In[ ]:=
```
(* Initialisation *)
(* Evaluate before starting writing "real code" *)
(* Usage e.g.: "ld [Spacekey]" becomes "⊨",
so writing "a ld 5" turns into "a⊨5" *)
SetOptions[EvaluationNotebook[],
            InputAutoReplacements → {(* special AceGen assignment operators: *)
   "ld" → "⊨", "ls" → "⊢", "rd" → "⥤", "rs" → "⊣",
                                     (* brackets and symbols: *) "dbl" → "⟦",
   "dbr" → "⟧", "lcb" → "{", "rcb" → "}", "lsb" → "[", "rsb" → "]", "->" → "→",
                                     (* shortcuts for
    starting/ending a comment block: *) "co" → "(*", "cc" → "*)"
                                    }
          ]
(* Output the current time,
so we know when AceGen has been executed the last time *)
Now
```

Out[ ]=
```
Thu 20 Jun 2024 11:22:54 GMT+2
```

In[ ]:=
```
(* Clear all old variables initially to have a fresh start *)
ClearAll["Global`*"]
(* Start AceGen *)
<< AceGen`;
(* Name of the to be created subroutine/function
 in the below specified programming language *)
NAME = "HyperElasticity";
(* Name of the AceGen session "NAME",
specify the programming language "Language"={C++,Matlab,Fortran,...},
and the execution mode "Mode"={Optimal,Prototype,Debug,Plain} *)
(* @note Changing the output programming language can be very simple here,
so feel free to take advantage of all the available languages. For instance,
first export to a Matlab-code,
because you can quickly and easily debug and check the code and its output. *)
SMSInitialize[NAME, "Language" → "Matlab", "Mode" → "Optimal"];
(* Start a module, which represents the to be created function,
with name "NAME" and the specified input and output arguments *)
SMSModule[NAME, Real[F$$[3, 3], listOfMaterialParameters$$[2],
    CauchyStress$$[3, 3], Tangent$$[3, 3, 3, 3]],
           "Input" → {F$$, listOfMaterialParameters$$},
           "Output" → {CauchyStress$$, Tangent$$}
          ];
(* Input declaration by copying AceGen variables to Mathematica variables *)
F ⩵ SMSReal[Table[F$$[i, j], {i, 3}, {j, 3}]];
listOfMaterialParameters ⩵ SMSReal[Table[listOfMaterialParameters$$[i], {i, 2}]];
```

In[ ]:=
```
(* Extract the bulk modulus kappa and the shear
 modulus mu from the list of material parameters *)
(* @note You cannot use the default underscore "_" in variable names,
instead the special character "[Esc]_[Esc]" is used. *)
(* @note You can also use greek/etc. symbols like "α" for variables as
 in classical Mathematica notebooks, e.g. "[Esc]kappa[Esc]" for κ. *)
κ ⩵ bulkMod_kappa ⩵ listOfMaterialParameters ⟦1⟧;
