Graphical simulation of a NC program on a PC

Citation for published version (APA):

Document status and date:
Published: 01/01/1987

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 09. Dec. 2019
Graphical Simulation of a NC Program on a PC

Author: Fan Muchang

WPA-Report No. 0419

Advisor: Ir. J. A. W. Hijink
GRAPHICAL SIMULATION OF A NC PROGRAM ON A PC

SUMMARY
MH432.PAS is a computer graphical simulation program on PC.XT or PC.AT for the MAH0432 controller, which uses the 3D graphic technique to simulate the cutting process of the CNC machine. The simulating cutting process is made by moving a tool matrix over the workpiece blank matrix and comparing the value of the corresponding elements which represent the depth of the workpiece. The 'manufactured' workpiece can be displayed either in a 3-plane projection (as with a working drawing) or in 3D-projection (similar to a graph).

This report presents the main principle which has been used in the program and some restrictions that the user must pay attention to.

1 INTRODUCTION
The MAH0432 is a 4-axes controller for a CNC milling machine. With the help of a computer aided part programming system such as DLOG (a system which caters a two dimensional profile in a very simple way) we can make the part program quite easy. But there are still some problems left. For example, you can't be sure if there are some mistakes in your part program, sometimes it is very dangerous such as a collision. The aim of this program is to check your part program line by line and print out the errors you have made. The workpiece can be displayed either in 3D-projection (Fig.1) or in 3-plane projection (Fig.2). The machine operator can therefore assure himself that the right program has been made before actual machining.

Fig.1 3D projection view
2 BASICS IDEAS OF THE PROGRAM

In this program both the workpiece and the tool are considered as a two dimensional matrix, and we let the column to represent Z coordinate, the row for the X coordinate and the value of the elements of the matrix to represent the Y coordinate. For example we can use a 140*80 matrix to depict a workpiece blank of 140*30*80 of the workpiece. If we let the name of the workpiece blank matrix be WP then the element WP[30,40] will represent a point which its x coordinate equals to 30, Z coordinate equals to 40. If the value of WP[30,40] is -23.5, that means the height of this point is -23.5.

Using the same method we can define a tool matrix. For example we can use a 11*11 matrix (for easy to calculation we let the lower bound of the matrix be equal to -5 and the upper bound +5) to represent a cylindrical shaft end mill with a radius of 5 mm. We choose the element T[0,0] as the centre of the tool. If the distance between any element T[I,J] and T[0,0] is less or equal to the tool radius we let the value of this element equal to 0 otherwise let the value of this element equal to -1. Here -1 only means this point is out of tool radius. If we have a finger head or some other shape mill the only thing we have to do is to change the value 0 to the real Y coordinate of the tool.

With these two matrix we can simulate a cutting process.
as follows:

Procedure DrawTool(X,Z,Y);
Begin
    For I:=-5 To 5 Do
        Begin
            For J:=-5 To 5 Do
                Begin
                    If T[I,J]<>-1 Then
                        Begin
                            If WP[X+I,Z+J]:=Y+T[I,J] Then
                                WP[X+I,Z+J]:=Y+T[I,J]
                        End
                End
        End
End;

Where X,Z are the coordinates of the tool centre, Y is the depth which we want to cut. If the tool depth is deeper than the material of the workpiece we change the height of workpiece to the tool depth. The cutting process can be simulated by changing the X,Z value along a line or a circle. If a 'cutting condition' happens in a rapid traverse then a collision is reported.

3. several important points
a)3D projection graph
During the simulation of cutting the workpiece matrix will be modified according to the cutting process. The value of each element represents the Y coordinate of each point. To draw the top view of the workpiece we compare each element of the workpiece matrix with the adjacent elements. If the value of this element is different from other elements then we draw this point into the screen. Using this method the intersection line of two planes can be drawn on the screen. The only restriction for this method is, for a declining plane, all the points of the surface then will be drawn on the screen. By good fortune most surface of the workpiece are not like this. To draw the two section view of the workpiece is in fact to display the value of one column or row of the workpiece matrix in the screen with a proper scale factor. The sectional planes can be shifted by choosing a different column or row of the workpiece matrix.

b)3D-projection graph
For the 3D projection graph we use a special method, first, cutting the workpiece slice by slice. The shape of a slice is the same as the section view of the workpiece. Then use the three dimensional graphical technique [1] to put these slices into the screen one by one. The cross sections closest to the viewer are generated first. As each section is generated, two arrays are constantly updated. In this program the array YMIN[K] maintains a record of the top most Y coordinate plotted. The second array YMAX[K] keeps a record of
the bottom most Y coordinate plotted in column K. At a specific stage of the surface generation, these arrays represent the upper and lower bounds of the vertical screen coordinates of visible points. If a point lies between YMIN[K] and YMAX[K], that point is not visible; it represents a hidden point of the workpiece. In order to plot a vertical line on the screen, the line must be plotted from the bottom to top.

c) Tool diameter compensation

The tool radius compensation is done by finding the intersection point between the shifted line. The shifted line is determined by the tool radius and the G code (G41, or G42). In this program, the procedure MakeFunction is used for this purpose. The function used for line is \( AX + BY = C \) for circle is

\[
(X-XC)^2 + (Y-YC)^2 = RC^2
\]

Three other small procedures 'LineToLine', 'LineToCircle', 'CircleToCircle' are used to calculate the intersection points between these lines. If there is no intersection between lines, the program will report a compensation calculation error. Because of the differences in calculation between this program and the controller machine, the user must pay attention to the next two conditions:

In Fig. 3(a) the two shifted circle have no intersection so that a compensation calculation error is reported but in MAHO machine this is not a problem. The real tool path is shown in Fig. 3(b) (See [2]). The difference between Fig. 4(a) and (b) is due to the same reason.

