Autolisp 繪製端板及圓桶形

各申機械設計工作室

2021-09-28

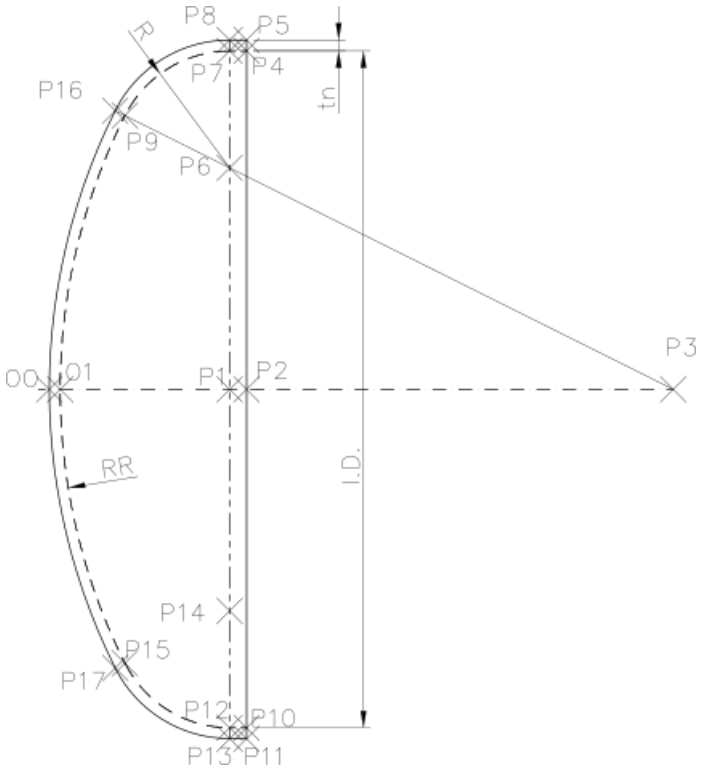
Contents

1 簡介

使用 autolisp **繪製近似半橢圓端板、碟形端板及圓桶形圖形。欲討論詳細內容請諮詢**

- **各申機械設計工作室**
  - **網址：**`https://sites.google.com/view/each-application/homepage`
  - Email：every.push.colin@gmail.com

