Driving parallel-kinematic machines with Mach3 and Rhino3D

-Some basic details and a call to arms

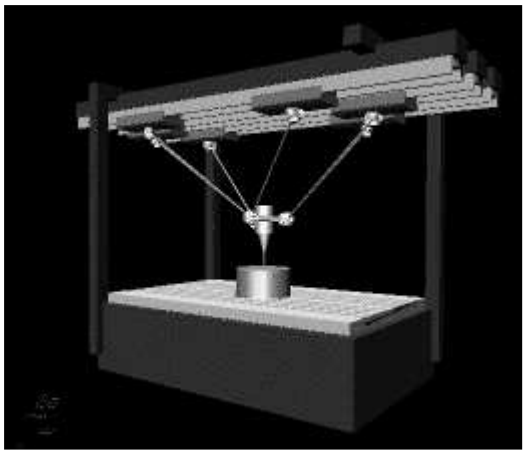
Introduction

Parallel machines have a fearsome reputation but does this have to be the case? The famous variable strut length hexapod is quite a complicated beast and its control and construction are not easy for the amateur. Making variable struts with any degree of stiffness is very difficult and that's before we get to the software. In this document I'm going to talk about a slightly different type of parallel machine that should be easier to build and a way of driving it with hobby type methods.

A small precision hexapod platform, ouch my brain hurts.

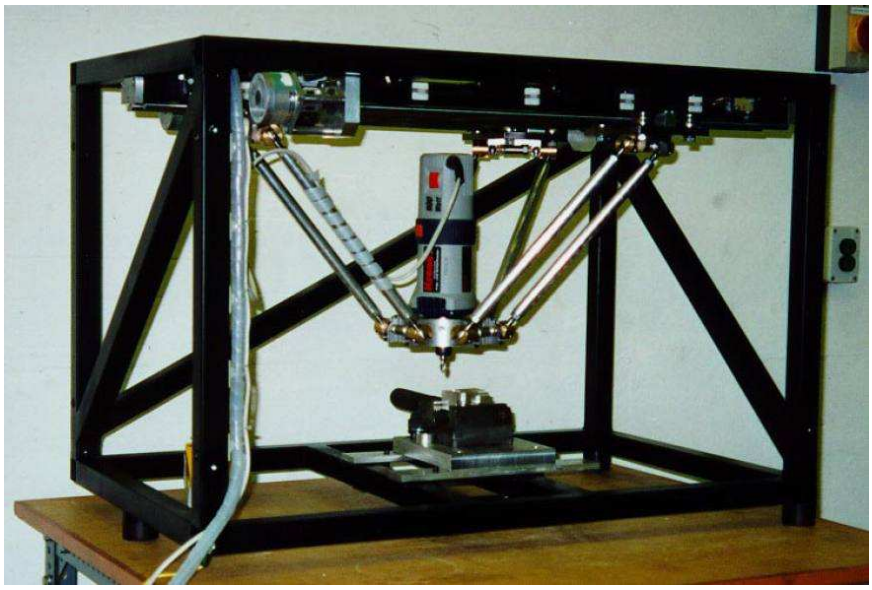
An easier machine:

How can you make a parallel machine without having to deal with the difficulties of variable length struts and complex kinematics. Firstly start with 2D/3D machines and secondly use fixed length struts. This leads to the "glide" range of parallel machines.



A hexaglide, much easier to build but still a tad complex.

The idea is simple, use fixed length struts attached between the moving part (table or tool) and a linear actuator, the position of the sliding elements determines the position of the platform. Of course the Hexaglide above is only one option, you can also build two and three axis versions. The triaglide has three carriages and a parallelogram type linkage on each; this is just to make the machine tool point in the correct direction (quite important).



This is a triaglide, it's a three axis machine using three linear actuators and fixed length struts.

In the figures all of the linear bearings are parallel but this does not need to be the case, you can make them fan out in a star or even put them up at an angle, this changes the working envelope the machine has.

A different configuration but still a triaglide, the working area can be tailored to suite the application. You can again see a pair of struts on each axis to keep the tool pointing down.

Fig. 2: The Sprint Z3 PKM tool head by DS Technologie GmbH.

If you ditch the parallelograms you get a three axis head that moves in the z direction and can be poised at "any" angle.

