Appendix C – OpenSees Code Files for Hyperelastic Material Programming

C.1 – HyperelasticMaterial.cpp

```
// File: ~/material/HyperelasticMaterial.C
//
// Written: RS
// Created: 10/2003
//
// Revision: B 02/08/04
//
// Description: This file contains the class implementation for
// a HyperelasticMaterial.
//
// What: "@(#) HyperelasticMaterial.C, revA"

//Reference File List:

#define MAT_TAG_HyperelasticMaterial 18

#include <HyperelasticMaterial.h>
#include <Vector.h>
#include <Channel.h>
#include <Information.h>
#include <OPS_Globals.h>
#include <math.h>

//Constructors for the Hyperelastic Material

HyperelasticMaterial::HyperelasticMaterial
(int tag, double A, double B, double C, double D, double L)
:UniaxialMaterial(tag,MAT_TAG_HyperelasticMaterial),
 trialStrain(0.0),
  a(A), b(B), c(C), d(D), len(L)
{
        Ttangent = (a*3)*pow(trialStrain*len,2)+(b*2)*(trialStrain*len)+(c);
        Tstress =
a*pow(trialStrain*len,3)+b*pow(trialStrain*len,2)+c*(trialStrain*len)+d;
        Cstress = 0.0;
        Cstrain = 0.0;
   Ctangent = Ttangent;
}

//Value of F must be > 0
```

```
HyperelasticMaterial::HyperelasticMaterial()
:UniaxialMaterial(0,MAT_TAG_HyperelasticMaterial),
 trialStrain(0.0),
  a(0.0), b(0.0), c(0.0), d(0.0)
{
        Ttangent = (a*3)*pow(trialStrain*len,2)+(b*2)*(trialStrain*len)+(c);
        Tstress =
a*pow(trialStrain*len,3)+b*pow(trialStrain*len,2)+c*(trialStrain*len)+d;
        Cstress = 0.0;
        Cstrain = 0.0;
   Ctangent = Ttangent;
}

HyperelasticMaterial::~HyperelasticMaterial()
{
  // does nothing since no memory is allocated to this uniaxialMaterial
}

//Determination of Material State

int
HyperelasticMaterial::setTrialStrain(double strain, double strainrate)
{

        trialStrain     = strain;
        double dStrain = trialStrain - Cstrain;

        strain *= len;

        if (trialStrain > 0) {
                Ttangent = (a*3)*pow(fabs(strain),2)+(b*2)*(fabs(strain))+(c);
                Tstress = a*pow(fabs(strain),3)+b*pow(fabs(strain),2)+c*(fabs(strain))+d;
        }
        else {
                Ttangent = (a*3)*pow(fabs(strain),2)+(b*2)*(fabs(strain))+(c);
                Tstress = -
1.0*(a*pow(fabs(strain),3)+b*pow(fabs(strain),2)+c*(fabs(strain))+d);
        }

  // Determine change in strain from last converged state



   return 0;
}
```

```
int
HyperelasticMaterial::setTrial(double strain, double &stress, double &tangent, double
strainrate)
{
   trialStrain     = strain;
          double dStrain = trialStrain - Cstrain;

          strain *= len;

          if (trialStrain > 0) {
                    Ttangent = (a*3)*pow(fabs(strain),2)+(b*2)*(fabs(strain))+(c);
                    Tstress = a*pow(fabs(strain),3)+b*pow(fabs(strain),2)+c*(fabs(strain))+d;
          }
          else {
                    Ttangent = (a*3)*pow(fabs(strain),2)+(b*2)*(fabs(strain))+(c);
                    Tstress = -
1.0*(a*pow(fabs(strain),3)+b*pow(fabs(strain),2)+c*(fabs(strain))+d);
          }

   stress = Tstress;
   tangent = Ttangent;

   return 0;
}

double
HyperelasticMaterial::getStress(void)
{
   return Tstress;

}

double
HyperelasticMaterial::getStrain(void)
{
          return trialStrain;

}

double
HyperelasticMaterial::getTangent(void)
{
          return Ttangent;
```

```
}


int
HyperelasticMaterial::commitState(void)
{
   Ctangent = Ttangent;
        Cstress = Tstress;
        Cstrain = trialStrain;
        return 0;
}


int
HyperelasticMaterial::revertToLastCommit(void)
{
   return 0;
