國立交通大學

機械工程研究所

碩士論文

數控工具機刀具半徑補正之研究與分析

The Cutter Radius Compensation for Computer Numerical Control Machine

研 究 生：黃朝群

指導教授：成維華 博士

中華民國九十三年六月

數控工具機刀具半徑補正之研究與分析

The Cutter Radius Compensation for Computer Numerical Control Machine

研究生：黃朝群　　　Student：chao-Qun Huang
指導教授：成維華　　　Advisor：Dr.Wei-Hua Chieng

國立交通大學
機械工程學系
碩士論文

A Thesis
Submitted to Department of Mechanical Engineering
College of Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Mechanical Engineering
June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

數控工具機刀具半徑補正之研究與分析

學生：黃朝群　　　　　　　　　　　　指導教授：成維華 博士

國立交通大學機械工程研究所碩士班

摘要

本篇論文最主要的參考資料為 Fanuc 工具機操作手冊，首先對手冊中基本程式指令功能做了解，接下來對其補償功能的指令做深入的探討。

其研究的目的在於推導電腦數值控制工具機的刀具半徑補正演算法，並將推導完成的數學公式以 C 語言實現之。為了驗證所推導的公式是否正確，利用 AutoCAD 軟體根據幾何關係建立不同型態的測試圖檔，將其在 AutoCAD 軟體裡所量測到的未補正座標輸入 C 語言所撰寫的程式，若程式所得到的輸出座標與 AutoCAD 軟體量測所得到的補正座標不符，則修改刀具半徑補正演算法與程式，直到兩者結果相符為止，方可得到正確的刀具半徑補正演算法。

若藉由本論文所發展的刀具半徑補正機制，則程式設計者只須依工件的實際尺寸編寫工具機程式，將可避免編寫程式時的許多困擾。

The Cutter Radius Compensation for Computer Numerical Control Machine

Student：Chao – Qun Huang   Advisor：Dr. Wei – Hua Chieng

Institute of Mechanical Engineering

National Chiao Tung University

ABSTRACT

The main reference of this thesis is Fanuc machine manual. First of all, I studied the basic program commands in the manual, and then investigated the commands of compensation in depth.

The purpose of this research is to derive the cutter radius compensation algorithm of computer numerical control machine and implement the derivational mathematics formula by C programming language. In order to test and verify if these formulas are correct, I made use of AutoCAD according to geometry to make different types of pattern, and input the uncompensated coordinates to C Programming language. If the output coordinates obtained from the program disagree with the compensated coordinates from AutoCAD, I have to modify the cutter radius compensation algorithm and my program until they correspond with each other. And then I got the correct cutter radius compensation algorithm.

If to rely on the cutter radius compensation function in this thesis, the programmers can compile the machine programs depending on geometric real size of workpieces and then diminish the mistakes when compiling program.

Acknowledgements

With accomplishing this thesis, I am going to achieve my Master degree. In this period, I would like to express my deepest gratitude to my adviser, Professor Wei-Hua Chieng, who has inspired me either with his professional knowledge or with the instruction in my researching attitude. In the meanwhile, I am also obliged to Prof. An-Qian Li, Prof. Ping-Yuan Bian and Prof. Fu-Hua Ren, my oral examiners, to their precious opinions that make my thesis much exact and perfect.

I thank my school alumni, especially Chang-Dao Yan, for giving me a favor unselfishly when I have problems on schoolwork. I am also grateful to my girl friend, Chu-Chun Hsu, for her refreshment when I was struggling in studying. I thank my good friend, Su-fang Chung, who gives helping hands in writing.

Last but not the least, I would like to express my love for my parents. Without their sacrifice, I might not have this achievement. I would also like to say thank you to my sister and brother for their care and supporting.

I am always indebted.

Contents

Page

List of Figures

Symbol Illustration

α：an angle of intersection created by tool paths

S：a position at which a single block is executed once.

L：the tool moves along a straight line.

C：the tool moves along an arc.

r：the cutter compensation value.

Chapter 1

Introduction

Nowadays, almost every machine control provides the functionality of automatic cutter radius compensation. This is a function that lets the control calculate the offset tool path based on the profile that is specified in the NC code. Cutter radius compensation is a process in which the machine controller automatically moves the cutter so that the edge cuts the programmed movement, rather than the center of the cutter following the programmed movement. As part of the job setup, the machine is told the radius of the cutter. This value is usually stored in a memory area called a tool offset. When the CNC program is run, commands are executed that instruct the controller to use a certain offset (D number) and which side of the programmed line to cut. [4]

There are several reasons why cutter radius compensation is so helpful. These reasons are our motive.

1. Changes in tool radius

Using a 1-cm radius end mill to machine the right side of a rectangular workpiece being held in a vise would be considered a simple operation by most experienced programmers. If a programmer prepares the program on the basis of the tool's centerline coordinates-not using cutter radius compensation the end mill must be kept away from the right side of the rectangular workpiece by precisely 0.5-cm throughout it motions. Imagine that the operator is making the setup and discovers the company is out of 1-cm and mills. There are 0.875-cm-radius mills and 1.25-cm-radius end mills, but no end mills left in 1-cm radius. In this case, the programmer would have to change the program in order to use and end mill radius other than the one programmed.

When cutter radius compensation is used, the operator is not forced to use the precise cutter radius programmed. Wile there are limitations with regard to minimum and maximum cutter sizes, a range of cutter sizes is possible. This allows the operator much more flexibility than if fixed centerline coordinates are used in the program. Also, no program modification is necessary if a cutter size must change. [1,5]

2. Deflection of the tool

Most machinists will agree that the cutting tool will seldom machine the workpiece as desired on the first try. The cutting tool, workpiece, and even the machine tool itself are under a great deal of pressure during machining. The more powerful the machining operation is, the greater the pressure is. Even when a milling cutter is kept quite rigid, there will be some deflection of the tool during machining. This is because the cutting edge of the tool will have a tendency to push away from surface being machined. Generally speaking, the weaker the machine tool and cutting tool are, the more potential the deflection is. While the small deflection may not be substantial enough to cause problems, there are times when it will. This is especially true when the accuracy required of the part is demanding. Also note that, as the cutter dulls, deflection will increase. This means a sharp cutter will have less deflection than a dull one. This change in deflection amount during the life of a cutting tool can present real headaches while machining if fixed center line coordinates are used in the program.

The programmer will be no longer necessary to repeatedly calculate center line coordinates by cutter radius compensation when the cutter dulls. [1,6]

3. Complex contours

A case can be made for using fixed centerlines for simple parts. However, as the surface to be milled becomes more complicated, calculating the centerline coordinates or the path of the end mill also becomes more complicated. If angular surfaces and radii must be machined, many times it can be difficult enough to calculate coordinates on the workpiece, let alone the coordinates for the centerline of the cutter.

When cutter radius compensation is used, it is no longer necessary that the programmer calculate the centerline coordinates of the tool. Only the workpiece surfaces need be calculated. While calculating the desired cutting path on the workpiece itself may be difficult, depending on how the print was dimensioned, at least the programmer does not worry about tool’s centerline coordinates. [1,4]

4. Roughing

The last reason we present for using cutter radius compensation has to do with roughing operations. Add to this the complication of having to allow for a constant amount of finishing stock throughout the surface to be machined during roughing. In essence, this doubles the amount of work the programmer must do. The programmer not only must calculate the centerline coordinates of the end mill during finishing, but also must calculate the centerline coordinates during the roughing operation.

When cutter radius compensation is used, it becomes very easy to make roughing asses. In fact the same series of coordinates used for the finishing pass can be used for roughing. How this is done involves understanding the offsets related to cutter radius compensation. For now, suffice it to say that there is a way to use the actual finishing coordinates and trick the control into allowing the desired finishing stock for a subsequent finishing tool. [1,5]

Some problems may take place when we use cutter radius compensation.

- Insufficient clearance at approach.

Almost all versions of cutter radius compensation require that you make a prior position movement in X and Y to get the tool to a position from which tool length compensation can be instated. With most controls, this prior position must be at least the cutter's radius away from the first surface to mill. If using a 1-inch radius cutter, for example, the tool must be at least 0.5 inch away from the first surface to mill. Note that with most controls, this prior position also determines the maximum cutter size that can be used. If the positioning movement stays 0.5 inch away from the surface, the largest cutter that will work is 1 inch in radius. By the way, this is one situation when a program that has successfully run before is now generating an alarm. The last time this program was run, the setup person used an appropriate cutter size, but today the cutter is larger. To avoid this problem, be sure to specify the maximum cutter size on the setup. [3,4]

- The milling cutter doesn't fit into a recess.

Once cutter radius compensation is instated, the control will simply keep he cutter on the right side or left side of all surfaces it sees coming up in the program. All current controls have a look-ahead feature that allows the control to scan at least a few commands into the program. As the cutter is moving along one surface, the control is looking ahead to see what is coming up in the program so it can end the current motion in the appropriate manner. With this look-ahead capability the control can also determine if the tool cannot completely machine one surface without violating another. If a surface is about to be violated, most controls will generate an over-cutting alarm. Finding this kind of problem can be difficult, especially if the drawing isn't made to scale. I recommend plotting the coordinates from your program on a piece of graph paper. Using a circle the same size as your cutter, try to moving the circle around the plotted path to see if the circle can move around all surfaces. [3,5]

- Attempting to machine multiple contours.

For each contour to be machined, you must instate, cut with and cancel cutter radius compensation. A common beginner's mistake is to instate cutter radius compensation once and then proceed to machine more that one contour. If you must rapid the tool to another surface, it should be take as a signal that you must cancel cutter radius compensation and then reinstate it one the next surface. [3,6]

- Forgetting to cancel.

If you do forget to cancel cutter radius compensation, it's likely that your series of motions will eventually break a cutter radius compensation rule and generate some kind of alarm. However, if no alarms are generated, you could be in for a nasty surprise. The next tool's movements will still be under the influence of cutter radius compensation, and of course its movements will not be correct. Say for example, that the tool following the milling cutter using cutter radius compensation is a drill. You forget to cancel cutter radius compensation, and as the drill is brought into its first position, you don't break any cutter radius compensation rules. The drill will machine its hole out of position by the amount stored in the cutter radius compensation offset register. But as you check the program, it's likely that you'll be checking the programmed coordinates for the drill, and of course, they are correct, It may take some time before it occurs to you to check whether cutter radius compensation is being canceled. [3]

- Offset is bigger than the smallest inside radius.

If using G02 or G03 to specify an inside(filet) radius, the tool must of course fit in the radius, meaning the radius of your milling cutter must be less that or equal to the radius you're trying to machine. [3]

There are some basic rules for cutter radius compensation.

- Typical controls need linear moves at begin and end of the machined profile (curve) to activate and cancel cutter compensation. In addition the length of these moves has to be bigger than the tool radius. [6]

- Ensure that the specified path curve does not include any undercut areas or arc moves that are smaller than the tool radius. Some modern controls may be able to handle such cases, but all others will most likely return some kind of compensation error message during machining. [6]

- Ensure that the operator specifies the correct offset value (tool radius) in the right place (offset register) of the machine controls memory. [5]

- Never use Cutter Comp in combination with multi-pass contouring. [6]

Chapter 2

Introduction to Computer Numerical Control

2.1 History of Computer Numerical Control

The technology of numerical control (NC) has been available since the middle of the twentieth century. Beginning with the development of a retrofit milling machine in the laboratories of MIT, NC has applications have grown worldwide to include drilling, turning, milling, welding, flame cutting, laser cutting, adhesives, and more.

In the early 1950's, NC machines were built using dedicated electronic controls. These hard-wired logic controls were designed to perform the control of one type of machine tool. Control wiring was tedious and time consuming. The term "NUMERICAL CONTROL" was derived from the fact that the NC programmer could specify the distance to be moved by the X axis using numerical data coded on paper tape. The programmer would prepare an NC program by typing the information on a typewriter-like device. There were two outputs generated from this action - one was a manuscript listing all the information on paper that had been typed (just like a typewriter); and the second output was a paper tape with holes punched in it. These holes represented data to the NC controller. These data were then translated by the control into the electrical signals required to effect the movement.

So how did we arrive at Computer Numerical Control (CNC)? In the early 1970's, INTEL Corporation developed the first computer on a chip known today as a microprocessor. Some original equipment manufacturers (OEMs) for NC machines opted to replace literally thousands of wires found in the hard-wired

NC controls with the soon-to-be-ubiquitous microprocessor. To differentiate their controls from competitors in the market place, these OEM's developed the name Computer Numerical Control. The word "COMPUTER" told potential clients that their machine had a computer in the controller and not just a hard-wired control.

2.2 NC and CNC

NC machines offered a reliable way of producing machine parts using pre-programmed commands. These commands consisted of alpha-numeric characters defined by the RS233 IEEE code. These characters were coded on punch paper tape in formats specifically planned for a certain machine tool. These programs (punched tape) would then be read into the NC control using a paper tape reader. If during testing a program error was detected, the paper tape would have to be edited. This process meant duplicating a tape up to the incorrect characters, retyping the correct characters, then continuing with the duplication process. Needless to say, this was time consuming. During the running of NC programs, if a tool would begin to wear causing part dimensions to approach tolerance limits, the operator would have to stop and adjust the tools to compensate for this wear.

CNC machines pick up where NC leaves off. Data for the control are still coded using either RS233 or the newer, more acceptable RSxyz ASCII code. Entire CNC programs may be loaded into the memory of the CNC control enabling the programmer or machine operator to edit the programs at the machine. If program changes are required, many CNC machines have built-in paper tape punch machines that allow for the generation of a new tape at the control. Tool wear is handled by adjusting program data in memory or calling in from a tool register one of several pre-programmed tool offsets.

As the implementation of microprocessors expanded, OEM's of NC/CNC machines began using them in the construction of controls. By the late 1970's, nearly all NC/CNC manufacturers were using microprocessors in their controls. Today the phrase NC is commonly used when referring to CNC machines because the need to differentiate the two no longer exists. Today, all modern NC machines are in-fact CNC machines.

2.3 Programming Terminology and Formats

NC programs are made up of a series of blocks or lines of code. A block is a series of words or commands. An NC word is a collection of related characters. Each character is punched on tape or stored in a computer memory in the form of a code as described earlier. When entered into a computer or punch tape preparation machine, an enter key is pressed at the end of each block.

Both NC and CNC machines require data to be entered in one of three programming formats.

- Fixed Block

With this format, each block of NC code consists of exactly the same number of characters. The words are fixed in length, so as the program is being read into the NC control memory, the computer counts character position to determine what data apply to what word.

- Tab Sequential

Tab sequential format relies on “tab” characters to separate each entered word. This format is characterized by an equal number of “tab” characters per block or line of code.

The tab characters result in the manuscript appearing in neat columns. In both fixed block and tab sequential formats, the programmer only enters the numeric data along with tabs and carriage returns. “Fixed Block” and “Tab Sequential” formats are no longer popular. They were more common on the hard-wired NC machines.

- Word Address

Word address has evolved as the format of choice and is the format used in nearly all CNC machines built today. With this format, the programmer actually types the letter of the word preceding the numeric data. There is virtually unlimited flexibility to programmer in the order of word entry and the length of each block. The control separates the entered numeric data into words by reading the "word address" prior to the numeric data. The words may be entered in any order although most users chose and follow a suggested order of entry to facilitate reading the manuscript.

Chapter 3

Fundamentals of Programming for CNC Machine

3.1 Preparatory Function

G codes are divided into the following two types.

