```
(* :Name: MultiLayerIndentation *)

(* :Title: Indentation of elastic multilayered structures *)

(* :Author:  Andrei Constantinescu, Alexander Korsunsky, Olivier Pison *)

(* :Summary:
      This package provides the indentation analysis of multilayered
      half-space under the assumptions of small strains, axial symmetry and linear isotropic

      Indentor shapes: conical, spherical, flat punch, blunted cone
*)

(* :Context: MultiLayerIndentation *)

(* :Package Version: 1 *)

(* :Copyright: Copyright 2013 Andrei Constainescu, Alexander Korsunsky *)

(* :History:
       OneLayerIndentation
       MultiLayerIdentation
*)

(* :Keywords:
      conical identor, spherical indentor, flat punch, blunted cone, multilayer, indentation

*)

(* :Source:

    A. Constantinescu and A.M. Korsunsky-Elasticity with Mathematica (r)
          Cambridge University Press, 2007

     A.M. Korsunsky, A. Constantinescu - The influence of indenter bluntness on the apparent
           Thin Solid Films 517 (2009) 48354844

    A. Constantinescu, A.M. Korsunsky, O. Pison, A. Oueslati - Symbolic and numerical soluti
           submitted to Int.J;Solids and structures, 2013

    H.Y. Yu, S.C. Sanday, B.B. Rath, The effect of substrate on the elastic properties of fi
          J. Mech. Phys. Solids 38 (6) (1990) 745.

    N.N. Lebedev, I.S. Uflyand, PMM 22 (1958) 320
*)

(* :Mathematica Version: 7.0 *)

(* :Limitation:

*)

BeginPackage["MultiLayerIndentation`"]

Unprotect[GaussStuff,GaussStuffInterval,GaussInterval,GaussIntegrate,LagrangeInterpolation,
OneLayerIndentation,MultiLayerIndentation,NLinearSolve];
```

```
OneLayerIndentation::usage = "

OneLayerIndentation[ depthslist, {punchform, punchvariables}, material, ndiscret] computes t
depths given as depthslist and returs a list containing in each element:

 {depth, indentation force, contact radius,  relative contact radius, relative contact modul

Punchforms:
 flat punch – {Flat, contact radius}
 blunted cone – {Rock, angle, radius}
 conical indentor – {Cone, angle}
 spherical indentor – {Hertz, sphere radius}

where Flat, Rock, Cone, Hertz are the corresponding strings

Material = {E1,ny1,E2,ny2}

ndiscret = {nequation, nkernel} – are the natural numbers describing the degree discretizati
         and of the kernel of the Fredholm integral equation
 "

MultiLayerIndentation::usage = "

MultiLayerIndentation[ depthslist, {punchform, punchvariables}, material, ndiscret] computes
depths given as depthslist and returs a list containing in each element:

 {depth, indentation force, contact radius,  relative contact radius, relative contact modul

Punchforms:
 flat punch – {Flat, contact radius}
 blunted cone – {Rock, angle, radius}
 conical indentor – {Cone, angle}
 spherical indentor – {Hertz, sphere radius}

where Flat, Rock, Cone, Hertz are the corresponding strings

Material =

