The Effect of Non-Linear Material Properties

On Compressive Behaviour

and

Failure of Composite Materials

By

Allan Sham

This thesis was submitted as part of the requirement for the
MEng. degree in Engineering

School of Engineering

University of Aberdeen

May 2012

Acknowledgement

I would like to thank my supervisor Professor Igor Guz for his guidance and advice throughout the duration of this project.

Abstract

This thesis investigates the compressive failure mechanisms of heterogeneous, non-linear layered composite materials using the model of piecewise-homogeneous medium and the equations of the 3D stability. The characteristic determinant was used along with the simplified version of Mooney's potential to account for the case of large deformations under equi-biaxial or uniaxial compressive loadings. The analysis finds the critical shortening factor for the case of perfectly bonded and sliding layers for the first four modes and discussion on the effects of material properties, types of loading and types of bonding on the material were examined.

Contents

List of Figures and Tables

Notation

C_{10}^r	Material constant of the reinforcement
C_{10}^m	Material constant of the matrix
h_r	Half thickness of each layer of reinforcement (m)
h_m	Half thickness of each layer of matrix (m)
l_i	Half wavelength of the modes of stability loss along OX_i axis (m)
α_r	Normalised wavelength of the reinforcement
α_m	Normalised wavelength of the matrix
ε	Strain
λ_1	Shortening factor
λ_{cr}^{sl}	Critical shortening factor for sliding without friction layers
λ_{cr}^{pb}	Critical shortening factor for perfectly bonded layers
λ_{cr}	Critical shortening factor

1. Introduction

When the words “composite material” are mentioned one might conjure up images of super-sized jumbo jets or million dollar sports car. This can be the case but in fact the first man made composite material has been around since the ancient Egyptians [8]. Instead of using a bricks made solely from clay, straw was added to reinforce these bricks leading to stronger and more stable buildings. Since then the technology of composite material has ever been advancing with the majority of these advancements coming after the 1940s.

Composite materials, or simply composites, are becoming ever more popular in the aircraft industry. About 50% of the new Boeing 787 and Airbus 350 will consist of composite material [10,11]. In theory the whole plane can be made from composites but in many cases a regular, cheaper monolithic material can do the job just as well. For example, aluminium is generally considered to be better in compression than most types of composite materials. Composites materials are generally 20% lighter in weight compared to a more conventional material such as aluminium [10,11]. This reduction in weight will lead to a reduction in fuel and hence a reduction in cost. Furthermore, composites are known to have a longer life span meaning the need for maintenance can be less frequent, and again, money can be saved. These advantages normally outweigh the main disadvantage which is the manufacturing cost. The manufacturing of composites requires expensive equipment to bond the already expensive constituents together.

Composite materials are comprised of two or more different materials which are bonded together in such a way that it produces an altogether new material. This new material has properties which are superior to the individual materials and often at a lighter weight. This decrease in weight along with an increase in mechanical properties makes composite materials ideal for use in applications where weight is an important characteristic such as aeroplanes and bridges. Composite materials can have improvements in: stiffness, ultimate strength, corrosion resistance, thermal resistance, thermal conductivity and more. Obviously not all these properties will, or can be improved at once, for example thermal resistance and thermal conductivity are opposite properties [7]. This is also one of the main differences of composite materials; generally a

composite material is specifically designed for the application as opposed to a material being chosen for the application.

This thesis, based on papers [1-6] investigated by Professor I.A. Guz and his colleagues, will investigate the failure of composite materials under compressive loading. The focus will be on elastic materials which deforms greatly before the point of failure is reached such as the rubber used in tyres. A set of equations considered to be "exact" derived in [1-2 and 6] characterises the considered material and a computer program will be written to solve these equations. A graphical user interface will then be developed for users to "communicate" with the program and using this, the effects of material properties, defects in the composite and changes in the loading will take place.

2. Background Knowledge

2.1 Internal Instability

This thesis focuses on compression of composite materials. Generally, the compressive strength of today's layered composite materials is 30-40% lower than the tensile strength [1] making compression one of the weak links. This contrast in strength is due to something called fibre instability or microbuckling (fig.1). A good way to visualise microbuckling is to imagine the effect of pushing in the ends of a plastic ruler. As one increases the compressive force on the ruler it will reach a point where the ruler will suddenly buckle in the middle. This sudden buckle is considered the point of failure of the material and is what this thesis will investigate. This microbuckling will then propagate and form a kink band as marked by the dashed lines in figure 2 [1]. This kink band can be the basis of other models used to analyse the failure of composite materials under compression. However, in these models an initial input has to be guessed [1] and hence might not be as accurate. The model used to analyse the material in this thesis will be based on microbuckling and is called the model of piecewise-homogeneous medium.

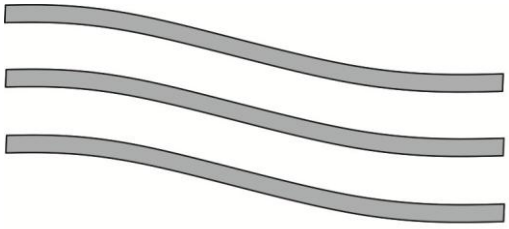


Fig. 1. Microbuckling [7]

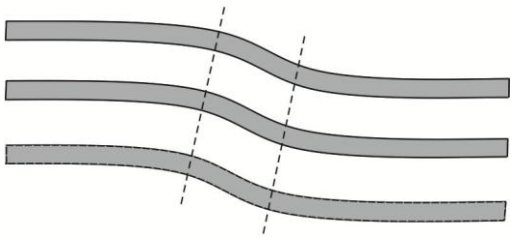


Fig. 2. Kink band [7]

2.2 Piecewise-Homogeneous Medium Model

When analysing composite materials in compression there are two main approaches that makes use of the 3D stability theory and are hence considered 'exact', these are the piecewise-homogeneous medium model and the continuum theory. The continuum theory includes many simplifications so that it is easier to implement but this also means a drop in accuracy. On the other hand, the piecewise-homogeneous medium model does not have these simplifications and consequently will be utilised in this thesis. A comparison of the two models were investigated in papers [1] and [4], it was

concluded that the continuum theory cannot be used to describe the modes of stability loss, other than for the 1st mode, when applied to incompressible non-linear materials undergoing large deformation. The usage of the piecewise-homogeneous medium model can also be used for other types of materials including compressible materials, linear materials and material undergoing small deformation.

2.3. Incompressible Non-Linear Hyperelastic Transversally Isotropic Material

The material considered in this thesis can be described by the phrase "incompressible non-linear hyperelastic transversally isotropic". This might be confusing but can be split up into four parts. Firstly, incompressible means that the material does not change in density this is the opposite of a compressible material. Secondly, non-linear means that the material does not deform proportionally to the applied loading or in other words it does not obey Hooke's law. Due to this, the material cannot be characterised by a simple equation such as Hooke's law. Instead, the simplified version of Mooney's potential, namely the so called neo-Hookean potential will be used to account for the hyperelastic nature of the material. Lastly, a transversally isotropic material is a material which has a plane where the mechanical properties is the same in all direction, called a plane of isotropy, but the mechanical properties normal to this plane has different properties. In figure 3 and 4 the plane of isotropy will be he X_1-X_2 plane and the direction normal to this plane is the OX_3 axis.

3. Theory

The full derivation of the equations which characterises the considered material can be found in [1,2,4,5 and 6]. The following chapter gives the beginning of the derivation, useful for future referencing and then chapter 3.2 states the derived equations in which this thesis is based on.

3.1 Problem Statement

The problem of instability of static non-axisymmetrical layered composites will be examined. The composite will consist of alternating layers of reinforcement and matrix denoted by the scripts r and m respectively. These layers of reinforcement and matrix are simulated by incompressible non-linear hyperelastic transversally isotropic material of thickness $2h_r$ and $2h_m$ respectively. The material will be under uniaxial compression (fig. 3) or equi-biaxial (fig. 4) compression, applied by static dead loads at infinity so that all layers have equal deformation.

Using the piecewise-homogeneous medium model and the 3D stability equations the following eigen-value problem is solved.

$$u_i^0 = (\lambda_i - 1)x_i\,, \qquad \lambda_i = const\,, \qquad \varepsilon_{ij}^0 = (\lambda_i - 1)\delta_{ij} \tag{1}$$

where u_i^0 and ε_{ij}^0 are the pre-critical axial displacement and strain respectively, λ_i is the shortening factor along the OX_i axis and δ_{ij} is the Kronecker symbol.

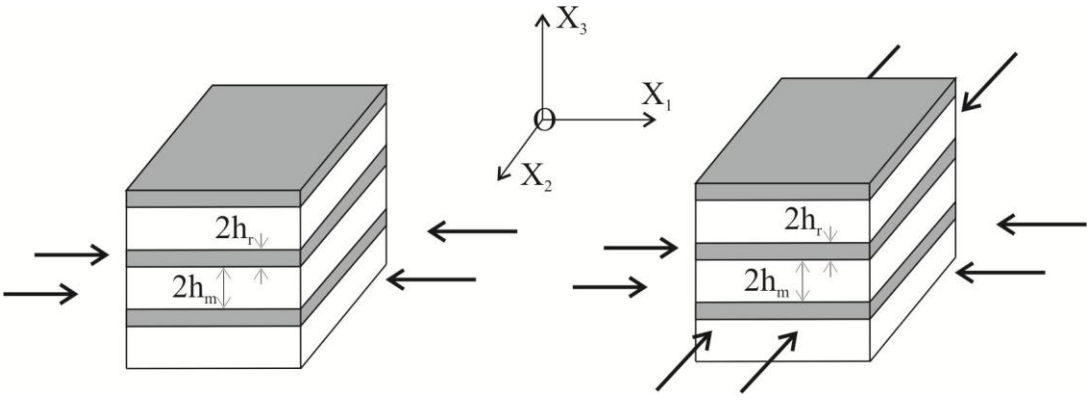


Fig. 3. Uniaxial compression [4]

Fig. 4. Equi-biaxial Compression [1]

The stability equations for the individual layers are given by

$$\frac{\partial}{\partial x_i} t_{ij}^r = 0, \qquad \frac{\partial}{\partial x_i} t_{ij}^m = 0; \qquad i,j = 1,2,3 \tag{2}$$

where t_{ij} is the non-symmetrical Piola-Kirchoff stress tensor. For incompressible solids ($\lambda_1\lambda_2\lambda_3 = 1$ is the incompressibility condition) t_{ij} has following form

$$t_{ij} = \kappa_{ij\alpha\beta} \frac{\partial u_\alpha}{\partial x_\beta} + \delta_{ij}\lambda_i^{-1} p \tag{3}$$

where p is the hydrostatic pressure and $\kappa_{ij\alpha\beta}$ in the most general form is

$$\kappa_{ij\alpha\beta} = \lambda_j\lambda_\alpha\left[\delta_{ij}\delta_{\alpha\beta}A_{\beta i} + \left(1-\delta_{ij}\right)\left(\delta_{i\alpha}\delta_{j\beta}\mu_{ij} + \delta_{i\beta}\delta_{j\alpha}\mu_{ji}\right)\right] + \delta_{i\beta}\delta_{j\alpha}S_{\beta\beta}^0 \tag{4}$$

$A_{\beta i}$ and μ_{ij} characterises the axial and shear stiffnesses respectively. The quantity characterising the pre-critical stress S_{ij}^0, the pre-critical strain ε_{ij}^0, and the shortening factor λ_i, are the parameters in respect to which the eigen-value problem is solved.

