THE DERIVE - NEWSLETTER #79

ISSN 1990-7079

THE BULLETIN OF THE

DERIVE®

USER GROUP

+ CAS-TI

Contents:

September 2010

Dear Josef and Michel,

Just a little precision. I noticed that the link in the Newsletter doesn't point toward my website !

> «Find a collection of animations from Canada (Genevieve Savard, Montréal)
> http://www.seg.etsmtl.ca/Math/Animations/index.html »

This website was created by Robert Michaud.

My website is http://www.seg.etsmtl.ca/GSavard/index.html and the animation page is http://www.seg.etsmtl.ca/GSavard/Animations/index.html

I wish you a very nice day,

Geneviève

Geneviève Savard
Maître d'enseignement en mathématiques
École de technologie supérieure

Interesting and recommended websites:

SeeLogo (APGS) is a computer language through which the user can create beautiful pictures, dynamic arts and make games. Here is a link to a book that uses the language to create mathematical art.

http: www.ithaca.edu/seelogo/

You can find and download another **free CAS-program CoCo**A (in many languages).

- CoCoA is a program to compute with numbers and polynomials.
- It is free.
- It works on many operating systems.
- It is used by many researchers, but can be useful even for "simple" computations.

http://cocoa.dima.unige.it/

Visual Interactive Tools for Advanced Learning:

http://www.mathe-vital.de

Bei MatheVital handelt es sich um eine modulare, frei zugängliche Sammlung interaktiver Materialien für den Unterricht in mathematiknahen Fächern.

Interoperable Interactive Geometry for Europe (many languages)
A new platform for Dynamic Geometry programs:

http://i2geo.net

Find more Links on page 3

Dear DUG Members,

As I promised in the last DNL I'll give a report of the official DUG Meeting which was held at TIME 2010 in Málaga. Every four years the DUG-board must be elected. There are no changes in the board, all members accepted staying in our commission: Bärbel Barzel, Josef & Noor Böhm, Walter Klinger, Bernhard Kutzler and Josef Lechner (in alphabetical order). Thanks to all of you for your work in the past and much success for the next 4 years' period.

We had many excellent talks and workshops in both Conference Strands. Unfortunately I could not attend so many of them because of giving my own lecture(s) and workshop or being occupied as chair of other sessions. So I am looking forward to browsing and studying the Conference Proceedings which should be ready soon. I´ll keep you informed. (If you want to have a look to it in advance, you can go to http://www.time2010.uma.es/abtracts.pdf.)

We had great keynotes. The picture shows Michel Beaudin talking about "Using the Real Power of Computer Algebra". I feel reminded on Hamlet with the ghost of Hamlet´s father in the background. (It is our friend Terence Etchells who could not participate, so he appeared as a good ghost in during Michel´s talk.)

We all are indebted to the generous sponsors of the Conference: University of Málaga and some faculties, Authorities of Málaga and Antequera, Texas Instruments, Unicaja. Many thanks to you all.

In this DNL you will not find so many articles as usual. The contributions of DNL#79 are very extended. Don Phillips provides a tool for Nonlinear Regression and 2-Stage Least Squares Regression and demonstrates in an impressive way that it is possible to transfer programs from DERIVE to the TI89/92/V200 and to TI-NspireCAS as well.

My article on the Huffman-Code makes use of JohannWiesenbauer´s tool from the last DNL for plotting binary trees.

Please pay attention to the many links to excellent websites given in the Information page and on page 3. Thanks again to Michael de Villiers from South Africa for his valuable notes.

It is a nice cooccurrence that Michel Beaudin wrote about cubics in the last DNL and we have another in this DNL request on the same issue. Btw there is an interesting paper on Cardano´s formula in The Montana Mathematics Enthusiast, 2005, vol. 2. You can download it, see the links.

Best regards as ever,

Download all *DNL*-DERIVE- and TI-files from
http://www.austromath.at/dug/

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE* & CAS-*TI User Group*. It is published at least four times a year with a contents of 40 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE*, *TI*-CAS and other CAS as well to create a group to discuss the possibilities of new methodical and didactical manners in teaching mathematics.

Editor: Mag. Josef Böhm
D´Lust 1, A-3042 Würmla
Austria
Phone: ++43-06604070480
e-mail: nojo.boehm@pgv.at

Contributions:
Please send all contributions to the Editor. Non-English speakers are encouraged to write their contributions in English to reinforce the international touch of the *DNL*. It must be said, though, that non-English articles will be warmly welcomed nonetheless. Your contributions will be edited but not assessed. By submitting articles the author gives his consent for reprinting it in the *DNL*. The more contributions you will send, the more lively and richer in contents the *DERIVE* & CAS-*TI Newsletter* will be.

Next issue: December 2010
Deadline 15 December 2010

Preview: Contributions waiting to be published

Some simulations of Random Experiments, J. Böhm, AUT, Lorenz Kopp, GER
Wonderful World of Pedal Curves, J. Böhm
Tools for 3D-Problems, P. Lüke-Rosendahl, GER
Financial Mathematics 4, M. R. Phillips
Hill-Encription, J. Böhm
Simulating a Graphing Calculator in *DERIVE*, J. Böhm
Henon & Co, J. Böhm
Do you know this? Cabri & CAS on PC and Handheld, W. Wegscheider, AUT
An Interesting Problem with a Triangle, Steiner Point, P. Lüke-Rosendahl, GER
Overcoming Branch & Bound by Simulation, J. Böhm, AUT
Diophantine Polynomials, D. E. McDougall, Canada
Graphics World, Currency Change, P. Charland, CAN
Cubics, Quartics – Interesting features, T. Koller & J. Böhm
Logos of Companies as an Inspiration for Math Teaching
Exciting Surfaces in the FAZ / Pierre Charland´s Graphics Gallery
BooleanPlots.mth, P. Schofield, UK
Old traditional examples for a CAS – what´s new? J. Böhm, AUT
Truth Tables on the TI, M. R. Phillips
Where oh Where is It? (GPS with CAS), C. & P. Leinbach, USA
Embroidery Patterns, H. Ludwig, GER
Mandelbrot and Newton with *DERIVE*, Roman Hašek, CZ
Snail-shells, Piotr Trebisz, GER
A Conics-Explorer, J. Böhm, AUT
Practise Working with Times
Tutorials for the NSpireCAS, G. Herweyers, BEL
Some Projects with Students, R. Schröder, GER
Dirac Algebra, Clifford Algebra, D. R. Lunsford, USA
The PROOF, C. Leinbach & J. Böhm
Treating Differential Equations (M. Beaudin, G. Piccard, Ch. Trottier)

and others

Impressum:
Medieninhaber: *DERIVE* User Group, A-3042 Würmla, D´Lust 1, AUSTRIA
Richtung: Fachzeitschrift
Herausgeber: Mag.Josef Böhm

More Links which might be of interest for you:
(Thanks to Michael de Villiers, who provided valueable links!)

The **Association for Mathematics Education of South Africa** Congress 2010 Proceedings

http://www.amesa.org.za/AMESA2010/Proceedings.htm

Volume 1: Lectures (337 pages), Volume 2: Workshops & How I Teach (330 pages)

Download free textbooks from bookboon:

Do you need math help? In our free mathematics books you hopefully will find answers to your questions. These textbooks will guide you through mathematical concepts and models, and hopefully give you a better understanding. Topics such as limit value, linear optimization and the decay constant are explained. Specifically for computer science students, we provide the book 'Mathematics for Computer Scientists'. (Announcement from bookboon)

http://bookboon.com/uk/student/mathematics

Examples: Applied Mathematics
Calculus
Complex Functions
Systems of Differential Equations
Probability for Finance a.o.

The July 2010 issue of **The Montana Mathematics Enthusiast** can be downloaded for free:

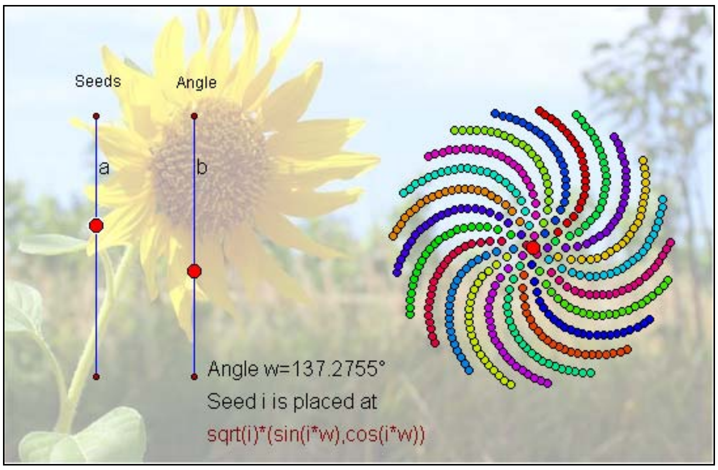
http://www.math.umt.edu/TMME/vol7no2and3/index.html
http://www.math.umt.edu/TMME/vol7no1/

This is the URL of the **Montana Council of Teachers of Mathematics**

http://www.montanamath.org/

You can download **CINDERELLA 1.4** for free from

http://www.cinderella.de/tiki-index.php



<u>**J.L., Austria**</u>

Lieber Josef,
ich möchte Dir zwei Files zur Formel von Cardano schicken.
Ich versuche gerade die Gleichung $x^3+3x^2+9x+9=0$ mit der F.v.C. zu lösen.

Das funktioniert mit TI-NSpire 2.1 problemlos, bei Derive kriege ich aber nicht das richtige Ergebnis (siehe #7). Irgendetwas läuft da nicht korrekt, entweder habe ich irgend einen trivialen Fehler gemacht oder eine Voreinstellung ist ungünstig oder ich kann mit Wurzeln nicht richtig umgehen oder ...

Dear Josef,
I am sending two files wrt Cardano´s formula. I am trying solving $x^3+3x^2+9x+9=0$ applying this formula. This is no problem with TI-Nspire, but I am not able to obtain the correct result with DERIVE, maybe that I made a trivial mistake, a typo, a wrong setting …

$$x^3+p\cdot x^2+q\cdot x+r=0 \qquad r+x^3+p\cdot x^2+q\cdot x=0$$

$$p:=3:q:=9:r:=9 \qquad 9$$

$$\text{approx}\left(\text{cSolve}\left(x^3+3\cdot x^2+9\cdot x+9=0,x\right)\right) \qquad x=-0.83626+2.46585\cdot i \text{ or } x=-0.83626-2.46585\cdot i \text{ or } x=-1.32748$$

$$u:=\left(\frac{-\left(2\cdot p^3-9\cdot p\cdot q+27\cdot r\right)}{54}+\left(\frac{\left(2\cdot p^3-9\cdot p\cdot q+27\cdot r\right)^2}{2916}+\frac{\left(3\cdot q-p^2\right)^3}{729}\right)^{\frac{1}{2}}\right)^{\frac{1}{3}} \qquad 2^{\frac{1}{3}}$$

$$v:=\left(\frac{-\left(2\cdot p^3-9\cdot p\cdot q+27\cdot r\right)}{54}-\left(\frac{\left(2\cdot p^3-9\cdot p\cdot q+27\cdot r\right)^2}{2916}+\frac{\left(3\cdot q-p^2\right)^3}{729}\right)^{\frac{1}{2}}\right)^{\frac{1}{3}} \qquad -2^{\frac{2}{3}}$$

$$x1:=u+v-\frac{p}{3} \qquad -2^{\frac{2}{3}}+2^{\frac{1}{3}}-1$$

$$\text{approx}\left(-2^{\frac{2}{3}}+2^{\frac{1}{3}}-1\right) \qquad -1.32748$$

$$x2:=\frac{-(u+v)}{2}+\frac{u-v}{2}\cdot\sqrt{3}\cdot i-\frac{p}{3} \qquad \frac{2^{\frac{2}{3}}}{2}-\frac{2^{\frac{1}{3}}}{2}-1+\frac{\left(2^{\frac{1}{3}}+1\right)\cdot\sqrt{3}\cdot 2^{\frac{1}{3}}}{2}\cdot i$$

11/99

$$v:=\left(\frac{-\left(2\cdot p^3-9\cdot p\cdot q+27\cdot r\right)}{54}-\left(\frac{\left(2\cdot p^3-9\cdot p\cdot q+27\cdot r\right)^2}{2916}+\frac{\left(3\cdot q-p^2\right)^3}{729}\right)^{\frac{1}{2}}\right)^{\frac{1}{3}} \qquad -2^{\frac{2}{3}}$$

$$x1:=u+v-\frac{p}{3} \qquad -2^{\frac{2}{3}}+2^{\frac{1}{3}}-1$$

$$\text{approx}\left(-2^{\frac{2}{3}}+2^{\frac{1}{3}}-1\right) \qquad -1.32748$$

$$x2:=\frac{-(u+v)}{2}+\frac{u-v}{2}\cdot\sqrt{3}\cdot i-\frac{p}{3} \qquad \frac{2^{\frac{2}{3}}}{2}-\frac{2^{\frac{1}{3}}}{2}-1+\frac{\left(2^{\frac{1}{3}}+1\right)\cdot\sqrt{3}\cdot 2^{\frac{1}{3}}}{2}\cdot i$$

$$\text{approx}\left(\frac{2^{\frac{2}{3}}}{2}-\frac{2^{\frac{1}{3}}}{2}-1+\frac{\left(2^{\frac{1}{3}}+1\right)\cdot\sqrt{3}\cdot 2^{\frac{1}{3}}}{2}\cdot i\right) \qquad -0.83626+2.46585\cdot i$$

$$x2:=\frac{-(u+v)}{2}-\frac{u-v}{2}\cdot\sqrt{3}\cdot i-\frac{p}{3} \qquad \frac{2^{\frac{2}{3}}}{2}-\frac{2^{\frac{1}{3}}}{2}-1-\frac{\left(2^{\frac{1}{3}}+1\right)\cdot\sqrt{3}\cdot 2^{\frac{1}{3}}}{2}\cdot i$$

$$\text{approx}\left(\frac{2^{\frac{2}{3}}}{2}-\frac{2^{\frac{1}{3}}}{2}-1-\frac{\left(2^{\frac{1}{3}}+1\right)\cdot\sqrt{3}\cdot 2^{\frac{1}{3}}}{2}\cdot i\right) \qquad -0.83626-2.46585\cdot i$$

11/99

The DERIVE file:

#1: $\text{SOLVE}(x^3 + 3\cdot x^2 + 9\cdot x + 9 = 0, x)$

#2: $x = \frac{2^{2/3}}{2} - \frac{2^{1/3}}{2} - 1 - i\cdot\left(\frac{\sqrt{3}\cdot 2^{2/3}}{2} + \frac{\sqrt{3}\cdot 2^{1/3}}{2}\right) \vee x = \frac{2^{2/3}}{2} - \frac{2^{1/3}}{2} - 1 + i\cdot\left(\frac{\sqrt{3}\cdot 2^{2/3}}{2} + \frac{\sqrt{3}\cdot 2^{1/3}}{2}\right) \vee x = -2^{2/3} + 2^{1/3} - 1$

#3: $x = -0.8362599989 - 2.465853272\cdot i \vee x = -0.8362599989 + 2.465853272\cdot i \vee x = -1.327480002$

#4: $[p := 3, q := 9, r := 9]$

#5: $u := \left(-\frac{2\cdot p^3 - 9\cdot p\cdot q + 27\cdot r}{54} + \left(\frac{(2\cdot p^3 - 9\cdot p\cdot q + 27\cdot r)^2}{2916} + \frac{(3\cdot q - p^2)^3}{729}\right)^{1/2}\right)^{1/3}$

#6: $u := 2^{1/3}$

#7: $v := \left(-\frac{2\cdot p^3 - 9\cdot p\cdot q + 27\cdot r}{54} - \left(\frac{(2\cdot p^3 - 9\cdot p\cdot q + 27\cdot r)^2}{2916} + \frac{(3\cdot q - p^2)^3}{729}\right)^{1/2}\right)^{1/3}$

#8: $v := \frac{2^{2/3}}{2} + \frac{\sqrt{3}\cdot 2^{2/3}\cdot i}{2}$

#9: $\left[x1 := u + v - \frac{p}{3}, x2 := -\frac{u + v}{2} - \frac{p}{3} + \frac{u - v}{2}\cdot(-3)^{1/2}, x3 := -\frac{u + v}{2} - \frac{p}{3} - \frac{u - v}{2}\cdot(-3)^{1/2}\right]$

#10: $[x1 := 1.053621575 + 1.374729636\cdot i, x2 := -0.8362599989 - 0.2836060010\cdot i, x3 := -3.217361576 - 1.091123635\cdot i]$

Compare #3 and #10!

DNL: Dear Josef, try this:

Set **Branch:=Real**, then it works!!