- One–shot G code：The G code is effective only in the block in which it is specified. [2]

- Modal G code：The G code is effective until another G code of the same group is specified. [2]

3.2 Absolute and Incremental Function (G90, G91)

There are two ways to command travels of the tool: the absolute command and the incremental command. In the absolute command, coordinate value of the end position is programmed; in the incremental command, move distance of the position itself is programmed. G90 and G91 are used to command absolute or incremental command respectively. [2]

3.3 Interpolation Function

3.3.1 Positioning (G00)

The format of G00 is “G00 IP_;”. For an absolute command, IP_ is the coordinates of an end position and for an incremental command, IP_ is the distance the tool moves. The G00 command moves a tool to the position in the workpiece system specified with an absolute or an incremental command at a rapid traverse rate. In the absolute command, coordinate value of the end point is programmed. In the incremental command the distance the tool moves is programmed. For Nonlinear interpolation positioning, the tool is positioned with the rapid traverse rate for each axis separately and the tool path is normally straight. For linear interpolation positioning, the tool path is the same as in linear interpolation (G01) and the tool is positioned within the shortest possible time at a speed that is not more than the rapid traverse rate for each axis. [2]

3.3.2 Linear Interpolation (G01)

The format of G01 is "G01 IP _F_ ;". For an absolute command, IP_ is the coordinates of an end point, and for an incremental command, IP_ is the distance the tool moves. Besides, F_ is the speed of tool feed. A tool will move along a line to the specified position at the federate specified in F. The feedrate specified in F is effective until a new value is specified. It need not be specified for each block. The feedrate commanded by the F code is measured along the tool path. If the F code is not commanded, the feedrate is regarded as zero. [2]

3.3.3 Circular Interpolation (G02, G03)

For arc in the XY plane, the format of G02 and G03 is "G02 (G03) X_ Y_ I_ J_ (R_) F_ ;", for arc in the ZX plane, the format of G02 and G03 is "G02(G03) X_ Z_ I_ K_(R_) F_ ;" and for arc in the YZ plane, the format of G02 and G03 is "G02(G03) Y_ Z_ J_ K_(R_) F_ ;". [2] Description of the command format as below:

- G17：Specification of arc on XY plane.
- G18：Specification of arc on ZX plane.
- G19：Specification of arc on YZ plane.
- G02：Circular Interpolation Clockwise direction (CW).
- G03：Circular Interpolation Counterclockwise direction (CCW).
- X_：Command values of X axis or its parallel axis.
- Y_：Command values of Y axis or its parallel axis.
- Z_：Command values of Z axis or its parallel axis.
- I_：X axis distance from the start point to the center of an arc with sign.
- J_：Y axis distance from the start point to the center of an arc with sign.
- k_：Z axis distance from the start point to the center of an arc with sign.
- R_：Arc radius (with sign).
- F_：Feedrate along the arc.

The concept of the direction is important, “Clockwise (G02)” and “counterclockwise (G03)” on the XY plane (ZX plane or YZ plane) are defined when the XY plane is viewed in the positive–to–negative direction of the Z axis (Y axis or X axis, respectively) in the Cartesian coordinate system. There are two expressions for G02 and G03:

1. From the start point to the center of arc

The arc center is specified by addresses I, J, and K for the X, Y, and Z axes, respectively. The numerical value following I, J, or K, however, is a vector component in which the arc center is seen from the start point, and is always specified as an incremental value irrespective of G90 and G91. I, J, and K must be signed according to the direction. I0, J0, and K0 can be omitted. When X, Y, and Z are omitted (the end point is the same as the start point) and the center is specified with I, J, and K, a 360° arc (circle) is specified.

2. Arc radius

The distance between an arc and the center of a circle that contains the arc can be specified using the radius, R, of the circle instead of I, J, and K. In this case, one arc is less than 180°, and the other is more than 180° are considered. When an arc exceeding 180° is commanded, the radius must be specified with a negative value. If X, Y, and Z are all omitted, if the end point is located at the same position as the start point and when R is used, an arc of 0° is programmed.

If I, J, K, and R addresses are specified simultaneously, the arc specified by address R takes precedence and the other are ignored. If an axis not comprising the specified plane is commanded, an alarm is displayed. When an arc having a center angle approaching 180° is specified, the calculated center coordinates may contain an error. In such a case, specify the center of the arc with I, J, and K.

Chapter 4

Programming for Cutter Radius Compensation and Introduction to Compensation C

4.1 Cutter Radius Compensation Left (G41)

G41 offsets the tool towards the left of the workpiece as you see when you face in the same direction as the movement of the cutting tool.

For the G00 (positioning) or G01 (linear interpolation), the format is "G41 X_ Y_ I_ J_ H_;". It Specifies a new vector to be created at right angles with the direction of (I,J) on the end point, and the tool center moves toward the point of the new vector from that of the old vector on the start point. (I,J) are expressed in an incremental value from the end point, and is significant only as a direction, and its amount is arbitrary. In case the old vector is 0, this command specifies the equipment to enter from the cancel mode into the cutter compensation mode. At this time, the offset number is specified by the H code. Unless otherwise specified, (I,J) are assumed to be equal to (X,Y). When the format, "G41 X_ Y_ ;", is specified, a vector perpendicular to a line connecting the start position and position (X,Y) is created.

For the G02 and G03 (Circular interpolation), the format is "G41…;…G02 (or G03) X_ Y_ R_;". It specifies a new vector to be created to the left looking toward the direction in which an arc advances on a line connecting the arc center and the arc end point, and the tool center to move along the arc advancing from the point of the old vector on the arc start point toward that of the new vector. This is, however, established on assumption the old vector is created correctly.

The offset vector is created toward the arc center or opposite direction against the arc center. [2]

4.2 Cutter Radius Compensation Right (G42)

G42, contrary to G41, specifies a tool to be offset to the right of workpiece looking toward the direction in which the tool advances. G42 has the same function and format as G41, except that the directions of the vectors created by the commands are the opposite. [2]

4.3 Cutter Radius Compensation Cancel (G40)

The format of G40 is "G40 X_Y_". When G40 is specified in the G00 or G01 mode, the tool moves from the head of the old vector at the start position to the end position (X,Y). In the G01 mode, the tool moves linearly. In the G00 mode, rapid traverse is carried out along each axis. It changes the mode of the equipment from the cutter compensation mode to the cancel mode. When only G40 is specified and X _ Y _ is not specified, the tool moves by the old vector amount in the opposite direction. There is one thing that we must notice. Cutter compensation cannot be canceled in the circular interpolation (G02 and G03) mode. [2,6]

4.4 Tool Movement Mode (Fig.4-1)

4.4.1 Start-up Mode (Fig.4-2)

A process that offset cancel mode is changed to offset mode is called start-up mode.

- Tool movement around an inner side of a corner (α≧180°) (Fig.4-3a, Fig.4-3b)

- Tool movement around the outside of a corner at an obtuse angle (90°≦α< 180°) (Fig.4-4a ~ Fig.4-5b)

  Tool path in start–up has two types A and B.

- Tool movement around the outside of an acute angle (α<90°) (Fig.4-6a ~ Fig.4-7b)

  Tool path in start–up mode has two types A and B, too.

4.4.2 Offset Mode (Fig.4-8)

A process that tool movement mode has not been changed after cutter radius compensation is called offset mode.

- Tool movement around the inside of a corner (α≧180°) (Fig.4-9a ~ Fig.4-9d)

- Tool movement around the outside corner at an obtuse angle (90°≦α<180°) (Fig.4-10a ~ Fig.4-10d)

- Tool movement around the outside corner at an acute angle (α<90°) (Fig.4-11a ~ Fig.4-11d) [4]

4.4.3 Cancel Mode (Fig.4-12)

A process that offset mode is changed to offset cancel mode is called cancel mode.

- Tool movement around an inside corner (α≧180°) (Fig.4-13a, Fig.4-13b)

- Tool movement around an outside corner at an obtuse angle (90°≦α＜180°) (Fig.4-14a, Fig.4-15b)

  Tool path has two types, A and B.

- Tool movement around an outside corner at an acute angle (α<90°) (Fig.4-16a, Fig.4-17b)

  Tool path has two types, A and B, too. [6]

4.5 Interference Check

4.5.1 Basic Definition

Tool overcutting is called interference. The interference check function checks for tool overcutting in advance. However, not all interference can be checked by this function. The interference check is still performed even if overcutting does not occur. [2]

4.5.2 Algorithm for Detecting Interference

According to the interference check of Fanuc machine manual, we try to derive the interference check algorithm. Our theory is that interference can be judge by vector intersection. [2,9]

Chapter 5
Mathematical Analysis and Derivation

5.1 Start-up Mode

5.1.1 α≧180°

- Line to Line (Fig.4-3a)

Our purpose is to obtain the point p_2. First we calculate the unit vector of $\overrightarrow{s_2 s_3}$ and let $\frac{\overrightarrow{s_2 s_3}}{\left|\overrightarrow{s_2 s_3}\right|} = (c,d)$. Because the distance between the point s_2 and the point p_2 is the cutter radius r, and the point p_2 is the projective point of the point s_2 on $\overrightarrow{p_2 p_3}$,

$$p_2 = s_2 + r(-d,c) \times n\,,\quad n = \begin{cases} 1, & if \quad G41 \\ -1, & if \quad G42 \end{cases} \tag{5-1}$$

- Line to Arc (Fig.4-3b)

We want to calculate the point p_2. By the point s_2 and the point c, we can obtain $\vec{R} = s_2 - C$, where the point c is the center of $\overset{\frown}{s_2 s_3}$. Since the point p_2 is the projective point of the point s_2 on the line which is parallel to the tangent line of $\overset{\frown}{s_2 s_3}$ at the point s_2,

$$p_2 = s_2 + r\frac{\vec{R}}{\left|\vec{R}\right|} \times n\,,\quad n = \begin{cases} 1, & if \quad G41\&G02 \quad or \quad G42\&G03 \\ -1, & if \quad G42\&G02 \quad or \quad G41\&G03 \end{cases} \tag{5-2}$$

where r is the cutter radius.

5.1.2 90°≦α＜180°

5.1.2.1 Type A

- Line to Line (Fig.4-4a)

  The method to calculate the point p_2 is the same as the eq.(5-1).

- Line to Arc (Fig.4-4b)

  The method to calculate the point p_2 is the same as the eq.(5-2).

5.1.2.2 Type B

- Line to Line (Fig.4-5a)

  The point p_{2_1}, the point p_{2_2} and the point p_{2_3} will come into existence after cutter radius compensation. The point p_{2_1} and the point p_{2_3} can be gained by the same method as the eq.(5-1). Because $\overrightarrow{p_{2\_1}p_{2\_2}}$ is parallel to $\overrightarrow{s_1s_2}$, and $\overrightarrow{p_{2\_2}p_{2\_3}}$ is parallel to $\overrightarrow{s_2s_3}$, by the relation of these vector measures, we can obtain the following simultaneous equation：

$$\begin{cases} \dfrac{p_{2\_2x}-p_{2\_1x}}{s_{2x}-s_{1x}}=\dfrac{p_{2\_2y}-p_{2\_1y}}{s_{2y}-s_{1y}} \\ \dfrac{p_{2\_3x}-p_{2\_2x}}{s_{3x}-s_{2x}}=\dfrac{p_{2\_3y}-p_{2\_2y}}{s_{3y}-s_{2y}} \end{cases} \tag{5-3}$$

Then we can solve the eq.(5-3) by Gramer Rule：

$$p_{2\_2}=\frac{1}{\begin{vmatrix} b & -a \\ d & -c \end{vmatrix}}\left(\begin{vmatrix} bx_0-ay_0 & -a \\ dx_1-cy_1 & -c \end{vmatrix},\begin{vmatrix} b & bx_0-ay_0 \\ d & dx_1-cy_1 \end{vmatrix}\right) \tag{5-4}$$

where x_0 is the x-coordinate of the point p_{2_1}, y_0 is the y-coordinate of the point p_{2_1}, x_1 is the x-coordinate of the point p_{2_3}, y_1 is the y-coordinate of the point p_{2_3}, (a,b) is the unit vector of $\overrightarrow{s_1s_2}$, and (c,d) is the unit vector of $\overrightarrow{s_2s_3}$.

- Line to Arc (Fig.4-5b)

The point p_{2_1} can be gained by the same method as the eq.(5-1), and the point p_{2_3} can be gained by the same method as the eq.(5-2). The point p_{2_2} can be also gained by the same method as the eq.(5-4). But, it is necessary to make a modification. Thus

$$(c,d)=(-z,w)\times n\,,\quad n=\begin{cases} 1, & if \quad G41\&G02 \quad or \quad G42\&G03 \\ -1, & if \quad G42\&G02 \quad or \quad G41\&G03 \end{cases} \tag{5-5}$$

where the vector $(-z,w)$ is perpendicular to the unit vector of $\vec{R}$.

5.1.3 α＜90°

5.1.3.1 Type A

- Line to Line (Fig.4-6a)

The point p_2 can be also obtained by the eq.(5-1).

- Line to Arc (Fig.4-6b)

The point p_2 can be also obtained by the eq.(5-2).

5.1.3.2 Type B

- Line to Line (Fig.4-7a)

The point p_{2_1} and the point p_{2_4} can be obtained by the same method as the eq.(5-1). After we obtain the point p_{2_1} and the point p_{2_4}, the point p_{2_2} and the point p_{2_3} will be able to be obtained as following：

$$p_{2\_2} = p_{2\_1} + r(a,b) \tag{5-6}$$

since $\overrightarrow{s_2 p_{2\_1}} \perp \overrightarrow{p_{2\_1} p_{2\_2}}$ and $\left|\overrightarrow{s_2 p_{2\_1}}\right| = \left|\overrightarrow{p_{2\_1} p_{2\_2}}\right|$.

Where r is the cutter radius and (a,b) is the unit vector of $\overrightarrow{s_1 s_2}$.

$$p_{2\_3} = p_{2\_4} - r(c,d) \tag{5-7}$$

since $\overrightarrow{s_2 p_{2\_4}} \perp \overrightarrow{p_{2\_4} p_{2\_3}}$ and $\left|\overrightarrow{s_2 p_{2\_4}}\right| = \left|\overrightarrow{p_{2\_4} p_{2\_3}}\right|$.

Where r is the cutter radius and (c,d) is the unit vector of $\overrightarrow{s_2 s_3}$.

- Line to Arc (Fig.4-7b)

How should we calculate the point p_{2_1}, the point p_{2_2}, the point p_{2_3} and the point p_{2_4}? Using the same methods as the eq.(5-1), the eq.(5-2), the eq.(5-6) and the eq.(5-7) is direct thoughts. However, the eq.(5-7) has to be modified in this case. By comparing the Fig.4-5a with the Fig.4-5b, we can make a change in the eq.(5-7) by the eq.(5-5). Then all points will be able to be obtained.

5.2 Offset Mode

5.2.1 α≧180°

- Line to Line (Fig.4-9a)

It is obvious that the point p_2 is the intersectional point of the two parallel lines. The concept to calculate the point p_2 is similar to the eq.(5-4). First we have to obtain the point p_1 and the point p_3 by the eq.(5-1) and then take advantage of the relation of these vector measures as the eq.(5-3) to obtain the point p_2. Finally, the point p_2 will be able to be solved by Gramer Rule as the eq.(5-4).

- Line to Arc (Fig.5-1)

From the figure, the point p_2 is the intersectional point of the parallel line and the concentric circle. It seems that solving the point p_2 is a difficult problem. In order to calculate the point p_2, we make a projective point of the point c on $\overrightarrow{p_1p_2}$ to obtain the point c' and let the point $c'=(x_0+at, y_0+bt)$, where x_0 which is calculated by the eq.(5-1) is the x-coordinate of the point p_1, y_0 which is calculated by the eq.(5-1) is the y-coordinate of the point p_1, and (a,b) is the unit vector of $\overrightarrow{s_1s_2}$. However, the variable t is still unknown. Employing $\overrightarrow{p_1p_2}$ and $\overrightarrow{cc'}$ is perpendicular to one another：

$$\frac{c_x-(x_0+at)}{c_y-(y_0+bt)}\times\frac{a}{b}=-1 \tag{5-8}$$

Afterward we can solve the variable t by the eq.(5-8)：

$$t=\frac{a(c_x-x_0)+b(c_y-y_0)}{a^2+b^2} \tag{5-9}$$

Then the point c' will be obtained consequently.

