UTILIZATION OF COMSOL MULTIPHYSICS JAVA API FOR THE INTEGRATION OF **COMPOSITE MATERIAL** MODULE WITH A **CUSTOMIZED USER INTERFACE**

Authors:
Dr. Venkateswaran P
Siemens Technology & Services Private Limited
Bangalore 560100
Venkateswaran.p@siemens.com

Mr. Rishabh Malav
Indian Institute of Technology, Madras

COMSOL CONFERENCE 2014 BANGALORE
Excerpt from the Proceedings of the 2014 COMSOL Conference in Bangalore

INTRODUCTION

Composite Materials

- Composites are made up of individual materials referred to as constituent materials
- Two main categories of constituent materials: Matrix and Fiber
- Some Advantages: Light Weight, High Strength, Durability, Strength Related to Weight, Corrosion Resistance and Design Flexibility

Need to integrate with COMSOL

- COMSOL Multiphysics is capable of modeling and simulation of any physics-based system
- Adding a user-friendly interface to integrate Composite Material properties will further enhance its capabilities
- Use of Composite Materials in various industries such as aircraft, automotive, etc is rapidly getting acceptance
- So, in the process of modeling and simulation, many a times user wants to use Composite Materials properties in the analysis

METHODOLOGY

- COMSOL API (Application Programming Interface) which is an interface based on Java, is used to develop custom application based on COMSOL
- An initial GUI is built based on a model of a simple 'Rectangular Block', having a point force acting on the edge
- Final GUI is built based on a model of 'Mechanical Part', having a combination of forces and moments acting on it
- Halpin-Tsai Model, a semi-empirical model chosen based on its accuracy and appropriateness, is used to develop algorithm to calculate Composite Material properties

HALPIN-TSAI MODEL

Halpin-Tsai equations are the handy forms of Hill's generalized self-consistent model results with engineering approximations to make them suitable for the designing of composite materials.

$$E_1 = E_f V_f + E_m V_m$$

$$\nu_{12} = \nu_f V_f + \nu_m V_m$$

$$\frac{M}{M_m} = \frac{1 + \varepsilon\eta V_f}{1 - \eta V_f}$$

$$\eta = \frac{\frac{M_f}{M_m} - 1}{\frac{M_f}{M_m} + \varepsilon}$$

where,

M = composite material modulus E_{22}, G_{12} or v_{23}

M_f = composite material modulus E_f, G_f or v_f

M_m = composite material modulus E_m, G_m, or v_m

PROGRAM ARCHITECTURE

```
import java.awt.BorderLayout;

class RectModelDemo implements ActionListener {
    private JFrame frame;
    private JFrame frame1
    private Model model;

    private JButton solveButton;
    private JButton calculateButton;

    private NumberFormat percentFormat;
    private JTextField editV;
    private JTextField editE1;
    private JTextField editE2;
```

Declaration of Class and Parameters

```
static String[] compProperty = new String[10];
public static void main(String[] args) {
    RectModelDemo demo = new RectModelDemo();
    demo.init();
    demo.start();
}
public void init() {
    ModelUtil.initStandalone(true);

}
public void start() {
    lookandfeel();
    frame= new JFrame("Model taken from COMSOL - GUI");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(1200,900);
    GUIPanel mainPanel = new GUIPanel();
```

Basic methods: Main, init, start

PROGRAM ARCHITECTURE

```
    frame.setJMenuBar(menubar);
    frame.setVisible(true);
    }

public void solve(){
    model.physics("solid").feature("lemm1").set("Evector_mat", "userdef");
    model.physics("solid").feature("lemm1").set("rho_mat", "userdef");
    model.physics("solid").feature("lemm1")
         .set("nuvector", new String[][]{{compProp7}, {compProp8}, {compProp9}});
    model.physics("solid").feature("lemm1")
         .set("Gvector", new String[][]{{compProp4}, {compProp5}, {compProp6}});
    model.physics("solid").feature("lemm1").set("rho", compProp10);
    model.physics("solid").feature("lemm1")
         .set("Evector", new String[][]{{compProp1}, {compProp2}, {compProp3}});
    model.physics("solid").feature("lemm1").set("Gvector_mat", "userdef");
```

'Solve' method to run model

```
            model.result("pg1").set("window", "window1");
            model.result("pg1").run();
        }
    });
}

private JPanel leftPanel(){
    MigLayout layout= new MigLayout("wrap 2");
    JPanel panel = new JPanel(layout);
    JLabel label = new JLabel("Pre-Processing");
    panel.add(label,"span, wrap 15px");
    label.setFont(new Font("Serif", Font.PLAIN, 25));
    label = new JLabel("Fiber Property:");
```

'leftPanel 'method for defining fields in input side of GUI

PROGRAM ARCHITECTURE

```
            model.result("pg4").set("window", "window1");
            model.result("pg4").run();
        }
    });
    return panel;
}
}
private JButton solveButton(){
    solveButton = new JButton("Simulate");
    model= RectModel.run();
    solveButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        solve();
    }
    });
    return solveButton;
}

private void sPlot(){
    model.result().dataset("cpl1")
       .set("genpoints", new String[][]{{xs1Coord, ys1Coord, zs1Coord}, {xs2Coord, ys2Co
       model.result().dataset("cpl1").set("planetype", "general");
    model.result("pg6").set("window", "window1");
    model.result("pg6").run();
}

private JButton calculateButton(){
    calculateButton = new JButton("Calculate Composite Properties");
    calculateButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String fiberPer = editFC.getText();
        double fibPercent = Double.parseDouble(fiberPer);
        String matrixPer = editMC.getText();
```

