NC プログラム　上級編

はじめに

このレポートは、NC プログラムの基本的な作り方、使い方を理解していることを前提として作られていますので、基本的な説明は省略されています。基本的な内容に関しては、Web サイト「NC プログラム基礎知識」を御覧ください。

本書内で作成したプログラムのほとんどは、テキストファイルとして添付してありますので、すぐにお使いいただくことができます。また、各プログラムは変数の初期設定をすべてローカル変数にしているので、これを引数にすることでマクロとしても使用できるように設計してあります（マクロにする場合は必要のない箇所を削除し、最後の M30 を M99 に変更するだけです）。

レポートの不備などがございましたら、下記メールアドレスか Web サイトからお願い致します。ご意見・ご感想などもお聞かせ頂けるとありがたいです。

メール：info2@s-projects.net
サイト：https://nc-program.s-projects.net/

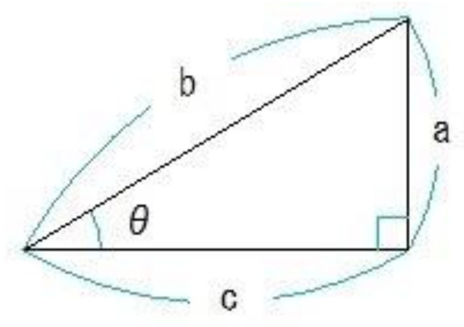
目次

第１章　数学の復習

本書では、テーパや R といった立体的な加工を扱いますので、どうしても三角関数や三平方の定理などの数学の知識が必要になります。そこで本書で扱う数学の知識を最初に掲載します。

三角関数

三角関数の基本のかたちは以下のようになります。図の辺 a、b、c は長さ、θ（シータ）は角度です。



- **$\sin\theta = a / b$**
- **$\cos\theta = c / b$**
- **$\tan\theta = a / c$**

一辺の長さと角度から辺の長さを求める

まず、上の式を見ても分かる通り、三角関数 sin（サイン）、cos（コサイン）、tan（タンジェント）とは、角度から二辺の比率を算出する関数です。どの二辺の比率を求めるのか、によってこの三つの関数を使い分けます。

ここでは少なくとも、角度とどれか一辺の長さが分かっている必要があります。

- 辺 a を求める
  - 辺 b と角度 θ が分かっている　**$a = b \times \sin\theta$**
  - 辺 c と角度 θ が分かっている　**$a = c \times \tan\theta$**

- 辺 b を求める
  - 辺 a と角度 θ が分かっている　**$b = a / \sin\theta$**
  - 辺 c と角度 θ が分かっている　**$b = c / \cos\theta$**

- 辺 c を求める
  - 辺 a と角度 θ が分かっている　**$c = a / \tan\theta$**
  - 辺 b と角度 θ が分かっている　**$c = b \times \cos\theta$**

二辺の長さから角度を求める

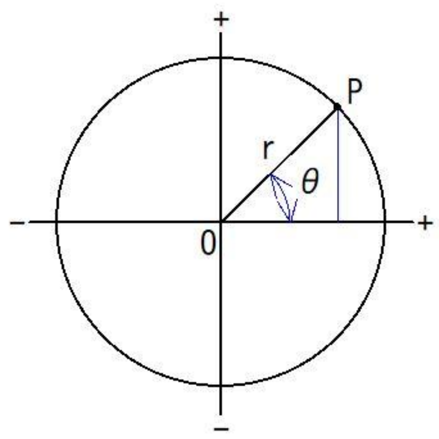
本書ではでてきませんが、二辺の長さから角度を求めるためには、$\sin^{-1}$（Asin・アークサイン）、$\cos^{-1}$（Acos・アークコサイン）、$\tan^{-1}$（Atan・アークタンジェント）を使用します。sin、cos、tan の逆関数で、二辺の比率から角度を算出するものです。

ここでは少なくとも、二辺の長さが分かっている必要があります。

- **$\theta = \sin^{-1}(a / b)$**
- **$\theta = \cos^{-1}(c / b)$**
- **$\theta = \tan^{-1}(a / c)$**