μ ⩵ shearMod_mu ⩵ listOfMaterialParameters ⟦2⟧;
λ ⩵ Lame_lambda ⩵ bulkMod_kappa - 2/3 shearMod_mu;
(* Compute the right Cauchy-
 Green (RCG) strain tensor "RCG_C" from the deformation gradient and "freeze" it,
because we later take the derivative with respect to this variable *)
(* @note If you directly take derivatives with respect
 to an input argument like "F", the "freeze" is not required,
because "F" already has a a so-called "unique signature" by using SMSReal *)
SMSFreeze[RCG_C, Fᵀ.F, "Symmetric" → True];
```

```
In[ ]:= (* Compute the strain energy density, here for example a neo-Hookean energy *)
(* @note Now that we froze RCG_C and take all our derivatives with respect
 to RCG_C, we also need to express our quantities in RCG_C. For instance,
you now cannot compute J⊨SMSDet[F] as AceGen, due to the Freeze,
does not see that RCG_C=RCG_C(F). Express J instead as a function of RCG_C. *)
(* Thus is can be sensible to use a SetDelayed ":=",
even though the default version can still see the deformation gradient F *)
Psi_NH[RCG_C_] := (
   J ⊨ SMSSqrt[SMSDet[RCG_C]];
   Psi_NH ⊨ shearMod_mu/2 (Tr[RCG_C] - 3) - shearMod_mu Log[J] + Lame_lambda/2 (Log[J])^2;
   Return[Psi_NH];
  );
(* Compute the second Piola-
 Kirchhoff (PK2) stress tensor "PK2_S" derived from the energy *)
PK2_S ⊨ 2 SMSD[Psi_NH[RCG_C], RCG_C, "Symmetric" → True];
(* Push-forward "PK2_S" to Cauchy stress "Cauchy_sigma" *)
Cauchy_sigma ⊨ 1/SMSDet[F] F . PK2_S . F^T;

In[ ]:= (* Export the output variables by copying
 the Mathematica variables to AceGen variables *)
(* Output/Export the stress vector "Cauchy_sigma" as variable CauchyStress$$ *)
SMSExport[Cauchy_sigma , Table[CauchyStress$$[i, j], {i, 3}, {j, 3}]];

(* Compute the Lagrangian elasticity tangent *)
DPK2_DE ⊨ 2 SMSD[PK2_S, RCG_C, "Symmetric" → True];
(* Output/Export the derivative as a fourth-order tensor *)
(* @note Symmetry of the derivative is numerically not ensured,
therefore it is recommended to either use the option
 "Symmetric" or take derivatives in Voigt/Nye/Vector notation *)
SMSExport[DPK2_DE, Table[Tangent$$[i, j, k, l], {i, 3}, {j, 3}, {k, 3}, {l, 3}]];
```

In[ ]:=
```
(* Debugging/Verification *)
(* Output the stress tensor to the screen *)
SMSPrintMessage[NAME <> "<< Cauchy stress AceGen=", Cauchy_sigma];
(* Compute analytical Cauchy stress tensor *)
Cauchy_sigma_ay := μ/SMSDet[F] (F.F^T - IdentityMatrix[3]) + λ Log[SMSDet[F]]/SMSDet[F] IdentityMatrix[3];
(* SMSPrintMessage[NAME<>"<< Cauchy stress analytical=",Cauchy_sigma_ay]; *)
SMSPrintMessage[NAME <> "<< Cauchy stress relative error= ",
