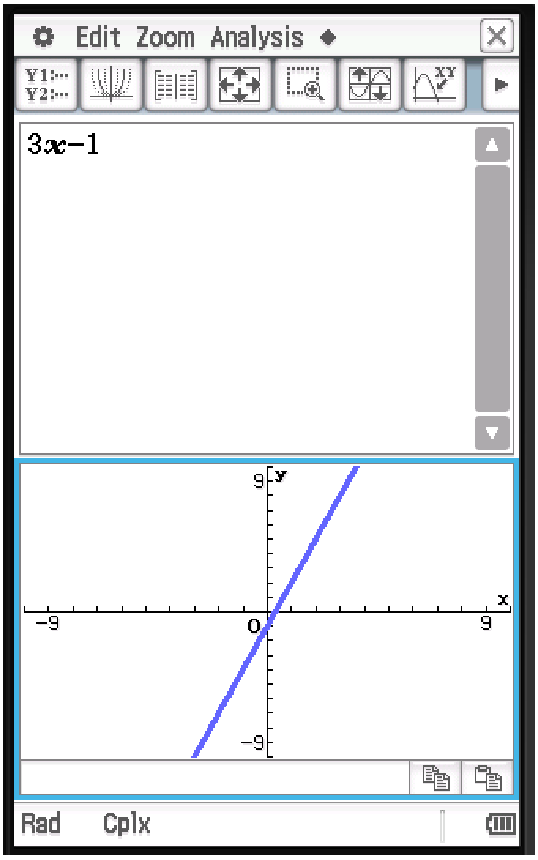
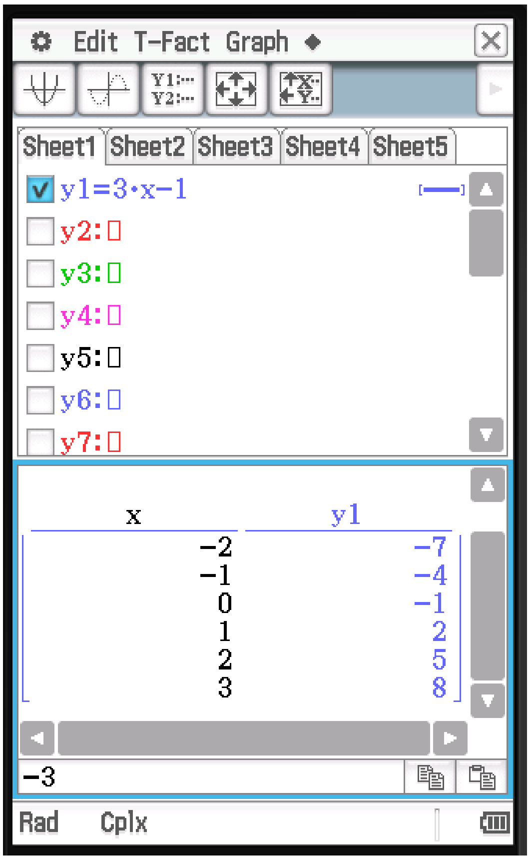
Numeric Approximations to Roots of $f(x) = 0$

Consider an equation of the form $f(x) = 0$, such as $3x - 1 = 0$

When looking for a *solution* to such an equation, we are looking for values of the variable it contains which satisfy the equation – implying that the value of the left hand side of the equality (in this case, $3x - 1$) is equal to the right hand side (in this case, 0) when the value is substituted for the variable.

We can always begin with a graph of $y = f(x)$ on a Cartesian plane, from which a solution may be evident. Such a graphical representation might be obtained after the creation of a table of values, through which pairs of points which satisfy the condition $y = f(x)$ for a selection of x values are identified. Sometimes the table of values itself will reveal a solution.





From both of these representations, graphical or tabular, we are looking to see approximately where x is such that $f(x) = 0$.

More specifically, we are looking for values of x between which our required solution lies.

If we accept that the x values between which our solution must be are a and b, then we would say that our solution lies between (a, b) or even that it is contained in the domain (a, b).

In our example above, we see that the domain which contains our solution is $(0,1)$, most obviously from the graph but clearly this can be deduced from the table of values also.

Upon checking our boundary values, we find that

$$f(a) = f(0) = -1$$

$$f(b) = f(1) = 2$$

If we find that these two function values have opposite signs, implying that $f(a) < 0\ \&\ f(b) > 0$ or vice versa ($f(a) > 0\ \&\ f(b) < 0$), then we can be sure that somewhere between these two x values lies a value for which $f(x) = 0$ provided $f(x)$ is continuous.

Why not, then, cut our interval in half (*bisect* it), and assess the value there?

If we calculate $\frac{a+b}{2}$ and then evaluate $f(\frac{a+b}{2})$, we can halve the range of values within which our solution must lie – and might even land on the solution if we are lucky.

If, when $f(\frac{a+b}{2})$ is evaluated, we find that it is a positive value, then we should replace our previous boundary value for which $f(x)$ was positive by our new $\frac{a+b}{2}$ value. Alternately, if we find $f(\frac{a+b}{2})$ to be a negative value, then we should replace our previous boundary value for which $f(x)$ was negative by our new $\frac{a+b}{2}$ value.

Let us apply this bisection method to our example of $f(x) = 3x - 1$ on $(0,1)$.

$$\frac{0+1}{2} = \frac{1}{2}$$

$$f\left(\frac{1}{2}\right) = 3 \times \frac{1}{2} - 1 = \frac{1}{2}$$

As $f\left(\frac{1}{2}\right) > 0$ we will replace our old b value, 1, by the new value, $\frac{1}{2}$.

Our new interval containing the solution to $f(x) = 0$ is now $(0, \frac{1}{2})$.