Furthermore, to complete the problem statement the boundary conditions are needed. This depends on the type of bonding between layers in the composite and in this thesis we will consider perfectly bonded layers and sliding without friction layers. For the case of perfectly bonded layers the boundary conditions for the stresses and the displacements are

$$\begin{aligned} &t_{31}^r = t_{31}^m, \qquad t_{32}^r = t_{32}^m, \qquad t_{33}^r = t_{33}^m, \\ &u_3^r = u_3^m, \qquad u_2^r = u_2^m, \qquad u_1^r = u_1^m \end{aligned} \tag{5}$$

and for the case of sliding without friction the conditions are

$$\begin{aligned} &t_{31}^r = t_{31}^m = t_{32}^r = t_{32}^m = 0, \\ &t_{33}^r = t_{33}^m, \qquad u_3^r = u_3^m \end{aligned} \tag{6}$$

More on the difference between perfectly bonding and sliding without friction layers will be discussed in chapter 3.3.

3.2 The Characteristic Equations

The eigen-value problem is solved yielding four six by six characteristic determinants. The four determinants will correspond to the two types of bonding and the two types of loading being considered. Accounting for the hyperelastic nature of the material we are considering in this thesis the neo-Hookean potential is applied to the determinants. After some rearranging, the following list of characteristic equations is derived for the case of alternating layers of matrix and reinforcement simulated by incompressible non-linear hyperelastic transversally isotropic material.

Equi-Bixial Perfectly Bonded Mode 1:

$$\begin{aligned}
&-\lambda_1^{-3}(1+\lambda_1^6)^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r\lambda_1^{-3})\,tanh(\alpha_m\lambda_1^{-3})\\
&-4\lambda_1^3\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,tanh(\alpha_m)\\
&+\left[2-(1+\lambda_1^6)\frac{C_{10}^r}{C_{10}^m}\right]^2 tanh(\alpha_r\lambda_1^{-3})\,tanh(\alpha_m)\\
&+\left(1+\lambda_1^6-2\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,tanh(\alpha_m\lambda_1^{-3})\\
&+(1-\lambda_1^6)^2\frac{C_{10}^r}{C_{10}^m}[tanh(\alpha_r)\,tanh(\alpha_r\lambda_1^{-3})+tanh(\alpha_m)\,tanh(\alpha_m\lambda_1^{-3})]\\
&=0
\end{aligned} \tag{7}$$

Equi-Bixial Perfectly Bonded Mode 2:

$$\begin{aligned}
&-\lambda_1^{-3}(1+\lambda_1^6)^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r\lambda_1^{-3})\,coth(\alpha_m\lambda_1^{-3})\\
&-4\lambda_1^3\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,coth(\alpha_m)\\
&+\left[2-(1+\lambda_1^6)\frac{C_{10}^r}{C_{10}^m}\right]^2 tanh(\alpha_r\lambda_1^{-3})\,coth(\alpha_m)\\
&+\left(1+\lambda_1^6-2\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,coth(\alpha_m\lambda_1^{-3})\\
&+(1-\lambda_1^6)^2\frac{C_{10}^r}{C_{10}^m}[tanh(\alpha_r)\,tanh(\alpha_r\lambda_1^{-3})+coth(\alpha_m)\,coth(\alpha_m\lambda_1^{-3})]\\
&=0
\end{aligned} \tag{8}$$

Equi-Bixial Perfectly Bonded Mode 3:

$$
\begin{aligned}
&-\lambda_1^{-3}(1+\lambda_1^6)^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 coth(\alpha_r\lambda_1^{-3})\,coth(\alpha_m\lambda_1^{-3})\\
&-4\lambda_1^3\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 coth(\alpha_r)\,coth(\alpha_m)\\
&+\left[2-(1+\lambda_1^6)\frac{C_{10}^r}{C_{10}^m}\right]^2 coth(\alpha_r\lambda_1^{-3})\,coth(\alpha_m)\\
&+\left(1+\lambda_1^6-2\frac{C_{10}^r}{C_{10}^m}\right)^2 coth(\alpha_r)\,coth(\alpha_m\lambda_1^{-3})\\
&+(1-\lambda_1^6)^2\frac{C_{10}^r}{C_{10}^m}[coth(\alpha_r)\,coth(\alpha_r\lambda_1^{-3})+coth(\alpha_m)\,coth(\alpha_m\lambda_1^{-3})]\\
&=0
\end{aligned}
\tag{9}
$$

Equi-Bixial Perfectly Bonded Mode 4:

$$
\begin{aligned}
&-\lambda_1^{-3}(1+\lambda_1^6)^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r\lambda_1^{-3})\,tanh(\alpha_m\lambda_1^{-3})\\
&-4\lambda_1^3\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,tanh(\alpha_m)\\
&+\left[2-(1+\lambda_1^6)\frac{C_{10}^r}{C_{10}^m}\right]^2 tanh(\alpha_r\lambda_1^{-3})\,tanh(\alpha_m)\\
&+\left(1+\lambda_1^6-2\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,tanh(\alpha_m\lambda_1^{-3})\\
&+(1-\lambda_1^6)^2\frac{C_{10}^r}{C_{10}^m}[tanh(\alpha_r)\,tanh(\alpha_r\lambda_1^{-3})+tanh(\alpha_m)\,tanh(\alpha_m\lambda_1^{-3})]\\
&=0
\end{aligned}
\tag{10}
$$

Uniaxial Perfectly Bonded Mode 1:

$$
\begin{aligned}
&-\lambda_1^{-2}(1+\lambda_1^4)^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r\lambda_1^{-2})\,tanh(\alpha_m\lambda_1^{-2})\\
&-4\lambda_1^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,tanh(\alpha_m)\\
&+\left[2-(1+\lambda_1^4)\frac{C_{10}^r}{C_{10}^m}\right]^2 tanh(\alpha_r\lambda_1^{-2})\,tanh(\alpha_m)\\
&+\left(1+\lambda_1^4-2\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,tanh(\alpha_m\lambda_1^{-2})\\
&+(1-\lambda_1^4)^2\frac{C_{10}^r}{C_{10}^m}[tanh(\alpha_r)\,tanh(\alpha_r\lambda_1^{-2})+tanh(\alpha_m)\,tanh(\alpha_m\lambda_1^{-2})]\\
&=0
\end{aligned}
\tag{11}
$$

Uniaxial Perfectly Bonded Mode 2:

$$
\begin{aligned}
&-\lambda_1^{-2}(1+\lambda_1^4)^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r\lambda_1^{-2})\,coth(\alpha_m\lambda_1^{-2})\\
&-4\lambda_1^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,coth(\alpha_m)\\
&+\left[2-(1+\lambda_1^4)\frac{C_{10}^r}{C_{10}^m}\right]^2 tanh(\alpha_r\lambda_1^{-2})\,coth(\alpha_m)\\
&+\left(1+\lambda_1^4-2\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,coth(\alpha_m\lambda_1^{-2})\\
&+(1-\lambda_1^4)^2\frac{C_{10}^r}{C_{10}^m}[tanh(\alpha_r)\,tanh(\alpha_r\lambda_1^{-2})+coth(\alpha_m)\,coth(\alpha_m\lambda_1^{-2})]\\
&=0
\end{aligned}
\tag{12}
$$

Uniaxial Perfectly Bonded Mode 3:

$$
\begin{aligned}
&-\lambda_1^{-2}(1+\lambda_1^4)^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 coth(\alpha_r\lambda_1^{-2})\,coth(\alpha_m\lambda_1^{-2})\\
&-4\lambda_1^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 coth(\alpha_r)\,coth(\alpha_m)\\
&+\left[2-(1+\lambda_1^4)\frac{C_{10}^r}{C_{10}^m}\right]^2 coth(\alpha_r\lambda_1^{-2})\,coth(\alpha_m)\\
&+\left(1+\lambda_1^4-2\frac{C_{10}^r}{C_{10}^m}\right)^2 coth(\alpha_r)\,coth(\alpha_m\lambda_1^{-2})\\
&+(1-\lambda_1^4)^2\frac{C_{10}^r}{C_{10}^m}[coth(\alpha_r)\,coth(\alpha_r\lambda_1^{-2})+coth(\alpha_m)\,coth(\alpha_m\lambda_1^{-2})]\\
&=0
\end{aligned}
\tag{13}
$$

Uniaxial Perfectly Bonded Mode 4:

$$
\begin{aligned}
&-\lambda_1^{-2}(1+\lambda_1^4)^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r\lambda_1^{-2})\,tanh(\alpha_m\lambda_1^{-2})\\
&-4\lambda_1^2\left(1-\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,tanh(\alpha_m)\\
&+\left[2-(1+\lambda_1^4)\frac{C_{10}^r}{C_{10}^m}\right]^2 tanh(\alpha_r\lambda_1^{-2})\,tanh(\alpha_m)\\
&+\left(1+\lambda_1^4-2\frac{C_{10}^r}{C_{10}^m}\right)^2 tanh(\alpha_r)\,tanh(\alpha_m\lambda_1^{-2})\\
&+(1-\lambda_1^4)^2\frac{C_{10}^r}{C_{10}^m}[tanh(\alpha_r)\,tanh(\alpha_r\lambda_1^{-2})+tanh(\alpha_m)\,tanh(\alpha_m\lambda_1^{-2})]\\
&=0
\end{aligned}
\tag{14}
$$

Equi-Bixial Sliding Layers Mode 1:

$$4\lambda_1^3[\frac{C_{10}^r}{C_{10}^m}\tanh(\alpha_r)+tanh(\alpha_m)]-(1+\lambda_1^6)^2[\frac{C_{10}^r}{C_{10}^m}\tanh\left(\frac{\alpha_r}{\lambda_1^3}\right)+\tanh(\frac{\alpha_m}{\lambda_1^3})]=0 \quad (15)$$

Equi-Bixial Sliding Layers Mode 2:

$$4\lambda_1^3[\frac{C_{10}^r}{C_{10}^m}\tanh(\alpha_r)+coth(\alpha_m)]-(1+\lambda_1^6)^2[\frac{C_{10}^r}{C_{10}^m}\tanh\left(\frac{\alpha_r}{\lambda_1^3}\right)+\coth(\frac{\alpha_m}{\lambda_1^3})]=0 \quad (16)$$

Equi-Bixial Sliding Layers Mode 3:

$$4\lambda_1^3[\frac{C_{10}^r}{C_{10}^m}\coth(\alpha_r)+coth(\alpha_m)]-(1+\lambda_1^6)^2[\frac{C_{10}^r}{C_{10}^m}\coth\left(\frac{\alpha_r}{\lambda_1^3}\right)+\coth(\frac{\alpha_m}{\lambda_1^3})]=0 \quad (17)$$

Equi-Bixial Sliding Layers Mode 4:

$$4\lambda_1^3[\frac{C_{10}^r}{C_{10}^m}\coth(\alpha_r)+tanh(\alpha_m)]-(1+\lambda_1^6)^2[\frac{C_{10}^r}{C_{10}^m}\coth\left(\frac{\alpha_r}{\lambda_1^3}\right)+\tanh(\frac{\alpha_m}{\lambda_1^3})]=0 \quad (18)$$

Uniaxial Sliding Layers Mode 1:

$$4\lambda_1^2[\frac{C_{10}^r}{C_{10}^m}\tanh(\alpha_r)+tanh(\alpha_m)]-(1+\lambda_1^4)^2[\frac{C_{10}^r}{C_{10}^m}\tanh\left(\frac{\alpha_r}{\lambda_1^2}\right)+\tanh(\frac{\alpha_m}{\lambda_1^2})]=0 \quad (19)$$