A simple example

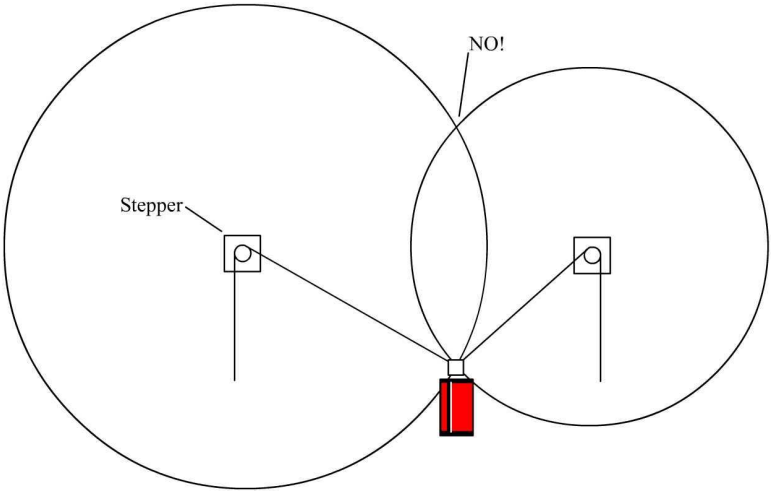
There is one parallel machine however which does have variable struts and is still easy to build, I think it has real appeal so I'm going to use it as an example.

This is Hektor in action, see http://www.hektor.ch for more details and art. The system works by hanging a solenoid operated spray can by two toothed belts that can be fed in and out by steppers. Gravity sorts everything else out.

Making equations, maths and the like:

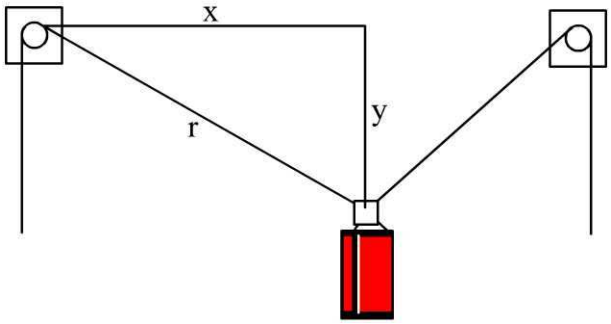
So you have built a machine like Hector or a glide type machine and you want to be able to control it. You need some equations that will convert the position you want to get to in x and y and convert them to a strut length or position.

You can see that the strut lengths for hector define two circles and it is where they cross that defines the point where the paint can is. Actually, when two circles overlap there are two places where they cross but for Hector one would require anti gravity!



Hektor's struts" force" the spray can to lie on the intersection of two circles but gravity chooses the lower intersection.

So let's do the maths. We know where we want the spray can to be so we know x and y. We can measure x and y from where we feel is appropriate, to make things easy lets make (0,0) the left hand motor.



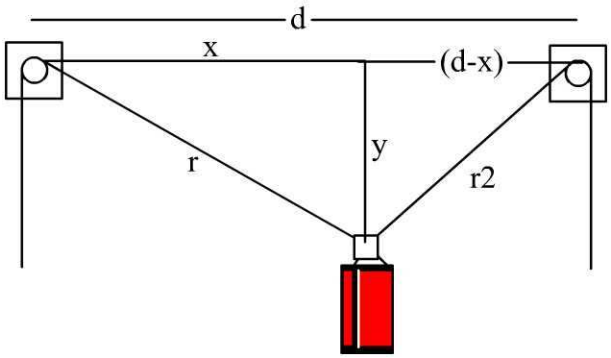
OK so we have an x and y and we want to know r for the left hand motor. Pythagoras says that:

$$r^2 = x^2 + y^2$$

so

$$r = \sqrt{x^2 + y^2}$$

We can produce a similar equation for the other "strut" but we must remember to keep the origin in the same place.



$$r_2^2 = (d-x)^2 + y^2$$
$$r_2 = \sqrt{(d-x)^2 + y^2}$$

Now we have two equations which we can use in the Mach3 CNC control software, we'll come to that later.

You may have noticed a slight circular argument, "We know where we want the spray can to be so we know x and y" but to start with do we know where we are? Err no.

In the case of the Hektor machine there is a very simple "calibration" procedure that can be used; one of the videos on the site shows how it works.

Basically you pull in one of the cables until the can hits a limit switch, now zero that strut length; you obviously need to make sure the other strut is nice and loose. Now do the same on the other side and assuming you know d you can move the machine to your zero in x and.

The same sort of procedure can be performed for any machine and it is not that dissimilar to zeroing a conventional serial machine only the tool position in a specific axis is related to both the actuators used.