        //Anything to do here?  See hysteretic material
}


int
HyperelasticMaterial::revertToStart(void)
{
   Ctangent = d;
        Cstress = 0.0;
        Cstrain = 0.0;
   return 0;
}

UniaxialMaterial *
HyperelasticMaterial::getCopy(void)
{
   HyperelasticMaterial *theCopy = new HyperelasticMaterial(this-
>getTag(),a,b,c,d,len);
   // Copy trial state variables
   theCopy->trialStrain = trialStrain;
   theCopy->Tstress = Tstress;
   theCopy->Ttangent = Ttangent;
   //Copy the committed state variables
   theCopy->Cstrain = Cstrain;
   theCopy->Cstress = Cstress;
   theCopy->Ctangent = Ctangent;

   return theCopy;
}
```

```
int
HyperelasticMaterial::sendSelf(int cTag, Channel &theChannel)
{
  int res = 0;
  static Vector data(12);
  data(0) = this->getTag();
  data(1) = a;
  data(2) = b;
  data(3) = c;
  data(4) = d;
  data(5) = Cstress;
  data(6) = Cstrain;
  data(7) = Ctangent;
  data(8) = Tstress;
  data(9) = trialStrain;
  data(10) = Ttangent;
  data(11) = len;

  res = theChannel.sendVector(this->getDbTag(), cTag, data);
  if (res < 0)
    opserr << "HyperelasticMaterial::sendSelf() - failed to send data\n";

  return res;
}

int
HyperelasticMaterial::recvSelf(int cTag, Channel &theChannel,
                               FEM_ObjectBroker &theBroker)
{
  int res = 0;
  static Vector data(12);
  res = theChannel.recvVector(this->getDbTag(), cTag, data);

  if (res < 0) {
     opserr << "HyperelasticMaterial::recvSelf() - failed to receive data\n";
     this->setTag(0);
  }
  else {
    this->setTag(data(0));
          a = data(1);
          b = data(2);
          c = data(3);
          d = data(4);
          Cstress = data(5);
          Cstrain = data(6);
```

```
        Ctangent = data(7);
        Tstress = data(8);
        trialStrain = data(9);
        Ttangent = data(10);
        len = data(11);

  }

  return res;
}

void
HyperelasticMaterial::Print(OPS_Stream &s, int flag)
{
    s << "Elastic tag: " << this->getTag() << endln;
    s << "  a: " << a << " b: " << b << " c: " << c << " d: " << d <<  endln;
}

int
HyperelasticMaterial::setParameter(const char **argv, int argc, Information &info)
{
        if (strcmp(argv[0],"a") == 0) {
                info.theType = DoubleType;
                return 1;
        }
        else if (strcmp(argv[0],"b") == 0) {
                info.theType = DoubleType;
                return 2;
        }
        else if (strcmp(argv[0],"c") == 0) {
                info.theType = DoubleType;
                return 3;
        }
        else if (strcmp(argv[0],"d") == 0) {
                info.theType = DoubleType;
                return 4;
        }
        else
                return -1;
}

int
HyperelasticMaterial::updateParameter(int parameterID, Information &info)
{
        switch(parameterID) {
        case -1:
```

```
                return -1;
        case 1:
                a = info.theDouble;
                return 0;
        case 2:
                b = info.theDouble;
                return 0;
        case 3:
                c = info.theDouble;
                return 0;
        case 4:
                d = info.theDouble;
                return 0;
        default:
                return -1;
        }
}
```

C.2 – HyperelasticMaterial.h

```
// File: ~/material/HyperelasticMaterial.h
//
// Written: RS
// Created: 10/2003
// Revision: A
//
// Description: This file contains the class definition for
// HyperelasticMaterial. HyperelasticMaterial provides the abstraction
