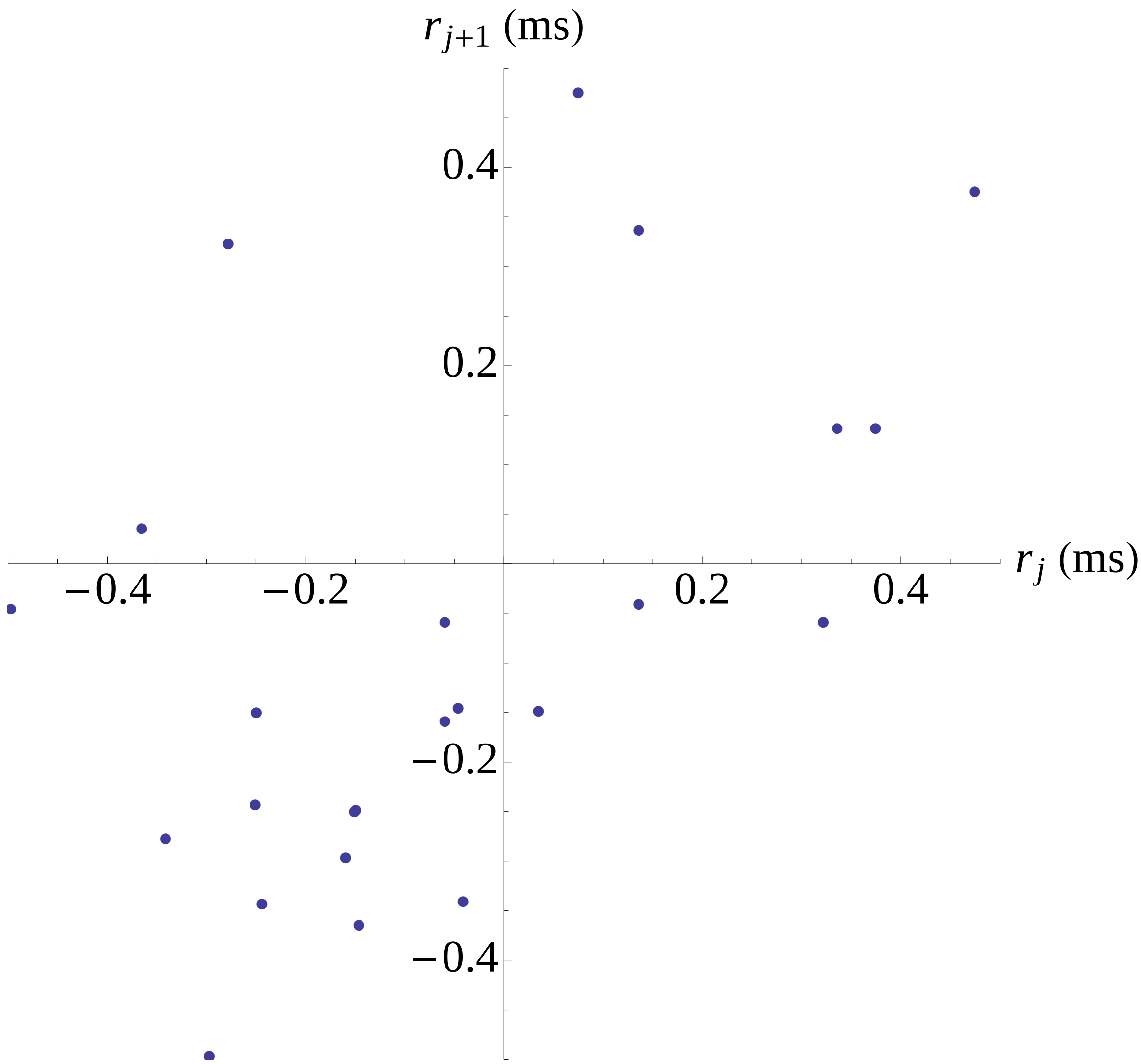
There is a clear trend in the fitting residuals, which indicates a systematic error.

Advanced topic: Further investigation of the residuals with different techniques (beyond the lecture)

For further inspection we generate a correlation plot:

In[23]:=
```
ListPlot[Table[{10^3 residuals[[j, 2]], 10^3 residuals[[j + 1, 2]]},
   {j, 1, Length[data] - 1}], PlotRange → {{-0.5, 0.5}, {-0.5, 0.5}},
  AxesLabel → {"r_j (ms)", "r_{j+1} (ms)"}, LabelStyle → 24, PlotMarkers → Automatic,
  ImageSize → 600, AxesOrigin → {0, 0}, AspectRatio → 1]
```

Out[23]=

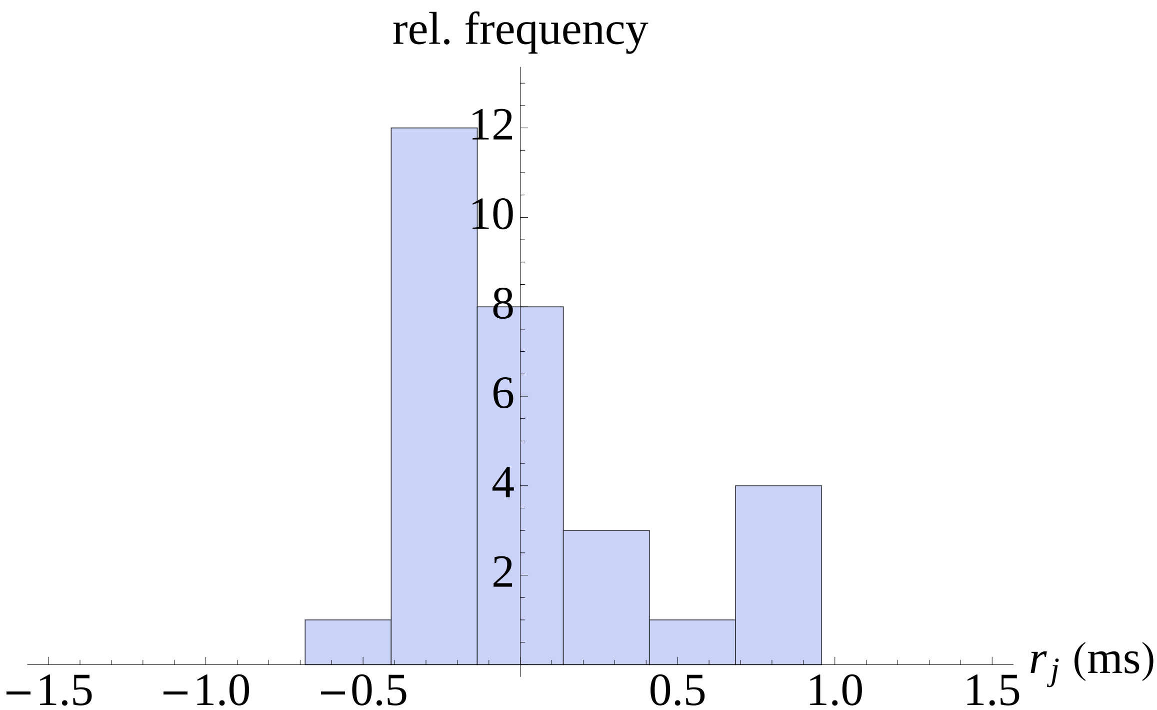


From the correlation plot we do not easily see any suspicious behavior in this case.

Another way to look at the residuals is a histogram plot. We choose histogram bars with a bin width of $\sqrt{\hat{\sigma}^2}/2$.