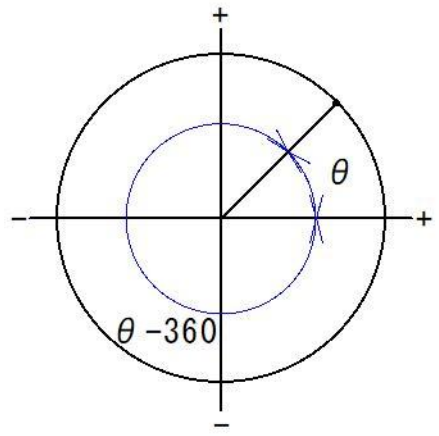
円と三角関数

本書では円周上の点の座標を求める式が頻繁に出てくるので、円と三角関数について基本的なことを復習します。



右図のように、点 O を中心とする円周上に任意の点 P を置いた場合の座標を求めます。このとき、横軸の右側がプラス、左側がマイナスとなり、縦軸は上側がプラス、下側がマイナスとなります。本書の中でも、円周上の座標を求めるための平面図はすべて、この向きになっています。

点 O から点 P までの距離を r とし、横軸のプラス側からの角度を θ とします。このときの点 P の横軸の座標は「$r \times \cos\theta$」、縦軸の座標は「$r \times \sin\theta$」となります。角度の方向は、反時計回りがプラス、時計回りがマイナスの角度となります。

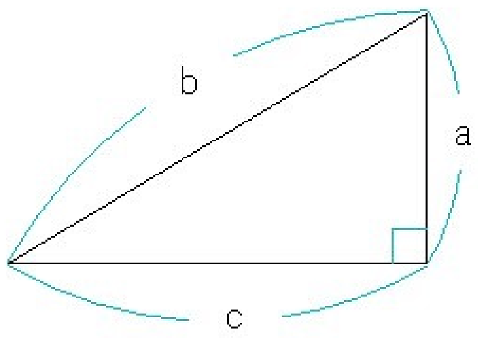


角度は「$\theta - 360$」としても計算結果は同じとなります。つまり、「$\theta = 45$」の場合は「$45 - 360 = -315$」となるので、「$r \times \cos 45°$」と「$r \times \cos -315°$」は同じです。同様に「$r \times \sin 45°$」と「$r \times \sin -315°$」も結果は同じです。これは θ が 90° を超えた場合にも成り立ちます。

三平方の定理

三平方の定理は、ピタゴラスの定理とも呼ばれるものです。これは、直角三角形の二辺の長さが分かっていれば、残りの一辺の長さを求めることができるというものです。

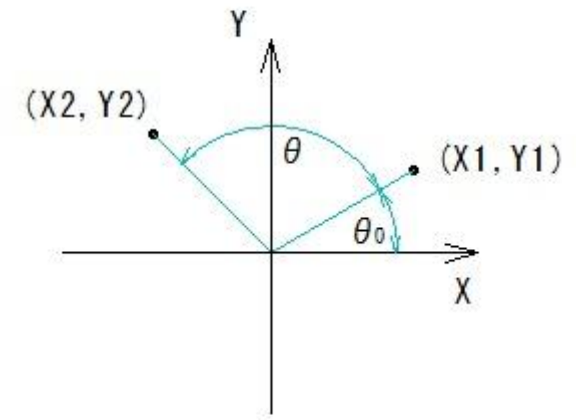
・辺ｂの長さを求める　$\mathbf{b = \sqrt{(a2 + c2)}}$
・辺ａの長さを求める　$\mathbf{a = \sqrt{(b2 - c2)}}$
・辺ｃの長さを求める　$\mathbf{c = \sqrt{(b2 - a2)}}$



斜辺（一番長い辺）の長さを求める場合は、残り二辺の自乗したものをたす。他の二辺を求める場合は、斜辺の自乗したものを残りの一辺の自乗したもので引けば良い、と言うことです。非常にシンプルな式ですので、覚えておきましょう。

座標回転

まずZ軸周りの回転について考えていきます。回転させる前の現在の座標を（X1,Y1）とし、角度θで回転させた後の座標を（X2,Y2）とします。回転中心（0,0）から（X1,Y1）までの直線距離をr、その角度はθ_0とします。