Finally, we try to obtain the point p_2. Because $\triangle cc'p_2$ is a right triangle in this figure, according to Pythagorean Theorem,

$$\left|\overrightarrow{c'p_2}\right|=\sqrt{(R')^2-\left|\overrightarrow{cc'}\right|^2} \tag{5-10}$$

where the radius R' is the compensated radius of $\overset{\frown}{s_2s_3}$ and

$R'=R+r,\quad if\quad G41\&G02\quad or\quad G42\&G03$

$R'=R-r,\quad if\quad G42\&G02\quad or\quad G41\&G03$

Nevertheless, it is possible that the point p_2 appears in front of the point c' or in back of the point c' on $\overrightarrow{p_1p_2}$ because of the different angle α and the type of arcs. For this reason, we can use $\overrightarrow{cs_2}$ and $\overrightarrow{cc'}$ to make a judgment, and let $\overrightarrow{cs_2}=(u,v)$ and $\overrightarrow{cc'}=(w,z)$. By the eq.(5-9) and the eq.(5-10),

$$p_2=c'+\operatorname{sgn}(uz-vw)\times n\cdot\left|\overrightarrow{c'p'}\right|(a,b),\quad n=\begin{cases}1, & if\quad G02\\ -1, & if\quad G03\end{cases} \tag{5-11}$$

where (a,b) is the unit vector of $\overrightarrow{s_1s_2}$.

- Arc to Line **(**Fig.5-2**)**

The idea of solving the point p_2 is the same as the eq.(5-11).

$$p_2=c'+\operatorname{sgn}(uz-vw)\times n\cdot\left|\overrightarrow{c'p'}\right|(c,d),\quad n=\begin{cases}1, & if\quad G02\\ -1, & if\quad G03\end{cases} \tag{5-12}$$

where (c,d) is the unit vector of $\overrightarrow{s_2s_3}$.

- Arc to Arc **(**Fig.5-3**)**

Up to now, calculating the point p$_2$ is the hardest problem we have met. The point p$_2$ is the intersectional point of the two concentric circles in this figure. First we decide the compensated radius of $\overset{\frown}{s_1s_2}$ and $\overset{\frown}{s_2s_3}$. Namely,

$$R_1' = R_1 + r \quad \& \quad R_2' = R_2 + r, \quad if \quad G41\&G02 \quad or \quad G42\&G03$$

$$R_1' = R_1 - r \quad \& \quad R_2' = R_2 - r, \quad if \quad G42\&G02 \quad or \quad G41\&G03$$

where r is the cutter radius.

For $\triangle p_2p_2'c_2$ and $\triangle p_2p_2'c_1$ are right triangles and own the identical height. In the light of Pythagorean Theorem：

$$(R_1')^2 - |c_1p_2'|^2 = (R_2')^2 - (|c_1c_2| - |c_1p_2'|)^2 \tag{5-13}$$

Afterward we can solve the variable $|c_1p_2'|$ by the eq.(5-13)：

$$|c_1p_2'| = \frac{(R_1')^2 - (R_2')^2 + |\overrightarrow{c_1c_2}|^2}{2|\overrightarrow{c_1c_2}|} \tag{5-14}$$

By the eq.(5-14),

$$p_2' = c_1 + |c_1p_2'| \cdot \frac{\overrightarrow{c_1c_2}}{|\overrightarrow{c_1c_2}|} \tag{5-15}$$

However, it is possible that the point p_2 appears on the right side of $\overrightarrow{c_1c_2}$ or on the left side of $\overrightarrow{c_1c_2}$ because of the different angle α and the type of arcs. For this reason, we can use $\overrightarrow{R_1}$ and $\overrightarrow{c_1c_2}$ to make a judgment, and let $\overrightarrow{R_1} = s_2 - c_1 = (R_{1x}, R_{1y})$ and $\overrightarrow{c_1c_2} = (c_x, c_y)$. Finally, according to the eq.(15),

$$p_2 = p_2' + \mathrm{sgn}(R_{1x}C_y - R_{1y}C_x)\sqrt{R_1'^2 - |\overrightarrow{cc'}|^2}\,(b, -a) \tag{5-16}$$

where (a,b) is the unit vector $\overrightarrow{c_1c_2}$ and $\sqrt{R'^2 - |\overrightarrow{cc'}|^2}$ is the magnitude of $\overrightarrow{p_2'p_2}$.

5.2.2 90°≦α＜180°

- Line to Line **(**Fig.4-10a**)**

It will be easy to use the same method as the eq.(5-4) to calculate the point p_2.

- Line to Arc **(**Fig.4-10b**)**

It is not correct to use the eq.(5-4) to calculate the point p_2 until we make a change in the eq.(5-4) by the eq.(5-5).

- Arc to Line **(**Fig.4-10c**)**

In order to use the eq.(5-4) to calculate the point p_2, we have to make a modification in the eq.(5-4). Thus

$$(a,b) = (-v,u)\times n\,,\quad n = \begin{cases} 1, & if \quad G41\&G02 \quad or \quad G42\&G03 \\ -1, & if \quad G42\&G02 \quad or \quad G41\&G03 \end{cases} \tag{5-17}$$

where the vector $(-v,u)$ is perpendicular to the unit vector of $\vec{R}$.

- Arc to Arc **(**Fig.4-10d**)**

If we want to utilize the eq.(5-4) to calculate the point p_2, we have to make two changes in the eq.(5-4) by the eq.(5-5) and the eq.(5-17).

5.2.3 $\alpha < 90^\circ$

- Line to Line (Fig.4-11a)

The point p_{2_1} and the point p_{2_4} can be obtained by the same method as the eq.(5-1). After we obtain the point p_{2_1}, the point p_{2_4}, the point p_{2_2} and the point p_{2_3} will be able to be obtained by the same methods as the eq.(5-6) and the eq.(5-7), too.

- Line to Arc (Fig.4-11b)

If we revise the eq.(5-7) by the eq.(5-5), we will be able to obtain these points in this figure by the eq.(5-1), the eq.(5-2), the eq.(5-6) and the eq.(5-7).

- Arc to Line (Fig.4-11c)

If we revise the eq.(5-6) by the eq.(5-17), all points in this figure will be able to be calculated by the eq.(5-1), the eq.(5-2), the eq.(5-6) and the eq.(5-7).

- Arc to Arc (Fig.4-11d)

If we want to calculate these points in this figure by the eq.(5-1), the eq.(5-2), the eq.(5-6) and the eq.(5-7), we have to make two changes. Namely, we revise the eq.(5-7) by the eq.(5-5), and the eq.(5-6) by the eq.(5-17).

5.3 Cancel Mode

5.3.1 $\alpha \geqq 180°$

- Line to Line (Fig.4-13a)

If we use the same method as the eq.(5-1), the point p_2 will be able to be obtained.

- Arc to Line (Fig.4-13b)

If we use the same method as the eq.(5-2), the point p_2 will be able to be obtained.

5.3.2 $90^\circ \leqq \alpha < 180^\circ$

5.3.2.1 Type A

- Line to Line (Fig.4-14a)

  The point p_2 will be able to be obtained by the eq.(5-1).

- Arc to Line (Fig.4-14b)

  The point p_2 will be able to be obtained by the eq.(5-2).

5.3.2.2 Type B

- Line to Line (Fig.4-15a)

  We just only to calculate the point p_{2_1} and the point p_{2_3} by the same method as the eq.(5-1) first ,and then using the same method as the eq.(5-4) will be quite convenient to calculate the point p_{2_2}.

- Arc to Line (Fig.4-15b)

  As above, we can calculate the point p_{2_1} and the point p_{2_3} by the eq.(5-1) first and make a modification in the eq.(5-4) by the eq.(5-17). Then the point p_{2_2} will be able to be calculated by the eq.(5-4).

5.3.3 $\alpha < 90^\circ$

5.3.3.1 Type A

- Line to Line (Fig.4-16a)

  The method to calculate the point p_2 is the equal to the eq.(5-1).

- Arc to Line (Fig.4-16b)

  The method to calculate the point p_2 is the equal to the eq.(5-2).

5.3.3.2 Type B

- Line to Line (Fig.4-17a)

  The point p_{2_1}, the point p_{2_2}, the point p_{2_3} and the point p_{2_4} will come into existence after cutter radius compensation. The point p_{2_1} and the point p_{2_4} can be obtained by the same method as the eq.(5-1). The point p_{2_2} and the point p_{2_3} will be able to be obtained by the same methods as the eq.(5-6) and the eq.(5-7) after we obtain the point p_{2_1} and the point p_{2_4}.

- Arc to Line (Fig.4-17b)

  All points in this figure will be able to be obtained by the eq.(5-1), the eq.(5-2), the eq.(5-6) and the eq.(5-7) after we modify the eq.(5-6) by the eq.(5-17).

5.4 Calculating α Angle

5.4.1 Line to Line (Fig.4-9a)

We would like to make use of the dot product of two vectors to calculate the angle α of a workpiece. If we let $\overrightarrow{s_1s_2}=(x,y)$ and $\overrightarrow{s_2s_3}=(u,v)$,the angle θ between $\overrightarrow{s_1s_2}$ and $\overrightarrow{s_2s_3}$ is as following：

$$\cos^{-1}\left(\frac{xu+yv}{\left|\overrightarrow{s_1s_2}\right|\left|\overrightarrow{s_2s_3}\right|}\right)=\theta \quad (0^\circ \le \theta \le 180^\circ) \tag{5-18}$$

According to the relation between the angle α and the angle θ,

$$\alpha=\begin{cases}\pi+\theta, & if \quad G41 \quad \& \quad (xv-yu)>0 \quad or \quad G42 \quad \& \quad (xv-yu)<0\\ \pi-\theta, & for \quad the \quad other \quad conditions\end{cases} \tag{5-19}$$

5.4.2 Line to Arc (Fig.4-9b)

If we use the eq.(5-5) to revise the eq.(5-18), the angle α can be calculated by the eq.(5-19).

5.4.3 Arc to Line (Fig.4-9c)

If we use the eq.(5-17) to revise the eq.(5-18), the angle α can be calculated by the eq.(5-19).

5.4.4 Arc to Arc (Fig.4-9d)

We have to revise the eq.(5-18) by the eq.(5-5) and the eq.(5-17) in order to calculate the angle α.

5.5 Interference Check

If $\overrightarrow{v1}$ intersects $\overrightarrow{v2}$, interference may occur. Nevertheless, how should we judge vector intersection? First we calculate $\overrightarrow{v3}$, and extend $\overrightarrow{v1}$ and $\overrightarrow{v2}$ to obtain the intersectional point (ix, iy). (Fig.5-4a)

$$\begin{cases} v3x = vtx + v2x \\ v3y = vty + v2y \end{cases} \tag{5-20}$$

By the relation of these vector measures,

$$\begin{cases} \dfrac{iy - 0}{ix - 0} = \dfrac{v1y}{v1x} \\ \dfrac{iy - v3y}{ix - v3x} = \dfrac{v2y}{v2x} \end{cases} \tag{5-21}$$

In order to simplify the expression of the intersectional point (ix, iy), we can let $b = v3x \cdot v2y - v2x \cdot v3y$ and $\det = v1x \cdot v2y - v1y \cdot v2x$. Then the intersectional point (ix, iy) can be solved.

$$\begin{cases} ix = v1x \cdot b / \det \\ iy = v1y \cdot b / \det \end{cases} \tag{5-22}$$

If the intersectional point (ix, iy) is on $\vec{v1}$ and $\vec{v2}$, we can confirm that $\vec{v1}$ intersects $\vec{v2}$. (Fig.5-4b)

Thus we can take advantage of the following two simultaneous inequalities to make a judgment.

$$\begin{cases} ix \cdot v1x \geq 0 \\ ix \cdot ix \leq v1x \cdot v1x \end{cases}, \quad if \quad (ix, iy) \quad is \quad on \quad \vec{v1}, \tag{5-23}$$

$$\begin{cases} ix2 \cdot v2x \geq 0 \\ ix2 \cdot ix2 \leq v2x \cdot v2x \end{cases}, \quad if \quad (ix, iy) \quad is \quad on \quad \vec{v2} \tag{5-24}$$

where $ix2$ can be obtained by ix and $\vec{vt}$.

Chapter 6

Conclusion

Formerly, the programs of machine were compiled by the cutter center deflected from one cutter radius along geometric shape of workpiece. However, for some complicated geometric workpieces, it usually had to apply a large number of complicated mathematics calculations. If to rely on the cutter radius compensation function in this thesis, the programmers can consider the cutter radius as zero directly and compile the machine programs depending on geometric shape of workpieces, real size, coordinate positions to avoid some perplexities and diminish the mistakes when compiling program. Furthermore, this function also allows the user doesn't make any modification and still can cut the same workpieces when using different cutter radius.

References

[1]Mike Lynch, Computer Numerical Control for Machining, McGraw-Hill, New York, 1994.

[2]Fanuc, Programming manual for Fanuc series 15-MA, North America, 2001.

[3]Mike Lynch, “Five common problems with cutter radius compensation ”, Modern Machine Shop, August 2001.

[4]Michael Sava, Joseph Pusztai, Computer Numerical Control Programming, Prentice Hall, 1992.

[5]Warren S. Seames, Computer Numerical Control Concepts & Programming, Third Edition, Delmar Publishers, 1995.

[6]Kelly Curran, Jon Stenerson, Computer Numerical Control ,Prentice Hall,1997.

[7]Allan Hansen, Farhad Arbab, “Fixed-axis tool positioning with built-in global interference checking for NC path generation”, Robotics and Automation, IEEE Journal of [see also IEEE Transactions on Robotics and Automation], Volume: 4, Issue: 6, Dec. 1988.

[8]Federico Thomas, Carme Torras, “Interference detection between non-convex polyhedra revisited with a practical aim”, Robotics and Automation, 1994. Proceedings, 1994 IEEE International Conference on, 8-13 May 1994.

[9]H. Noborio, S. Fukuda, S. Arimoto, ”A new interference check algorithm using octree”, Robotics and Automation, 1987 IEEE International Conference on , Volume: 4 , Mar 1987.