Uniaxial Sliding Layers Mode 2:

$$4\lambda_1^2[\frac{C_{10}^r}{C_{10}^m}\tanh(\alpha_r)+coth(\alpha_m)]-(1+\lambda_1^4)^2[\frac{C_{10}^r}{C_{10}^m}\tanh\left(\frac{\alpha_r}{\lambda_1^2}\right)+\coth(\frac{\alpha_m}{\lambda_1^2})]=0 \quad (20)$$

Uniaxial Sliding Layers Mode 3:

$$4\lambda_1^2[\frac{C_{10}^r}{C_{10}^m}\coth(\alpha_r)+coth(\alpha_m)]-(1+\lambda_1^4)^2[\frac{C_{10}^r}{C_{10}^m}\coth\left(\frac{\alpha_r}{\lambda_1^2}\right)+\coth(\frac{\alpha_m}{\lambda_1^2})]=0 \quad (21)$$

Uniaxial Sliding Layers Mode 4:

$$4\lambda_1^2[\frac{C_{10}^r}{C_{10}^m}\coth(\alpha_r)+tanh(\alpha_m)]-(1+\lambda_1^4)^2[\frac{C_{10}^r}{C_{10}^m}\coth\left(\frac{\alpha_r}{\lambda_1^2}\right)+\tanh(\frac{\alpha_m}{\lambda_1^2})]=0 \quad (22)$$

C_{10} is a material constant, α is the normalised wavelength, λ_1 is the shortening factor and is related to the strain by equation (1). Note that to change the equi-biaxial equation to the corresponding uniaxial equation we merely replace λ_1^{-3}, λ_1^3 *and* λ_1^6 with $\lambda_1^{-2}, \lambda_1^2$ *and* λ_1^{-4} respectively

In the next few chapters a more thorough explanation of these variables, meaning of perfectly bonded, sliding layers and the four modes will take place.

3.3 Difference between Perfectly Bonded and Sliding Layers

Many times we assume the layers of reinforcements and matrix are perfectly bonded together meaning each layer does not slide over each other. However, in reality this is not the case, various forms of adhesion breakdown (such as cracks and delaminations) may be present at the interface of the reinforcement and matrix. These often occur during the manufacturing process or may occur in use. An example of interlaminar adhesion breakdown occurs when infinitesimal sliding between the layers are allowed yet there are no gaps between these layers (fig. 5). This type of cleavage-type delamination is called defects with connected edges. Now, if in a composite material all interfaces considered to be perfectly bonded are removed and only defects with connected edges are present the composite material is now classed as a sliding without friction material (fig. 5).

A composite with fully perfectly bonded or sliding without friction interfaces probably does not exist, instead a composite will generally be somewhere between these two extremes. To find the upper and the lower extreme we first have to consider the load to cause failure or the critical load. If the critical load for the composite with perfectly bonded layers was to be compared to the sliding without friction composite,

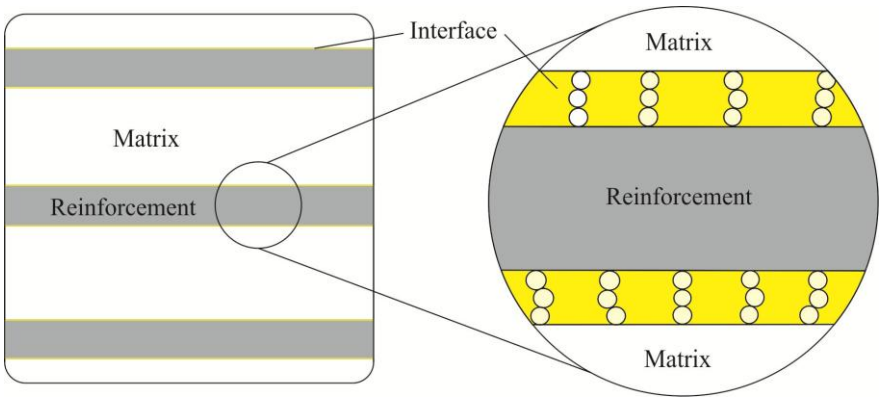


Fig. 5. A composite with sliding layers. Interface with cleavage-type delamination [2]

the critical load of the perfectly bonded composite will be greater. This is because in the sliding without friction composite there are fewer connections between the layers and hence less force is needed to cause it to fail. This in turns means the critical strain of the perfectly bonded composite will be greater than that of the sliding without friction composite ($\varepsilon_{cr}^{pb} > \varepsilon_{cr}^{sl}$). Now recall equation 1, a maximum strain corresponds to a minimum shortening factor and hence, the perfectly bonded critical shortening factor will be lower than the sliding without friction shortening factor ($\lambda_{cr}^{pb} < \lambda_{cr}^{sl}$). So in reality the critical shortening factor of a composite material will be between the critical shortening factor of the sliding layer and perfectly bonded materials, or in mathematical terms:

$$\lambda_{cr}^{pb} \leq \lambda_{cr} \leq \lambda_{cr}^{sl} \tag{23}$$

These are the bounds to be considered, the perfectly bonded is the upper bound for critical loads and the sliding layers is considered the lower bound. In terms of critical shortening factor the perfectly bonded is the lower bound for shortening factor. To avoid confusion, this thesis we will talk about the bounds in terms of critical loads. These bounds are used to generalise the characteristic equations but in theory we can find the exact critical shortening factor of a material. This is a more difficult as knowledge is required of the exact nature of the defects in the interface and even if this is known various forms of defects *within* each layer will change the critical shortening factor of a material.

3.4 The 4 Modes of Stability Loss

As mentioned earlier, a composite under compression will fail by microbuckling (internal instability). However, there are a few different ways the composite can lose stability (fig. 6). This thesis will consider the four most common types of stability loss. The first mode, called the shear mode occurs when the reinforcements buckle in phase with each other. Here the matrix between the reinforcements doesn't change in length in the vertical direction and causes a shearing effect on the matrix hence the name [7]. Mode 2 or the extension mode occurs when the reinforcements buckles out of phase with each other causing the matrix between the reinforcements to extend or contract. The 3rd and 4th mode occurs less frequently and occurs when the matrix buckles in and out of phase respectively.

The half-wavelength of stability loss, l_i, along the OX_i direction is related by

$$l^{-1} = \sqrt{l_1^{-2} + l_2^{-2}}$$
$$\alpha_r = \pi h_r l^{-1} \qquad \alpha_m = \pi h_m l^{-1} \tag{24}$$

where, h is the half thickness of the each layer, and α is the normalised wavelength. Note that as α_r and α_m tends to zero l_i tends to infinity.

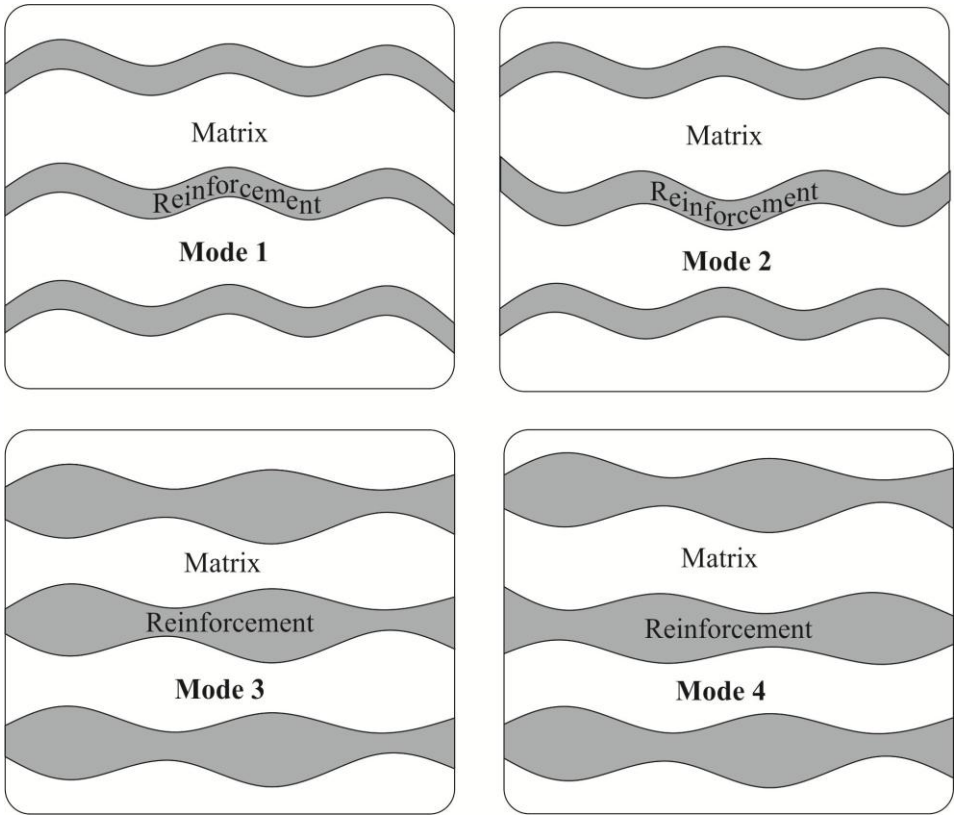


Fig. 6. The four modes of stability loss [1]

4. Mathematical Computation

Now let us put the characteristic equations (7-22) to use. The first step we have to take is to reduce the number of unknown variables. Equations (7-22) in their current form have 3 unknown variables: λ_1, α_r and α_m, we can reduce this to two variables. Rearranging equation 24 we get:

$$\alpha_m = \alpha_r \frac{h_m}{h_r} \tag{25}$$

Then substituting this equation into the characteristic equations (7-22) reduces the unknown variables to two, λ_1 and α_r. Now with only two variables we can select a range of α_r to input into the characteristics equations and find the corresponding shortening factors and hence a graph of λ_1 against α_r can be plotted.

4.1 Matlab Functions

Matlab 7.9.0 (R2009b) was implemented to solve the characteristic equations and to plot the graphs used for later analysis. Firstly, 4 individual functions were written, for the cases of:

- Equi-biaxial compression for perfectly bonded (Appendix A)
- Equi-biaxial compression for sliding layers (Appendix B)
- Uniaxial compression for perfectly bonded (Appendix C)
- Uniaxial compression for sliding layers (Appendix D)

These functions contain the corresponding characteristic equations for the four modes. For example the function for equi-biaxial compression for sliding layers contains equations (15-18), the function for uniaxial compression for perfectly bonded contains equations (11-14) and so forth. All four functions also contain equation (25) to reduce the number of variables to two.

The functions receives the variables h_r, h_m, C_{10}^r, C_{10}^m, the minimum normalized wave-length, the maximum normalized wavelength and the increments the normalized wavelength goes up in. The function then loops through the chosen range of normalized wavelength and computes the corresponding shortening factor. This is achieved using the build in Matlab command called fzero. The fzero command looks for a root of an

equation which is equal to zero, such as the characteristic equations. The outcome is a list of normalized wavelength with the corresponding shortening factor. Using these data a graph can be plotted of shortening factor, λ_1, against the normalised wavelength, α_r.

4.2 Matlab GUI

Next, a graphical user interface (GUI) was developed to allow the user to "communicate" with the Matlab code (fig. 7). The GUI pulls together the four functions and lets the user decide what needs to be plotted and with what constants. There are four input boxes for entering the values of the reinforcement and matrix. Although, only two input boxes are actually needed as the ratios C_{10}^r / C_{10}^m and h_r / h_m are the ones of importance rather than the individual numbers (eqns. 7-22 and 25).

Three more input boxes are used for the maximum, minimum and increments of the normalized wavelength. Generally, when analysing a graph in which the shape of the graph is of importance an increment of 0.01 can be used. This can be increased to 0.001 if one wants to compute a more exact value of critical shortening factor.