In[24]:=
```
Histogram[10^3 residuals[[Range[1, Length[data]], 2]],
  10^3 {-11 Sqrt[σest2]/4, 11 Sqrt[σest2]/4, Sqrt[σest2]/2}, AxesOrigin → {0, 0},
  LabelStyle → 24, AxesLabel → {"r_j (ms)", "rel. frequency"}, ImageSize → 600]
```

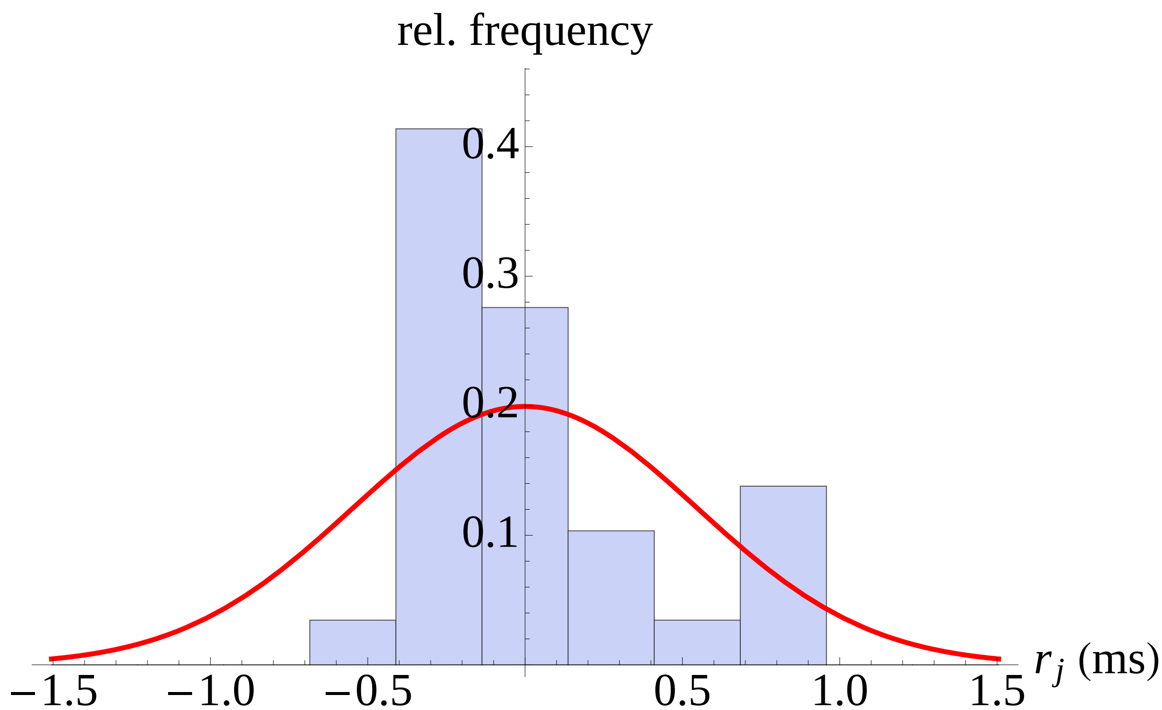
Out[24]=



This distribution of counts should be brought about by a normal distribution, as assumed in our model. We therefore plot the relative frequency (counts normalized to the number of data points) together with the normal distribution which has σ_{est} as its standard deviation.

In[25]:=
```
Show[Histogram[10^3 residuals[[Range[1, Length[data]], 2]],
  10^3 {-11 Sqrt[σest2]/4, 11 Sqrt[σest2]/4, Sqrt[σest2]/2}, "Probability", AxesOrigin → {0, 0},
  LabelStyle → 24, AxesLabel → {"r_j (ms)", "rel. frequency"}, ImageSize → 600],
 Plot[(10^3 Sqrt[σest2])/2 PDF[NormalDistribution[0, 10^3 Sqrt[σest2]], x],
  {x, -11/4 10^3 Sqrt[σest2], 11/4 10^3 Sqrt[σest2]}, PlotStyle → {Red, Thickness[0.005]}]]
```

Out[25]=



However, what is now missing, is a way to compare the relative frequency with the red distribution. To this end we may use the number of counts in each bin for estimating the probabilities that a particular measured value of the noise arises within this bin. For this purpose we generate a list of counts in the individual bins of the histogram:

In[26]:=
```
hl = HistogramList[10^3 residuals[[Range[1, Length[data]], 2]],
   10^3 {-11 Sqrt[σest2]/4, 11 Sqrt[σest2]/4, Sqrt[σest2]/2}][[2]]
```

Out[26]= {0, 0, 0, 1, 12, 8, 3, 1, 4, 0, 0}

For estimating a probability from the number of counts in each bin, we use the mean of the Beta-Distribution BetaDistribution[k+1,n-k+1]. Below we generate a list of these values, which will later be plotted.

In[27]:=
```
ml = Table[{(j - 6) (10^3 Sqrt[σest2])/2,
    Mean[BetaDistribution[hl[[j]] + 1, Length[data] - hl[[j]] + 1]]}, {j, 1, 11}];
Grid[Prepend[ml, {Style["bin center (ms)", Bold], Style["〈p〉", Bold]}],
 Background → {None, {Gray, {LightGray, White}}},
 Dividers → {Black, {Black, Black}}, Frame → True]
```

Out[28]=

bin center (ms)	〈p〉
-1.36823	1/33
-1.09458	1/33
-0.820938	1/33
-0.547292	2/33
-0.273646	13/33
0.	3/11
0.273646	4/33
0.547292	2/33
0.820938	5/33
1.09458	1/33
1.36823	1/33

Below we generate a list of lines that will be used as the error bars for the probability estimate. The error bars span the range from the 5% to the 95% quantile.

