# Optimizing the sheep pen

## Introduction

This activity is about **optimization** – finding the best (or at least a good) solution to a problem that requires a function to be maximised. The particular problem tackled here is to maximise the area of an isosceles triangle. TI-Nspire offers a wide range of possible approaches to such problems and teachers who work through the following account will find a wealth of innovative ideas that can be adapted for use in the classroom.

This account was first published in *Inspiring Maths in the classroom* by Alison Clark-Wilson and Adrian Oldknow. The full text of that booklet is available for download from http://education.ti.com/educationportal/sites/UK/nonProductMulti/case_study.html?bid=2.

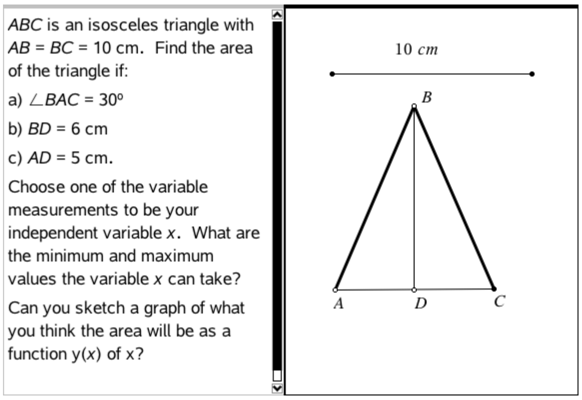
## Resources

These notes, written primarily for teachers, are accompanied by a TI-Nspire document, SheepPen.tns. This includes all the pages and that are referred to and pictured in the text. This file is best used on TI-Nspire software, rather than on the handheld.

## The problem

A Northumberland sheep farmer has two pieces of fencing, each 10m long, which she wants to fix to a straight stone wall to make a triangular sheep pen whose area is as big as possible. We can make a "scale model" by using, say, 1cm to represent 1m.

We will look at a variety of different ways using *TI-Nspire* in which we can approach such a problem: numeric, graphic, symbolic, geometric, statistic, algorithmic and synoptic.



*ABC* is an isosceles triangle with *AB* = *BC* = 10 cm. Find the area of the triangle if:

a) ∠*BAC* = 30º

b) *BD* = 6 cm

c) *AD* = 5 cm.

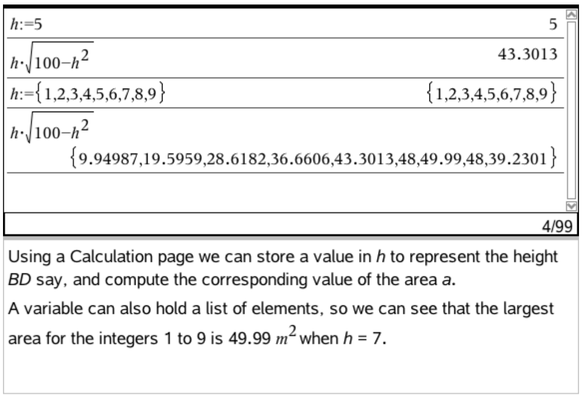
Choose one of the variable measurements to be your independent variable *x*. What are the minimum and maximum values the variable *x* can take?

Can you sketch a graph of what you think the area will be as a function y(*x*) of x?

## Seven approaches

### Numeric

For the numeric approach we will illustrate the uses of both Calculator and Lists & Spreadsheet pages, as well as storing values in variables.



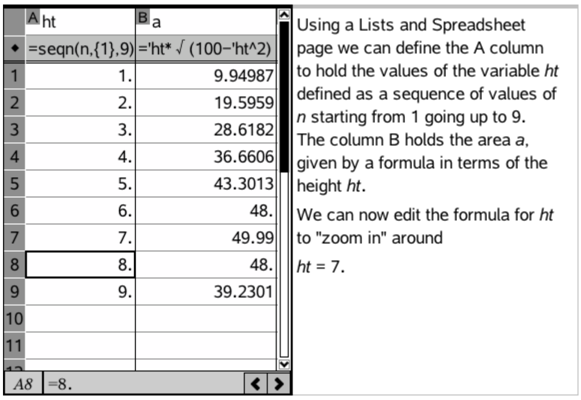
```
h:=5                                                        5
h·√(100-h²)                                           43.3013
h:={1,2,3,4,5,6,7,8,9}                    {1,2,3,4,5,6,7,8,9}
h·√(100-h²)
      {9.94987,19.5959,28.6182,36.6606,43.3013,48,49.99,48,39.2301}
```

4/99

Using a Calculation page we can store a value in *h* to represent the height *BD* say, and compute the corresponding value of the area *a*.

