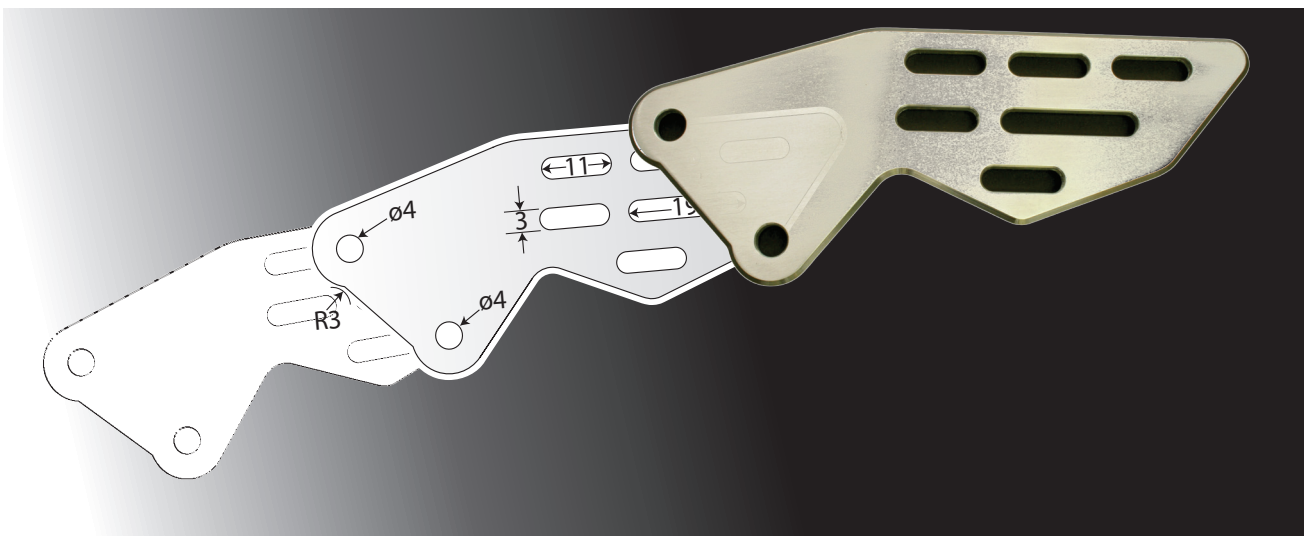
CNC in the Workshop



Part 12

Note: excludes the second half of this article, which consists of the beginning of the Etch Folder. The material on the Etch Folder, which appeared in MEW issues 214 to 218 is now contained in a single document.

Part 12

In this part of the series, we look at interpolation. Originally, the article then included the first part of the material on the Etch Folder project. This was done for editorial convenience, to fit the space available. However; it makes more sense to collect all the material for that project in one place, so it now appears in a separate document.

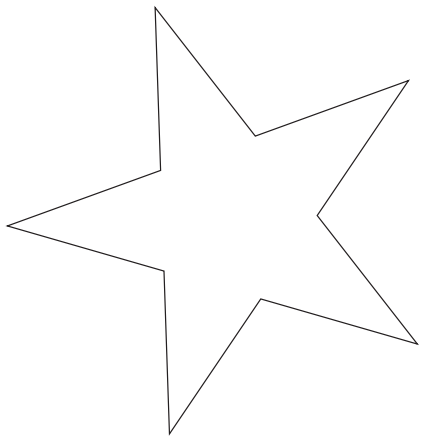


Fig 57: A star shape and the corresponding path between the points.

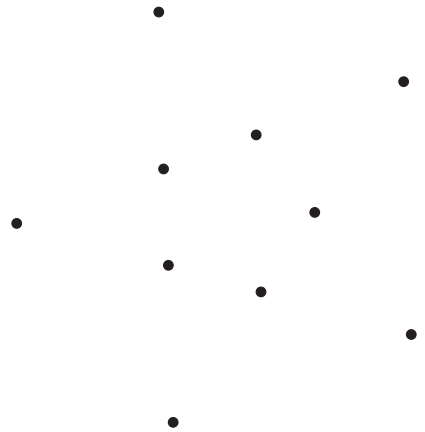


Fig 58: Points on a shape, which need to be joined to form a path.

TIME IS MONEY

It's often said that time is money, and I suppose that's true, at least in industry. The NIST core software at the heart of Mach3 was, of course, written for industrial use more than anything else, so it takes a pragmatic approach to movements of the Controlled Point (CP).