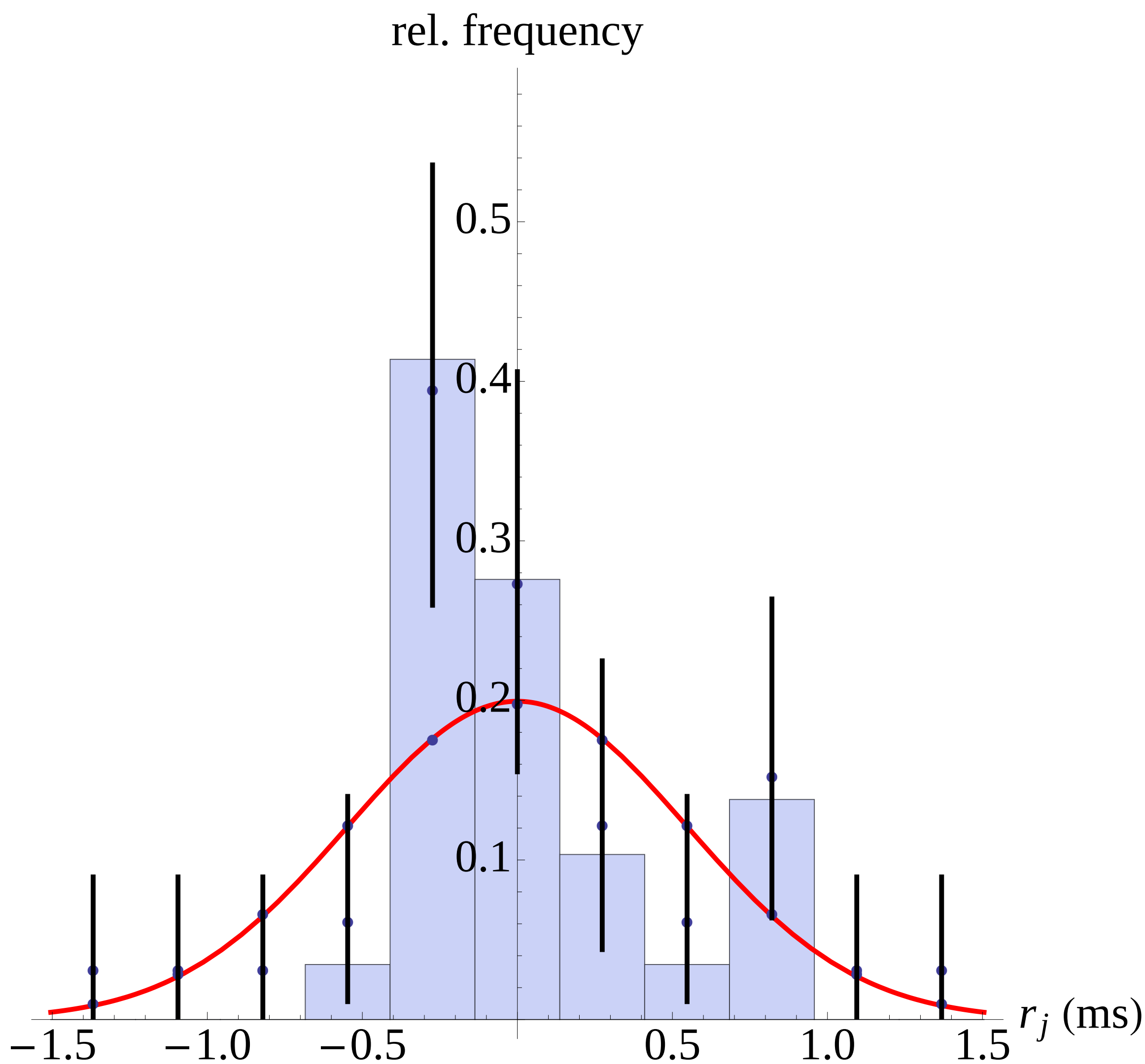
In[29]:=
```
eb = Table[Line[{{(j - 6) (10^3 Sqrt[σest2])/2,
      N[Quantile[BetaDistribution[hl[[j]] + 1, Length[data] - hl[[j]] + 1], 0.95]]},
    {(j - 6) (10^3 Sqrt[σest2])/2, N[Quantile[BetaDistribution[hl[[j]] + 1,
         Length[data] - hl[[j]] + 1], 0.05]]}}], {j, 1, 11}]
```

Out[29]=
```
{Line[{{-1.36823, 0.0893682}, {-1.36823, 0.00160163}}],
 Line[{{-1.09458, 0.0893682}, {-1.09458, 0.00160163}}],
 Line[{{-0.820938, 0.0893682}, {-0.820938, 0.00160163}}],
 Line[{{-0.547292, 0.139849}, {-0.547292, 0.0112195}}],
 Line[{{-0.273646, 0.535639}, {-0.273646, 0.259662}}],
 Line[{{0., 0.406057}, {0., 0.155278}}],
 Line[{{0.273646, 0.224816}, {0.273646, 0.0438454}}],
 Line[{{0.547292, 0.139849}, {0.547292, 0.0112195}}],
 Line[{{0.820938, 0.263597}, {0.820938, 0.0636527}}],
 Line[{{1.09458, 0.0893682}, {1.09458, 0.00160163}}],
 Line[{{1.36823, 0.0893682}, {1.36823, 0.00160163}}]}
```

Below we plot the all these results in one plot. The plot comprises the histogram of the data, this time plotted as a relative frequency (counts normalized to the number of data points), the normal distribution that results from our estimated value σ_{est} (red curve with blue dots), and the estimated probabilities with their error bars (blue dots with black error bars).

In[30]:=
```
Show[Histogram[10^3 residuals[[Range[1, Length[data]], 2]],
   10^3 {-11 Sqrt[σest2]/4, 11 Sqrt[σest2]/4, Sqrt[σest2]/2}, "Probability", AxesOrigin → {0, 0},
   LabelStyle → 24, AxesLabel → {"r_j (ms)", "rel. frequency"}, ImageSize → 600],
  ListPlot[ml, PlotMarkers → Automatic],
  Plot[10^3 Sqrt[σest2]/2 PDF[NormalDistribution[0, 10^3 Sqrt[σest2]], x], {x, -11/4 10^3 Sqrt[σest2],
    11/4 10^3 Sqrt[σest2]}, PlotStyle → {Red, Thickness[0.005]}], ListPlot[
   Table[{k 10^3 Sqrt[σest2]/2, (CDF[NormalDistribution[0, Sqrt[σest2]], Sqrt[σest2]/4 + k Sqrt[σest2]/2] -
        CDF[NormalDistribution[0, Sqrt[σest2]], -Sqrt[σest2]/4 + k Sqrt[σest2]/2])},
    {k, -5, 5, 1}], LabelStyle → 24, PlotMarkers → Automatic],
  Graphics[{Thickness[0.005], eb}], AspectRatio → 1]
```

Out[30]=



There is one histogram bar suggesting a probability for the bin that can hardly be produced by the normal distribution. (We have to be careful here: since we are dealing with probabilities, such a situation *can* occur; there is no reason that a normal distribution would not bring about such an exceptional bin. However, we would have to call this a *rare event*.)

Analysis using Model I1 following the lecture

Defining a new mean square error function

The new fitting function is

$$t_j = \sqrt{\frac{2\left(z_j - z_0\right)}{g}} =: h\left(g, z_0, z_j\right) \tag{2}$$

The parameters of the model are now g, z_0, and σ. The mean square error function of model II is now

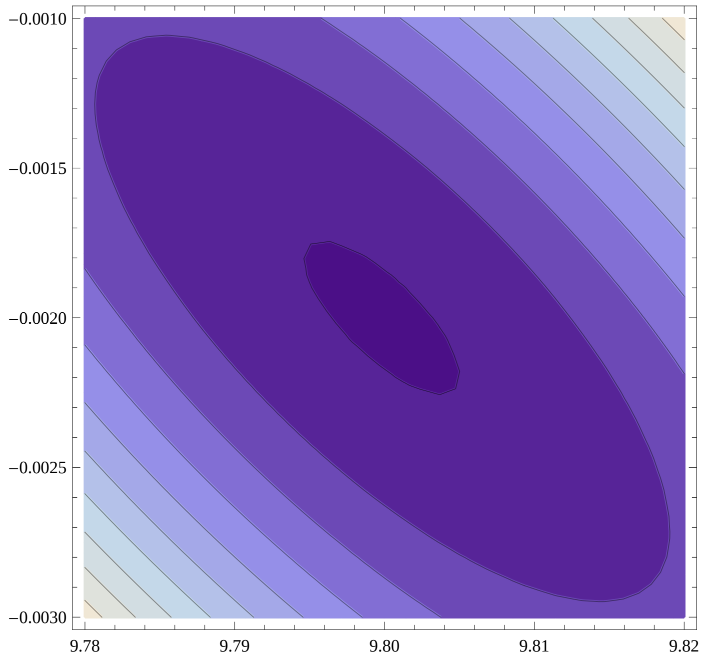
In[31]:=
$$Q[g\_, z0\_, data\_] := \frac{1}{\text{Length}[data]} \sum_{j=1}^{\text{Length}[data]} \left(data[[j, 2]] - \sqrt{\frac{2\,(data[[j, 1]] - z0)}{g}} \right)^2;$$

Plotting the mean square error function

We use a contour plot to look at the mean square error function around its minimum.

