FAB500016

The benefits of reading the manual - subtractive machining with a Kuka arm

Stefan Knorr

designing.berlin

Learning Objectives

1. understand the basic concept on how robots work, what robots can do and why they can do it
2. differenciate a 5 axis mill and a 6 axis robot with regards to their capabilities
3. be able to generate 6 axis fusion360 toolpath that are ready for be executed on a Kuka KRC controller
4. understand the downsides of using a robot arm for milling applications.

Description

Robots are the most univeral machines in terms of manufacturing. They can manipulate objects or manufacturing tools in a big work envelope at impressive speeds and with significant precision. That comes at the cost of being harder to configure / use / program. As the CAM tools such as Fusion 360 are improving on a regular basis, the potential to use robots in the machining world is steadly growing.
The initially published postprocessor for KUKA (KRL - kuka robot language) is a first step to bring the power of Fusion 360 to a milling robot. Based on that post, I developed and tested a postprocessor with more features for milling applications.

Since robots are intrinsicly prone to vibrations, all cutting path should be as smooth as possible. This is achived by bypassing the regular motion planning algorithm in the Kuka contol. The results are very promising.
This class highlights a few details on robot milling based on personal experiences by the speaker.

The handout includes some detailed information sections in gray. You can skip these unless you want to really dive deep into the corresponding topic.

About the speaker

Stefan Knorr is an engineer from Berlin, the capital city of Germany. He completed a Dipl. Ing. Physikalische Ingenieurwissenschaft with honor at the Technische Universität Berlin. In his past work at Charité and Biotronik GmbH he further developed his engineering skills, trained the creative parts of his brain and improved in debugging soft- and hardware.

He got dragged into digital fabrication once the first Makerbot hit Berlin.

Stefan was the initial CNC class teacher in Fab Lab Berlin and got involved into reseach projects later. A first project covered intense 3D parametric modeling with Fusion 360 in the realm of prosthetic hands. The second project goal was non-planar FDM printing with PA based materials using a Kuka arm. That involved the development of software and hardware to perform slicing, extrusion and robot control.

In 2011 Stefan started the company 'designing.berlin' and offers constulting and hands-on for various stages of customers brainstorming, researching, designing, developing, prototyping, testing, fabricating, assembling, optimizing. In house manufacturing capabilities cover manual machines as well as a 4 axis high speed mill, 7 axis Kuka robot arm and FDM printing.

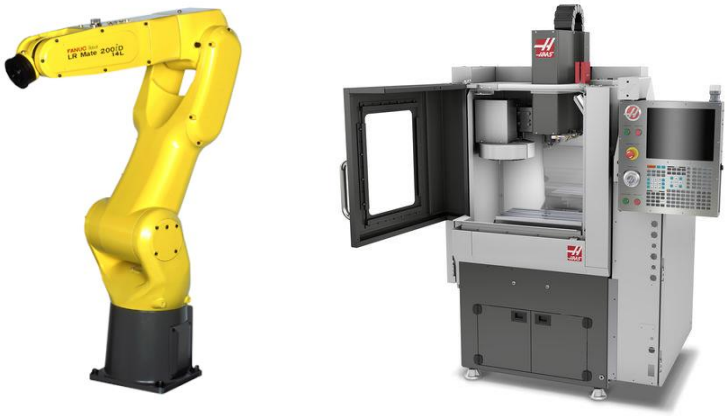
Robot arm vs. CNC machine

What are the differences?

What makes a robot arm so different from a CNC machine?

The answer can fill plenty pages. Lets focus on the relevant aspects:

Robot arm	CNC machine
• Only rotary axes, usually named A1 to A6 starting at the robot base • Rather flexible and prone to vibration • Big work envelope • High dynamics with minor precision • Thermal drift during operation • mainly build for handling / welding	• Linear axes for XYZ and optional rotary axes • Rigid geometry • Rather small work envelope • High dynamics with precision • Thermal control is available • build for machining



Random robot and CNC examples

The most relevant differences with regards to milling applications are the general machine rigidity and precision. Both are related by the underlaying machine kinematics. Just looking at the robot arm, the lack of rigidity is obvious: 6 serial rotary joints and 2 rather long arms don't look rigid.