Mach3, the equations feature and what it does and doesn't do:

Mach3 is a windows based CNC controller for parallel port based step and direction control. It's a rather good bit of software and very economic. A feature quite recently introduced is that of equations, it is possible to define an axis' position in terms of the position of other axes or even its own position. It was incorporated to allow software machine squaring, basically as the tool moves in one axis it may move somewhat in the other if the axes are not entirely square.

The relationship is something like: $x = x + ky$

So the position in x is as you want it to be (equal to x) plus something that depends on the y position.

So in the equations section of Mach you type f(x) = x - 0.01y or similar (note the minus). In this example the tool moves 0.01units in x for every unit of y. Where you type equations we will cover later.

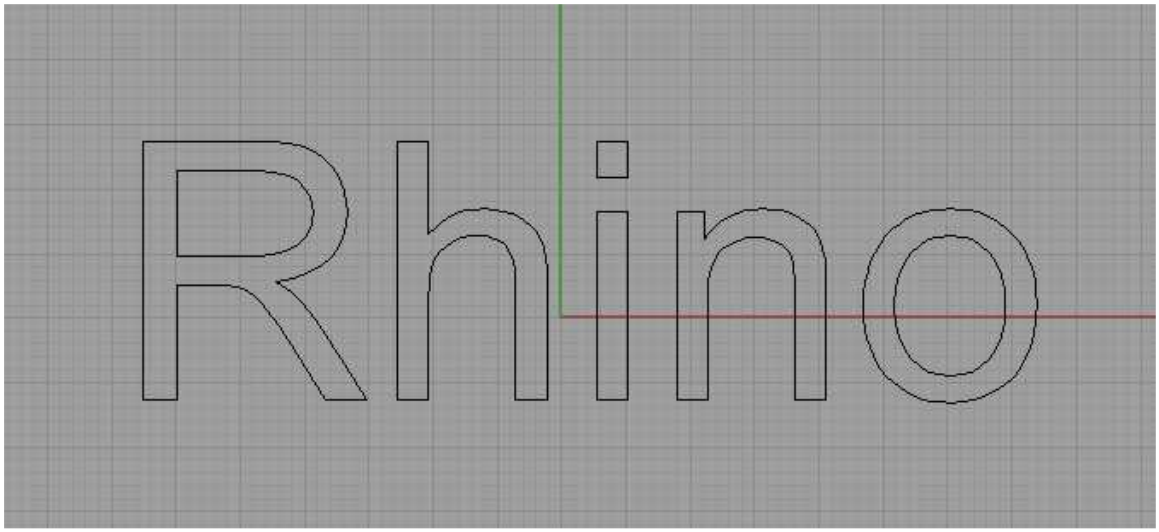
The thing is that the use of axis squaring still assumes that each axis move is a nice straight line. If for example the machine above was at 0,0 and you wanted to move to 100,0 then without squaring it would move to 100,1 in the physical world. With squaring as long at the machine aims for 100,-1 it will end up at the 100,0. It doesn't matter a jot about the equation between the start and end points of the move as the move is a straight one.

The problem for our kinematic machines is that the equations need to be obeyed all of the time even on straight line cuts, it is very much like we have curved axes. To allow us to do this we have to split up our tool paths into segments. The machine will physically move in small curves between the ends of these segments but if they are small enough it doesn't matter.

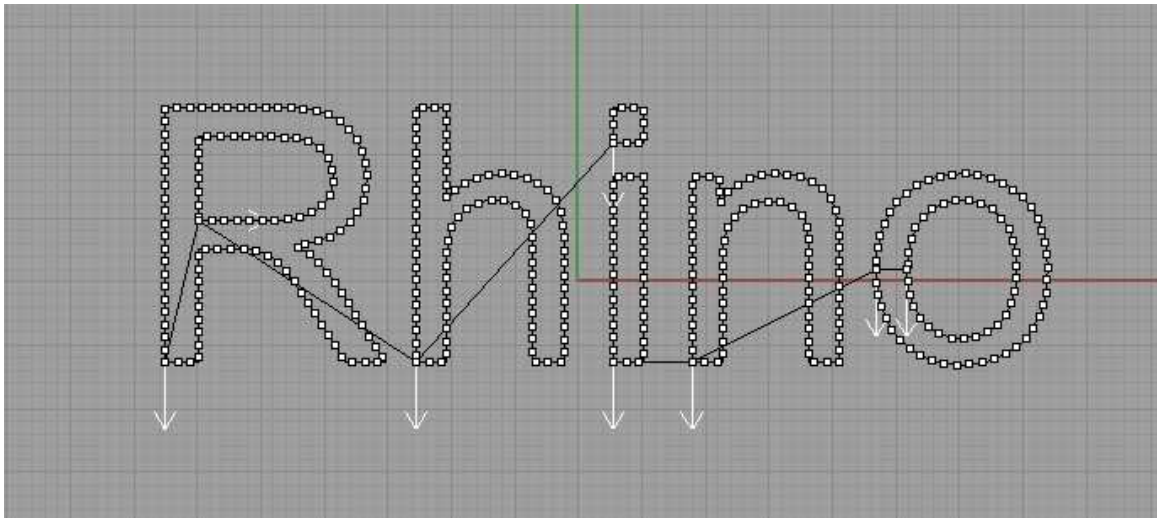
So before returning to Mach to implement the equations let's just split a tool path or rather the CAD drawing of the tool path. For this I am using Rhino3D but most CAD packages should be able to do similar.