Ponder under what circumstances following this process that we might be able to land upon the solution to $f(x) = 0$. Take the time to consider the cases when the roots are:

- Irrational
- Rational

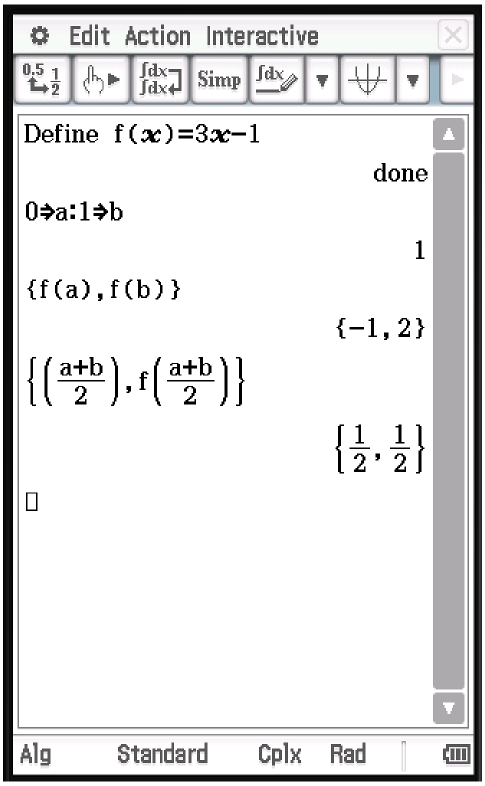
When the roots are rational, can you predict when we could land on the root? How will this occur?

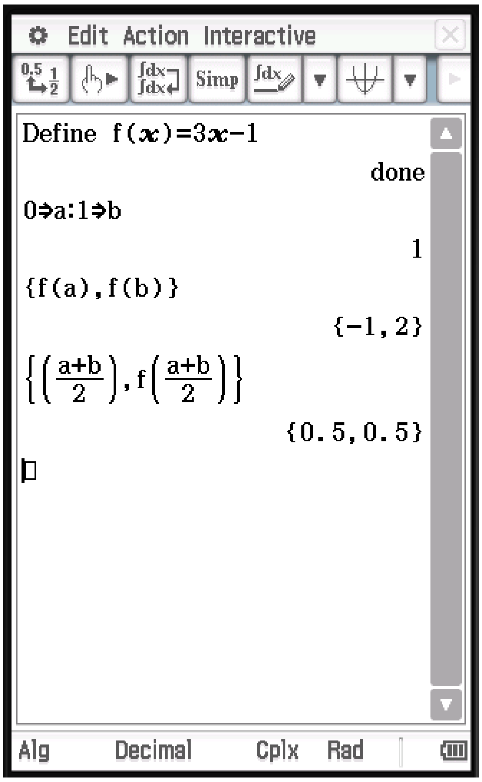
Numeric Approximations to Roots of $f(x) = 0$

The Bisection Method on the CASIO Classpad

This bisection method can be adopted and used iteratively using the Classpad.

Firstly, it could be achieved in Main, by calculating a series of steps, following which a decision is made to replace one of our boundary values prior to recalculation:

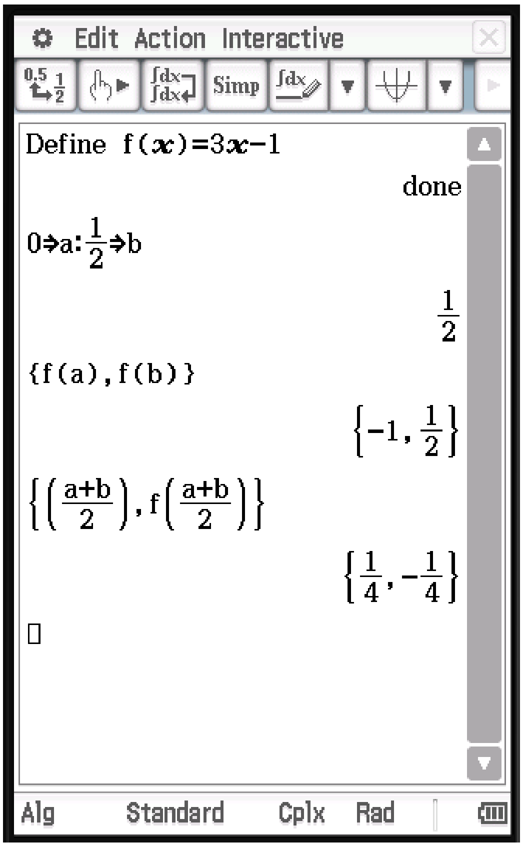


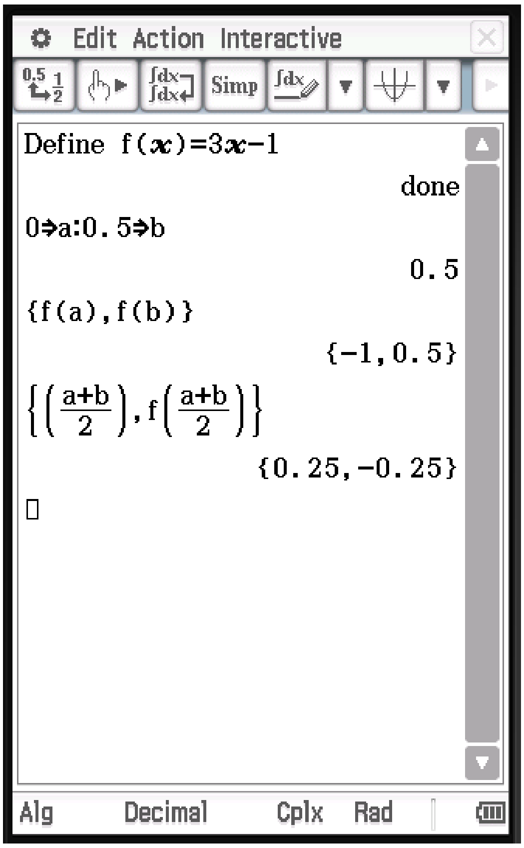


Two screen shots are shown here – one using Standard mode and the other Decimal mode.

You might wish to discuss which of these will be most appropriate. Will this always be the case? Under what circumstances will each mode be preferable?

As $f\left(\frac{a+b}{2}\right) > 0$ we should now drag the current value of the midpoint of our interval $\left(\frac{a+b}{2}\right)$ and substitute it for the old value of b as $f(b)$ was positive previously:





Clearly this process can be performed iteratively.