'Solve Button' which calls 'solve' method

'Calculate Button' method calculates composite properties

PROGRAM ARCHITECTURE

```
public static void fiberProp(){
        try {

            FileInputStream fiberProperty = new FileInputStream(new File("D:\\Ris
            HSSFWorkbook workbook = new HSSFWorkbook(fiberProperty);
            for(int i=0; i<13; i++){
                HSSFSheet fiber = workbook.getSheetAt(i);
                Iterator<Row> rowIterator = fiber.iterator();
                int row_index=0;
                while(rowIterator.hasNext()) {
                    Row row = rowIterator.next();
```

'fiberProp' method for defining fiber properties

```
public static void matrixProp(){
    try {

        FileInputStream matrixProperty = new FileInputStream(new File("D:\\Rishabh
        HSSFWorkbook workbook = new HSSFWorkbook(matrixProperty);
        for(int i=0; i<13; i++){
            HSSFSheet matrix= workbook.getSheetAt(i);
            Iterator<Row> rowIterator = matrix.iterator();
            int mat_index=0;
            while(rowIterator.hasNext()) {
                Row row = rowIterator.next();
```

'matrixProp' method for defining matrix properties

PROGRAM ARCHITECTURE

```
private JPanel rightPanel() {
    MigLayout layout = new MigLayout("wrap 6");
    final JPanel panell = new JPanel(layout);
    JLabel label = new JLabel("Post-Processing");
    panell.add(label,"span, wrap 15px");
    label.setFont(new Font("Serif", Font.PLAIN, 25));
    label = new JLabel("Contour Plot:");
    panell.add(label,"span, wrap 10px");
    label.setFont(new Font("Serif", Font.PLAIN, 17));
    String[] contourType = {"Select Contour Type", "Lines", "Filled"};
```

'rightPanel' method for defining fields in output side of GUI

```
private JButton linePlot(){
    JButton linePlot = new JButton("Line Plot");
    linePlot.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            x1Coord = editX1.getText();
            y1Coord = editY1.getText();
            z1Coord = editZ1.getText();
            x2Coord = editX2.getText();
            y2Coord = editY2.getText();
            z2Coord = editZ2.getText();
```

'linePlot' method for plotting Line Plot

PROGRAM ARCHITECTURE

```
private void plot(){
            model.result().dataset("cln1")
            .set("genpoints", new String[][]{{x1Coord, y1Coord, z1Coord},
                    {x2Coord, y2Coord, z2Coord}});
            model.result("pg2").set("window", "window1");
            model.result("pg2").run();
}

private JButton surfacePlot(){

    JButton surfacePlot = new JButton("Surface Plot");
    surfacePlot.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            xs1Coord = editXs1.getText();
```

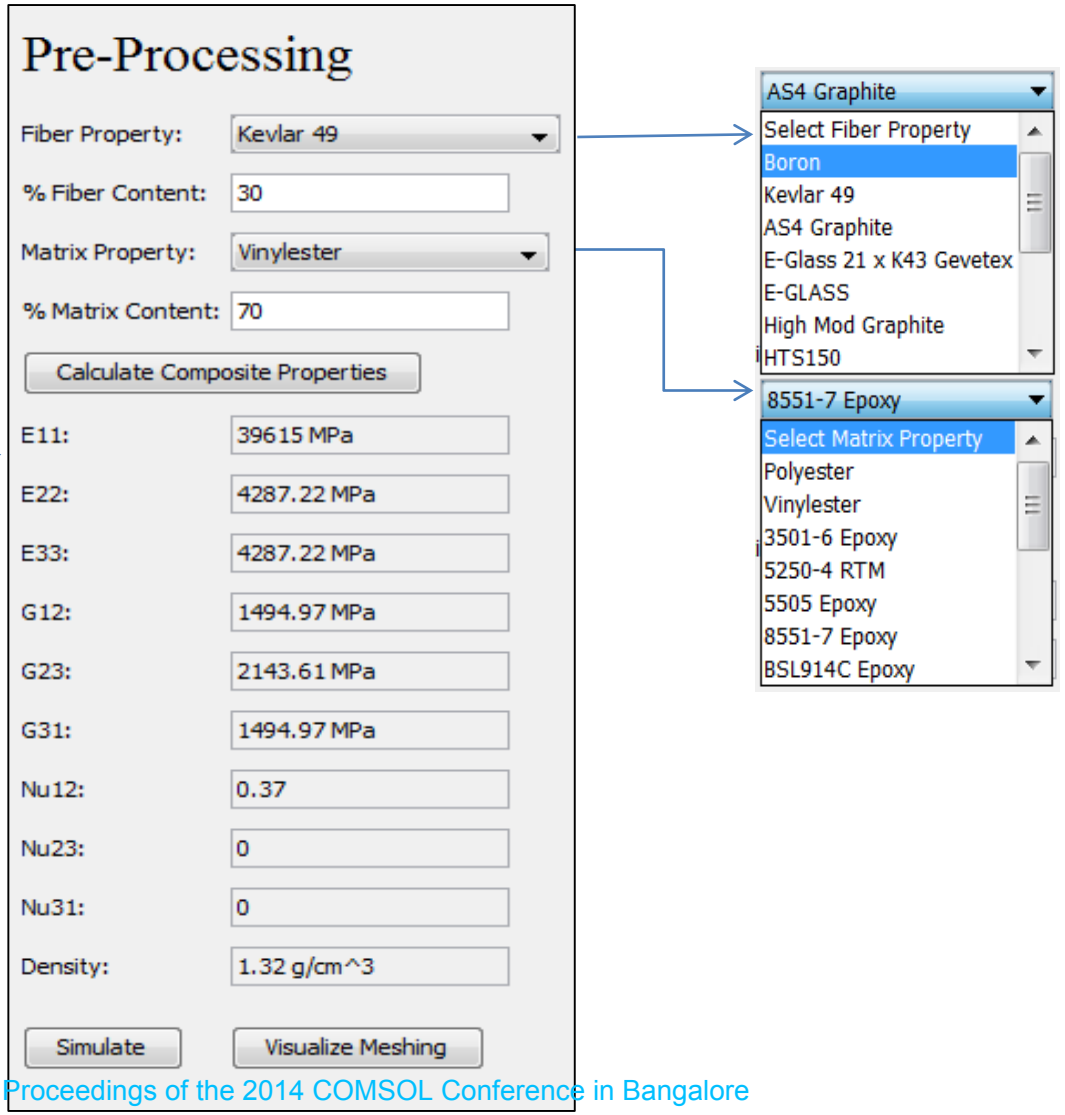
'plot' method and 'surfacePlot' method for plotting surface Plot

```
private JMenuBar menu() {
      JMenuBar menubar = new JMenuBar();
      JMenu menu = new JMenu("File");
      menubar.add(menu);
      JMenuItem item = new JMenuItem("Open");
      menu.add(item);
      item = new JMenuItem("Exit");
      item.setActionCommand("exit");
```