ndiscret = {nequation, nkernel} – are the natural numbers describing the degree discretizati
         and of the kernel of the Fredholm integral equation

"

GaussStuff::usage = "

GaussStuff[n]

Provides a list{ gausspoints, gaussweights} on the Interval [–1,1] coressponding
   to the Roots of a LegendreP polynomial of order n

"

GaussStuffInterval::usage = "
```

```
GaussStuffInterval[n, {a,b}]

Provides a list{ gausspoints, gaussweights} on the Interval [a,b] coressponding
    to the Roots of a LegendreP polynomial of order n  "

GaussIntegrate::usage = "

GaussIntegrate[ f , {a,b},  n ]

Numerically integrates the function f on the Interval [a,b]  using a sum over the gausspoint
 with the gaussweights provided by the roots of a LegendreP polynomial of order n

"

NLinearSolve::usage = "

NLinearSolve[m,f]

gives the solution of the linear equation m . x = f
    with m a matrix and f a vector

"
```

```
Begin["MultiLayerIndentation`Private`"]

(* Gauss - Legendre integration on arbitrary interval *)

GaussStuff[n_] := GaussStuff[n] = Module[{ gausspoints, gaussweights},
      DLP[x_] = D[ LegendreP[n, x] , x];
      gausspoints = Table[Root[ Function[y,LegendreP[ n , y ]] , i],{i,1,n}];
      gaussweights = Map[Function[x,2 / (1 - x^2) / DLP[x]^2] , gausspoints];
      N[{gausspoints, gaussweights},MachinePrecision]
      ]

GaussStuffInterval[n_, interval_List] := GaussStuffInterval[n, interval] =
    Module[ {b = interval[[2]], a = interval[[1]]  } ,
      {Map[Function[x,a + 0.5 (b - a)  ( x + 1)],GaussStuff[n][[1]]] ,
        0.5 (b - a)  GaussStuff[n][[2]]}
      ]

GaussIntegrate[ f_ , interval_List,  n_Integer ] :=
  Module[ {b = interval[[2]], a = interval[[1]]  } ,
    gausspoints = GaussStuffInterval[n, {a, b} ][[1]];
    gaussweights =  GaussStuffInterval[n, {a, b}][[2]];
    Map[ f, gausspoints ] .  gaussweights  ]
```

```
NLinearSolve = Compile[{{m,_Real,2},{f, _Real, 1}},
     LinearSolve[m,f]]

OneLayerIndentation[dlist_?ListQ, punch_?ListQ, bimat_?ListQ, ndiscret_?ListQ] :=
Module[{a,P,d,γ,aovergamma,f,Pstar, results = {},
       a1,a2,b1,b2,k1,k2,beta,
       mu1,mu2,E1,ny1,E2,ny2,
       B1,B2,B3,B4,B5,B6,C1,C2,C3,C4,C5,
       mat,matf,myK,myKf,H,g},

E1=bimat[[1]]; ny1=bimat[[2]]; E2=bimat[[3]]; ny2=bimat[[4]];


bond = bimat[[5]];

nequation = ndiscret[[1]];
nkernel = ndiscret[[2]];


(* Construction of g *)

mu1 = E1/2/(1+ny1); mu2 = E2/2/(1+ny2); eta=a2 mu1/ a1/ mu2;
beta = mu1/mu2; beta2 = beta^2; beta3 = beta^3;
a1 = 1-ny1; b1 = 1-2*ny1; k1 = 3-4*ny1;
a2 = 1-ny2; b2 = 1-2*ny2; k2 = 3-4*ny2;
B1 = 4*a1*a2*(1+b2*beta)*beta;
B2 = -1-3*b2*beta+(k2-2*b2^2)*beta2+b2*k2*beta3;
B3 = 2*a2*(1+2*k1+2*b2*k1*beta+k2*beta^2)*beta;
B4 = k1+(-b2+2*b2*k1+4*a1*a2)*beta+(k1+4*a1*a2*k2+2*b1*b2^2)*beta2+b2*k2*beta3;
B5 = -1-(1+4*b2)*beta+(1-2*b2)*k2*beta2+k2^2*beta3;
B6 = k1+(1+2*k1*k2)*beta+(k2+4*a2*b2*k1+16*a1*a2^2+2*b1*b2^2)*beta2+k2^2*beta3;
C1 = 4*a1^2*(1+b2*beta);