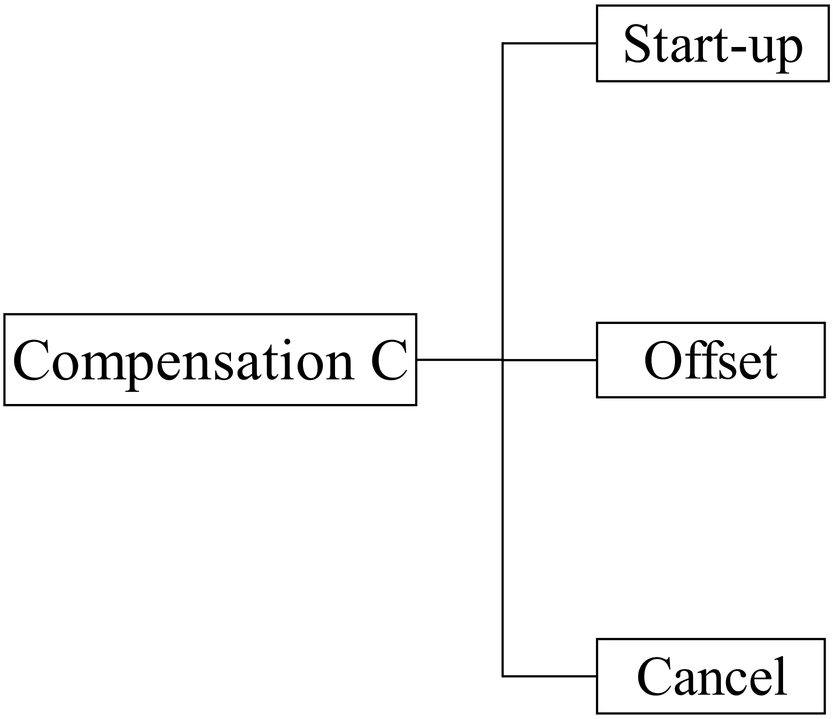


Fig.4-1 FANUC Cutter Compensation C

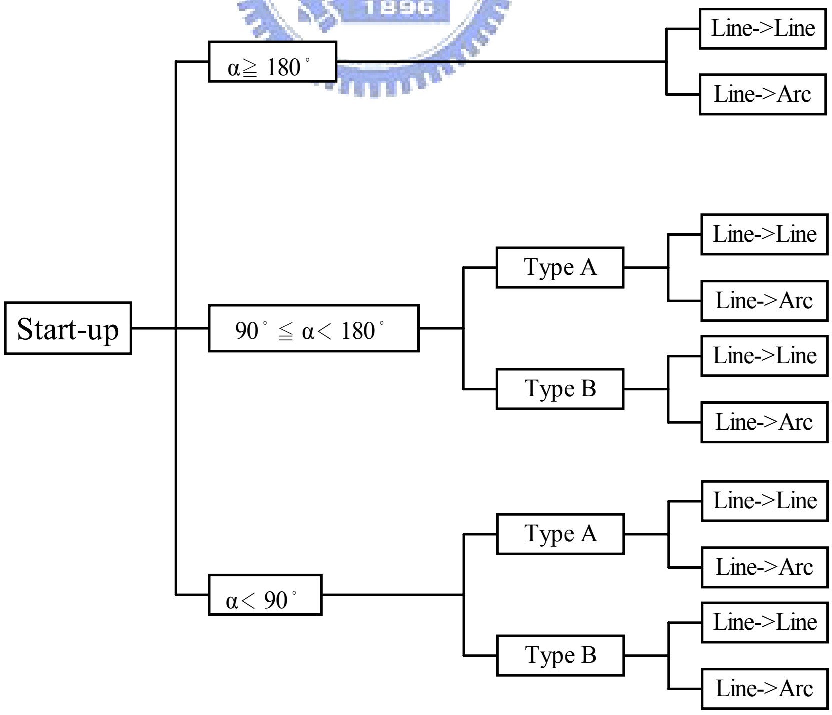


Fig.4-2 Start-up Mode

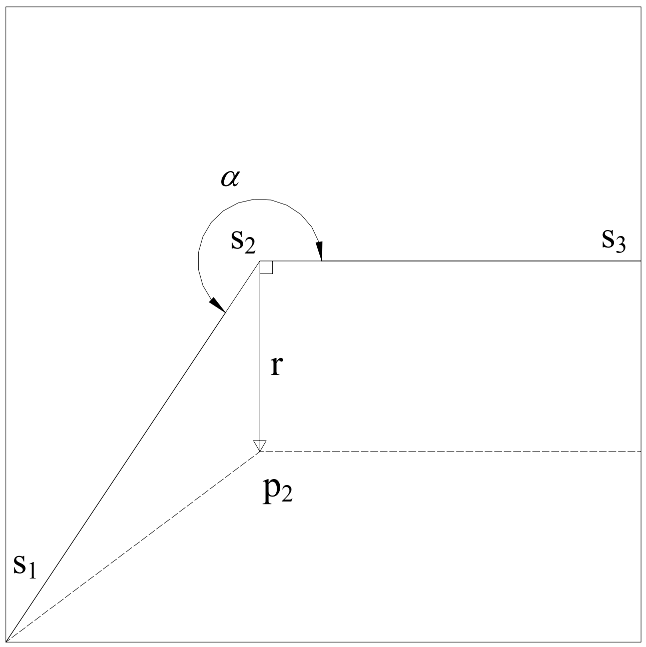


Fig.4-3a Start-up Mode：Line → Line (α≧180°)

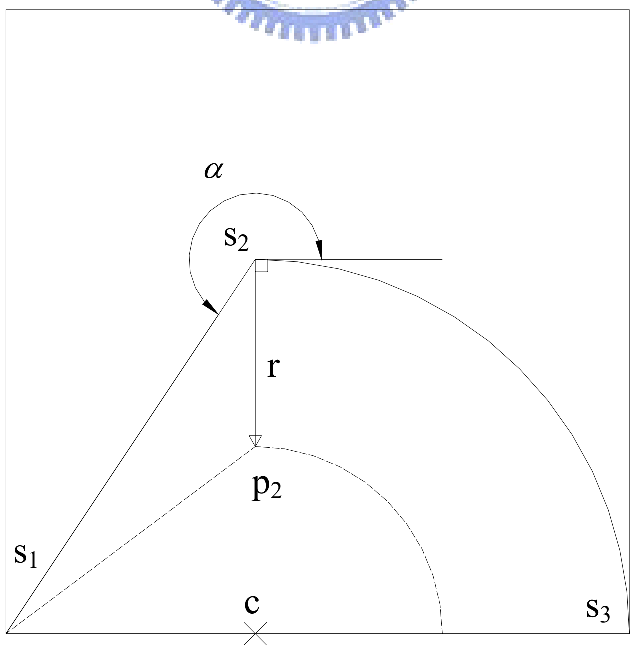


Fig.4-3b Start-up Mode：Line → Arc (α≧180°)

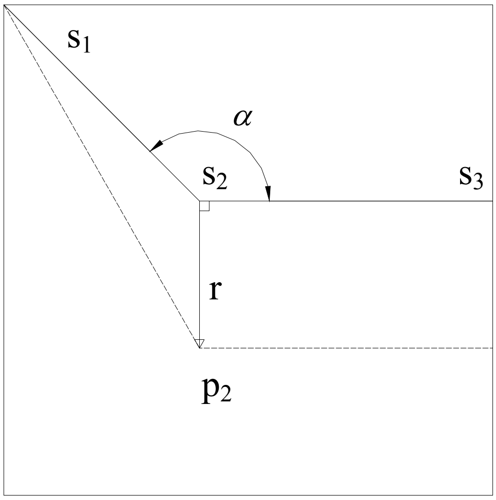


Fig.4-4a Start-up Mode：Line → Line (90°≦α＜180°-Type A)

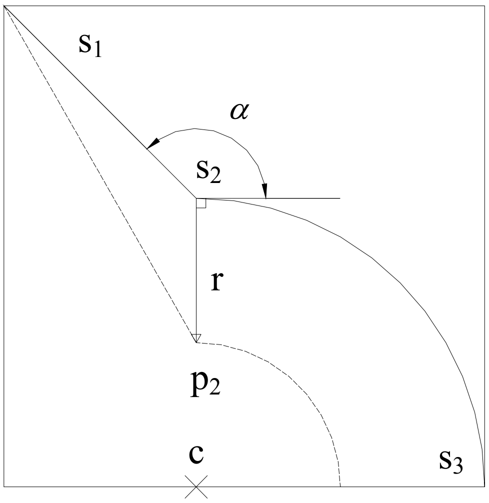


Fig.4-4b Start-up Mode：Line → Arc (90°≦α＜180°-Type A)

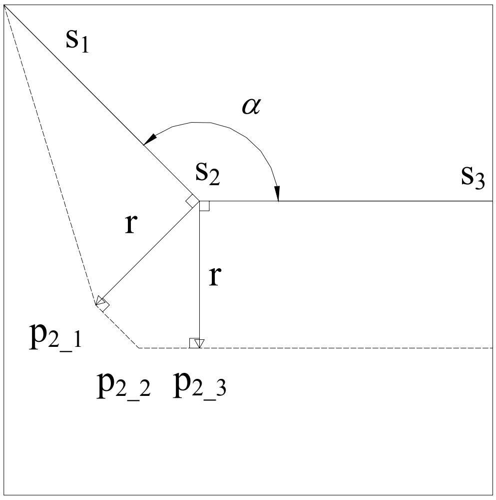


Fig.4-5a Start-up Mode：Line → Line (90°≦α＜180°-Type B)

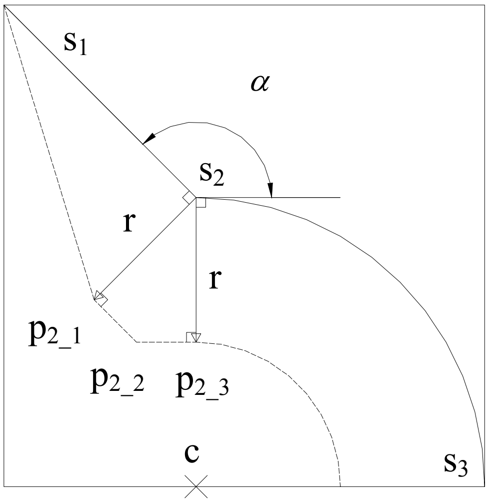


Fig.4-5b Start-up Mode：Line → Arc (90°≦α＜180°-Type B)

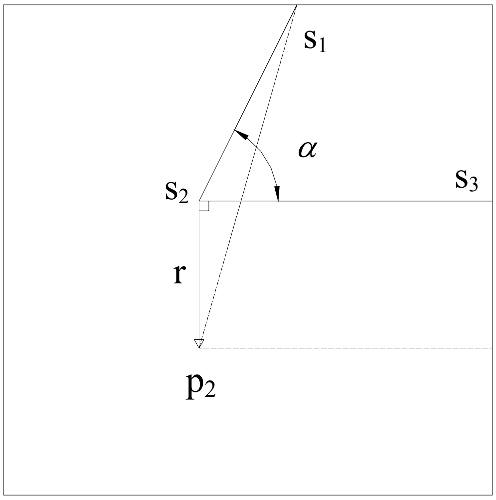


Fig.4-6a Start-up Mode：Line → Line (α＜90° -Type A)

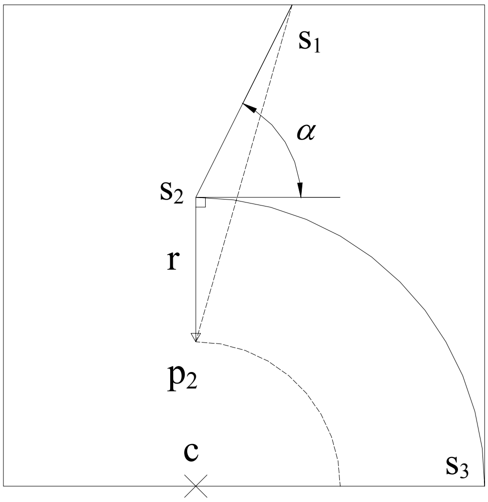


Fig.4-6b Start-up Mode：Line → Arc (α＜90° -Type A)

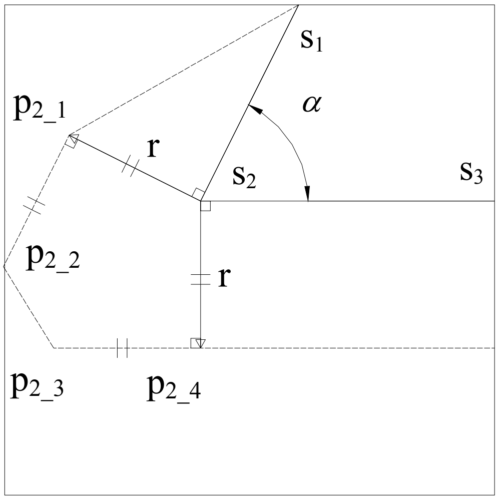


Fig.4-7a Start-up Mode：Line → Line (α＜90° -Type B)

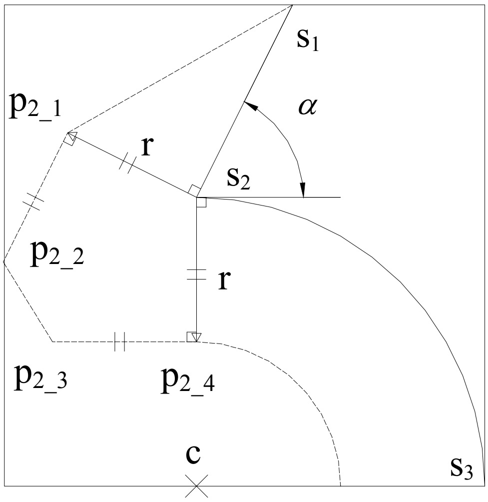


Fig.4-7b Start-up Mode：Line → Arc (α＜90° -Type B)

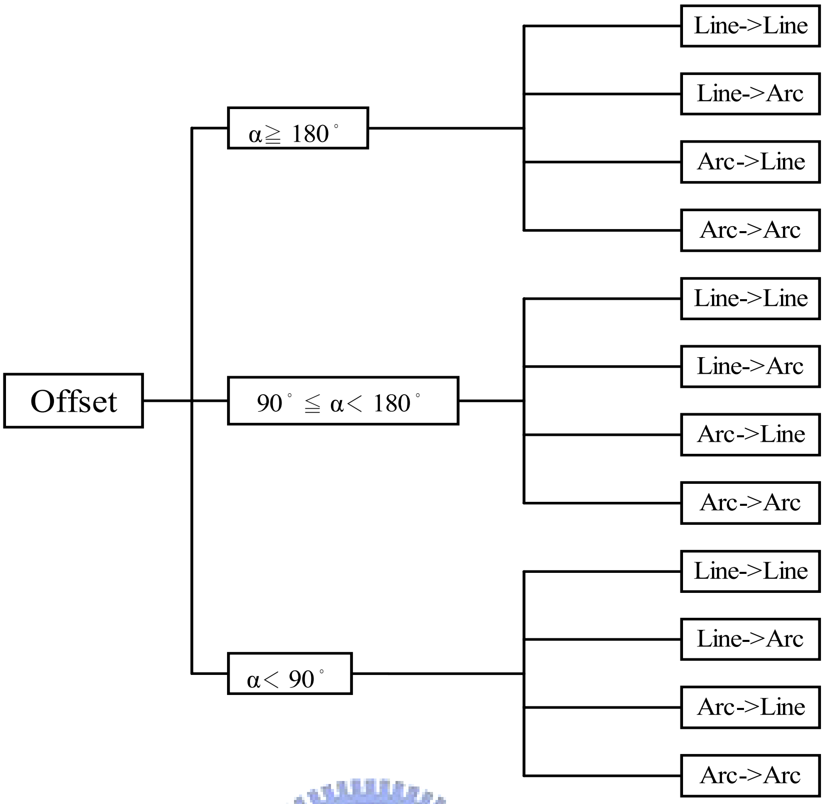


Fig.4-8 Offset Mode (α<90°)

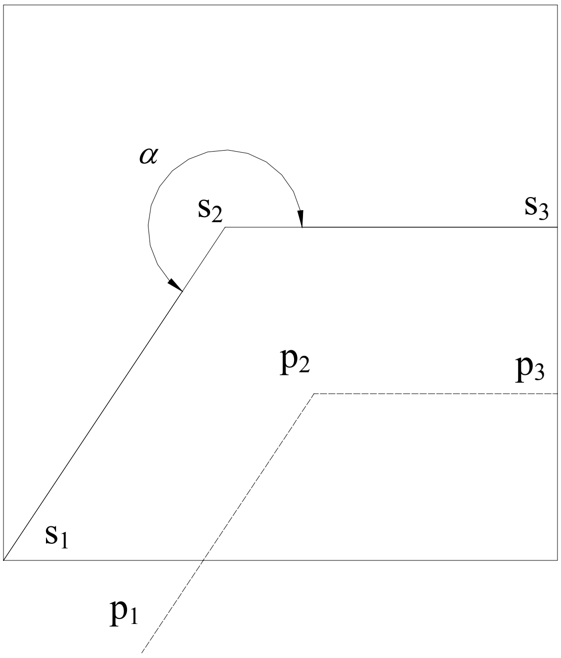


Fig.4-9a Offset Mode：Line → Line (α≧180°)