It's easy to check that by putting an indicator at the robot flange and pushing it by hand in various directions. The needle will move. The same test will fail on most CNC machines.

Doesn't sound good? Yes, but robot manufacturers put a lot of effort in their controllers to compensate for some of the aspects, like deflection by gravity and dynamic load. Once you

specify the tool weight, center of gravity and inertia, the controller will take that into account and do its best still be accurate during motion.

In fact the accuracy of modern robots is rather impressive. Using and indicator to ‘draw circles’ on a surface plate will result in deviations below 0.1mm (4 thou).

A similar test while manually jogging the robot in along a line and back with an 0.001mm indicator reveals deviations in Z below 0.06mm. (During manual jogging the controller does not apply and weight compensation.)

Moving an indicator along a line with a robot on a granite surface plate

Robot specials

There are a few special aspects about robots kinematics that are robot specific. Lets have a look at some:

Multiple poses

Imagine you want to move a 3 axis CNC machine to a given position in its work envelope. There is only one possible combination of X, Y and Z values to get to that point. Try the same with a robot and you might be surprised, becaue there are multiple combination of the joint angles A1, A2, A3, A4, A5 and A6 to reach the given point. Here is an example of the same robot reaching a point with different poses.

So how to chose the pose? In the Fusion 360 workflow to control the robot, the pose selection will happen indirectly by selecting a suitable start position. Like a park position for a CNC machine. The pose will not alter by running a CNC toolpath.

Singularities

You might have heard the term, but what does it mean? It's a special joint configuration that perfectly aligns two rotary joints. In that configuration the robot looses one degree of freedom and turns into a 5 axis robot.

In my experience the A4-A6 singularity is the most common one, but not the only one. Here a visual example for a move that passes the A4-A6 singularity.

Imagine moving the robot along the curve from one blue configuration to the other. In the process the joint A5 aligns A4 and A6 onto the same line, just like the orange configuration. So what will happen? Instant robot halt? Blue screen on the controller? Thunderstorm?

Nothing like that. The robot will move along the curve, but the actual velocity of the tool along the curve will slow down significantly, while A4 and A6 start rotating fast in opposite directions, only limited by their maximum velocity.

That behaviour near singularity configurations is not wanted. It gets worse the closer the robot gets to the actal singularity.

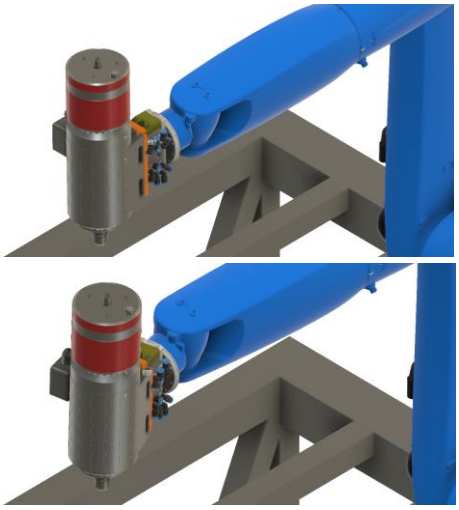
Imagine you are milling and the feedrate suddenly drops. Or you dispense glue at a constant extrusion rate. Not good, but there is hope …

Extra axis – one more degree of freedom

Do you care about how the milling spindle is oriented along its axis while machining? You might in special cases like probing with a calibrated 3D probe. Otherwise not, because the spinning endmill has no preference around it's axis.

The robot arm itself has the ability to rotate the spindle around its X, Y and Z axis. Assuming the Z axis aligne with the spindle axis, the rotation around the Z axis is ambigious. So it can be choosen it to whatever value – it won't influence the milling result.

Well, than let's make a clever choice and stay away from singularities !



The first picture shows a milling spindle and the robot in a A4-A6 singularity. The second picture shows the spindle at the same position, but slightly rotated around its Z axis to avoid the A4-A6 singularity.

Making a clever choice for the spindle Z rotation lets you steer away from singularities. That's neat

External axis – turntable or linear rails

The additional degree of freedom to reorient the spindle can be utilized to stay away from singularitie. In a similar fashion additional axis like turntables, swivel heads or linear rails offer more degrees of freedom. Additional degrees of freedom can be utilized to achive certain goals. That can be milling in a certain spindle orientation for gravitational chip removal.