2 端板



2.1 使用 block-make 環境

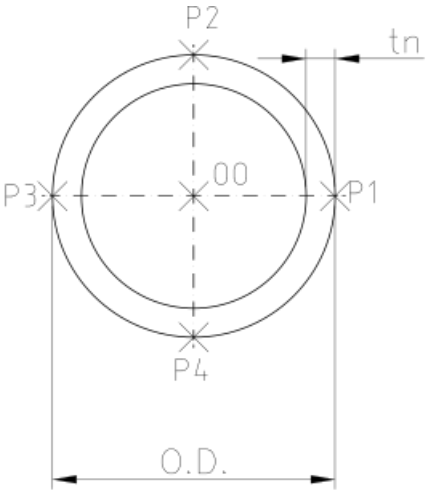
;;; 端板，近似半橢圓及碟形端板

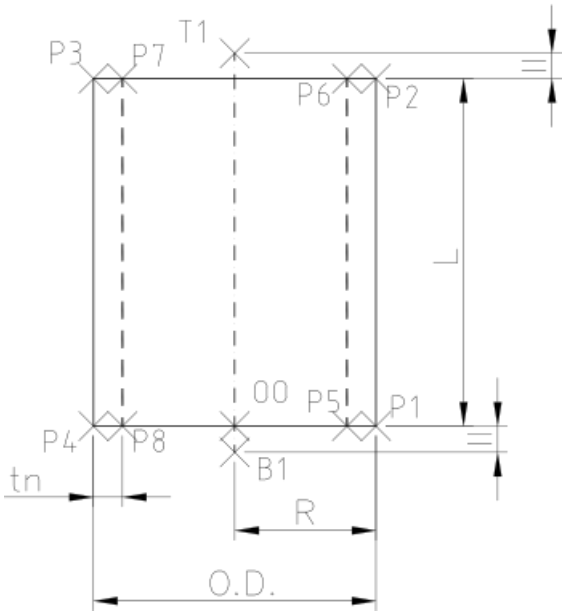
```
(defun head-f-lambda (arglist
                      / args
                      ID tn RR R L
                      SETA OO O1 P1 P2 P3 P4 P5 P6 P7 P8
                      P9 P10 P11 P12 P13 P14 P15 P16 P17
                      head_item)
  (setq args (cdr (assoc "head-f" arglist)))

  ;; 加載 "cal" 外部程式
  (if (null C:CAL) (arxload "GEOMCAL"))

  (setq ID (cdr (assoc 'ID args))
        tn (cdr (assoc 'tn args))
        RR (cdr (assoc 'RR args))
        R (cdr (assoc 'R args))
        L (cdr (assoc 'L args)))

  (setq SETA ((lambda (x y)
                (cal "asin((ID*0.5 -y)/(x - y))"))
              RR R))
```

```
(setq OO '(0 0)
      O1 (polar OO 0 tn)
      P3 (polar O1 0 RR)
      P1 (polar O1 0 (cal "RR - (RR-r)*cos(SETA)"))
      P2 (polar P1 0 L)
      P4 (polar P2 (* pi 0.5) (/ ID 2))
      P5 (polar P4 (* pi 0.5) tn)
      P7 (polar P1 (* pi 0.5) (/ ID 2))
      P8 (polar P7 (* pi 0.5) tn)
      P6 (polar P7 (* pi 1.5) R)
      P9 (polar P3 (* pi (/ (- 180.0 SETA) 180.0)) RR)
      P10 (polar P4 (* pi 1.5) ID)
      P11 (polar P10 (* pi 1.5) tn)
      P12 (polar P7 (* pi 1.5) ID)
      P13 (polar P12 (* pi 1.5) tn)
      P14 (polar P12 (* pi 0.5) R)
      P15 (polar P3 (* pi (/ (+ 180.0 SETA) 180.0)) RR)
      P16 (polar P3 (* pi (/ (- 180.0 SETA) 180.0)) (+ RR tn))
      P17 (polar P3 (* pi (/ (+ 180.0 SETA) 180.0)) (+ RR tn)))

(lwpolyline-draw
 (list '(8 . "yb-hidden-line")
       (cons 10 P4)
       (cons 10 P7)
       (cons 42 (bulge P7 P9 P6))
       (cons 10 P9)
       (cons 42 (bulge P9 P15 P3))
       (cons 10 P15)
       (cons 42 (bulge P15 P12 P14))
       (cons 10 P12)
       (cons 10 P10)))

(lwpolyline-draw
 (list '(8 . "yb-solid-line")
       '(70 . 1)
       (cons 10 P5)
       (cons 10 P8)
       (cons 42 (bulge P8 P16 P6))
       (cons 10 P16)
       (cons 42 (bulge P16 P17 P3))
       (cons 10 P17)
       (cons 42 (bulge P17 P13 P14))
       (cons 10 P13)
       (cons 10 P11)))

(lwpolyline-draw
 (list '(8 . "yb-dashed-line")
```

```
        (cons 10 P8)
        (cons 10 P13)))

  (center-line-draw (list (cons 'P1 OO)
                          (cons 'P2 P2))))

(defun head-f (ID tn RR R L blockname
               / arglist OO SETA O1 P1 P2)

  (if (null C:CAL) (arxload "GEOMCAL"))

  ;;(print SETA)
  (setq OO '(0 0)
        SETA ((lambda (x y)
                (cal "asin((ID * 0.5 - y)/(x - y))"))
              RR R)
        O1 (polar OO 0 tn)
        P1 (polar O1 0 (cal "RR - (RR-r) * cos(SETA)"))
        P2 (polar P1 0 L))
  ;;(print (list P1 P2))

  (setq arglist (list (cons 2 blockname)
                      (cons 10 P2)
                      (cons 8 "0")
                      (cons 'app 'head-f-lambda)
                      (list -3
                            (list "head-f"
                                  (cons 'ID ID)
                                  (cons 'tn tn)
                                  (cons 'RR RR)
                                  (cons 'R R)
                                  (cons 'L L)))))
  (print arglist)
  (block-make arglist))
```