'menu' method for menu bar at the top of GUI

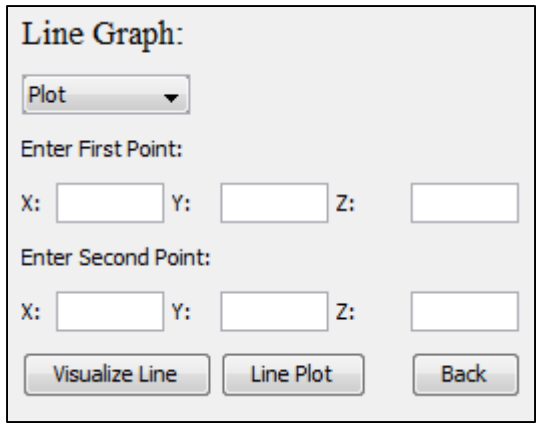
USER INTERFACE…

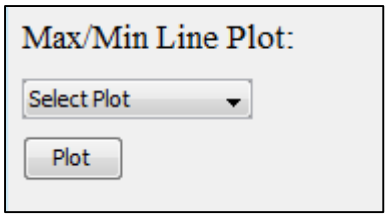
INPUT INTERFACE

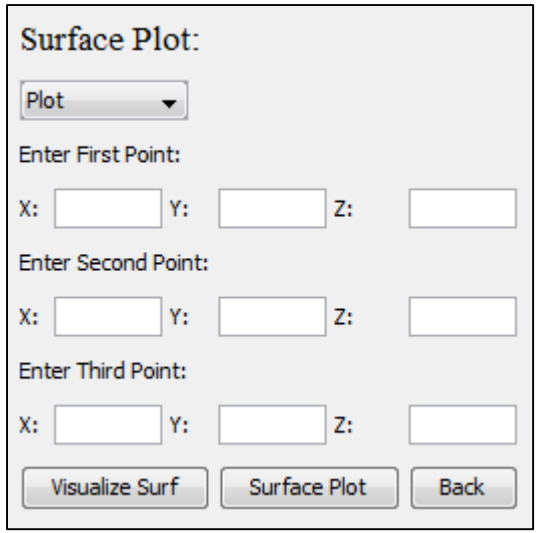


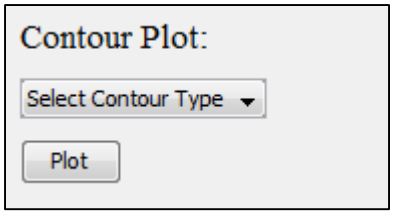
...User Interface

Interface for Post-Processing of Results

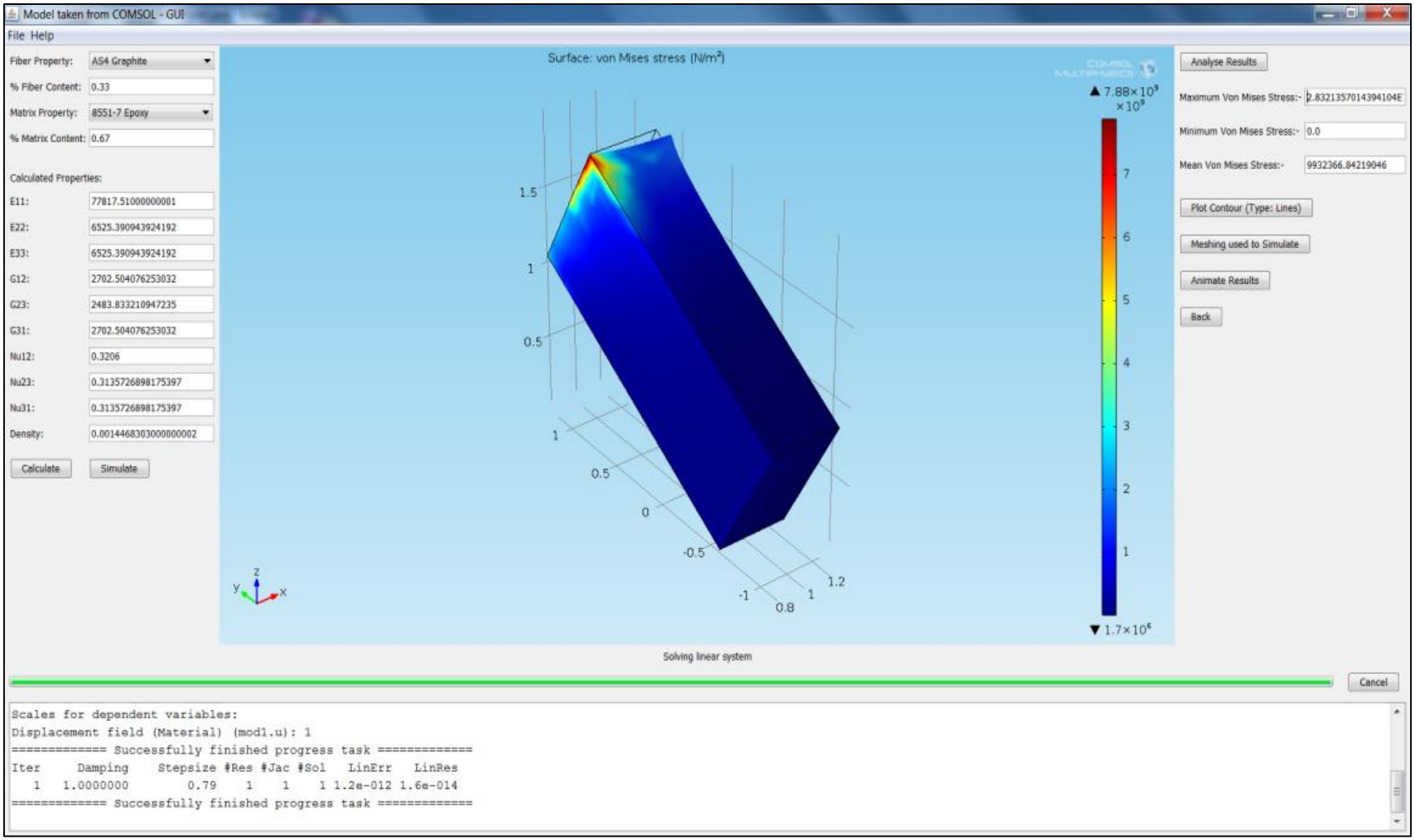




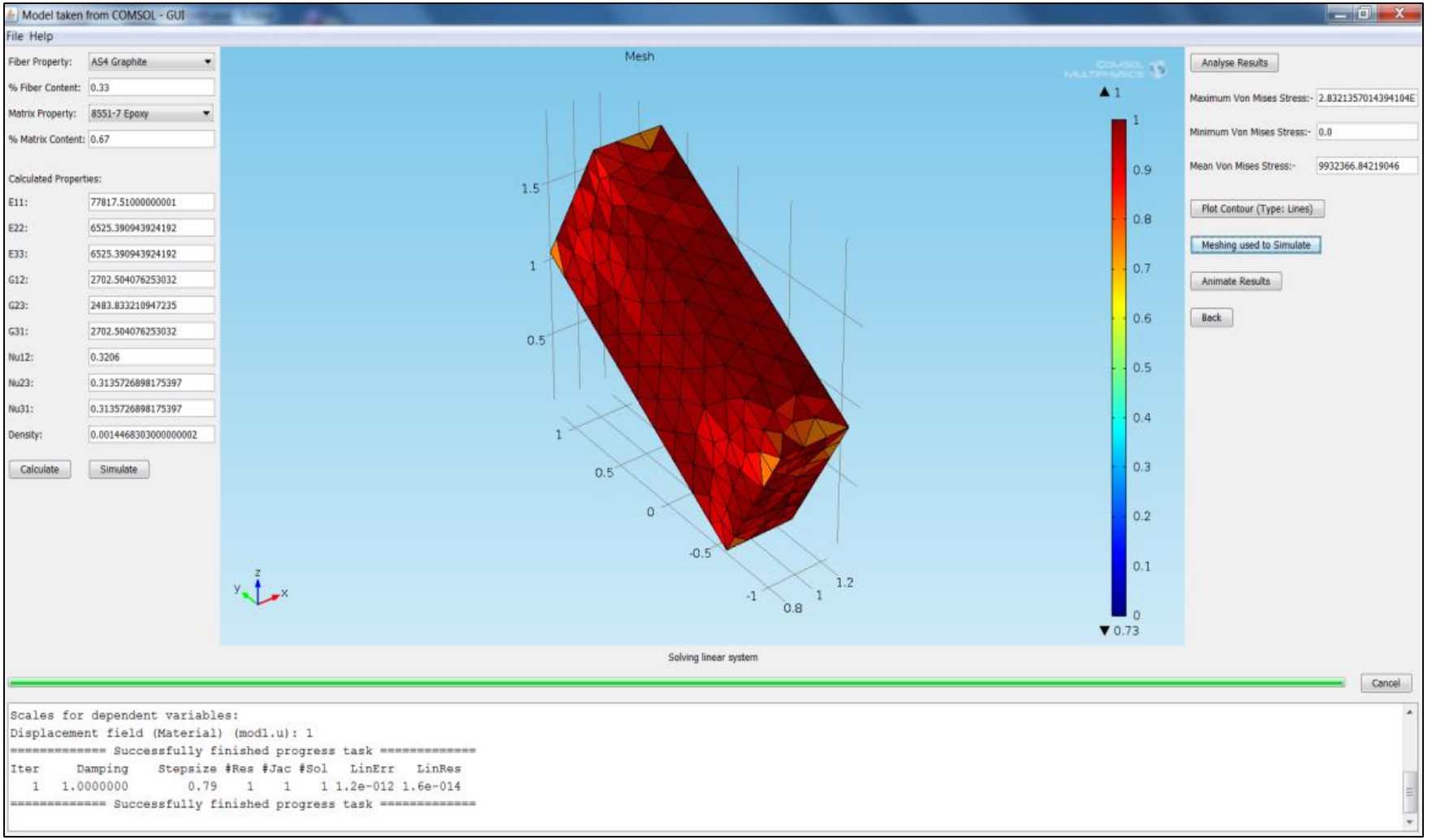




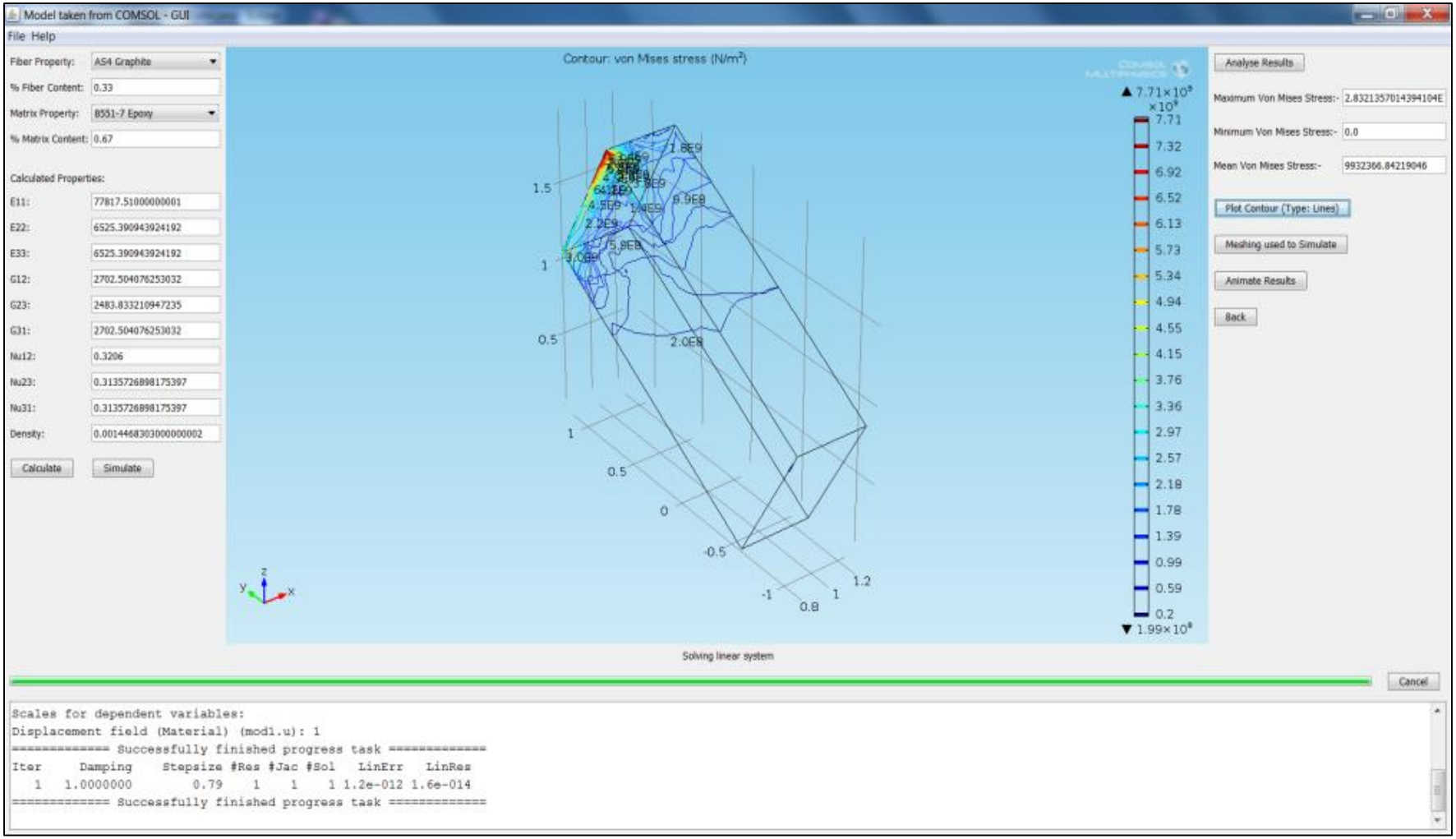
RESULTS


Model taken from COMSOL - GUI
File Help
Fiber Property: AS4 Graphite
% Fiber Content: 0.33
Matrix Property: 8551-7 Epoxy
% Matrix Content: 0.67
Calculated Properties:
E11: 77817.51000000001
E22: 6525.390943924192
E33: 6525.390943924192
G12: 2702.504076253032
G23: 2483.833210947235
G31: 2702.504076253032
Nu12: 0.3206
Nu23: 0.3135726898175397
Nu31: 0.3135726898175397
Density: 0.0014468303000000002
Calculate
Simulate
Surface: von Mises stress (N/m²)
▲ 7.88×10⁹
×10⁹
▼ 1.7×10⁶
Analyse Results
Maximum Von Mises Stress:- 2.8321357014394104E
Minimum Von Mises Stress:- 0.0
Mean Von Mises Stress:- 9932366.84219046
Plot Contour (Type: Lines)
Meshing used to Simulate
Animate Results
Back
Solving linear system
Cancel
Scales for dependent variables:
Displacement field (Material) (mod1.u): 1
============= Successfully finished progress task =============
Iter Damping Stepsize #Res #Jac #Sol LinErr LinRes
1 1.0000000 0.79 1 1 1 1.2e-012 1.6e-014
============= Successfully finished progress task =============