#11: Branch := Real

#12: $u := \left(-\frac{2\cdot p^3 - 9\cdot p\cdot q + 27\cdot r}{54} + \left(\frac{(2\cdot p^3 - 9\cdot p\cdot q + 27\cdot r)^2}{2916} + \frac{(3\cdot q - p^2)^3}{729}\right)^{1/2}\right)^{1/3}$

#13: $u := 2^{1/3}$

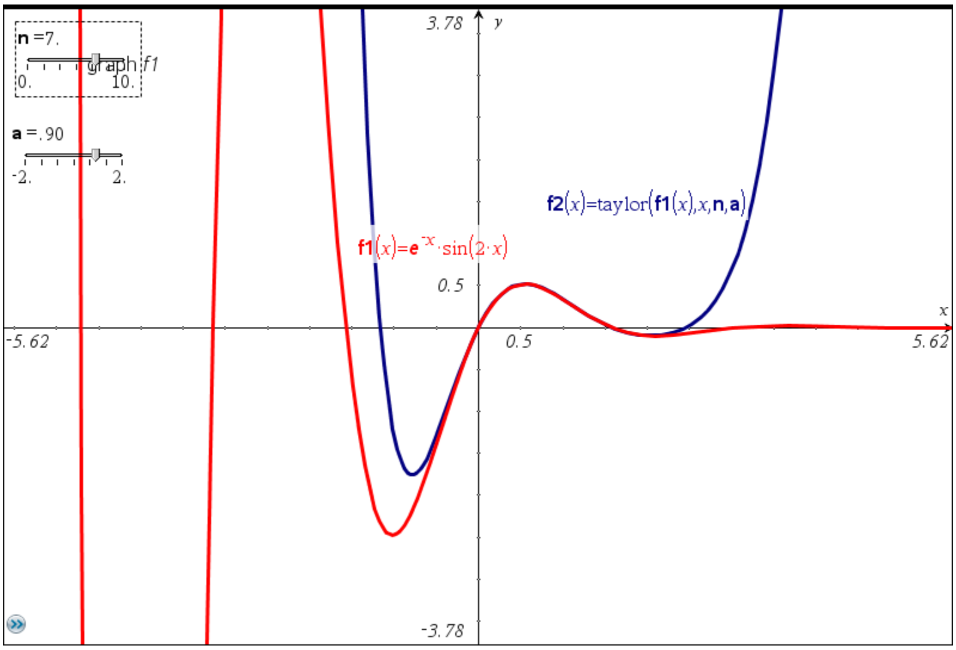
#14: $v := \left(-\frac{2\cdot p^3 - 9\cdot p\cdot q + 27\cdot r}{54} - \left(\frac{(2\cdot p^3 - 9\cdot p\cdot q + 27\cdot r)^2}{2916} + \frac{(3\cdot q - p^2)^3}{729}\right)^{1/2}\right)^{1/3}$

#15: $v := -2^{2/3}$

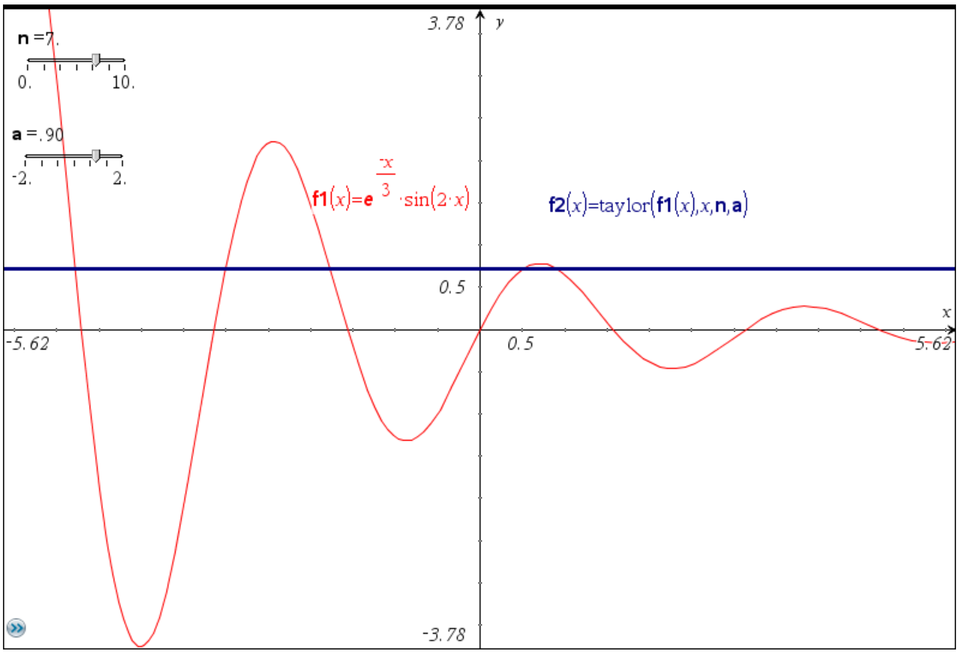
#16: $\left[x1 := u + v - \frac{p}{3}, x2 := -\frac{u + v}{2} - \frac{p}{3} + \frac{u - v}{2}\cdot(-3)^{1/2}, x3 := -\frac{u + v}{2} - \frac{p}{3} - \frac{u - v}{2}\cdot(-3)^{1/2}\right]$

#17: $[x1 := -1.327480002, x2 := -0.8362599989 + 2.465853272\cdot i, x3 := -0.8362599989 - 2.465853272\cdot i]$

When I prepared my talk for TIME 2010 about the use and didactical value of sliders in mathemtics education I came across a strange behaviour of TI-NspireCAS.
I wanted to demonstrate the Taylor approximation supported by two sliders – one for the location of the Taylor expansion (= a) and another one for the order of the Taylor polynomial (= n).



As you can see in the screen shot above this works for function $e^{-x}\cdot\sin(2x)$.



Then I tried $e^{-\frac{x}{3}}\cdot\sin(2x)$ and I failed. I wrote to TI (Gosia Brothers) and she answered:

We know that this is a problem with our series code dealing with floats. You need only put approx() around taylor() to see this problem in Calculator or Notes. David Stoutemyer developed this code and he is currently working on updating it. Not sure when the fix will be in but I will let you know as soon as I know.

Thnaks to Gosia for the immediate reply. The bug has not been resolved in TI-NspireCAS 2.0.

Zipped today? – The Huffman-Code

Josef Böhm, Würmla, Austria

When you want to send big amounts of data via email, then you will probably compress the file(s). Some programs do the job. One of these programs gave the name: "zipping". (Which one?) Compressed files show mostly the file extension zip or rar. Especially graphic files (photographs, scanned images, …, and dfw-files, of course) may become very large. It is easy to reach a couple of megabytes. Intelligent algorithms provide a compression to a significant smaller amount of data without loss of data.

Name	Typ	Datum	Größe	Komp...	Kompri...	Pl
Verkehr.dfw	Derive Worksheet	13.10.2004 15:37	7 220 411	99%	91 381	

The picture shows the result of „zipping" or „packing" a Derive-file which contains some graphs. You can see the efficiency of the compression algorithm.

A very simple method is the following: in a graph appears a sequence of 3878 white image points (pixels) followed by a sequence of 132 black pixels. Instead of listing w, w, ..., w, b, b, ..., b one can note much shorter: w, 3878, b, 132, loosing no information at all. The next paragraph is some text (in German) which shall be used for demonstrating compression for transmission.

> "bei der komprimierung von texten laesst man sich von der unterschiedlichen haeufigkeit der zeichen in dem zu codierenden text leiten. wir wollen das an einem einfachen beispiel demonstrieren, wobei wir nur kleinbuchstaben, zwischenraeume und satzzeichen verwenden wollen. dieser text wird verwendet."

Characters with a high frequency will be assigned to short code words in order to save bits! This basic idea is also realized in the Morse-Code. Character "e" is encoded by a code word of length 1, the "·", character "q" which is pretty rare by a code word of length 4, "– – · –".

We create a „binary tree". This is a directed graph consisting of vertices or nodes and edges (arrows) (= branches of the tree). Our first message "demo" to be encoded is "dieser text wird verwendet". Using the ASCII-Code without check bit the length of the message is 189 bit.

First of all we find out the frequency of the characters by simply counting. This can be done manually. For extended texts we will use the computer.

#1: $\text{h(list, number)} := \sum_{i=1}^{\text{DIM(list)}} \text{IF}(\text{list}_i = \text{number})$

#2: $\text{freq(plain)} := \text{SORT(SELECT}(v_1 \ne 0,\ v,\ \text{VECTOR([h(NAME\_TO\_CODES(plain), k), CODES\_TO\_NAME(k)], k, 32, 127)))'}$

#3: demo := dieser text wird verwendet.

#4: DIM(demo) = 27

#5: $\text{freq(demo)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 3 & 6 \\ . & n & s & v & x & i & w & \ & d & r & t & e \end{bmatrix}$

Expression #8 (next page) shows the frequency table for the paragraph from above starting with: "bei der komprimierung … ". (Don't forget to write the text under quotes. The quotes are not visible on the DERIVE screen, they are visible in the Edit-Line.)

```
#6:   paragraph := bei der komprimierung von texten laesst man sich von der unterschiedlichen
        haeufigkeit der zeichen in dem zu codierenden text leiten. wir wollen das an einem
        einfachen beispiel demonstrieren, wobei wir nur kleinbuchstaben, zwischenraeume und
        satzzeichen verwenden wollen. dieser text wird verwendet.

#7:   freq(paragraph)
```

#8:

$$\begin{bmatrix} 2 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 5 & 5 & 7 & 8 & 8 & 9 & 9 & 9 & 9 & 9 & 11 & 14 & 14 & 17 & 25 & 29 & 40 & 50 \\ , & f & g & p & . & k & x & v & b & z & m & o & u & a & c & h & l & w & s & d & t & r & i & n & \ & e \end{bmatrix}$$

The procedure is performed as follows:

Generate a node for every character appearing in the message. Label all nodes with their weights (= frequencies).
Until there is only one node remaining with no arrow directed to it, do:

Connect two nodes with minimal weights which are not end points of an arrow by a new node. The weight of this "parent node" is the sum of the weights of the "children".
    The arrows are directed from the parent node to the children nodes.
    The arrows are named as 0 (the left edge) and 1 (the right edge) by convention.

On the left you can find the structogram for the algorithm. We will follow the instructions and create the "Huffman-Tree" for the code of our message.

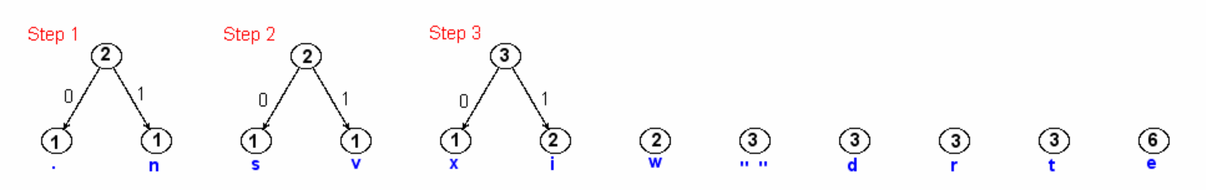
Then we will check the efficiency of the code, apply it and demonstrate how to decode the encoded message into a readable form again.

This is another definition (found in http://en.wikipedia.org/wiki/Huffman_coding):

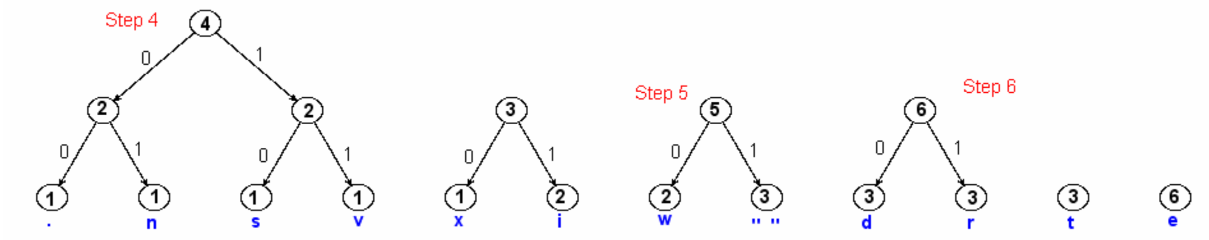
The process essentially begins with the leaf nodes containing the probabilities of the symbol they represent, then a new node whose children are the 2 nodes with smallest probability is created, such that the new node's probability is equal to the sum of the children's probability. With the previous 2 nodes merged into one node (thus not considering them anymore), and with the new node being now considered, the procedure is repeated until only one node remains, the Huffman tree.

Steps 1 to 3:

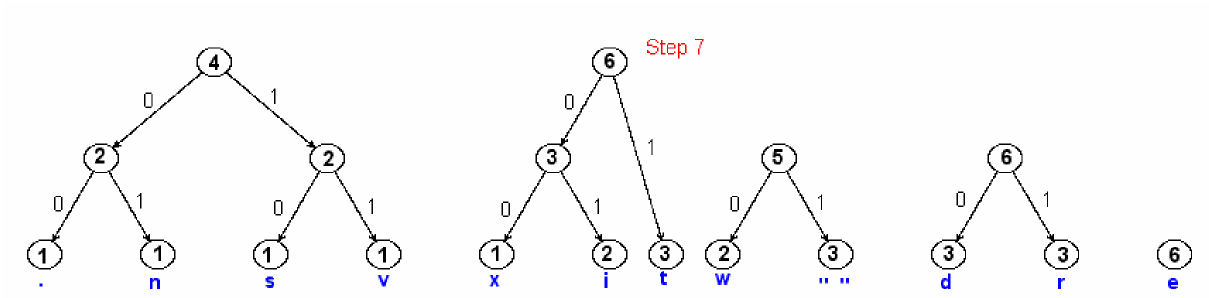
We select the pairs of nodes with minimal weights one after the other.



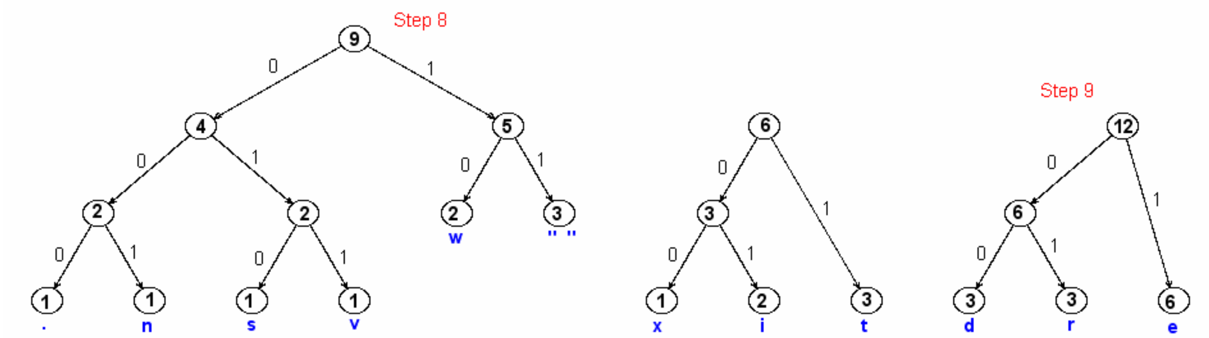
Steps 4 to 6: Pair (2,2) is connected by parent node 4. Then we find pair (2,3) and connect them giving node with value 5. “d” and “r” are following in step 6.



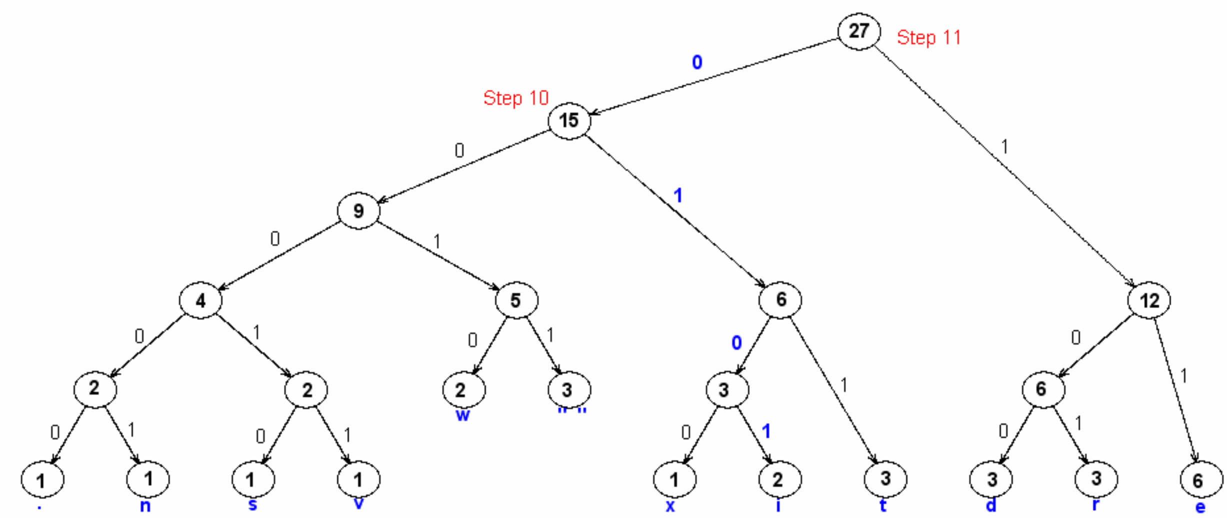
Inspecting the graph we see that we need to connect the “t”-node with the value 3 node in the second row according to the algorithm rules. I rearrange the base line and insert the connection resulting in a parent node with value 6:



The minimum nodes to be connected by edges are form pair (4,5) giving the sum 9. It is again necessary to rearrange the binary tree in order to obtain a clear structure - without crossing edges (step 8)). Step 9 results in node 12.



Finally we connect 6 and 9 and the last step gives the “root” of the binary tree (or **Huffman-Tree**) with value (or weight) 27.



Having connected 9 and 6 to 15 the branch starting with node 12 is remaining. Both have the same root (parent) 27.

Only one vertex (27) with no arrow directed is remaining. According to our instructions the job is done. It is an easy check to compare the weight of the final vertex (the root of the binary tree) with the sum of all (absolute) frequencies which is 27.

I used Johann Wiesenbauer´s tool for plotting directed graphs which appeared in DNL#78. These are the expressions which result iwhen plotted n the final Huffman-Tree:

$$\text{huff5} := \begin{bmatrix} -11 & -9 & -7 & -5 & 1 & 3 & 5 & 7 & 9 & 11 & -10 & -6 & -3 & -1 & 2 & 8 & -8 & -2 & 3 & -5 & 10 & -1 & 4.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 4 & 4 & 4 & 6 & 4 & 8 & 10 \end{bmatrix}$$

```
h5 := [{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {1, 2}, {3, 4}, {}, {}, {5, 6}, {8,
  9}, {11, 12}, {13, 14}, {15, 7}, {17, 18}, {16, 10}, {20, 19}, {21, 22}]

graph(huff5', h5, 0.4, true)
```

We observe that frequently used characters are located close to the root, rarely ones can be found in the “tops of the tree“.

The codes for the characters are yielded by following the graph from the “root” to the “leaf“ and noting the labels of the edges along the path. The code for the “i“ is 0101 (follow the blue numbers), the code for the “x“ is 0100, the “r“-code is 101 and the “v“ is encoded by 00011. You see again that rare characters result in long paths which are equivalent to long code words and frequently appearing characters give short paths and consequently short code words. The Huffman-Code is the code which needs the minimal number of bits. (This can be proved.)

Here is the complete code - given as a matrix - followed by the encoded message:

(demo is the message “dieser text wird verwendet.”)

$$\#10:\quad \text{hcode1} := \begin{bmatrix} \text{e} & \text{d} & \text{r} & \text{t} & \text{i} & \text{x} & \text{ } & \text{w} & \text{v} & \text{s} & \text{n} & . \\ 11 & 100 & 101 & 011 & 0101 & 0100 & 0011 & 0010 & 00011 & 00010 & 00001 & 00000 \end{bmatrix}$$

```
#11:  mess := huffcode(demo, hcode1)

#12:  mess :=

        10001011100010111010011011110100011001100100101101100001100011111010010110~
        00011001101100000

#13:  DIM(mess) = 91
```

The characters e, d, … and the code words 11, 100, … are strings. So they are entered under quotes: “e”, “d”, …,”11”, “100”, …

The encoding procedure is done by a small program huffcode:

```
        huffcode(plain, code, p, cplain) :=
          Prog
            cplain := ""
            code := code`
            p := VECTOR(plain↓i, i, DIM(plain))
#9:         Loop
              If p = []
                 RETURN cplain
              cplain := APPEND(cplain, (SELECT(v↓1 = FIRST(p), v, code))↓1↓2)
              p := REST(p)
```

The encoded message is decoded by following the path starting in the root bit for bit (go left for 0 or right for 1) until reaching a leaf. There you will find the respective character:

"100|0101|11|00010| ….. results in: dies ...