C2 = 1+3*b2*beta-(k2-2*b2^2)*beta2-b2*k2*beta3;
C3 = k1-(1-2*k1)*b2*beta+(k2+16*a1*a2^2+2*b1*b2^2)*beta2+b2*k2*beta3;
C4 = 4*a1^2+4*a1*(2*a2+a1*k2)*beta+8*a1*a2*b1*beta2;
C5 = 1+(k2+2*b2)*beta-(1-2*b2)*k2*beta2-k2^2*beta3;

Switch[ bond,

  "Bound",  (* Perfect Bounded layer *)
  g = Compile[{{w, _Real}},
    1 - ((B1 + B2 w + B3 Sinh[w]^2 ) Exp[ - w ]
    + (B4 + B5 w + B6 Sinh[w]^2 ) Sinh[w] ) /
    ((C1 + C2 w^2 + C3 Sinh[w]^2 ) Exp[ -w] +
    (C4 + C5 w^2 + B6 Sinh[w]^2 ) Sinh[ w] )],

  "Slide",  (* Frictionless adhesion *)
    g = Compile[{{w, _Real}},  1 - (Sinh[w]^2 +
    eta (w +Sinh[w]Cosh[w]))/(w+ Sinh[w]Cosh[w] +
    eta (Sinh[w]^2-w^2) ) ],
  _,
  Print["Bad word for Bound or Slide"];Abort[]
  ];



(* Terms depending on the form of the punch *)

Switch[ punch[[1]],
  "Flat", f = 1 &;
      aovergamma := 1;
      Pstar := P / 2 /d /a,

  "Cone", f = 1 - punch[[2]] 0.5 Pi aa # / d &;
      aovergamma := 2 d / punch[[2]] / Pi;
      Pstar := P punch[[2]] Pi /2 /d^2,

  "Hertz",f = 1 - aa^2 #^2 punch[[2]]/ d  &;
      aovergamma := (d/punch[[2]])^0.5;
      Pstar := 0.75 P (punch[[2]]/d^3)^0.5,

  "Rock", f = 1 - aa^2 #^2 punch[[2]] / punch[[3]] /d
        (1 + If [ aa # - punch[[3]] < 0, 0,
        punch[[3]]/aa/# ArcCos[ punch[[3]] / aa / # ]
       - ( 1- (punch[[3]] / aa / # )^2 )^0.5  ] ) &;
       aovergamma := 2  d/punch[[2]]/Pi;
       Pstar := P  punch[[2]] Pi /2 /d^2,

  _,
 Print["Bad word for Punch, i.e Flat, Cone, ... "];Abort[]
 ];

F[a_, d_] := Map[ f /. aa -> a ,  gp ] ;

(* computing function giving the matrix
     of the coated halfspace *)

{gp03, gw03} = GaussStuffInterval[nkernel, {0, 3} ];
{gp310,gw310} = GaussStuffInterval[nkernel, {3, 10} ];

myK =  Map[ a g[#] Cos[a u #] & , gp03 ] .  gw03 +
 Map[ a g[#] Cos[a u #] & , gp310 ] .  gw310 ;
myKf = Evaluate[myK /. u -> Plus[#1, #2] ] &;

{gp, gw} = GaussStuffInterval[nequation, {0, 1}];

 matf =  IdentityMatrix[nequation] -
     1/ Pi   Inner[ Times,
        Outer[ myKf , gp   , - gp  ] , gw ,
        List] -
     1/ Pi    Inner[ Times,
        Outer[ myKf ,  gp  ,   gp ], gw ,
        List];

 mat = Compile[ {{a, _Real}}, Evaluate[ matf] ];

 H[a_, d_] :=
      NLinearSolve[ mat[ a ] , F[a, d ] ];

 (*Choosing of a contact radius a.
 Resolution of the integral equation.
 Verification of the value of the contact radius (sigmazz=
  0 in a for all indenters except the flat punch) *)

  (* For every indentation depth given in "dlist" *)

  num = Length[dlist];


  If[punch[[1]] == "Flat",
   (

    result = Timing[
       Table[
        d = dlist[[j]];
                          γ = 1; a = punch[[2]];

        (* P is found for each d by integration *)
        {d,
         2 E1 / (1 - ny1^2) a d H[a,d] . gw}
        , {j, 1, num}]];
    ),
         (
         result = Timing[
       Table[
        d = dlist[[j]];

        (* Finding of the contact radius : resolution of H[a,d][[