RESULTS

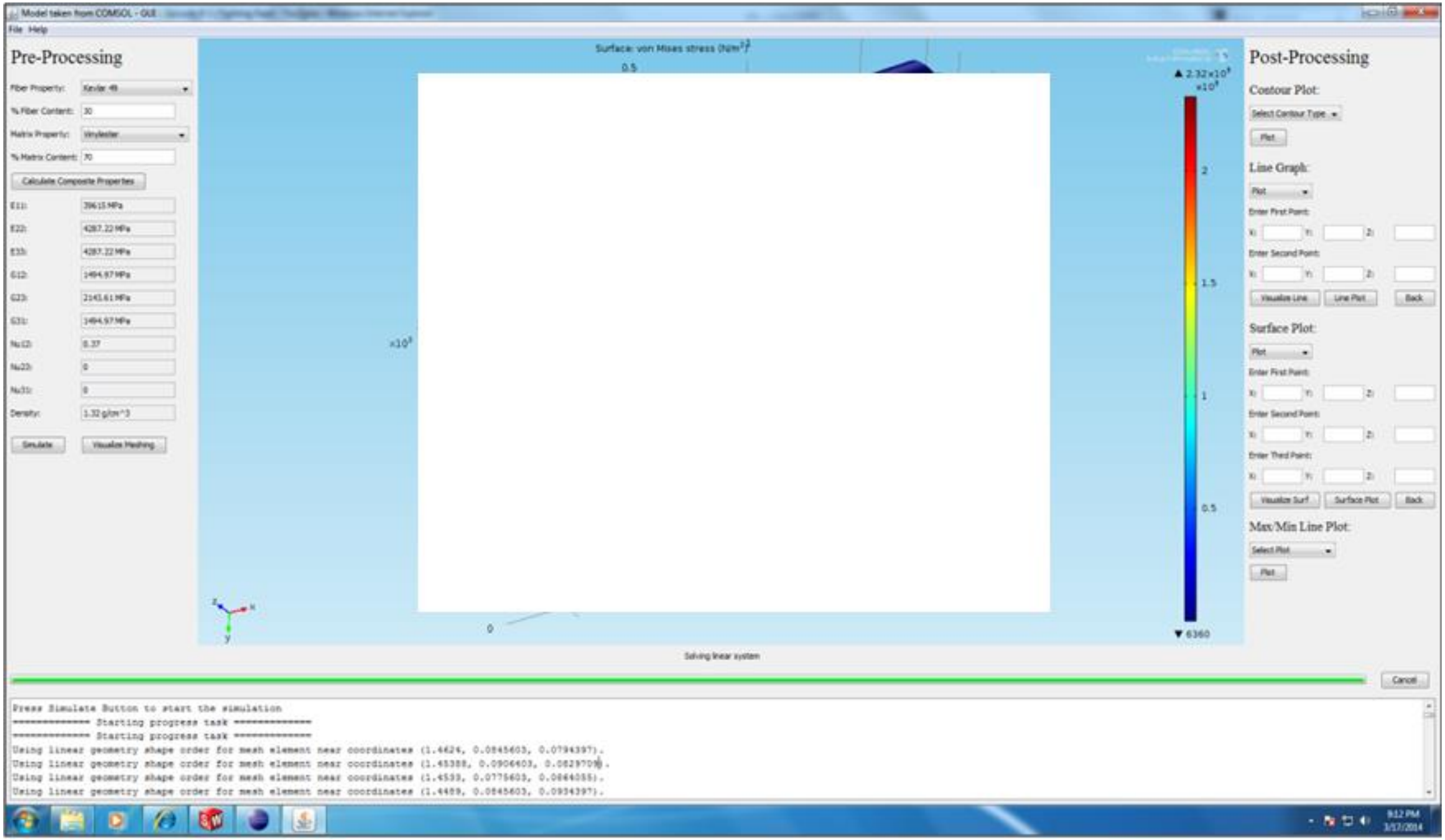

Model taken from COMSOL - GUI
File Help
Fiber Property: AS4 Graphite
% Fiber Content: 0.33
Matrix Property: 8551-7 Epoxy
% Matrix Content: 0.67
Calculated Properties:
E11: 77817.51000000001
E22: 6525.390943924192
E33: 6525.390943924192
G12: 2702.504076253032
G23: 2483.833210947235
G31: 2702.504076253032
Nu12: 0.3206
Nu23: 0.3135726898175397
Nu31: 0.3135726898175397
Density: 0.0014468303000000002
Calculate
Simulate
Mesh
Analyse Results
Maximum Von Mises Stress:- 2.83213570143941046
Minimum Von Mises Stress:- 0.0
Mean Von Mises Stress:- 9932366.84219046
Plot Contour (Type: Lines)
Meshing used to Simulate
Animate Results
Back
Solving linear system
Cancel
Scales for dependent variables:
Displacement field (Material) (mod1.u): 1
========== Successfully finished progress task ==========
Iter Damping Stepsize #Res #Jac #Sol LinErr LinRes
1 1.0000000 0.79 1 1 1 1.2e-012 1.6e-014
========== Successfully finished progress task ==========

