Design of the Computer Numerical Control program using AutoLISP and AutoCAD2000's two-dimensional polyline vertexes

Cerc. pr. I ing. Virgiliu Streian
Universitatea "Tibiscus" din Timişoara

REZUMAT. Lucrarea prezintă o metodă de proiectare a programelor pentru maşini cu comandă numerică utilizând mediul de proiectare asistată de calculator AutoCAD 2000. Sunt descrise principiile de proiectare a unui postprocesor, instrucțiunile G-Code utilizate de o maşină cu comandă numerică, precum şi un program AutoLISP de conversie a unui desen AutoCAD 2000 în instrucțiuni G-Code.

1 Motivation

Faced with competitive pressures to shorten time-to-market, reduce costs, increase productivity, and improve quality, industrial manufacturers are adopting the advent of computer technologies to all production steps.

2 Purpose

The purpose of this article is to become familiar with design of numerical control (NC) part programming of the computer aided manufacturing (CAM) process. This part consists in design of the CAM programs (postprocessors). The postprocessors are programs that convert a drawing, created by AutoCAD2000 by example, into G-code instructions which are used by G-code interpreter programs that in turn run the machine's controller. Conversion of the drawing to a g-code file is an important step. The conversion must be accurate (without errors and corresponding to the specific computer numerical control (CNC) machine) in order to enable the

CNC machine part production with minimum human intervention. There are different types of g-code formats depending on the type of CNC machine:

- 2-Axis Mill/Drill and/or Machining Center
- 3-Axis Mill and/or Machining Center
- 4-Axis Machining Center with indexing (positioning only) rotary table
- 4-Axis Tilting Wire EDM machine (merged 2D profiles method only)
- Flat Pattern machine (Punch Press, Flame Cutter, etc.)
- 5-Axis Machining Center with contouring gantry-type pivoting rotary head
- 4-Axis Machining Center with contouring rotary table
- 5-Axis Machining Center with rotary and tilting table
- 5-Axis Machining Center with rotary table and tilting head
- 5-Axis Machining Center with two rotary tables, one mounted on the other
- 5-Axis Machining Center with one non-orthogonal rotary axis
- 4-Axis Tilting Wire EDM machine (merged 2D profiles and/or Multax methods)
- 4-Axis control-merged or tape-merged lathes having a vertical or horizontal spindle
- Mill/Turn centers, i.e. "C-Axis" lathes with a rotation all y addressable spindle and milling attachments.
- … Etc. …

3 CAD/CAM systems description

Computer aided design (CAD) and computer aided manufacturing (CAM) are steps dealing with automation of the engineering functions in design and manufacturing. CAD includes design conceptualization, product design, modeling, simulation, analysis, evaluation and documentation of the final outcome. The geometric model formed during the CAD process is the basis for the CAM process. AutoCAD, produced by AutoDesk Inc. from California USA, is one of the numerous CAD programs for creating the geometric model. CAM includes process planning and numerical control (NC) part programming.

In CAD/CAM systems the design and manufacturing steps are integrated by computer systems. The output of the CAD system, the design specification of the product, is automatically converted to a process plan for making the product, the g-code program. This program is then downloaded to the machine tool through a communication interfaces and then physical production is implemented.

4 Basic overview and explanation of CNC

Computer control of machine tools (numerical control - NC, computer numerical control - CNC, direct numerical control - DNC) has made it possible to produce large varieties of very complex parts on individual machines. The machine axes are driven by computer controlled drives and so complex, coordinated motions that could not be produced by mechanical devices are possible and programming easily reconfigures machine motions.

The parts that make up a CNC system are:

- The personal Computer (PC).
- Programs that allow the users to design their products (ex. AutoCAD 2000).
- Programs that convert the designs into G code instructions which are used by G code interpreter programs that in turn run the machine's controller (purpose of this article).
- Programs that interpret the instructions and operate the machine.
- The PC parallel port that sends the signals to the machine's controller.
- The controller that reads the signals generated by the PC via the program and the parallel port, which operates stepper motors in a controlled manner.
- The stepper motor that moves the axes of the machine to the requirement of the instruction sent to the controller.
- The machine that moves each axis individually or simultaneously. Usually these are three-axis machines.