In[32]:=
```
ContourPlot[Q[g, z0, data], {g, 9.78, 9.82}, {z0, -0.003, -0.001}]
```

Out[32]=



The contour lines are ellipses with their main axes tilted with respect to the coordinate axes. This indicates that the parameters g and z_0 are correlated.

Numerical minimization of the mean square error

In[33]:=
```
minQ = FindMinimum[Q[g, z0, data], {g, 9.78}, {z0, 0}];
Qmin = minQ[[1]];
gg = g /. minQ[[2]];
zz0 = z0 /. minQ[[2]];
```
$$\sigma est2 = \frac{\text{Length}[data]}{\text{Length}[data] - 2}\, Qmin;$$

Standard error of the parameters

In[38]:=
$$\sigma\sigma 2 = \frac{\text{Length}[data]}{\text{Length}[data] - 2}\, Qmin \sqrt{\frac{2}{\text{Length}[data] - 4}};$$

In order to find the uncertainties in the estimated parameters g and z_0 we calculate the Hesse Matrix:

In[39]:=
```
hessematrix =
  ( ∂g ∂g Q[g, z0, data] /. {g → gg, z0 → zz0}    ∂z0 ∂g Q[g, z0, data] /. {g → gg, z0 → zz0}
    ∂g ∂z0 Q[g, z0, data] /. {g → gg, z0 → zz0}   ∂z0 ∂z0 Q[g, z0, data] /. {g → gg, z0 → zz0} )
  ;
```

Inversion of the Hesse matrix gives the matrix of the variances and covariances of the parameters:

In[40]:=
```
correlationmatrix = Inverse[(Length[data] - 2)/(2 Qmin) hessematrix];
σg = Sqrt[correlationmatrix[[1, 1]]];
σz0 = Sqrt[correlationmatrix[[2, 2]]];
ρ = correlationmatrix[[1, 2]]/(σg σz0);
```

Here we print our final results:

In[72]:=
```
Print["g = (", Round[gg, 0.0001], "±", Round[σg, 0.0001], ") m/s^2"]
Print["z_0 = (", Round[10^3 zz0, 0.01], "±", Round[10^3 σz0, 0.01], ") mm"]
Print["σ^2 = (", Round[10^6 σest2, 0.001], "±", Round[10^6 σσ2, 0.001], ") ms^2"]
Print["ρ_{g,z_0} = ", Round[ρ, 0.01]]
```