        nequation]]==0 through dichotomic search *)

        a0 = 0.8;
        b0 = 1.2;

        While[Abs[H[a0 aovergamma, d][[nequation]]] > preci,
         c = (a0 + b0)/2;
         If[H[c aovergamma, d][[nequation]] > 0, a0 = c, b0 = c]];

        γ = a0; a = γ aovergamma;
        (* P is found for each d by integration *)

         P = 2 E1 / (1 - ny1^2) a d H[a,d] . gw;

        {dlist[[j]],P,a,γ,Pstar}
        , {j, 1, num}]];
    )
   ];
  result[[2]]
 ]
```

```
MultiLayerIndentation[depl_?ListQ,  punch_?ListQ, material_?ListQ, ndiscret_?ListQ]  :=
 Module[{a, P, d, γ, aovergamma, f, Pstar, result,
          a0, b0, c, Reglenu, Reglemu, Regleh, λ, i, A5, A6,
   RH, RI, A1, A2, A3, A4, H1, H2, H3,
   H4, ν, νf, μ, μf, h, hf, mat, matf, myK, myKf, H,
 k, g, lm, lmc, n, preci, aux, aa, num, gp, gw, gpK, gwK, nequation,
  nkernel, F, NLinearSolve, u},

n = Dimensions[material[[1]]][[1]] - 1;

(* Initialization of the rules affecting the values of the parameters according to the inp

Reglenu =
 Thread[ Append[Table[ν[i], {i, n}], νf] -> material[[1]]];
Reglemu =
 Thread[Append[ Table[μ[i], {i, n}], μf] ->
   material[[2]]];
Regleh =
 Prepend[Append[ Thread[ Table[h[i], {i, n}] -> material[[3]] ] ,
   hf -> Last[material[[3]]]], h[0] -> 0. ];

preci = 10^-5;

(* Equations for the last interface *)

RH = {1/(2 μf (-1 + ν[i])) Sinh[λ (hf - h[-1 + i])]^2 (-μf (A6 +
        A5 (2 + hf λ - 2 νf)) + μf (A5 + A6 +
        A5 hf λ -
        2 A5 νf) Coth[λ (hf - h[-1 + i])] + (A6 +
        A5 hf λ - (A6 +
           A5 (3 + hf λ - 4 νf)) Coth[λ (hf -
             h[-1 + i])]) μ[i]),
  1/(2 μf (-1 + ν[i])) Sinh[λ (hf -
       h[-1 + i])]^2 (-μf (A5 + A6 + A5 hf λ -
        2 A5 νf) + μf (A6 +
        A5 (2 + hf λ - 2 νf)) Coth[λ (hf -
          h[-1 + i])] + (A6 +
        A5 (3 + hf λ - 4 νf) - (A6 +
           A5 hf λ) Coth[λ (hf - h[-1 + i])]) μ[
       i]), 1/(2 μf (-1 + ν[i])) Sinh[λ (hf - h[-1 + i])]^2 (μf (hf λ (A5 + A6 +
           A5 hf λ - 2 A5 νf) + (A6 (3 - hf λ) +
            A5 (3 - 6 νf +
              hf λ (1 - hf λ +
                 2 νf))) Coth[λ (hf - h[-1 + i])] -
        4 (A5 + A6 + A5 hf λ -
           2 A5 νf) Coth[λ (hf - h[-1 + i])] ν[
          i]) + μ[
       i] (A6 (2 - hf λ) -
        A5 hf λ (1 + hf λ -
           4 νf) + (A6 (-1 + hf λ) +
           A5 (-3 + hf λ (-1 + hf λ) +
              4 νf)) Coth[λ (hf - h[-1 + i])] +
        2 (-A6 -
           A5 hf λ + (A6 +
              A5 (3 + hf λ - 4 νf)) Coth[λ (hf -
                 h[-1 + i])]) ν[i])),
  1/(2 μf (-1 + ν[i])) Sinh[λ (hf - h[-1 + i])] (-hf λ μf (A5 + A6 +
        A5 hf λ -
        2 A5 νf) Cosh[λ (hf -
```

```
          h[-1 + i])] + μf Sinh[λ (hf -
          h[-1 + i])] (A6 (-3 + hf λ) +
        A5 (-3 + hf λ (-1 + hf λ - 2 νf) +
           6 νf) +
        4 (A5 + A6 + A5 hf λ - 2 A5 νf) ν[
          i]) + μ[
       i] (Cosh[λ (hf -
             h[-1 + i])] (A6 (-2 + hf λ) +