The PC is the brains that runs the various software programs that then send signals to the parallel port instructing the step motor controller when and much to move the stepper motor attached to each axis. There are several programs that are used on a PC.

- The first is a design program generally called Computer Aided Design (CAD). This is the program where you design your part. Here we are referring to AutoCAD 2000.
- The next piece of software, which may or may not be part of the CAD program, is the Computer Aided Manufacturing (CAM) program. A

CAM program alias postprocessor, works with the user to generate the instructions for the machine. The program then creates a tool path file in G-Code - language to control motion and I/O tool operations alias RS-274D standard or a variant of this standard - and it allows the user to set the various speeds, feed rates and depth of cuts. Tool Path is a series of vector coordinate positions that define a cutting path. This cutting path can be a simple 2D or sophisticated 3D (even 4D or more) path used to machine out the shape of a desired part. There are different types of g-code formats depending on the type of CNC machine. In this article our purpose is designing such programs.

RS-274D, often called G-Code, is the recommended standard for numerically controlled machines. These standards provide a basis for the writing of numeric control programs. The full NIST Enhanced Machine Controller (EMC) is NC programmed using a variant of the RS274D language to control motion and I/O tool operations. This variant is called RS276NGC because it was developed for the Next Generation Controller, a project of the National Center for Manufacturing Science.

CNC programming is concerned with the planning and documentation of the sequence of processing steps to be performed by a NC machine. The planning portion of the part programming requires knowledge of machining as well as geometry and trigonometry. The sequence of processing steps in NC machines involves a series of movements of the processing head with respect to the machine table and work part. The program consists of a list of tasks, such as controlling the motion of the cutting tool, moving the part being formed against a rotating tool, or changing the cutting tools.

The standard language of CNC (RS274D alias G-code) have instruction that specifies a series of machine moves (ex. rapid travel at maximum feed rate, linear interpolation at current feed rate, clockwise circular interpolation at current feed rate - X-Y plane only, counter clockwise circular interpolation at current feed rate - X-Y plane only) and I/O tool operations (e.g., feed rate, cutting speed, tool selection, and miscellaneous functions to stop or initiate machine, tools, and accessories). The moves can involve simultaneous motion of one or more axes (drives). Machine tool controllers and drives vary in sophistication from being able to execute single, linear moves to being able to drive many axes simultaneously. There are different types of G-code formats depending on the type of CNC machine.

The CNC programs is executed a line at a time and all code in one line is executed at the same time. Usually an CNC program have an

- Initial section in which the machine is set up;
- A section in which the part is machined;
- A section in which the machine is shut down.

The below listed file is an example of G-code program - Nxxxx is number of line, Gn is un G-code, Mn is a M code, Q is feed increment and Fn is a desired federate:

```
% header section
N0000 G91 X0 Y0 F200.000
N0005 G65 H01 P32 Q000.200
% pen down
N0010 M12
N0015 G01 X000.000 Y-85.024
N0020 G01 X118.340 Y000.000
N0025 G01 X000.000 Y085.024
N0026 G01 X-118.340 Y000.000
% pen up
N0030 M13
% moving to next entity
N0035 G00 X-97.457 Y-78.068
% pen down
N0040 M12
N0045 G01 X059.557 Y017.005
N0050 G03 X-07.734 Y035.556 I-05.006 J017.530
N0055 G02 X-30.939 Y017.777 I-04.286 J028.352
N0060 G01 X-20.884 Y-70.338
% Pen up at end of program
N0065 M13
N0070 M02
```

Most of the G-codes tend to be related to motion or sets of motions. M codes are used to control many of the I/O functions of a machine.