A variable can also hold a list of elements, so we can see that the largest area for the integers 1 to 9 is 49.99 $m^2$ when *h* = 7.

In many cases it is more helpful to use a Lists & Spreadsheet page as this will cause values to be recomputed when a value or formula is changed. We define the column A to hold the variable *ht* as a list defined to be the sequence of numbers *n* starting from *n*=1 and going on for 9 terms.



| | A ht | B a |
|---|---|---|
| ◆ | =seqn(n,{1},9) | ='ht* √ (100−'ht^2) |
| 1 | 1. | 9.94987 |
| 2 | 2. | 19.5959 |
| 3 | 3. | 28.6182 |
| 4 | 4. | 36.6606 |
| 5 | 5. | 43.3013 |
| 6 | 6. | 48. |
| 7 | 7. | 49.99 |
| 8 | 8. | 48. |
| 9 | 9. | 39.2301 |
| 10 | | |
| 11 | | |

A8 =8.

Using a Lists and Spreadsheet page we can define the A column to hold the values of the variable *ht* defined as a sequence of values of *n* starting from 1 going up to 9. The column B holds the area *a*, given by a formula in terms of the height *ht*.

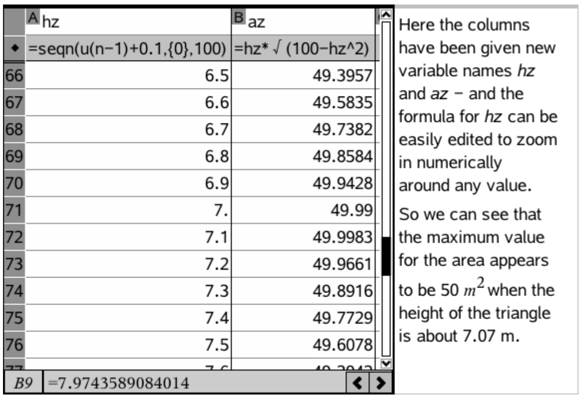
We can now edit the formula for *ht* to "zoom in" around *ht* = 7.

If we make a little change in the definition of the variable in column A we can make it much easier to "zoom in" on any particular value that seems interesting.

Here the variable *hz* is defined by the recurrence formula:

u(*n*) = u(*n*-1) + 0.01
with u(1) = 7 and *n*=2,3,4,....,10

Column B now holds the list of values of the variable *az* defined in terms of *hz*.



| | A hz | B az |
|---|---|---|
| ◆ | =seqn(u(n−1)+0.1,{0},100) | =hz* √ (100−hz^2) |
| 66 | 6.5 | 49.3957 |
| 67 | 6.6 | 49.5835 |
| 68 | 6.7 | 49.7382 |
| 69 | 6.8 | 49.8584 |
| 70 | 6.9 | 49.9428 |
| 71 | 7. | 49.99 |
| 72 | 7.1 | 49.9983 |
| 73 | 7.2 | 49.9661 |
| 74 | 7.3 | 49.8916 |
| 75 | 7.4 | 49.7729 |
| 76 | 7.5 | 49.6078 |

B9 =7.9743589084014

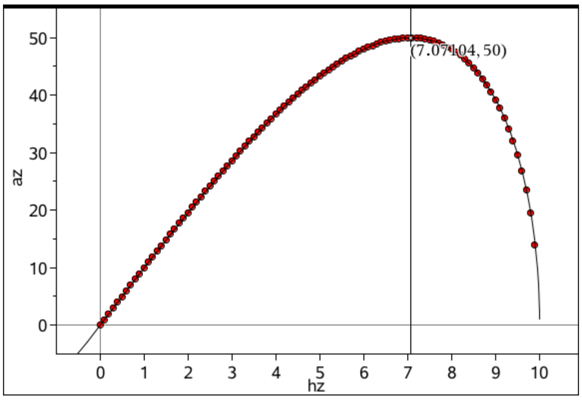
Here the columns have been given new variable names *hz* and *az* − and the formula for *hz* can be easily edited to zoom in numerically around any value.

So we can see that the maximum value for the area appears to be 50 $m^2$ when the height of the triangle is about 7.07 m.