  1/Cauchy_sigma[[1, 1]] Sum[(Cauchy_sigma - Cauchy_sigma_ay)[[i, j]]^2, {i, 3}, {j, 3}]];
(* Compute the analytical tangent and compare it to the AceGen-Output *)
RCGi := SMSInverse[RCG_C];
dRCGi_dRCG :=
  -1/2 Table[RCGi[[i, k]] RCGi[[j, l]] + RCGi[[i, l]] RCGi[[j, k]], {i, 3}, {j, 3}, {k, 3}, {l, 3}];
DPK2_DE_ay := (-2 μ + 2 λ Log[SMSDet[F]]) dRCGi_dRCG + λ RCGi ⊗ RCGi;

SMSPrintMessage[NAME <> "<< relative error in tangent=", 1/DPK2_DE[[1, 1, 1, 1]]
   Sum[(DPK2_DE - DPK2_DE_ay)[[i, j, k, l]]^2, {i, 3}, {j, 3}, {k, 3}, {l, 3}]];
```

In[ ]:=
```
(* Output the time at the end of the execution *)
Now
(* Write output file containing all the
 above defined functions introduced by SMSModule *)
(* Create output file named "NAME", '"LocalAuxiliaryVariables " → True' is a command to exclude the AceGen internal array "v" from the list of input and output arguments of the created subroutine *)
SMSWrite[NAME, "LocalAuxiliaryVariables " → True];
(* Print the content of the just created file on screen (sensible only for small file sizes) *)
FilePrint[StringJoin[NAME, Which[SMSLanguage == "Fortran", ".f",
                                 SMSLanguage == "Matlab", ".m",
                                 SMSLanguage == "C++", ".cpp",
                                 SMSLanguage == "C", ".c"
                                ]
                    ]
          ]
```

Out[ ]= Thu 20 Jun 2024 11:23:01 GMT+2

File : HyperElasticity .m **Size :** 8278 **Time :** 5

Method	HyperElasticity
No.Formulae	127
No.Leafs	2891

```
%************************************************************
%* AceGen     7.505 Linux (16 Aug 22)                       *
%*            Co. J. Korelc  2020           20 Jun 24 11:23:03  *
%************************************************************
% User      : Full professional version
% Notebook  : AceGen-HyperElasticity
% Evaluation time                  : 5 s     Mode   : Optimal
% Number of formulae               : 127     Method: Automatic
% Subroutine                       : HyperElasticity  size: 2891
% Total size of Mathematica  code  : 2891 subexpressions
% Total size of Matlab code        : 7367 bytes

%*********************   F U N C T I O N ************************
function[CauchyStress,Tangent]=HyperElasticity(F,listOfMaterialParameters);
persistent v;
if size(v)<328
  v=zeros(328,'double');
end;
v(1)=F(1,1);
v(121)=(v(1)*v(1));
v(2)=F(1,2);
v(122)=(v(2)*v(2));
v(3)=F(1,3);
v(123)=(v(3)*v(3));
v(4)=F(2,1);
v(128)=(v(4)*v(4));
v(5)=F(2,2);
v(129)=(v(5)*v(5));
v(6)=F(2,3);
v(130)=(v(6)*v(6));
v(7)=F(3,1);
v(134)=(v(7)*v(7));