The partial list of G-codes available to G-code programmer is included in table G-Codes List shown below.

G-Codes List

G code	Function and Description	Example
G00	Rapid Positioning. Using a G00 in your code is equivalent to saying "go rapidly to point xxxx yyyy". This code causes motion to occur at the maximum traverse rate	**N100 G00 X10.00 Y5.00** - go rapidly to point X10.00 Y5.00 **N100 G00 Z1.5** - move spindle above obstacles
G01	Linear Interpolation. G01 causes the machine to travel in a straight line with the benefit of a programmed feed rate (using "F" and the desired feedrate). This is used for actual machining and contouring.	**N120 G01 Z0.1 F6.0** - move the tool down to Z=0.1 at a rate of 6 inches/minute **N130 G01 Z-.125 F3.0** - move tool into the work piece at 3 inches/minute **N140 G01 X2.5 F8.0** - move the table, so that the spindle travels to X=2.5 at a rate of 8 inches/minute
G02 **G03**	Circular/Helical Interpolation (CW/CCW). G02/G03 causes clockwise / counterclockwise circular motion to be generated at a specified feed rate (F). When coding circular moves, you must specify where the machine must go and where the center of the arc is in either of two ways: By specifying the center of the arc with J words, and I or giving the radius as an R word. I is the incremental distance from the X starting point to the X coordinate of the center of the arc. J is the incremental distance from the Y starting point to the Y coordinate of the center of the arc.	**N100 G01 X0.0 Y1.0 F20.0** - go to X0.0, Y1.0 at a feed rate of 20 inches/minute **N200 G02 X1.0 Y0.0 I0.0 J-1.0** - go in an CW arc from X0.0, Y1.0 to X1.0 Y0.0, with the center of the arc at X0.0, Y0.0 **N300 G01 X0.0 Y1.0 F20.0** - go to X1.0, Y0.0 at a feed rate of 20 inches/minute **N400 G02 X1.0 Y0.0 R1.0** - go in an CW arc from X0.0, Y1.0 to X1.0 Y0.0, with a radius of R=1.0
G20	Inch system selection.	
G21	Millimeter system selection.	
G90	Absolute Command. Use the absolutes coordinates in G00, G01, etc. g codes	
G91	Increment Command. Use the incremental coordinates in G00, G01, and etc. g codes.	

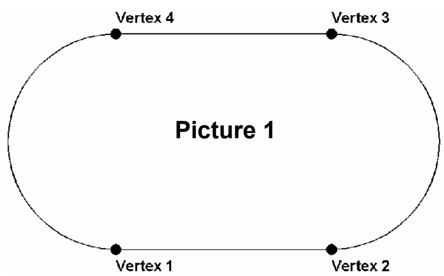
M codes are used to control many of the I/O functions of a machine. M codes can start the spindle and turn on mist or flood coolant. M codes also signal the end of a program or a stop within a program.
The partial list of M codes available to G-code programmer is included in table M Codes List shown below.

M codes List

M code	Function
M00	Stop and wait for start button to restart
M02	End of Program
M03	Stop and Go to Console Mode
M11	Spindle on clockwise
M12	Spindle on counterclockwise
M13	Spindle off
M21	Coolant on
M22	Coolant off
M30	Output on
M31	Output off
M98	Call subprogram
M99	End of subprogram, return

5 Use AutoCAD2000's two-dimensional polyline vertexes for CNC program

AutoCAD's *PLINE* command creates a two-dimensional polyline. A polyline is an AutoCAD object composed of one or more connected line segments or circular arcs treated as a single object (also called pline). A polyline vertex is a location where polyline segments meet. The two-dimensional polyline vertex can start a line segment (see **Picture 1** vertex 1) or a circular arc (see **Picture 1** vertex 2). A CNC program for a NC cutter machine like a Flame Cutter can process this vertexes coordinates.