           A5 hf λ (1 + hf λ - 4 νf) +
           2 (A6 + A5 hf λ) ν[i]) +
        Sinh[λ (hf - h[-1 + i])] (A6 - A6 hf λ +
           A5 (3 + hf λ (1 - hf λ) - 4 νf) -
           2 (A6 + A5 (3 + hf λ - 4 νf)) ν[i]))))};

(* Equations for the interface between the layers i and i+1 *)

RI = {1/(4 μ[1 + i] (-1 + ν[i])) Csch[λ (h[i] -
       h[1 + i])] ((A3[1 + i] + λ A2[1 + i] h[i]) Sinh[
       2 λ (h[-1 + i] - h[i])] (μ[i] - μ[1 + i]) +
     2 A4[1 +
        i] Sinh[λ (h[-1 + i] - h[i])]^2 (-μ[i] + μ[
         1 + i]) - 2 A2[1 + i] μ[1 + i] (-1 + ν[1 + i]) +
     2 A2[1 + i] Cosh[2 λ (h[-1 + i] - h[i])] μ[
       1 + i] (-1 + ν[1 + i]) +
     A1[1 + i] (2 λ h[
          i] Sinh[λ (h[-1 + i] - h[i])]^2 (-μ[
             i] + μ[1 + i]) +
        Sinh[2 λ (h[-1 + i] - h[i])] (μ[
             1 + i] (1 - 2 ν[1 + i]) + μ[
             i] (-3 + 4 ν[1 + i])))),
  1/(4 μ[1 + i] (-1 + ν[i]))  Csch[λ (h[i] -  h[1 + i])] (-(A3[1 + i] + λ A2[1 + i] h[i])
     Cosh[
       2 λ (h[-1 + i] - h[i])] (A3[
         1 + i] + λ A2[1 + i] h[i]) (μ[i] - μ[
         1 + i]) +
     Sinh[2 λ (h[-1 + i] - h[i])] (A4[
          1 + i] (-μ[i] + μ[1 + i]) +
        2 A2[1 + i] μ[1 + i] (-1 + ν[1 + i])) +
     A1[1 + i] (3 μ[i] - μ[
         1 + i] + λ h[i] Sinh[
          2 λ (h[-1 + i] - h[i])] (-μ[i] + μ[
            1 + i]) - 4 μ[i] ν[1 + i] +
        2 μ[1 + i] ν[1 + i] +
        Cosh[2 λ (h[-1 + i] - h[i])] (μ[
             1 + i] (1 - 2 ν[1 + i]) + μ[
             i] (-3 + 4 ν[1 + i])))),
  1/(4 μ[1 + i] (-1 + ν[i])) Csch[λ (h[i] - h[1 + i])] (A4[
       1 + i] (λ h[i] Sinh[
          2 λ (-h[-1 + i] + h[i])] (-μ[i] + μ[
            1 + i]) - 2 μ[i] (-1 + ν[i]) +
        2 Cosh[2 λ (-h[-1 + i] + h[i])] μ[
          i] (-1 + ν[i])) + λ Cosh[
       2 λ (-h[-1 + i] + h[i])] h[
     i] (-(A3[1 + i] + λ A2[1 + i] h[i]) (μ[
          i] - μ[1 + i]) +
      A1[1 + i] (μ[
           i] (1 + 2 ν[i] - 4 ν[1 + i]) + μ[
           1 + i] (-1 + 2 ν[1 + i]))) + λ h[
     i] ((A3[1 + i] + λ A2[1 + i] h[i]) (μ[i] - μ[
          1 + i]) +
      A1[1 + i] (μ[1 + i] (1 - 2 ν[1 + i]) + μ[
           i] (-1 - 2 ν[i] + 4 ν[1 + i]))) +
   Sinh[2 λ (-h[-1 + i] + h[i])] (A3[
        1 + i] (μ[1 + i] (3 - 4 ν[i]) + μ[
           i] (-1 + 2 ν[i])) + λ A2[1 + i] h[
        i] (μ[i] (-1 + 2 ν[i]) + μ[
           1 + i] (1 - 4 ν[i] + 2 ν[1 + i])) +
      A1[1 + i] (μ[
           1 + i] (λ^2 h[
              i]^2 - (-3 + 4 ν[i]) (-1 +
               2 ν[1 + i])) - μ[
           i] (λ^2 h[
              i]^2 - (-1 + 2 ν[i]) (-3 + 4 ν[1 + i]))))))),
1/(4 μ[1 + i] (-1 + ν[i])) Csch[λ (h[i] - h[1 + i])] (2 A4[
     1 + i] (λ h[
        i] Sinh[λ (h[-1 + i] - h[i])]^2 (μ[i] - μ[
          1 + i]) +
      Sinh[2 λ (h[-1 + i] - h[i])] μ[
        i] (-1 + ν[i])) +
   A3[1 + i] (-μ[i] + 3 μ[1 + i] + λ h[i] Sinh[
        2 λ (h[-1 + i] - h[i])] (-μ[i] + μ[
          1 + i]) + 2 μ[i] ν[i] -