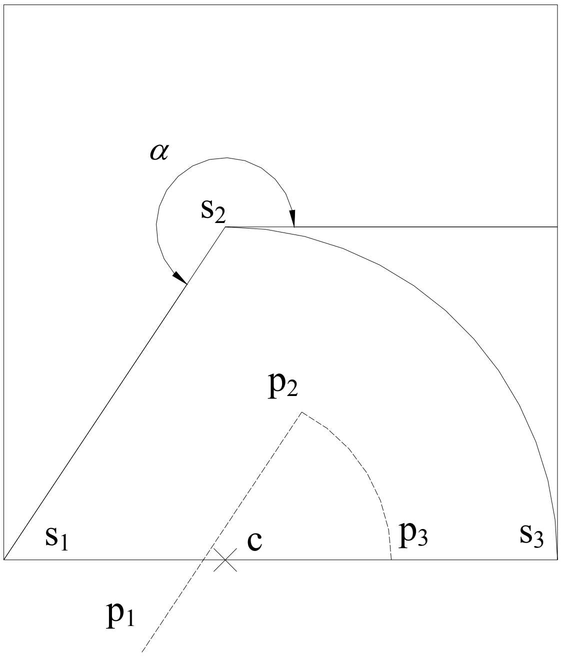


Fig.4-9b Offset Mode：Line → Arc ($\alpha \geqq 180^\circ$)

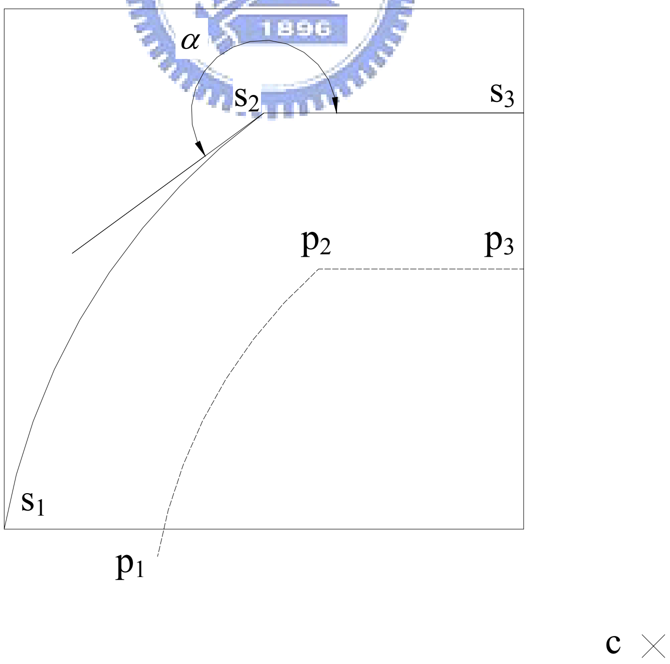


Fig.4-9c Offset Mode：Arc → Line ($\alpha \geqq 180^\circ$)

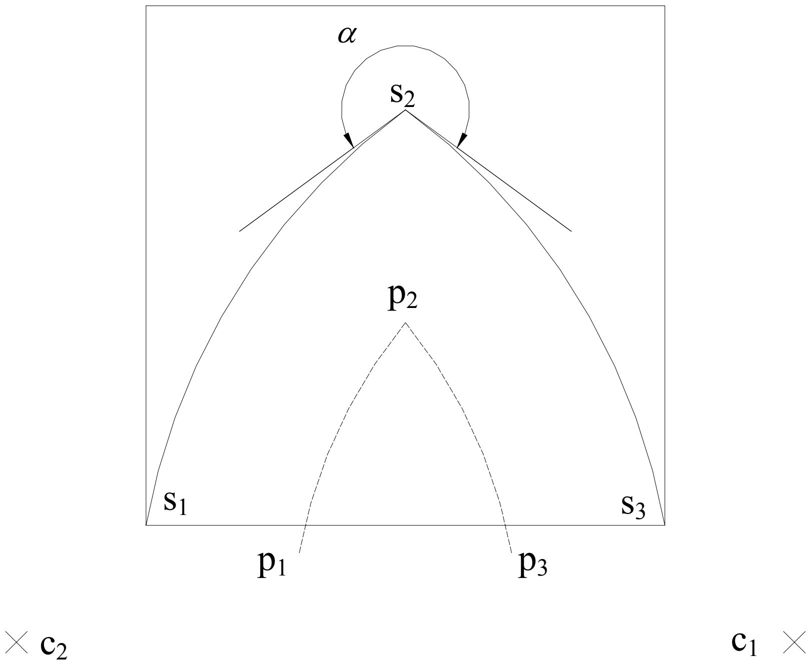


Fig.4-9d Offset Mode：Arc → Arc (α≧180°)

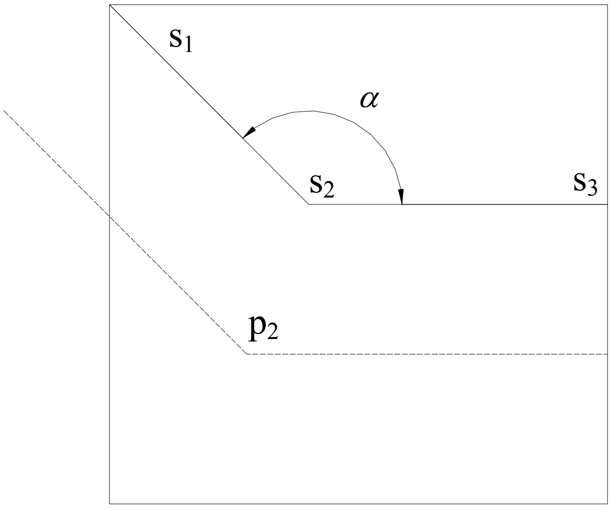


Fig.4-10a Offset Mode：Line → Line (90°≦α＜180°)

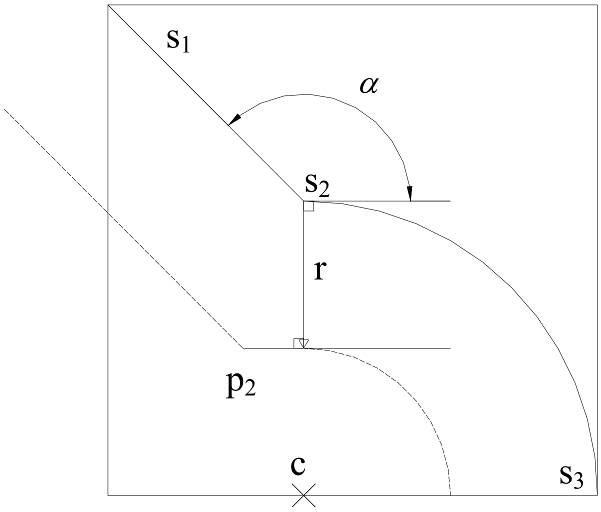


Fig.4-10b Offset Mode：Line → Arc (90°≦α＜180°)

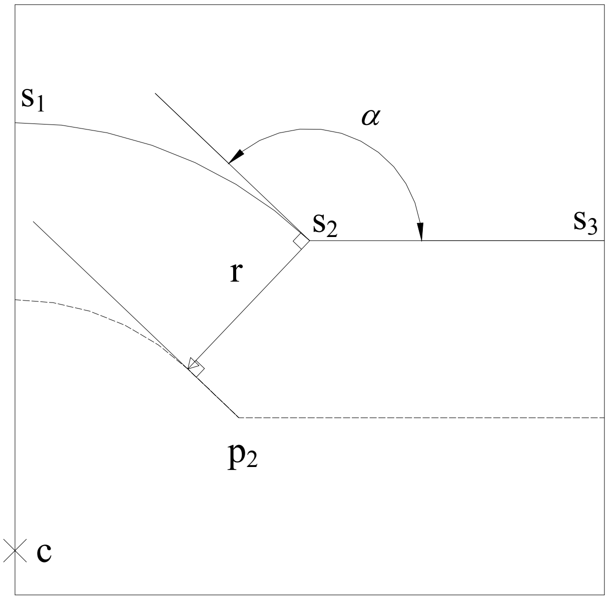


Fig.4-10c Offset Mode：Arc → Line (90°≦α＜180°)

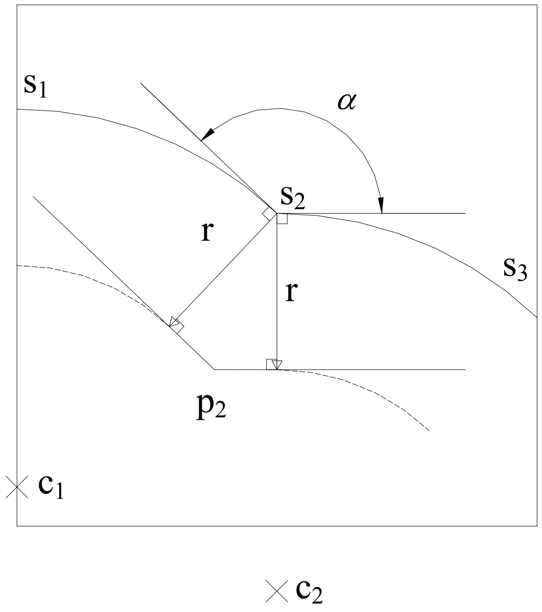


Fig.4-10d Offset Mode：Arc → Arc (90°≦α＜180°)

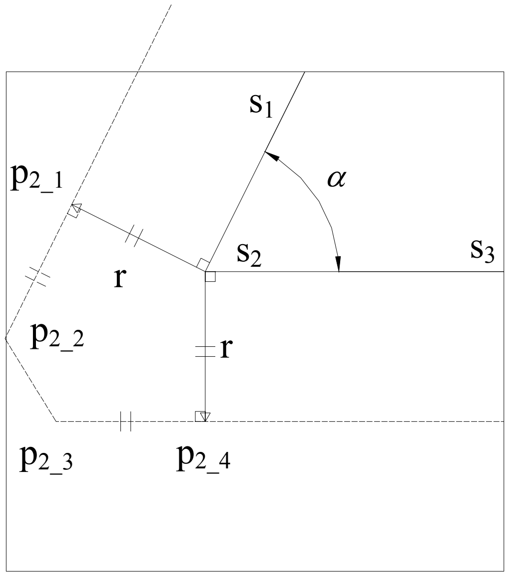


Fig.4-11a Offset Mode：Line → Line (α＜90°)

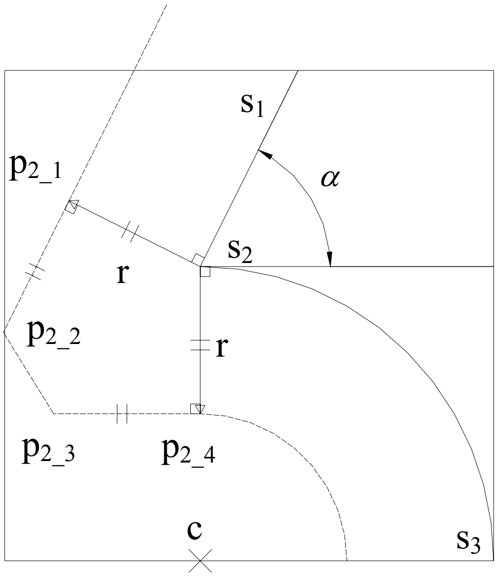


Fig.4-11b Offset Mode：Line → Arc ($\alpha < 90°$)

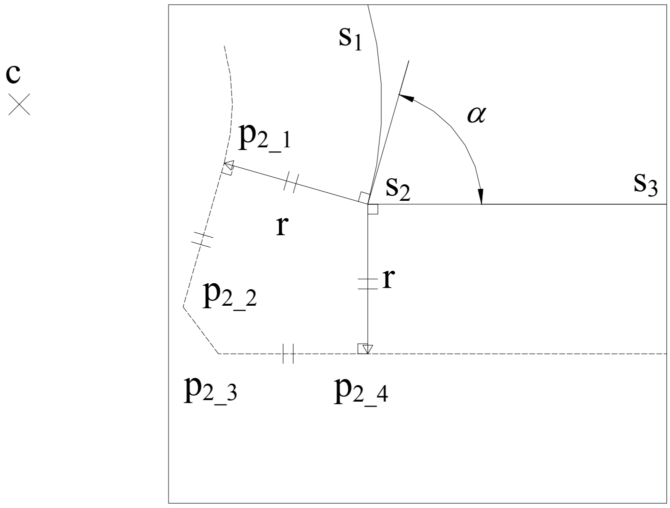


Fig.4-11c Offset Mode：Arc → Line ($\alpha < 90°$)

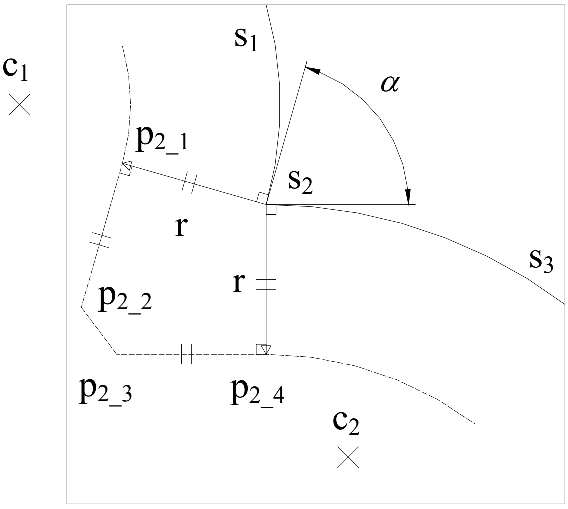


Fig.4-11d Offset Mode：Arc → Arc (α＜90°)

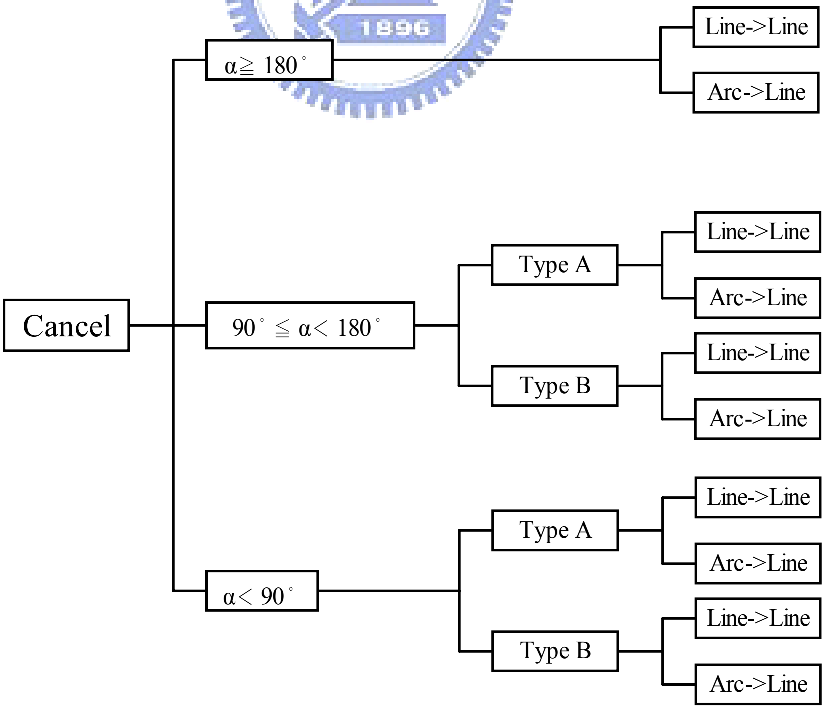


Fig.4-12 Cancel Mode

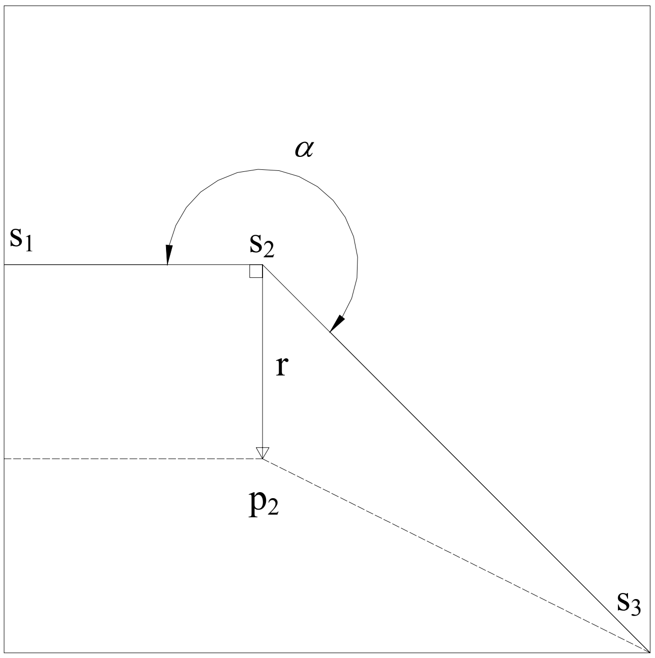


Fig.4-13a Cancel Mode：Line → Line ($\alpha \geqq 180^\circ$)