Integrating external axis into a post processor can be turn into a complex business, depending on what is to be achieved with the additional degree of freedom. Most CAM toolchains historically don't have to deal with ambigious degrees of freedom.

How to move a Robot?

Prepration

Initially a robot is a machine that only knows about itself. All the joints, their axis limits, the arm length, the complete kinematic chain. Assuming the robot base is fixed in space, the robot can move its flange freely to points within reach. It just requires a given the target position and orientation.

A basic robot milling setup will include a spindle mounted to the robot flange. Additionally there will be and some kind of workholding mounted in relation to the robot base. That can be a vise, vacuum table, T-slot table or zero clamping system.

Before you can move a robot like a CNC machine you have to teach it the position and orientation of your workholding (called BASE) with regards to the robot base and you have to teach the position and orientation of the spindle (called TOOL) with regards to the robot flange. The robot controllers provide different methods to teach the robot a BASE and a TOOL coordinate system. You can store multiple BASE and TOOL configurations and activate them as needed. Multiple BASE systems can represent different stationary vise jaws, like different work offsets on a CNC control. Different TOOL positions can be used to represent different milling tools. Often the term TCP (tool center point) is used in robotics to refer to the tool tip.

> Teaching the BASE and TOOL position are really critical steps for the precision of all proceeding milling operations. Any positional and orientational deviations should be minimized. You should at least verify your results with proper metrology to know what to expect. Custom teaching routines using indicators or 3D touch probes will be required to gain top notch calibrations results.
>
> This whole procedure relates to tramming a spindle and squaring the axes on a 3 axis CNC machine. It is as critical for precise operations.

The spindle weight and center of gravity have to be specified to allow the controller to compensate accordingly. With a spindle mounted to the flange, the joint limits for A5 should be set tighter to avoid the spindle hitting the robot arm itself.

Move a robot by joint angles A1 … A6

The most intuitive way to change the robot position is to move one joint at a time. With a teach pendant you can jog each joint manually, just like you can move X, Y or Z on a CNC. In order to

familiarize with a robot and its kinematic you should move a robot in that mode manually. It really helps to get a good basic understanding.

Moving a robot by altering joint angles does not require any BASE or TOOL definitions. Singularities can are no issue, because no inverse kinematics will be solved.

You can also use joint based moves in the KRL code like:

```
PTP {A1 10.0,    A2 -80.6,    A3 -50.0,    A4 0.0,    A5 14.2,    A6 0.0}
```

The robot will move the joints to the given target positions. You can specify only a subset of joint to move as well:

```
PTP { A6 180.0}
```

The robot will orient the flange in the opposite direction without moving any other joint.

Move a robot by carthesian coordinates X, Y, Z and angles A, B, C

Once a BASE and TOOL is defined and selected, you can move your endmill (TCP) in the BASE coordinate system. The robot controller will do all the required inverse kinematics, apply tool offsets and move all 6 joints at the same time to perform the requested move.

You can use the teach pendant to jog the TCP along X, Y or Z and the tool will magically move in just one of the directions in space without changing the spindle orientation. All 6 joints move at the same time at varying speeds. Rather magic.

Assume the selected BASE1 point represents a static vise jaw and X is aligned along the jaw.

```
PTP {X 0.0,    Y 0.0,    Z 10.0,    A 0.0,    B 0.0,    C 0.0}
```

Moves the endmill 10mm above the vise jaw corner, pointing downwards.

```
LIN {X 100.0}
```

Moves the tool along the vise jaw at the same Y and Z position without altering the orientation. Just like g-code. Add some more lines and you can square your stock! How cool is that. Now imagine you have a second vise jaw at BASE2. Activate it and run the same squaring-stock code. The robot will move to the second vise and square stock there. Might sound very familiar .. G54, G55, G56 …

> A technical detail about what the values A, B and C stand for. They are angles to define the orientation at a given point X, Y, Z. The unit is degree. A, B and C are called Euler angles and they are a very common way to define 3D rotations in space. There are many different ways on how to use Euler angles, all sharing similarities. Here is how Euler angles are used in the Kuka control. Lets assume a given BASE and we want to move our TCP to the location {X 200.0, Y500.0, Z100.0, A 45.0, B10.0, C -20}. The translation in X, Y and Z should be easy. Just move 200mm along positive X, 500mm along positive Y and 100mm along positive Z. Now you are at the new TCP position.
>
> And the orientation?
>
> 1. Rotate the tool 45° (A value) around the Z axis in positive direction, following the right hand rule. Now you have a new orientation for the X and Y axis.