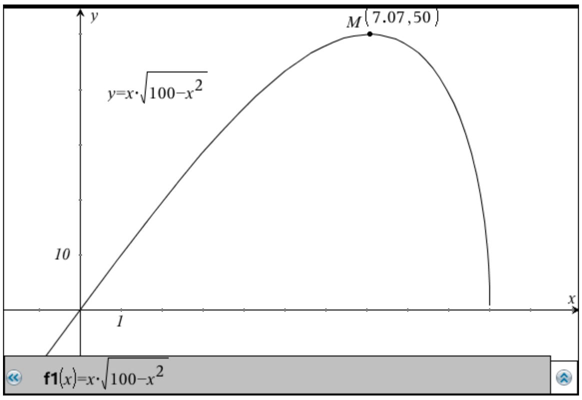
## Graphic

We can generate a scatterplot from the numerical data in any pair of lists of the same length. These can be generated in either a Calculator or a Lists & Spreadsheet page, but we'll need a lot more data points than just 10.

Here we use a Data & Statistics page to draw the scattergraph of the list ***hz*** extended to 100 values.



***hz***:=seqn(u(*n*-1)+0.1,{0},100) is the horizontal variable and the list ***az***:=***hz****√(100-***hz***$^2$) is the vertical variable.
In the Data & Statistics page there is an Analyze icon which brings down a menu including Plot Function – so we can draw the graph of f1(*x*) = *x**√(100-*x*$^2$) and show, of course, that it goes through all the data points.



So that leads us into the main way of graphing i.e. using a Graphs page. In the Data & Statistics page we already defined the function f1(*x*) so it's there waiting to be drawn. If you prefer you can start a New Problem so that all the earlier definitions are "forgotten" and you can enter your function as a new definition for f1(*x*).

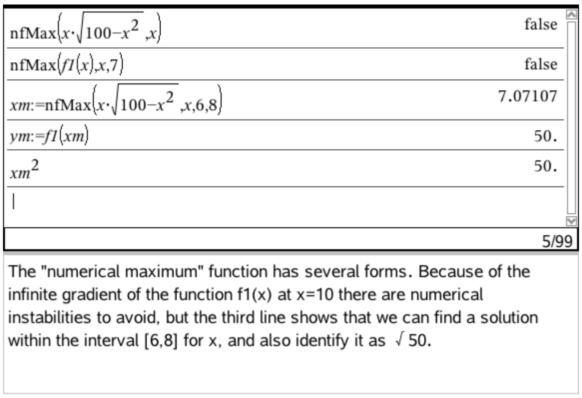
You can use Windows Setting to control the scales on each axis and, with a right click on the graph, you can define its Attributes, and choose the layout for the equation. Using Graph Trace you drag a point along the curve, and when it reaches a maximum (or minimum) it shows an "M" symbol, allowing you to press Enter and capture the point.

So if the Graphs application can determine a maximum, it suggests that the Calculator application should also be able to do so! The required function is called "nfMax" which will find, if it can, a local maximum for a function f(*x*). The *TI-Nspire* reference guide lists three forms:

1. nfMax(Expr, Var) ⇒ value
2. nfMax(Expr, Var, lowBound) ⇒ value
3. nfMax(Expr, Var, lowBound, upBound) ⇒ value

We can use either the explicit form of the Expr: $x*\sqrt{}(100\text{-}x^2)$, or its stored function form: f1(*x*). In either case the Var in question is called *x*. But our function is not a very well behaved one! It is not defined for *x*>10 or *x*<-10. There is a minimum at around (-7.07,50), an inflection at (0,0) and a maximum at around (7.07,50) – but the behaviour at (-10,0) and (10,0) is distinctly nasty!

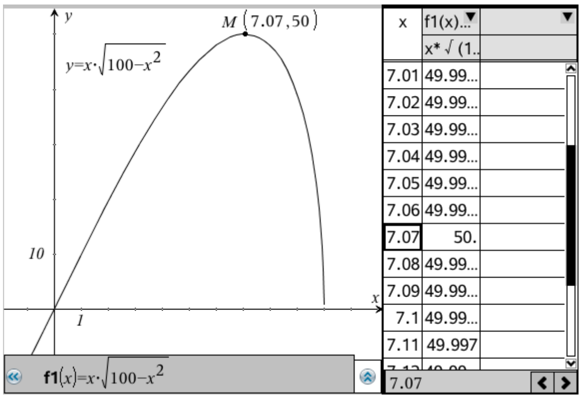
Of course negative values of *x* make no sense for the actual triangle problem we started with, but we haven't told the software that we don't want to consider negative values of *x*. As you see from the screen shown, neither of the first or second forms comes up with a solution, whereas the third one does!



| | |
|---|---|
| $\text{nfMax}\left(x\cdot\sqrt{100-x^2},x\right)$ | false |
| $\text{nfMax}\left(f1(x),x,7\right)$ | false |
| $xm:=\text{nfMax}\left(x\cdot\sqrt{100-x^2},x,6,8\right)$ | 7.07107 |
| $ym:=f1(xm)$ | 50. |
| $xm^2$ | 50. |

The "numerical maximum" function has several forms. Because of the infinite gradient of the function f1(x) at x=10 there are numerical instabilities to avoid, but the third line shows that we can find a solution within the interval [6,8] for x, and also identify it as √ 50.