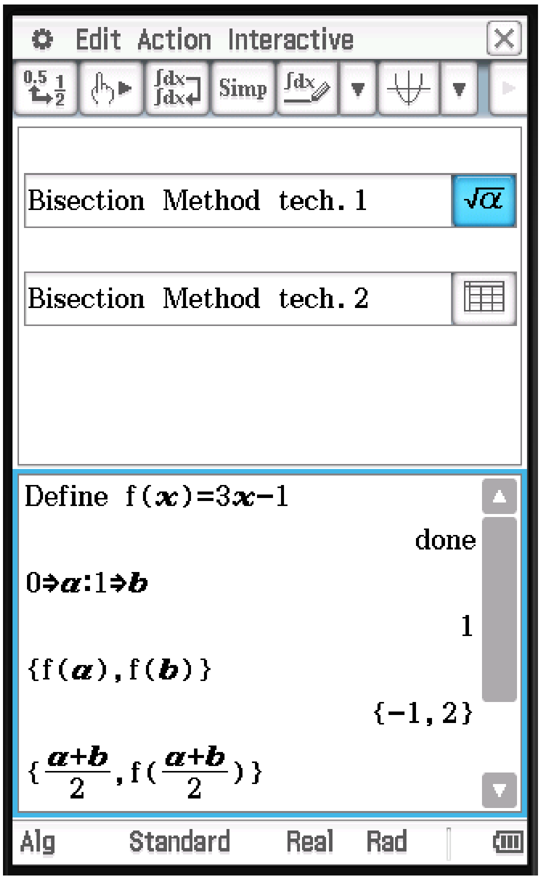
In this particular example, under what circumstances will the exact solution be possible to be 'landed upon'? Will it always be found, sometimes be found, or never be found?

Certainly it would appear that you might usefully continue to cycle around until your solution is as near as you might require, if landing upon it is not possible.

Numeric Approximations to Roots of $f(x) = 0$

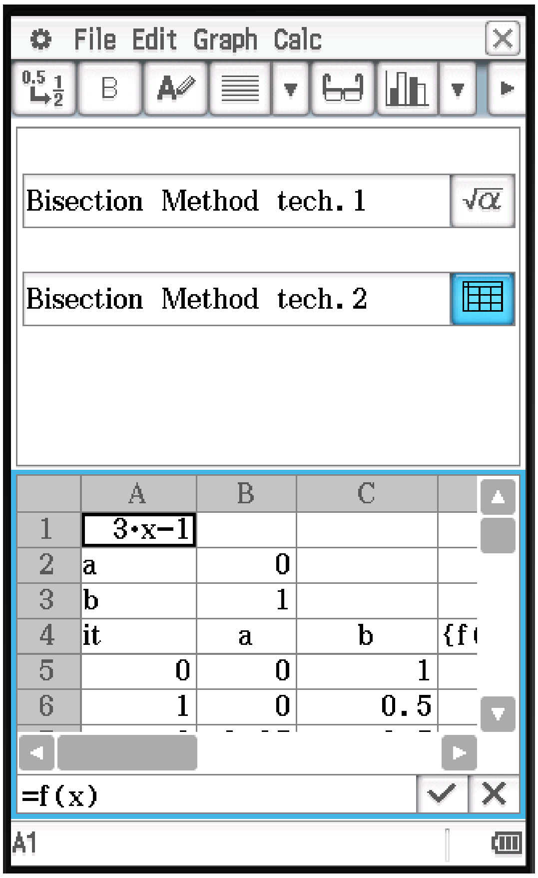
An eActivity Implementation of the Bisection Method

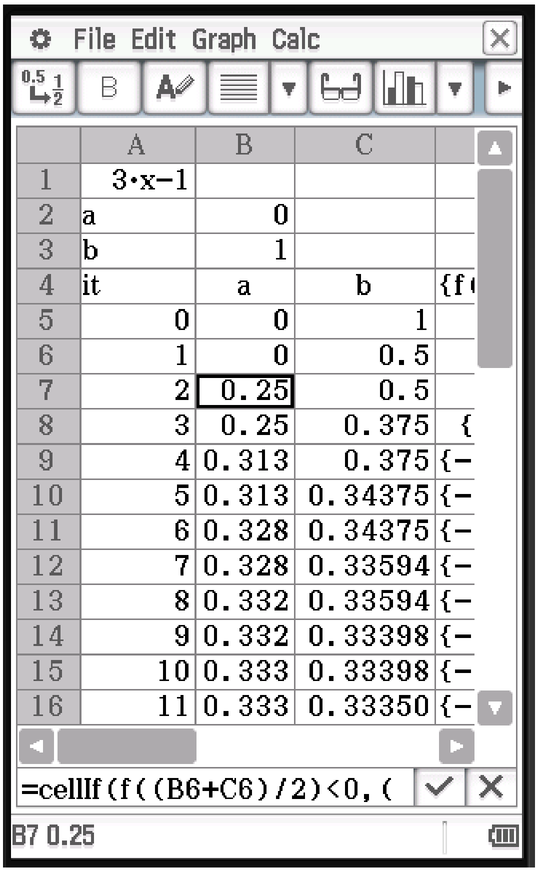
As this bisection method will be used on more than one occasion, you might like to define and use an eActivity which has the predefined steps set up. An example can be seen below, and copies of this eActivity called 'nsolvbisect' can be imported into your Classpad – instructions for this are supplied in Appendix 1.



Note that the function is defined within the strip, with the advantage that this (and, indeed, the other variable values) are self contained within the eActivity and will not conflict with any other definitions you may be maintaining in Main. You should resize your screen to allow ease of access to drag values as will be needed to iteratively close in on the solution.

Note also that there is a second strip shown – this leads to a spreadsheet, in which mode multiple calculations can be conducted rapidly, to save the need for copying and decision making.





The spreadsheet is an under-utilised area of many people's Classpads, and really deserves to be better understood for its abilities to take advantage of the logic processes and copying of iterative command sequences, as well as its ability to generate tables of values which provide a scrollable visual representation of the results.