Of course, the code must be transmitted together with the encoded message (if the code is generated from the message). Each language has typical frequencies for the occurrences of the letters. If the partners agree – and the message is sufficiently long – you can do without sending the code and rely on the typical frequency of the letters in the respective language.

Let´s try decoding using huffdecode:

```
        huffdecode(codtxt, code, plain, z, zz, branch, k) :=
          Prog
            k := 0
            plain := ""
            code := code`
            Loop
              If codtxt = ""
                 RETURN plain
#14:          codtext := REST(codtxt)
              branch := FIRST(codtxt)
              Loop
                z := SELECT(v↓2 = branch, v, code)
                If z ≠ [] exit
                codtxt := REST(codtxt)
                branch := APPEND(branch, FIRST(codtxt))
              codtxt := REST(codtxt)
              plain := APPEND(plain, z↓1↓1)

#15:  huffdecode(mess, hcode1) = dieser text wird verwendet.
```

It seems to work!

Possible questions and problems for students:

1 It is possible to read off the length of the encoded message from the Huffman-Tree. Can you do this?

2 Compare the number of the bits of the plain text "demo" and its compressed form. If we had only 12 characters we would need only 4 bits applying conventional coding. Would we benefit of Huffman-encoding?

3 Decode the compressed mess manually.

4 Why does the "adaptive method", which determines the code from the plain text (as we did it here) bring a real advantage only if applied for longer messages?

6 Do a web research for frequency tables of letters in various languages.

7 Encode the message “this sentence will be compressed”. Develop the Huffman-Tree in order to find the code, encode and decode.

The programs for encoding and decoding are not so difficult. The real challenge for me was writing a program which returns the code – such that I need not “plant” the Huffman-Tree.

```
      hufftree(plain, s := 0, plainh, tab, e1, e2, f1, f2, ne, tree) :=
        Prog
          plainh := freq(plain)
          tab := VECTOR([v↓1, [v↓2, ""]], v, plainh')
          tree := [[]]
          Loop
            tree := APPEND(tree, [tab])
            If DIM(tab) = 1
               If s = 0
                  RETURN tab↓1↓2'
                  RETURN REST(tree)
#16:        e1 := (FIRST(tab))↓2
            If e1↓2 = ""
               e1 := [e1]
            f1 := VECTOR([v↓1, APPEND("0", v↓2)], v, e1)
            e2 := (FIRST(REST(tab)))↓2
            If e2↓2 = ""
               e2 := [e2]
            f2 := VECTOR([v↓1, APPEND("1", v↓2)], v, e2)
            ne := [(FIRST(tab))↓1 + (FIRST(REST(tab)))↓1, APPEND(f1, f2)]
            tab := REST(REST(tab))
            tab := APPEND([ne], tab)
            tab := SORT(tab)
```

#17: hufftree(demo) =

e	d	r	t	x	i	w	.	n	s	v	
00	010	011	100	1010	1011	1100	11010	11011	11100	11101	1111

Comparing with hcode1 you will notice that both codes are different. Yes, it is true, there is no unique optimal code. We should further see that the encoded message using this code will have the same length of 91 characters.

#18: democode :=

e	d	r	t	x	i	w	.	n	s	v	
00	010	011	100	1010	1011	1100	11010	11011	11100	11101	1111

```
#19:  demo_mess := huffcode(demo, democode)

#20:  demo_mess :=

        010101100111000001111111000010101001111110010110110101111111010001111000011011010~
        0010011010

#21:  DIM(demo_mess) = 91

#22:  huffdecode(demo_mess, democode)

#23:  dieser text wird verwendet.
```

We will apply huftree on the longer message paragraph from above. Then we will encode and decode this text.

```
#30:   par_code := hufftree(paragraph)

#31:   par_code := [ d     t     n    x       v       o      u      ,        f
                     0000  0001  001  010000  010001  01001  01010  0101100  0101101

       g        p        r     a      c      h      l      w      b       z
       0101110  0101111  0110  01110  01111  10000  10001  10010  100110  100111  101

       s      .        k        m       i     e   ]
       11000  1100100  1100101  110011  1101  111 ]

#32:   par_mess := huffcode(paragraph, par_code)

#33:   par_mess :=

       10011011111011010000111011010111001010100111001101011110110110111001111011111101100~
       101000101011101010100010100100110100011110100000001111001101100010111011111000110~
       000001101110011011100011011100011010111110000101010001010010011010000111011010101~
       010001000111101101100001111100001101111000010001110101111100001110011011000001110~
       111010100101101110101011101100101111110100011010000111011010110011111111010111110~
       000111001101110100110100001111100111011001110101010101111010010000110111101101110~
       010000111001101000111101000000001101100011111101000111100111001001011001011010101101~
       011001001001100011000111100110100000111011000101011100011011111101001111110011101~
       111110100101011010111001111100001110011011001101111101110000101111110111110001101~
       000011111001101001001110000000101101101111011011100101011001011001001001100110111 1~
       101101100101101011010100101010011010111001011000111111010011001100101001111100001~
       100000010111010011011100101011001011001111001011011100001111100001110010110011101~
       110101011001111110101010001000010111000011100001100111100111111110101111100001110~
       011010100011110110100101110010000111001101100100100110001100011110011100100101000~
       011011111100011101101010001111010000000011011001011010110000010101000111101101001 0~
       1110010000111000111001 00

#34:   DIM(par_mess) = 1239

#35:   huffdecode(par_mess, par_code)

#36:   bei der komprimierung von texten laesst man sich von der unterschiedlichen
         haeufigkeit der zeichen in dem zu codierenden text leiten. wir wollen das an
         einem einfachen beispiel demonstrieren, wobei wir nur kleinbuchstaben,
         zwischenraeume und satzzeichen verwenden wollen. dieser text wird verwendet.
```

My program hufftree2 has a tighter code and returns another – but also optimal Huffman-Code.

```
#39:   hufftree2(demo) = [ d    r    e   t    x     .      n      i     s      v      w
                           000  001  01  100  1010  10110  10111  1100  11010  11011  1110

       1111 ]

#40:   huffcode(demo, hufftree2(demo))

#41:   000110001110100100111111000110101001111111011000010001111110110100111100110111000 01~
         10010110

#42:   DIM(huffcode(demo, hufftree2(demo))) = 91
```

```
      hufftree2(plain, s := 0, plainh, tab, e1, e2, f1, f2, ne, tree) :=
        Prog
          plainh := freq(plain)
          tab := VECTOR([v↓1, [[v↓2, ""]]], v, plainh')
          tree := [[]]
          Loop
            tree := APPEND(tree, [tab])
            If DIM(tab) = 1
               If s = 0
#37:              RETURN tab↓1↓2'
                  RETURN REST(tree)
            e1 := (FIRST(tab))↓2
            f1 := VECTOR([v↓1, APPEND("0", v↓2)], v, e1)
            e2 := (FIRST(REST(tab)))↓2
            f2 := VECTOR([v↓1, APPEND("1", v↓2)], v, e2)
            ne := [(FIRST(tab))↓1 + (FIRST(REST(tab)))↓1, APPEND(f1, f2)]
            tab := REST(REST(tab))
            tab := APPEND([ne], tab)
            tab := SORT(tab)
```

Default for s (second parameter in the parameter list) in hufftree and hufftree2 as well is 0. If you enter any other value then you will obtain another output: You can follow the Huffman-Tree growing (from its leaves and branches down to its root). I will demonstrate this using again demo:

#24: hufftree(demo, 1)

#25:

$$\left[\begin{bmatrix} 1 & \text{[.,]} \\ 1 & \text{[n,]} \\ 1 & \text{[s,]} \\ 1 & \text{[v,]} \\ 1 & \text{[x,]} \\ 2 & \text{[i,]} \\ 2 & \text{[w,]} \\ 3 & \text{[,]} \\ 3 & \text{[d,]} \\ 3 & \text{[r,]} \\ 3 & \text{[t,]} \\ 6 & \text{[e,]} \end{bmatrix}, \begin{bmatrix} 1 & \text{[s,]} \\ 1 & \text{[v,]} \\ 1 & \text{[x,]} \\ 2 & \text{[i,]} \\ 2 & \text{[w,]} \\ 2 & \begin{bmatrix} . & 0 \\ n & 1 \end{bmatrix} \\ 3 & \text{[,]} \\ 3 & \text{[d,]} \\ 3 & \text{[r,]} \\ 3 & \text{[t,]} \\ 6 & \text{[e,]} \end{bmatrix}, \begin{bmatrix} 1 & \text{[x,]} \\ 2 & \text{[i,]} \\ 2 & \text{[w,]} \\ 2 & \begin{bmatrix} . & 0 \\ n & 1 \end{bmatrix} \\ 2 & \begin{bmatrix} s & 0 \\ v & 1 \end{bmatrix} \\ 3 & \text{[,]} \\ 3 & \text{[d,]} \\ 3 & \text{[r,]} \\ 3 & \text{[t,]} \\ 6 & \text{[e,]} \end{bmatrix}, \begin{bmatrix} 2 & \text{[w,]} \\ 2 & \begin{bmatrix} . & 0 \\ n & 1 \end{bmatrix} \\ 2 & \begin{bmatrix} s & 0 \\ v & 1 \end{bmatrix} \\ 3 & \text{[,]} \\ 3 & \text{[d,]} \\ 3 & \text{[r,]} \\ 3 & \text{[t,]} \\ 3 & \begin{bmatrix} x & 0 \\ i & 1 \end{bmatrix} \\ 6 & \text{[e,]} \end{bmatrix}, \begin{bmatrix} 2 & \begin{bmatrix} s & 0 \\ v & 1 \end{bmatrix} \\ 3 & \text{[,]} \\ 3 & \text{[d,]} \\ 3 & \text{[r,]} \\ 3 & \text{[t,]} \\ 3 & \begin{bmatrix} x & 0 \\ i & 1 \end{bmatrix} \\ 4 & \begin{bmatrix} w & 0 \\ . & 10 \\ n & 11 \end{bmatrix} \\ 6 & \text{[e,]} \end{bmatrix}, \begin{bmatrix} 3 & \text{[d,]} \\ 3 & \text{[r,]} \\ 3 & \text{[t,]} \\ 3 & \begin{bmatrix} x & 0 \\ i & 1 \end{bmatrix} \\ 4 & \begin{bmatrix} w & 0 \\ . & 10 \\ n & 11 \end{bmatrix} \\ 5 & \begin{bmatrix} s & 00 \\ v & 01 \\ & 1 \end{bmatrix} \\ 6 & \text{[e,]} \end{bmatrix},\right.$$

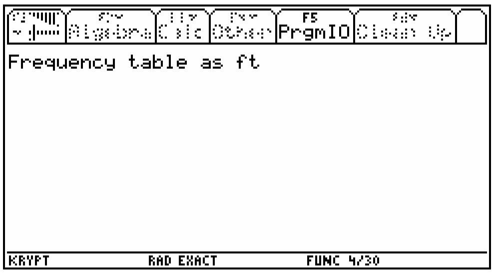
$$\left.\begin{bmatrix} 3 & \text{[t,]} \\ 3 & \begin{bmatrix} x & 0 \\ i & 1 \end{bmatrix} \\ 4 & \begin{bmatrix} w & 0 \\ . & 10 \\ n & 11 \end{bmatrix} \\ 5 & \begin{bmatrix} s & 00 \\ v & 01 \\ & 1 \end{bmatrix} \\ 6 & \text{[e,]} \\ 6 & \begin{bmatrix} d & 0 \\ r & 1 \end{bmatrix} \end{bmatrix}, \begin{bmatrix} 4 & \begin{bmatrix} w & 0 \\ . & 10 \\ n & 11 \end{bmatrix} \\ 5 & \begin{bmatrix} s & 00 \\ v & 01 \\ & 1 \end{bmatrix} \\ 6 & \text{[e,]} \\ 6 & \begin{bmatrix} d & 0 \\ r & 1 \end{bmatrix} \\ 6 & \begin{bmatrix} t & 0 \\ x & 10 \\ i & 11 \end{bmatrix} \end{bmatrix}, \begin{bmatrix} 6 & \text{[e,]} \\ 6 & \begin{bmatrix} d & 0 \\ r & 1 \end{bmatrix} \\ 6 & \begin{bmatrix} t & 0 \\ x & 10 \\ i & 11 \end{bmatrix} \\ 9 & \begin{bmatrix} w & 00 \\ . & 010 \\ n & 011 \\ s & 100 \\ v & 101 \\ & 11 \end{bmatrix} \end{bmatrix}, \begin{bmatrix} 6 & \begin{bmatrix} t & 0 \\ x & 10 \\ i & 11 \end{bmatrix} \\ 9 & \begin{bmatrix} w & 00 \\ . & 010 \\ n & 011 \\ s & 100 \\ v & 101 \\ & 11 \end{bmatrix} \\ 12 & \begin{bmatrix} e & 0 \\ d & 10 \\ r & 11 \end{bmatrix} \end{bmatrix}, \begin{bmatrix} 12 & \begin{bmatrix} e & 0 \\ d & 10 \\ r & 11 \end{bmatrix} \\ 15 & \begin{bmatrix} t & 00 \\ x & 010 \\ i & 011 \\ w & 100 \\ . & 1010 \\ n & 1011 \\ s & 1100 \\ v & 1101 \\ & 111 \end{bmatrix} \end{bmatrix},\right.$$

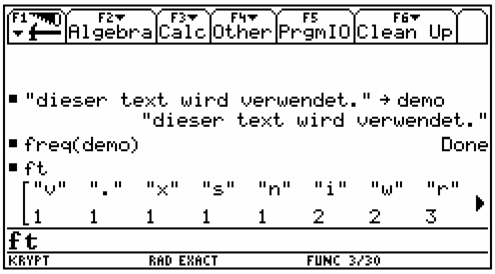
$$\left[\left[27,\ \left[\begin{array}{ll} e & 00 \\ d & 010 \\ r & 011 \\ t & 100 \\ x & 1010 \\ i & 1011 \\ w & 1100 \\ . & 11010 \\ n & 11011 \\ s & 11100 \\ v & 11101 \\ & 1111 \end{array}\right]\right]\right]$$

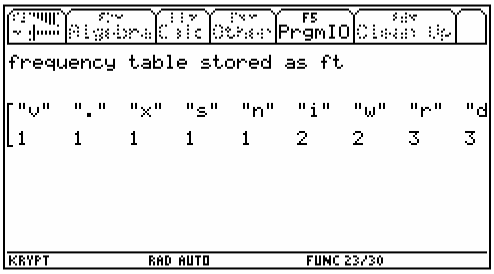
I am sure that you can "read" the output. You can see step by step how the vertices are collected and how the weights are added. The last element gives the weight of the root and the complete code.

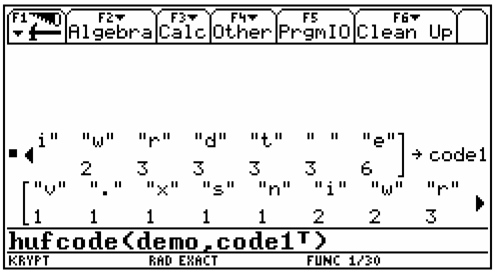
Solution for task 7 (page 12): The encoded message should contain 120 characters.

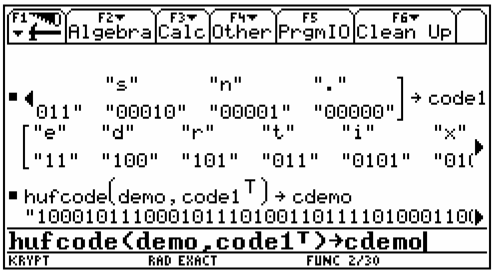
Huffman-Code on the TIs:





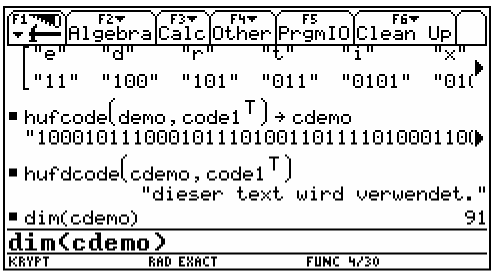






I did not program (until now) the hufftree routine to generate a code. Maybe that one of the readers will transfer the DERIVE program from page 12 to the Voyage 200 or to TI-NspireCAS?
Let me know.

(The TI-functions are among the files which can be downloaded.)



The Josephus Problem

Roland Schröder, Celle, Germany

In 70 AC 40 rebellious Jews were captured in Rome which should be sold as slaves in punishment for their behaviour. In order to avoid their doom they agreed on a procedure for mutual extinction: They formed a circle and every seventh in the row should be killed (continuing counting in the same direction). The remaining last one should commit suicide. The later historian Flavius Josephus chose a position that he remained as the last one – and he didn't commit suicide.

The story is wholly invented. In Scandinavia the following legend is passed on: In times when St Petrus strolled on earth he repaired on a ship with 15 Swedes and 15 Norwegian. The ship got in a thunderstorm followed by distress at sea. Salvage seems only possible if half of the passengers will go overboard. St: Petrus provides the following counting method: The passengers form a circle. St. Petrus starts at a certain position counting until 9. Person number 9 has to leave the ship (jump from board) and the circle will be closed immediately. St. Petrus keeps the direction and counts again up to 9 – and the next person jumps (or will be thrown). The procedure goes on until 15 people are remaining. How had Petrus organized the starting positions of the 30 passengers that the Swedes – which were preferred by him – had been saved? (Sorry for our Norwegian DUG-Members!)

But this story is also imaginary. There are also versions with Christians and Turks, (and people from Vienna and Klagenfurt in DNL#52, Rüdeger Baumann). The mathematical problem behind is: n elements are arranged in a circle and numbered from 1 to n. Then every k^{th} element is removed. Which element (Josephus) or which elements (St. Petrus) will remain after e countings?