このときのX2の座標を求める式はこのようになります。

$$\mathbf{X2 = r\cos(\theta+\theta_0)}$$

しかし、これでは回転させるたびにrとθ_0を求めなければならないので、式を変換していきます。

$$\begin{aligned}\mathbf{X2} &= \mathbf{r\cos(\theta+\theta_0)}\\ &= \mathbf{r\cos\theta\ \cos\theta_0 - r\sin\theta\ \sin\theta_0}\\ &= \mathbf{r\cos\theta_0\cos\theta\ - r\sin\theta_0\sin\theta}\\ &= \mathbf{X1\cos\theta\ - Y1\sin\theta}\end{aligned}$$

２番目の式への変換は加法定理を用います。

$$\cos(\alpha+\beta) = \cos\alpha\ \cos\beta\ - \sin\alpha\ \sin\beta$$

３番目の式で$r\cos\theta_0$というのが出てきますが、これはX1の座標を求める式と同じですのでX1と置き換えることができます。同様に$r\sin\theta_0$もY1と置くことができます。

Y座標も同様に加法定理を用います。

$$\sin(\alpha+\beta) = \sin\alpha\ \cos\beta\ + \cos\alpha\ \sin\beta$$

$$\begin{aligned}\mathbf{Y2} &= \mathbf{r\sin(\theta+\theta_0)}\\ &= \mathbf{r\sin\theta\ \cos\theta_0 + r\cos\theta\ \sin\theta_0}\\ &= \mathbf{r\cos\theta_0\sin\theta\ + r\sin\theta_0\cos\theta}\\ &= \mathbf{X1\sin\theta\ + Y1\cos\theta}\end{aligned}$$

Z 軸の座標は変わらないので、Z2 = Z1 となります。

X 軸周りの回転も Y 軸周りの回転も、同じプロセスで式を求めることができ、まとめると下記のようになります。

Z 軸周りの回転

- **$X2 = X1 \cos\theta - Y1 \sin\theta$**
- **$Y2 = X1 \sin\theta + Y1 \cos\theta$**
- **$Z2 = Z1$**

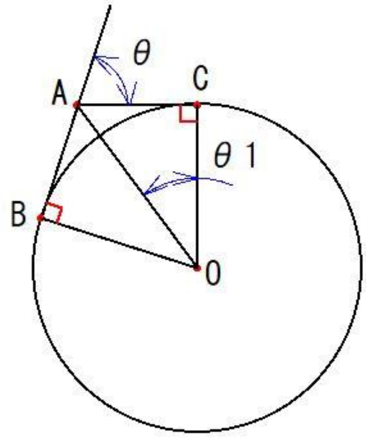
X 軸周りの回転

- **$X2 = X1$**
- **$Y2 = Y1 \cos\theta - Z1 \sin\theta$**
- **$Z2 = Y1 \sin\theta + Z1 \cos\theta$**

Y 軸周りの回転

- **$X2 = Z1 \sin\theta + X1 \cos\theta$**
- **$Y2 = Y1$**
- **$Z2 = Z1 \cos\theta - X1 \sin\theta$**

円と２つの接線



右図のように、まず点 O を中心とする円を描き、その円の外側に点 A を置きます。点 A を通る円との接線を 2 本引き、その接点を点 B、点 C とします。２つの接点と点 O を結ぶ線を引くと四角形 OBAC ができます。この四角形の対角線 OA を引いたとき、∠AOC（θ1）を求めます。今回はθの角度がわかっている場合を考えます。

四角形 OBAC について、∠OBA と∠OCA は接線に垂直な線となるので共に 90°、四角形の内角の和は 360°なので、残りの 2 つの角度（∠BAC と∠BOC）の和は 180°であることがわかります。そして∠BAC は「180 - θ」となるので、「∠BAC＋ ∠BOC＝180」より「∠BOC＝ θ」となります。

次に△OBA と△OCA について、OA は共通の線、OB と OC は共に円の半径となるので「OB＝OC」が成り立ち、2 つの直角三角形の合同条件である「斜辺と他の 1 辺がそれぞれ等しい」に当てはまるので、この 2 つの三角形は合同であるといえます。従って「∠AOB＝∠AOC」なので「∠BOC＝ θ」より「∠AOC＝ θ ／2」となり「θ1＝ θ ／2」となります。

