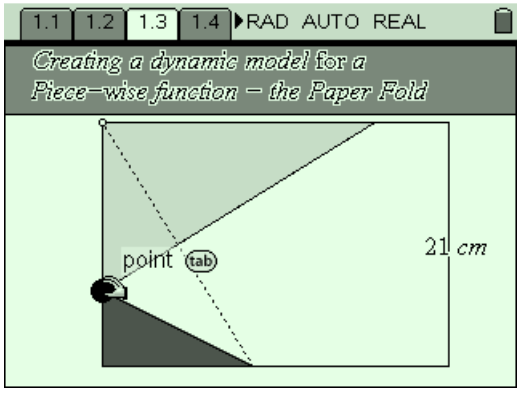
I didn't know you could do THAT!

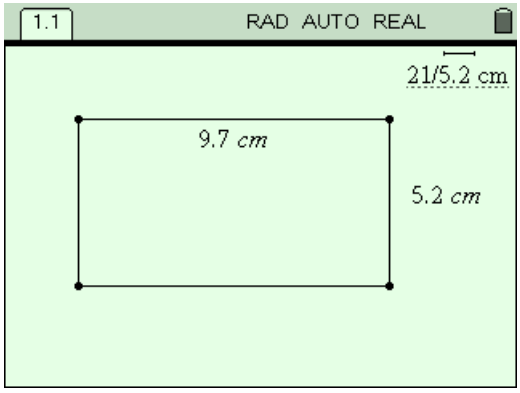
Steve Arnold, T3 Australia

Over the past two years, I have been extremely fortunate in having both access to the new TI-Nspire platform AND the time to really push its limits and to work with it extensively. My roles as materials developer, editor and reviewer, as well as work in aspects of product development have provided me with skills and insights that would not have been possible had I been doing a "real job" over that time. The examples discussed here are some of the fruit of that experience – cool and useful things that I have learned and which may be helpful for others in their uptake and effective use of this wonderful new technology for teaching and learning.

I have chosen three areas of focus for this conversation: neat geometrical constructions associated with piece-wise constructions (as exemplified by the paper folding activity and the now-classic falling ladder), an introduction to some ideas concerning what we are calling "dynamic algebra", and the possibilities being presented by the new programming capabilities of TI-Nspire. While many of the features I describe here have application across the platform, the main focus of my interest has been with the computer algebra aspects of TI-Nspire, and so assume access to TI-Nspire CAS.

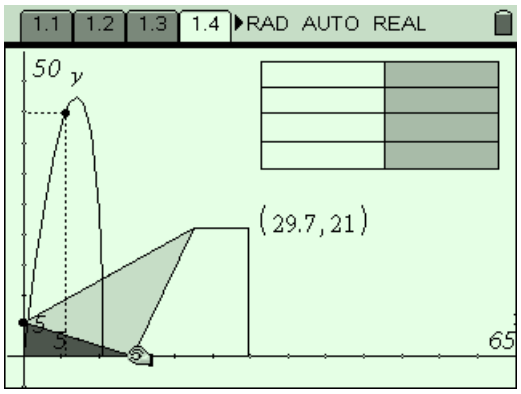
1. Some Neat Constructions





Many of the activities I have developed for TI-Nspire involve constructing a dynamic geometric model which, through measurement, generates data leading to a graphical model. Students may then work backwards – developing and verifying an algebraic model using the geometric and graphical representations. As with any good software tool, TI-Nspire offers several choices along the way and certain powerful teachers which make it uniquely well-suited to the task of supporting algebraic modelling.

The first choice lies in whether to use the graphing view or the geometric view for your construction. The former allows the graph to exist as a "floating" analytical window, while the construction can be placed to one side. The wonderful "Scale" feature (MENU > View > Show/Hide Scale) offers an excellent facility for constructing scale models. In the case of the paperfold, for example, I might want my page to be 29.7cm x 21 cm (A4) or 11 in x 8.5 in (US letter).