Spliting tool paths in Rhino:

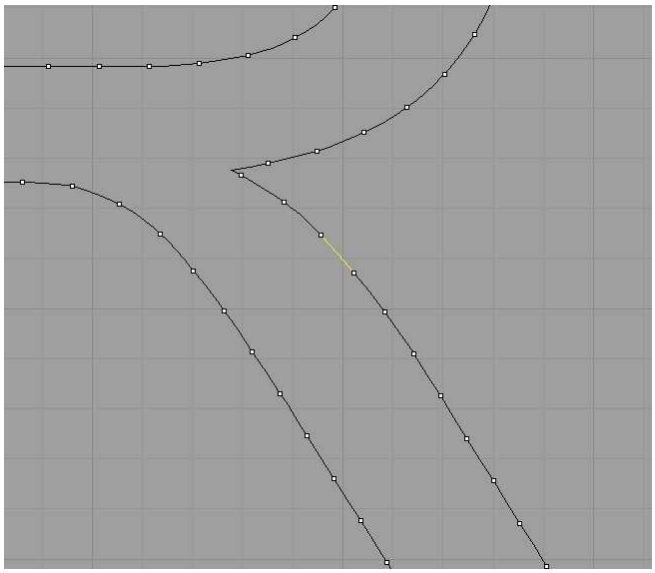
Draw something; this is just a text object consisting of straight lines and arcs:



Select the lines and type divide in the command line and accept the seam adjustment. Type L and press return to allow the segment length to be defined and then enter the segment length you want.



The drawing is now covered in dots that are the distance apart you specified. We will use these to split the drawing into little segments. Type split and when asked for the cutting object select all of the points by dragging a box or type SelPt and press return. Then press return and the lines will be split up

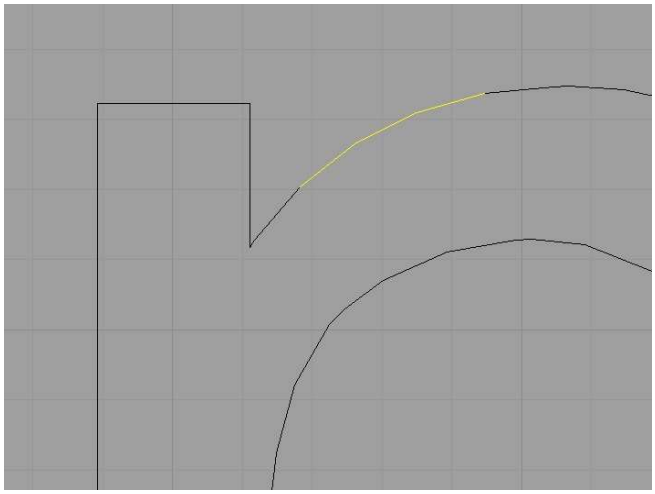


To get rid of all the dots use the SelPt command again to select them all and press delete.

What we have now is the drawing split into segments but these segments may be curved, it won't hurt but as the machine will ignore the equations between the ends of the segments they might as well be straight and possibly faster to process.

To make them straight select them all and type ChangeDegree, make the new degree one.

Then finally select all the lines and type explode, you now have a drawing made up entirely of straight lines of the length you specified or shorter.



Now you need to convert the drawing to tool paths, do that with your normal CAM package or import as a DXF into Mach directly.

