```cpp
// -*- C++ -*-
#ifndef MATERIALMODEL_DERIVATIVETEST_H
#define MATERIALMODEL_DERIVATIVETEST_H

#include "/src/Definitions.h"
#include "PJ2Utilities.h"

namespace MaterialModels {

  template <class MaterialModel>
  bool
  testMaterialModelDerivatives(const MaterialModel & model) {

    // Typedefs are created through copying from the existing ones in the MaterialModel
    typedef typename MaterialModel::DisplacementGradient DisplacementGradient;
    typedef typename MaterialModel::Stress               Stress;
    typedef typename MaterialModel::TangentMatrix        TangentMatrix;

    DisplacementGradient displacementGradient
      = DisplacementGradient::Random();
    unsigned int numberOfDisplacementGradientComponents
      = displacementGradient.size();

    //TODO: compute the exact energy, stresses and tangent matrix using the input object "model"
    double        energy  = model.computeEnergy(displacementGradient);
    Stress        stress = model.computeStress(displacementGradient);
    TangentMatrix tangent = model.computeTangentMatrix(displacementGradient);

    cout << "[MMT] Materials::testMaterialModelDerivatives started: " << endl;

    //TODO: define the perturbation and tolerance values
    double perturbation = 0.000001;
    double tolerance    = 0.0001;



    // ===================================
    // === TEST : STRESS - ANA vs. NUM ===
    // ===================================

    Stress numericalStresses = Stress::Zero();
    //TODO: recompute stresses from the material model energy by taking numerical derivatives
    for (   unsigned int index_I = 0; index_I < numberOfDisplacementGradientComponents; index_I++ ){


      // Evaluate perturbated energy
      // Add perturbation
      displacementGradient(index_I,0) += perturbation;

      // Evaluate energy of perturbed displacement state
      double energyOfPertubedDisplacementState =
      model.computeEnergy(displacementGradient);

      // Remove perturbation
      displacementGradient(index_I,0) -= perturbation;

      // Evaluate numerical stress approximation

      numericalStresses(index_I,0) = (energyOfPertubedDisplacementState - energy) / perturbation;


      printf("[MMT] Stress (Ana): %8.5f | Stress (Num): %8.5f\n",
             stress           (index_I,0),
             numericalStresses(index_I,0));
    }

    // TODO: compute Frobenius norm
    double errorStresses = sqrt((numericalStresses - stress).dot(numericalStresses -
```

```
        stress)) / sqrt(stress.dot(stress));
68      cout << "[MMT] Error of method computeStress = " << errorStresses << endl;
69
70
71
72      // ====================================
73      // === TEST : TANGENT - ANA vs. NUM ===
74      // ====================================
75
76      TangentMatrix numericalTangent = TangentMatrix::Zero();
77      //TODO: recompute tangent matrix from the material model stresses by taking
        numerical derivatives
78      for (   unsigned int index_I = 0; index_I <
        numberOfDisplacementGradientComponents; index_I++ ){
79
80        // Add perturbation
81        displacementGradient(index_I,0) += perturbation;
82
83        // Evaluate perturbated stress
84        Stress stressOfPertubedDisplacementState = Stress::Zero();
85        stressOfPertubedDisplacementState = model.computeStress(displacementGradient);
86
87        // Remove perturbation
88        displacementGradient(index_I,0) -= perturbation;
89
90        // Evaluate numerical Tangent approximation
91
92        for ( unsigned int index_J = 0;
93          index_J < numberOfDisplacementGradientComponents;
94          index_J++){
95
96          numericalTangent(index_J,index_I) =
            (stressOfPertubedDisplacementState(index_J,0) - stress(index_J,0)) /
            perturbation;
97
98
99          printf("[MMT] (%d,%d) Tangent (Ana): %8.5f | Tangent (Num): %8.5f\n",
100                 index_I,
101                 index_J,
102                 tangent         (index_J,index_I),
103                 numericalTangent(index_J,index_I) );
104       }
105
106     }
107
108     //TODO: compute error between analytical and numerical tangent matrices
109     //double errorTangent = (tangent - numericalTangent).norm() / tangent.norm();
110     //double errorTangent = (tangent - numericalTangent).determinant() /
        tangent.determinant();
111     Matrix<double, 9, 9> tangentMinusNumericalTangent = tangent - numericalTangent;
112     double tangentMinusNumericalTangentDotTangentMinusNumericalTangent = 0.0;
113     double tangentDotTangent = 0.0;
114     for (int i = 0; i<9; i++){
115       for (int j=0; j<9; j++){
116         tangentMinusNumericalTangentDotTangentMinusNumericalTangent +=
            tangentMinusNumericalTangent(i,j)*tangentMinusNumericalTangent(i,j);
117         tangentDotTangent += tangent(i,j)*tangent(i,j);
118       }
119     }
120     double normOfTangentMinusNumericalTangent =
        sqrt(tangentMinusNumericalTangentDotTangentMinusNumericalTangent);
121     double normOfTangent =sqrt(tangentDotTangent);
122     double errorTangent = normOfTangentMinusNumericalTangent / normOfTangent;
123     cout << "[MMT] Error of method computeTangentMatrix = " << errorTangent << endl;
124
125
126
127     // ============================
128     // === FINAL OUTPUT & RETURN ===
129     // ============================
130
131     if (errorStresses >= tolerance || errorTangent >= tolerance) {
132       cout << "[MMT] ============================================" << endl;
```

```
133         cout << "[MMT] Warning: material model derivatives test failed." << endl <<
            endl;
134         cout << "[MMT] ================================================" << endl;
135         return false;
136       }
137       /*if (errorStresses >= tolerance ) {
138         cout << "[MMT] ================================================" << endl;
139         cout << "[MMT] Warning: material model stress derivatives test failed." <<
            endl << endl;
140         cout << "[MMT] ================================================" << endl;
141         return false;
142       }
143       if (errorTangent >= tolerance ) {
144         cout << "[MMT] ================================================" << endl;
145         cout << "[MMT] Warning: material model Tangent derivatives test failed." <<
            endl << endl;
146         cout << "[MMT] ================================================" << endl;
147         return false;
148       }*/
149       else {
150         cout << endl << "[MMT] ======================================" << endl;
151         cout << "[MMT] Material model derivatives test passed." << endl;
152         cout << "[MMT] ======================================" << endl << endl;
153         return true;
154       }
155
156       ignoreUnusedVariables(energy,perturbation);
157
158       return true;
159     }
160
161
162
163   }
164
165   #endif  // MATERIALMODEL_DERIVATIVETEST_H
166
```