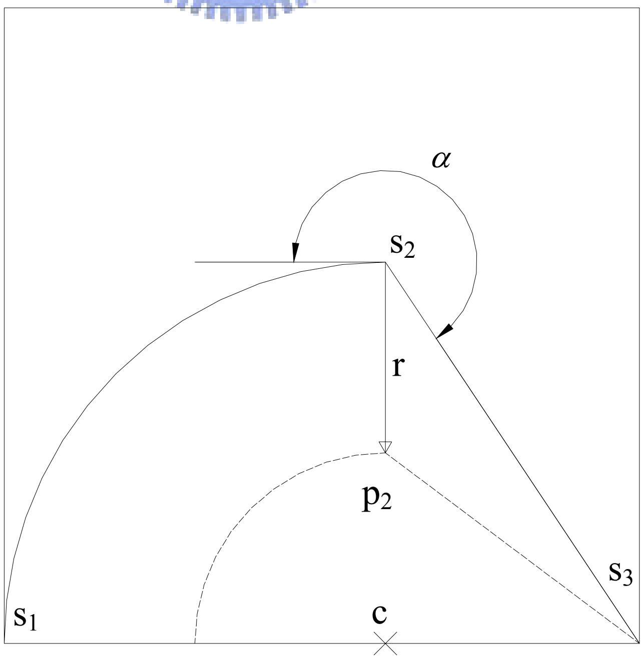


Fig.4-13b Cancel Mode：Arc → Line ($\alpha \geqq 180^\circ$)

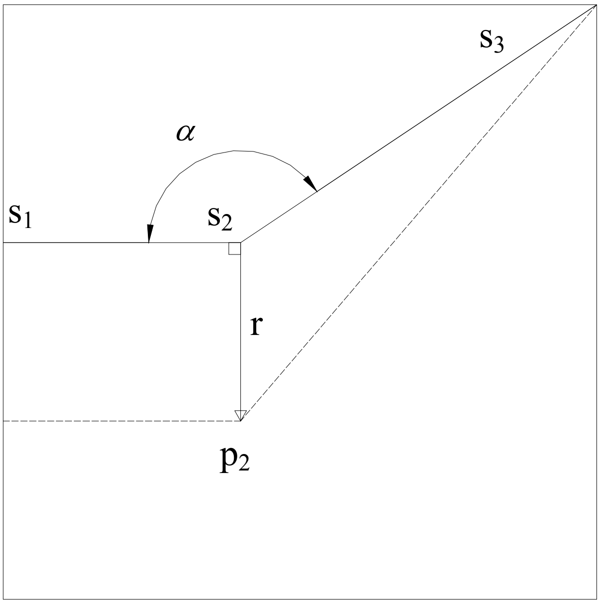


Fig.4-14a Cancel Mode：Line → Line (90°≦α＜180°-Type A)

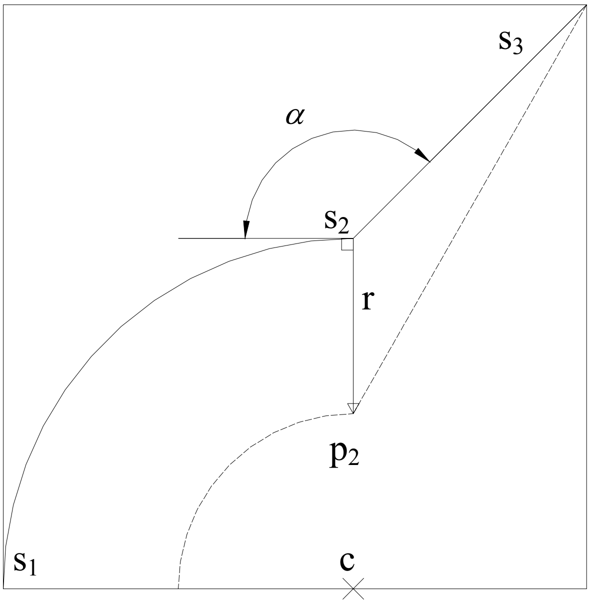


Fig.4-14b Cancel Mode：Arc → Line (90°≦α＜180°-Type A)

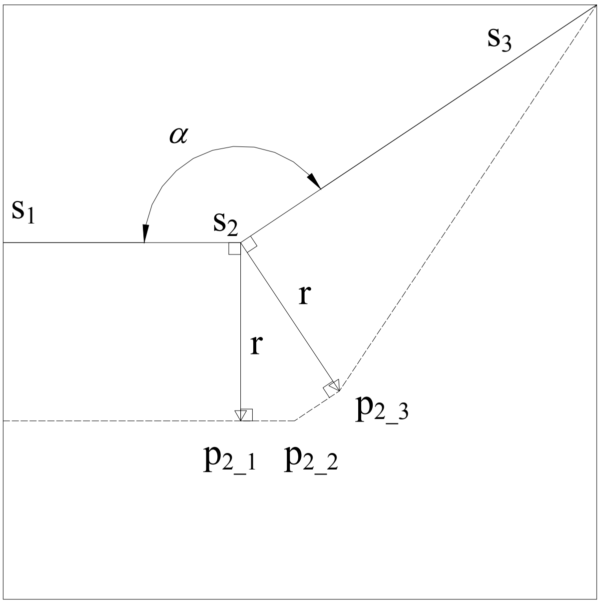


Fig.4-15a Cancel Mode：Line → Line ($90^\circ \leqq \alpha < 180^\circ$-Type B)

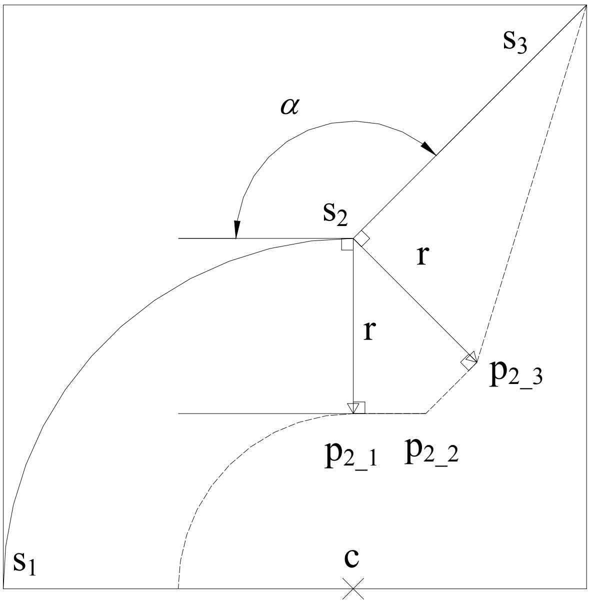


Fig.4-15b Cancel Mode：Arc → Line ($90^\circ \leqq \alpha < 180^\circ$-Type B)

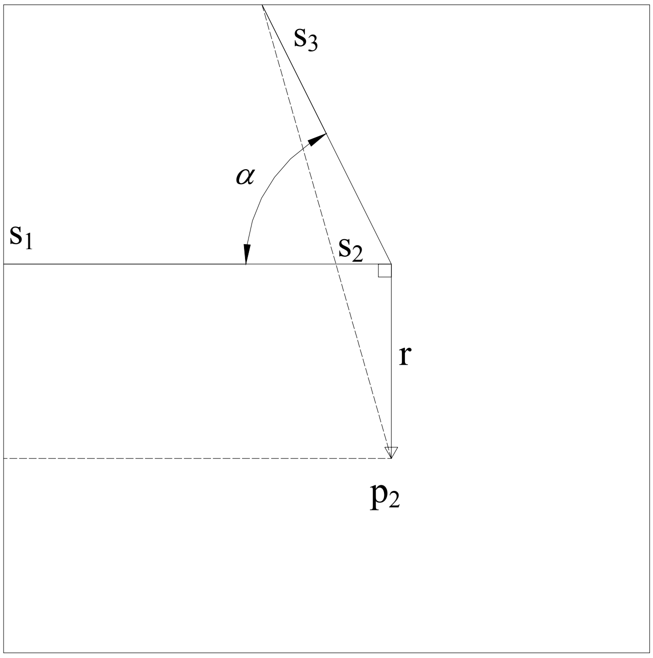


Fig.4-16a Cancel Mode：Line → Line (α<90°-Type A)

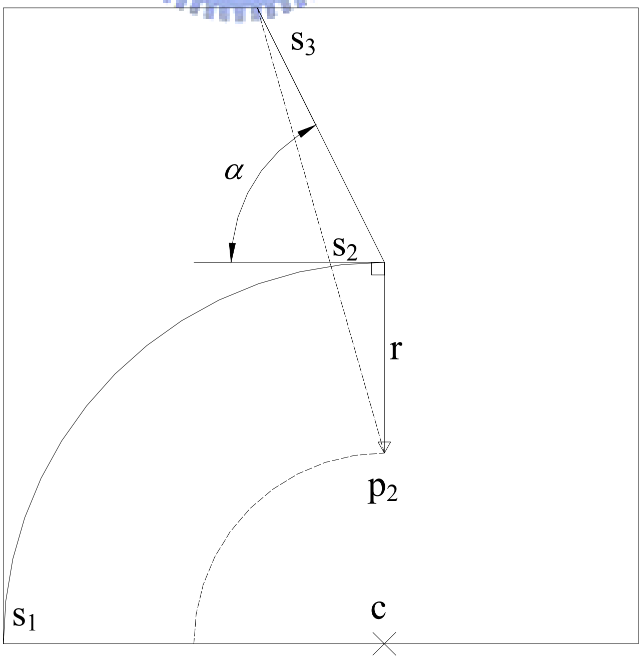


Fig.4-16b Cancel Mode：Arc → Line (α<90°-Type A)

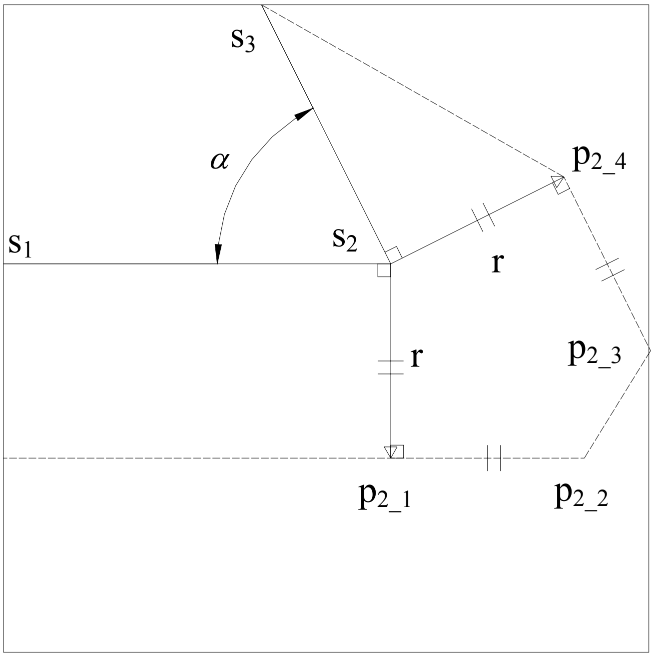


Fig.4-17a Cancel Mode：Arc → Line (α<90°-Type B)

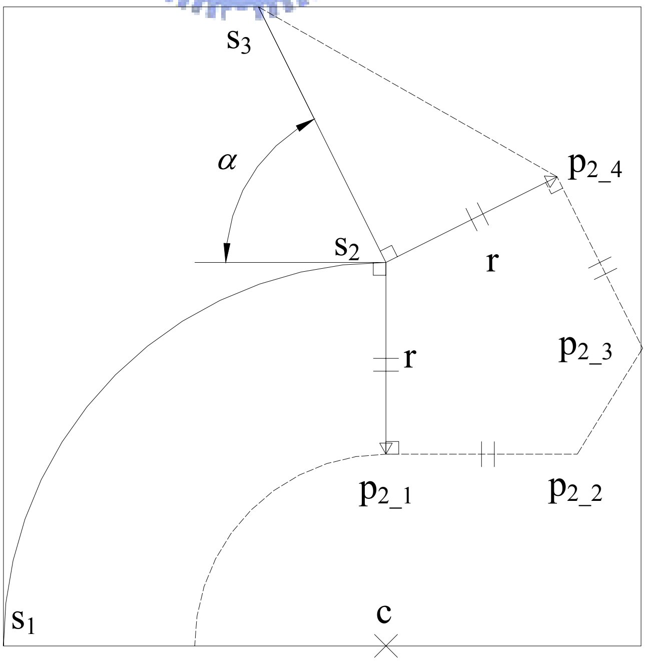


Fig.4-17b Cancel Mode：Line → Line (α<90°-Type B)

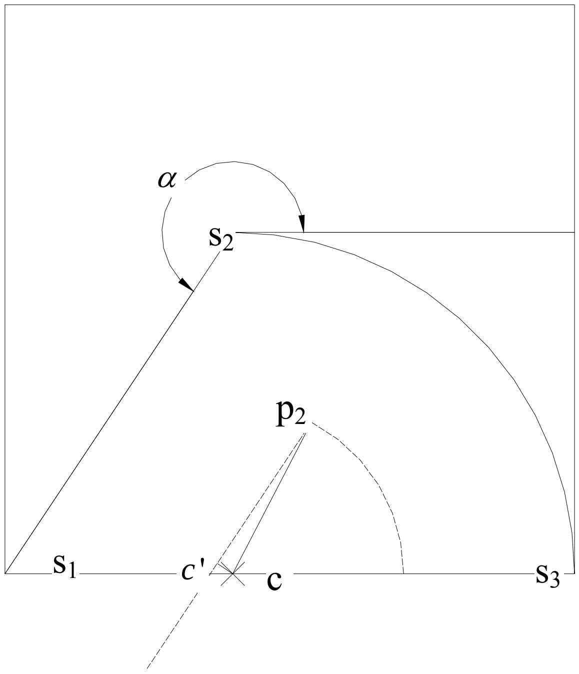


Fig.5-1 Offset Mode：Line → Arc (α≧180°)

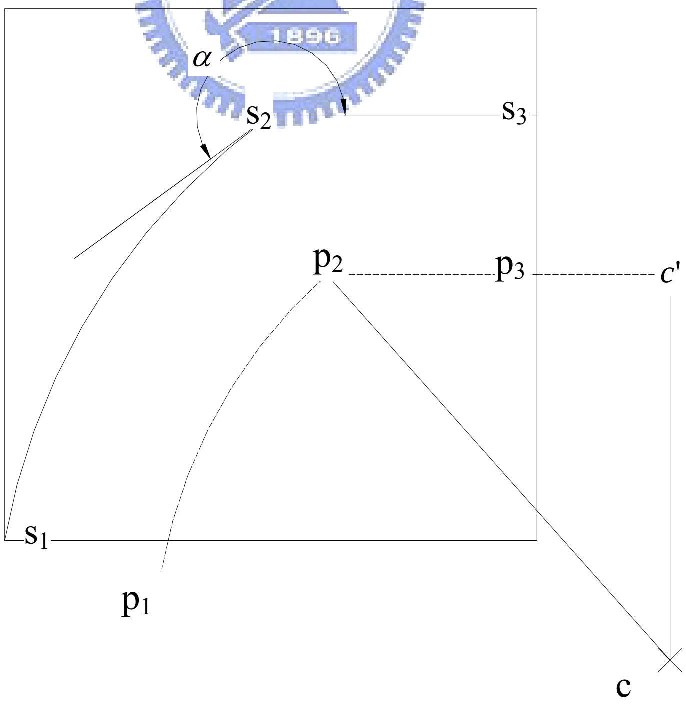


Fig.5-2 Offset Mode：Arc → Line (α≧180°)