Putting oneself in an affected person's position, one would like to have an easy and quick algorithm available to find an advantageous position for oneself. In both tales narrated above the numbers are small, so the simulation of counting by paper and pencil will deliver the requested solution very soon. Taking numbers above n = 100 makes manual simulation so laborious that it makes sense to look for a more elegant solution of the problem. The first mathematical treatment of the problem originates – according to the author´s knowledge – from Leonhard Euler, who found a recursion formula, which will not be used now.

Instead of this we will develop some DERIVE-functions. The function "jo(x,yk)" shall relieve us of the manual paper and pencil work and deliver immediately the surviving person. The meaning of the functions are self explanatory calculating some appropriate examples. We will do this now.

[1] DNLs #52, 57 (Josephus Problem) & #53, 55 (Josephus Permutations)
[2] http://web.me.com/ntheriau/josephus.pdf (Generalization of the Josephus Problem, Tait´s Algorithm))
[3] Donald Knuth a.o., Concrete Mathematics

```
#1:   r(x) := VECTOR(y, y, x)

#2:   r(12) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

      g(v, k) := VECTOR(v , n, IF(k = DIM(v), DIM(v) - 1, MOD(k, DIM(v))))
#3:                      n

#4:   g(r(12), 7) = [1, 2, 3, 4, 5, 6, 7]

      f(v, k) := VECTOR(v , n, IF(k = DIM(v), DIM(v) + 1, MOD(k, DIM(v)) + 1), DIM(v))
#5:                      n

#6:   f(r(12), 7) = [8, 9, 10, 11, 12]

#7:   h(v, k) := APPEND(f(v, k), g(v, k))

#8:   h(r(12), 7) = [8, 9, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7]
```

h(v, k) puts together the two parts g(v, k) and f(v, k) of the vector v = r(x) in inverse order. Then the last element of h(v,k) is removed and the same procedure is applied on this newly generated vector.

Taking to pieces – Inverse joining – Removing the last element.

```
#9:   i(v, k) := DELETE(h(v, k), DIM(v))

#10:  i(r(12), 7) = [8, 9, 10, 11, 12, 1, 2, 3, 4, 5, 6]

      jo(x, k) := (ITERATE(i(w, k), w, r(x), x - 1))
#11:                                                 1

#12:  jo(40, 7) = 24

#13:  jo(41, 3) = 31

#14:  jo(100, 7) = 50
```

Josephus stood on position 24 – and he survived.

We produce a value table and the respective graph for the relation between the number of persons forming the circle and the position number of the surviving person with a fixed k (here k = 7, k = 5 and k = 3):

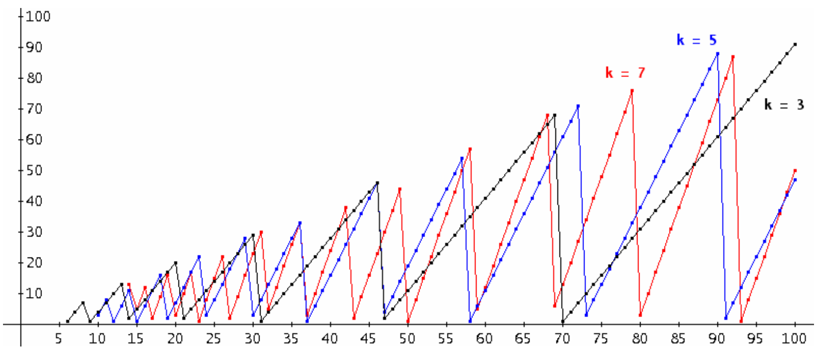
```
#15:  Joseph(y, k) := VECTOR([x, jo(x, k)], x, 2·k, y)

#16:  Joseph(100, 7)

#17:  Joseph(100, 5)

#18:  Joseph(100, 3)
```

The tables are leading to scatter diagrams which look as follows:

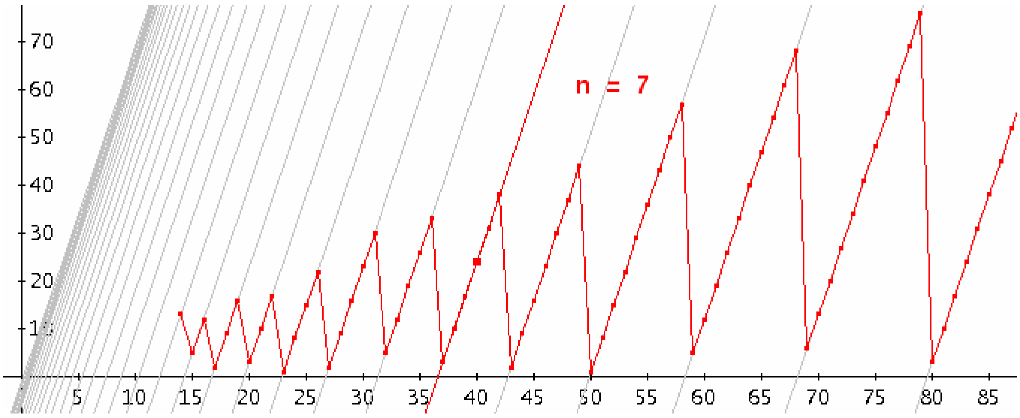


The points are lying on lines with slope k. The accurate knowledge of the equations of the lines - their y-intercepts in addition to k - can help solving the Josephus problem in another way. When among n participants every kth will drop out then the y-intercepts can be generated recursively (the proof is left for the reader).

#16: $L(n, k) := \text{ITERATES}\left(\text{FLOOR}\left(\frac{k\cdot x}{k - 1}\right) + 1, x, 1, n\right)$

#17: L(40, 7) = [1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 18, 22, 26, 31, 37, 44, 52, 61, 72, 85, 100, 117, 137, 160, 187, 219, 256, 299, 349, 408, 477, 557, 650, 759, 886, 1034, 1207, 1409, 1644, 1919, 2239]

There is one line with equation y = 7x + b which contains point [40; jo(40,7)]. As the y-value must be positive and less or equal x only one line is possible: y = 7x – 256. It contains point (40; 24). So follows: Josephus survived standing on position 24.



The sequence L(40,7) can easily be found by paper and pencil: For calculating the successor of a_n one has to find the next number $b_n > a_n$ which is divisible by 6 – even then when a_n is divisible by 6. Then $a_{n+1} = a_n + b_n/6$. The procedure is terminated when $a_{n+1} > 7 \cdot 40$. Then a_n is the requested y-intercept and $280 – a_n$ is the position to surive.

(I tried to automate the procedure using SELECT, Josef):

#21: $\text{pos}(n, k) := (\text{SELECT}(p > 0 \wedge p < n, p, \text{SUBST}(\text{VECTOR}(k\cdot x - b, b, L(n, k)), x, n)))_1$

#22: [pos(40, 7), pos(100, 7), pos(100, 2)] = [24, 50, 73]

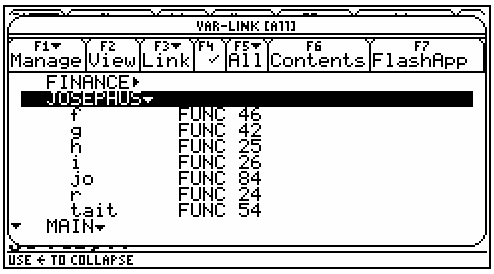
I found a nice recursive algorithm to produce the jo's[2]:

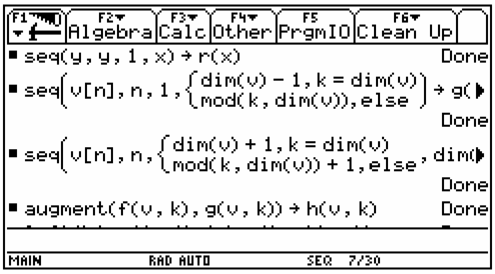
```
#24:  tait(n, k) :=
        If n = 1
           1
           MOD(tait(n - 1, k) + k - 1, n) + 1
```

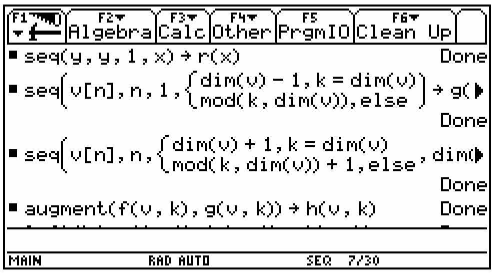
#25: [tait(40, 7), tait(100, 7), tait(100, 2)] = [24, 50, 73]

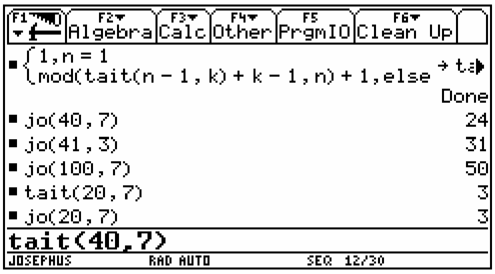
See the TI-treatment of JOSEPHUS on the next page, Josef.

The functions are named as in JOSEPHUS from above.

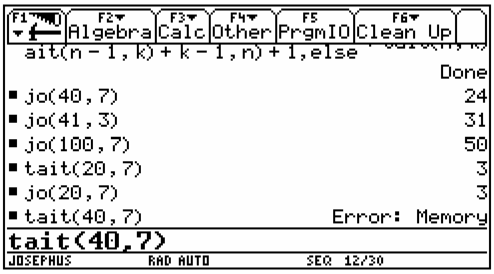




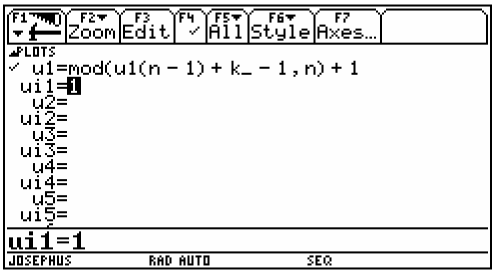


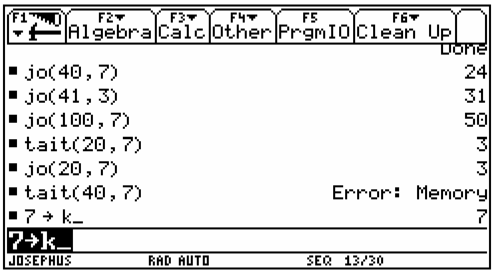


Calculating jo takes some time:

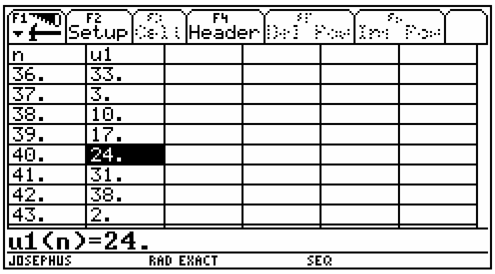


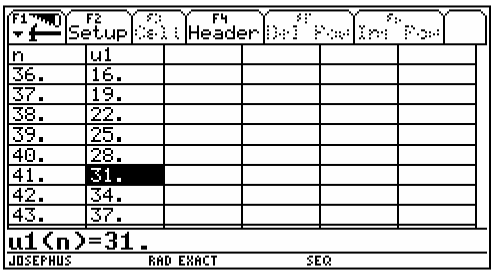
I am very soon Out of Memory!



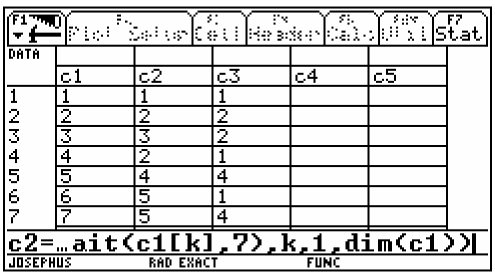


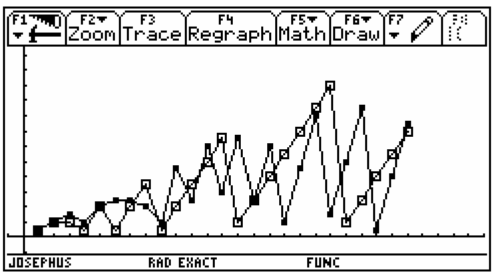
I define the recursive function in Sequence Mode





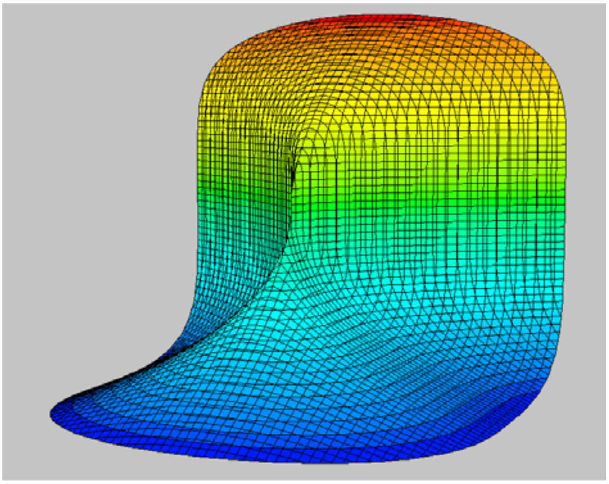
Check the „last positions“!

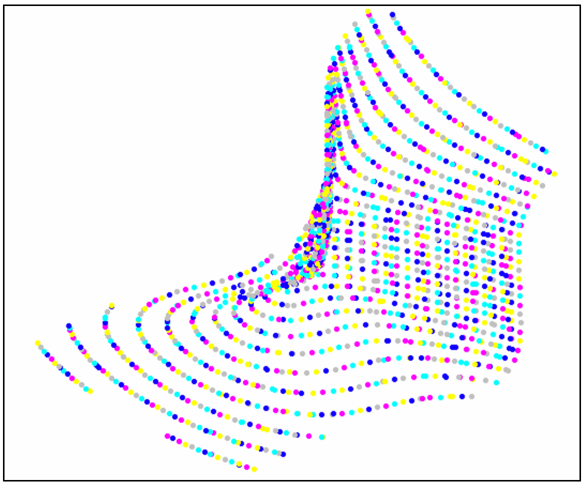




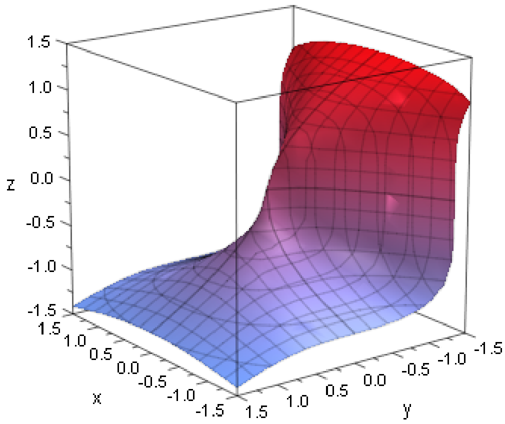
Finally the “Joseph“-Scatter diagrams:

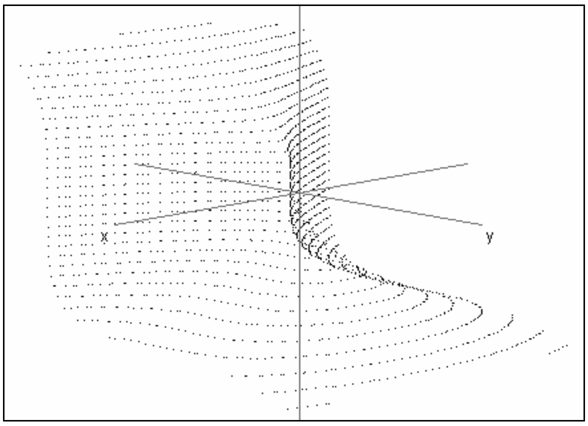
Surface #11: $x^2 + y^3 + z^5 = 0$





```
plot(plot::Implicit3d(x^2 + y^3 + z^5,
                      x = -1.5..1.5,
                      y = -1.5..1.5,
                      z = -1.5..1.5),
     Scaling = Constrained)
```

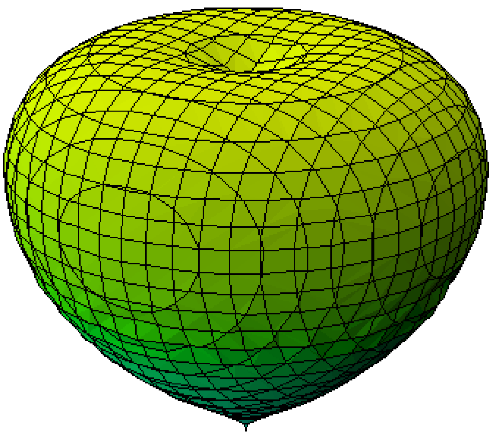


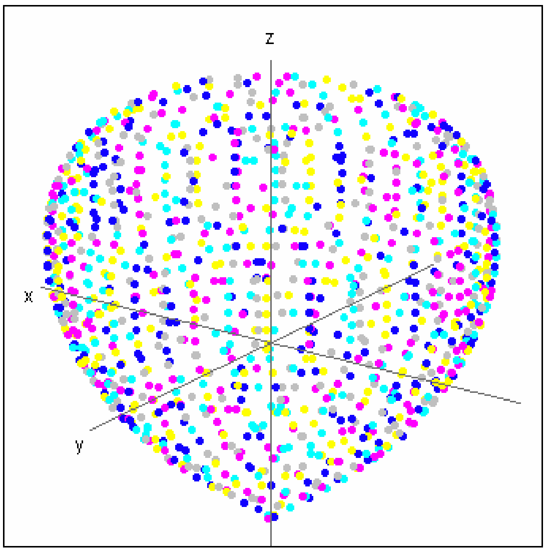


DPGraph and MuPad

DERIVE

Surface #12: $\left(x^2 + y^2 + z^2 - 1\right)^3 = x^2 z^3 + y^2 z^3$





DPGraph

DERIVE

Nonlinear Regression, Logistic Regression for Binary Dependent Data, And Two-Stage Least Squares Regression for the TI-89

MacDonald R. Phillips, don.phillips@gmail.com

The routines in this folder solve nonlinear regression problems using the Gauss_Newton Method with Step-Halving, logistic regression problems for binary dependent data using the probit, normit, or complementary log-log link functions, and two-stage least squares regression.

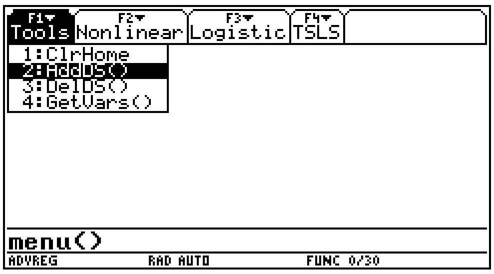
NOTE: These programs are offered "as is." I make no claim that they are entirely bug free, although I believe they are. If you encounter any problems with the programs, please send me an email so I can correct them.