One very important feature about powerful mathematical software is that it allows us to see various representations of the same bit of mathematics and see how they inter-relate. *TI-Nspire* layout allows us to mix different types of representations on the same page. In this screenshot this is done from within a Graphs page by asking for a Function Table.



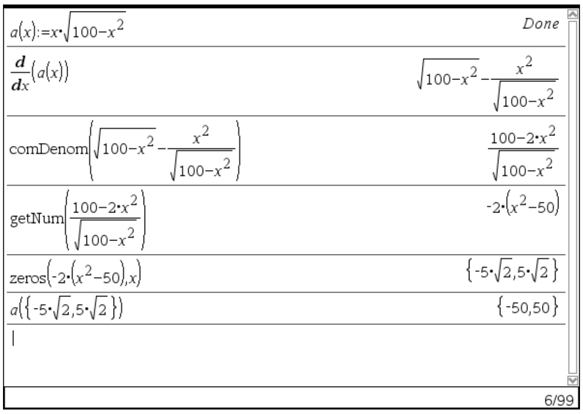
| x | f1(x)...<br>x* √ (1.. | |
|---|---|---|
| 7.01 | 49.99... | |
| 7.02 | 49.99... | |
| 7.03 | 49.99... | |
| 7.04 | 49.99... | |
| 7.05 | 49.99... | |
| 7.06 | 49.99... | |
| 7.07 | 50. | |
| 7.08 | 49.99... | |
| 7.09 | 49.99... | |
| 7.1 | 49.99... | |
| 7.11 | 49.997 | |

So now we have a nice way of showing at the same time a numeric representation in the form of the table, together with a graphic representation in the form of the curve and a symbolic representation in the form of the algebraic expression for the function f1(*x*) whose graph and table are displayed.

**Symbolic** (optional: CAS)

*TI-Nspire* is available in two versions known as CAS and non-CAS. "CAS" stands for "Computer Algebra System" and implies that the software has algorithms for symbolic manipulations in algebra, including differential and integral calculus. So here is a little glimpse into the CAS world to show just how powerful such tools are.

The second line shows a symbolic derivative, and the third shows it manipulated algebraically to a proper fraction. So we can see that the denominator can be zero (at $x$= -10 or $x$= 10) confirming the infinite slope of the function at (-10,0) and (10,0). The fourth line shows the extraction of the numerator, and the fifth shows the zeros of the resulting quadratic function. Finally the sixth line shows the evaluation of the function $a(x)$ at this list of zeros, returning a list of the corresponding values.



$a(x):=x\cdot\sqrt{100-x^2}$ Done

$\frac{d}{dx}(a(x))$ $\sqrt{100-x^2}-\frac{x^2}{\sqrt{100-x^2}}$

$\mathrm{comDenom}\left(\sqrt{100-x^2}-\frac{x^2}{\sqrt{100-x^2}}\right)$ $\frac{100-2\cdot x^2}{\sqrt{100-x^2}}$

$\mathrm{getNum}\left(\frac{100-2\cdot x^2}{\sqrt{100-x^2}}\right)$ $-2\cdot(x^2-50)$

$\mathrm{zeros}(-2\cdot(x^2-50),x)$ $\{-5\cdot\sqrt{2},5\cdot\sqrt{2}\}$

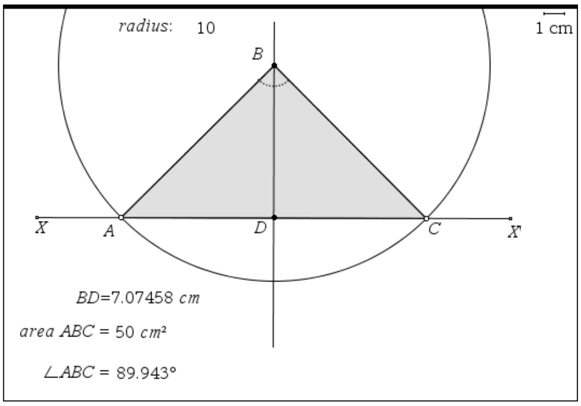
$a(\{-5\cdot\sqrt{2},5\cdot\sqrt{2}\})$ $\{-50,50\}$

