

```

;
;  seting linked lists of equidistant nodes nodes
;
def set_list
  ver_head = null
  i_pnt = i_head
  loop while i_pnt # null
    node_pnt = i_node_head(i_pnt)
    loop while node_pnt # null
      z_pos = in_pos(node_pnt,3)
      ver_pnt = ver_head
      section
        if ver_head = null then
          new = get_mem(5)
          mem(new) = ver_head
          ver_head = new
          mem(new+2) = z_pos
          mem(new+4) = gp_near(0,0,z_pos)
          hnew = get_mem(2)
          mem(hnew+1) = node_pnt
          mem(new+1) = hnew
          mem(hnew) = null
          exit section
        end_if
      loop while ver_pnt # null
        ver_pnt_n = mem(ver_pnt)
        if abs(mem(ver_pnt+2)-z_pos)<0.1 then
          hnew = get_mem(2)
          mem(hnew+1) = node_pnt
          mem(hnew) = mem(ver_pnt+1)
          mem(ver_pnt+1) = hnew
          exit section
        end_if
      if ver_pnt = ver_head then
        if z_pos > mem(ver_pnt+2) then
          new = get_mem(5)
          mem(new) = ver_head
          ver_head = new
          mem(new+2) = z_pos
          mem(new+4) = gp_near(0,0,z_pos)
          hnew = get_mem(2)
          mem(hnew+1) = node_pnt
          mem(new+1) = hnew
          mem(hnew) = null

```

```

        exit section
    end_if
end_if
if ver_pnt_n = null then
    new = get_mem(5)
    mem(new) = null
    mem(ver_pnt) = new
    mem(new+2) = z_pos
    mem(new+4) = gp_near(0, 0, z_pos)
    hnew = get_mem(2)
    mem(hnew+1) = node_pnt
    mem(new+1) = hnew
    mem(hnew) = null
    exit section
end_if
if z_pos < mem(ver_pnt+2) then
    if z_pos > mem(ver_pnt_n+2) then
        new = get_mem(5)
        mem(new) = ver_pnt_n
        mem(ver_pnt) = new
        mem(new+2) = z_pos
        mem(new+4) = gp_near(0, 0, z_pos)
        hnew = get_mem(2)
        mem(hnew+1) = node_pnt
        mem(new+1) = hnew
        mem(hnew) = null
        exit section
    end_if
end_if
ver_pnt = ver_pnt_n
end_loop
end_section
node_pnt = in_next(node_pnt)
end_loop
i_pnt = i_next(i_pnt)
end_loop
;
; --- calculate length corresponding to each segment
;
ver_pnt = ver_head
z_up = mem(ver_pnt+2)
loop while ver_pnt # null
    ver_pnt_n = mem(ver_pnt)
    if ver_pnt_n # null then

```

```

        z_down = mem(ver_pnt_n+2)
    else
        z_down = mem(ver_pnt+2)
    end_if
    mem(ver_pnt+3) = 0.5*(z_up-z_down)
    z_up = mem(ver_pnt+2)
    ver_pnt = ver_pnt_n
end_loop
end
set_list
;
; calculation of average pressure during calculation
;
def gen_curve
    while_stepping
        if step-100*int(step/100)=0 then
            tot_reac = 0.
            vcount = 0
            ver_pnt = ver_head
            loop while ver_pnt # null
                vcount = vcount+1
                hor_pnt = mem(ver_pnt+1)
                hor_for = 0.
                hcount = 0
                loop while hor_pnt # null
                    node_pnt = mem(hor_pnt+1)
                    hcount = hcount+1
                    area = in_area(node_pnt)
                    hor_nor = in_nstr(node_pnt)
                    hor_she = in_sstr(node_pnt,1)
                ;
            ; --- finding orientation of the normal ---
            ;
                t_zone = in_ztarget(node_pnt)
                t_face = in_ftarget(node_pnt)
                t_gp1 = z_facegp(t_zone, t_face, 1)
                t_gp2 = z_facegp(t_zone, t_face, 2)
                t_gp3 = z_facegp(t_zone, t_face, 3)
            ;
                dx1 = gp_xpos(t_gp2)-gp_xpos(t_gp1)
                dy1 = gp_ypos(t_gp2)-gp_ypos(t_gp1)
                dz1 = gp_zpos(t_gp2)-gp_zpos(t_gp1)
            ;
                dx2 = gp_xpos(t_gp3)-gp_xpos(t_gp1)

```

```

dy2      = gp_ypos(t_gp3)-gp_ypos(t_gp1)
dz2      = gp_zpos(t_gp3)-gp_zpos(t_gp1)
;
dxn      = dy1*dz2-dz1*dy2
dyn      = dz1*dx2-dx1*dz2
dzn      = dx1*dy2-dy1*dx2
dn       = sqrt(dxn*dxn+dyn*dyn+dzn*dzn)
dxn      = dxn/dn
;
hor_nor = hor_nor*dxn
hor_for = hor_for+(hor_nor+hor_she)*area
hor_pnt = mem(hor_pnt)
end_loop
gp_pile = mem(ver_pnt+4)
hor_dis = gp_xdisp(gp_pile)
tot_reac = tot_reac+hor_for
table(vcount,hor_dis) = hor_for/mem(ver_pnt+3)
ver_pnt = mem(ver_pnt)
end_loop
end_if
end

```