$g = (9.7998 \pm 0.0033)\ m/s^2$

$z_0 = (-2. \pm 0.16)\ mm$

$\sigma^2 = (0.049 \pm 0.013)\ ms^2$

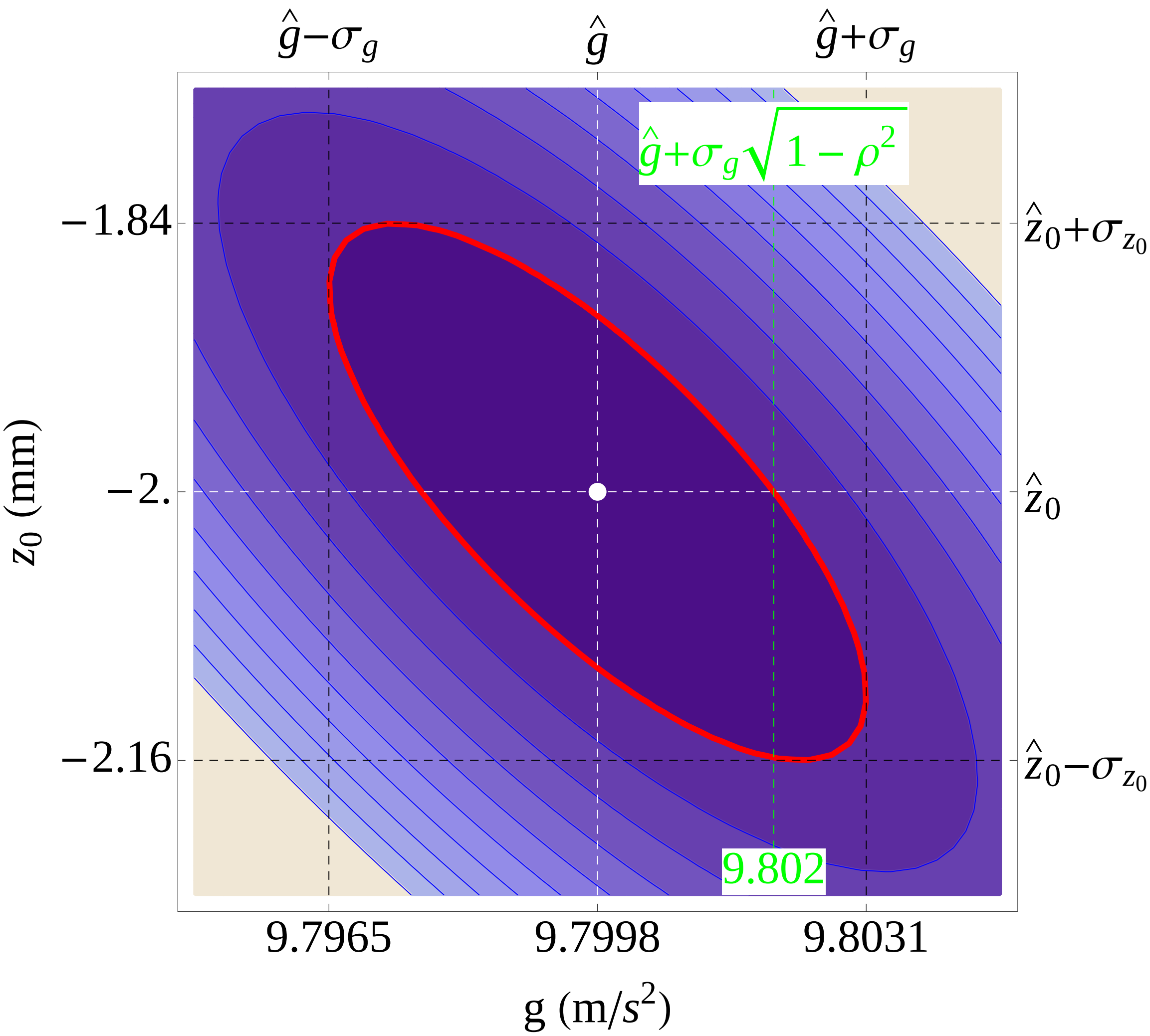
$\rho_{g,z_0} = -0.75$

Graphical determination of the standard error of g and z_0

We replot the mean square error function in order to illustrate the estimated values (white spot) and their uncertainties. The innermost (red) contour line is at $Q_{min}\left(1 + \frac{1}{n-2}\right)$. Its extent in g-direction indicated by the vertical black dashed lines gives the $\pm\sigma_g$ range around the estimate. Its extent in z_0-direction indicated by the horizontal black dashed lines gives the $\pm\sigma_{z0}$ range around the estimate of z_0. The green dashed line indicates the intersection of the red contour line with the white horizontal line. It has a distance $\sigma_g \sqrt{1-\rho^2}$ from the white vertical dashed line and can therefore serve for the graphical determination of ρ.

```
In[48]:= Show[ContourPlot[Q[g, z0, data], {g, gg - 3/2 σg, gg + 3/2 σg}, {z0, zz0 - 3/2 σz0, zz0 + 3/2 σz0},
   Frame → True, Axes → True, AxesOrigin → {gg - 3/2 σg, zz0 - 3/2 σz0},
   Contours → Table[Qmin (1 + k/(Length[data] - 2)), {k, 1, 10}], ContourStyle →
    {{Red, Thickness[0.007]}, Blue, Blue, Blue, Blue, Blue, Blue, Blue, Blue, Blue},
   FrameTicks → {{{gg, Round[gg, 0.0001]}, {gg + σg, Round[gg + σg, 0.0001]},
      {gg - σg, Round[gg - σg, 0.0001]}},
     {{zz0, Round[10^3 zz0, 0.01]}, {zz0 - σz0, Round[10^3 (zz0 - σz0), 0.01]},
      {zz0 + σz0, Round[10^3 (zz0 + σz0), 0.01]}}, {{gg, "ĝ"}, {gg + σg, "ĝ+σg"},
      {gg - σg, "ĝ-σg"}}, {{zz0, "ẑ0"}, {zz0 - σz0, "ẑ0-σz0"}, {zz0 + σz0, "ẑ0+σz0"}}},
   FrameLabel → {"g (m/s^2)", "z0 (mm)"}], Graphics[
   {{White, Dashed, Line[{{gg, zz0 - 3/2 σz0}, {gg, zz0 + 3/2 σz0}}],
     Line[{{gg - 3/2 σg, zz0}, {gg + 3/2 σg, zz0}}]},
    {White, Disk[{gg, zz0}, {σg / 30, σz0 / 30}]},
    {Black, Dashed, Line[{{gg + σg, zz0 - 3/2 σz0}, {gg + σg, zz0 + 3/2 σz0}}],
     Line[{{gg - σg, zz0 - 3/2 σz0}, {gg - σg, zz0 + 3/2 σz0}}],
     Line[{{gg - 3/2 σg, zz0 - σz0}, {gg + 3/2 σg, zz0 - σz0}}],
     Line[{{gg - 3/2 σg, zz0 + σz0}, {gg + 3/2 σg, zz0 + σz0}}]}, {Green, Dashed,
     Line[{{gg + σg Sqrt[1 - ρ^2], zz0 - 3/2 σz0}, {gg + σg Sqrt[1 - ρ^2], zz0 + 3/2 σz0}}]},
    {Green, Text[Style[Round[gg + σg Sqrt[1 - ρ^2], 0.0001], 24],
      {gg + σg Sqrt[1 - ρ^2], zz0 - 3/2 σz0}, {0, -1}, {1, 0}, Background → White]},
    {Green, Text[Style["ĝ+σg Sqrt[1 - ρ^2]", 24], {gg + σg Sqrt[1 - ρ^2], zz0 + 8/7 σz0},
      {0, -1}, {1, 0}, Background → White]}}],
  AxesLabel → {"g (m/s^2)", "z0 (m)"}, LabelStyle → 24,
  ImageSize → 600,
  Ticks → {{Round[gg - σg, 0.0001], Round[gg, 0.0001], Round[gg + σg, 0.0001]},
    {Round[zz0 - σz0, 0.00001], Round[zz0, 0.00001], Round[zz0 + σz0, 0.00001]}}]
```

Out[48]=

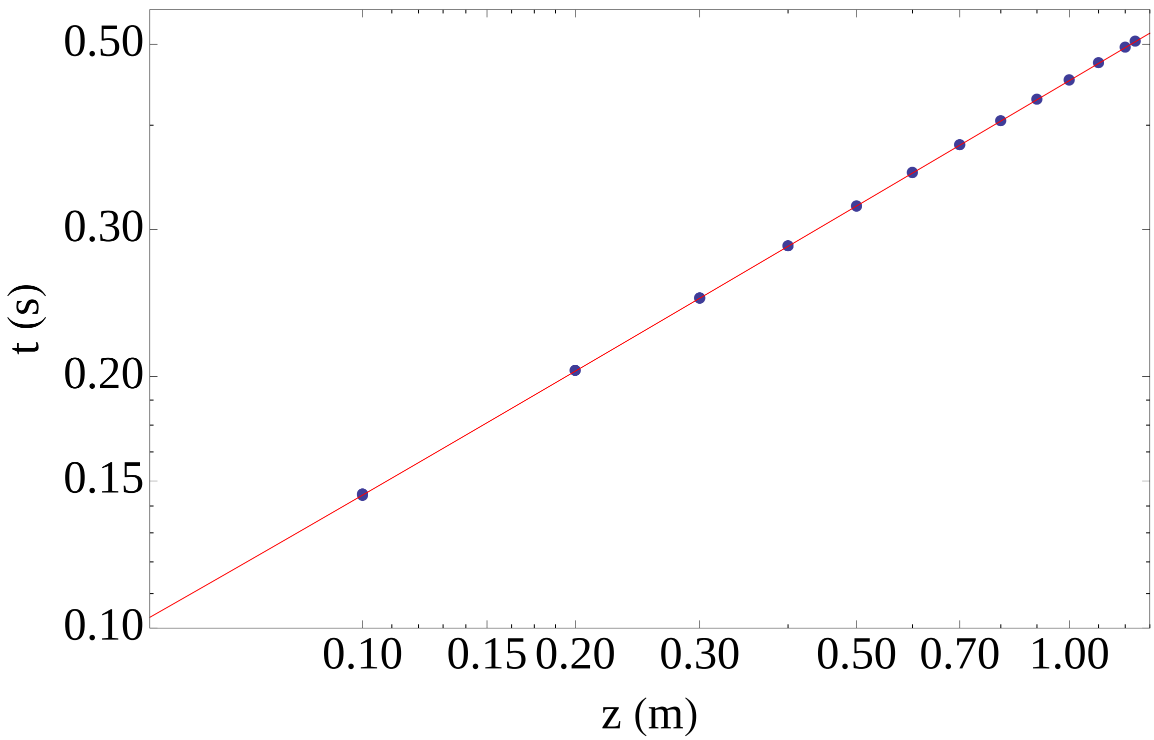


$\hat{g}-\sigma_g$
$\hat{g}$
$\hat{g}+\sigma_g$
$\hat{g}+\sigma_g\sqrt{1-\rho^2}$
−1.84
−2.
−2.16
$\hat{z}_0+\sigma_{z_0}$
$\hat{z}_0$
$\hat{z}_0-\sigma_{z_0}$
z_0 (mm)
9.802
9.7965
9.7998
9.8031
g (m/s^2)

Plotting the data with the fitted curve

In[49]:=
```
Show[g1, LogLogPlot[Sqrt[2 (z - zz0)/gg], {z, 0.05, 1.30}, PlotStyle → {Red}]]
```

Out[49]=



Inspecting the residuals of the fit

Again we inspect the fitting residuals:

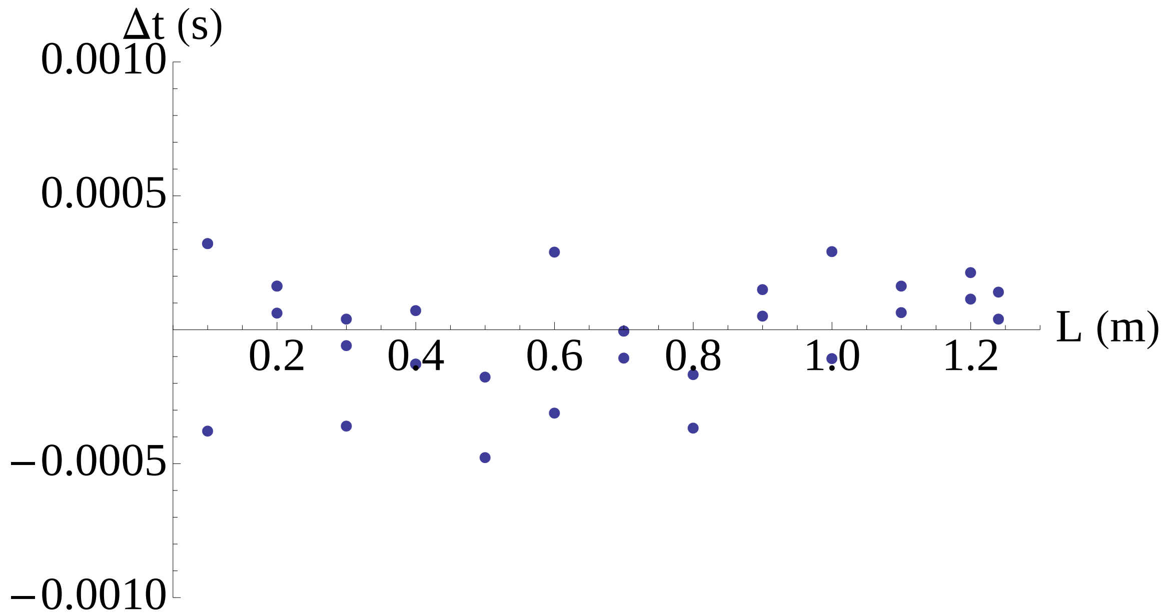
In[50]:=
```
residuals = Table[
  {data[[j, 1]], data[[j, 2]] - Sqrt[(2 (data[[j, 1]] - zz0))/gg]}, {j, 1, Length[data]}];
Grid[Prepend[residuals, {Style["z (m)", Bold], Style["r (s)", Bold]}],
 Background → {None, {Gray, {LightGray, White}}},
 Dividers → {Black, {Black, Black}}, Frame → True]
```

Out[51]=

z **(m)**	***r*** **(s)**
0.1	0.000320171
0.1	0.000320171
0.1	−0.000379829
0.2	0.000160041
0.2	0.0000600409
0.2	0.000160041
0.3	−0.000361517
0.3	0.0000384831
0.3	−0.0000615169
0.4	−0.000130335
0.4	0.0000696654
0.4	−0.000130335
0.5	−0.00017951
0.5	−0.00047951
0.6	−0.000313121
0.6	0.000286879
0.7	-7.60103×10^{-6}
0.7	-7.60103×10^{-6}
0.7	−0.000107601
0.8	−0.000169598
0.8	−0.000369598
0.9	0.000148586
0.9	0.0000485862
1.	−0.000109765
1.	0.000290235
1.1	0.000161424
1.1	0.0000614237
1.2	0.000211414
1.2	0.000111414
1.24	0.000137782
1.24	0.0000377815

In[52]:=
```
ListPlot[residuals, PlotRange → {{0.05, 1.30}, {-0.001, 0.001}},
 AxesLabel → {"L (m)", "Δt (s)"}, LabelStyle → 24,
 PlotMarkers → Automatic, ImageSize → 600, AxesOrigin → {0.05, 0}]
```

Out[52]=

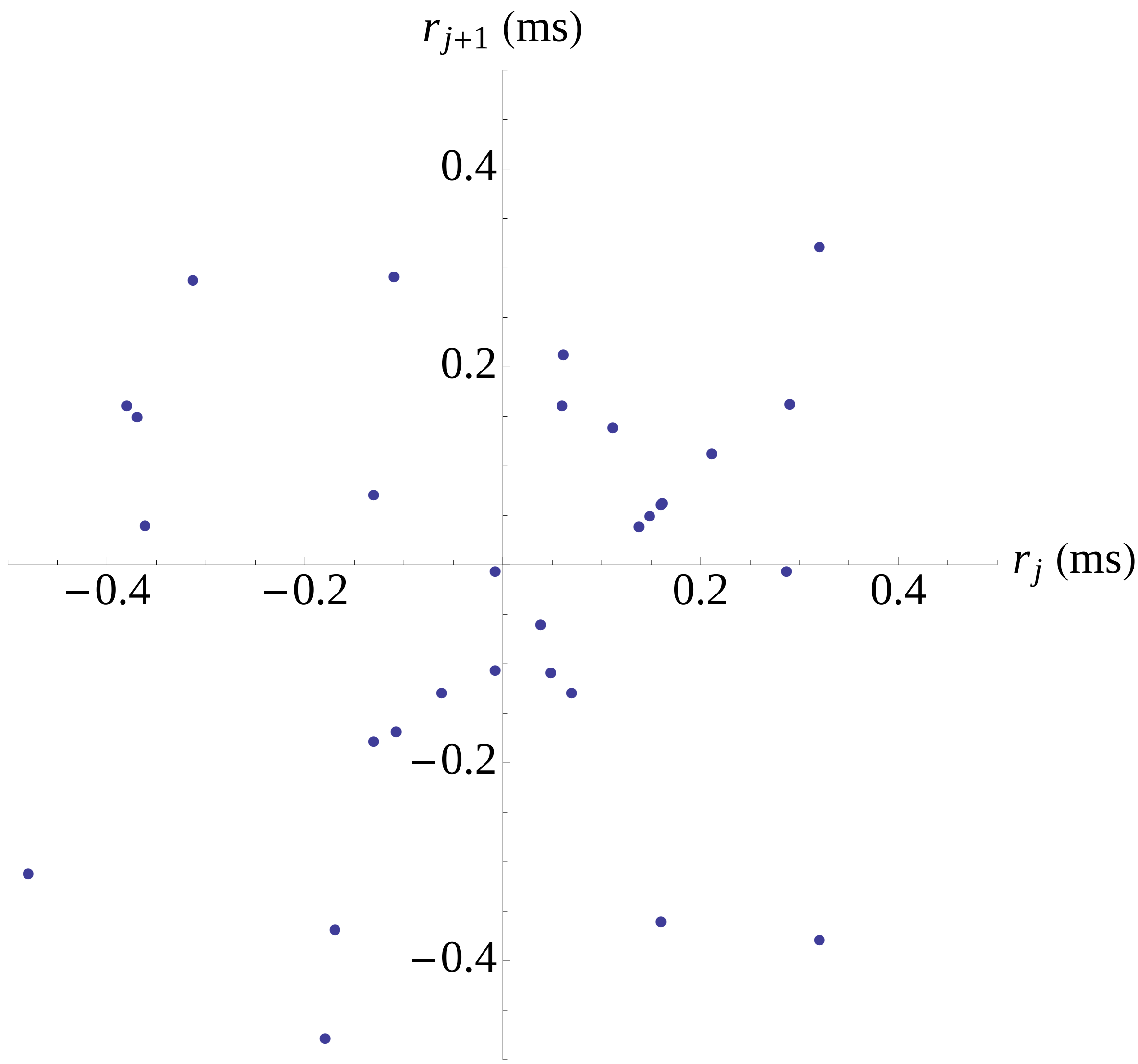


We see that the improved model has removed the trend in the residuals.