2. Rotate the tool 10° (B value) around the new Y axis in positive direction. This results in a new Z axis orientation and the X axis orientation changed again.
3. Rotate the tool 20° (C value) around the newest X axis in the negative direction. The Y and Z axes orientations change into their final values.

Now the tool is located at the specified XYZ position and is oriented according to the ABC values.

I can't emphasise it enough: once the robot BASE and TOOL definitions are done, you can use the robot just like a carthesian X, Y, Z machine. Move along X, Y or Z, move all at the same time, move along a circle, move along anything. The robot controller does all the magic math to move all six at the same time. It really makes it easy.
In fact there are many robot applications, where the robot newer changes the tool orientation and just acts like a big carthesian XYZ machine:

- Palletizing robots grab a bottle from the conveyor belt and put it into a matching crate.
- Glue dispensing robots use a nozzle that points down to the target object.
- 3D printers use a robot arm to gain a huge build volume. Most of them only print parallel slices.

Remember the multiple poses I mentioned earlier? Different joint angle combinations A1…A6 allow the robot to reach the identical position and orientation. Now imagine you excecute a program twice and you see two different poses? That would be a strange thing to happen, but it won't. Here is why:

The controller expects an initial move command in the program to define a certain robot pose, otherwise you'll get an error. That initial move command is defined by the six joint angles. From that point on the pose is implicitly defined.

I usually manipulate the robot manually by joint angles to define a suitable start position for a certain job. Hereby I define the robot pose for the proceeding job. Being suitable includes several aspects such as

- clearing the workholding to allow easy workpiece handling
- serviceability of the tool in the spindle
- spindle orientation being close to toolpath start orientation
- enough axis range, especially for A4, A5 and A6

The chosen start position is saved in the controller as a HOME1 … HOME16 position. Any HOME position can later be selected as the initial target position in the post processor options.

Why reading the manual?

That section refers to the title of the talk. Why should you read the manual?

A robot is a complex machine and most details are not obvious or intuitive. There are plenty of technical terms and concepts in the robotics world that you might not be familiar with. Some aspects are manufacturer specific. While reading the manual you might:

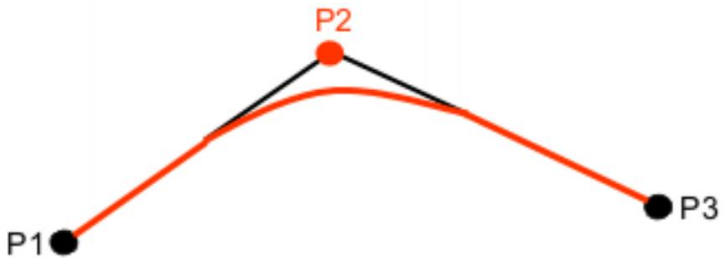
- Familiarize with the used terminology
- Discover programming or motion commands that fit your application
- Identify suitable addon packages to help you out
- Gain deeper understanding for your machine
- Understand interfacing with hardware extensions, like digital or analog I/O

Depending on your intended robot use, you might need some rarely used features that the robot control offers. And some might be excluded in the training sessions you participated. So why not reading the manual?

What did I read? – personal history

Honestly I did not read the full documentation before using a robot arm. I had a really good hands on training and that served as a good knowledge foundation to get started. Slicing and 3D printing kind of worked and the turntable was doing its trick. So far so good.

I got used to the various approximation settings in the control. They define how close the robot gets to a given waypoint while passing on to the next waypoint. At the extemes you can force a full stop at each point or define an allowed deviation from reaching the point to maintain a certain minimal velocity. The settings were always a compromise between precision of the toolpath and reaching a rather constant extruder velocity. Constant nozzle speed reduces induced vibration and helped to get consistant 3D printed object.