v(8)=F(3,2);
v(135)=(v(8)*v(8));
v(9)=F(3,3);
v(209)=v(8)*v(9);
v(136)=(v(9)*v(9));
v(124)=v(3)*(-(v(5)*v(7))+v(4)*v(8))+v(2)*(v(6)*v(7)-v(4)*v(9))+v(1)*(-(v(6)*v(8))+v(5)*v(9));
v(11)=listOfMaterialParameters(2);
v(211)=v(11)/v(124);
v(29)=v(11)/2e0;
v(12)=listOfMaterialParameters(1)+(-2e0/3e0)*v(11);
v(199)=v(12)*log(v(124));
v(175)=-2e0*v(11)+2e0*v(199);
v(217)=v(175)/2e0;
v(213)=v(12)-v(175);
v(131)=v(199)/v(124);
v(13)=v(121)+v(128)+v(134);
v(14)=v(1)*v(2)+v(4)*v(5)+v(7)*v(8);
```

```
v(15)=v(1)*v(3)+v(4)*v(6)+v(7)*v(9);
v(212)=v(14)*v(15);
v(16)=v(122)+v(129)+v(135);
v(200)=-(v(15)*v(16));
v(52)=-(v(14)*v(14))+v(13)*v(16);
v(17)=v(209)+v(2)*v(3)+v(5)*v(6);
v(201)=v(14)*v(17);
v(143)=v(200)+v(201);
v(49)=2e0*(v(200)+v(201));
v(47)=-2e0*v(17);
v(51)=2e0*v(212)+v(13)*v(47);
v(18)=v(123)+v(130)+v(136);
v(202)=v(14)*v(18);
v(141)=-(v(15)*v(17))+v(202);
v(50)=-(v(15)*v(15))+v(13)*v(18);
v(48)=-2e0*v(202)-v(15)*v(47);
v(23)=-(v(17)*v(17))+v(16)*v(18);
v(24)=-(v(14)*v(141))+v(143)*v(15)+v(13)*v(23);
v(53)=sqrt(v(24));
v(210)=v(12)/(v(24)*Power(v(53),2));
v(207)=(-v(11)+v(12)*log(v(53)))/2e0;
v(74)=-(v(207)/Power(v(24),2));
v(203)=v(210)/4e0+v(74);
v(79)=v(203)*v(52);
v(78)=v(203)*v(51);
v(204)=2e0*v(78);
v(111)=v(204)*v(52);
v(103)=v(204)*v(50);
v(77)=v(203)*v(50);
v(76)=v(203)*v(49);
v(205)=2e0*v(76);
v(110)=v(205)*v(52);
v(82)=v(205)*v(23);
v(75)=v(203)*v(48);
v(206)=2e0*v(75);
v(100)=v(206)*v(50);
v(81)=v(206)*v(23);
v(26)=v(207)/v(24);
v(208)=-2e0*v(26);
v(107)=v(13)*v(208)+v(51)*v(78);
v(97)=v(14)*v(208);
v(98)=v(49)*v(78)-v(97);
v(95)=-(v(15)*v(208));
v(101)=2e0*v(49)*v(77)-2e0*v(95);
v(94)=v(16)*v(208)+v(49)*v(76);
v(109)=2e0*(v(48)*v(79)+v(97));
v(91)=v(48)*v(78)+v(95);
v(88)=v(26)*v(47);
v(89)=v(48)*v(76)-v(88);
v(87)=v(18)*v(208)+v(48)*v(75);
v(84)=2e0*(v(23)*v(78)+v(88));
v(25)=2e0*(v(23)*v(26)+v(29));
v(27)=v(26)*v(48);
v(28)=v(26)*v(49);
v(41)=v(25)*v(4)+v(27)*v(5)+v(28)*v(6);
```

```
v(35)=v(1)*v(25)+v(2)*v(27)+v(28)*v(3);
v(30)=2e0*(v(29)+v(26)*v(50));
v(31)=v(26)*v(51);
v(42)=v(27)*v(4)+v(30)*v(5)+v(31)*v(6);
v(36)=v(1)*v(27)+v(2)*v(30)+v(3)*v(31);
v(32)=2e0*(v(29)+v(26)*v(52));
v(43)=v(28)*v(4)+v(31)*v(5)+v(32)*v(6);
v(37)=v(1)*v(28)+v(2)*v(31)+v(3)*v(32);
v(33)=(v(1)*v(35)+v(2)*v(36)+v(3)*v(37))/v(124);
v(38)=(v(35)*v(4)+v(36)*v(5)+v(37)*v(6))/v(124);
v(39)=(v(35)*v(7)+v(36)*v(8)+v(37)*v(9))/v(124);
v(40)=(v(4)*v(41)+v(42)*v(5)+v(43)*v(6))/v(124);
v(44)=(v(41)*v(7)+v(42)*v(8)+v(43)*v(9))/v(124);
v(45)=(v(134)*v(25)+v(135)*v(30)+2e0*v(209)*v(31)+v(136)*v(32)+2e0*v(7)*(v(27)*v(8)+v(28)*v(9)))/v     ...