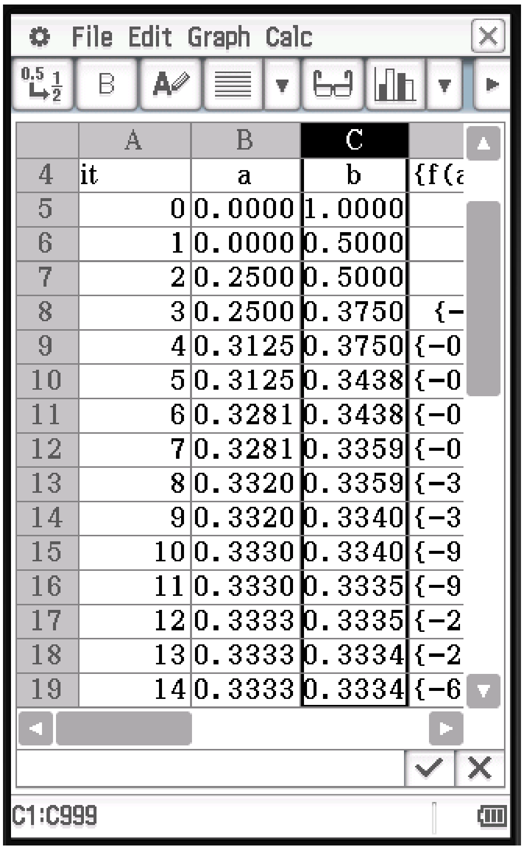
As can be seen from the resized screen to the right, this spreadsheet automatically carries out the bisection method to a pre-defined number of iterations, deciding at each iteration which boundary value should be replaced.

The function is taken from the existing function definition in the eActivity strip called 'Bisection Method tech. 1'.

Experiment with the starting values to fathom the behaviour should it 'land on' the solution.

Being able to see a list of values also enables us to observe the convergence to a solution.

In the case below where 4 decimal places are shown, it can be seen to converge to 2 decimal places of accuracy within 9 iterations.



	A	B	C	
4	it	a	b	{f(a
5	0	0.0000	1.0000	
6	1	0.0000	0.5000	
7	2	0.2500	0.5000	
8	3	0.2500	0.3750	{−
9	4	0.3125	0.3750	{−0
10	5	0.3125	0.3438	{−0
11	6	0.3281	0.3438	{−0
12	7	0.3281	0.3359	{−0
13	8	0.3320	0.3359	{−3
14	9	0.3320	0.3340	{−3
15	10	0.3330	0.3340	{−9
16	11	0.3330	0.3335	{−9
17	12	0.3333	0.3335	{−2
18	13	0.3333	0.3334	{−2
19	14	0.3333	0.3334	{−6

Program Mode Implementation of the Bisection Method

Having now seen the process of iteration being carried out automatically within a spreadsheet, the logical conclusion becomes a program which will replicate the process:

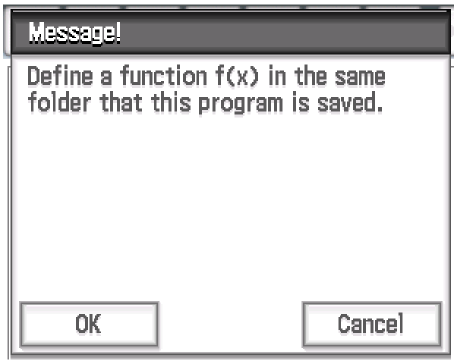
There are three programs described here, all of which are available to be loaded onto your calculator - instructions for this process are included in Appendix 2.

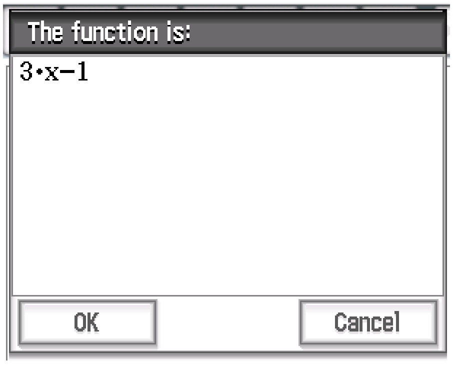
Note that these programs require that the function $f(x)$to be defined in Main.

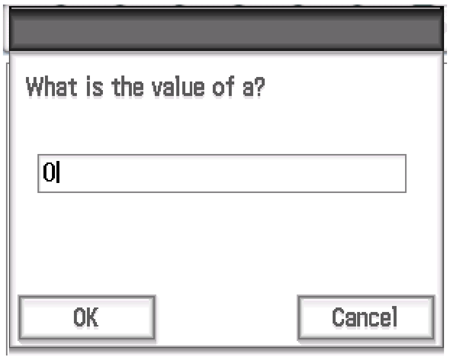
The program Bisecti1 will provide all values in Standard (fractional) form.

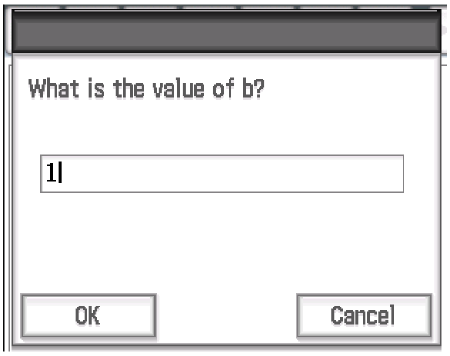
The program Bisecti2 will provide all values in Decimal form.

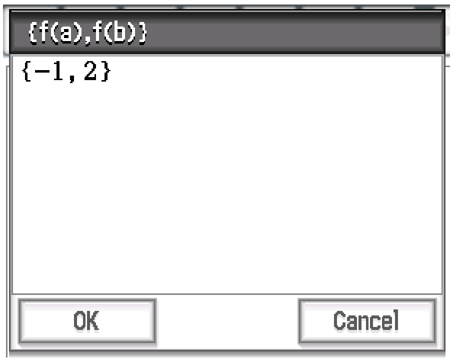
The screen shots which follow show the first few iterations of Standard form (Bisecti1).