For a 3-axis machine it is going to be trickier, at the moment the only solution I can suggest is to use the CNC toolkit, that will draw a 3D tool path and subdivide it:

www.rainnea.com/cnc_toolkit.htm

Setting up Mach's equations

Warning – The equations are ignored for jogging, they only work for g-code commands so to make the machine behave as you want you must run from a g-code file or from the MDI.

In the equations dialogue box you have a box for each axis where you can insert an equation. For our parallel machines there are two ways of setting the equations up. If we return to Hektor we can connect one of the motors to x and the other to y. The equations end up being:

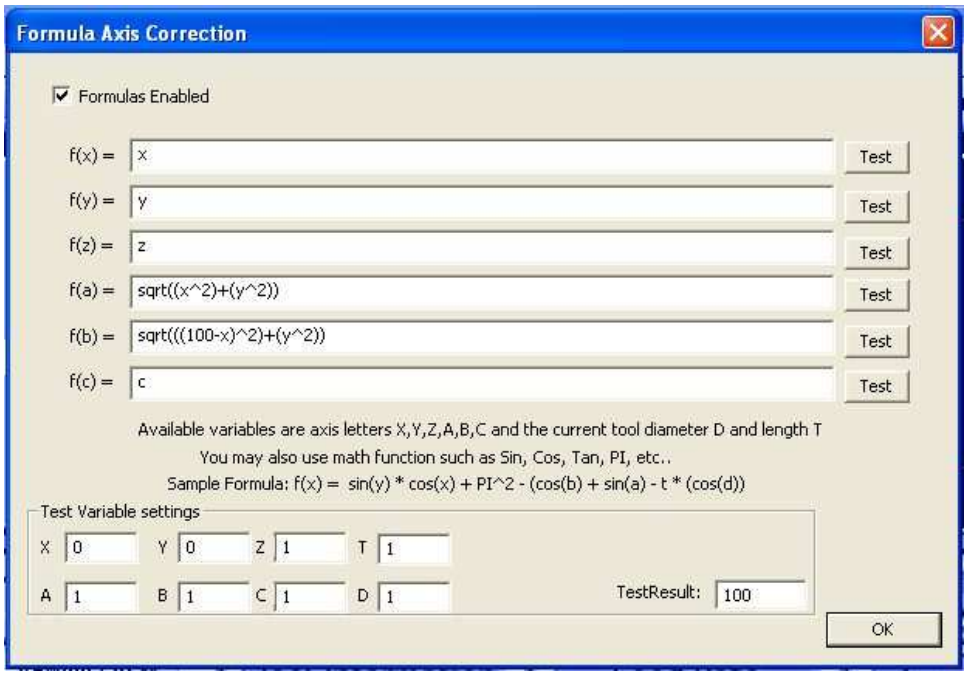
$f(x) = sqrt((x\text{^}2) + (y\text{^}2))$
$f(y) = sqrt(((d - x)\text{^}2) + (y\text{^}2))$ (but actually put a number in for d)

This is sort of weird though isn't it because apart from anything else what you see in the DROs is not x and y lengths but the strut lengths and when it comes to feed rate it's anyone's guess.

The better way is to define two different axes such as A and B as the strut lengths

$f(a) = sqrt((x\text{^}2) + (y\text{^}2))$
$f(b) = sqrt(((d - x)\text{^}2) + (y\text{^}2))$

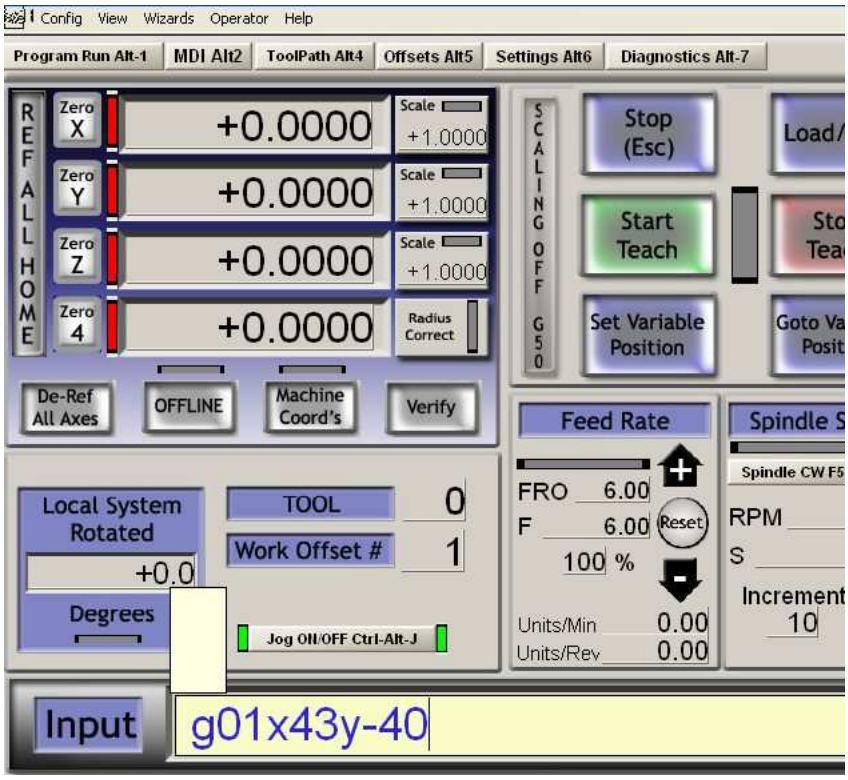
Now the x and y DROs will react as expected as will the feed rate as long as the A and B axis can keep up.