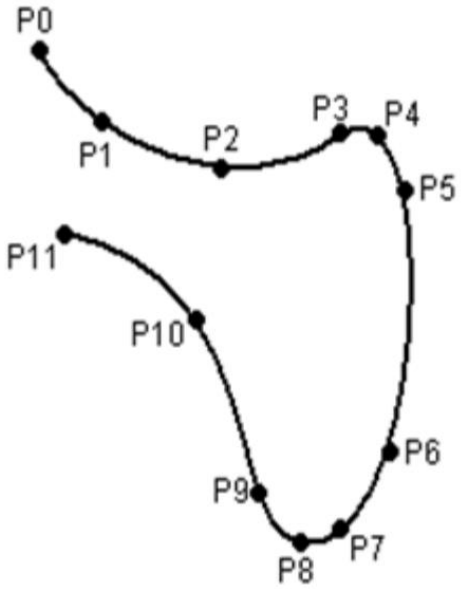


The picture shows a typical approximation artefact when the robot moves from P1 via P2 to P3. Long polyline toolpath will have a similar artefact at each waypoint.

> The size of the artefact usually depends on the excecution speed of the robot. At low velocity the spatial deviation will be small - the robot can get very close to P2, while still maintaining that low velocity. In a similar fashion the spatial deviation will increase at higher speed to avoid slowing down.

> Alternatively the robot can slow down at each waypoint to get closer to it. Running the same toolpath at different robot velocities will result in slightly different toolpath due to the approximation artefacts. I didn't like that.

At some point in time I had to integrate some external hardware and checked the manual for some details about interrupt triggers and such things. I accidentally stumbled across the motion control section and got hooked by this picture:



A toolpath without approximation artefacts. How cool was that? I kept on reading. It turned out that there was a completely independent motion planning algorithm in the controller. It is made to perform SPLINE movements in contrast to the LIN movement we used at that time. So what are the differences here? A polyline with enough waypoints looks like a spline, doesn't it? Here are the main differences:

- a SPLINE path will pass every defining point – no approximation deviations involved
- a SPLINE path will not vary at different execution speeds
- a SPLINE path is one single motion command, just like one straight line
- a SPLINE path can be exceguted a constant velocity alon the path – within limits

All good news. The first test prints using SPLINE motion command where amazingly consistant. The constant nozzle velocity translated into a slowly changing extrusion rate, without slight changes at each waypoint. That in itself turned out to be key for good printing results.

> A SPLINE path may contain up to roughly 5000 waypoints and will be precalculated by the controller. That calculation may take several seconds before the robot actually executes the SPLINE motion. The robot will not move during that period. Too many waypoints will result in some strange warnings by the controller, but below 8000 waypoints the robot will eventually finish the precalculation and proceed with the SPLINE motion. Beyond 8000 waypoints you might force the controller into strange out-of-memory states. You might have to restart your controller.

That waypoint limitation was in issue for 3D printing larger structures with complex cross sections. The helical tube on page 4 has a circular cross sections with only 20 waypoints per slice. The complete print with more that 200 layers required only 4000 waypoints and could be fitted into one single SPLINE command. One command! That command takes a few hours to execute and you end up with a finished 3D printed object. One command!

For more complex parts the capacity of one single SPLINE command will not be enough to describe the geometry. In that case I usually used one SPLINE command per layer, like for the tree structure on page 4.

I can only encourage you to check the manufacturer manual. You'll certainly find usefull information in there.

Robot milling

Once the BASE and TOOL definitions are perfomed to the wanted level of accuracy, you can start to load and execute toolpath - start making chips. Based on the general differences between CNC machines and robot arms, there are a few things to consider in the CAM process.

Reduce toolpath dynamics

As robots are more prone to vibrations, you should try to avoid them as best a you can. They can be triggered by instant changes in the cutting direction or the spindle orientation. Vibrations can even be triggered along smooth toolpath by changing the feedrate along the path.

That's more important for finishing operations than for roughing. Here is what to consider in Fusion 360 when using a robot to execute the toolpath.