 (124);
CauchyStress(1,1)=v(33);
CauchyStress(1,2)=v(38);
CauchyStress(1,3)=v(39);
CauchyStress(2,1)=v(38);
CauchyStress(2,2)=v(40);
CauchyStress(2,3)=v(44);
CauchyStress(3,1)=v(39);
CauchyStress(3,2)=v(44);
CauchyStress(3,3)=v(45);
v(113)=(v(23)*v(23))*(v(210)+4e0*v(74));
v(114)=4e0*(v(18)*v(26)+v(23)*v(77));
v(115)=4e0*(v(16)*v(26)+v(23)*v(79));
v(116)=4e0*v(50)*v(77);
v(117)=4e0*(v(13)*v(26)+v(50)*v(79));
v(118)=4e0*v(52)*v(79);
Tangent(1,1,1,1)=v(113);
Tangent(1,1,1,2)=v(81);
Tangent(1,1,1,3)=v(82);
Tangent(1,1,2,1)=v(81);
Tangent(1,1,2,2)=v(114);
Tangent(1,1,2,3)=v(84);
Tangent(1,1,3,1)=v(82);
Tangent(1,1,3,2)=v(84);
Tangent(1,1,3,3)=v(115);
Tangent(1,2,1,1)=v(81);
Tangent(1,2,1,2)=v(87);
Tangent(1,2,1,3)=v(89);
Tangent(1,2,2,1)=v(87);
Tangent(1,2,2,2)=v(100);
Tangent(1,2,2,3)=v(91);
Tangent(1,2,3,1)=v(89);
Tangent(1,2,3,2)=v(91);
Tangent(1,2,3,3)=v(109);
Tangent(1,3,1,1)=v(82);
Tangent(1,3,1,2)=v(89);
Tangent(1,3,1,3)=v(94);
Tangent(1,3,2,1)=v(89);
Tangent(1,3,2,2)=v(101);
Tangent(1,3,2,3)=v(98);
Tangent(1,3,3,1)=v(94);
```

```
Tangent(1,3,3,2)=v(98);
Tangent(1,3,3,3)=v(110);
Tangent(2,1,1,1)=v(81);
Tangent(2,1,1,2)=v(87);
Tangent(2,1,1,3)=v(89);
Tangent(2,1,2,1)=v(87);
Tangent(2,1,2,2)=v(100);
Tangent(2,1,2,3)=v(91);
Tangent(2,1,3,1)=v(89);
Tangent(2,1,3,2)=v(91);
Tangent(2,1,3,3)=v(109);
Tangent(2,2,1,1)=v(114);
Tangent(2,2,1,2)=v(100);
Tangent(2,2,1,3)=v(101);
Tangent(2,2,2,1)=v(100);
Tangent(2,2,2,2)=v(116);
Tangent(2,2,2,3)=v(103);
Tangent(2,2,3,1)=v(101);
Tangent(2,2,3,2)=v(103);
Tangent(2,2,3,3)=v(117);
Tangent(2,3,1,1)=v(84);
Tangent(2,3,1,2)=v(91);
Tangent(2,3,1,3)=v(98);
Tangent(2,3,2,1)=v(91);
Tangent(2,3,2,2)=v(103);
Tangent(2,3,2,3)=v(107);
Tangent(2,3,3,1)=v(98);
Tangent(2,3,3,2)=v(107);
Tangent(2,3,3,3)=v(111);
Tangent(3,1,1,1)=v(82);
Tangent(3,1,1,2)=v(89);
Tangent(3,1,1,3)=v(94);
Tangent(3,1,2,1)=v(89);
Tangent(3,1,2,2)=v(101);
Tangent(3,1,2,3)=v(98);
Tangent(3,1,3,1)=v(94);
Tangent(3,1,3,2)=v(98);
Tangent(3,1,3,3)=v(110);
Tangent(3,2,1,1)=v(84);
Tangent(3,2,1,2)=v(91);
Tangent(3,2,1,3)=v(98);
Tangent(3,2,2,1)=v(91);
Tangent(3,2,2,2)=v(103);
Tangent(3,2,2,3)=v(107);
Tangent(3,2,3,1)=v(98);
Tangent(3,2,3,2)=v(107);
Tangent(3,2,3,3)=v(111);
Tangent(3,3,1,1)=v(115);
Tangent(3,3,1,2)=v(109);
Tangent(3,3,1,3)=v(110);
Tangent(3,3,2,1)=v(109);
Tangent(3,3,2,2)=v(117);
Tangent(3,3,2,3)=v(111);
Tangent(3,3,3,1)=v(110);
Tangent(3,3,3,2)=v(111);
```

```
Tangent(3,3,3,3)=v(118);
disp(sprintf("\n%s  %f %f %f %f %f %f %f %f %f ","HyperElasticity<<  Cauchy stress AceGen="
 (38),v(39),v(38),v(40),v(44),v(39),v(44),v(45)));
disp(sprintf("\n%s  %f ","HyperElasticity<<  Cauchy stress relative error= ",(Power(-v(131)-(-1e0
 (121)+v(122)+v(123))*v(211)+v(33),2)+Power(-v(131)-(-1e0+v(128)+v(129)+v(130))*v(211)+v(40),2)      ...