6/99

**Geometric**

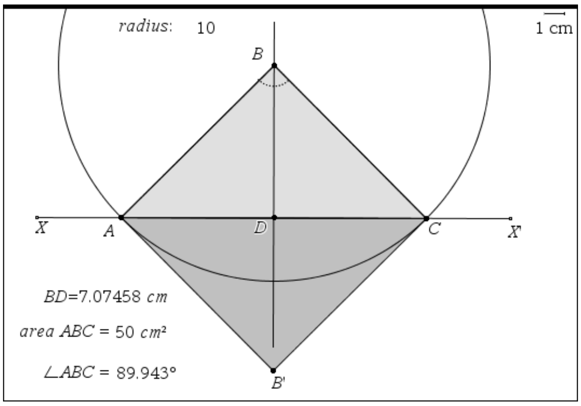
As this was, after all, a problem posed geometrically, it should be one which we can solve geometrically – and maybe that will also give us a greater insight into the shape of the optimum sheep pen – something which the numbers, lists, graphs, tables and symbols have so far failed to reveal!

The chosen diagram starts with a section of the “wall” *XX’*; a segment joining two points. The fixed point *D* is the midpoint of *XX’*. The line through *D* perpendicular to *XX’* is drawn and the slideable point *B* created on it. So in terms our problem we now have the point *B* as our **independent variable**, and its **domain** is the perpendicular bisector of *XX’*. The number 10 is entered as text on the screen, and used with the Compass tool and point *B* to construct the circle centre *B* radius 10 cm.



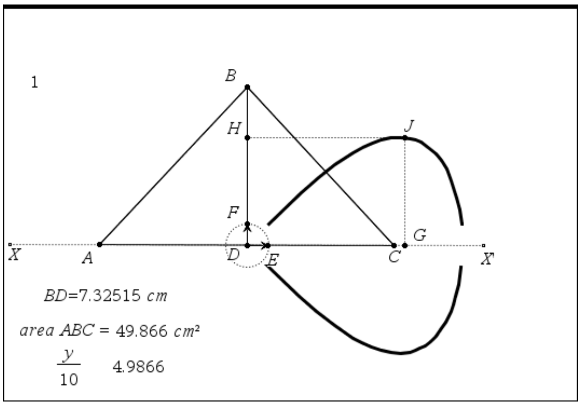
(Of course the actual length in cm will depend on the size of display used!) This circle may or may not intersect the segment *XX’*, but where it does so we label the points *A,B*. Finally we construct the triangle *ABC* and change its attributes to fill it in. We can then measure the distance *BD* (i.e. the height $x$) and the area *ABC*. As we slide *B* up and down these measurements will change and we can find a good approximation to the optimum shape.

Because we have a geometric view now, we can see – and check – that *AD*, *DC* and *DB* are all more or less equal and so estimate the angles in the figure. If we now reflect *ABC* in *XX’* we get the triangle *AB’C*, which together with *ABC* makes a rhombus *ABCB’* of side 10cm. So the optimum configuration will occur when *ABC* is half a square of side 10 i.e. with an area of 50 cm$^2$.



There are other interesting ways in which we can represent graphically data collected from measurements and calculations on a Geometry page. In the first example we will use a geometric approach to making our own graph dynamically. First we hide some of the “clutter” we have acquired such as the circle and *AB’C*. Using the text value 1 and the point *D* as origin we can use Compass to create a unit circle around *D*, and its intersections *E,F* with our new “axes” *DX* and *DB*. Creating vectors *DE* and *DF* gives us use of an important geometric tool called Measurement Transfer. The length *BD* lies between 0 and 10 and so we can happily Transfer it to vector *DE* to produce the point *G* on the *x*-axis. However the area ranges between 0 and 50 which means we need to scale it before plotting on the *y*-axis. So we can enter the scaling formula e.g. “*y*/10” as text, and then Calculate the formula using the value of the area to substitute for *y*. Then the resulting, scaled, value can be transferred to the vector *DF* to create the point *H*. As *B* slides so the points *G* and *H* move along their respective axes. We just have to create perpendiculars to the axes through *G* and *H* to meet at the point *J,* which is the graphical representation of the point whose coordinates are given by the height and area of the triangle *ABC*.