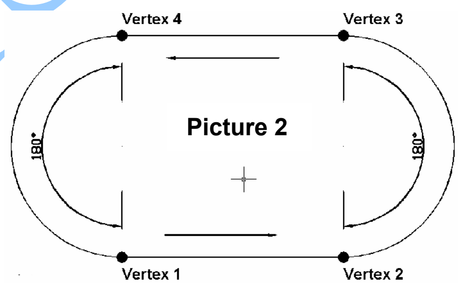


AutoCAD drawings are largely consisted of structured containers for graphic entities. Each graphic entity is memorized inside the drawing database as a list-type structure called association-list of entity. Association-list for the **Picture 1** polyline entity looks like this:

```
(
(-1 . <Entity name: 198c980>) (0 . "LWPOLYLINE")
(330 . <Entity name: 198c8f8>) (5 . "30") (100 . "AcDbEntity")
(67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbPolyline") (90 . 4) (70 . 1)
(43 . 0.0) (38 . 0.0) (39 . 0.0)
; Vertex 1 - start a line segment
(10 100.0 100.0) (40 . 0.0) (41 . 0.0) (42 . 0.0)
; Vertex 2 – start a circular arc
(10 200.0 100.0) (40 . 0.0) (41 . 0.0) (42 . 1.0)
; Vertex 3 - start a line segment
(10 200.0 200.0) (40 . 0.0) (41 . 0.0) (42 . 0.0)
; Vertex 4 - start a circular arc
(10 100.0 200.0) (40 . 0.0) (41 . 0.0) (42 . 1.0)
;
(210 0.0 0.0 1.0)
)
```

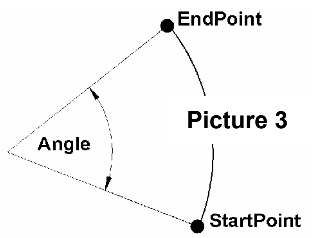
Each item (sub list) in the list is specified by a DXF group code (Ex. –1, 0, 2, 100, **70**, **10**, 40, 41, **42** and 210). The DXF codes are a standard for representing graphic information.

The first item in the list contains the entity's current name: (-1 . <Entity name: 198c980>). The group **(10 100.0 100.0**) (40 . 0.0) (41 . 0.0) (**42 . 0.0**) identify a vertex.



In the dotted pair **(70 . n)**, **n** represents the polyline flag. If **n** = 0 (default value) the polyline is open, like in the **Picture 4**, else if **n** = 1 the polyline is closed like in the **Picture 2** (another value of **n** is 128 for Plinegen).
List like identify the coordinates of the vertexes: **(10 x y).** The dotted pair: **(42 . x)** show that the vertex start a line segment if **x** = 0.0 or a circular arc if **x** <> 0.0 (x<0 clockwise arc).

The context of the vertex, which starting a circular arc is not clearly explained in AutoCAD/AutoLISP. After a series of individual tests we have reached some conclusions.



In the dotted pair: **(42 . x)**, **x** alias **bulge** is a real number, and 4xArctan(**x**) represent **Angle** of the Arc (see **Picture 3**).

Angle= 4 x Arctan (x)

The fallowing AutoLISP sequences compute this **Angle.**

```
(setq P_al            ; retrieves the two-dimensional polyline association-list
  (entget
    (nth 0 (entsel "Please choose a 2D polyline object:"))
  )
)
(setq B_ulge (cdr (assoc 42 P_al))) ; give the bulge value from dotted pair
(42. x)
(setq A_ngle (* 4 (cvunit (atan BULGE) "radian" "degree"))) ; compute
degree Angle
```

Like **x**, if **Angle** < 0.0 then CW arc, else if **Angle** > 0.0 then CCW arc.