Constant feedrates

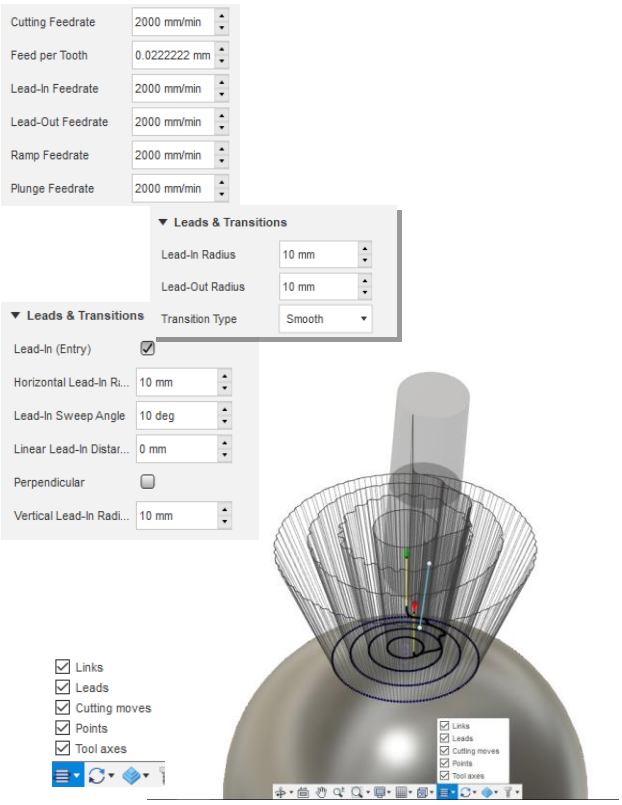
Try to keep the feedrates constant. Different lead-in and cutting feedrates may induce vibrations at the transition point. That's where you start cutting your part. That applies to all feedrate transitions. So why not using only one feedrate throughout?

Smooth leads

The lead-in motions should be adopted as well. Increase the radius to get a smooth geometric transition between the lead-in and the cutting part of the toolpath. For some toolpath you can reduce the sweep angle at the same time to avoid cutting big semicircles in air.

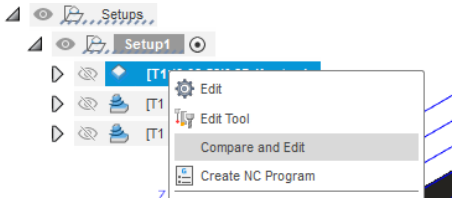
Check tool orientation

There is an easy way to check the tool orientation for 5 axis toolpath in Fusion 360 in the display settings. Use that to make sure no sudden jumps or oscillating orientations will be generated.



The picture shows four concentric circles on a sphere. All tool axes for one circle should reassemble as a perfect cone. They don't - not good. The robot will move along the cirlcle and constantly oszillate the orientations along the way. Vibrations are granted.

You can increase the internal CAM triangulation precision to generate more accurate tool axes: select the toolpath, rightclick, in the drop-down menu *Compare and Edit → Surface Triangulation* and adjust the value to smaller values.

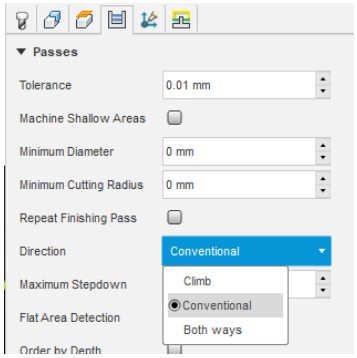


Consider machine rigidity

CNC machines are superior over robot arms with regards to rigidity. You should consider that in the toolpath generation and rethink using conventional milling vs. climb milling.

During conventional milling the tool tends to deflect away from the part. A Robot arm will allow slight deflection and a spring pass might be required to get the target part dimensions.

Climb milling with non rigid machines has a bad failure mode: the tool deflects into the part, increasing chipload, increasing deflection and finally ending in tears.



KRL post processor

This post processor and the resulting *.krl files should be treated as experimental only. Don't run the code unattended without prior testing and be careful during testing.