// of a hyperelastic uniaxial material,
// i.e. stress = E*strain + eta*strainrate where E is found from the
// equation for force-deformation: F = Ax^3 + Bx^2 + Cx + D where Modulus (E) =
dF/dx
//
// What: "@(#) HyperelasticMaterial.h, revA"


#include <UniaxialMaterial.h>

class HyperelasticMaterial : public UniaxialMaterial
{
  public:
    HyperelasticMaterial(int tag, double A, double B, double C, double D, double L);
    HyperelasticMaterial();
    ~HyperelasticMaterial();

    int setTrialStrain(double strain, double strainrate = 0.0);
    int setTrial(double strain, double &stress, double &tangent, double strainrate = 0.0);
    double getStrain(void);
    double getStress(void);
    double getTangent(void);
    double getInitialTangent(void) {return d;};

    int commitState(void);
    int revertToLastCommit(void);
    int revertToStart(void);

    UniaxialMaterial *getCopy(void);

    int sendSelf(int commitTag, Channel &theChannel);
    int recvSelf(int commitTag, Channel &theChannel,
                 FEM_ObjectBroker &theBroker);

    void Print(OPS_Stream &s, int flag =0);
```

```
    int setParameter(const char **argv, int argc, Information &info);
    int updateParameter(int parameterID, Information &info);

  protected:

  private:
    double trialStrain;
    double a;
          double b;
          double c;
          double d;
          double len;
          double Cstress;
          double Cstrain;
    double Ctangent;
          double Tstress;
          double Ttangent;
};


#endif
```

C.3 – TclModelBuilderUniaxialMaterialCommand.cpp

```
//New Hyperelastic Material Section***************

   if (strcmp(argv[1],"Hyperelastic") == 0) {
        if (argc < 8) {
           opserr << "WARNING invalid number of arguments\n";
           printCommand(argc,argv);
           opserr << "Want: uniaxialMaterial Hyperelastic tag? A? B? C? D? L?" <<
endln;
           return TCL_ERROR;
        }

        int tag;
        double A, B, C, D, L;

        if (Tcl_GetInt(interp, argv[2], &tag) != TCL_OK) {
           opserr << "WARNING invalid uniaxialMaterial Hyperelastic tag" << endln;
           return TCL_ERROR;
        }

        if (Tcl_GetDouble(interp, argv[3], &A) != TCL_OK) {
           opserr << "WARNING invalid A";
           opserr << "uniaxiaMaterial Hyperelastic: " << tag << endln;
           return TCL_ERROR;
        }

        if (Tcl_GetDouble(interp, argv[4], &B) != TCL_OK) {
           opserr << "WARNING invalid B";
           opserr << "uniaxiaMaterial Hyperelastic: " << tag << endln;
           return TCL_ERROR;
        }

        if (Tcl_GetDouble(interp, argv[5], &C) != TCL_OK) {
           opserr << "WARNING invalid C";
           opserr << "uniaxiaMaterial Hyperelastic: " << tag << endln;
           return TCL_ERROR;
        }

        if (Tcl_GetDouble(interp, argv[6], &D) != TCL_OK) {
           opserr << "WARNING invalid D";
           opserr << "uniaxiaMaterial Hyperelastic: " << tag << endln;
           return TCL_ERROR;
        }
        if (argv[6]<0) {
           opserr << "WARNING invalid D";
```

```
        opserr << "uniaxiaMaterial Hyperelastic: " << tag << endln;
        return TCL_ERROR;
    }

    if (Tcl_GetDouble(interp, argv[7], &L) != TCL_OK) {
        opserr << "WARNING invalid L";
        opserr << "uniaxiaMaterial Hyperelastic: " << tag << endln;
        return TCL_ERROR;
    }

    // Parsing was successful, allocate the material
    theMaterial = new HyperelasticMaterial(tag, A, B, C, D, L);
}
```