For example the user has drawn a two-dimensional polyline (**Picture 2**) with the following AutoCAD2000 sequence of commands:

Command: _pline↵
Specify start point: ***50,50*** ↵
Current line-width is 0.0000
Specify next point or [Arc/Close/Halfwidth/Length/Undo/Width]: ***150,150*** ↵
Specify next point or [Arc/Close/Halfwidth/Length/Undo/Width]: ***Arc*** ↵
Specify endpoint of arc or [Angle/CEnter/CLose/Direction/Halfwidth/Line/Radius/Second pt/Undo/Width]: ***220,90*** ↵
Specify endpoint of arc or [Angle/CEnter/CLose/Direction/Halfwidth/Line/Radius/Second pt/Undo/Width]: ↵

Association-list of this polyline entity obtained with AutoLISP sequence *(entget (nth 0 (entsel "Please choose a 2D polyline object:"))* is shown below:

(
(-1 . <Entity name: 198c980>) (0 . "LWPOLYLINE")
(330 . <Entity name: 198c8f8>) (5 . "30") (100 . "AcDbEntity")
(67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbPolyline") (90 . 4) (**70 . 1**)
(43 . 0.0) (38 . 0.0) (39 . 0.0)
; Vertex 1 -- start a line segment
(**10 100.0 100.0**) (40 . 0.0) (41 . 0.0) (**42 . 0.0**)
; Vertex 2 -- start a circular CCW arc
(**10 200.0 100.0**) (40 . 0.0) (41 . 0.0) (**42 . 1.0**)

; Vertex 3 -- start a line segment
(**10 200.0 200.0**) (40 . 0.0) (41 . 0.0) (**42 . 0.0**)
; Vertex 4 -- start a circular CCW arc
(**10 100.0 200.0**) (40 . 0.0) (41 . 0.0) (**42 . 1.0**)
;
(210 0.0 0.0 1.0))

Both the Vertex 2 and Vertex 4 *start a circular arc*:
(**10 200.0 100.0**) (40 . 0.0) (41 . 0.0) (**42 . 1.0**) --- Vertex 2
(**10 100.0 200.0**) (40 . 0.0) (41 . 0.0) (**42 . 1.0**) --- Vertex 4

Where
(**10 200.0 100.0**)/(**10 100.0 200.0**) ; *Vertex 2/4 coordinates*
(**42 . 1.0**)/(**42 . 1.0**) ; *start a circular CCW arc 2/4*

The list **(10 200.0 100.0)** respective (**10 100.0 200.0)** identify the beginner points for the circular arcs at 200;100 respective 100;200, and dotted pair **(42 . 1)**, identify the **bulge** at value 1.0 and CCW arc.

Angle = 4xAtan(1)
Atan(1) = 45°
Angle= 4x45°

Angle = 180° (see **Picture 2** and **Picture 3**)

Start Point, *Angle* and *Endpoint* define both circular arcs of the two-dimensional polyline from **Picture 2** (see also **Picture 3**):

StartPoint: 200 , 100 (from Vertex 2 coordinate list: **(10 200.0 100.0)**)
Angle: 180° (bulge from dotted pair: **(42 . 1)**)
EndPoint: 200 , 200 (from next Vertex 3 coordinate list: **(10 200.0 200.0)**)

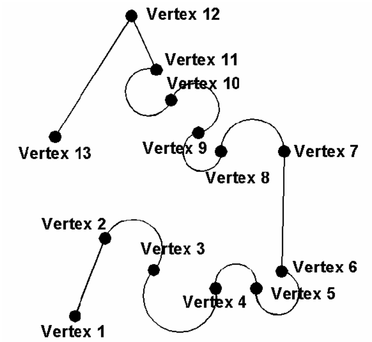
Start point: 100 , 200 (from Vertex 4 coordinate list: **(10 100.0 200.0)**)
Angle: 180° (bulge from dotted pair: **(42 . 1)**)
End point: 100 , 100 (from next Vertex 1 coordinate list: **(10 100.0 100.0)**)

6 Design CNC programs using AutoLISP

AutoLISP is a high level programming language for AutoCAD. AutoLISP, dialect for Common Lisp, is a standard feature of AutoCAD. It is an interpreter available only within the AutoCAD drawing editor. With AutoLISP we can access to geometrical database information (drawing database) of graphic entity within AutoCAD drawing editor. This feature we can use for design CNC programs.