NOTE: These programs require the use of the Statistics with List Editor Flash application.

My aim is to teach you how to use these programs, not to teach statistics. Thus, when I mention the ANOVA table or logit link function, I assume you already know what they are and/or when they are used, or are learning about them either in a class or on you own.

Fitting data to an arbitrary function is more of an art than a science. Convergence to a solution can be very sensitive to the initial starting values, i.e., guesses. And, there may be more than one solution or local minimum around the starting values. If you get error messages such as singular matrix, this may mean that there is no solution or you need to choose a different set of starting values.

The routines are in a group file, AdvReg.89g. Use TI-Connect to transfer them to your calculator. There is a menu program; this needs to be run in order to use the regression routines. The custom menu sets up four pull-down menus: Tools, Nonlinear, Logistic, and TSLS.



Don´t worry about the changed menu bar. [2ND] [3] restores the default menu.

All regression routines use a data matrix created with the Data/Matrix Editor. The data matrix consists of the variables in any order. The first row of the matrix <u>must</u> be the variable names; I recommend one-letter names. In any case, just make sure they do not conflict with any of the variable names in the AdvReg folder or TI reserved names. (There are no variables with one-letter names in the folder.) When performing regression there is no need to use all of the variables in a dataset; this means you can do many different regressions on a dataset without having to enter a new data matrix each time. It also means you can do "model building" and test the significance of adding variables to a regression equation; this ability is one of the routines.

Give the data matrix a name and save it. Open the Tools menu (F1) and select AddDS(). You will be prompted to enter the name of the data matrix. When you run NonLin(), Logit(), or TSLS() you will be asked to select a dataset from the list of datasets created with AddDS(). AddDS() also archives the dataset.

Tools Menu

The options under the F1:Tools menu are straightforward. Option 1 clears the home history screen.

Option 2, AddDS(), prompts you to add the name of a dataset to the list of datasets; this list is how you tell the programs what dataset to use. It also archives the dataset.

Option 3, DelDS(), deletes a dataset from memory and the dataset list when you select its name from dataset list.

Option 4, GetVars(), displays the variables in a dataset, in case you forgot what they were, as well as the number of observations in the dataset. Choose the dataset from the list presented.

Nonlinear Regression

The nonlinear regression routine can also handle weighted regressions as well as regressions with complex data. However, I'm not sure one can do weighted regressions with complex data; at least I've never seen an example against which I can test the program. The nonlinear regression program will, of course, do linear regressions.

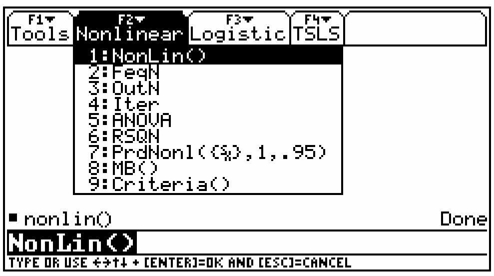
<u>Example 1</u>

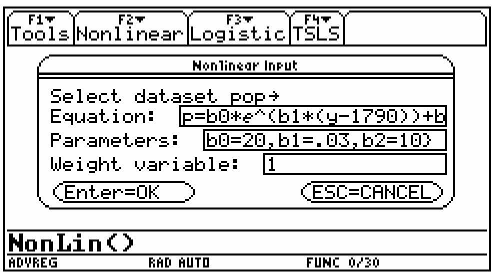
Most regressions are linear regressions or can be transformed into linear regressions. Some, however, cannot be transformed. For instance, an exponential equation of the form

$$p_i = b_0 \times \exp(b_1 \times (y_i - 1790)) + b_2$$

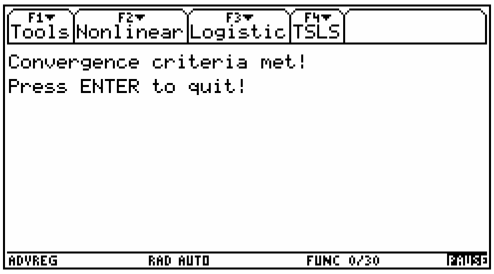
may model the growth of the U.S. population by decade from 1790, but it cannot be transformed into a linear regression problem. (If the b_2 variable was not there, it could be transformed into a linear regression problem.) A nonlinear regression program is needed. Press F2, Nonlinear, and select the first menu item, NonLin(), and then press enter, once or twice as needed. This sets up a data input form.

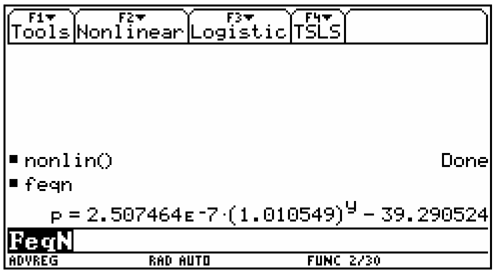
The first item is "Select dataset." Press the right arrow key to see the list of datasets. Scroll to "pop," if needed, and press ENTER. "pop" has the U.S. population figures, in millions, from 1790 to 1990, by decade. Now, press the down arrow key to enter the regression equation. The next line is used to enter the regression equation; the equation there is the one displayed above. So enter p=b0*e^(b1*(y-1790))+b3. Press the down arrow key to enter a list of the parameters and their initial guesses. Enter {b0=20, b1=.03, b2=10}. Finally, enter the weight variable; if none, enter the number 1. The screen should look like this:





After the data is input, press enter to begin computing the regression. The program keeps you informed as to what is going on. It first sets up the necessary matrices, etc., needed to compute the regression. After that, each iteration is displayed along with the current sum-of-squared-errors. At the end, a message will be displayed indicating whether or not the routine converged to an answer. (As seen below, the convergence criteria can be changed.)





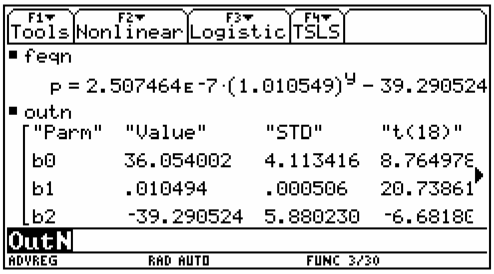
The other options under F2 (Nonlinear) display the output of the regression.

Option 2, FeqN, displays the fitted equation. In this case it is

$$p = 2.50746E^{-7}(1.01055)^{y} - 39.29052$$

(It is unfortunate that the calculator simplifies the answer instead of leaving it in the form of an exponential equation.)

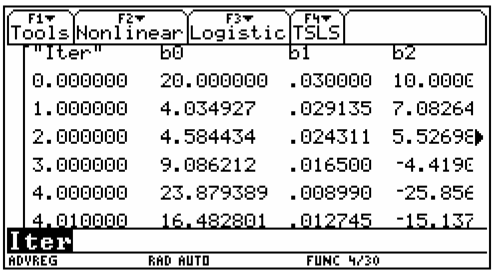
Option 3 under the F2 menu, OutN, displays a matrix of the parameters, their values, standard errors, t values and probability(t). For this regression the output is

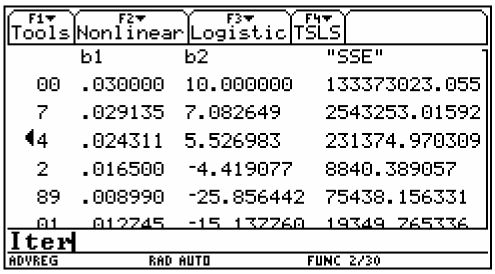


"Parm"	"Value"	"STD"	"$t(18)$"	"Prob(t)"
$b0$	36.054	4.11342	8.76498	$6.51895E^{-8}$
$b1$	.010494	.00051	20.73862	$5.1456E^{-14}$
$b2$	-39.29052	5.88023	-6.6818	$2.87623E^{-6}$

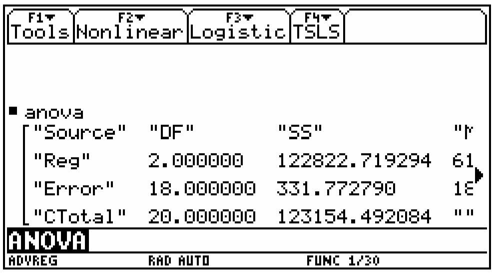
(The 18 in “t(18)” is the degrees of freedom of the t statistics.)

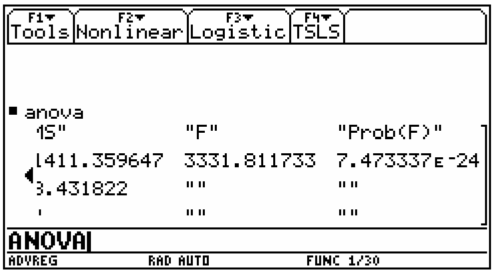
Option 4 under the F2 menu, Iter, displays a matrix of the iterations the program went through to reach the estimated values of the parameters. The iteration number, or sub iteration number, parameter values, and sum-of-square errors are displayed for each iteration.





Option 5 under the F2 menu, ANOVA, displays the analysis of variance matrix.

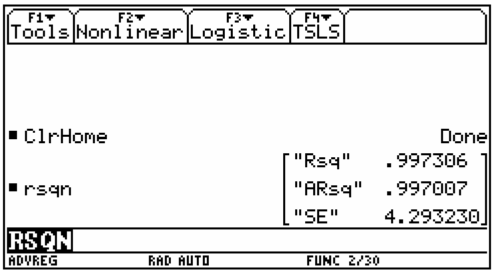




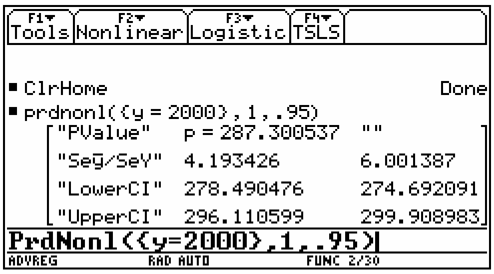
"Source"	"DF"	"SS"	"MS"	""F"	"Prob(F)"
"Reg"	2.	122823.	61411.4	3331.81	$7.47334E^{-24}$
"Error"	18.	331.773	18.4318	""	""
"CTotal"	20.	123154.	""	""	""

The “CTotal” in the ANOVA matrix stands for corrected total degrees of freedom and sum of squared errors. The corrected totals are used when there is an intercept in the regression equation. If there is no intercept, then the uncorrected totals are used. However, R^2 and adjR^2 are always computed with the corrected totals. (See a “Cautionary Note About R^2” by Tarald O. Kvalseth in *The American Statistician*, November 1985, pp. 279-85.)

Option 6 under the F2 menu displays the R square, adjusted R square, and standard error of the regression statistics. For this problem they are: 0.99731, 0.99701, and 4.29323.

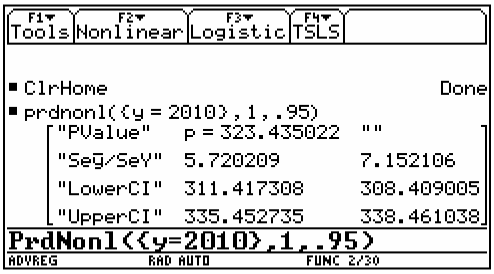


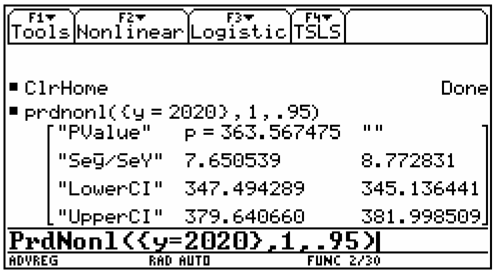
Option 7 under F2, PrdNonl({ }, 1, .95), computes the predicted values for the mean and individual values of the dependent variable. The default weight is 1 (indicating no weight) and the default confidence interval is .95 for a 95 percent confidence interval. Enter a list of the independent variables and their values. In this case find the estimated population for the year 2000. Enter y=2000 in the list.



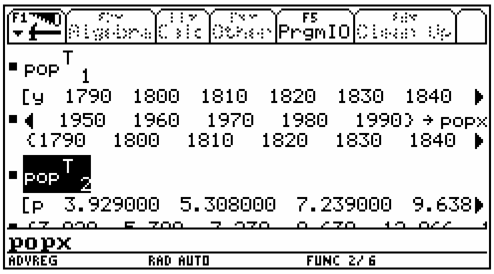
"PValue"	p = 287.30054	""
"Seȳ/SeY"	4.19343	6.00139
"LowerCI"	278.49048	274.69209
"UpperCI"	296.11060	299.90898

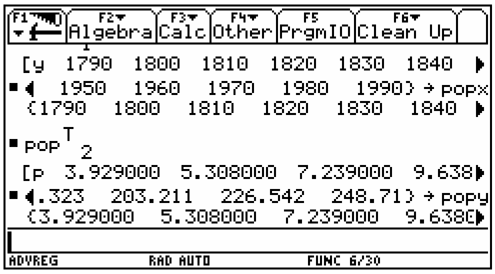
The first line of the matrix gives the predicted value of the equation for the year 2000, 287.3 million people. The second line gives the standard errors of the mean and individual values of the dependent variable, in this case p. The third and fourth lines give the 95 percent confidence interval for the mean and individual values of p for the year 2000.

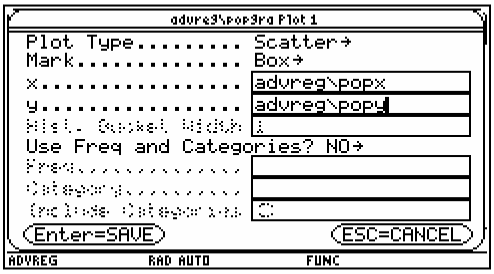


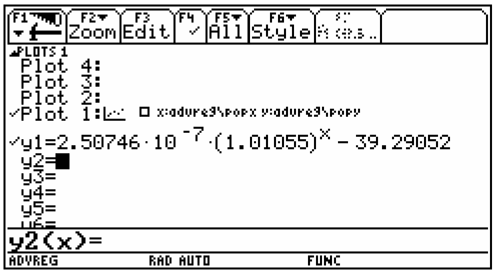


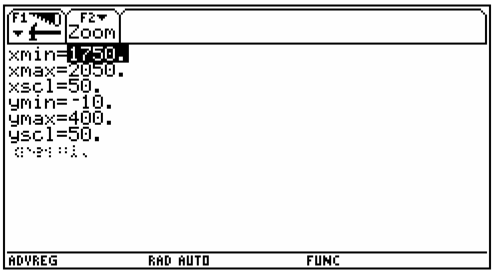
Finally I´d like to plot the pop-data together with the regression line (Josef)

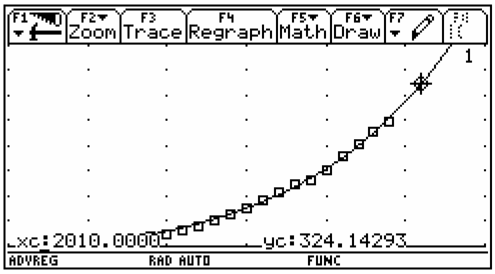






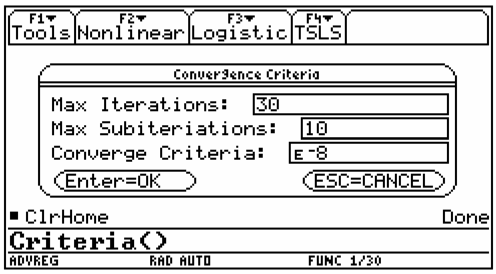






Option 8 under the F2 menu, MB(), is for “model building.” If you added one or more variables to the previous regression, MB() will compute the F statistic and probability associated with adding the variable(s).

Option 9 under the F2 menu allows you to change the convergence criteria by setting the maximum number of iterations and subiterations and the criteria for the percentage change in successive sum-of-squares values. The values I have set are 30, 10, and 10^{-8}.



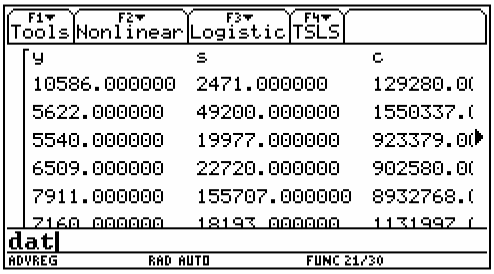
(NOTE: NonLin() may be used for linear regression also, that is, where the equation is linear in its parameters. There are no restrictions on the independent variables. They may be any differentiable function. For instance, if x is an independent variable, it may occur in the equation as x^2 or x^5, etc., or SIN(x), LN(x), EXP(x), etc. When using NonLin() for linear regression, you may set the initial guesses of the parameters equal to 1.)

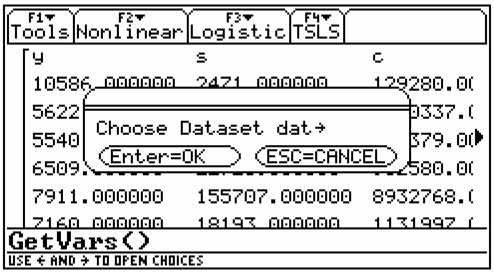
You can find more worked problems at the end of this article, Josef

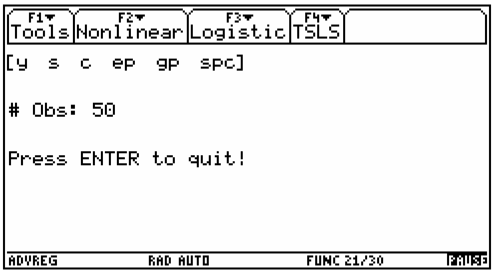
Example 2: Weighted Regression

The weight variable may be specifically included in the dataset, or it may simply be one of the independent variables already in the dataset. The weights are usually put in an n by n matrix along the diagonals. However, for large datasets such a matrix can be too big to be handled by the TI-89's limited memory. So I developed a routine to that uses the weights in a vector (list) instead of a matrix. For anyone interested the function is 'dmmul'.

The dataset "dat" is used for the weighted regression. You may use the GetVars() program under F1 to see the variable names and the number of observations in the dataset.