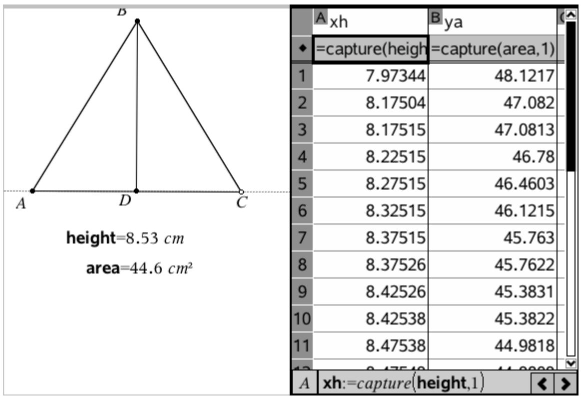
So as *B* slides, *J* now traces out a path that is the graph of the area as a function of height. The set of all positions of a point such as *J* is called a **locus**, so we just need to construct the locus of *J* as a function of *B* to get the curve shown. The attributes of *GJ*, *HJ* and the locus have been changed to help with clarity. Since our construction allows *B* to pass below *XX’* we have acquired a spurious branch of the locus. You might like to think how our construction could have been started differently to ensure that *B* is always “above” *XX’*.



**Statistic**

An alternative way to make a graph from the geometric measurements is to capture them into lists in a spreadsheet. First we copy the geometric construction into a new Geometry page, which we divide vertically to include a Lists & Spreadsheets area as well. We hide unnecessary parts of the construction leaving just the “bare bones” of a resizeable triangle controlled by the point *B*, and the two measurements: length *BD* and area *ABC*. We now want to store the two measurements into variables. So using a right click with the length selected we can select the Store option and type in a name, like “height” to use for this measurement. Similarly we can store the other measurement in the variable “area”.

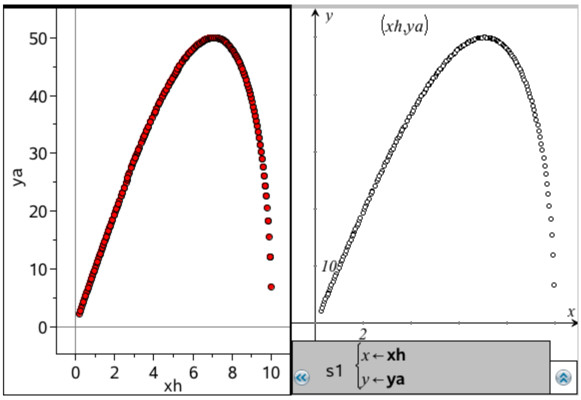
Now we can set up the spreadsheet to collect data automatically from these lists when we start to drag the point *B* about. The lists in columns A and B have been given the names *xh* and *ya*. The formula entered for *xh* uses the Data Capture option and then the Automated Data selection. You edit the formula *xh*:=capture(Var,1) to replace Var with *height*, and similarly set up *ya* as the list formed by capturing *area*. Now as you drag the point *B* in the Geometry window so the data lists grow in the Lists & Spreadsheet window.



| | A xh | B ya |
|---|---|---|
| ◆ | =capture(heigh | =capture(area,1) |
| 1 | 7.97344 | 48.1217 |
| 2 | 8.17504 | 47.082 |
| 3 | 8.17515 | 47.0813 |
| 4 | 8.22515 | 46.78 |
| 5 | 8.27515 | 46.4603 |
| 6 | 8.32515 | 46.1215 |
| 7 | 8.37515 | 45.763 |
| 8 | 8.37526 | 45.7622 |
| 9 | 8.42526 | 45.3831 |
| 10 | 8.42538 | 45.3822 |
| 11 | 8.47538 | 44.9818 |

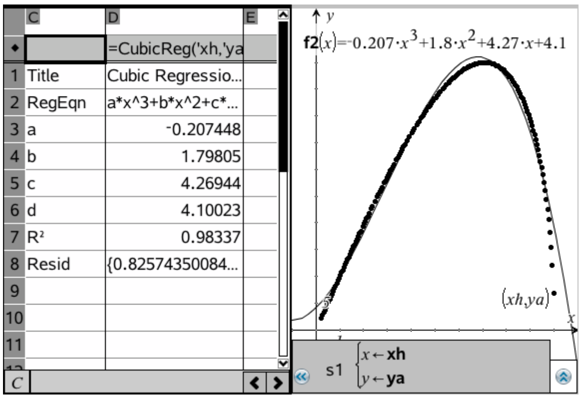
A xh:=capture(height,1)

All we now need is to plot the data as a scatterplot, which we can do either in a Data & Statistics window or a Graphs window – or, as in this screenshot, both.



As a final exercise in this section we shall show how to fit a particular model to the scatterplot data.