Advanced topic : Further investigation of the residuals with different techniques (beyond the lecture)

In[53]:=
```
ListPlot[Table[{10^3 residuals[[j, 2]], 10^3 residuals[[j + 1, 2]]},
   {j, 1, Length[data] - 1}], PlotRange → {{-0.5, 0.5}, {-0.5, 0.5}},
  AxesLabel → {"r_j (ms)", "r_j+1 (ms)"}, LabelStyle → 24, PlotMarkers → Automatic,
  ImageSize → 600, AxesOrigin → {0, 0}, AspectRatio → 1]
```

Out[53]=



The same histogram analysis done above leads to

In[54]:=
```
σest = √σest2 ;
hl = HistogramList[10^3 residuals[[Range[1, Length[data]], 2]],
    10^3 {-11 σest/4, 11 σest/4, σest/2}][[2]]
```

Out[55]= {0, 1, 4, 2, 5, 5, 9, 1, 4, 0, 0}

In[56]:=
```
ml = Table[{(j - 6) (10^3 σest)/2,
    Mean[BetaDistribution[hl[[j]] + 1, Length[data] - hl[[j]] + 1]]}, {j, 1, 11}];
Grid[Prepend[ml, {Style["bin center (ms)", Bold], Style["⟨p⟩", Bold]}],
 Background → {None, {Gray, {LightGray, White}}},
 Dividers → {Black, {Black, Black}}, Frame → True]
```

Out[57]=

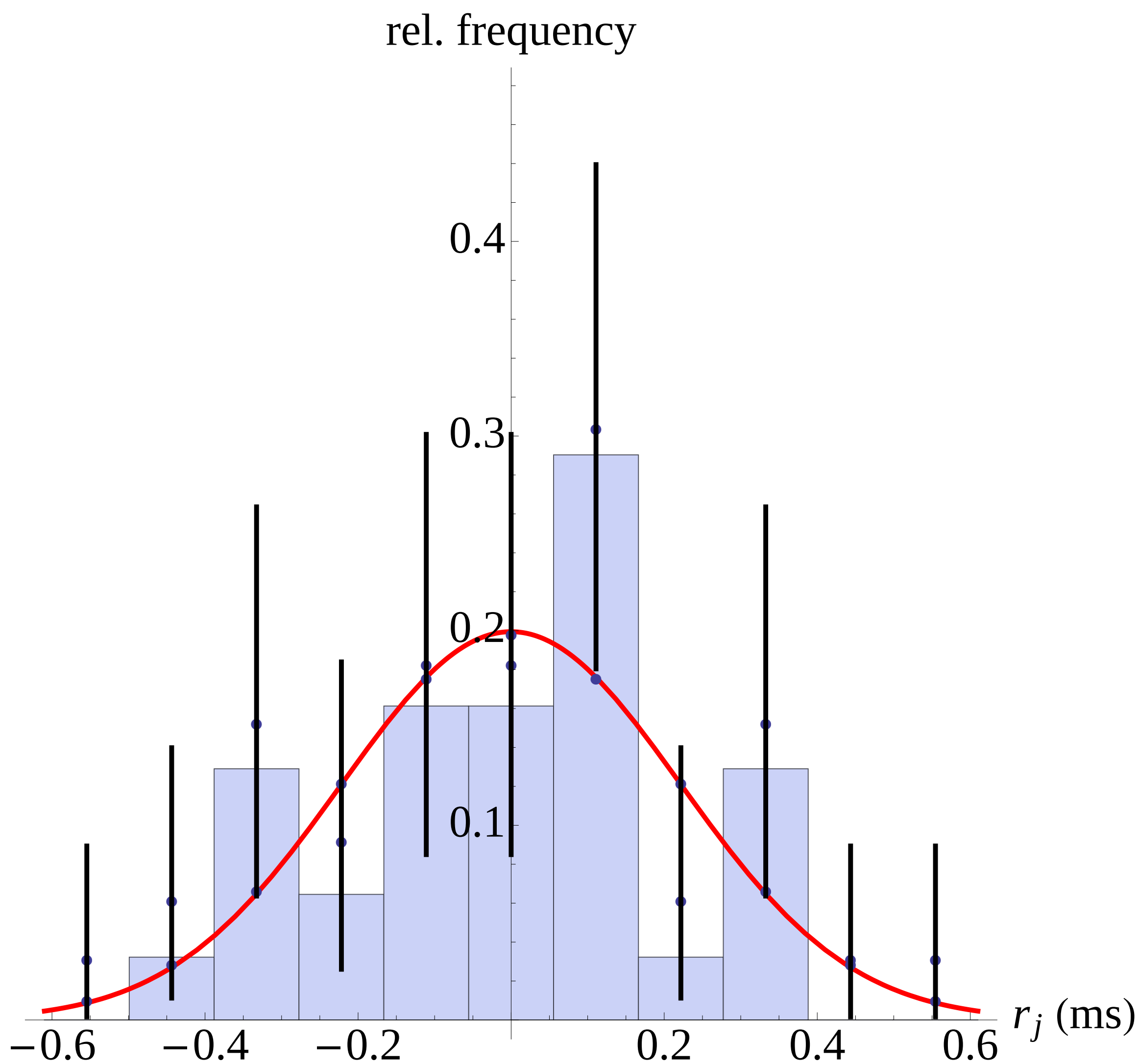
bin center (ms)	**⟨p⟩**
-0.554476	$\frac{1}{33}$
-0.443581	$\frac{2}{33}$
-0.332686	$\frac{5}{33}$
-0.22179	$\frac{1}{11}$
-0.110895	$\frac{2}{11}$
0.	$\frac{2}{11}$
0.110895	$\frac{10}{33}$
0.22179	$\frac{2}{33}$
0.332686	$\frac{5}{33}$
0.443581	$\frac{1}{33}$
0.554476	$\frac{1}{33}$

In[58]:=
```
eb = Table[Line[{{(j - 6) (10^3 σest)/2,
      N[Quantile[BetaDistribution[hl[[j]] + 1, Length[data] - hl[[j]] + 1], 0.95]]},
     {(j - 6) (10^3 σest)/2, N[Quantile[BetaDistribution[hl[[j]] + 1,
          Length[data] - hl[[j]] + 1], 0.05]]}}], {j, 1, 11}]
```

Out[58]=
```
{Line[{{-0.554476, 0.0893682}, {-0.554476, 0.00160163}}],
 Line[{{-0.443581, 0.139849}, {-0.443581, 0.0112195}}],
 Line[{{-0.332686, 0.263597}, {-0.332686, 0.0636527}}],
 Line[{{-0.22179, 0.183943}, {-0.22179, 0.026043}}],
 Line[{{-0.110895, 0.300842}, {-0.110895, 0.0849545}}],
 Line[{{0., 0.300842}, {0., 0.0849545}}],
 Line[{{0.110895, 0.439447}, {0.110895, 0.180385}}],
 Line[{{0.22179, 0.139849}, {0.22179, 0.0112195}}],
 Line[{{0.332686, 0.263597}, {0.332686, 0.0636527}}],
 Line[{{0.443581, 0.0893682}, {0.443581, 0.00160163}}],
 Line[{{0.554476, 0.0893682}, {0.554476, 0.00160163}}]}
```

In[59]:=
```
Show[Histogram[10^3 residuals[[Range[1, Length[data]], 2]],
   10^3 {-11 σest/4, 11 σest/4, σest/2}, "Probability", AxesOrigin → {0, 0},
   LabelStyle → 24, AxesLabel → {"r_j  (ms)", "rel. frequency"}, ImageSize → 600],
  ListPlot[ml, PlotMarkers → Automatic],
  Plot[10^3 σest/2 PDF[NormalDistribution[0, 10^3 σest], x],
   {x, -11/4 10^3 σest, 11/4 10^3 σest}, PlotStyle → {Red, Thickness[0.005]}],
  ListPlot[Table[{k 10^3 σest/2, (CDF[NormalDistribution[0, σest], σest/4 + k σest/2] -
        CDF[NormalDistribution[0, σest], -σest/4 + k σest/2])},
    {k, -5, 5, 1}], LabelStyle → 24, PlotMarkers → Automatic],
  Graphics[{Thickness[0.005], eb}], AspectRatio → 1]
```

Out[59]=



The red curve runs reasonably within the error bars of the probability estimate, confirming qualitatively that the assumption of a normal distribution for the scatter of the time measurement is reasonable. The exceptional bin encountered in the analysis with the previous model has disappeared. However, there is nothing in this comparison that would tell us under which circumstances we better reject the hypothesis of a normal distribution. The situation could be made to look much worse, if we were to plot error bars spanning only the range between the 30% and the 70% quantile.