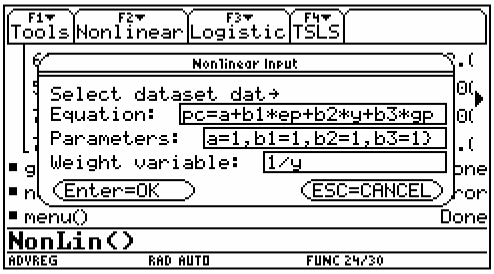


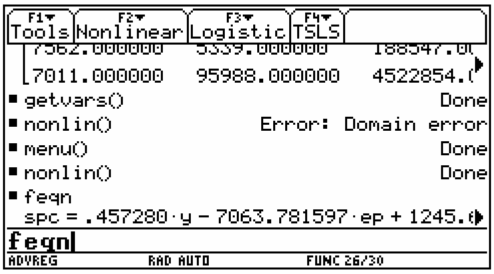


'spc' is the sales of electricity per customer in a state; 'y' is the per capita income in that state; 'ep' is the price per kilowatt hour; and 'gp' is the price of an amount of natural gas that that the energy equivalent of a kilowatt hour of electricity. The data is to be weighted by the inverse of the income per capita, 'y'.

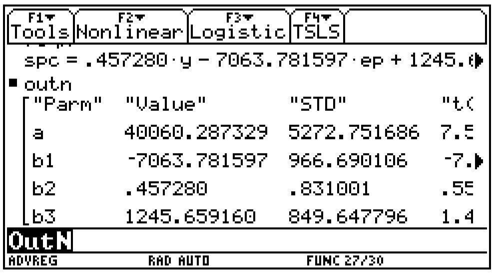
The equation we estimate is spc = a + b1*ep + b2*y + b3*gp with a weight of 1/y. Since this is a linear regression, enter the initial guesses as 1.

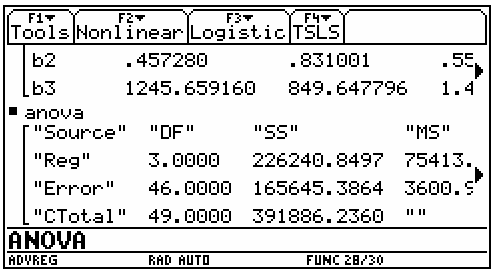
The fitted equation is: spc = 0.45728*y – 7063.7816*ep + 1245.65916*gp + 40060.28733.



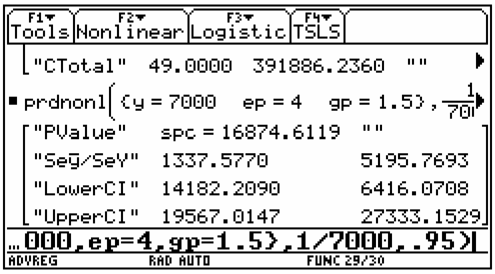


The parameters and their standard errors and the ANOVA table are:



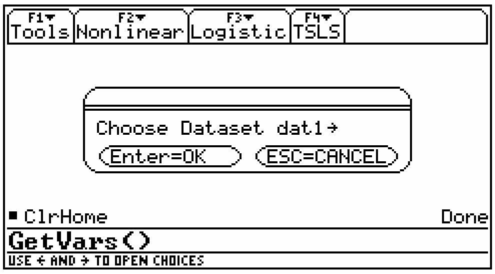


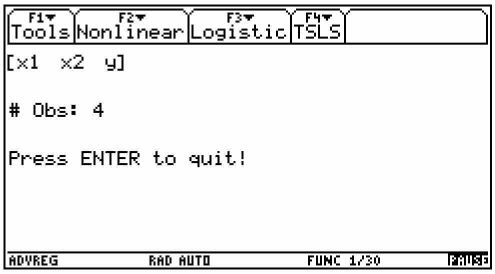
The predicted value for y = 7000, ep = 4, and gp = 1.5 is spc = 16874.6. Notice that the weight is entered as a number, i.e. 1/7000 for 1/y. The standard errors of the mean and predicted values are computed as well as the confidence interval for each.



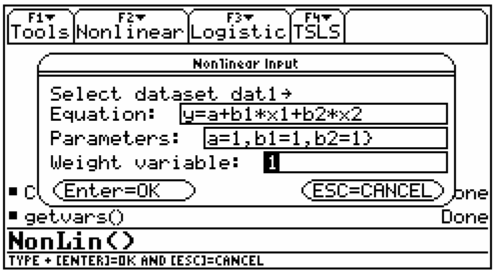
Example 3: Regression with Complex Data

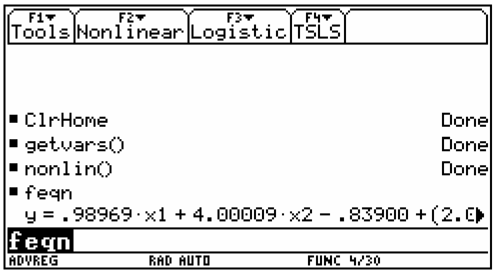
The complex dataset is “dat1” with variables x1, x2, and y. It has only 4 observations.





The input for the regression is: y = a + b1*x1 + b2*x2 with a weight of 1, meaning no weight. The initial guesses are all 1.





The fitted equation is:

$$y = 0.98969*x1 + 4.00009*x2 - 0.83900 + (2.02107*x1 + 2.93831*x2 - .59506)*\boldsymbol{i}$$

where ***i*** is the square root of -1.

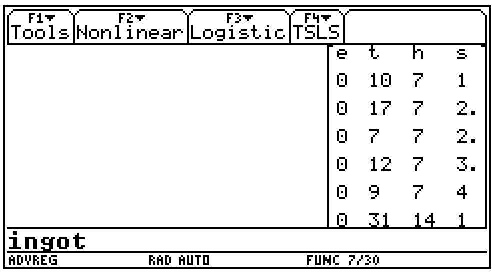
Logistic Regression

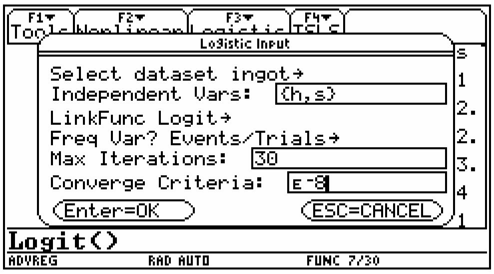
This program computes the logistic regression for binary dependent data using the logit, probit (normit), or complementary log-log link functions. Binary dependent data is in the form of 0s and 1s, where 1 signifies the occurrence of an event and 0 its nonoccurrence. The output is a regression equation that can be used to predict the probability of an event happening given a set of values for the independent variable(s).

The dataset may have a frequency variable or two variables denoting the results of a binomial experiment. The two variables are the number of successful events out of the total number of trials. If the dataset is from a binomial experiment, there is no dependent variable to enter; the program will create it from the events and trials data.

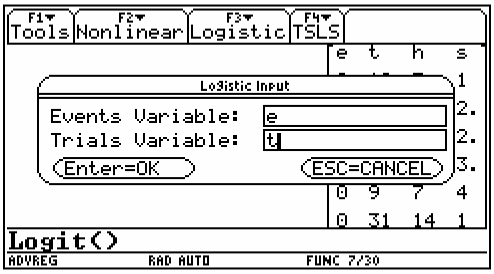
Example 4: Logistic Regression

The F3 menu is for computing the displaying the results of a logistic regression. Option 1, Logit(), is for entering the information and computing the regression. It sets up an input form. The first item is to select a dataset. Select “ingot” to be used in this example. Next, enter a list of the independent variables. The variables for this example are {h,s}. Next, you are prompted to enter the link function; use the logit link function. (The other link functions are the Probit and Complementary LogLog functions.) Next, you will be prompted for a frequency variable. The options are “No” for none, “Yes” for a frequency variable, and “Events/Trials” for a binomial experiment. Select “Events/Trials.” The last two options are to change the maximum number of iterations and convergence criteria. Leave them at 30 and E^{-8} for now. Press Enter to continue.





Having selected “Events/Trials”, you are now prompted to enter the events and trials variables. For this example they are e and t. Press Enter.

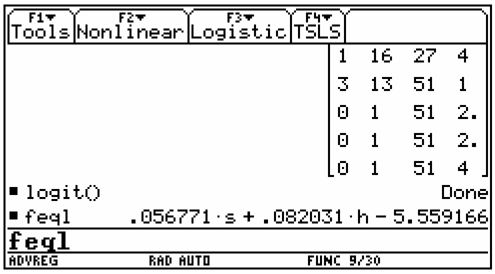


The program will now run and take several minutes to complete. A message will be displayed indicating whether the program was successful in estimating the regression.

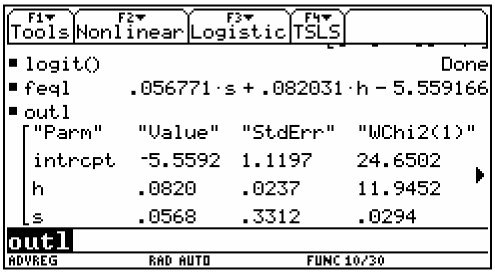
(If you had selected “Yes” for a frequency variable, you would have been prompted to enter the name of the frequency variable. Then you would be prompted to enter the name of the dependent binary variable. If you had selected “No” for frequency or binomial experiment variables, you would have been prompted to enter the name of the dependent binary variable.)

Option 2 under the F3 menu, FeqL, displays the fitted equation. For this example, it is:

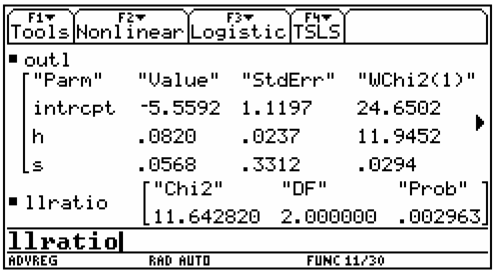
$$0.056771*s + 0.082031*h - 5.55917$$



Option 3, OutL, displays a matrix of the parameters, their values, standard errors, and the Wald chi-square statistics and probabilities.



"Parm"	"Value"	"StdErr"	"WChi2(1)"	"Prob(W)"
intercept	–5.55917	1.11969	24.6502	6.87383E^{-7}
h	0.082031	0.023734	11.9452	0.000548
s	0.056771	0.331213	0.029379	0.863906

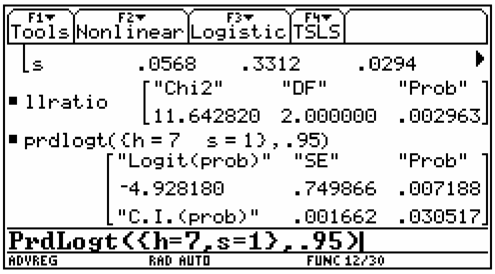


Option 4, LLRatio, displays the –2 log likelihood ratio that tests the significance of the covariates, that is, of the independent variables taken together. The output is:

"Chi2"	"DF"	"Prob"
11.6428	2	0.002963

Chi2 is the chi-square statistic, DF the degrees of freedom, and Prob the probability of obtaining that value by chance.

Option 5, PrdLogt, computes the logit (or probit or complementary log-log) of "p," where "p" is the probability of the event occurring given a set of values for the independent variables, the value of "p," and the confidence interval of "p." If h=7 and s=1, entering PrdLogt({h=7,s=1}, .95) produces:



"Logit(prob)"	"SE"	"Prob"
-4.92818	0.749866	0.007188
"C.I.(prob)"	0.001662	0.030517

Reading across, the value of logit(p) is –4.92818, the standard error is 0.74987, and the value of p is 0.007188. The 95 percent confidence interval around p is 0.00166 to 0.03052.

Two-Stage Least Squares

Two-stage least squares is the most widely used single-equation method for estimating simultaneous system of equations. Let **Y** be the endogenous or dependent variable in the system and **X** the exogenous or predetermined variables. The equations to be estimated are of the form:

$$\mathbf{y} = \mathbf{Y_1}*\boldsymbol{\beta} + \mathbf{X_1}*\boldsymbol{\zeta} + \mathbf{u}$$

$\mathbf{y}$ is an n by 1 vector of observations on the “dependent” variable.

$\mathbf{Y_1}$ is an n by g matrix of observations on the other endogenous variables included in the equation.

$\boldsymbol{\beta}$ is the g by 1 vector of coefficients associated with $\mathbf{Y_1}$.

$\mathbf{X_1}$ is the n by k matrix of observations on the predetermined or instrumental variables appearing in the equation.

γ is the k by 1 vector of coefficients associated with $\mathbf{X_1}$.

$\mathbf{u}$ is the n by 1 disturbances in the equation.

The problem of applying OLS to the above equation is that the variables in $\mathbf{Y_1}$ are correlated with $\mathbf{u}$. The essence of two-stage least squares regression is the replacement of $\mathbf{Y_1}$ by a computed matrix **Y_hat$_1$**, where hopefully the stochastic element is purged, and then performing an OLS regression of $\mathbf{y}$ on **Y_hat$_1$** and $\mathbf{X_1}$.

The matrix **Y_hat$_1$** is computed in the first stage by regressing each variable in $\mathbf{Y_1}$ on all the instrumental variables in the complete model and replacing the actual observations on the **Y** variables by the corresponding regression values. Thus,

$$\mathbf{Y\_hat_1} = \mathbf{X}*(\mathbf{X}^T*\mathbf{X})^{-1}*\mathbf{X}^T*\mathbf{Y_1}$$

where $\mathbf{X} = [\mathbf{X_1}\ \ \mathbf{X_2}]$. **X** is the n by k matrix of observations on all the instrumental variables in the complete model, $\mathbf{X_2}$ being the matrix of observations on those instrumental variables excluded from the equation under study.

In the second stage $\mathbf{y}$ is regressed on **Y_hat$_1$** and $\mathbf{X_1}$. The equation for the 2SLS estimates can then be written as:

$$\begin{bmatrix} Y_1^{\mathrm{T}} * X * (X^T * X)^{-1} * X^T * Y_1 & Y_1^T * X_1 \\ X_1^T * Y_1 & X_1^T * X_1 \end{bmatrix} * \begin{bmatrix} b \\ c \end{bmatrix} = \begin{bmatrix} Y_1^T * X * (X^T * X)^{-1} * X^T * y \\ X_1^T * y \end{bmatrix}$$

where $\mathbf{b}$ is the vector of coefficients on the other endogenous variables and $\mathbf{c}$ is the vector of coefficients on the predetermined variables in the equation, including the intercept if any.

The following two examples are models from political economics:

Example 5:

The data used in this example are for a simplified model designed to explain variations in the consumption and price of food. The data are from Kmenta, pp. 563-65.

The variables are:

q = food consumption per head

p = ratio of food prices to general consumer prices

d = disposable income in constant prices

f = ratio of preceding year's prices received by farmers to general consumer prices

a = time in years

The endogenous (dependent) variables are q and p. The exogenous (independent) variables are d, f, and a.

Estimate the following equations:

$q = \gamma 0 + \beta 1 * p + \gamma 1 * d$ (the demand equation)

$q = \gamma 0 + \beta 1 * p + \gamma 1 * f + \gamma 2 * a$ (the supply equation)

Press F4: 1 (TSLS) ENTER to begin the program. Select the "kmenta" dataset from the pull down menu.

In the equation box enter q = p + d.

(Note: you do not enter the coefficients for the equation. That is done by the program.)

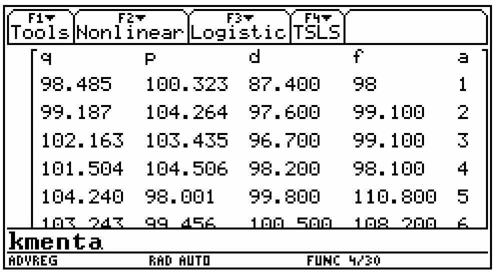
In the Endog. Vars. box enter in a list the dependent variables in the dataset: {q,p}.

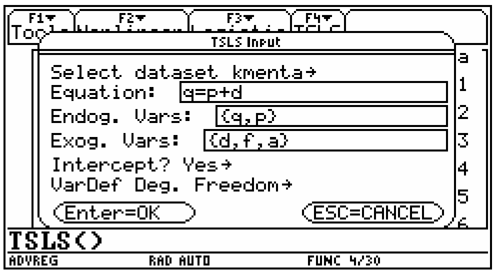
In the Exog. Vars. box enter in a list the independent variables: {d,f,a}.

From the Intercept? pull down menu select Yes for an intercept.

(Note: According to the SAS statistical software, if the intercept is set to No, the definition of the R^2 statistic for two-stage least squares is changed to 1 – (Residual Sum of Squares/Uncorrected Total Sum of Squares.)

From the VarDef pull down menu select Deg. Freedom with which to calculate the variances. (The other option is to select # Obs. for number of observations.) The input form looks like this:

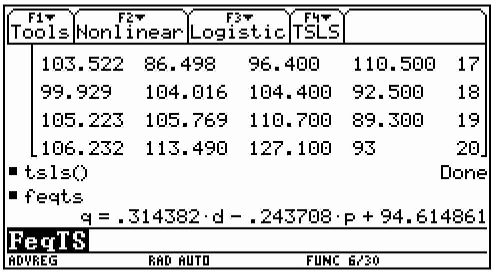




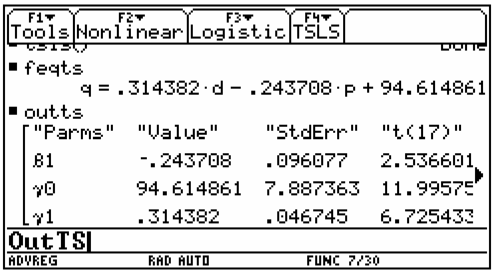
Press ENTER to begin the program.

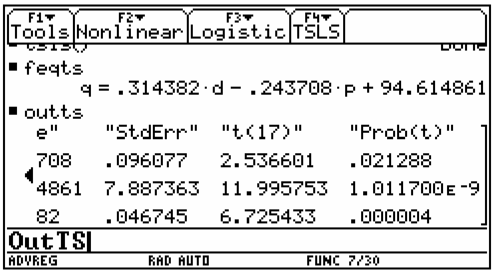
Press F4: 2 (FeqTS) to display the fitted equation.

It is: q = 0.314382*d - 0.243708*p + 94.614861

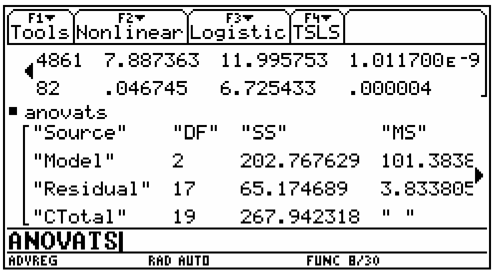


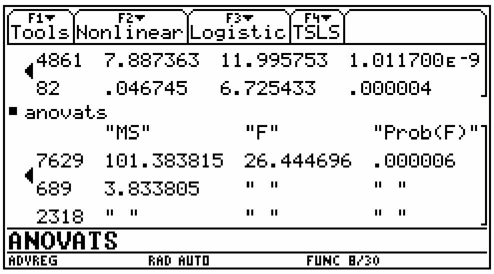
Press F4: 3 (OutTS) to display the parameters, their values, standard errors, t-values, and the t probabilities.