A new window is split between Lists & Spreadsheet and Graphs & Geometry. If we select Stat Calculations in the left hand window we can specify the two lists to use (*xh* and *ya*), and the name of the function (f2) into which the equation will be stored. Here we see that a cubic fit has quite a close correlation, but doesn't fit close to (0,0) or (10,0) – which isn't surprising!



| | C | D | E |
|---|---|---|---|
| ◆ | | =CubicReg('xh,'ya | |
| 1 | Title | Cubic Regressio... | |
| 2 | RegEqn | a*x^3+b*x^2+c*... | |
| 3 | a | -0.207448 | |
| 4 | b | 1.79805 | |
| 5 | c | 4.26944 | |
| 6 | d | 4.10023 | |
| 7 | R² | 0.98337 | |
| 8 | Resid | {0.82574350084... | |
| 9 | | | |
| 10 | | | |
| 11 | | | |

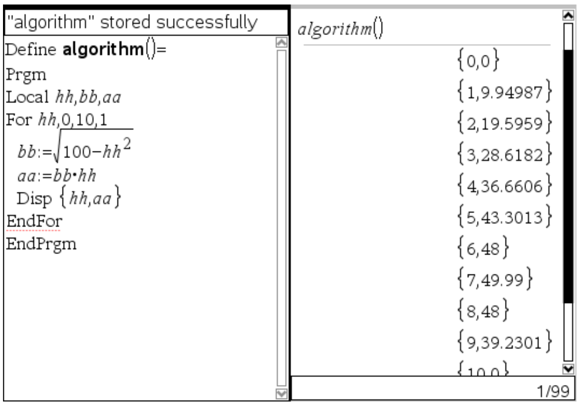
$\mathbf{f2}(x)=-0.207\cdot x^3+1.8\cdot x^2+4.27\cdot x+4.1$

## Algorithmic

*TI-Nspire* includes a powerful programming language. Programs and functions can be defined by inserting a Program Editor area and choosing to build a New Program. Here our program is given the name algorithm and does not pass any arguments. This means it can be called, for example from a Calculator page by entering "algorithm()". The Editor opens up with 3 lines already completed:

```
Define algorithm()=
Prgm


EndPrgm
```



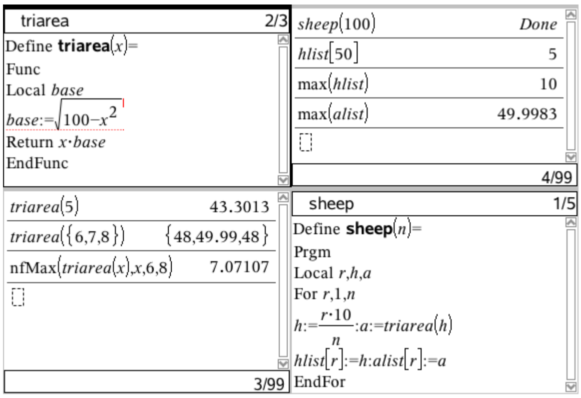
```
"algorithm" stored successfully
Define algorithm()=
Prgm
Local hh,bb,aa
For hh,0,10,1
  bb:=√(100-hh^2)
  aa:=bb•hh
  Disp {hh,aa}
EndFor
EndPrgm
```

```
algorithm()
                {0,0}
                {1,9.94987}
                {2,19.5959}
                {3,28.6182}
                {4,36.6606}
                {5,43.3013}
                {6,48}
                {7,49.99}
                {8,48}
                {9,39.2301}
                {10,0}
                                1/99
```

We can enter our program in the gap provided, using assignment statements such as *a*:=3, I/O statements such as Disp *b*, and structures such as For, While, If etc. Variables used in a program can either be declared to be Local, or otherwise the program will set or reset variable values which can be accessed after the program has run.

Once you have entered the program you can select Check Syntax & Store and, hopefully, you get a "success" message.

Once the program has been successfully entered you can use its name from another *TI-Nspire* page such as Calculator, or use it within another program definition. To illustrate the point we'll define a new program that computes the area of a triangle of given height, and then use it in a program to compute lists of heights and areas.