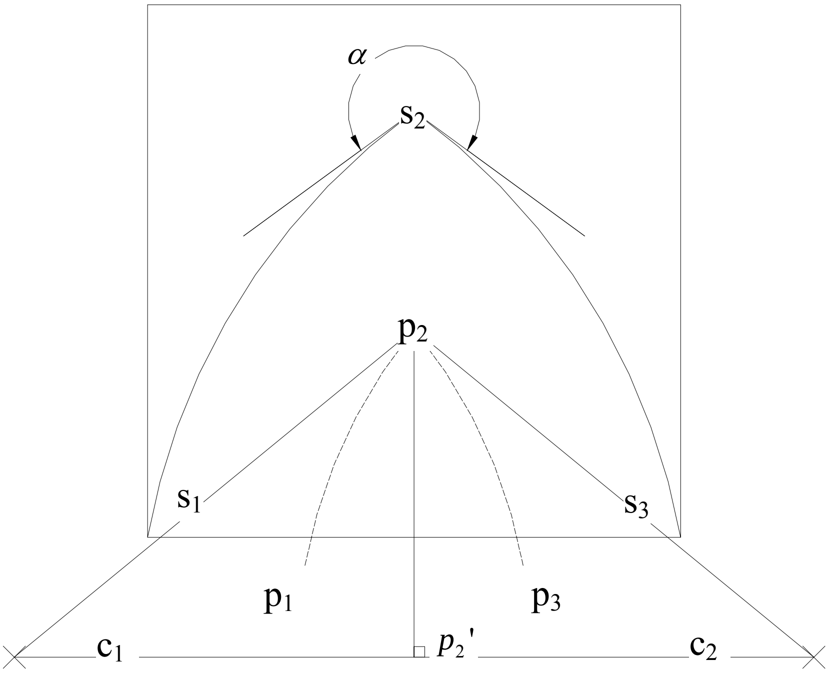


Fig.5-3 Offset Mode：Arc → Arc (α≧180°)

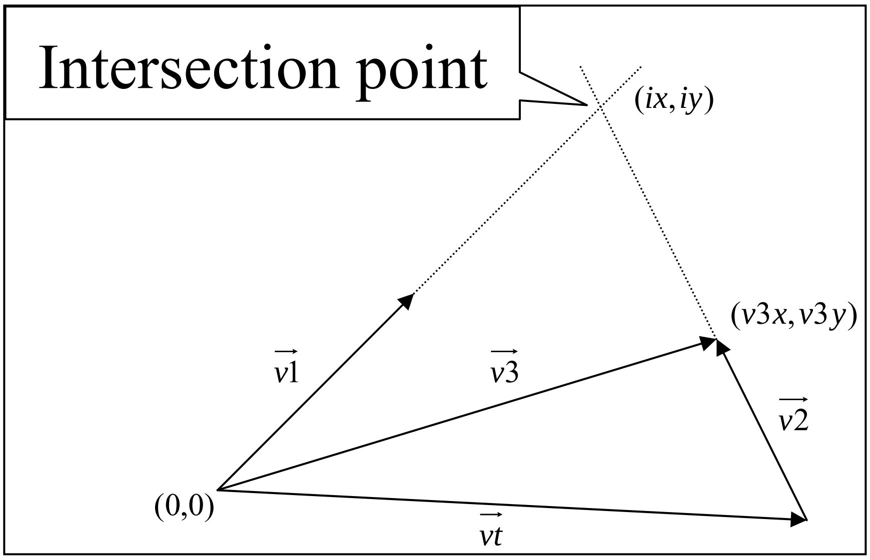


Fig.5-4a Interference Check：Intersection Point

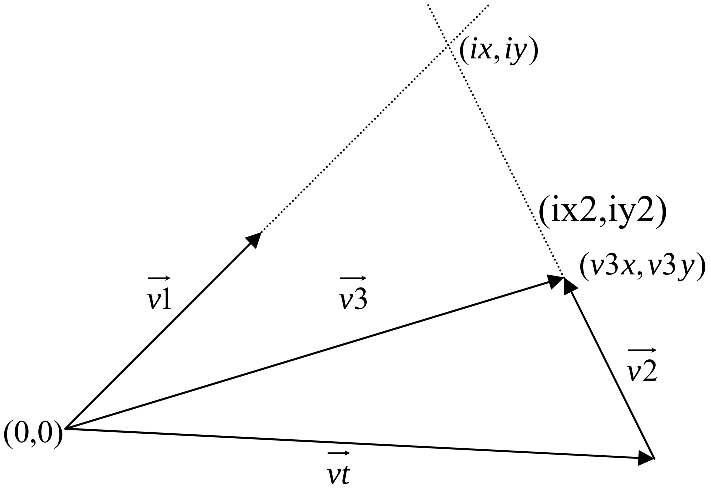


Fig.5-4b Interference Check：Vector Intersection

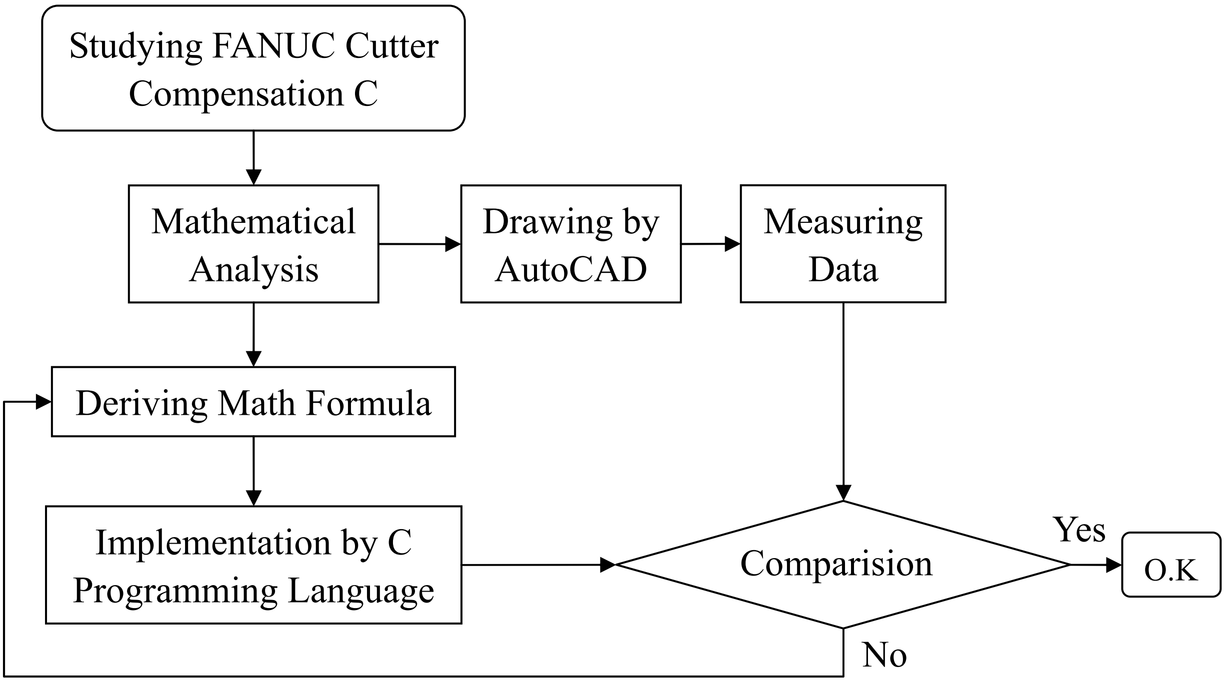


Fig.1a Thesis Flowchart

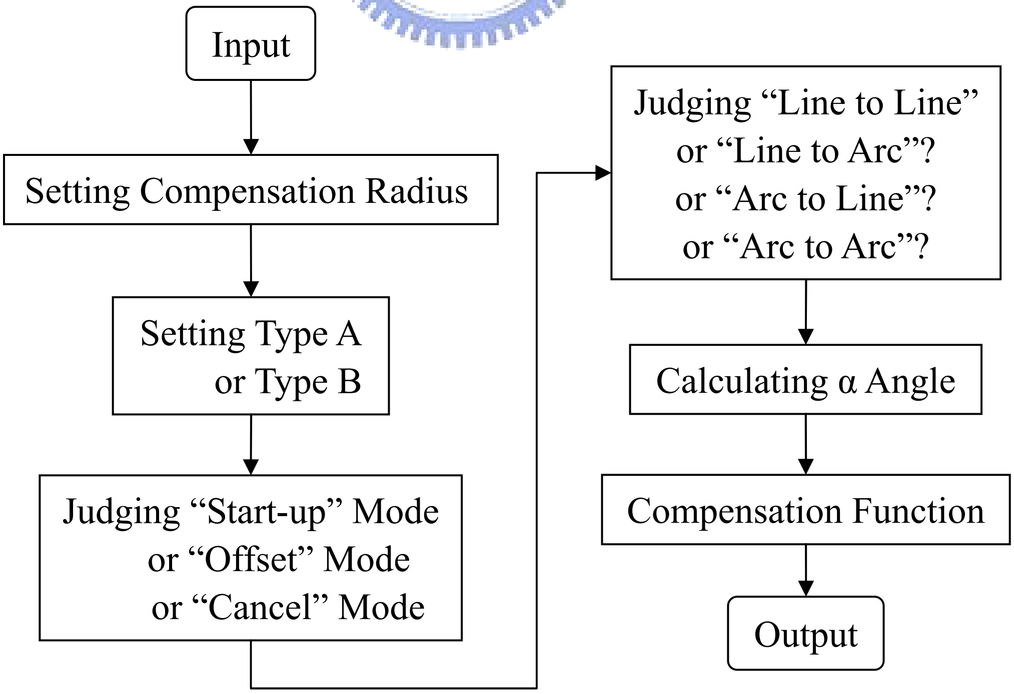
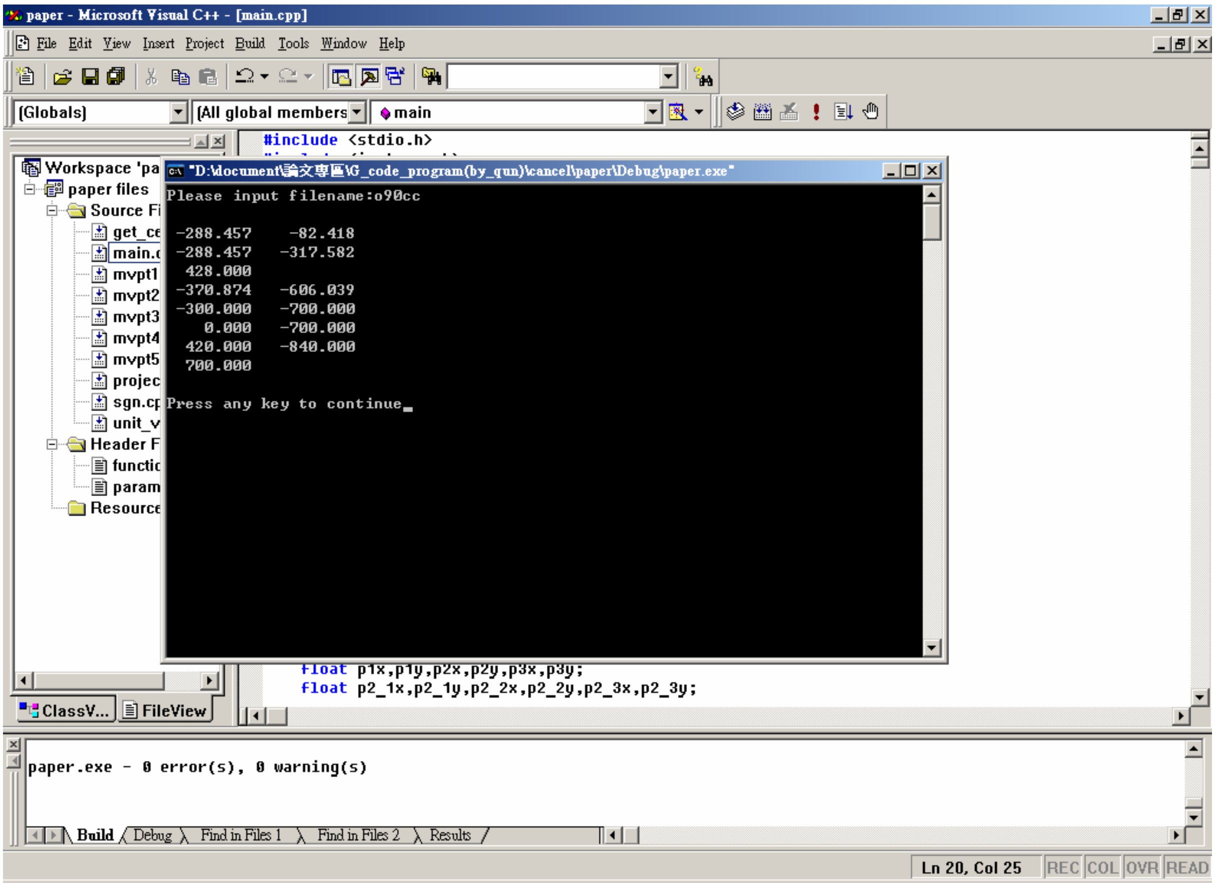


Fig.1b Programming Flowchart



```
#include <stdio.h>
```

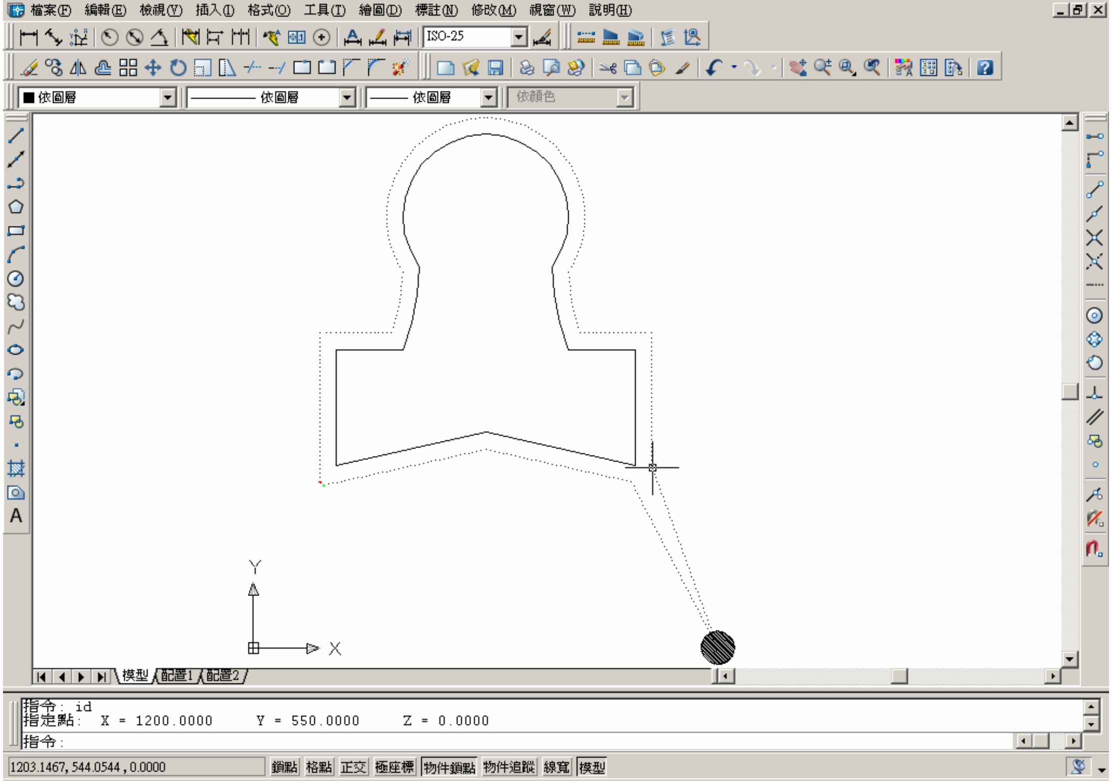
"D:\document\論文專區\G_code_program(by_qun)\cancel\paper\Debug\paper.exe"

```
Please input filename:o90cc

 -288.457    -82.418
 -288.457   -317.582
  428.000
 -370.874   -606.039
 -300.000   -700.000
    0.000   -700.000
  420.000   -840.000
  700.000

Press any key to continue_
```

```
    float p1x,p1y,p2x,p2y,p3x,p3y;
    float p2_1x,p2_1y,p2_2x,p2_2y,p2_3x,p2_3y;
```

```
paper.exe - 0 error(s), 0 warning(s)
```