3 圓桶形





3.1 初始作法

;;; 圓桶形，殼板

```
;;(setq *cylinder-type* nil)

;; 參數類型： OD tn L
(defun cylinder-f (OD tn L blockname
                   / R ll OO P1 P2 P3 P4 P5 P6 P7 P8 B1 T1)
  (command "_.undo" "e")
  (command "_.undo" "BE")
  (YB_env_set "0" 0 YB_error_env_revert)
  (setq R (/ OD 2)
```

```
        ;; 中心線突長
        ll (* 2 (/ (max OD L) 50.0)))
  (setq OO '(0 0 0))
  (setq P1 (polar OO 0.0 R)
        P2 (polar P1 (* pi 0.5) L)
        P3 (polar P2 pi OD)
        P4 (polar OO pi R)
        P5 (polar P1 pi tn)
        P6 (polar P2 pi tn)
        P7 (polar P3 0 tn)
        P8 (polar P4 0 tn)
        B1 (polar OO (* pi 1.5) ll)
        T1 (polar OO (* pi 0.5) (+ L ll)))

  (entmake (list '(0 . "BLOCK")
                 '(100 . "AcDbEntity")
                 '(100 . "AcDbBlockBegin")
                 (cons 2 blockname)
                 '(70 . 0)
                 (cons 10 OO)))

  (line-draw P1 P2 "yb-solid-line")
  (line-draw P2 P3 "yb-solid-line")
  (line-draw P3 P4 "yb-solid-line")
  (line-draw P4 P1 "yb-solid-line")
  (line-draw P5 P6 "yb-hidden-line")
  (line-draw P7 P8 "yb-hidden-line")
  (line-draw B1 T1 "yb-center-line")

  (entmake (list '(0 . "ENDBLK")
                 '(100 . "AcDbEntity")
                 '(100 . "AcDbBlockEnd")))
  (command "_.undo" "e")
  (YB_env_revert))

;; 參數類型： ID tn L
;; 殼板
(defun shell-f (ID tn L blockname)
  ;;OD = ID + 2 * tn
  (cylinder-f (+ ID (* 2 tn)) tn L blockname))

;;參數類型： OD ID L
;;甜甜圈;墊片
(defun donuts-f (OD ID L blockname)
  ;; tn = (OD-ID)/2
  (cylinder-f OD (/ (- OD ID) 2) L blockname))
```

```
;;; 上視圖
(defun cylinder-t (OD tn blockname / OO Ro Ri ll P1 P2 P3 P4)
  ;; 結束上一次的 undo be
  (command "_.undo" "e")
  (command "_.undo" "BE")
  (YB_env_set "0" 0 YB_error_env_revert)
  (setq OO '(0 0 0)
        Ro (/ OD 2)
        Ri (- Ro tn)
        ll (+ Ro (* 2.0 (/ OD 50.0))))
  (setq P1 (polar OO 0 ll)
        P2 (polar OO (* pi 0.5) ll)
        P3 (polar OO pi ll)
        P4 (polar OO (* pi 1.5) ll))

  (entmake (list '(0 . "BLOCK")
                 '(100 . "AcDbEntity")
                 '(100 . "AcDbBlockBegin")
                 (cons 2 blockname)
                 '(70 . 0)
                 (cons 10 OO)))

  (circle-draw OO Ro "yb-solid-line")
  (circle-draw OO Ri "yb-solid-line")

  (line-draw OO P1 "yb-center-line")
  (line-draw OO P2 "yb-center-line")
  (line-draw OO P3 "yb-center-line")
  (line-draw OO P4 "yb-center-line")

  (entmake (list '(0 . "ENDBLK")
                 '(100 . "AcDbEntity")
                 '(100 . "AcDbBlockEnd")))

  (command "_.undo" "e")
  (YB_env_revert))


;; Shell-t
(defun Shell-t (ID tn blockname)
  ;; OD = ID + tn * 2
  (cylinder-t (+ ID (* tn 2)) tn blockname))

;;donuts-t
;; 墊片-t
(defun donuts-t (OD ID blockname)
  ;; tn = (OD - ID)/2
```

```
  (cylinder-t OD (/ (- OD ID) 2) blockname))
```