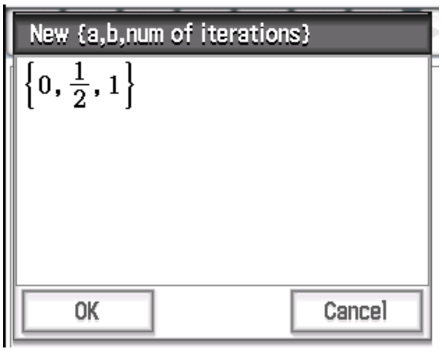


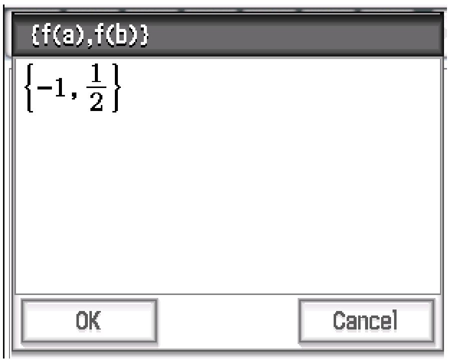


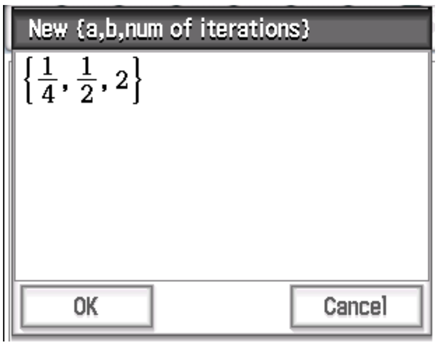






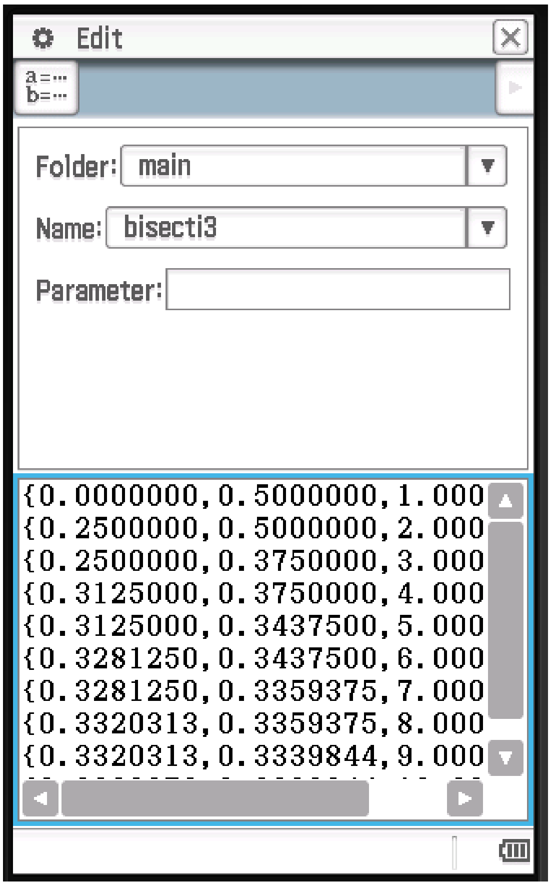






This process will now continue until the user chooses to Cancel out of the loop.

The program Bisecti3 allows for the user to follow a similar process, but provide a value for the number of iterations. After a rapid calculation, it will provide a scrollable listing of all calculations in those iterations:



Here again the convergence can be observed.

Experiment with different starting values to determine the behaviour of the system should it land on the solution.

Other Functions and Forms

Try using the Bisection Method in each of the methods described above on a few other examples, with non-linear forms including components such as exponentials, periodic phenomena and higher order polynomials, perhaps not equal to zero:

$$e^{2x} - x = 0$$

$$\sin 2x - x = 1$$

$$x^6 + 5x^3 - 5x - 1 = 0$$

Also attempt to solve cases where multiple roots exist in comparatively close proximity.

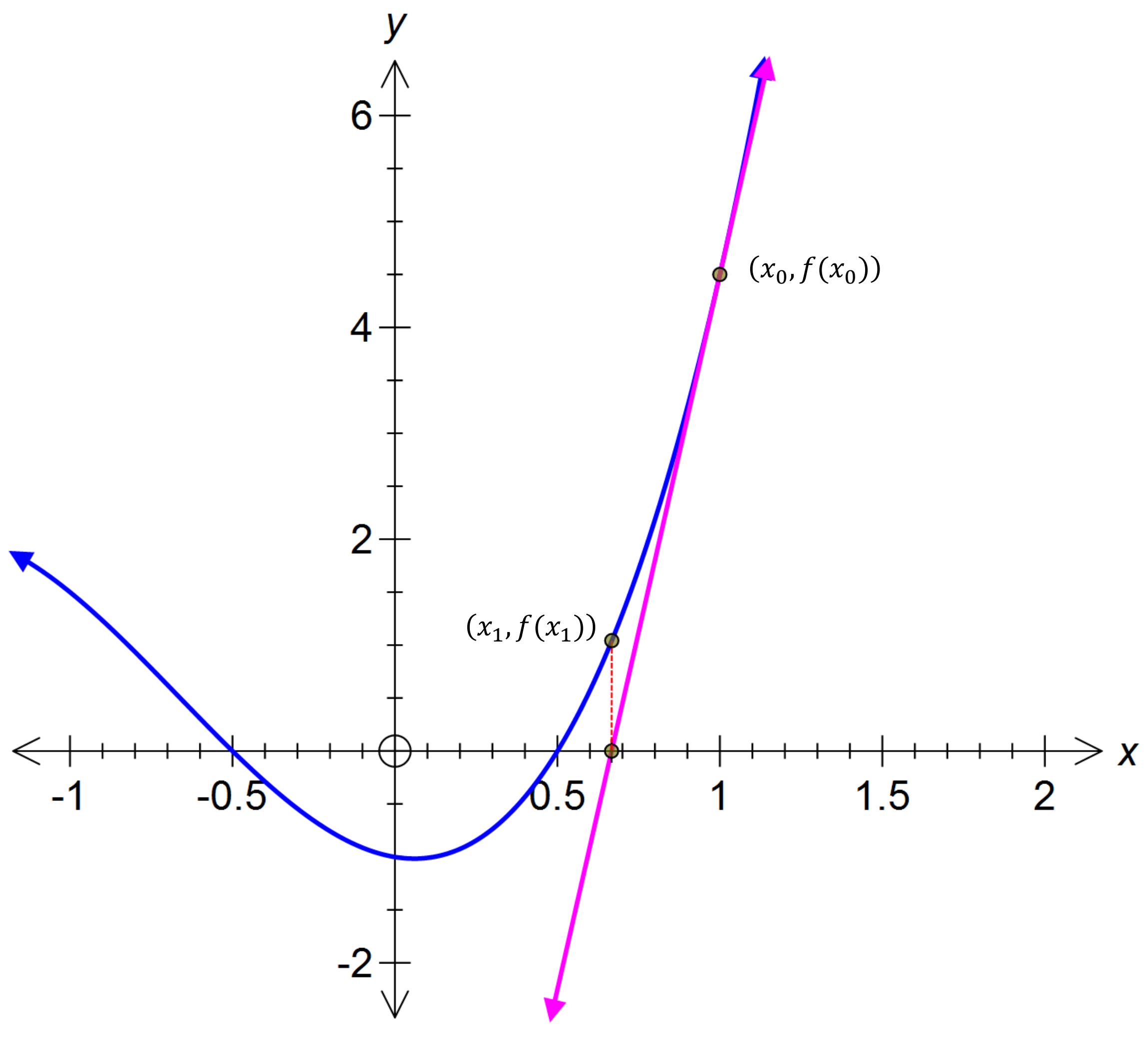
Numeric Approximations to Roots of $f(x) = 0$