Fusion 360 offers a KRL post for Kuka robot arms on their HSM post page. Other manufacturers are supported as well.
I had a closer look at the post, generated some KRL code and ran that on the robot. It worked pretty good.
Based on the robot workflow I developed during 3D printing, I wanted to modify the post to

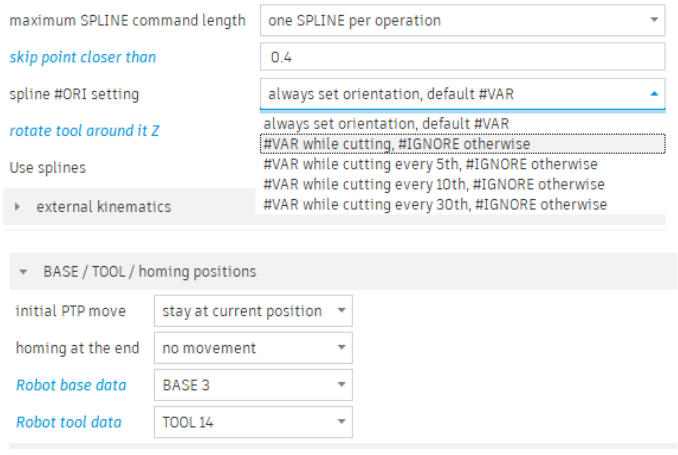
- better suit my personal workflow in terms of HOME, BASE and TOOL selection
- support SPLINE motion to benefit from its expected advantages in milling applications
- enable basic turntable support as exteral axis E1
- human readable feedrates – the default unit is m/s and I'm not used to that at all. Adding a comment with mm/min feedrates at the end of a line helped me a lot
- generate only one file to simplify file handling on the robot controller

It took me a while to get into post processor programming, but using the documentation provided by Autodesk helps a lot.
The five modifications are done and working as is. It needs more testing and certainly modifications in the future, but I ran test milling programs using SPLINE toolpath on the turntable. Sample videos are linked in my corresponding Autodesk forum post.

It's used just like any other post processor. Copy it into your local drive or to your cloud assets, select it as a post and adjust the settings to your needs.

Post Specific Options

The most significant options are related to the spline toolpath generation. Initially there was one checkbox only to enable the use of splines. Based on different test toolpath more and more options were added. They result from the size limit for the number of waypoints along a spline motion. Long finishing toolpath with several thousands of toolpath points need to be split automatically.



Base and Tool selections are not based on the CAM setup WCS or the tool number in the tool library. Both are set explicitly in the post configuration. The initial move to define the robot pose can be selected explicitly as well. That position must be predefined in the robot controller, unless you choose the actual position to be the initial position for the code execution. Same applies for the end position. I can only recommend to use the same predefined position for both options. In that case you always end up with the same end position and there will be no unexpected motions based on strange initial positions.

Possible future additions

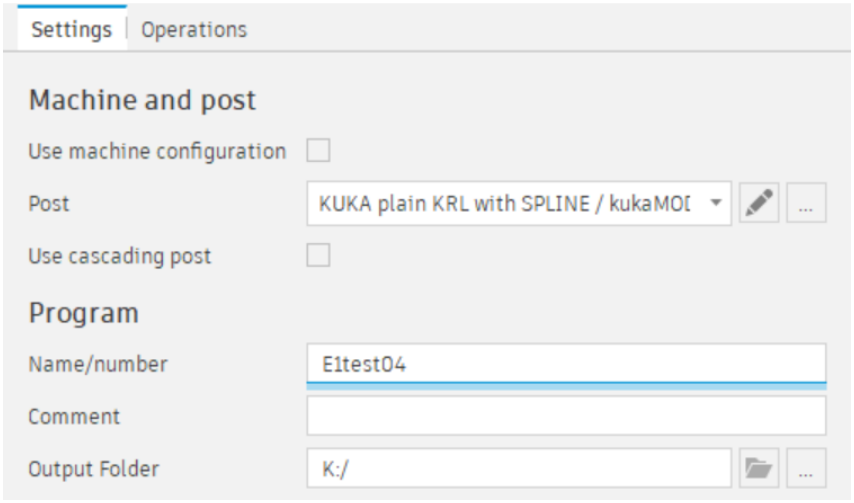
There are a few things that I'd like the post handle in the future. Here is an incomplete list of possible future additions:

- probing support with a 3D touchprobe for setting up a local BASE (define a new WCS)
- tool probing for length compensation after automatic tool change (ATC)
- tool probing for wear compensation for high wear tools (like an abrasive cutting disks)
- support for automatic endeffektor change from within Fuison 360. That allows a 'tool change' beyond regular ATC. Applications are various like milling with one spindle and deburring or engraving with a second air powered high speed spindle.