For example we consider the two-dimensional polyline contour **Picture 4** created throw *PLINE* AutoCAD2000 command.

Picture 4



The vertexes 1, 6 and 12 start a line segment and the vertexes 2, 3, 4, 5, 7, 8, 9 and 10 start a circular arc. The below table show the characteristics of vertexes:

Vertex no.	**X**$_{start}$	**Y**$_{start}$	**Radius of Arc** (R<0 -- CW) (R>0 -- CCW)	**X**$_{center}$ **of Arc**	**Y**$_{center}$ **of Arc**
1	84.6485	68.051			
2	97.332	100.561	-12.4125	108.896	96.0498
3	117.736	87.3367	12.4125	108.896	96.0498
4	143.654,	79.6224	-8.72067	152.202	81.3514
5	160.749	79.6224	8.72067	152.202	81.3514
6	171.227	86.7857			
7	172.33	136.929	13.2381	159.095	137.22
8	145.86	136.929	-8.00818	137.854	136.752
9	136.485	144.643	8.00818	137.854	136.752
10	124.905	158.418	-9.88147	115.604	161.755
11	118.839	171.092			
12	108.361	193.684			
13	76.3767,	142.99			

Association-list of this polyline entity obtained with AutoLISP sequence *(entget (nth 0 (entsel "Please choose a 2D polyline object:"))* is shown below:

((-1 . <Entity name: 19e9d70>) (0 . "LWPOLYLINE") (330 . <Entity name: 19e9cf8>) (5 . "2E") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "cote") (100 . "AcDbPolyline") (90 . 13) (70 . 0) (43 . 0.3) (38 . 0.0) (39 . 0.0) (10 84.6485 68.051) (40 . 0.3) (41 . 0.3) (42 . 0.0) (10 97.332 100.561) (40 . 0.3) (41 . 0.3) (42 . -1.22691) (10 117.736 87.3367) (40 . 0.3) (41 . 0.3) (42 . 1.6646) (10 143.654 79.6224) (40 . 0.3) (41 . 0.3) (42 . -1.22252) (10 160.749 79.6224) (40 . 0.3) (41 . 0.3) (42 . 2.46234) (10 171.227 86.7857) (40 . 0.3) (41 . 0.3) (42 . 0.0) (10 172.33 136.929) (40 . 0.3) (41 . 0.3) (42 . 1.02224) (10 145.86 136.929) (40 . 0.3) (41 . 0.3) (42 . -2.17968) (10 136.485 144.643) (40 . 0.3) (41 . 0.3) (42 . 1.73951) (10 124.905 158.418) (40 . 0.3) (41 . 0.3) (42 . -2.39575) (10 118.839 171.092) (40 . 0.3) (41 . 0.3) (42 . 0.0) (10 108.361 193.684) (40 . 0.3) (41 . 0.3) (42 . 0.0) (10 76.3767 142.99) (40 . 0.3) (41 . 0.3) (42 . 0.0) (210 0.0 0.0 1.0))

The tool path file alias G code file creation trough a CNC AutoLISP program for a NC cutter machine like a Flame Cutter most can process following steps:

- Retrieves the two-dimensional polyline association-list.
- Extract X, Y coordinates for each vertex from association-list of the two-dimensional polyline contour (cod DXF 10).
- Testing if vertex start a segment line or start a circular arc (cod DXF 42).
- Generating for each vertex a G code instructions
- Saving for each vertex a line into the tool path file.