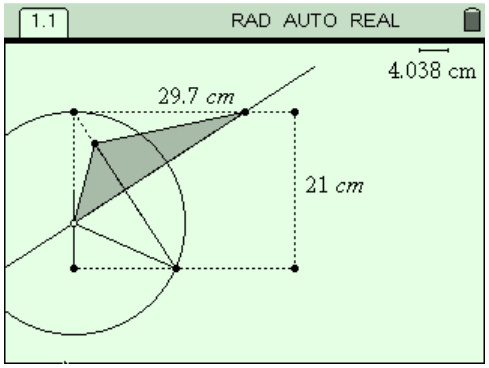


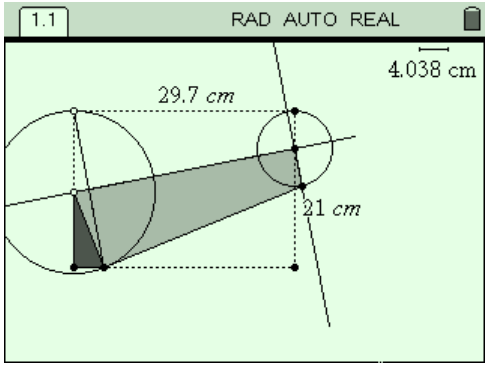
Simply construct your rectangle, measure the dimensions (say it is 9.7 cm x 5.2 cm), and then take the scale (by default, 1 cm): set it to what you want (21 cm) divided by what you have got (5.2) and press ENTER – your width is now 21 cm! Drag the length to the required size and you have your scale model.

If you use the Graphing View, then simply setting the coordinates of the top right corner appropriately will give you the rectangle of your choice – even easier!

The construction of the folding paper involves a few simple considerations: it is best to create a segment on the left-hand side to serve as the "track" for your moving point (you may even want to limit the segment to half the left-hand side – you will see why when you create your model!)

Put a Point On the segment and this is the height of your fold. Folding a real piece of paper will show that the length from this fold point to the top left corner, upon folding, becomes the hypotenuse of the triangle that is the focus of the activity – what fold gives the triangle of greatest area? So we need this length to be reproduced to meet the bottom of the rectangle – construct a circle with centre at our fold point and radius up to the top left corner point. The Intersection Point of circle and rectangle gives the base point of the hypotenuse, and we have our triangle. If we just wanted the data, this is sufficient – we have the height and the area of the triangle, from these we could create a graph and do the work required for our model.





However, what we really want is a cool looking model that is visually pleasing! To put in the fold, we create a segment from the top left corner to the new point on the base. The fold will be perpendicular to this segment from our fold point on the side. (If you want the neat dragging of the corner down, you should put a Point On this last segment and draw a triangle to include that point, as shown).

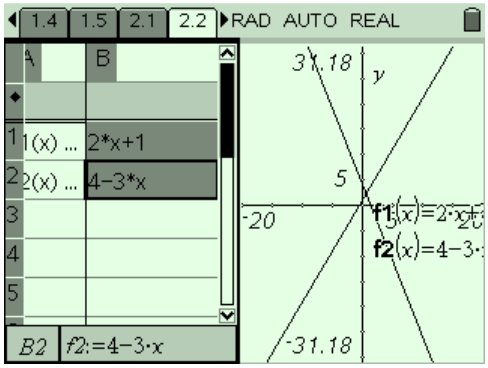
The real challenge in this construction occurs when the height is close to its maximum value and the triangle formed above the fold becomes a quadrilateral, as shown. We would like to simulate the fold of the top right corner down over, just as the corner of the paper does. A circle to fix the distance and a perpendicular from the fold produce the required point. Construct the polygon as shown and then hide all the unnecessary features and we have a pretty neat model.