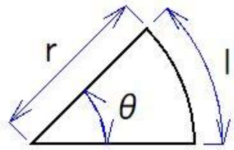
因みに AC の長さは「OC×TANθ1」なので「AC＝OC×TAN(θ ／2)」となります。

本書では、テーパ加工に関連するものにこの考え方が出てきます。

弧度法

一般的に角の大きさは、一回転を 360 等分したものを単位とする度数法が用いられますが、数学の世界では計算するのに便利な弧度法がよく用いられます。

弧度法では、右図のように半径を r、弧の長さを l とした場合の角の大きさ θ は、「$\theta = l / r$」として定義されます。このときの角の単位はラジアンと呼ばれます。角度が 180° となる半円の周の長さは「$l = \pi r$」（円周率×半径）となるので「180° ＝ π ラジアン」となります。従って、1 ラジアンは「180° / π」となります。



NC プログラムでは三角関数を使うときに弧度法は使えないので、度数法に変換する必要があります。本書では弧の長さから角度を算出する式が出てきますが、ラジアン（l / r）に「$180 / \pi$」をかければ度数法になりますので、角度 θ は「$(180 \times l) / (\pi r)$」となります。

弧度法は高校数学の教科書にも出てきますが、日常生活で使うことがないので、おそらく覚えていない人がほとんどだと思います。なので、本書ではラジアンという言葉を使わないために以下のように少し式を変形していますが、内容は全く同じです。

角度 $\theta = 360 / (2\pi r / l)$

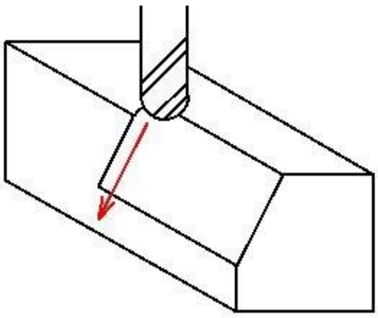
「$2\pi r / l$」は「$2\pi r$」で円周の長さを出して、それを弧の長さ l で割ることで、l が何個で円ができるかを示しており、その個数で 360° を割ることで、1 個の l が何度になるのかを求めています。

第２章　勾配・テーパ加工

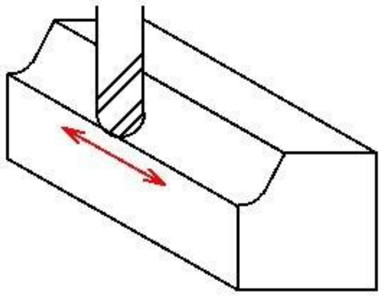
テーパ加工は専用工具を使えば簡単に加工できますが、テーパの種類ごとに工具をそろえていくと結構な数量になってしまうこともあります。ここではラジアスエンドミルやボールエンドミルなどの汎用的な工具を使い、プログラムによってテーパ形状を作る方法を紹介します。

なお、テーパ（taper）という用語は、一般的には円柱や角柱などの全周に入るもの、あるいは一方向から見たときに左右対称に入るものを指しますが、本書では一辺のみに入る勾配（slope）にもテーパという用語を使用します。

汎用工具を使う方法は、大きく分けると下記の２種類があります。



(1)

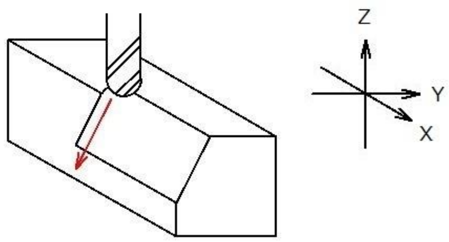


(2)

説明を簡単にするために、まずは図のように直線にテーパ形状を入れる方法を考えていきます。その後で円のテーパ加工に発展させます。

直線のテーパ加工（１）

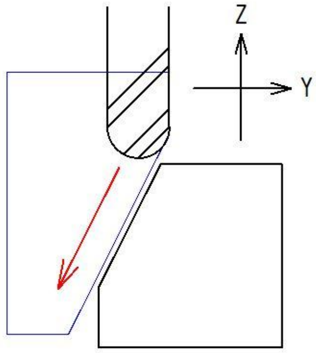
図の赤い矢印のようにテーパ角度にそって工具を移動させる方法で、X 軸を徐々にずらし、この軌道を繰り返すことでテーパ形状を作っていきます。