      4 μ[1 + i] ν[i] +
      Cosh[2 λ (h[-1 + i] - h[i])] (μ[
           i] (1 - 2 ν[i]) + μ[
           1 + i] (-3 + 4 ν[i]))) + λ A2[1 + i] h[
     i] (-μ[i] (λ h[i] Sinh[
           2 λ (h[-1 + i] - h[i])] +
         2 Sinh[λ (h[-1 + i] - h[i])]^2 (-1 +
            2 ν[i])) + μ[
        1 + i] (λ h[i] Sinh[
           2 λ (h[-1 + i] - h[i])] +
         2 Sinh[λ (h[-1 + i] - h[i])]^2 (-1 +
            4 ν[i] - 2 ν[1 + i]))) +
   A1[1 + i] (μ[
        1 + i] (-2 λ^2 h[
           i]^2 Sinh[λ (h[-1 + i] -
              h[i])]^2 + λ h[i] Sinh[
           2 λ (h[-1 + i] - h[i])] (-1 +
            2 ν[1 + i]) +
         2 Sinh[λ (h[-1 + i] - h[i])]^2 (-3 +
            4 ν[i]) (-1 + 2 ν[1 + i])) + μ[i] (2 λ^2 h[ i]^2 Sinh[λ (h[-1 + i] -
              h[i])]^2 + λ h[i] Sinh[
           2 λ (h[-1 + i] - h[i])] (1 + 2 ν[i] -
            4 ν[1 + i]) -
         2 Sinh[λ (h[-1 + i] - h[i])]^2 (-1 +
              2 ν[i]) (-3 + 4 ν[1 + i]))))};
lm = Table[
  Normal[((CoefficientArrays[
       RI /. i -> l /. Reglenu /. Reglemu /. Regleh, {A1[l + 1],
        A2[l + 1], A3[l + 1], A4[l + 1]}])[[2]]))]
  , {l, 1, n - 1}];

(* Construction of the list of the change of layer matrixes*)

lm = Append[lm,
  Normal[(CoefficientArrays[
      RH /. i -> n /. Reglenu /. Reglemu /. Regleh, {A5, A6}])[[
    2]]]];

lmc = Map[ Compile[{{λ, _Real}}, Evaluate[#]] &, lm ];

(* Product of the change of layer matrixes *)

k[l_] := Dot @@ Map[ #[l] &, lmc];

(* Definition of lambda -> g (lambda/h1) *)

g[ll_?NumericQ] := ( aux = k[1/(h[1] /. Regleh)*ll];
  1 + (-aux[[3, 1]] aux[[1, 2]] +
       aux[[1, 1]] aux[[3, 2]])/(aux[[4,
         1]]*(aux[[1, 2]] - aux[[3, 2]] -
          2 ν[1] aux[[1, 2]]) + (aux[[4, 2]] +
          2 aux[[2, 2]] (ν[1] - 1)) (aux[[3, 1]] +
          aux[[1, 1]] (2 ν[1] - 1)) -
       2 aux[[2,
         1]] (ν[1] - 1) (aux[[3, 2]] +
          aux[[1, 2]] (2 ν[1] - 1))) /. Reglenu);

(* Terms depending on the form of the punch *)

Switch[ punch[[1]],

     "Flat", f = 1 &;
             aovergamma := 1;
             Pstar := P / 2 /d /a,

     "Cone", f = 1 - punch[[2]] 0.5 Pi aa # / d &;
             aovergamma := 2 d / punch[[2]] / Pi;
             Pstar := P punch[[2]] Pi /2 /d^2,
 (* punch[[2]] is the cotangent of the semi-
 apical angle of the cone, in radians! *)

     "Hertz", f = 1 - aa^2 #^2 punch[[2]]/ d  &;
             aovergamma := (d/punch[[2]])^0.5;
             Pstar := 0.75 P (punch[[2]]/d^3)^0.5,
 (* punch[[2]] is the inverse of the radius of the sphere *)

     "Rock", f = 1 - aa^2 #^2 punch[[2]] / punch[[3]] /d
                     (1 + If [ aa # - punch[[3]] < 0, 0,
        punch[[3]]/aa/# ArcCos[ punch[[3]] / aa / # ]
                       - ( 1 - (punch[[3]] / aa / # )^2 )^0.5  ] ) &;
              aovergamma := 2  d/punch[[2]]/Pi;
              Pstar := P  punch[[2]] Pi /2 /d^2,

      _,
     Print["Bad word for Punch, i.e Flat, Cone, ... "]; Abort[]