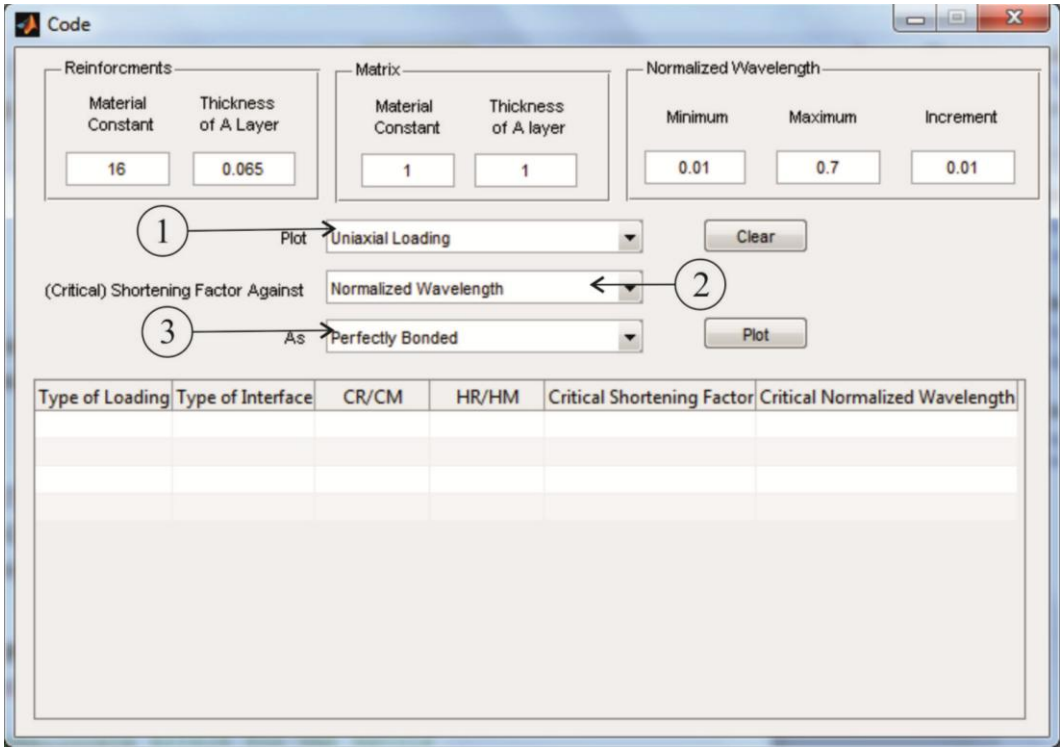


Fig. 7. The graphical user interface (GUI)

The GUI incorporates three drop-down menus marked 1, 2 and 3 on figure 7. The drop down menus allows the user to select what needs to be plotted with what type of loading and bonding. The options of each drop-down menu are listed below.

Drop-Down Menu 1:	**Drop-Down Menu 2:**	**Drop-Down Menu 3:**
• Uniaxial loading	• Normalised wavelength	• Perfectly Bonded
• Equi-biaxial loading	• Ratio of material constant	• Sliding Layers
• Both on the same graph	• Ratio of material thickness	• Both on the same graph

Drop-down menu 2 selects the x-axis of the graph to be plotted. If the ratio of material constant or the ratio of material thickness is selected another window will appear (fig.(8a) or fig.(8b)). This window allows the user to select the minimum, maximum and the increments of the chosen variable the user would like to plot. Care should be taken when entering the increment as a very small increment can take a very long time to compute but a large increment can give an inaccurate graph. Generally, an increment of 0.01-0.05 of the range will be used in this thesis i.e.

$$0.01\ to\ 0.05 = \frac{increment}{max - min} \tag{26}$$

The GUI also incorporates a table at the bottom of the window displaying the critical shortening factor and the corresponding normalized wavelength for the chosen variables. This allows the user to compare the critical shortening factor of different loading schemes and different variables

4.3 Matlab Code

The main code for the GUI can be found in appendix E and F. The two appendices are of one file but have been split into two for convenience. Appendix E is the code which is automatically written by Matlab when creating a GUI and the various push buttons or drop-down menus. It also contains code for the "Clear" button which clears the input boxes on the GUI.

Appendix F is the main code executed when the "Plot" button is clicked and is split into two main sections. The first section contains the code to collect the data selected

by the user such as numbers entered into the input boxes, the selection of the drop-down menus and also code for any error messages that may appear. Errors which have been taken into account include for example, when the value of the minimum normalized wavelength is greater than the maximum and when a required input box has been left blank etc. The second section is split into 9 parts corresponding to the 3 options in each of the 3 drop down menus. Each part will contain the corresponding functions, plots the necessary graphs and includes any additional error messages.

(a) (b)

Fig. 8. Ratio of material constant/thickness window

5. Discussion

5.1 Shortening Factor against Normalised Wavelength

Firstly, let us consider the shortening factor against normalized wavelength graph (λ_1 vs. α_r) such as figure 9. The point of failure or **the critical shortening factor is the highest point of the four modes** as calculated by the piecewise-homogeneous model. Written mathematically:

$$\lambda_{cr} = \max_N \lambda_{cr}^{(N)} = \max_N \left(\max_{\alpha_r} \lambda_1^{(N)}\right) \tag{27}$$

where N is the number of mode.

Purely by observation of figure 9 the critical shortening factor (λ_{cr}) for the case of equi-biaxial compression of a perfectly bonded material with $C_{10}^r / C_{10}^m = 30$ and h_r / h_m=0.065, is about 0.95 and the corresponding normalised wavelength (α_r) is about 0.25. More accurately, Matlab calculates a λ_{cr} of 0.9453 and a corresponding α_r of 0.249.

The mode of instability the material will take upon stability loss will correspond to the mode of the critical shortening factor. So for the case of figure 9 the mode of

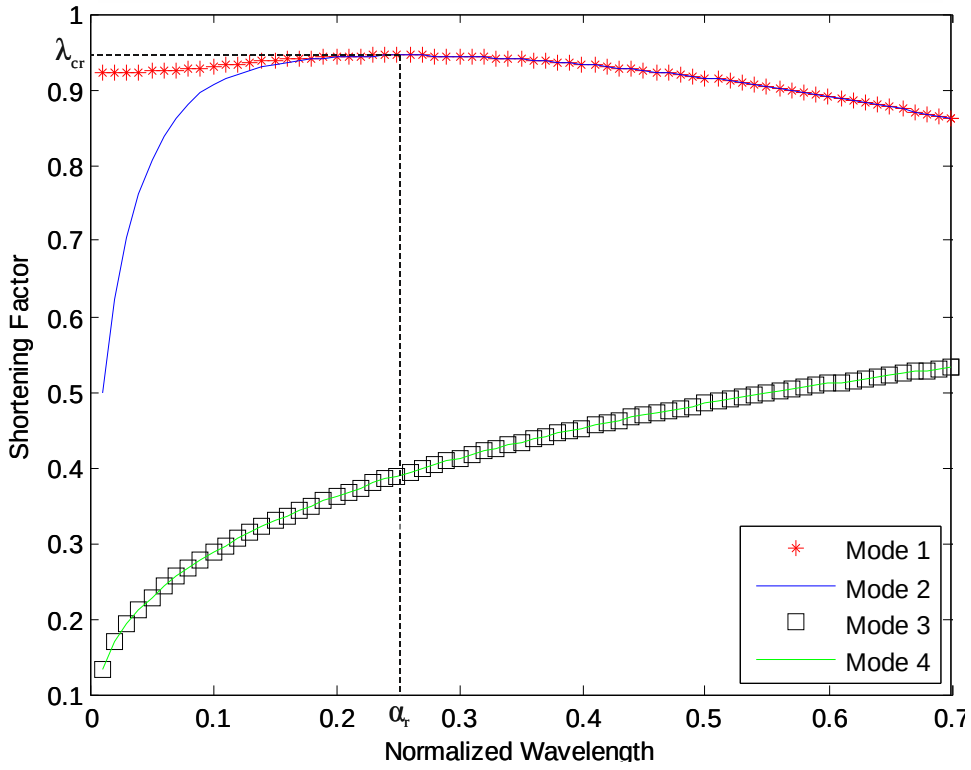


Fig. 9. Equi-biaxial, perfectly bonded with $C_{10}^r / C_{10}^m = 30$ and h_r / h_m=0.065

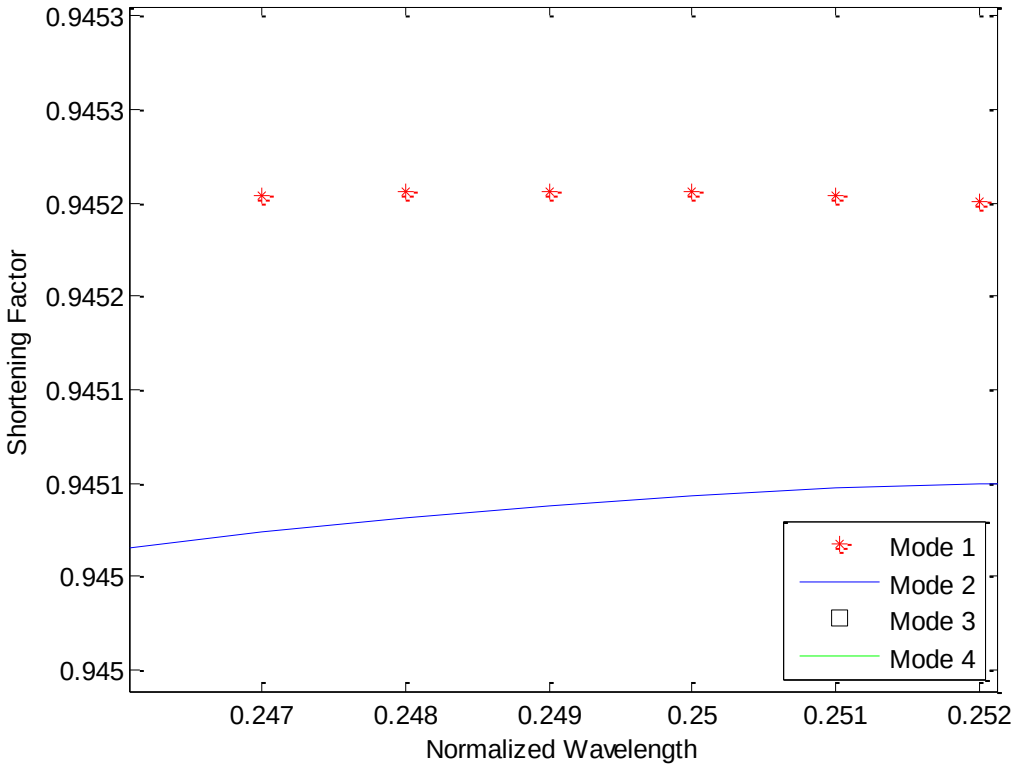


Fig. 10. Zoom in at the critical shortening factor of figure 9

instability will either be mode 1 or mode 2. If we were to zoom into the critical shortening factor(fig. 10), we can see mode 1 has a maximum λ_1 of 0.001 greater than mode 2. Therefore, according to the characteristic equations the material will fail by mode 1.