Newton's Method

Newton's Method (or, more technically correct, the Newton-Raphson Method) for the numeric approximation to the roots of a function is an iterative process which takes an informed guess as a starting point. At this point, the tangent to the function is calculated, and the x-intercept of this tangent line will typically be a better approximation for the unknown root of the function.

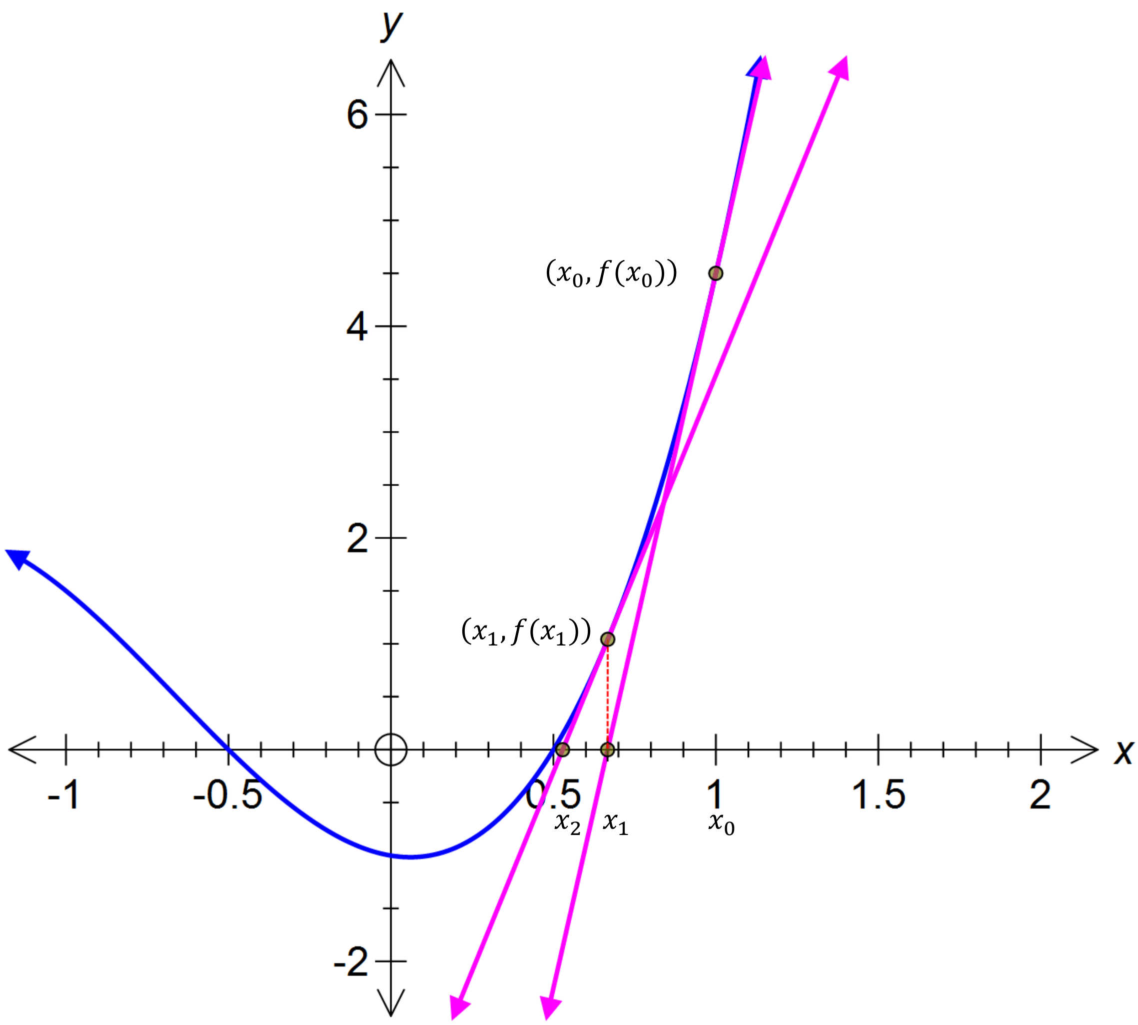
Algebraically, if the initial guess were to be x_0 then a better numeric approximation for the root nearby would be

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$



This process can be iteratively applied, using

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



The above diagram showing the second approximation illustrates the speed with which this method can converge to the actual root.