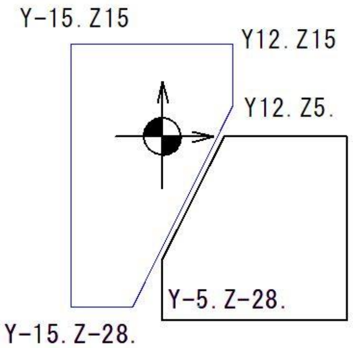
この工具軌道のテーパ加工は、工具径補正を使うと簡単に実現できますが、ちょっとした問題もあります。ここではまず、工具径補正を使った方法を紹介し、その後汎用性の高いプログラムの作り方を紹介します。

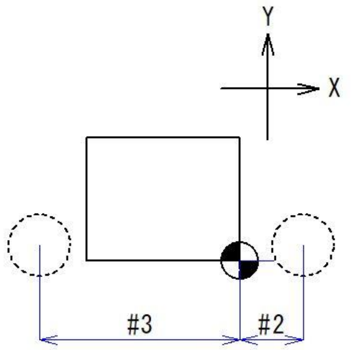
工具径補正を使う方法

右の図のように、YZ 平面（G19）設定で工具径補正を使うと簡単に実現できます。補正方向は X 軸のプラス側から見たものになるので、赤い矢印の方向に進む場合の工具径補正は G42(右側補正）で、逆に回る場合は G41（左側補正）になります。ここでは右図と同じ G42 でプログラムを作ります。



では、実際に右の図の座標を用いてプログラムを作ってみます。ワークの長さ合わせて X 軸の移動量を変更できるように、X 軸の開始座標と終了座標をそれぞれ#2 と#3 の変数にします。工具径補正値はあらかじめ「D1」に入れておくものとします（プログラム内でシステム変数を使って入力しても問題ありません)。



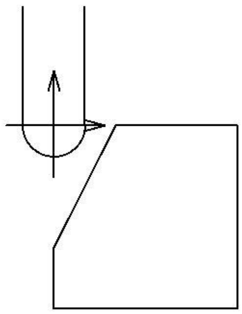



```
%
O100
(D1=5.0)
#2 = 5. (X START)
#3 = -20. (X END)
#9 = 1000 (F)
#17 = 0.1 (X PITCH)
G19
G90 G00 G54 X#2 Y-20.
Z15.
M3 S2000
M8
G01 G42 Y-15. D1 F3000
#100=#2
#30 = 0 (COUNT)

N100(START)
WHILE [#30 LE 1] DO1
WHILE [#100 GE #3] DO2
G01 X#100 F3000
Y12.
Z5. F#9
Y-5. Z-28.
Y-15. F3000
Z15.
#100 = #100 - #17
END2
```

```
#30 = #30 + 1
#100 = #3
END1

G00Z100.
G40 X0 Y0 Z150.
M9
M5
G91 G28 Z0
G28 Y0
M02
%
```



この方法は工具 R の中心を加工原点にするか、あるいは R の大きさ分の座標値をずらす必要があります。また、Y 軸方向の加工原点も同様に R の中心となり、ボールエンドミルの場合は工具の中心となりますが、ラジアスエンドミルの場合は工具の中心ではなく工具 R の中心となります。

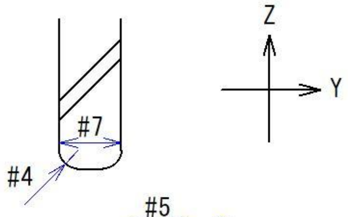
加工原点に気をつければ問題なく加工はできますが、汎用性を高めるために座標値を変数に変えると、テーパの角度によっては工具が抜けきらなかったり、それを防ぐために逃げ量を多くとることによって無駄な移動が増えてしまうこともあります。もちろん、テーパ角度によって逃げ量を変えることも可能ですが、工具径補正量を考慮しなければならないため計算がかなり複雑になります。

もう一つの問題点として、この方法は YZ 平面か ZX 平面でしか使えないということです。

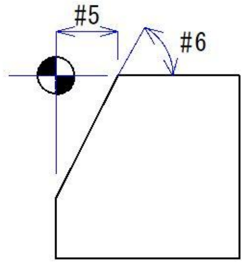
そこでこれらの問題を解決し、円形状のテーパ加工まで発展させるために、工具径補正を使用しない方法を次に紹介します。