An example of an AutoLISP CNC program is called *G_CODE* (main function of the *CNC.LSP* file shown below). The application (two AutoLISP functions) reads the association-list of the two-dimensional polyline shown in **Picture 4** and generates G code instructions, which are saved in the tool path file *GCODE.CNC*.

```
      ;;
      ;;
      ;; File CNC.LSP
      ;;
      ;; AutoCAD2000 code
      ;; This code is for training purposes
      ;;
      ;;
(defun CNC (PAL I)
 (if (/= PAL (list)) ; PAL – two-dimensional polyline association-list not empty?
 ; Then
  (progn
```

```
  (if (= (nth 0 (setq V (nth 0 PAL))) 10)
  ; then - write records into gcode file
  ; Nxxxx Ggg Xxxxx Yyyyy Rrrrr
  ; Ex.
  ;   N1010 G90 -- absolute Command (absolute mode)
  ;   N1010 G21 -- millimeter system selection
  ;   N1010 G00 X100.00 Y100.00  --- go rapidly to point xxxx yyyy
  ;   N1010 G01 X100.00 Y100.00    --- moves to point xxxx yyyy
  ;   N1010 G03 X100.00 Y100.00 R50  --- go in an CCW arc from current point to
xxxx yyyy (incremental point) with a R50 radius
  ; Only g-code G90, G21, G00, G01 and G03 are generated
   (progn
    (setq XY (cdr V)) ; extract X,Y of vertex
    (setq X (nth 0 XY)) ; extract X of vertex
    (setq Y (nth 1 XY)) ; extract Y of vertex
    (if (= I 1000)
    ; then – only for first record
     (progn
      (princ
       (strcat
           "N"
           (itoa (setq I (+ I 10)))
         ) gcode)                              ; Nxxxx - line number
      (princ " " gcode)                        ; blank
      (princ "G21" gcode)                      ; G21 -- millimeter system selection
      (princ " " gcode)                        ; blank
      (princ "G90" gcode)                      ; G90 -- Absolute (coordinates)
Command
      (princ "\n" gcode)                       ; end record -- a newline
;
      (princ
       (strcat
           "N"
           (itoa (setq I (+ I 10))) ;  Increments the counter by ten
         ) gcode)                              ; Nxxxx - line number
      (princ " " gcode)                        ; blank
      (princ "G00" gcode)                      ; G00 -- go rapidly to point X, Y (absolute
mode)
      (princ " " gcode)                        ; blank
      (princ (strcat "X" (rtos X)) gcode)      ; Xxxxx
      (princ " " gcode)                        ; blank
      (princ (strcat "Y" (rtos Y)) gcode)      ; Yyyyy
      (princ "\n" gcode)                       ; end record -- a newline
     )
```

```
    ) ; end if
    (princ
      (strcat
          "N"
          (itoa (setq I (+ I 10)))
        ) gcode)                          ; Nxxxx - line number
    (princ " " gcode)                     ; blank
    (setq TG (cdr (assoc 42 PAL))) ;extract tangent of bulge
    (if (= TG 0) ; testing if vertex start a segment line or start an circular arc
    ; then –  vertex start a line
     (progn
      (princ "G01" gcode)                ; G01 -- moves to point xxxx yyyy
(absolute mode)
      (princ " " gcode)                  ; blank
      (princ (strcat "X" (rtos X)) gcode)    ; Xxxxx
      (princ " " gcode)                  ; blank
      (princ (strcat "Y" (rtos Y)) gcode)    ; Yyyyy
     )
    ; Else – vertex start an arc
     (progn
      (setq ATGR (* 4.0 (atan TG))) ; AngleArc = 4 * bulge arc - radian angle
      (setq ATG
       (cvunit ATGR "radian" "degree")  ;convert radian to degree angle
      )
      (setq C (osnap XY "_cen")) ; center of arc
      (setq R (distance XY C))   ; radius of arc
      (setq E (polar C ATGR R))  ; endpoint of arc
      (setq XI (- (nth 0 E) X))  ; incremental X of endpoint
      (setq YI (- (nth 1 E) Y))  ; incremental Y of endpoint
      (if (< TG 0) (setq R (* -1 R))) ; r=-r for clockwise
      (princ "G03" gcode)  ; G03 -- go in an CCW arc from current point to xxxx
yyyy
                          ; (Incremental point) with a  Rrrrr radius
      (princ " " gcode)                  ; blank
      (princ (strcat "X" (rtos XI)) gcode)    ; Xxxxx
      (princ " " gcode)                  ; blank
      (princ (strcat "Y" (rtos YI)) gcode)    ; Yyyyy
      (princ " " gcode)                  ; blank
      (princ (strcat "R" (rtos R)) gcode)    ; Rrrrr
     )
    ) ; end if
    (princ "\n" gcode)                    ; end record – a newline
   )  ; end prog
  ) ; end if
```

```
  (setq PAL (cdr PAL))
  (CNC PAL I) ; recursive call
 ) ; end prog
; else
 (progn
  (princ)  ; Exit quietly
 )
) ; end if
)  ; end CNC function
;;-----------------------------
;;
;; Main function
;;
(defun C:G_Code ()
 (setq PAL        ; retrieves the two-dimensional polyline association-list
  (entget (nth 0 (entsel "Please choose a 2D polyline object:")))
 )
 (setq gcode (open "GCODE.CNC" "w"))
 (CNC PAL 1000)
 (close gcode)
 (princ "\nFile GCODE.CNC created")
 (princ)
)  ; end G_code function
          ;;
          ;; End of CNC.LSP file
          ;;
```