The curves for mode 3 and mode 4 are significantly below the first two modes and can generally be ignored as they don't affect the critical shortening factor. In fact, for all considered cases, whether it is equi-biaxial or uniaxial, perfectly bonded or sliding layers, mode 1 and 2 has a maximum shortening factor of at least 0.2 greater than modes 3 and 4. Recall back in chapter 3.4, we noted mode 3 and mode 4 occurs when the matrix buckles. The matrix is a lot less stiff meaning it will take a larger strain to cause it to buckle and hence a smaller critical shortening factor (eqn. 1). Furthermore, the matrix is a lot thicker than the reinforcement making it even harder to buckle. Now figure 9 seems to show the shortening factor of modes 1 and 2 decreasing and the curves of modes 3 and 4 are increasing as α_r increases. If we continue increasing α_r (fig. 11) we can see that this trend continues and eventually levels out but the curves of modes 3 and 4 never crosses over the curves of mode 1 and 2. This seems to hold true for all types of ratios and bonding.

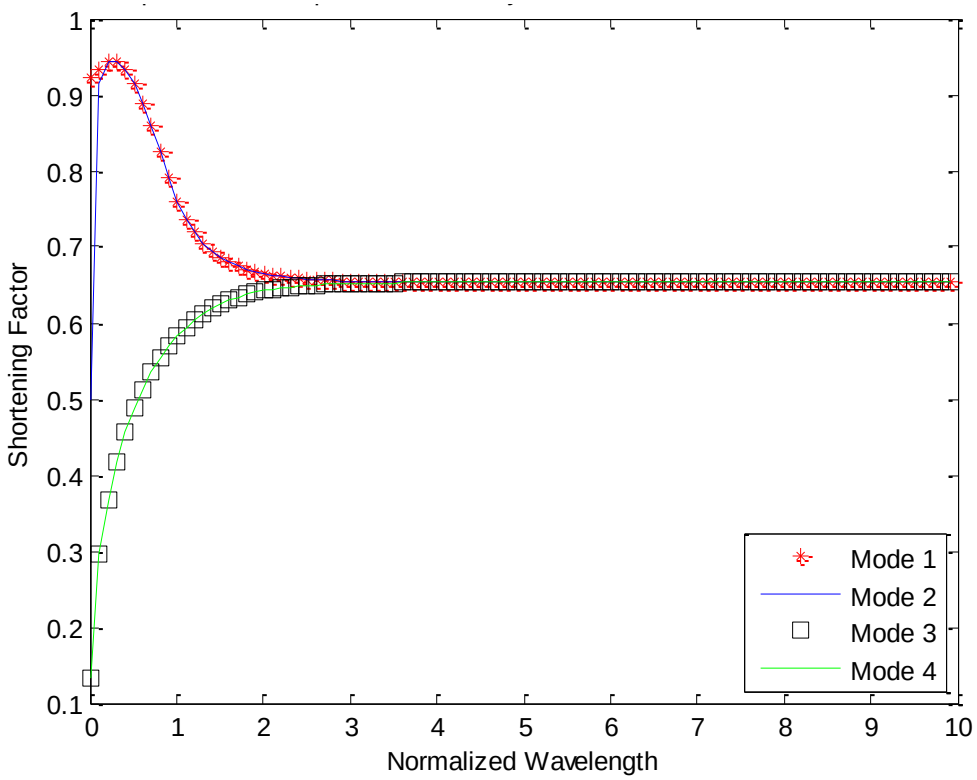


Fig. 11. Equi-biaxial, perfectly bonded with $C_{10}^r / C_{10}^m = 30$, h_r / h_m=0.065 for the range of $0 \leq \alpha_r \leq 10$

5.2 Sliding without Friction vs. Perfectly Bonded

Let us now compare the case of perfectly bonded layers to the case of sliding without friction layers (fig. 12). Figure 12 agrees with equation (23) in terms of λ_{cr}^{sl} being greater than λ_{cr}^{pb}. Also from equation (23), the actual λ_{cr} the material will buckle will be between λ_{cr}^{sl} and λ_{cr}^{pb}, for this case, Matlab gives $0.9453 \leq \lambda_{cr} \leq 0.9499$. We mentioned earlier that for the case of figure 12, the material will fail by mode 1 but now that we are also considering sliding layers, this might not be correct. For both types of bonding, mode 1 is slightly greater than mode 2. Both modes can be assumed to have a chance to occur with mode 1 having the greater frequency. The exact mode the composite will fail by will depend on the exact nature of the interfaces of the composite. This knowledge might not be simple to acquire, moreover it should be noted that knowing the mode of instability loss is not of great practical importance. Instead one should focus on determining the critical shortening factor.

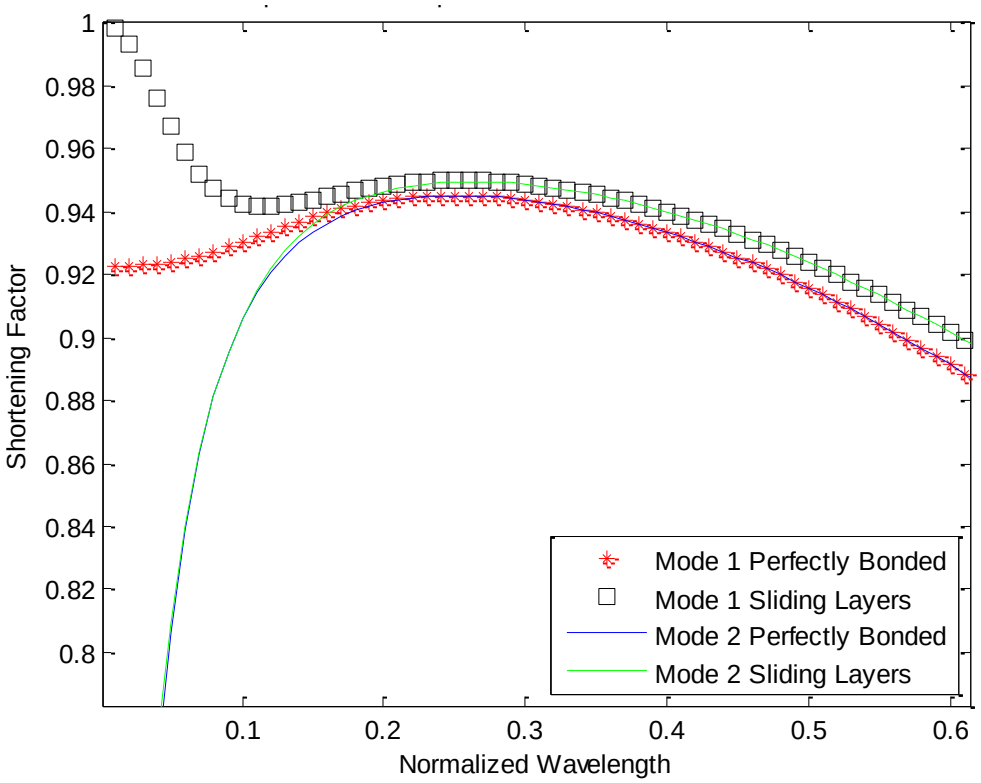


Fig. 12. Equi-biaxial, perfectly bonded and sliding layers with $C_{10}^r / C_{10}^m = 30$ and h_r / h_m=0.065

Additionally for sliding layers we have to increase the minimum α_r to 0.05-0.1, depending on the material constant ratio and the layer thickness ratio. As an example, we might change the minimum α_r of figure 12 to 0.09 as this is the point mode 1 curves upwards. It seems that it is always mode 1 that has this characteristic of curving upwards whereas mode 2 follows a similar curve to the perfectly bonded curve.

The reason we should make this change is because currently the critical shortening factor of figure 12 is very close to one. Recall equation (1), as λ_{cr} tends to 1, the strain tends to zero meaning the material is failing with very little change in length. If this is really the case the material is probably not failing by microbuckling and if it is, the material is not a hyperelastic one. This change is important when plotting the graphs for material constant ratio or layer thickness ratio to ensure an accurate graph.

5.3 Critical Shortening Factor against Material Constant Ratio

Let us now consider the λ_{cr} against the ratio of material constant, C_{10}^r / C_{10}^m, graph. Matlab draws this graph by plotting the λ_1 vs. α_r graph for each C_{10}^r / C_{10}^m in the selected

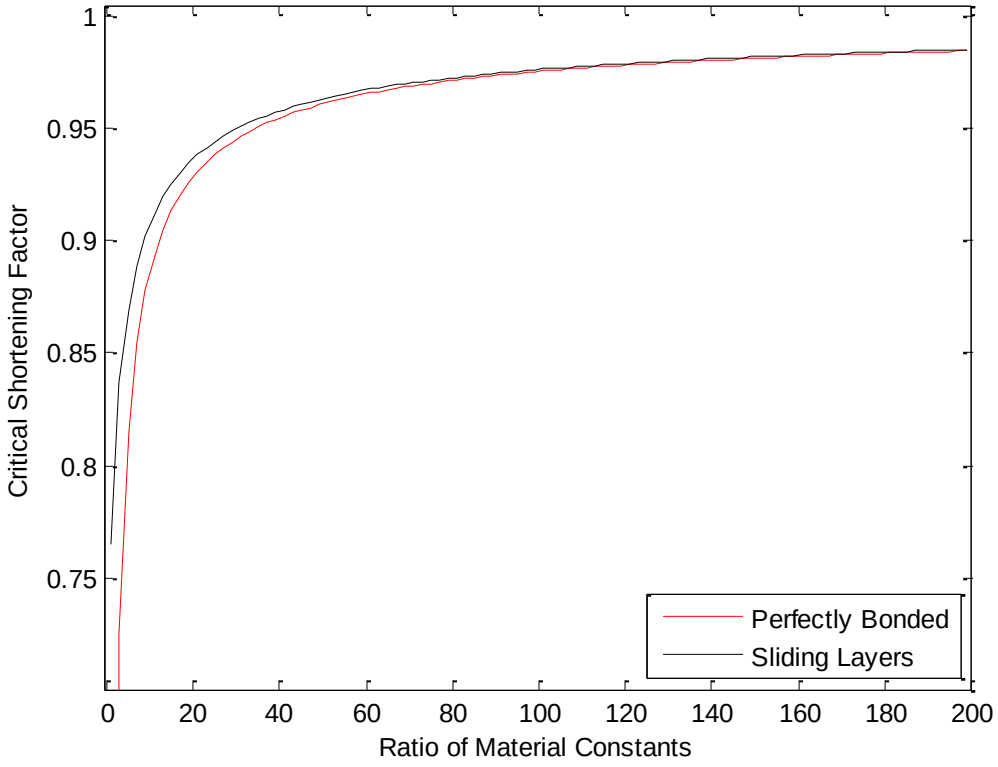


Fig. 13. Equi-biaxial, perfectly bonded and sliding layers with h_r / h_m=0.05 for the range of $1 < C_{10}^r / C_{10}^m < 200$

range and finding the critical shortening factor for each one. Matlab then plots λ_{cr} with the corresponding C_{10}^r / C_{10}^m to obtain a graph such as figure 13.

The critical shortening factor increases with C_{10}^r / C_{10}^m but not linearly. At smaller C_{10}^r / C_{10}^m the rate of increase in λ_{cr} is much greater than for a larger material constant ratio. A possible reason for this is because the change in C_{10}^r / C_{10}^m at small ratios is greater than the change in a larger C_{10}^r / C_{10}^m. For example $C_{10}^r / C_{10}^m = 8$ is twice as big as $C_{10}^r / C_{10}^m = 4$ whereas $C_{10}^r / C_{10}^m = 178$ is only 1.02 larger than $C_{10}^r / C_{10}^m = 174$.