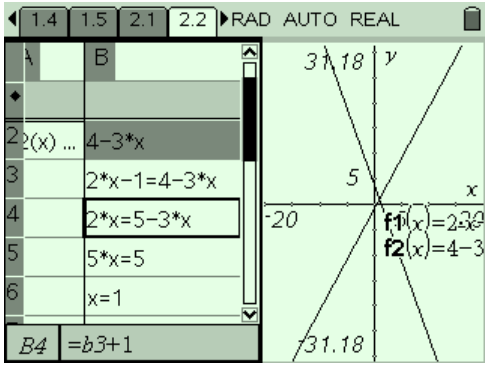
The use of measurements to generate the data again offers a choice – do we use data collection (automatic or manual?) or do we do a measurement transfer onto the axes? Each is a matter of preference and convenience – I have come to prefer manual data collection over automatic – while the latter is more fun, it quickly generates huge amounts of data which can be a problem to get rid of; manual data collection forces students to be active and strategic in the places where they choose to capture points – I much prefer this deliberateness. Overall, however, I like the measurement transfer method since it shows a very direct connection between the points on the model and the data generated.

2. What is Dynamic Algebra?

When we use the Calculator (for algebraic work, in particular) the nature of the History as a static record of computation may not always be preferred. If I am involved in some type of extended algebraic process, then there are times when I would like to be able to change an initial condition and see that represented dynamically throughout the computation – similar to what we see when a numerical spreadsheet automatically recalculates.

Fortunately, TI-Nspire CAS does offer such a feature – and it exists within the **Lists & Spreadsheet** environment, one of the only algebraic spreadsheets in existence.





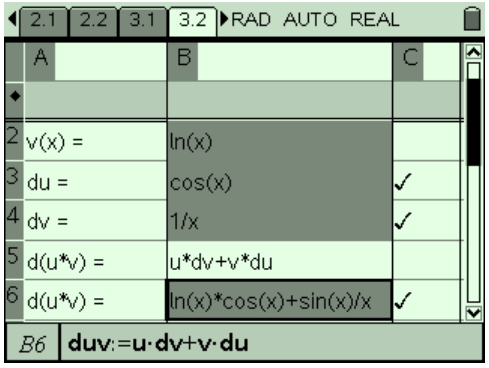
We might begin with a very simple but useful "trick" – creating an interaction between the spreadsheet and the **Graphs & Geometry** page: I would like to be able to enter functions into either my L&S page or into the G&G environment, and have the other update auotmatically.

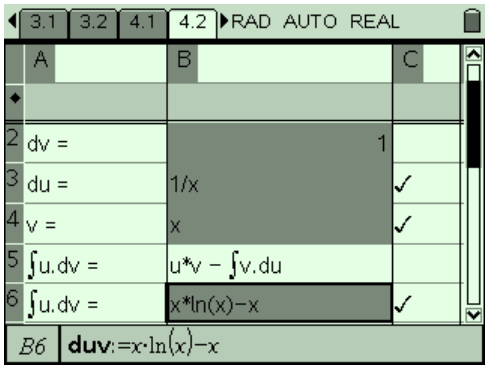
Easy – just put something into **f1** and **f2** in the graph entry line (I use ***x*** and ***–x***). Now enter the functions you want into the spreadsheet, as shown. Finally. **Store** those cells to **f1** and **f2** using the **VAR** button (or **CTRL-L**). Done!

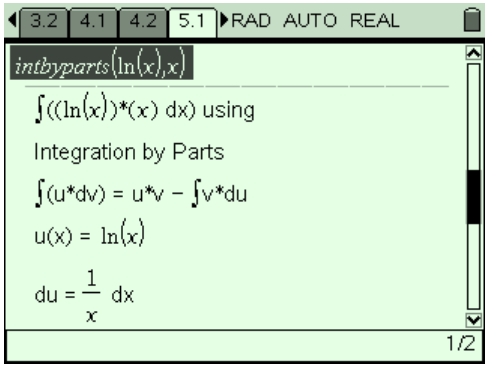
Now when you change the functions in the spreadsheet, the graph will automatically update – and the same will happen if we change the graphed functions – the spreadsheet will reflect this change.

If we wanted to solve these equations simultaneously using our algebraic spreadsheet, we could set the two functions as equal in cell B3, and then solve step-by-step in subsequent cells. Students can simply type in each step or refer to the cell above (preceding with an "=" sign).