Model I, a shortcut offered by Mathematica

The command NonlinearModelFit performs most of the tasks that we did above:

In[60]:=

```
nlm = NonlinearModelFit[data, Sqrt[2 z / g], {g}, z];
```

The functional form of the fitted model is:

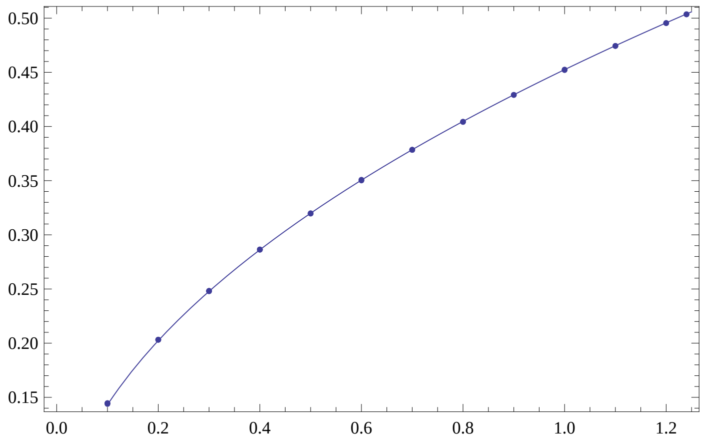
In[61]:=
```
Normal[nlm]
```

Out[61]= $0.452465\sqrt{z}$

We can plot the data together with the result of the fit:

In[62]:=
```
Show[ListPlot[data], Plot[nlm[z], {z, 0.1, 1.25}], Frame → True]
```

Out[62]=



The fitting parameters and their standard errors are also available:

In[63]:=
```
nlm["ParameterTable"]
```

Out[63]=

	Estimate	Standard Error	t-Statistic	P-Value
g	9.76921	0.00531386	1838.44	2.41573×10^{-77}

The list of fit residuals can be extracted...

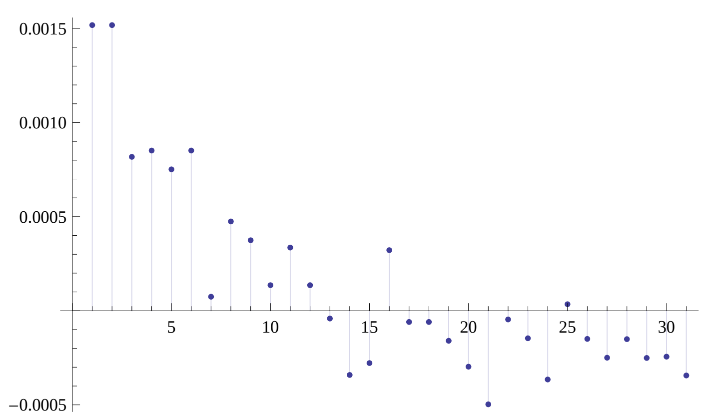
In[64]:=
```
nlm["FitResiduals"]
```

Out[64]=
```
{0.00151793, 0.00151793, 0.00081793, 0.000851396, 0.000751396, 0.000851396,
 0.0000745852, 0.000474585, 0.000374585, 0.00013586, 0.00033586,
 0.00013586, -0.0000412347, -0.000341235, -0.000278063, 0.000321937,
 -0.0000595741, -0.0000595741, -0.000159574, -0.000297208, -0.000497208,
 -0.0000462098, -0.00014621, -0.000365233, 0.0000347667, -0.00014954,
 -0.00024954, -0.00015083, -0.00025083, -0.00024396, -0.00034396}
```

... and plotted.

In[65]:=
```
ListPlot[nlm["FitResiduals"], Filling → Axis]
```

Out[65]=



Model II, using the shortcut offered by Mathematica

We use a constraint for the offset making sure that the values under the square root are certainly positive. Furthermore, we use the constraint that g is positive.

In[66]:=
```
nlm =
  NonlinearModelFit[data, {Sqrt[2 (z - offs)/g], {-0.1 < offs < 0.1, g > 0}}, {g, offs}, z];
```

$$\texttt{nlm} = \texttt{NonlinearModelFit}\left[\texttt{data}, \left\{\sqrt{\frac{2\,(z-\text{offs})}{g}}, \{-0.1<\text{offs}<0.1,\ g>0\}\right\}, \{g, \text{offs}\}, z\right];$$

The functional form of the fitted model is:

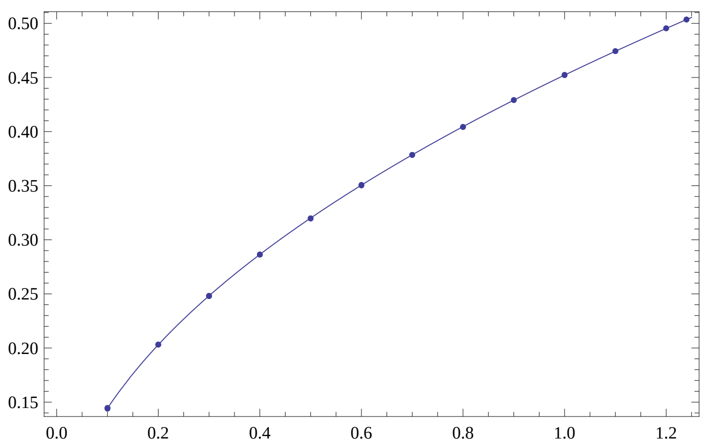
In[67]:=
```
Normal[nlm]
```

Out[67]= $0.451764\,\sqrt{0.00199476 + z}$

We can plot the data together with the result of the fit:

In[68]:=
```
Show[ListPlot[data], Plot[nlm[z], {z, 0.1, 1.25}], Frame → True]
```

Out[68]=



The fitting parameters and their standard errors are also available:

In[69]:=
```
nlm["ParameterTable"]
```

FittedModel::constr :
The property values {ParameterTable} assume an unconstrained model. The results for these properties may not be valid, particularly if the fitted parameters are near a constraint boundary. ≫

Out[69]=

	Estimate	Standard Error	t-Statistic	P-Value
g	9.79958	0.00329276	2976.1	4.32471×10^{-81}
offs	−0.00199476	0.000162307	−12.29	5.0609×10^{-13}

The list of fit residuals can be extracted...

In[70]:= **nlm["FitResiduals"]**

Out[70]= {0.000321987, 0.000321987, -0.000378013, 0.000160152, 0.0000601519,
0.000160152, -0.000362391, 0.0000376091, -0.0000623909, -0.000131928,
0.0000680717, -0.000131928, -0.000181684, -0.000481684, -0.00031579,
0.00028421, -0.0000107054, -0.0000107054, -0.000110705, -0.000173095,
-0.000373095, 0.000144731, 0.0000447311, -0.000113952, 0.000286048,
0.000156926, 0.0000569258, 0.000206624, 0.000106624, 0.000132879, 0.0000328788}

... and plotted.

In[71]:= **ListPlot[nlm["FitResiduals"], Filling → Axis]**

Out[71]=