RESULTS



CONTOUR PLOT

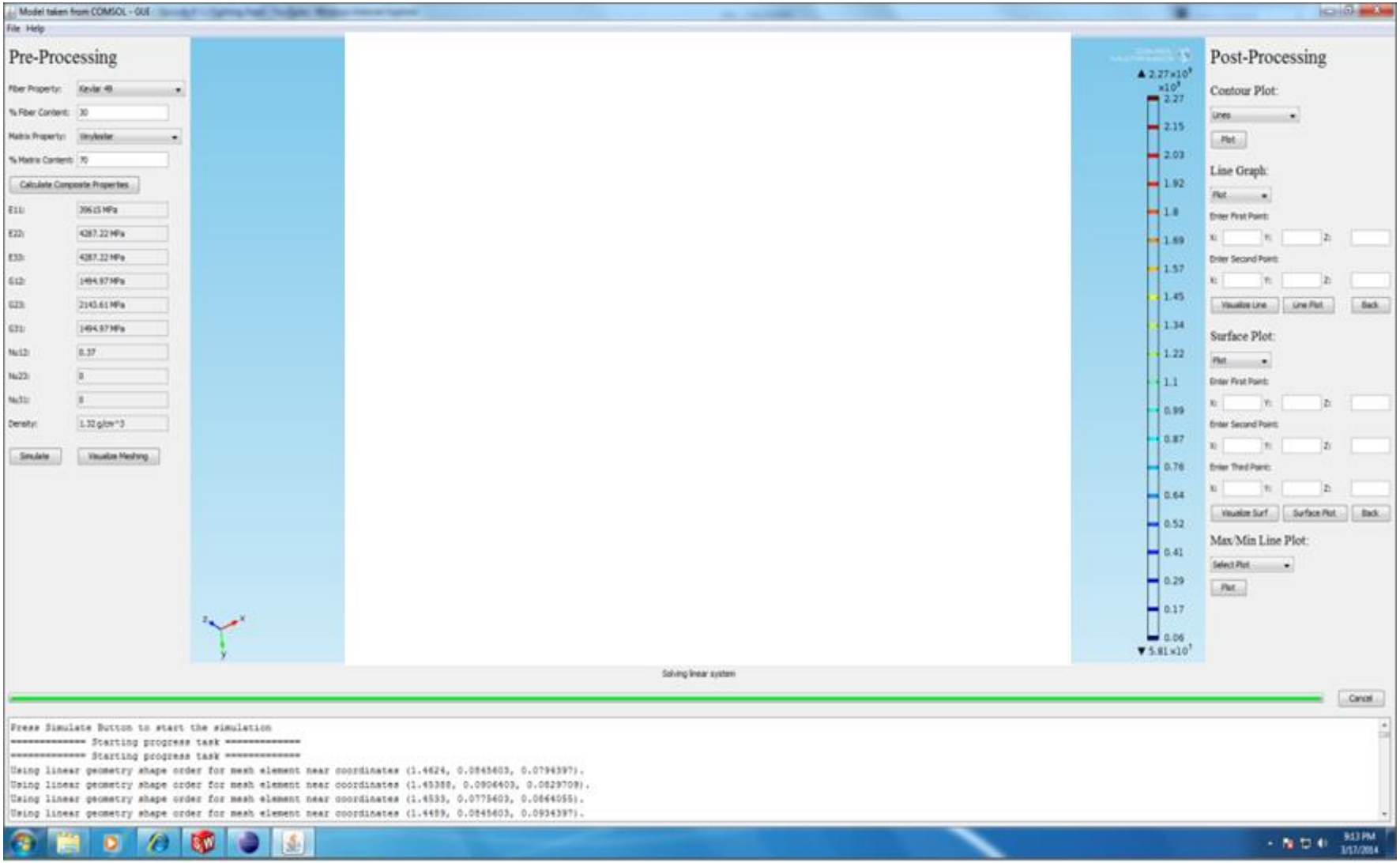
RESULTS



SIMULATION ON A MORE COMPLEX MODEL

Excerpt from the Proceedings of the 2014 COMSOL Conference in Bangalore

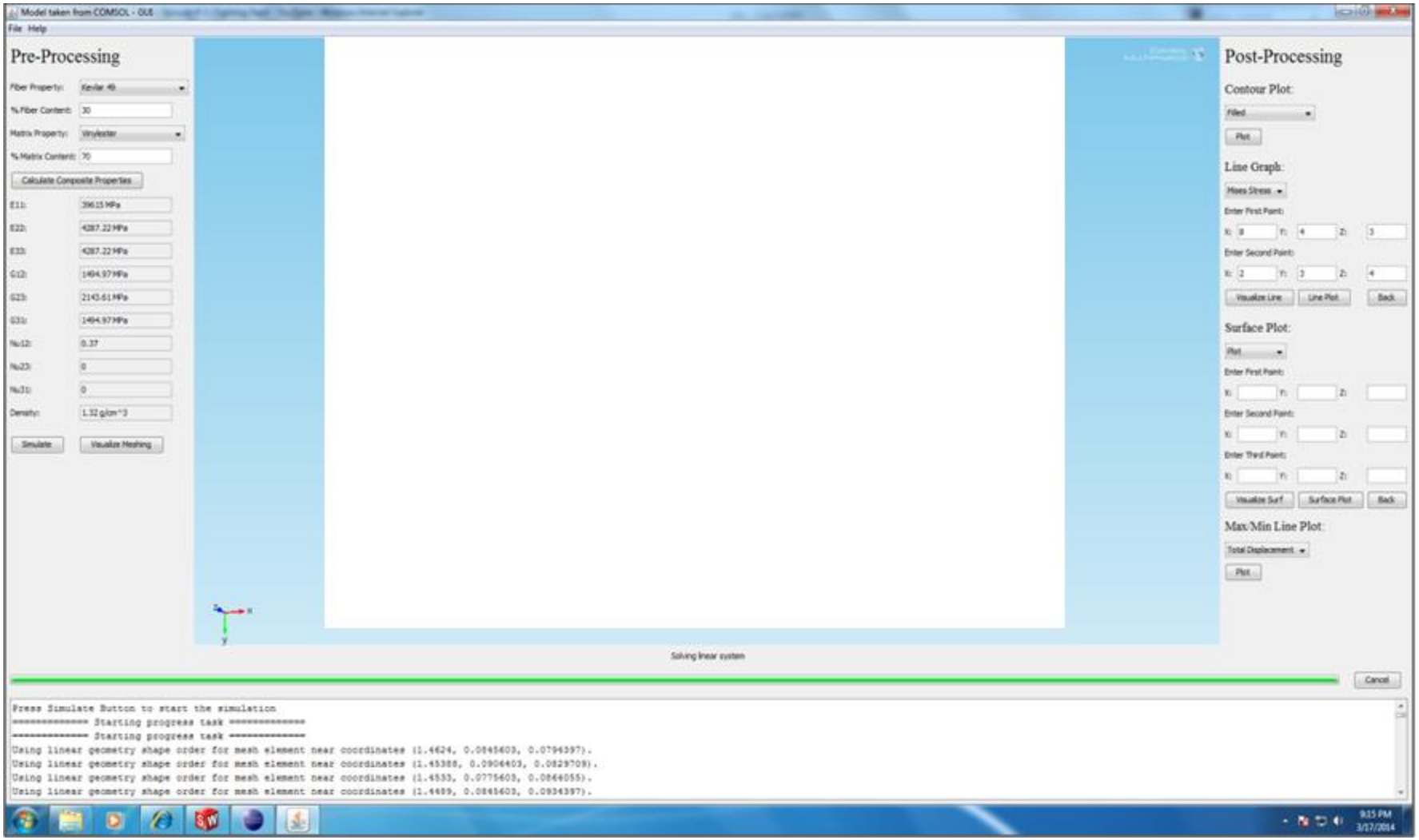
RESULTS



Excerpt from the Proceedings of the 2014 COMSOL Conference in Bangalore

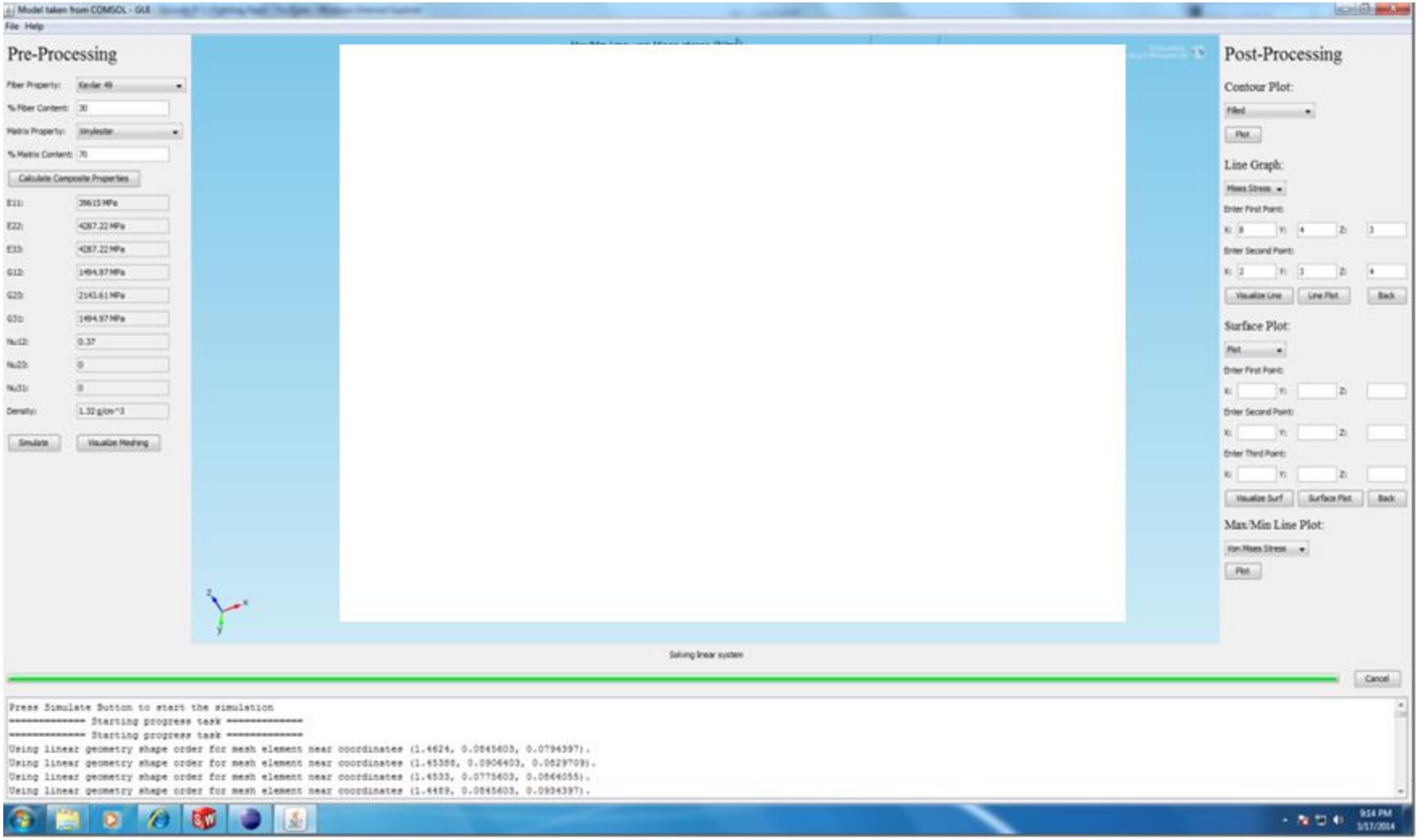
CONTOUR PLOT

RESULTS



LINE GRAPH

RESULTS



MAX/MIN LINE PLOT

Excerpt from the Proceedings of the 2014 COMSOL Conference in Bangalore

LIMITATIONS AND CHALLENGES

- Integrating failure criteria's for composite materials is not possible in an user interface
- Selecting points in the 3-D model is not possible by using mouse events on graphics panel
- Meshing of an complex model is difficult to perform in an user interface

CONCLUSION

- Composite Material properties were successfully integrated with COMSOL Multiphysics in a user-friendly interface
- GUI created is compatible to be used for any model for the purpose of performing simulation with Composite Material properties
- GUI can also be used to perform some of the important Post-Processing which can be cumbersome to do in COMSOL Multiphysics GUI for an inexperienced user

THANKS FOR YOUR ATTENTION