File handling CAM to robot

The easiest way to get your KRL file into the robot control is via a USB drive. Once the drive is plugged in, you can copy the file into the according directory and excecute it. For frequent iterations especially during development and testing the plugging and unplugging is tedious.

To get rid of that, I mapped a network drive (K://) to the robot control and used that as a target directory for the post processor. Setting up the robot controller for that is nicely explained by Karl Singline in his video tutorial. This reduces file handling to a single copy&paste operation on the robot control.



Sample KRL code

Here is how KRL code might look like.

```
; Load Tool    tool override from 133 to 5 during posting
;TOOL_CHANGE(5)

; Set Tool
BAS(#TOOL,5)
$TOOL = TOOL_DATA[5]

; Spindle ON/speed
;SPINDLE_ON(35000)

;   ####################################
;   ####################################
;   #  new operation  sFlow3
;   ####################################
;FOLD SPLINE - start spline
SPLINE
SPL {E6POS:X 23.2769,Y 16.7132,Z 182.2085,A 48.105,B 35.489,C -156.450} WITH $VEL.CP=0.083333, $ORI_TYPE=#IGNORE; 2500mm/min
; Lead In Move Starts
SPL {E6POS:X 23.2469,Y 16.6898,Z 182.0622,A 48.105,B 35.489,C -156.450} WITH $VEL.CP=0.05, $ORI_TYPE=#IGNORE; 1500mm/min
SPL {E6POS:X 23.1598,Y 16.6216,Z 181.9327,A 48.105,B 35.489,C -156.450} WITH $VEL.CP=0.05, $ORI_TYPE=#IGNORE; 1500mm/min
SPL {E6POS:X 23.1128,Y 16.4783,Z 181.8229,A 48.105,B 35.489,C -156.450} WITH $VEL.CP=0.05, $ORI_TYPE=#IGNORE; 1500mm/min
SPL {E6POS:X 23.1524,Y 16.3005,Z 181.7823,A 48.105,B 35.489,C -156.450} WITH $VEL.CP=0.05, $ORI_TYPE=#IGNORE; 1500mm/min
; Cutting Move Starts
; Finish Cutting Move Starts
SPL {E6POS:X 23.6847,Y 15.1644,Z 181.7953,A 48.300,B 34.797,C -155.685} WITH $VEL.CP=0.083333; 2500mm/min
SPL {E6POS:X 24.1822,Y 14.0296,Z 181.8072,A 48.397,B 33.714,C -154.687} WITH $VEL.CP=0.083333; 2500mm/min
SPL {E6POS:X 24.6414,Y 12.8947,Z 181.8176,A 48.456,B 32.548,C -153.618} WITH $VEL.CP=0.083333; 2500mm/min
SPL {E6POS:X 25.0620,Y 11.7596,Z 181.8270,A 48.486,B 31.446,C -152.592} WITH $VEL.CP=0.083333; 2500mm/min
SPL {E6POS:X 25.4445,Y 10.6243,Z 181.8377,A 48.576,B 30.233,C -151.338} WITH $VEL.CP=0.083333; 2500mm/min
SPL {E6POS:X 25.7877,Y 9.4884,Z 181.8447,A 48.580,B 29.175,C -150.287} WITH $VEL.CP=0.083333; 2500mm/min
SPL {E6POS:X 26.0927,Y 8.3524,Z 181.8520,A 48.572,B 28.055,C -149.137} WITH $VEL.CP=0.083333; 2500mm/min
```

Additional Resources

There is a wide selection of additional resources to deepen your knowledge

- manufacturer manuals
- specific tranings by the manufacturer of third party
- forum discussions (autodesk forum, roboter-forum and more)

Manufacturer specific pictures

Machine pictures and a few manual illustration from Kuka, Fanuc and Haas have been used in the handout and presentation.