Another point that can be observed from figure 13 is the difference between the perfectly bonded curve and sliding layers curve. As mentioned before it is expected that the perfectly bonded material will have a lower critical shortening factor than the equivalent sliding layers curve. But there is a difference at low and high values of C_{10}^r / C_{10}^m. At a low C_{10}^r / C_{10}^m , say 4 the difference between the two curves is 0.08, compare this to $C_{10}^r / C_{10}^m = 200$ the difference is only 0.002. This can be utilise if a high accuracy is needed where the upper and lower critical bounds are required to be as close to together as possible.

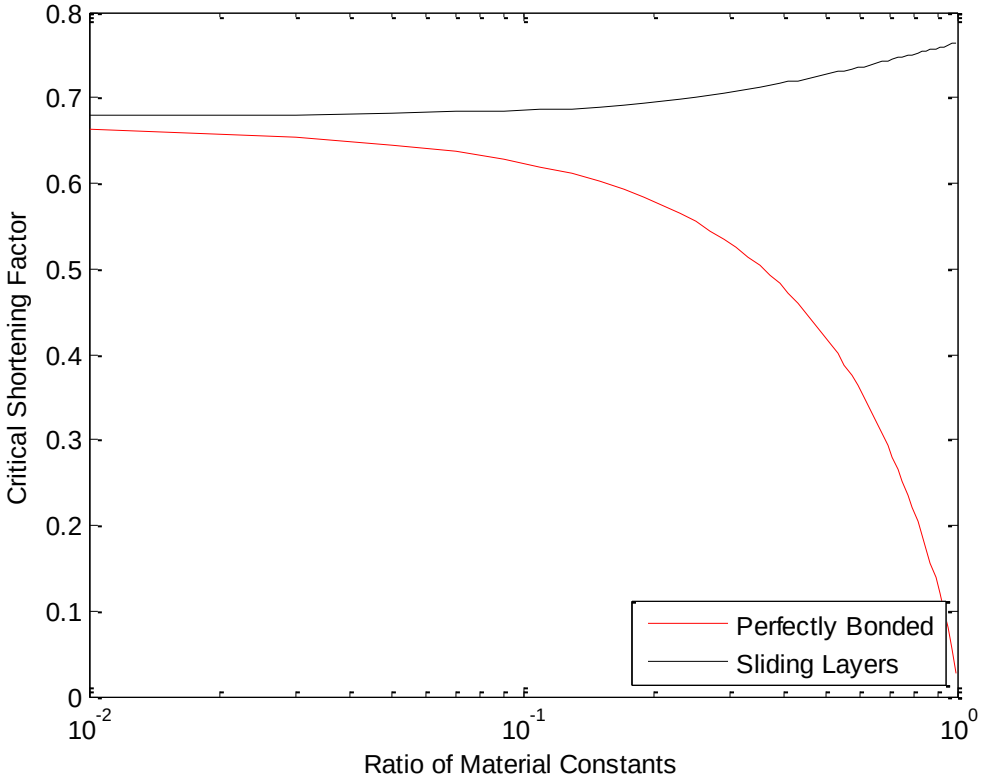


Fig. 14. Equi-biaxial, perfectly bonded and sliding layer with h_r / h_m=0.05 for the range of $0 < C_{10}^r / C_{10}^m < 1$

Re-plotting the same conditions of figure 13 but for the range of $0 < C_{10}^r / C_{10}^m < 1$ we get figure 14. This time a log scale is implemented on the x-axis as we're dealing with a ratio. The mirror image of the perfectly bonded curves (between fig.13 and 14) is expected as $C_{10}^r / C_{10}^m < 1$ just means the matrix layer now has a material constant greater than the reinforcement and so the matrix has effectively become the reinforcement. Again the rate of change in λ_{cr} is greater when C_{10}^r / C_{10}^m is closer to 1.

One difference between figure 13 and 14 is the critical shortening factor the curves level out at. For $C_{10}^r / C_{10}^m > 1$ (fig. 13), λ_{cr} levels out at about 0.98 whereas for $C_{10}^r / C_{10}^m < 1$ (fig. 14), λ_{cr} levels out at about 0.67. The reason for this large difference is that h_r/h_m is kept constant for both plots. If figure 14 was re-plotted but with h_r / h_m=20 (20 is the inverse of 0.05) the λ_{cr} levels out 0.97. This is still not exactly the same but a lot closer than before. This shows that even although it might make sense that simply inversing both the material constant ratio and the layer thickness ratio we will get the same material, this is not actually the case when using the characteristic equations.

Another difference of figure 13 and 14 is the curves for the sliding layers. In figure 14 the sliding curve starts at a λ_{cr} of 0.69 before rising slightly to 0.77. This differs with the perfectly bonded curve which drops all the way to a λ_{cr} of 0 (fig. 14) before rising back up again (fig. 13). Again the difference between the perfectly bonded and the sliding layer curves becomes greater as C_{10}^r/C_{10}^m tends to 1.

5.3 Critical Shortening Factor against Layer Thickness Ratio

Now we shall look into the effects of altering the layer thickness ratio (h_r / h_m) (fig. 15). Let us first examine the curve of perfectly bonding. The main characteristic of this curve is the horizontal part in the range $0 \leq h_r/h_m \leq 0.07$. This flat horizontal section of the curve occurs at a low h_r/h_m. At a low h_r/h_m the reinforcements are relatively far apart and the reinforcement layers are very thin compared to the matrix. When the reinforcements are far apart there are no interactions between reinforcements and effectively becomes just a matrix with one reinforcement layer. Once h_r/h_m reach a certain point (0.07 for this case) the reinforcements are close enough to interact and the stiffness of the composite increases and the critical shortening factor increase.

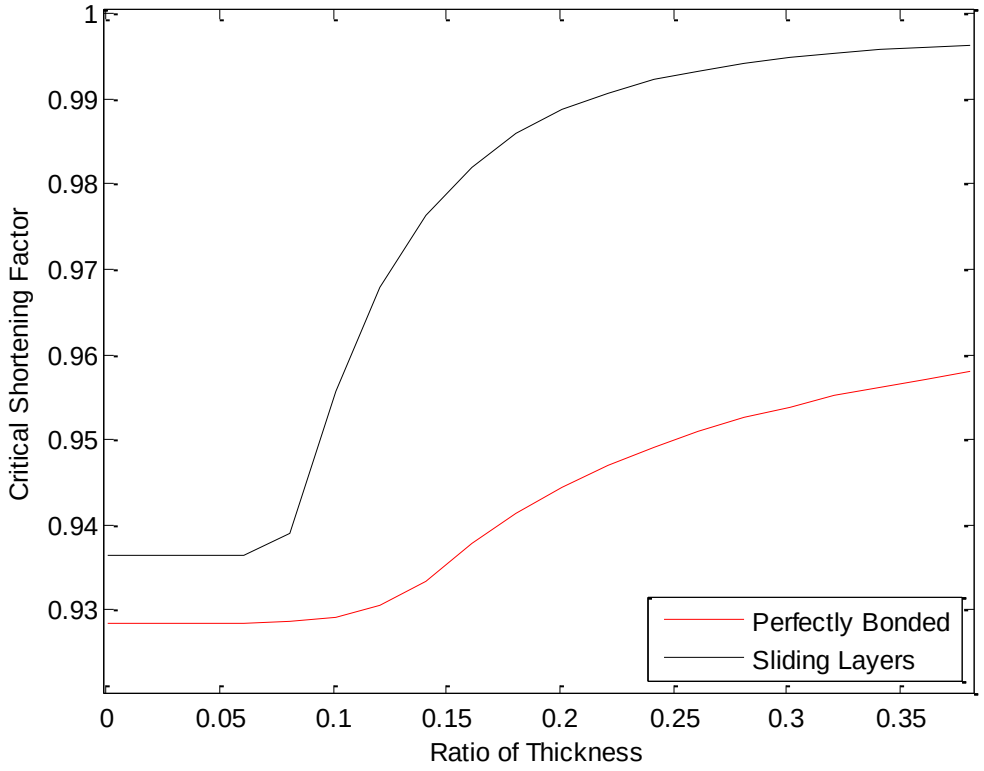


Fig. 15. Equi-biaxial, perfectly bonded and sliding layers with $C_{10}^r/\,C_{10}^m = 20$ at $0.01 \leq h_r/h_m \leq 0.4$

As we continue increasing h_r/h_m (fig. 16) the λ_{cr} increases to a maximum of 0.965 which occurs at $h_r/h_m=1$. At a layer thickness ratio of one the matrix layers and the reinforcement layers has the same thickness. Moreover, the maximum λ_{cr} always seems to occur at $h_r/h_m=1$ whatever the material constant ratio. As the reinforcement continues to get thicker relative to the matrix the λ_{cr} continues to decrease and will eventually level out at about 0.5 (for this case).

We shall inspect the curves for sliding layers now. In figure 15 the curve has the same horizontal part at $0 \leq h_r/h_m \leq 0.07$. But as the thickness ratio increases, the difference between the sliding layers and perfectly bonded curves also increases. This continues onto figure 16, in fact the curve for the sliding layers once over $h_r/h_m = 1$ is one straight horizontal line. This does not seem to make sense as this means a change in ratio thickness doesn't affect the critical shortening factor. A possible reason for this is because of mode 1 curving back upwards as mentioned in chapter 5.2. This only happens with mode 1 so let us take a look at the λ_{cr} against h_r/h_m graph without taking mode 1 into account.

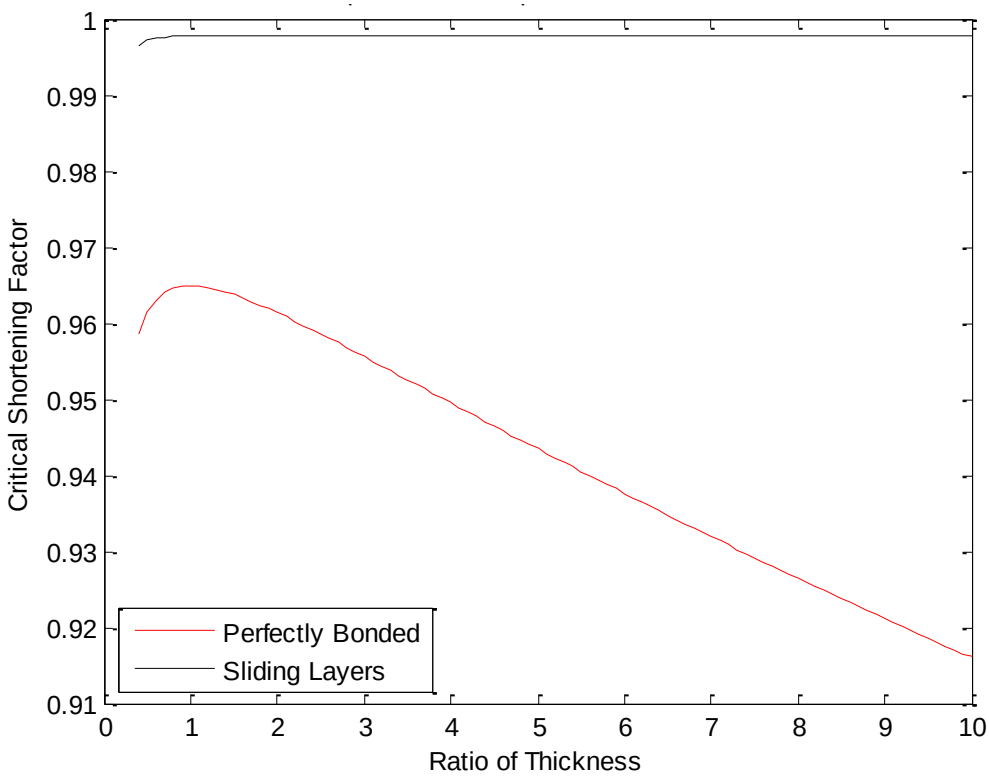


Fig. 16. Equi-biaxial, perfectly bonded and sliding layers with $C_{10}^r / C_{10}^m = 20$ at $0.4 \leq h_r / h_m \leq 10$