The **real** challenge is using CAS effectively for teaching and learning (especially with younger students) is to *not let the tool do all the work!* My current preferred model offers students scaffolding without doing the work for them – just checking their work as they go!







Consider a process such as the **Product Rule** for differentiation. Students enter the two factors of the product, then enter the derivatives of each (and this is checked for correctness!). Each step is given a variable name, consistent with the product rule formula, and students can have this rule displayed for them, if desired. In the example shown, ***u(x) = sin(x)*** and ***v(x) = ln(x)***. Students may enter the final result using their own values, or they may take advantage of one more level of scaffolding, and simply enter the formula as stated, since each part has been stored as a variable.

The same process is shown applied to the process of **Integration by Parts**. Notice that the layout provides a third level of scaffolding as students learn how to actually set out a complete solution to such problems? In some ways, this skill is equally important as the calculus-related skills which make up the steps of the process.

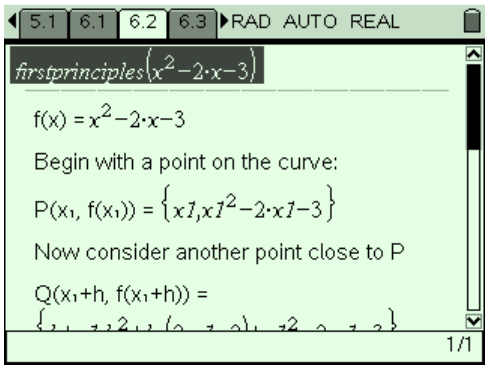
Such scaffolding may be further supported using programming. Although not (yet) interactive, TI-Nspire CAS programs can easily be written to demonstrate step-by-step processes and so to offer students a way to quickly and easily check their own working and to play "what if" games with these processes.

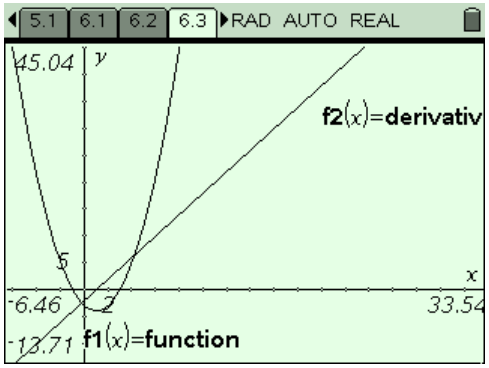
It may even be useful to terminate the program early and allow the student to complete the final steps – and then check their answer against a stored **result** variable!

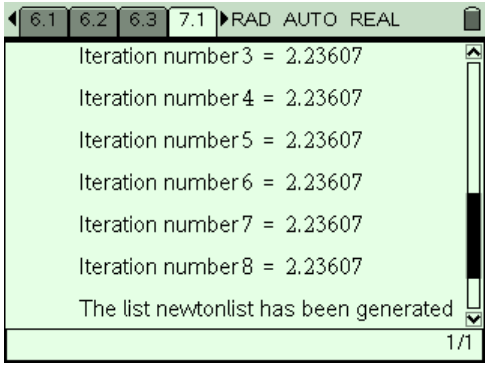
Much remains to be explored in this exciting application of "dynamic algebra" as embodied within TI-Nspire CAS. As with any powerful mathematical tools, their use with students needs to be tempered and cautious. At the same time, there exists great potential for the learning of the difficult manipulative aspects of algebra – the syntax which has formed the principle stumbling block for most students in their acquisition of algebraic facility and understanding. Coupled with deliberate and careful applications which promote students in their use of the "semantics" – the meaning of those symbols – then we have a complete approach to algebraic learning in schools. These semantics arise from meaningful applications, such as the paperfolding and falling ladder activities, in which students build and verify their own algebraic models based upon geometric and graphical representations. They arise, too, from

concrete models, such as area models for algebraic symbols which give students opportunities to build firm concrete foundations for the manipulations they will be using in future studies.