工具径補正を使わない方法

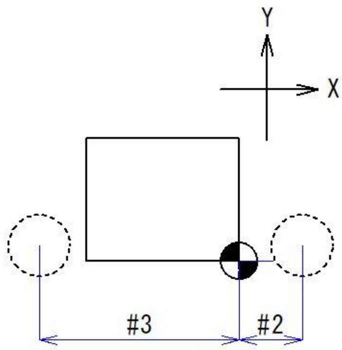
まず、必要になる変数を書き出します。右の YZ 平面の図のように、テーパ形状の寸法は、今回は Y 軸方向のテーパ幅（#5）と角度（#6）にします。そして工具直径（#7)、工具 R（#4)。



右下の XY 平面の図は、どこからどこまでテーパ加工を行うかという X 軸方向の開始座標（#2）と終了座標（#3）です。残るは加工速度（#9）と X 方向のピッチ（#17)。そして図には入れませんでしたが Z 軸の逃げ位置（#11）と、加工原点が別の位置にあった場合を考えて、Y 軸方向の位置（#1）も変数にしておきます。



また、今回はプログラムが複雑にならないように、X 軸の開始座標から終了座標までは、X 方向のピッチ分ずつ必ずマイナス方向に進んでいく加工にします。つまり「#2 > #3」となります。

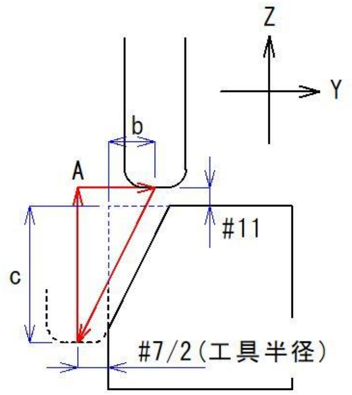


まとめると以下のようになります。数値は適当に入れました。因みに、1 ブロック目の(Y POS)の「POS」は「position（ポジション)」の略です。

```
#1 = 0 (Y POS)
#2 = 10. (X START)
#3 = -20. (X END)
#7 = 10. (TOOL CYOKKEI)
#11 = 1.0 (Z NIGE)
#4 = 3. (TOOL R)
#5 = 5. (TAPER Y HABA)
#6 = 60. (TAPER KAKUDO)
#9 = 1000 (SPEED)
#17 = 0.1 (PITCH)
```

実際にプログラムを作っていくときは、変数の前に工具軌道をイメージしてから行いますが、工具軌道は「工具径補正を使う方法」とほぼ同じなので省略しました。ただし、工具径補正を使うよりも単純化できます。

右の図の赤い線が工具の中心軌道となるので、3 点の座標だけを考えれば良いことになります。工具がワークの上の位置にあるときの Z 軸の座標を「#11」、工具がテーパの下の位置に来たときの Y 軸の座標を「#7/2」（工具半径）とすれば、点 A の座標が決定するので、計算で求める必要があるのは b と c の距離だけです。

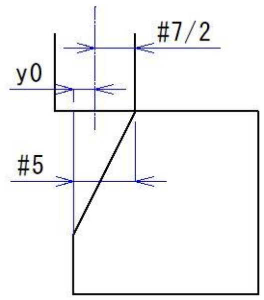


今回はボールエンドミルとラジアスエンドミルの両方で使えるプログラムを作りたいので、計算が少々複雑になりますが、1 つずつ見ていきたいと思います。

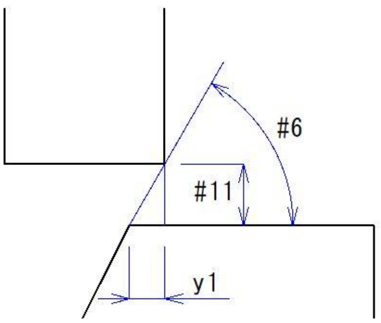
距離 b を求め方

まずは b の距離を求めます。考え方はいくつかありますが今回は、まず工具 R が 0 の状態の座標を求めた後、Z 軸の逃げ（#11）による Y 軸の変化量と、工具 R による Y 軸の変化量を足していく方法を取ります。