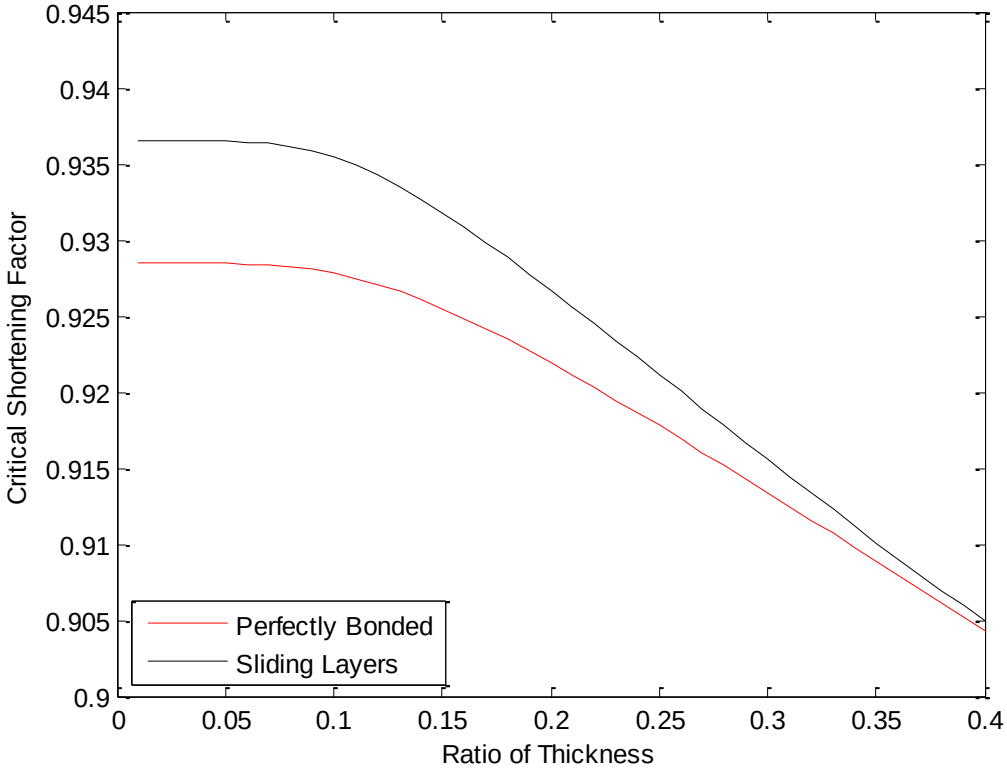


Fig 17. Mode 2. Equi-biaxial, perfectly bonded and sliding layers with $C_{10}^{r}/\ C_{10}^{m} = 20$ and $0.01 \leq \ h_r / h_m \leq 0.4$

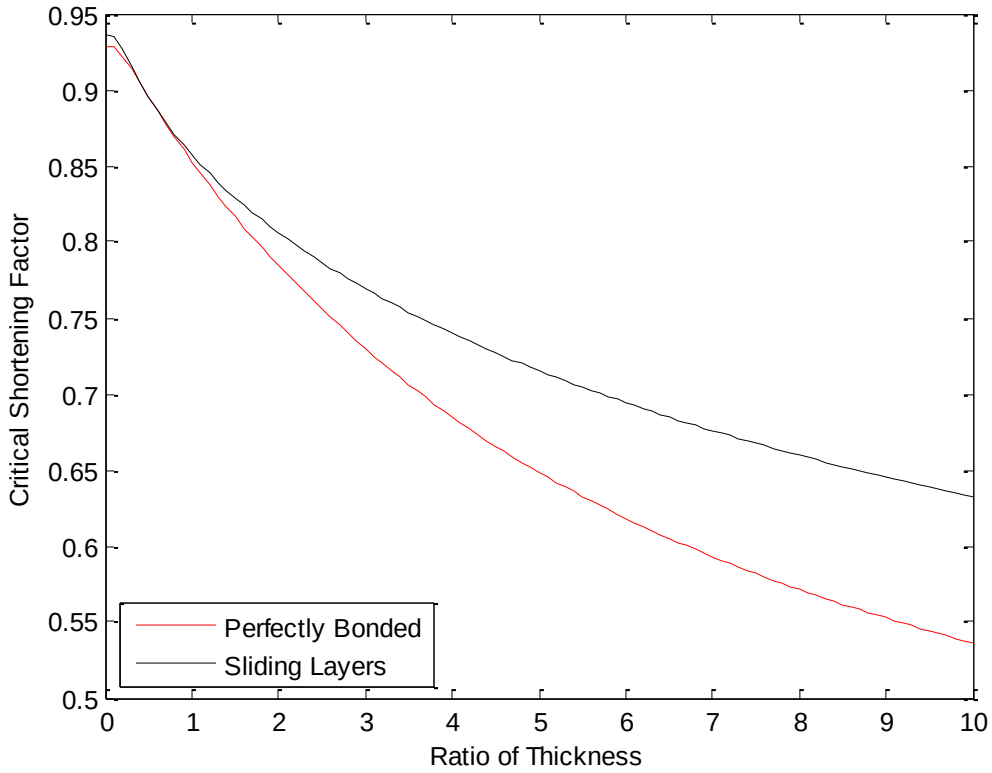


Fig. 18. Mode 2. Equi-biaxial, perfectly bonded and sliding layers with $C_{10}^{r}/\ C_{10}^{m} = 20$ at $0.4 \leq \ h_r / h_m \leq 10$

Figure 17 and 18 shows the same graph as figure 15 and 16 but does not take mode 1 into consideration. Again in figure 17 the curves have the characteristic horizontal line and at the same λ_{cr} as figure 15. However, instead of curving upwards the line curves downwards. It continues downwards and as h_r/h_m becomes greater than 0.5 the difference between the sliding and perfectly bonded curves increases. In reality, mode 1 cannot simply be avoided but this problem highlighted by the sliding layers curve of figure 16 persists. The problem could be caused by errors in the Matlab code or simply the characteristic equations shouldn't be used for a large material thickness ratio. In any case the author recommends the usage of a typical range of material thickness ratio (i.e. $0 \leq h_r/h_m \leq 0.4$).

5.4 Uniaxial Compression

So far we have only considered a composite under equi-biaxial loading but let us now take a look at the case of uniaxial compression instead. Table 1 compares the critical shortening factor of uniaxial and equi-biaxial compression and is listed in descending order of critical shortening factor. The greatest λ_{cr} belongs to the case of equi-biaxial sliding. This is expected as there is more overall load in equi-biaxial loading so the material will buckle at a lower strain and hence a greater shortening factor (eqn. 1). Furthermore, it has already been mentioned that the λ_{cr} for sliding layers should be higher than that of the equivalent perfectly bonded curve. The next largest is for the case of equi-biaxial perfectly bonding and then the two uniaxial cases. This order remains for other sets of ratios.

Another point to note is that the normalised wavelength the composite loses stability (α_{cr}) is independent of the type of loading (table 1). This suggest that for a composite with a set $C_{10}^r/\ C_{10}^m$, h_r/h_m and interface, there is a "weak" α_{cr} in which the material

Type of Loading	**Type of Bonding**	λ_{cr}	α_{cr}
Equi-Biaxial	Sliding	0.9499	0.256
Equi-Biaxial	Perfect	0.9453	0.249
Uniaxial	Sliding	0.9258	0.256
Uniaxial	Perfect	0.9190	0.249

Table 1. Comparison of λ_{cr} with different types of loading and bonding for a material with $C_{10}^r/\ C_{10}^m = 30$ and h_r/h_m=0.065

will fail at, no matter the type of loading. Although, it might be of interest to check if this holds true for other types of loading such as in tri-axial compression.

The normalised wavelength is related to the half wavelength of stability loss (l_i) by equation (24). However, since there are l_i in two directions (l_1 and l_2) we need one of the l_i before the other one can be found, even if the normalized wavelength is known. Now in equi-biaxial loading, there are equal magnitude of force applied in the two directions it make sense that l_1 and l_2 are equal. On the other hand if the composite was under uniaxial loading, l_1 is probably not equal to l_2. Hence it is hypothesized that l_i does not only depend on α_r but also the type of loading being applied.

Figure 19 shows the graph of the critical normalized wavelength (α_{cr}) against the ratio of material constant. This graph was plotted for the case of equi-biaxial but as already stated the graph of uniaxial will look exactly the same. At small C_{10}^r / C_{10}^m (<7) the α_{cr} rises at a fast rate before slowly decreasing until a point is reach where α_{cr} suddenly drops to 0. This point at which it suddenly drops down to 0 seems to depend on the material thickness ratio, the greater the h_r / h_m the smaller the C_{10}^r / C_{10}^m at which it drops. As stated earlier, as the normalized wavelength tends to 0, l_i tends to infinity. However, if l_i is infinity it means that there are no modes of stability loss (such as the ones in

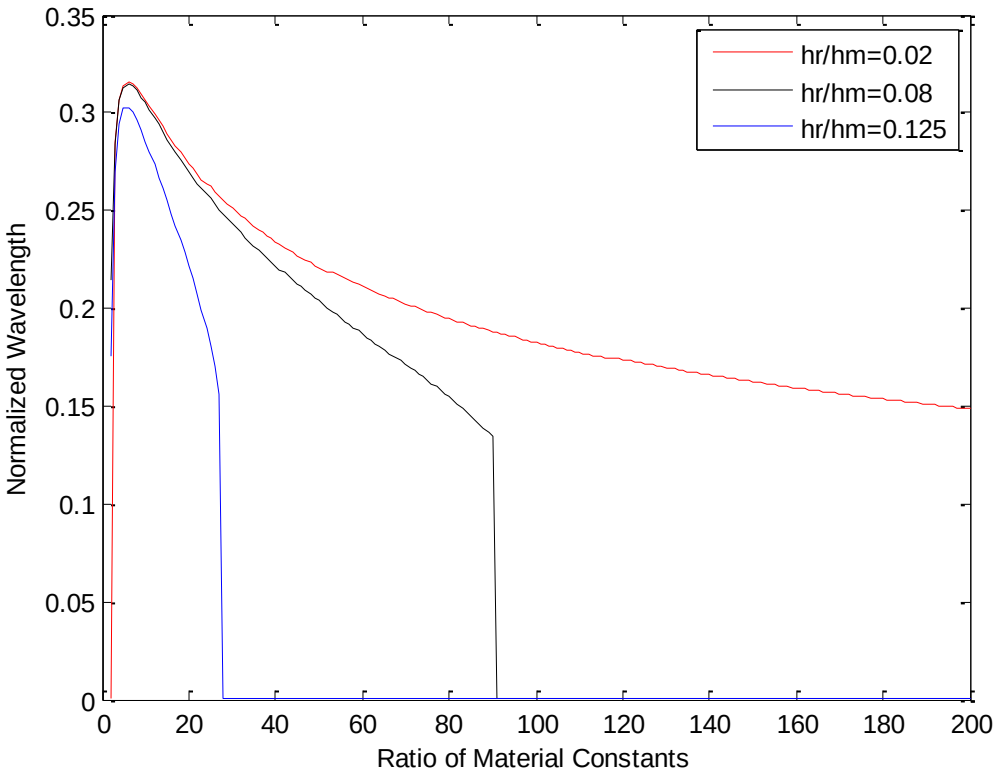


Fig 19. Critical normalized wavelength against Material Constant Ratio

fig. 6) present in the composite. Instead the reinforcements are straight, as if in the state of pre-loading. This does not make sense as this applies that the material is failing without any modes of stability loss. The reason for this is because there are two l_i, if α_r is equal to 0, either l_1 or l_2 can be equal to infinity (eqn. 24). So according to the characteristic equations, it is possible at failure for the composite to have two different modes of stability loss. One plane can remain unchanged while the other will contain one of the modes of stability loss. This goes against the idea in the previous paragraph that under equi-biaxial loading l_1 and l_2 are equal.