This method will point you directly to the root in cases of linear functions, such as $f(x) = 3x - 1$ which was illustrated earlier with the Bisection method. You should convince yourself of that fact algebraically. The following screenshot shows the calculation:

Define $f(x)=3x-1$

done

$1-\frac{f(1)}{diff(f(x),x,1,1)}$

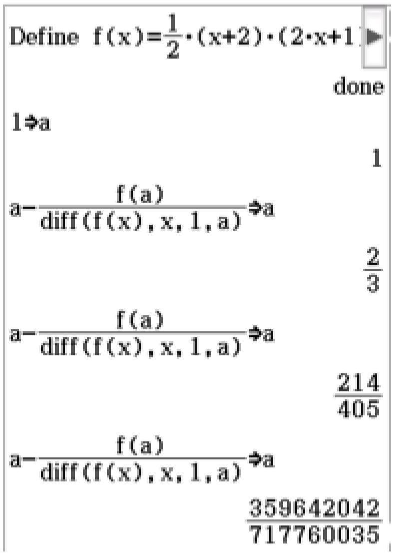
$\frac{1}{3}$

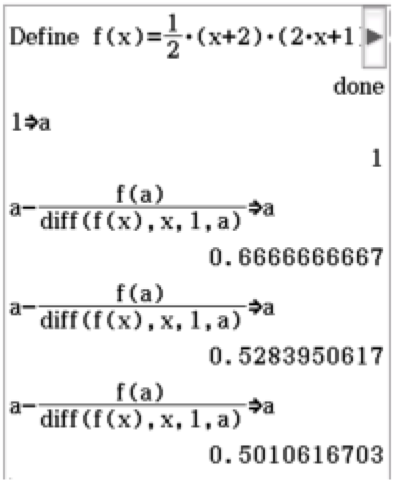
$f\left(\frac{1}{3}\right)$

0

The example illustrated in our earlier graphics was a cubic: $f(x) = \frac{1}{2}(x+2)(2x+1)(2x-1)$ where an initial guess of $x_0 = 1$ was used.

The screen shots below illustrate Main calculations, performed in Standard mode on the left, and Decimal mode on the right:



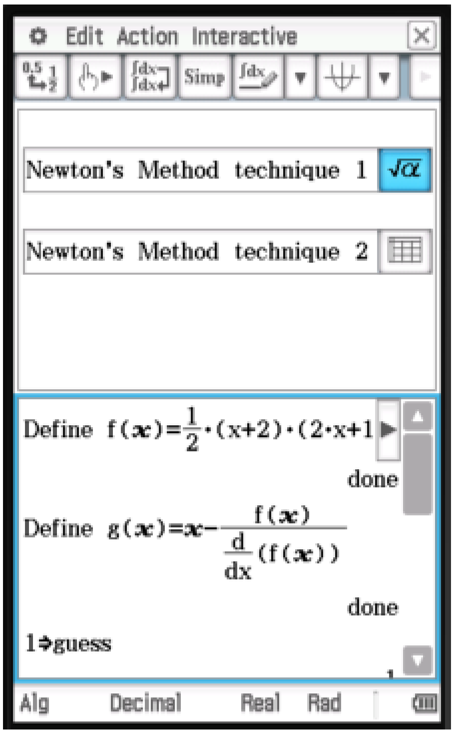


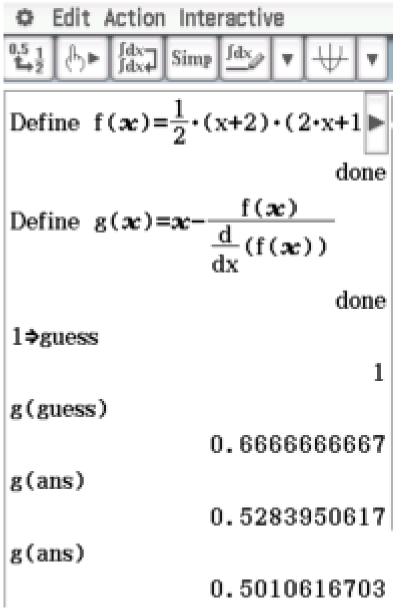
Both have converged to the actual root at $x = \frac{1}{2}$ quite rapidly.

An eActivity Implementation of the Newton-Raphson Method

As the Newton-Raphson method will be used on more than one occasion, you might like to define and use an eActivity which has the predefined steps set up. An example can be seen below, and copies of this eActivity called ‘nsolvnewton’ can be imported into your Classpad – instructions for this are supplied in Appendix 1.

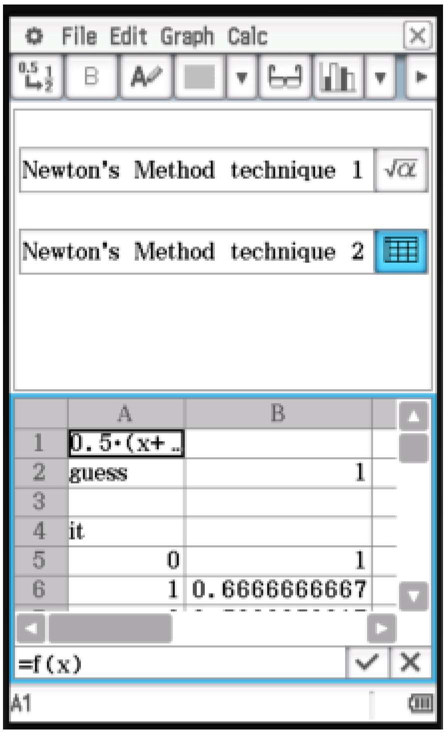
As was the case earlier with the bisection method, two strips are supplied. The first uses Main

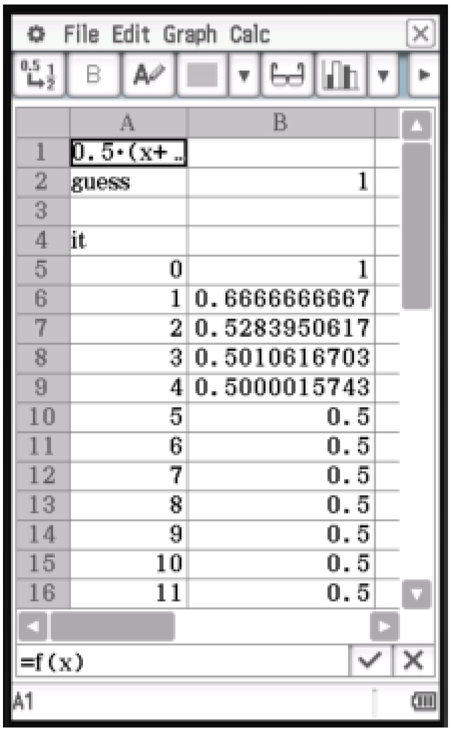




This method converges to 0.5 in 5 iterations.