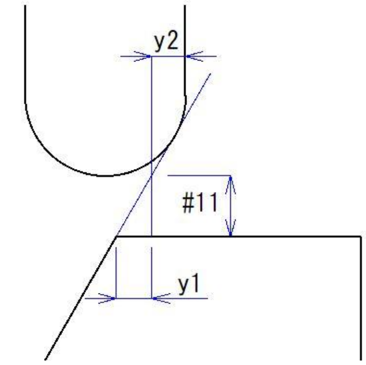
「工具 R が 0 」で「Z 軸の逃げ量も 0 」の場合は「テーパ幅 - 工具半径」となるので、工具の中心座標を y0 とおくと、式は「y0 = #5 - #7 / 2」となります。



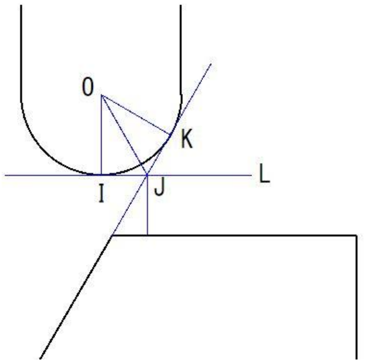
Z 軸の逃げ（#11）による Y 軸の変化量は、Y 軸の変化量を y1 とおくと「TAN[#6] = #11 / y1」となるので、左辺が y1 となるように整理すると「y1 = #11 / TAN[#6]」となります。



次に工具Rを入れたときのY軸の変化量を考えていきます。工具 R が 0 のときは y1 の位置に工具の側面がきていましたが、右の図を見ても分かる通り、工具 R が入ると y2 の分だけ変化します。つまり y2 を求めればよいわけです。



右の図はいくつか補助線を引き、各点に記号をつけたものです。点 J は上の図の y 1 と#11 の座標となる点で、点 O は工具 R の中心です。点 J から円との接線を 2 本引き、その交点を点 I と点 K とし、線 IJ の延長線上に点 L をおきます。このとき、上の図の y2 というのは「工具 R - IJ」となるので、IJ の距離を求めれば良いことになります。OI の距離は工具 R と同じなので、後は∠IOJ が分かれば「IJ = OI * TAN[∠IOJ]」で IJ を求めることができます。

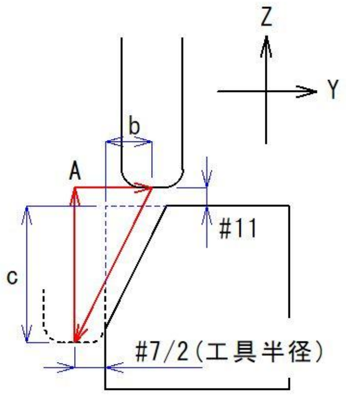


四角形 IJKO について、∠OIJ と∠OKJ は定義上 90°（接線と垂直な線）で、四角形の内角の和は 360°なので、残りの∠IOK と∠IJK の和は 180°となります。また、∠KJL というのは、テーパ角度（#6）と同じなので「∠IJK = 180 - #6」となり、従って「∠IOK = 180 - ∠IJK」より「∠IOK = #6」ということがわかります。

また、△OIJ と△OKJ について、辺 OI = 辺 OK、辺 OJ は共通の辺なので、2 つの直角三角形の合同条件である「斜辺と他の 1 辺がそれぞれ等しい」に当てはまるので、この 2 つの三角形は合同であるといえます。従って「∠IOJ = ∠KOJ」となるので、「∠IOJ = #6 / 2」だとわかります。

さて、私たちは y2 を求めようとしていたわけですが、「y2 = #4（工具 R）- IJ」から始まり、「IJ = #4 * TAN[#6 / 2]」がわかりましたので、一本目の式に代入すると「y2 = #4 - #4 * TAN[#6 / 2]」となります。

これで距離 b を求める式が出そろいました。まとめると、以下のようになります。



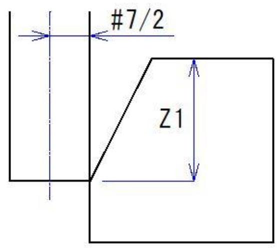
・距離 b = y0 + y1 + y2
・y0 = #5 - #7 / 2
・y1 = #11 / TAN[#6]
・y2 = #4 - #4 * TAN[#6 / 2]