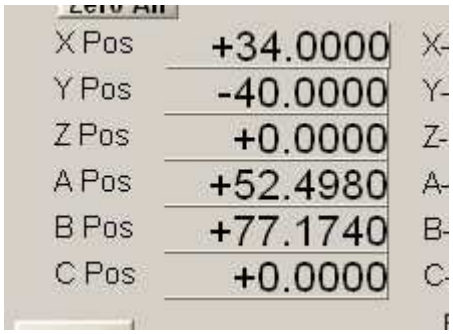
Here the equations have been entered into the dialogue box and the enable formulas box checked. With x = 0 and y = 0 in the variable test settings I get zero when I click test for A and 100 when I press test for B. This is because d was set to 100.

Once you have inserted your equations into Mach the next thing is to test them. Enter some test values for x and y into the boxes and press the test button next to the axis you are interested in. For the less mathematically inclined and for more complex mechanisms producing test numbers in CAD is an easy exercise.

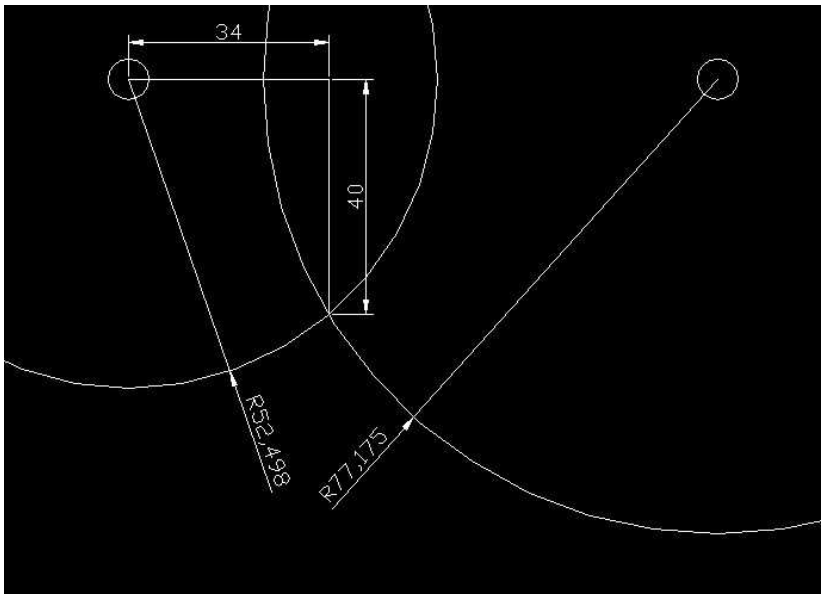
So as a final test for these equations I will move the machine to x = 34 and y = - 40



First enter a command into the MDI.



After the move the x and y axis positions are correct but what about A and B ?



A quick sanity check in CAD shows that A and B are as they should be.

A few notes:

When you came to use this machine, all feed rates would relate to x and y movement even though the x and y axis are not assigned (directly) to any motors. You should set up the A and B axis as just straight forward axis to start with to get max pulse rates and accelerations. A and B are effectively slaved to x and y so x and y dictate the speeds although A and B need to be able to keep or they will lower the feed rate.

A work in progress

This document is a work in progress, I have been playing with parallel machines and I think more people should too. I hope I have dispelled some of the fears involved and provided a means for anyone to consider at least 2-axis machines.

Any comments or requests for additional material should be sent to eexgs@nottingham.ac.uk