So far, we have dealt with direct MDI commands and small programs, with just a few lines of code. That's fine, and we will continue to look at how much can be done with some very simple short programs, because there is much to learn from that.

But the more ambitious we become, the more we need to be able to have our CNC software process large numbers of instructions and significant quantities of data. Bearing in mind that it takes a finite time to execute an instruction and make a corresponding movement of the CP, that suggests some programs will take a very long time to run. In fact, even short programs are capable of generating huge numbers of movements and requiring the machine to run for ages.

Industrially, that represents a significant cost, and in the home workshop it is preferable if the machine stops running before we fall asleep, so there has always been a need to make pragmatic compromises in what the machine is actually being asked to do. One of those compromises is about 'necessary' accuracy.

There is always the assumption that CNC programs and machines are super-accurate, but that is not necessarily the case. Even if they were, that doesn't mean they provide a perfect and workable solution for all jobs.

When a CNC program like Mach3 runs, the core interpreter looks ahead at what moves are coming up next. In fact it looks ahead several commands. Then it tries to figure out the best way to make those moves. One way to proceed is to assume the programmer knew exactly how the CP should move, and simply make each of those moves. Making a move involves mechanics, though, and that's where things begin to get less straightforward. Mechanical parts have mass and if they are moving they have momentum, which is the tendency to want to keep moving in their present direction. To get a slide moving from rest, the program needs to accelerate the slide as specified in the initial settings for the steppers; then it needs to maintain a constant speed during the move, until the end is within a specific number of steps; then it needs to decelerate so that the slide comes to rest at the point specified in the command that is being executed. That's all fine, but in real life one command follows another, in a program, and there is little point in decelerating towards an end point if the next command will simply accelerate, move at constant speed then decelerate in more or less the same direction. There is a significant efficiency gain, both in terms of time saved and in accuracy achieved, by looking ahead for a few commands, and seeing where the accelerate-decelerate-accelerate sequence could be omitted or at least modified. That seems like common sense, and it is.

But there's a bit more to this. The points the CP will visit during a program constitute a path. In fact, the CP may follow more than one path during a program, so for our purposes, a path is a continuous sequence of points between an initial point (the start of the path) and a terminal point (the end of the path). If you were drawing a shape to be machined, a path would be a line you could draw without lifting your pencil. When you have to lift the pencil, that's the end of the path.

Paths are a useful concept because we can turn the process on its head and ask whether there is a path between a set of points we want the CP to visit, and, if there is, whether we can optimise the path by taking some shortcuts.

Take the shape shown in fig 57 for example. It's a star shape and there is an obvious path from one point to another around the outline, in and out repeatedly. If you imagine the points without the line (fig 58), the question

is: can we find an optimum path between those points? Of course we do want to produce that shape, so we can't join the points in any old order, but thinking mathematically, can we approximate to the shape without necessarily having to visit each point exactly? If we can get reasonably close to each point without having to land exactly on each point, we can avoid the abrupt changes of direction and remove some of the accelerate-decelerate cycles or at least reduce their effect a little, and speed up the machining process. One of the other effects of the accelerate-decelerate sequence is that our carefully calculated optimum cutting speed is varying during those periods, and that variation in speed may mean a poor cutting action and a varying quality of finish on the work. So even without optimising the path, we are in danger of reduced quality. In the real world of machines, there are lots of necessary compromises, and this is one of them.

Back to our points

Smoothing out the potential path just a little, and removing the restriction that the CP has to visit each point precisely, gives a range of increasingly rounded paths, some of which are shown in fig 59.

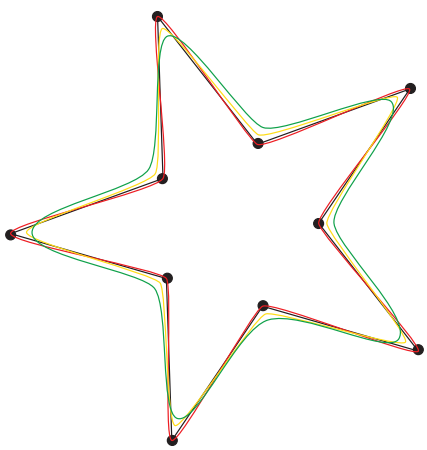


Fig 59: Possible paths between the points frm fig 58.