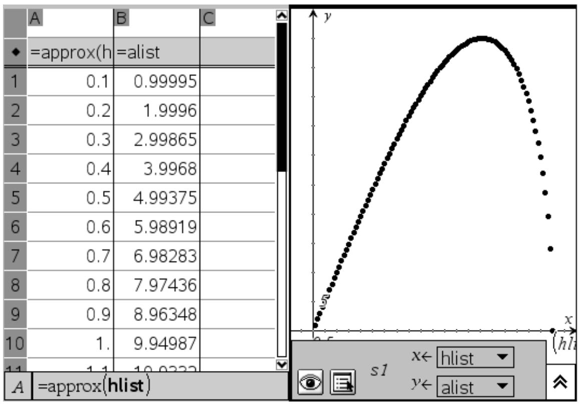
```
triarea                                  2/3
Define triarea(x)=
Func
Local base
base:=√(100-x^2)
Return x·base
EndFunc
```

```
sheep(100)                              Done
hlist[50]                                  5
max(hlist)                                10
max(alist)                           49.9983
                                        4/99
```

```
triarea(5)                           43.3013
triarea({6,7,8})              {48,49.99,48}
nfMax(triarea(x),x,6,8)              7.07107
                                        3/99
```

```
sheep                                    1/5
Define sheep(n)=
Prgm
Local r,h,a
For r,1,n
h:=(r·10)/n:a:=triarea(h)
hlist[r]:=h:alist[r]:=a
EndFor
```

The program triarea(*x*) has been defined using a few more lines than strictly necessary to make the point that this structure allows us to define quite complicated functions, for example, returning different expressions for different ranges of input variable *x*. When we call the program from a Calculator area, we just put in a value (constant or list) for the variable *x*. Now that the function triarea() has been defined we can make use of it in the program called sheep(). In order to use this you specify the value of *n*, which will be the number of data points generated. The program uses both local variables (*r, h, a*) and also global variables (*hlist, rlist*). The square brackets notation: *hlist*[*r*] means the *r*-th element of the list *hlist*. So when the program has been successfully stored we can run it with a value for *n* such as 100 – and then we generate the lists *hlist* and *alist* "as a by-product" of the program. These lists can then be used within other areas of TI-Nspire such as Lists & Spreadsheet or Graphs.



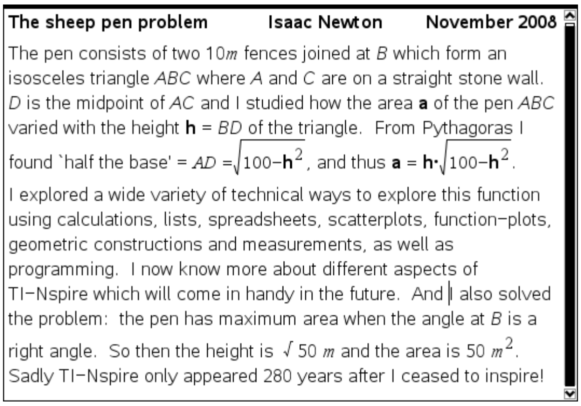
| | A | B | C |
|---|---|---|---|
| ◆ | =approx(h | =alist | |
| 1 | 0.1 | 0.99995 | |
| 2 | 0.2 | 1.9996 | |
| 3 | 0.3 | 2.99865 | |
| 4 | 0.4 | 3.9968 | |
| 5 | 0.5 | 4.99375 | |
| 6 | 0.6 | 5.98919 | |
| 7 | 0.7 | 6.98283 | |
| 8 | 0.8 | 7.97436 | |
| 9 | 0.9 | 8.96348 | |
| 10 | 1. | 9.94987 | |

A =approx(hlist)

## Synoptic

One of the key features of *TI-Nspire* is that it can be used to build up a document consisting of pages each of which can be divided into 1, 2, 3 or 4 different areas. The pages can be organised within sections called Problems – so that the final saved "tns" file is really a book made up from components which are mixtures of:

- Calculator
- Graphs
- Geometry
- Lists & Spreadsheet
- Notes
- Data & Statistics
- Program Editor.



**The sheep pen problem Isaac Newton November 2008**

The pen consists of two 10*m* fences joined at *B* which form an isosceles triangle *ABC* where *A* and *C* are on a straight stone wall. *D* is the midpoint of *AC* and I studied how the area **a** of the pen *ABC* varied with the height **h** = *BD* of the triangle. From Pythagoras I found `half the base' = *AD* = $\sqrt{100-\mathbf{h}^2}$, and thus **a** = **h**·$\sqrt{100-\mathbf{h}^2}$.

I explored a wide variety of technical ways to explore this function using calculations, lists, spreadsheets, scatterplots, function−plots, geometric constructions and measurements, as well as programming. I now know more about different aspects of TI−Nspire which will come in handy in the future. And I also solved the problem: the pen has maximum area when the angle at *B* is a right angle. So then the height is $\sqrt{50}$ *m* and the area is 50 $m^2$. Sadly TI−Nspire only appeared 280 years after I ceased to inspire!

Including Notes areas and/or pages such as the example shown here, allows you to describe aspects of your problem: your strategy, suppositions, conclusions, next steps etc.