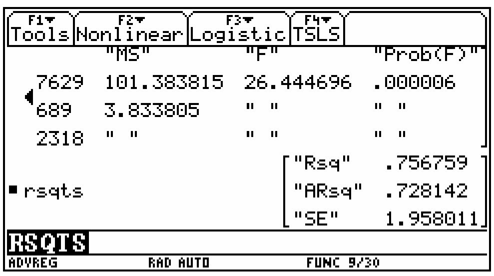


Press F4: 4 (ANOVATS) to display the analysis of variance table.

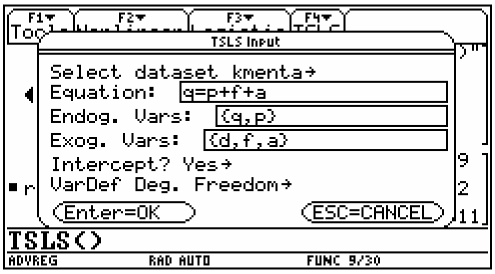


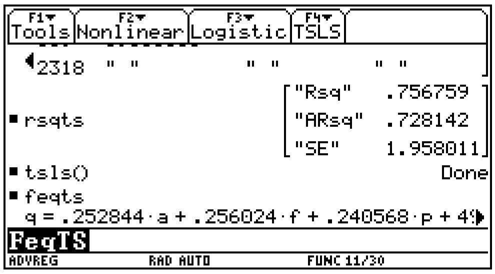


Press F4: 5 to display the R^2, adjR^2, and SE stats.



To compute the other equation run the TSLS program again and just change the equation to $q = p + f + a$. Everything else remains the same.





The fitted equation is $q = 0.252844*a + 0.256024*f + 0.240568*p + 49.448206$.

The other statistics can be displayed as shown above.

Example 6

This example is based on Klein's model 1 (1950). The endogenous variables are **c**, **p**, **w**, **I**, **x**, **wsum**, **k**, and **y**. The exogenous variables are **klag**, **plag**, **xlag**, **wp**, **g**, **t**, and **yr**.

yr = year – 1931

c = consumption

p = profits

w = private wage bill

I = investment

x = private production

wp = government wage bill

g = government demand

t = taxes

k = capital stock

wsum = total wage bill

plag = profits lagged

xlag = private product lagged

klap = capitol stock lagged

$y = c + i + g - t$ (national income)

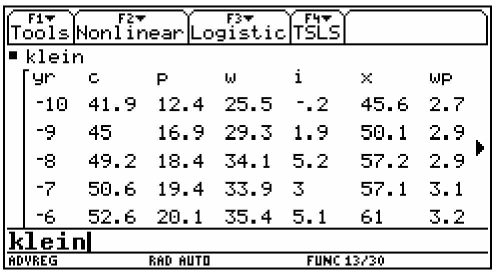
Estimate the following equations:

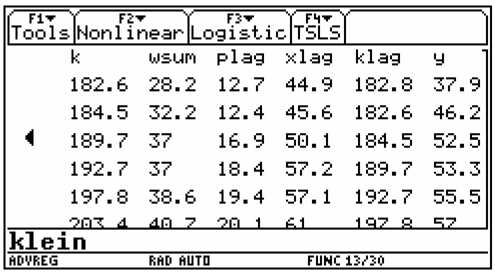
$$c = p + plag + wsum$$

$$i = p + plag + wsum$$

$$w = x + xlag + yr$$

Initiate the TSLS program and select the klein dataset.

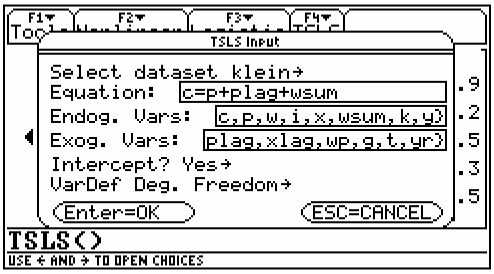


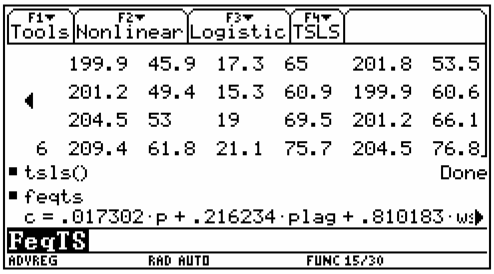


Enter the first equation. For the endogenous variables enter {c, p, w, i, x, wsum, k, y}.

For the exogenous variables enter {klag, plag, xlag, wp, g, t, yr}.

Select Yes for the intercept and Deg. Freedom for the variance definition.

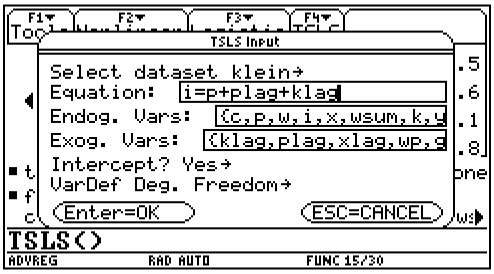


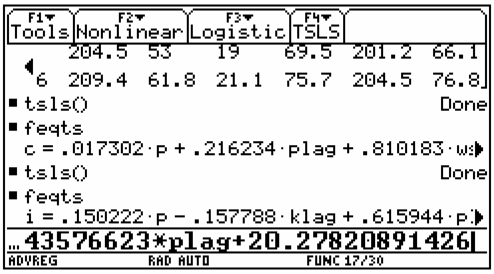


The fitted equation is:

$$c = 0.017302*p + 0.216234*plag + 0.810183*wsum + 16.554756.$$

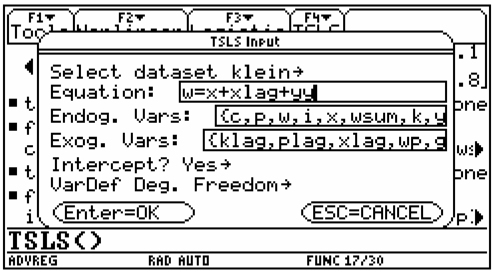
To compute the other two equations just change the equation in the TSLS input form.

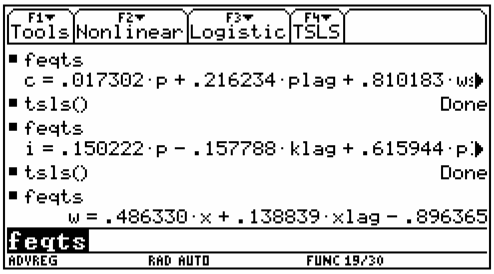




The second fitted equation is:

$$I = 0.150222*p - 0.157788*klag + 0.615944*plag + 20.278209$$





And the third fitted equation is:

$$w = 0.438859*x + 0.146674*xlag + 0.130396*yr + 1.500297$$

(Note: Once the programs have been run at least once, all programs and functions in the AdvStat folder may be archived. DO NOT archive anything else in the folder. The datasets are archived by the AddDS() program.)

I hope you find the programs useful and enjoyable. I had fun programming them. I have also programmed these routines for DERIVE 6.1. If you would like them, just drop me an email.

Any comments, suggestions, frustrations with the programs? If so, just drop me an email.

References:

J. Johnson, *Econometric Methods*, 2 ed., McGraw-Hill Book Co., New York, 1972.

Jan Kmenta, *Elements of Econometrics*, MacMillian Publishing Co., New York, 1971.

L. Klein, *Economic Fluctuations in The United States: 1921-1940*, John Wileys and Sons, New York, 1950.

J.K. Binkley and Nelson, G. (1984), “Impact of Alterative Degrees of Freedom Corrections in Two and Three Stage Least Squares,” *Journal of Econometrics*, 24, 3, 223-233.

Some additional examples (Josef):

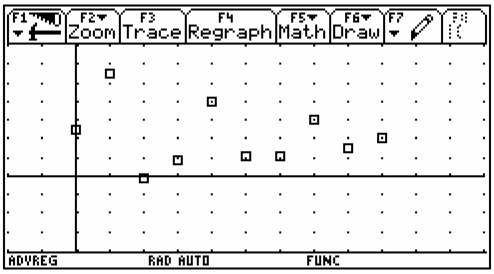
The following data set is given:

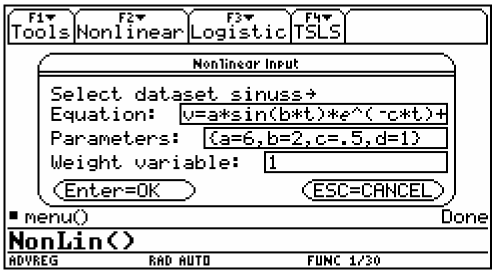
t	0	1	2	3	4	5	6	7	8	9
v	2.5	5.5	-0.1	0.9	4	1.1	1.1	3	1.5	2

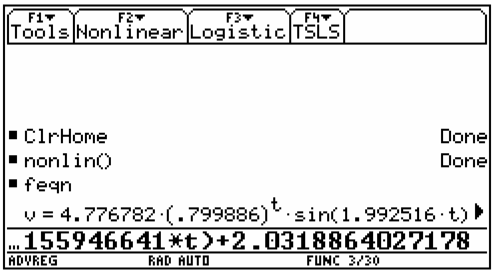
I plot the scatter diagram and assume that a damped oscillation – vertically shifted – might give an appropriate fit.

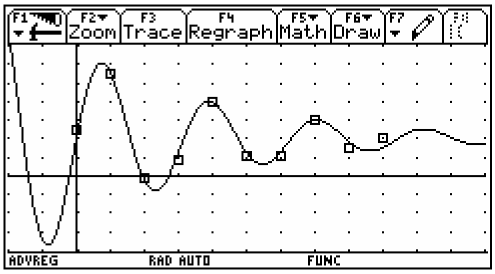
$$v = a \cdot \sin(b\, t) \cdot e^{-c \cdot t} + d.$$

My initial guesses are: a = 6, b = 2, c = 0.5 and d = 1.









Plotting the fit function shows a very satisfying result.

Next is an example from our textbook "*Mathe mit Gewinn*" (= "*Mathematics with Profit*")[1] vol 4.

Given is the distribution of ages of football players of English Premier League with a transfer sum above 1 Million Pounds.

Beispiel: Einer Zeitschrift entnehmen wir die Altersverteilung der Spieler bei den Transfers in der britischen Premier League mit Transfersummen über 1 Million Pfund:

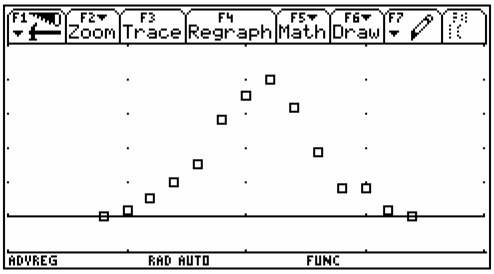
Alter	19	20	21	22	23	24	25	26	27	28	29	30	31	32
% der Transfers	0	1	2,5	5	7,5	14	17,5	20	16	9,5	4	2	1	0

a) Suche eine geeignete Verteilungskurve und

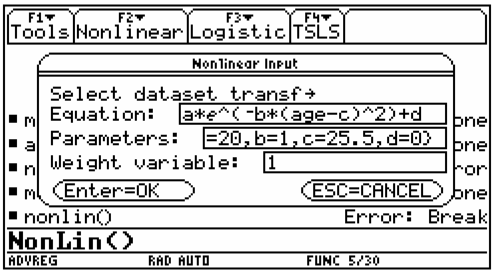
b) suche eine Kurve für die kumulierten Werte (% der Transfers bis zum Alter von *x* Jahren).

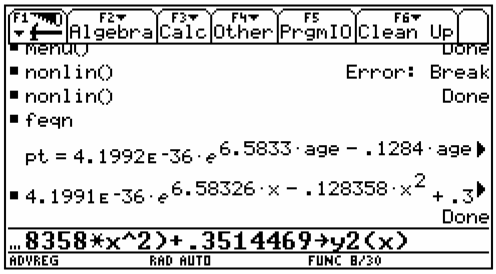
[1] H.D. Hinkelmann a.o., Mathe mit Gewinn 1-4, hpt Vienna

The problem is to find an appropriate distribution curve and an curve for the cumulated values (% of the transfers up to an age of x years)

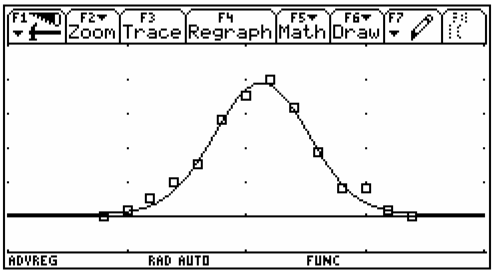


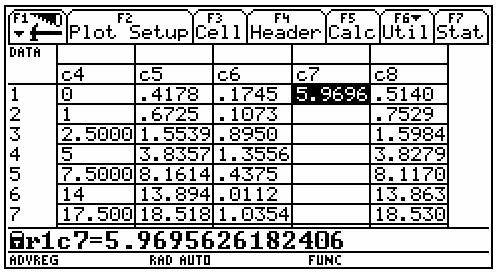
The form of the scatter diagram gives the idea to try with a bell shaped curve (normal distribution). See the equation and my guesses for the parameters:



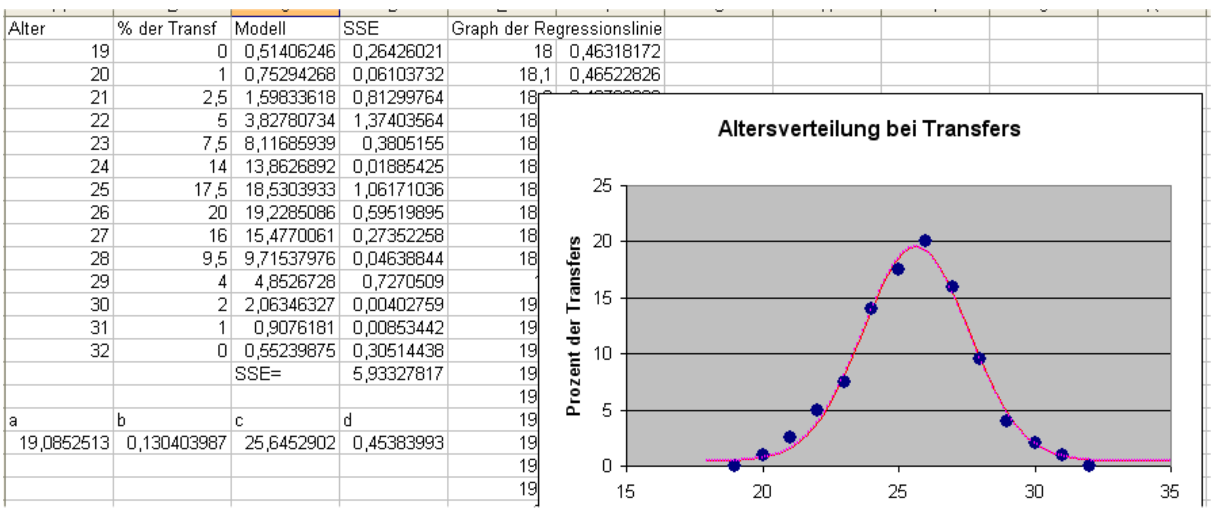


The first attempt is successful.





In our textbook we use the Solver of Excel in order to find the distribution curve. Compare the results.



Alter	% der Transf	Modell	SSE	Graph der Regressionslinie	
19	0	0,51406246	0,26426021	18	0,46318172
20	1	0,75294268	0,06103732	18,1	0,46522826
21	2,5	1,59833618	0,81299764	18	
22	5	3,82780734	1,37403564	18	
23	7,5	8,11685939	0,3805155	18	
24	14	13,8626892	0,01885425	18	
25	17,5	18,5303933	1,06171036	18	
26	20	19,2285086	0,59519895	18	
27	16	15,4770061	0,27352258	18	
28	9,5	9,71537976	0,04638844	18	
29	4	4,8526728	0,7270509		
30	2	2,06346327	0,00402759	19	
31	1	0,9076181	0,00853442	19	
32	0	0,55239875	0,30514438	19	
		SSE=	5,93327817	19	
				19	
a	b	c	d	19	
19,0852513	0,130403987	25,6452902	0,45383993	19	
				19	
				19	

See next pages: Don´s tools provided for DERIVE and TI-NSpireCAS

Some time ago Don provided a DERIVE utility for doing non linear regression. Two numerical methods are well known for solving minimization problems with several variables: Gauss-Newton- and Levenberg-Marquardt-Algorithm.

This is the pop-problem from page 23 treated with DERIVE:

#9: GAUSS_NEWTON(p = c·NORMAL(a + b·(y − 1790)), [a, b, c], [−2.4, 0.012, 400], pop)

#11: $feq = (p = 203.55302\cdot(ERF(4.9268136\cdot10^{-12}\cdot(1.8124180\cdot10^{9}\cdot y - 3.5747336\cdot10^{12})) + 1))$

#12: $GAUSS\_NEWTON(p = c\cdot e^{b\cdot(y - 1790)} + a, [a, b, c], [10, 0.03, 20], pop)$

#13:

Gauss_Newton Method

Convergence criteria met!