Fig.2a Visual C++ Integrated Development Environment



```
指令: id
指定點:   X = 1200.0000     Y = 550.0000     Z = 0.0000
指令:
```

1203.1467, 544.0544 , 0.0000

Fig.2b AutoCAD Measuring Environment

Table.3-1a Start-up Mode (Fig.4-1a ~ Fig.4-5b)

Start-up Mode		
α≧180°	90°≦α＜180°	α＜90°
Line to Line	**Line to Line**	**Line to Line**
0 42	0 42	0 42
0 0	0 0	0 0
400.0 600.0	400.0 -400.0	-200.0 -400.0
1000.0 600.0	1000.0 -400.0	400.0 -400.0
Line to Arc	**Line to Arc**	**Line to Arc**
0 42	0 42	0 42
0 0	0 0	0 0
400.0 600.0	400.0 -400.0	-200.0 -400.0
2 1000.0	2 1000.0	2 400.0
0.0 600.0	-1000.0 600.0	-1000.0 600.0

Table.3-1b Offset Mode (Fig.4-6a ~ Fig.4-8d)

Offset Mode		
$\alpha \geqq 180^\circ$	$90^\circ \leqq \alpha < 180^\circ$	$\alpha < 90^\circ$
Line to Line 42 0 0 0 400.0 600.0 1000.0 600.0	**Line to Line** 42 0 0 0 400.0 -400.0 1000.0 -400.0	**Line to Line** 42 0 0 0 -200.0 -400.0 400.0 -400.0
Line to Arc 42 0 0 0 400.0 600.0 2 1000.0 0.0 600.0	**Line to Arc** 42 0 0 0 400.0 -400.0 2 1000.0 -1000.0 600.0	**Line to Arc** 42 0 0 0 -200.0 -400.0 2 400.0 -1000.0 600.0
Arc to Line 42 2 0 0 500.0 800.0 1270 0 1000.0 800.0	**Arc to Line** 42 2 0 0 500.0 -200.0 725.0 0 1000.0 -200.0	**Arc to Line** 42 2 0 0 0 -400.0 728 0 600.0 -400.0
Arc to Arc 42 2 0 0 500.0 800.0 1270.0 2 1000.0 0.0 1270.0	**Arc to Arc** 42 2 0 0 500.0 -200.0 725.0 2 1000.0 -400.0 725.0	**Arc to Arc** 42 2 0 0 0 -400.0 728.0 2 600.0 -600.0 1000.0

Table.3-1c Cancel Mode (Fig.4-9a ~ Fig.4-13b)

Cancel Mode		
α≧180°	90°≦α<180°	α<90°
Line to Line 42 0 0 0 400.0 0.0 0 40 1000.0 -600.0 **Arc to Line** 42 2 0 0 600.0 600.0 600.0 0 40 1000.0 0.0	**Line to Line** 42 0 0 0 400.0 0.0 0 40 1000.0 400.0 **Arc to Line** 42 2 0 0 600.0 600.0 600.0 0 40 1000.0 1000.0	**Line to Line** 42 0 0 0 600.0 0 0 40 400.0 400.0 **Arc to Line** 42 2 0 0 600.0 600.0 600.0 0 40 400.0 1000.0

Table.3-2a Start-up Mode (Precision：0.000) (Fig.4-1a ~ Fig.4-5b)

Start-up Mode		
α≧180°	90°≦α＜180°	α＜90°
Line to Line *X* *Y* 0.000 0.000 400.000 300.000 1000.000 300.000 **Line to Arc** *X* *Y* 0.000 0.000 400.000 300.000 700.000 0.000 *R* 300.000	● Type A **Line to Line** *X* *Y* 0.000 0.000 400.000 -700.000 1000.000 -700.000 **Line to Arc** *X* *Y* 0.000 0.000 400.000 -700.000 700.000 -1000.000 *R* 300.000	● Type A **Line to Line** *X* *Y* 0.000 0.000 -200.000 -700.000 400.000 -700.000 **Line to Arc** *X* *Y* 0.000 0.000 -200.000 -700.000 100.000 -1000.000 *R* 300.000
	● Type B **Line to Line** *X* *Y* 0.000 0.000 187.868 -612.132 275.736 -700.000 1000.000 -700.000 **Line to Arc** *X* *Y* 0.000 0.000 187.868 -612.132 275.736 -700.000 400.000 -700.000 700.000 -1000.000 *R* 300.000	● Type B **Line to Line** *X* *Y* 0.000 0.000 -468.328 -265.836 -602.492 -534.164 -500.000 -700.000 400.000 -700.000 **Line to Arc** *X* *Y* 0.000 0.000 -468.328 -265.836 -602.492 -534.164 -500.000 -700.000 -200.000 -700.000 100.000 -1000.000 *R* 300.000

Table.3-2b Offset Mode (Precision：0.000) (Fig.4-6a ~ Fig.4-8d)

Offset Mode					
α≧180°		90°≦α＜180°		α＜90°	
Line to Line		**Line to Line**		**Line to Line**	
X	*Y*	*X*	*Y*	*X*	*Y*
249.615	-166.410	-212.132	-212.132	-268.328	134.164
560.555	300.000	275.736	-700.000	-602.492	-534.164
1000.000	300.000	1000.000	-700.000	-500.000	-700.000
				400.000	-700.000
Line to Arc		**Line to Arc**		**Line to Arc**	
X	*Y*	*X*	*Y*	*X*	*Y*
249.615	-166.410	-212.132	-212.132	-268.328	134.164
538.103	266.322	275.736	-700.000	-602.492	-534.164
700.000	0.000	400.000	-700.000	-500.000	-700.000
R	300.000	700.000	-1000.000	-200.000	-700.000
		R	300.000	100.000	-1000.000
				R	300.000
Arc to Line		**Arc to Line**		**Arc to Line**	
X	*Y*	*X*	*Y*	*X*	*Y*
295.256	-53.138	0.000	-300.000	-288.457	-82.418
605.444	500.000	293.103	-417.241	-288.457	-317.582
R	970.000	*R*	425.000	*R*	428.000
X	*Y*	*X*	*Y*	*X*	*Y*
1000.000	500.000	380.000	-500.000	-370.874	-606.039
		1000.000	-500.000	-300.000	-700.000
				600.000	-700.000
Arc to Arc		**Arc to Arc**		**Arc to Arc**	
X	*Y*	*X*	*Y*	*X*	*Y*
295.256	-53.138	0.000	-300.000	-288.457	-82.418
500.000	390.291	293.103	-417.241	-288.457	-317.582
R	970.000	*R*	425.000	*R*	428.000
X	*Y*	*X*	*Y*	*X*	*Y*
704.744	-53.138	380.000	-500.000	-370.874	-606.039
R	970.000	500.000	-500.000	-300.000	-700.000
		793.103	-617.241	0.000	-700.000
		R	425.000	420.000	-840.000
				R	700.000

Table.3-2c Cancel Mode (Precision：0.000) (Fig.4-9a ~ Fig.4-13b)

Cancel Mode		
α≧180°	90°≦α＜180°	α＜90°
Line to Line *X* *Y* 0.000 -300.000 400.000 -300.000 1000.000 -600.000 **Arc to Line** *X* *Y* 300.000 0.000 600.000 300.000 *R* 300.000 *X* *Y* 1000.000 0.000	● Type A **Line to Line** *X* *Y* 0.000 -300.000 400.000 -300.000 1000.000 400.000 **Arc to Line** *X* *Y* 300.000 0.000 600.000 300.000 *R* 300.000 *X* *Y* 1000.000 1000.000	● Type A **Line to Line** *X* *Y* 0.000 -300.000 600.000 -300.000 400.000 400.000 **Arc to Line** *X* *Y* 300.000 0.000 600.000 300.000 *R* 300.000 *X* *Y* 400.000 1000.000
	● Type B **Line to Line** *X* *Y* 0.000 -300.000 490.833 -300.000 566.410 -249.615 1000.000 400.000 **Arc to Line** *X* *Y* 300.000 0.000 600.000 300.000 *R* 300.000 *X* *Y* 724.264 300.000 812.132 387.868 1000.000 1000.000	● Type B **Line to Line** *X* *Y* 0.000 -300.000 900.000 -300.000 1002.492 -134.164 868.328 134.164 400.000 400.000 **Arc to Line** *X* *Y* 300.000 0.000 600.000 300.000 *R* 300.000 *X* *Y* 900.000 300.000 1002.492 465.836 868.328 734.164 400.000 1000.000

Table.3-3a Start-up Mode (Precision：0.000) (Fig.4-1a ~ Fig.4-5b)

Start-up Mode		
α≧180°	90°≦α＜180°	α＜90°
Line to Line X = 0.000　Y = 0.000 X = 400.000　Y = 300.000 X = 1000.000　Y = 300.000 **Line to Arc** X = 0.0000　Y = 0.0000 X = 400.0000　Y = 300.0000 X = 700.0000　Y = 0.0000 R = 300.000	● Type A **Line to Line** X = 0.000　Y = 0.000 X = 400.000　Y = -700.000 X = 1000.000　Y = -700.000 **Line to Arc** X = 0.000　Y = 0.000 X = 400.000　Y = -700.000 X = 700.000　Y = -1000.000 R = 300.000	● Type A **Line to Line** X = 0.000　Y = 0.000 X = -200.000　Y = -700.000 X = 400.000　Y = -700.000 **Line to Arc** X = 0.000　Y = 0.000 X = -200.000　Y = -700.000 X = 100.000　Y = -1000.000 R = 300.000
	● Type B **Line to Line** X = 0.000　Y = 0.000 X = 187.868　Y = -612.132 X = 275.736　Y = -700.000 X = 1000.000　Y = -700.000 **Line to Arc** X = 0.000　Y = 0.000 X = 187.868　Y = -612.132 X = 275.736　Y = -700.000 X = 400.000　Y = -700.000 X = 700.000　Y = -1000.000 R = 300.000	● Type B **Line to Line** X = 0.000　Y = 0.000 X = -468.328　Y = -265.836 X = -602.492　Y = -534.164 X = -500.000　Y = -700.000 X = 400.000　Y = -700.000 **Line to Arc** X = 0.000　Y = 0.000 X = -468.328　Y = -265.836 X = -602.492　Y = -534.164 X = -500.000　Y = -700.000 X = -200.000　Y = -700.000 X = 100.000　Y = -1000.000 R = 300.000

Table.3-3b Offset Mode (Precision：0.000) (Fig.4-6a ~ Fig.4-8d)

Offset Mode		
α≧180°	90°≦α＜180°	α＜90°
Line to Line X = 249.615 Y = -166.410 X = 560.555 Y = 300.000 X = 1000.000 Y = 300.000	**Line to Line** X = -212.132 Y = -212.132 X = 275.736 Y = -700.000 X = 1000.000 Y = -700.000	**Line to Line** X = -268.328 Y = 134.164 X = -602.492 Y = -534.164 X = 500.000 Y = -700.000 X = 400.000 Y = -700.000
Line to Arc X = 249.615 Y = -166.410 X = 538.103 Y = 266.322 X = 700.000 Y = 0.000 R = 300.000	**Line to Arc** X = -212.132 Y = -212.132 X = 275.736 Y = -700.000 X = 400.000 Y = -700.000 X = 700.000 Y = -1000.000 R = 300.000	**Line to Arc** X = -268.328 Y = 134.164 X = -602.492 Y = -534.164 X = -500.000 Y = -700.000 X = -200.000 Y = -700.000 X = 100.000 Y = -1000.000 R = 300.000
Arc to Line X = 295.256 Y = -53.138 X = 605.444 Y = 500.000 R = 970.000 X = 1000.000 Y = 500.000	**Arc to Line** X = 0.000 Y = -300.000 X = 293.103 Y = -417.241 R = 425.000 X = 380.000 Y = -500.000 X = 1000.000 Y = -500.000	**Arc to Line** X = -288.457 Y = -82.418 X = -288.457 Y = -317.582 R = 428.000 X-370.874 Y = -606.039 X = -300.000 Y = -700.000 X = 600.000 Y = -700.000
Arc to Arc X = 295.256 Y = -53.138 X = 500.000 Y = 390.291 R = 970.000 X = 704.744 Y = -53.138 R = 970.000	**Arc to Arc** X = 0.000 Y = -300.000 X = 293.103 Y = -417.241 R = 425.000 X = 380.000 Y = -500.000 X = 500.000 Y = -500.000 X = 793.103 Y = -617.241 R = 425.000	**Arc to Arc** X = -288.457 Y = -82.418 X = -288.457 Y = -317.582 R = 428.000 X = -370.874 Y = -606.039 X = -300.000 Y = -700.000 X = 0.000 Y = -700.000 X = 420.000 Y = -840.000 R = 700.000

Table.3-3c Cancel Mode (Precision：0.000) (Fig.4-9a ~ Fig.4-13b)

<table>
<tr><th colspan="3"><i>Cancel Mode</i></th></tr>
<tr><th>α≧180°</th><th>90°≦α＜180°</th><th>α＜90°</th></tr>
<tr>
<td rowspan="2">Line to Line
X = 0.000 Y = -300.000
X = 400.000 Y = -300.000
X = 1000.000 Y = -600.000
Arc to Line
X = 300.000 Y = 0.000
X = 600.000 Y = 300.000
R = 300.000
X = 1000.000 Y = 0.000</td>
<td>● Type A
Line to Line
X = 0.000 Y = -300.000
X = 400.000 Y = -300.000
X = 1000.000 Y = 400.000
Arc to Line
X = 300.000 Y = 0.000
X = 600.000 Y = 300.000
R = 300.000
X = 1000.000 Y = 1000.000</td>
<td>● Type A
Line to Line
X = 0.000 Y = -300.000
X = 600.000 Y = -300.000
X = 400.000 Y = 400.000
Arc to Line
X = 300.000 Y = 0.000
X = 600.000 Y = 300.000
R = 300.000
X = 400.000 Y = 1000.000</td>
</tr>
<tr>
<td>● Type B
Line to Line
X = 0.000 Y = -300.000
X = 490.833 Y = -300.000
X = 566.410 Y = -249.615
X = 1000.000 Y = 400.000
Arc to Line
X = 300.000 Y = 0.000
X = 600.000 Y = 300.000
R = 300.000
X = 724.264 Y = 300.000
X = 812.132 Y = 387.868
X = 1000.000 Y = 1000.000</td>
<td>● Type B
Line to Line
X = 0.000 Y = -300.000
X = 900.000 Y = -300.000
X = 1002.492 Y = -134.164
X = 868.328 Y = 134.164
X = 400.000 Y = 400.000
Arc to Line
X = 300.000 Y = 0.000
X = 600.000 Y = 300.000
R = 300.000
X = 900.000 Y = 300.000
X = 1002.492 Y = 465.836
X = 868.328 Y = 734.164
X = 400.000 Y = 1000.000</td>
</tr>
</table>