 +Power(-v(131)-(-1e0+v(134)+v(135)+v(136))*v(211)+v(45),2))/v(33)));
v(140)=v(23)/v(24);
v(142)=-(v(141)/v(24));
v(154)=(v(142)*v(142));
v(144)=v(143)/v(24);
v(160)=(v(144)*v(144));
v(145)=v(50)/v(24);
v(214)=v(12)*v(145);
v(146)=(-(v(13)*v(17))+v(212))/v(24);
v(215)=v(140)*v(146);
v(168)=(v(146)*v(146));
v(162)=-(v(142)*v(146));
v(147)=v(52)/v(24);
v(218)=v(142)*v(147);
v(216)=v(140)*v(147);
v(152)=-(v(142)*v(144));
v(159)=-(v(144)*v(146));
disp(sprintf("\n%s  %f ","HyperElasticity<<  relative error in tangent=",(Power(v(113)-(v(140)*v(14
 )*v(213),2)+4e0*Power(v(100)-v(142)*v(145)*v(213),2)+Power(v(116)-(v(145)*v(145))*v(213),2)     ...
 +4e0*Power(v(103)-v(145)*v(146)*v(213),2)+4e0*Power(v(110)-v(144)*v(147)*v(213),2)+4e0*Power(v(111)     ...
 -v(146)*v(147)*v(213),2)+Power(v(118)-(v(147)*v(147))*v(213),2)+2e0*Power(v(114)+v(154)*v(175)-v     ...
 (140)*v(214),2)+4e0*Power(v(101)-v(162)*v(175)-v(144)*v(214),2)+2e0*Power(v(117)+v(168)*v(175)-v     ...
 (147)*v(214),2)+2e0*Power(v(115)+v(160)*v(175)-v(12)*v(216),2)+4e0*Power(v(107)-v(12)*v(168)-(-(v     ...
 (145)*v(147))-v(168))*v(217),2)+4e0*Power(v(109)-v(159)*v(175)-v(12)*v(218),2)+4e0*Power(-(v(140)*v     ...
 (142)*v(213))+v(81),2)+4e0*Power(-(v(140)*v(144)*v(213))+v(82),2)+4e0*Power(-(v(152)*v(175))-v(12     ...
 )*v(215)+v(84),2)+4e0*Power(-(v(12)*v(154))-(-(v(140)*v(145))-v(154))*v(217)+v(87),2)+8e0*Power(v     ...
 (12)*v(152)-(v(152)-v(215))*v(217)+v(89),2)+8e0*Power(v(12)*v(162)-(-(v(144)*v(145))+v(162))*v(217)     ...
 +v(91),2)+4e0*Power(-(v(12)*v(160))-(-v(160)-v(216))*v(217)+v(94),2)+8e0*Power(v(12)*v(159)-v(217)*     ...
 (v(159)-v(218))+v(98),2))/v(113)));


function [x]=SMSKDelta(i,j)

if (i==j) , x=1; else x=0; end;

end

function [x]=SMSDeltaPart(a,i,j,k)

l=round(i/j);

if (mod(i,j) ~= 0 | l>k) , x=0; else x=a(l); end;

end

function [x]=Power(a,b)

x=a^b;

end
```

```
function [x]=SMSTernaryOperator(a,b,c)
if (c) , x=a; else x=b; end;
end
end
```