Parm	Value	STD	t(16)	Prob(t)
a	−31.280723	5.6498704	−5.5365382	0.99997744
b	0.011526649	0.00066099698	17.438278	0
c	29.557374	3.9832616	7.4203948	$7.2532831\cdot10^{-7}$

Source	DF	SS	MS	F	Prob(F)
Reg	2	$7.1682365\cdot10^{4}$	$3.5841182\cdot10^{4}$	2385.4704	0
Error	16	240.39657	15.024786		
Total	18	$7.1922761\cdot10^{4}$			

SE	R^2	AdjR^2
3.8761818	0.99665757	0.99623976

$$p = 3.2359135\cdot10^{-8}\cdot e^{0.011526649\cdot y} - 31.280723$$

#14: Residuals =

1	−0.47677616
2	−0.17050976
3	0.31662513
4	−0.15855545
5	−0.22490851
6	−0.013728937
7	0.32515872

#15: iter =

Iter	a	b	c	SSE
0	10	0.03	20	$3.8879337\cdot10^{7}$
1	2.4600075	0.028868887	4.9418825	$9.5103350\cdot10^{5}$
2	0.74364661	0.023658055	5.7144443	$7.6785286\cdot10^{4}$
3	−8.3537131	0.016006799	10.798732	5181.2360
4	−25.058218	0.010389344	23.693736	$2.1008407\cdot10^{4}$
4.01	−16.705965	0.013198071	17.246234	6507.4962

#16: [Predicted_Values([2010]), Predicted_Values([2020])]

#17:

95% Confidence Interval		
Value	341.94977	
Se_yhat/Se_Y0	10.463537	11.158423
CI_yhat	323.68166	360.21787
CI_Y0	322.46847	361.43106

95% Confidence Interval		
Value	387.54825	
Se_yhat/Se_Y0	13.764359	14.299733
CI_yhat	363.51730	411.57920
CI_Y0	362.58260	412.51390

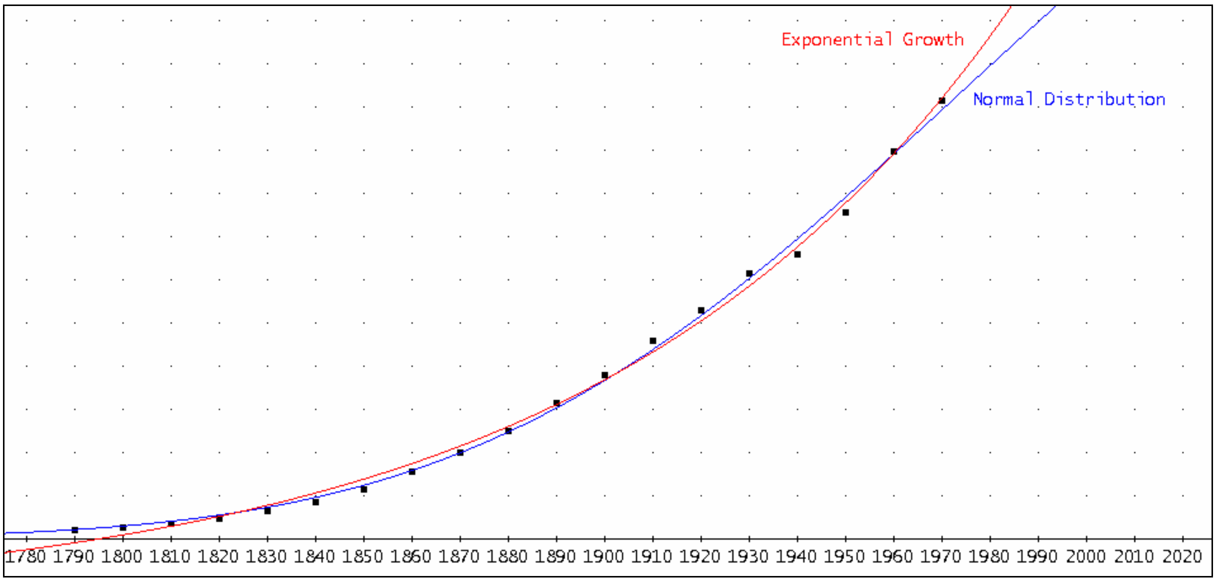
#18: anova =

Source	DF	SS	MS	F	Prob(F)
Reg	2	$7.1682365\cdot10^{4}$	$3.5841182\cdot10^{4}$	2385.4704	0
Error	16	240.39657	15.024786		
Total	18	$7.1922761\cdot10^{4}$			

#19:
REST([pop↓↓2, pop↓↓1]')

$$203.55302\cdot(\mathrm{ERF}(4.9268136\cdot10^{-12}\cdot(1.812418\cdot10^{9}\cdot x - 3.5747336\cdot10^{12})) + 1)$$

$$3.2359135\cdot10^{-8}\cdot e^{0.011526649\cdot x} - 31.280723$$



Applying the Marquardt-Levenberg algorithm gives the same result:

#20: $\mathrm{MARQUARDT}(p = c\cdot e^{b\cdot(y - 1790)} + a,\ [a, b, c],\ [10, 0.03, 20],\ pop)$

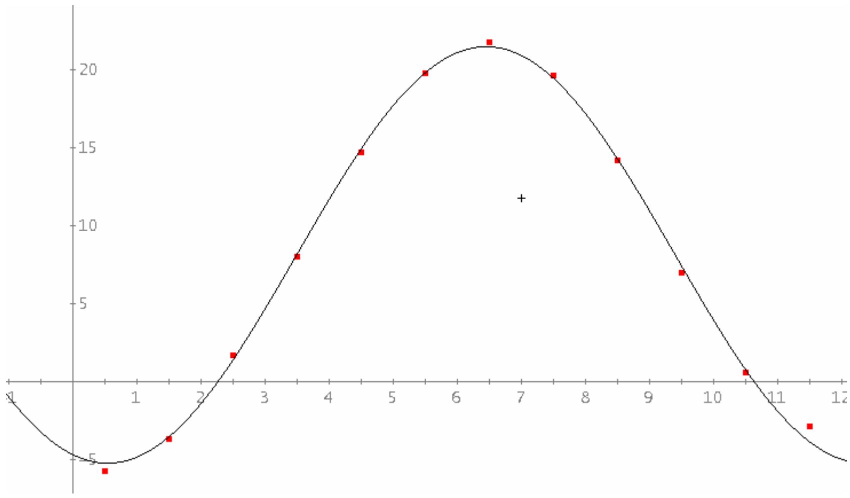
#21: $feq = (p = 3.2358222\cdot10^{-8}\cdot e^{0.011526663\cdot y} - 31.280608)$

Next example tries a trigonometric fit of temperature values.

#22: $\text{temp} := \begin{bmatrix} x & 0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 & 9.5 & 10.5 & 11.5 \\ y & -5.7 & -3.7 & 1.7 & 8 & 14.7 & 19.8 & 21.8 & 19.6 & 14.2 & 7 & 0.6 & -2.9 \end{bmatrix}$,

#23: $\text{GAUSS\_NEWTON}(y = a\cdot \text{SIN}(b\cdot x + c) + d, [a, b, c, d], [13.5, 1, 1, 8.5], \text{temp})$

#24: $\text{feq} = (y = 13.353080\cdot \text{SIN}(0.53176799\cdot x + 4.4243910) + 8.1265919)$



Marquardt-Levenberg did not work!

Sometimes it is necessary to try some different guesses of the initial values for the parameter in order to receive a satisfying answer: Once more the age-transfer model from page 32:

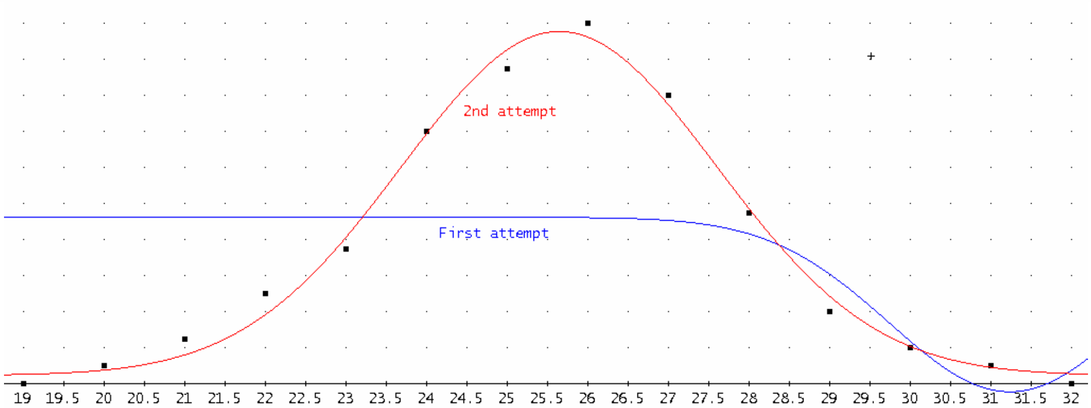
#34: $\text{transf} := \begin{bmatrix} a & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \\ t & 0 & 1 & 2.5 & 5 & 7.5 & 14 & 17.5 & 20 & 16 & 9.5 & 4 & 2 & 1 & 0 \end{bmatrix}$,

#35: $\text{GAUSS\_NEWTON}\left(t = b0\cdot e^{-b1\cdot(a - b2)^2} + b3, [b0, b1, b2, b3], [20, 1, 30, 1], \text{transf}\right)$

#36: $\text{feq} = 9.2273592 - 5.0942497\cdot 10^{-95}\cdot e^{14.047781\cdot a - 0.22487615\cdot a^2}$

#37: $\text{GAUSS\_NEWTON}\left(t = b0\cdot e^{-b1\cdot(a - b2)^2} + b3, [b0, b1, b2, b3], [20, 0.1, 30, 0.1], \text{transf}\right)$

#38: $\text{feq} = 1.0808899\cdot 10^{-36}\cdot e^{6.6884918\cdot a - 0.13040384\cdot a^2} + 0.45382339$

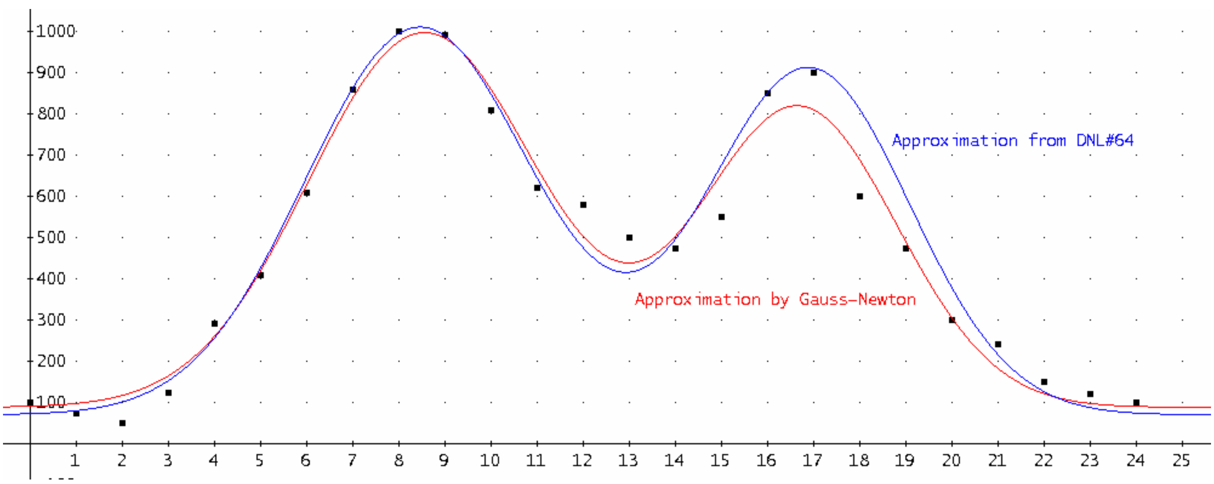


This is a more complicated function to approximate. In DNL#64 I presented a “Traffic Density Application”. In DNL#64 I used the slider bars to find a function composed of two bell shaped functions. Now let´s try with Don´s tools and compare the results:

#45: $\text{tabvz} := \begin{bmatrix} h\_ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\ d\_ & 100 & 75 & 50 & 125 & 290 & 410 & 610 & 860 & 1000 & 990 & 810 & 620 & 580 & 500 & 475 & 550 & 850 & 900 & 600 & 475 & 300 & 240 & 150 & 120 & 100 \end{bmatrix},$

#46: $\text{GAUSS\_NEWTON}\left(d\_ = b0 \cdot e^{-((h\_ - b1)/b2)^2} + b3 \cdot e^{-((h\_ - b4)/b5)^2} + b6,\ [b0, b1, b2, b3, b4, b5, b6],\ [1000, 9, 3, 900, 17, 3, 1],\ \text{tabvz}\right)$

#47: $\text{feq} = \left(d\_ = 2.5550863 \cdot e^{1.374721 \cdot h\_ - 0.08044227 \cdot h\_^2} + 4.6785709 \cdot 10^{-11} \cdot e^{3.643272 \cdot h\_ - 0.10924957 \cdot h\_^2} + 88.332324\right)$



MARQUARDT delivers the same result as Gauss-Newton!

Comparing the Sum of Squared Errors (SSE) gives $6.0314 \cdot 10^4$ for Gauss-Newton and $1.1293 \cdot 10^5$ for the manually performed “Slider Bar Approximation”.

Don provides the Two Stage Approximation for DERIVE, too This is the documentation followed by the two TI-examples which are shown above:

```
The program
          TwoStage(eq_,endog_,exog_,data_,incept_:=1,vardef_:=1)
computes the 2SLS regression coefficients, standard errors of the coeffi-
cients, t values and their probabilities, the ANOVA table, root MSE,
R_square, and adjusted R_square.

The inputs are:

eq_:     the equation to be solved for
endog_:  a vector of the endogenous variables
exog_:   a vector of the exogenous variables
data_:   the name of the dataset where the first row of the data matrix
contains the names of the variables
incept_: the default of 1 indicates that an intercept is to be computed
for the equation; set incept:=0 if you do not want an intercept.
vardef_: the default of 1 sets the variance denominator to the degrees of
freedom; changing vardef_ to 0 sets the denominator to the number of ob-
servations.

Note: According to SAS, if the no intercept option is set (incept_=0),
the definition of the R_square statistic for two-stage least squares is
changed to 1 - (Residal Sum of Squares/Uncorrected Total Sum of Squares).
```

In addition, the program

`Model(mm_,endog_,exog_,data_,incept_:=1,vardef_:=1)`

solves for all the of equations at once. `mm` is a matrix of the equations with one equation per row.

$$\#9:\quad \text{Model}\left(\begin{bmatrix} q = p + d \\ q = p + f + a \end{bmatrix}, [q, p], [d, f, a], \text{data}\right)$$

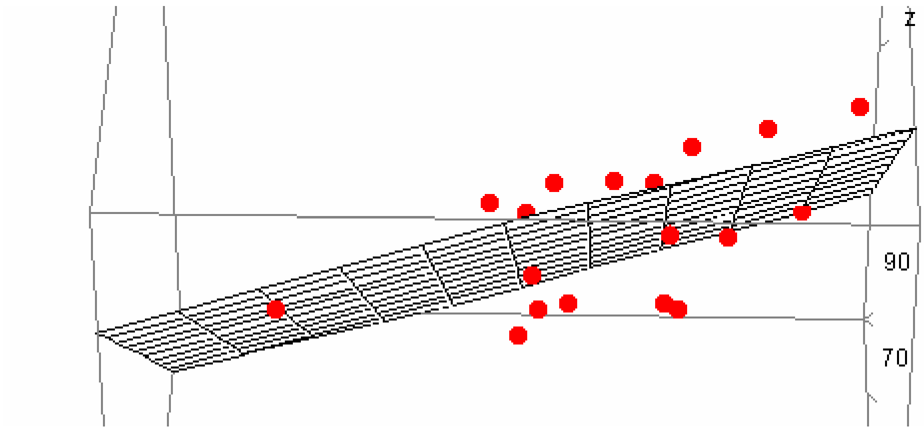
$q = -\ 0.2437076898\cdot p + 0.3143821013\cdot d + 94.61485336$

$q = 0.240567722\cdot p + 0.2528436233\cdot a + 0.2560236611\cdot f + 49.44820072$

I tried to produce a 3D presentation of the q(p,d) model. I am satisfied with the result. (Josef)

plot := VECTOR([pt], pt, REST([data↓↓2, data↓↓3, data↓↓1]'))

$z = -\ 0.2437076898\cdot x + 0.3143821013\cdot y + 94.61485336$



And this is the second example from above performed with DERIVE (with and without inter-cepts):

$$\text{Model}\left(\begin{bmatrix} c = p + plag + wsum \\ i = p + plag + klag \\ w = x + xlag + yr \end{bmatrix}, [c, p, w, i, x, wsum, k, y], [klag, plag, xlag, wp, g, t, yr], klein\right)$$

$c = 0.01730294013\cdot p + 0.8101827436\cdot wsum + 0.2162334038\cdot plag + 16.55475199$

$i = 0.1502217351\cdot p + 0.6159436535\cdot plag - 0.1577876492\cdot klag + 20.27821173$

$w = 0.438858637\cdot x + 0.1466742262\cdot xlag + 0.1303957913\cdot yr + 1.500299135$

$$\text{Model}\left(\begin{bmatrix} c = p + plag + wsum \\ i = p + plag + klag \\ w = x + xlag + yr \end{bmatrix}, [c, p, w, i, x, wsum, k, y], [klag, plag, xlag, wp, g, t, yr], klein, 0\right)$$

$c = 0.1583375706\cdot p + 1.121162681\cdot wsum + 0.2651214661\cdot plag$

$i = 0.4459657099\cdot p + 0.3685120348\cdot plag - 0.06148091416\cdot klag$

$w = 0.4444584525\cdot x + 0.166299164\cdot xlag + 0.1129951702\cdot yr$

Don has no problems to transfer his programs from DERIVE to the TI-handhelds (V200 & TI-89) but also to TI-NspireCAS. File Phillps_AdvReg.tns for TI-NspireCAS provides Nonlinear Regression for TI-Nspire. I print two screenshots because this might be of special interest for us teachers when showing the students how to fit a power function by transforming it to a linear function. We know that this is not quite correct because the linear regression is applied on transformed data. This is what Don has to tell - and to demonstrate - us:

Linear vs. Nonlinear Regression

In some cases a nonlinear function can be linearized through a transformation of the data. While this usually gives reasonable results, sometimes it may not. The reason is that a linear regression with transformed data does not minimize the sum of squared errors of the non-transformed data. The data in spreadsheet 1.8 is from a project fitting a power curve to the data. The curve was originally estimated with Excel.

gaussnewton$\left(\ln(r\_)=a+b\cdot\ln(m\_),\{a=1,b=1\},1,30,10,10^{-8}\right)$

This is a natural log transformation of the equation r_=a*m_^b. The output is in 1.9. The fitted equation is solved for r_, and a=18.319 and b=−0.3535. This is also the result that the TI−Nspire produces for the Power Regression (see 1.8). In other words, it is not a true power regression; it is a linear regression of a log transformed equation and data.

The untransformed equation is also fit.

gaussnewton$\left(r\_=a\cdot m\_^{b},\{a=18,b=-0.35\},1,30,10,10^{-8}\right)$

Here a =57.821 and b=−0.52927. The graphs in 1.10 show both equations. It is up to the analyst, of course, to determine which fit is best for her purposes.