3. Programming in TI-Nspire CAS: Just what is possible?







Powerful programming facilities now exist within the TI-Nspire platform, and especially for TI-Nspire CAS. The inputs of such programs may be functions as well as numerical values, as may be the outputs – although they may not, as yet, be presented using the usual function notation.

For example, the program **firstprinciples(function)** is shown. Students enter a function and the program generates the step-by-step process of differentiation by first principles, as applied to this function.

The program also generated the derivative of the original function, as derived from the first principles process. Both function and derivative are outputted as variables of the same name.

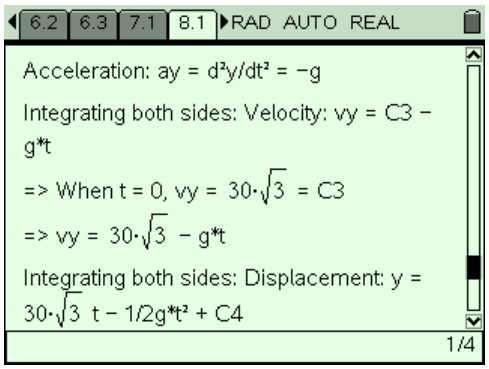
If a **Graphs & Geometry** page is prepared in advance, with **f1(x) = function** and **f2(x) = derivative**, then running the program will automatically graph the function and its derivative as shown.

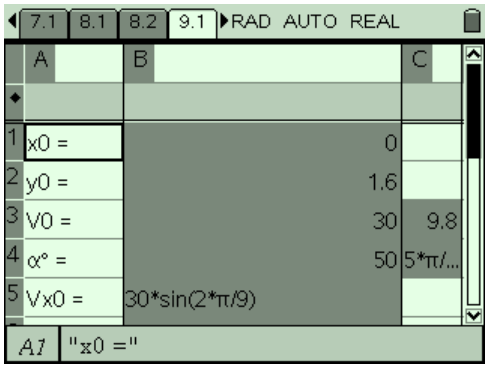
The program **NewtonCAS(function, initial_guess, number_of_iterations)** will generate a list of the iterations, which, if entered into a **Lists & Spreadsheet** column name, will display these values, and allow further exploration and comparison, including scatter plots.

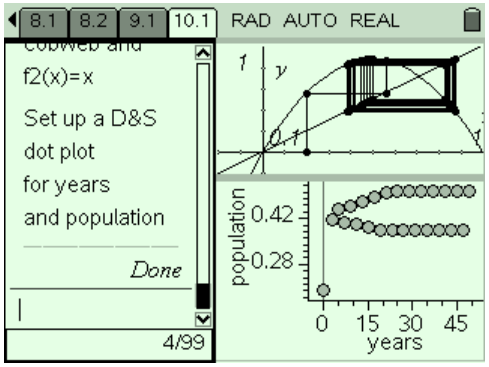
This facility for programs to generate variables which may be numerical, functional or lists, seems sure to prove useful in many ways.

Many of our algebraic contexts involve students in developing extended solutions, for which the process is just as important as the final result.

Consider, for example, the study of **projectile motion**. It is expected that students should be able to derive the







trajectory path for such motion from the initial conditions and the horizontal and vertical components of acceleration – differential equations which give rise the velocity and, finally displacment functions.

These may then be combined and, most importantly, then applied to problems involving the motion – what is the maximum distance, or maximum height, or time taken for this motion?

While CAS can be used to generate such results, careful use of scaffolding by both programs and algebraic spreadsheet can assist students to master both the manipulative skills required, but also the process, layout and working which is just as important.

I am drawn to the idea of leaving students to finish the job themselves, and then being able to check their results.

While there exist some very real limitations at present in the programming environment available for TI-Nspire, much more is possible than many people yet realize. Programs such as the **webplot**, shown, can be used to control graphs and variables, and offer an ideal tool for experimentation and for student enquiry. And, of course, the very best programs remain those that students write themselves, demonstrating their own understanding of the concepts and skills that we are trying so hard to impart!