     ];

  (* Construction of the integral equation in discretized form *)

  nequation = ndiscret[[1]]; nkernel = ndiscret[[2]];
  NLinearSolve = Compile[{{m, _Real, 2}, {f, _Real, 1}},
         LinearSolve[m, f]];

   (* F represents the shape of the indenter *)

  F[a_, d_] := Map[ f /. aa -> a ,  gp ] ;

  (* Integration of g (lambda) to get K (u) *)
  {gpK, gwK} =
   GaussStuffInterval[nkernel, {0, 5} ];

  myK =  Map[ a g[#] Cos[a u #] & , gpK ] .  gwK;
  myKf = Evaluate[myK /. u -> Plus[#1, #2] ] &;

  (* Construction of the matrix for the discretized Fredholm integral \
equation*)
  {gp, gw} = GaussStuffInterval[nequation, {0, 1}];

  matf =  IdentityMatrix[nequation] -
          1/ Pi   Inner[ Times,
                Outer[ myKf , gp   , - gp  ] , gw ,
                List] -
          1/ Pi    Inner[ Times,
                Outer[ myKf ,  gp  ,   gp ], gw ,
                List];
  mat = Compile[ {{a, _Real}}, Evaluate[ matf] ];

  H[a_, d_] :=
        NLinearSolve[ mat[ a ] , F[a, d ] ];

  (*Choosing of a contact radius a.
  Resolution of the integral equation.
  Verification of the value of the contact radius (sigmazz=
  0 in a for all indenters except the flat punch) *)

  (* For every indentation depth given in "depl" *)

  num = Length[depl];


  If[punch[[1]] == "Flat",
   (
    `
     result = Timing[
        Table[
         d = depl[[j]];
                           γ = 1; a = punch[[2]];

         (* P is found for each d by integration *)
         {d,
          4*μ[1] / (1 - ν[1]) a d H[a, d] . gw /. Reglenu /.
           Reglemu}
         , {j, 1, num}]];
     ),
          (
          result = Timing[
        Table[
         d = depl[[j]];

         (* Finding of the contact radius : resolution of H[a,d][[
         nequation]]==0 through dichotomic search *)

         a0 = 0.8;
         b0 = 1.2;

         While[Abs[H[a0 aovergamma, d][[nequation]]] > preci,
          c = (a0 + b0)/2;
          If[H[c aovergamma, d][[nequation]] > 0, a0 = c, b0 = c]];

         γ = a0; a = γ aovergamma;
         (* P is found for each d by integration *)

          P = 4*μ[1] / (1 - ν[1]) a d H[a, d] . gw /. Reglenu /. Reglemu;

         {depl[[j]],P,a,γ,Pstar}
         , {j, 1, num}]];
     )
    ];
   result[[2]]
   ]
```

```
End[ ];          (* "MuliLayerIndentation`Private`" *)

Attributes[GaussStuff] = {ReadProtected};
Attributes[GaussStuffInterval] = {ReadProtected};
Attributes[GaussInterval] = {ReadProtected};
Attributes[GaussIntegrate] = {ReadProtected};
Attributes[LagrangeInterpolation] = {ReadProtected};
Attributes[OneLayerIndentation] = {ReadProtected};
Attributes[MultiLayerIndentation] = {ReadProtected};
Attributes[NLinearSolve] = {ReadProtected};

(*
Protect[GaussStuff,GaussStuffInterval,GaussInterval,GaussIntegrate,LagrangeInterpolation];
*)

Protect[OneLayerIndentation, MultiLayerIndentation];
```

```
EndPackage[ ]
```