3.2 使用 block-make

;;; 使用 block-make 產生 cylinder 圖塊

;;前視圖

```
(defun cylinder-f-lambda (arglist
                          / args OD ID tn L R ll OO
                          P1 P2 P3 P4 P5 P6 P7 P8 B1 T1)
  (cond ((setq args (cdr (assoc "cylinder-f" arglist)))
         (setq OD (cdr (assoc 'od args))
               tn (cdr (assoc 'tn args))
               L (cdr (assoc 'L args))))
        ((setq args (cdr (assoc "shell-f" arglist)))
         (setq ID (cdr (assoc 'id args))
               tn (cdr (assoc 'tn args))
               L (cdr (assoc 'L args)))
         (setq OD (+ Id (* tn 2))))
        ((setq args (cdr (assoc "donuts-f" arglist)))
         (setq OD (cdr (assoc 'od args))
               ID (cdr (assoc 'id args))
               L  (cdr (assoc 'L args)))
         (setq tn (/ (- OD ID) 2)))
        (t
         (setq args nil)))
  (if args
      (progn
        (setq R (/ OD 2)
              ;; 中心線突長
              ll (* 2 (/ (max OD L) 50.0)))
        (setq OO '(0 0 0))
        (setq P1 (polar OO 0.0 R)
              P2 (polar P1 (* pi 0.5) L)
              P3 (polar P2 pi OD)
              P4 (polar OO pi R)
              P5 (polar P1 pi tn)
              P6 (polar P2 pi tn)
              P7 (polar P3 0 tn)
              P8 (polar P4 0 tn)
              B1 (polar OO (* pi 1.5) ll)
              T1 (polar OO (* pi 0.5) (+ L ll)))

        (mapcar '(lambda (x)
                   (apply 'line-draw x))
                (list (list P1 P2 "yb-solid-line")
```

```
                    (list P2 P3 "yb-solid-line")
                    (list P3 P4 "yb-solid-line")
                    (list P4 P1 "yb-solid-line")
                    (list P5 P6 "yb-hidden-line")
                    (list P7 P8 "yb-hidden-line")
                    (list B1 T1 "yb-center-line"))))
      (print "沒有繪製圓桶，參數未設置")))

;; 上視圖
(defun cylinder-t-lambda (arglist
                          / args OO Ro Ri ll P1 P2 P3 P4)
  (cond ((setq args (cdr (assoc "cylinder-t" arglist)))
         (setq OD (cdr (assoc 'od args))
               tn (cdr (assoc 'tn args))))
        ((setq args (cdr (assoc "shell-t" arglist)))
         (setq ID (cdr (assoc 'id args))
               tn (cdr (assoc 'tn args)))
         (setq OD (+ ID (* tn 2))))
        ((setq args (cdr (assoc "donuts-t" arglist)))
         (setq OD (cdr (assoc 'od args))
               Id (cdr (assoc 'id args)))
         (setq tn (/ (- OD ID) 2)))
        (t
         (setq args nil)))
  (if args
      (progn
        (setq OO '(0 0 0)
              Ro (/ OD 2)
              Ri (- Ro tn)
              P1 (polar OO 0 Ro)
              P2 (polar OO (* pi 0.5) Ro)
              P3 (polar OO pi Ro)
              P4 (polar OO (* pi 1.5) Ro))

        (mapcar '(lambda (x)
                   (apply 'circle-draw x))
                (list (list OO Ro "yb-solid-line")
                      (list OO Ri "yb-solid-line")))

        (mapcar '(lambda (x)
                   (center-line-draw (list
                                       (cons 'P0 OO)
                                       (cons 'P2 x))))
                (list P1 P2 P3 P4)))

      (print "圓桶形上視未繪製，參數設置錯誤")))
```

```
(defun cylinder-f (OD tn L blockname / arglist)
  (setq arglist (list (cons 2 blockname)
                      (cons 10 '(0 0 0))
                      (cons 8 "0")
                      (cons 'app 'cylinder-f-lambda)
                      (list -3 (list "cylinder-f"
                                     (cons 'od OD)
                                     (cons 'tn tn)
                                     (cons 'L L)))))
  (block-make arglist))

(defun cylinder-t (OD tn blockname / arglist)
  (setq arglist (list (cons 2 blockname)
                      (cons 10 '(0 0 0))
                      (cons 8 "0")
                      (cons 'app 'cylinder-t-lambda)
                      (list -3 (list "cylinder-t"
                                     (cons 'od OD)
                                     (cons 'tn tn)))))
  (block-make arglist))
```