Once again there is a second strip shown which leads to a spreadsheet, in which mode multiple calculations can be conducted rapidly.





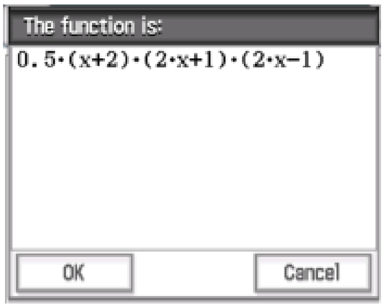
The function is sourced from the first strip definition shown previously..

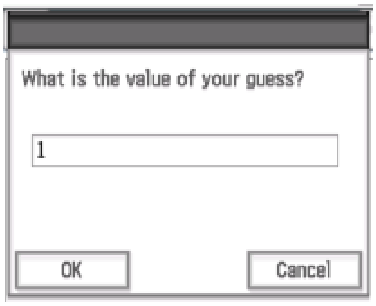
Numeric Approximations to Roots of $f(x) = 0$

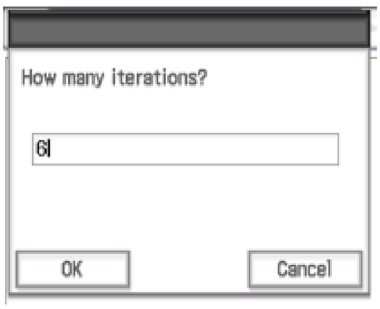
Program Mode Implementation of the Newton-Raphson Method

Having now seen the process of iteration being carried out automatically within a spreadsheet, again there is an option of a program which will replicate the process:

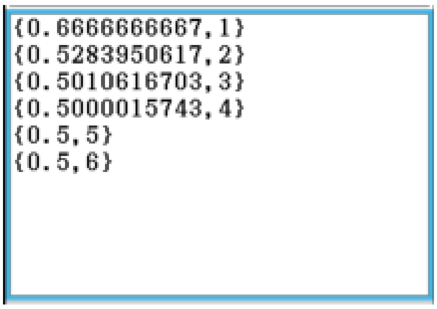
The program will remind you that the function for which a root is to be found must be predefined, then present options for an initial guess and a number of iterations to apply:







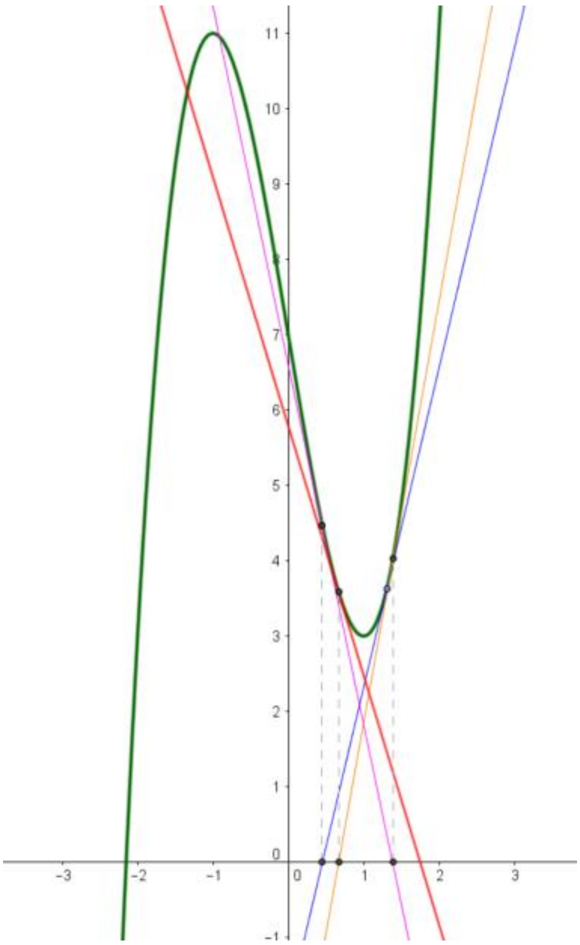
Working is presented, showing successive x_n values:



PMI Analysis of each method

The Bisection method will always home in on a solution for $f(x) = 0$ – it is a slow but robust method which will always provide a small window within which the root must lie if it does not land on the root itself, which can provide an answer correct to the desired number of decimal places. It does, however, demand that you have an idea of the location of a root, and that you can provide bounds within which it must lie.

The Newton-Raphson method is fast, but flaky. It can rapidly converge to the value of the root to as many decimal places as required; or it can fail. It specifically requires that the first derivative not be zero in the zone from its starting value to the root itself, and the second derivative must maintain a constant sign in the zone from its starting value to the root itself. In effect that implies the function ought to be either increasing or decreasing within the operating domain to be successful. It is also unreliable if the absolute value of the first derivative is large (steep gradient of a tangent).



At left is an illustration of the Newton-Raphson method in a scenario where it breaks down.

This is for the $f(x) = 2x^3 - 6x + 7$ where the initial guess is to the right of a local minima, with an inflection point between the initial guess and the root.

The x_n values oscillate about the local minima.

Another example would be $f(x) = x^{\frac{1}{3}}$, where successive iterations get further away from the root at $x = 0$.

As the numeric solutions to roots of functions were of great importance to mathematicians, and users of mathematics, especially in the days prior to the invention of technology to assist in that regard, much is written about the techniques described in this document, and further reading is recommended, especially in older mathematical texts.