The mathematics of interpolation is fascinating, and if I understood anything of it I would gladly share that knowledge with you. Interpolation is the process of fitting the best path to a set of points, and that's what we are trying to do here. Visiting each precise point is one way, but going close to each point, without necessarily having to land exactly on it, can produce a path which is a better match for the need to smooth out the variations in acceleration and deceleration, so achieving a more constant and more optimum cutting speed (i.e. the speed of the CP between points on the path).

For some jobs, we need to visit each point exactly, using what Mach3 calls Exact Stop mode. For other jobs, we can allow a defined amount of deviation, in an attempt to get close to a constant cutting speed, using Constant Velocity mode.

There is an up side and a down side to both of these modes, and although you might think Exact Stop is the obvious mode to use, that's not always the case.

If you are running a job where a precise shape doesn't matter, or you are doing a preliminary operation involving sweeping cuts across and area to be cleared, Constant Velocity makes more sense. Where you require a precise shape, but are willing to live with a compromised cutting speed, Exact Stop mode might be the best choice.

It all depends on what you will do with the shape (fig 59). If the workpiece requires a precise outline, perhaps to mate with another part, Exact Stop might be best. If you are making 100 of them, and the outside shape has a wide tolerance, Constant Velocity has a lot to commend it.

Fig 60 shows one line of an engraving pattern, and the overall pattern comprises 500 similar lines. Constant Velocity allows some deviation from the path, but if all lines are the same and are cut with the same deviation, who will notice? At 500 repetitions, there may be a substantial saving in time, and the surface finish might be a little better.

To invoke Exact Stop mode, use G61

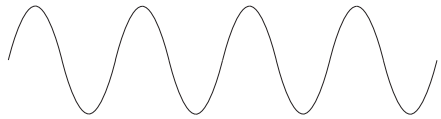


Fig 60: One line of an engraving pattern.

To invoke Constant Velocity mode, use G64

Only one of those commands can be in effect at any one time, but the modes can be changed from within a program, so you could do a set of commands while G64 is in effect, then switch to Exact Stop mode using G61 and issue some other commands.

The path control mode commands G61 and G64 are modal commands. That means only one of them can be in effect at a time, but that command will stay in effect until it is changed. There is a note on this in the Mach3 manual, in section 10.6. Mach3 uses several modal groups, and they are listed in the manual in the table under 10.6.

The best way to treat the path control mode commands is to set the default in the Mach3 menu: Config > General Config

The second column from the left has an area entitled Startup Modals, which we have met before, and within that a choice of Motion Mode settings (fig 61).

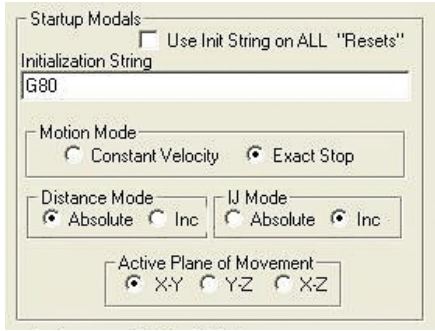


Fig 61: The Motion Mode area of the Config > General Config menu.

Choose your preferred default, Exact Stop or Constant Velocity. My own preference is Exact Stop, especially since we can change to Constant Velocity at any time. It's just that

when Mach3 starts up, it invokes the mode set in this Config panel. Setting the mode here means we know how Mach3 will treat Path mode unless we use the other command.

If you look now at the Initialisation Sequence suggested for the start of all programs, in MEW 209, you can add G61 (or G64) to that sequence. That way, we know that if I write a program and you subsequently run it on your machine, you should get a predictable result because I have set the motion mode appropriately.

So the Initialisation Sequence then becomes:

G17 G21 G40 G49 G54 G61 G80 G90 G91.1 G92.1 G94 G98

Now look at the Config > General Config menu screen, and the section just above where you set the default motion mode. There's a box for an Initialisation String, and it probably just has G80 in there, to begin with. You could always enter our Initialisation Sequence in there; then you would know exactly how the software should behave, when you start it up ready for work. If you are running a program, it won't matter, because you will include the Initialisation Sequence in every program, but if you are using MDI commands, it is a comfort to know that all will be well in the digital world, right from the word go.

There's a check box above that area, and it allows you to specify whether Mach3 will carry out the Initialisation String every time you do a Reset. That might be a good idea, because then you know how the machine is supposed to behave afterwards.