For the tool radius compensation G43 and G44, the program uses an imaginal circle. The middle point of the circle is the nominal terminal point, the radius of the circle is equal to the tool radius. Then we find the intersection points between the tool path and the imaginal circle, choose one of the point according to the G code G43 or G44.

d) Canned circle

The MAHO machine has a lot of canned circle. Such as G81, G83 (pecking), G87 (rectangular pocket), G89 (circular pocket), G88 (slot milling) etc. When the program reads such functions, it sets a flag with the name of the G code
as prefix. For example if G87 code is encountered we can set G87Flg to 1 and put all the necessarily parameters into the correspond variables. This function will be executed when a G77 or G79 function is encountered.
e) Repeat circle
The repeat circle is a quite difficult part of the program. Due to the restriction of the memory size that can be used is not big enough, the simulate program read the part program line by line. Every time a G14 function is encountered the repeat circle will be completed as follows: (1) put the beginning line number N1 the last line number N2, the repeat times J1 into corresponding variables (2) put the return address (part program line number) into a buffer (3) close the part program file (4) open the part program again (5) read part program file line by line until the line number N1 (6) begin the simulate procedure until line N2 (7) the repeat times subtracts 1, if the repeat times not equals to 0 then repeat the from step 3 (9) close the part program file (10) open the part program and return to the simulate program line which we put into the buffer before.
If there is another repeat circle nested in the circle, we must first run the nested circle, this makes the problem even more complicated so that in this program only one circle can be nested in other circle.

4 CONCLUSION
Simulating a cutting process in a microcomputer is an easy and useful method to check the part program, so that it can be part of a computer aided part programming system.
This program can be used to check the workpiece shape, report collision, check the speed of spindle, the feed, and some syntax error in the part program and these abilities can be expanded later.
Since a matrix is used to depict the workpiece, the bigger the matrix the better resolution we can obtain time. This program is write in Turbo Pascal. Due to the restriction of this language the workpiece matrix is 150*150. If the size of workpiece is bigger than 150 then the resolution will less then 1 mm.

REFERENCE
[1] Apple graphics
Harold J. Bailey J. Edward Kerlin
[2] Inleidend cursus Numerieke Besturing voor technici
G. J. G. Van de Molengraft Nov. 1985
## APPENDIX 1 SOME IMPORTANT GLOBE VARIABLE

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIR</td>
<td>circle radius for circular pockets</td>
</tr>
<tr>
<td>Line</td>
<td>the string of the part program line which is simulated at moment</td>
</tr>
<tr>
<td>Lineg1</td>
<td>the next line after the line</td>
</tr>
<tr>
<td>Lineg2</td>
<td>the next line after the linel</td>
</tr>
<tr>
<td>ProgramLineNumber</td>
<td>the number of the program that is simulated at moment</td>
</tr>
<tr>
<td>FSL</td>
<td>first side length for canned circles</td>
</tr>
<tr>
<td>J141</td>
<td>the repeat times of repeat circle</td>
</tr>
<tr>
<td>J142</td>
<td>the repeat times of the nested repeat circle</td>
</tr>
<tr>
<td>LA1, LB1, LC1</td>
<td>parameters of shifted line for tool radius compensation</td>
</tr>
<tr>
<td>LA2, LB2, LC2</td>
<td>parameters of shifted next line for tool radius compensation</td>
</tr>
<tr>
<td>CR1, CI1, CK1</td>
<td>parameters of shifted circle for tool radius compensation</td>
</tr>
<tr>
<td>CR2, CI2, CK2</td>
<td>parameters of shifted next circle for tool radius compensation</td>
</tr>
<tr>
<td>N11, N12</td>
<td>the beginning and end line number of repeat circle</td>
</tr>
<tr>
<td>N21, N22</td>
<td>the beginning and end line number of the nested repeat circle</td>
</tr>
<tr>
<td>SSL</td>
<td>second side length for canned circle</td>
</tr>
<tr>
<td>SUC</td>
<td>set-up clearance for canned circle</td>
</tr>
<tr>
<td>TD</td>
<td>the radius of tool</td>
</tr>
<tr>
<td>TN</td>
<td>the number of tool</td>
</tr>
<tr>
<td>XBLOCK, YBLOCK, ZBLOCK</td>
<td>the size of workpiece blank</td>
</tr>
<tr>
<td>XC, YC, ZC</td>
<td>the X, Y, Z coordinate of the terminal point of last program line.</td>
</tr>
<tr>
<td>XCC, ZCC</td>
<td>buffer of XC, YC, ZC</td>
</tr>
<tr>
<td>YNIN, YMAX</td>
<td>upper and lower bounds of the vertical screen coordinates</td>
</tr>
</tbody>
</table>