代入すると
・距離 b = [#5 - #7 / 2] + [#11 / TAN[#6]] + [#4 - #4 * TAN[#6 / 2]]
となります。

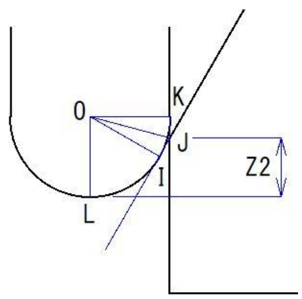
距離 c 求め方

距離 b と同じく、こちらも工具 R が 0 のときの座標を求めた後、工具 R の分を補正していく方法を取ります。

Y 軸方向の逃げは工具半径（#7 / 2）と決めておいたので、工具 R が 0 の場合の Z 軸方向の座標は、テーパの Z 軸方向の距離と同じとなります。テーパの Y 軸方向の距離（#5）と角度（#6）は初期設定で変数になっているわけですから、「Z1 ＝ #5 ＊ TAN [#6]」となります。



次に工具 R が入った場合について考えていきます。右の図はいくつか補助線を引き、各点に記号をつけたものです。少しわかりにくいかもしれませんが、点 J はワークのテーパと Z 軸と平行な部分との交点です。つまり、Y 座標は 0 （ゼロ）、Z 座標は Z1 の位置です。



図を見るとわかりますが、工具 R が入ると Z2 の距離分、工具が下がることがわかります。従って Z2 を求めればよいわけです。「Z2 ＝ OL - KJ」で求められ、OL は工具 R（#4）と同じ、残りの KJ は「KJ＝OK（工具 R） ＊TAN [∠JOK]」で求められるので、∠JOK を求めていきます。

距離 b を求めたときと同様に、∠LOI というのはテーパ角度（#6）と同じなので「∠IOK ＝ 90. - #6」だとわかります。また、線 OJ というのは∠IOK を 2 等分する線だということ

もわかっているので、「∠JOK = [90. - #6] / 2」ということになります。

まとめると、「KJ = #4 * TAN[[90. - #6] / 2]」となり、これを「Z2 = #4 - KJ」に代入すると以下の式が得られます。

・Z2 = #4 - #4 * TAN[[90. - #6] / 2]

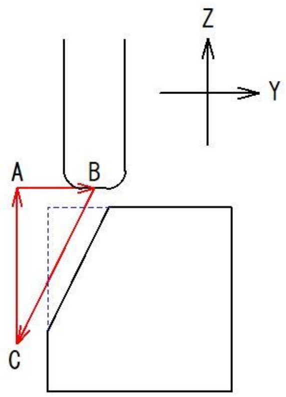
これで距離 c を求める式がそろいました。まとめると以下のようになります。

・距離 c　= Z1 + Z2
・Z1 = #5 * TAN[#6]
・Z2 = #4 - #4 * TAN[[90. - #6] / 2]

代入すると
・距離 c　= [#5 * TAN[#6]] + [#4 - #4 * TAN[[90. - #6] / 2]]
となります。

プログラムのコーディング

すべての座標がそろいましたので、実際にプログラムをコーディングしていきます。まず、右の図の 3 つの座標をプログラム内で使えるように変数にしていきます。座標は 3 つですが、点 A の座標は、点 C の Y 座標と点 B の Z 座標と同じですから、実質 2 つの座標（Y 軸 Z 軸合わせて 4 つの座標値）ということになります。



工具半径はあらかじめ計算して変数にしておきます。これは必ず必要というわけではありませんが、ほとんどのプログラムでは工具直径よりも工具半径を使用することが多いので、私の場合は習慣のようなものです。それと、加工原点とテーパの Y 軸の位置が違う場合を想定して、#1 の変数も定義していましたので、Y 軸の座標には#1 も足していきます。

工具半径
・#27 = #7 / 2
点 B
・Y 座標（距離 b）#31 = #1+ [#5 - #27] + [#11 / TAN[#6]] + [#4 - #4 * TAN[#6 / 2]]
・Z 座標（Z 逃げ位置）#11