General design specifications

ENTSEL function - Prompts the user to select a single object (entity) by specifying a point.
(entsel "Please choose a 2D polyline object:")

ENTGET function - Retrieves an object's (entity's) definition data - association-list of entity -.
(entget (nth 0 (entsel "Please choose a 2D polyline object:"))

OSNAP function - Returns a 3D point that is the result of applying an Object Snap mode to a specified point.
(osnap XY "_cen") ; center of arc

Starting the application

*Command: **(Load "CNC.LSP")**↵*
*Command: **G_CODE**↵*

Please choose a 2D polyline object:
File GCODE.CNC created

Where:

Command: -- AutoCAD prompter

CNC.LSP is the file, which contained the AutoLISP functions of the application.

DISTANCE function - Returns the 3D distance between two points.
(distance XY C)

POLAR function - . Returns the UCS 3D point at a specified angle and distance from a point.
(polar C ATGR R)

CVUNIT function - . Converts a value from one unit of measurement to another.
(cvunit ATGR "radian" "degree")
;convert radian to degree angle

C:G_CODE is the name of the main function of the application. Using the C:XXX feature, we have defined an AutoCAD command.
(defun **C:G_Code** () .Etc....)
G_CODE is now defined as a command, in addition to being an AutoLISP function. This means we can issue the command from the AutoCAD Command prompt like this:
Command: ***G_CODE*** ↵

The resulting tool path file *GCODE.CNC*, shown below, contains series of vector coordinate positions that define a cutting path (a cutting CNC program sequence) for a Flat Pattern machine - Flame Cutter. Tool Path is a series of vector coordinate positions that define a cutting path.

The resulting tool path file *GCODE.CNC*
(A cutting CNC program sequence that define a cutting path)

```
N1010 G21 G90
N1020 G00 X84.6485 Y68.051
N1030 G01 X84.6485 Y68.051
N1040 G03 X0.161 Y0.3927 R-12.4125
N1050 G03 X-15.7777 Y-1.5797 R12.4125
N1060 G03 X0.5124 Y5.1181 R-8.7207
N1070 G03 X-8.3059 Y-6.9884 R8.7207
N1080 G01 X171.2271 Y86.7857
N1090 G03 X-26.4603 Y-0.291 R13.2381
N1100 G03 X-9.2009 Y7.7425 R-8.0082
N1110 G03 X-2.5839 Y-14.8553 R8.0082
N1120 G03 X-9.4087 Y13.2175 R-9.8815
N1130 G01 X118.8388 Y171.0918
N1140 G01 X108.3611 Y193.6837
N1150 G01 X76.3767 Y142.9898
```