In general the (α_r vs. λ_1), (C_{10}^r/C_{10}^m vs. λ_{cr}) and (h_r/h_m vs λ_{cr}) graphs of the uniaxial case doesn't differ much compared to the corresponding equi-biaxial graph. The only real difference being the shortening factor of the uniaxial is slightly lower than that of the equi-biaxial, the reason having been already explained. As an example the uniaxial curve of figure 20 follows the same pattern as the equi-biaxial curve but at a slightly lower λ_{cr}.

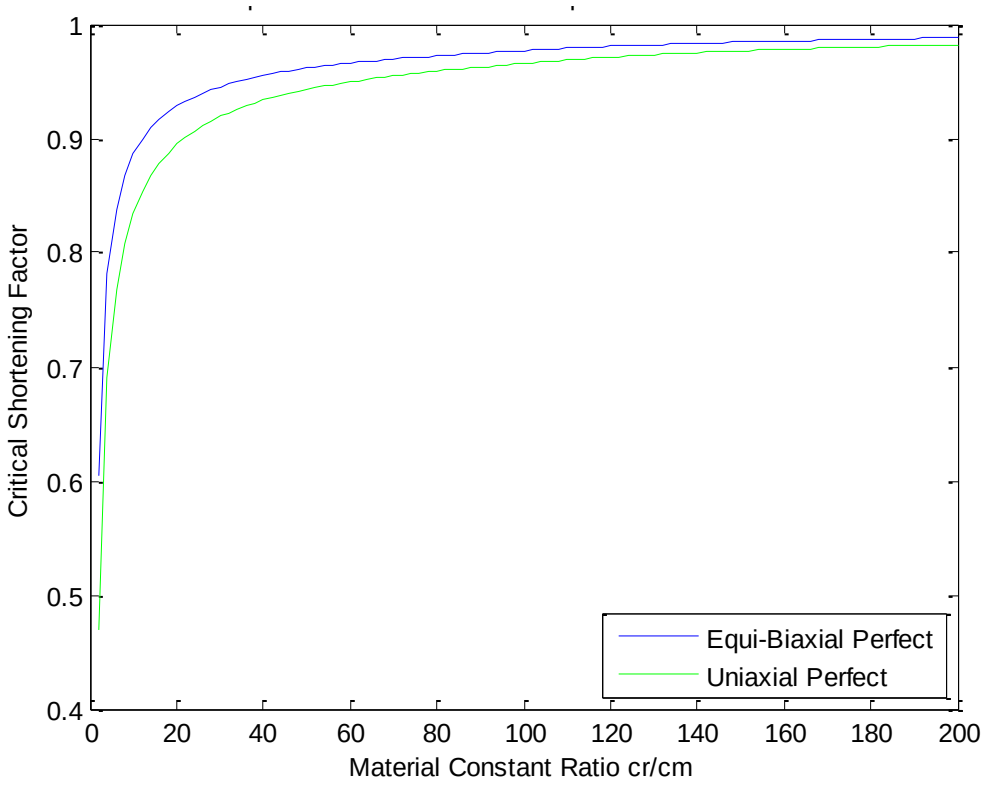


Fig 20. Equi-biaxial and uniaxial, perfectly bonded with h_r/h_m=0.08

6. Limitations

The main problem the author found was with the use of the fzero function. The code `fzero(f,g)` finds a root of equation `f` near the number `g`. Many times when computing for unusual ranges (e.g. $C_{10}^r / C_{10}^m < 1$ or $h_r/h_m>1$) or for normalized wave-length approaching 0, Matlab plots an unexpected graph with random spikes in the curve (fig. 21). These spikes always seemed to peak at a critical shortening factor of 1 or 0. This indicates that there are also roots at 0 and 1 for the characteristic equations. When these spikes occurred, iteration was used until a value of `g` was found which give a nice smooth curve. Another way to combat this problem was to enter `g` as a range (e.g. `[0.01,0.999]`) but in this case Matlab gave an error when computing $C_{10}^r / C_{10}^m = 1$ and various other times which the author does not understand. The author thought about letting the user change this value but this required many more lines of code and might confuse the user. In the end a set value of `g` was chosen depending on the mode, bonding and loading. These values can be found in appendix A to D

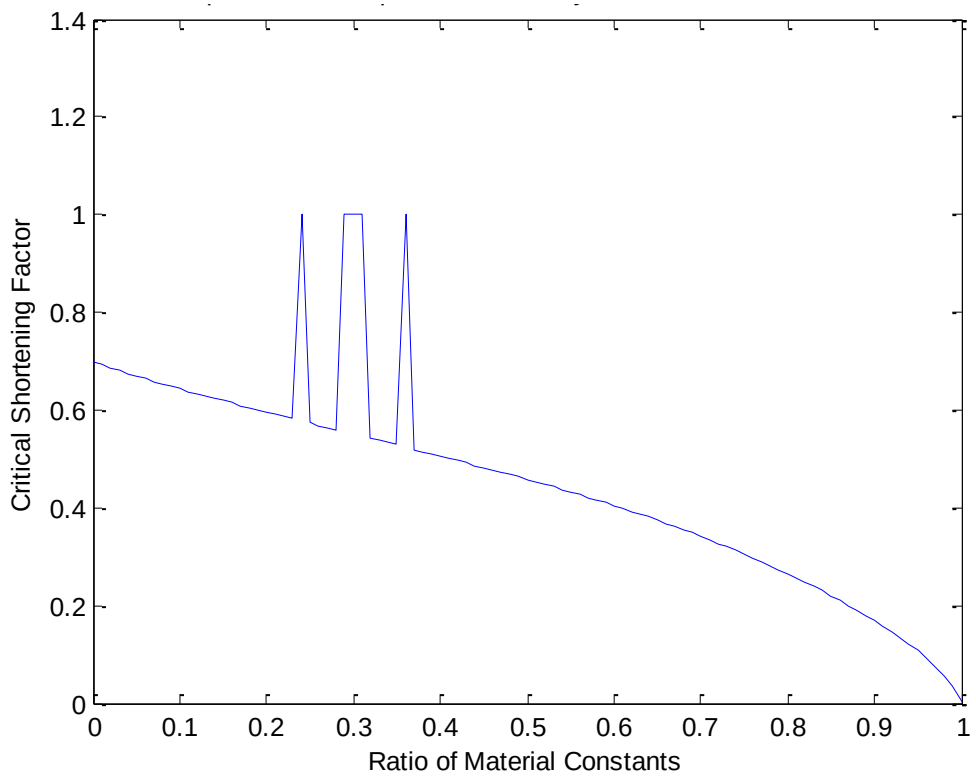


Fig. 21. A graph showing the problems encountered when using the fzero function

7. Conclusion

The compressive behaviour of an incompressible non-linear hyperelastic transversally isotropic material was investigated. Attention was given to the point of instability caused by microbuckling under uniaxial and equi-biaxial loadings for the case of large deformation. A Matlab program was written to solve the characteristic equations derived using the 3D stability theory and the piecewise-homogeneous medium model. The written program makes use of a graphical user interface allowing the user to easily change variables, types of bonding and types of loading for comparison.

The two types of bonding considered were used to establish the upper and lower bound of critical loading corresponding to perfectly bonded layers and sliding without friction layers. These two types of bonding were used to account for any adhesion breakdown that may occur between layers. It was found that the two types of bonding have similar characteristics except in some usual circumstances (e.g. fig. 14 and 16). In regards to the type of loading, the difference between uniaxial and equi-biaxial loading was found to be minimal. In fact the critical shortening factor occurs at the same normalized wavelength for both.

Moreover, the increase rate of the critical shortening factor was found to be much greater for small values of material constant ratio. On the other hand the increase rate of critical shortening factor is zero for small values of layer thickness ratio. Finally, according to the derived characteristic equations, it was found that simply inversing both the mentioned ratios one does not get the exact same material as would be expected.

8. Improvements and Suggestions

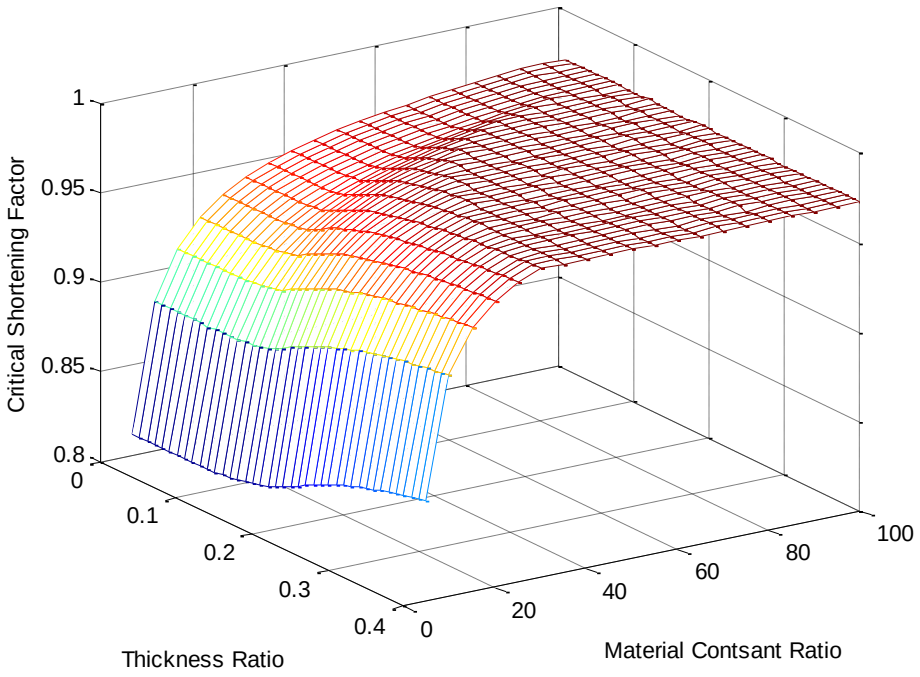


Fig 22. A 3D Plot

A feature that can be added to the GUI is to allow the user to plot 3D graphs such as the one in figure 22. The author attempted this but the 3D plot was considered to be difficult to read and hence will be hard to compare the changes due to loading and bonding. Additionally the plot of figure 22 shows the graph of λ_{cr} against C_{10}^r / C_{10}^m and how it changes with h_r / h_m but the author was not able to code Matlab so that it plotted the graph of λ_{cr} against h_r / h_m and how it changes with C_{10}^r / C_{10}^m . A way in which we can mimic a 3D graph is to plot a 2D graph but with two or more curves for other ratios such as figure 19. This is a good idea but many more lines of codes will still have to be written. Furthermore, the 3D graphs took a long time to compute as it has to calculate the critical shortening factor for all the possible combinations of ratios entered by the user and in the end, the 3D graphing function was not implemented into the Matlab program.

An area that can be investigated in future works is the effects of *intra*laminar defects. In this thesis we only looked at the effects of *inter*laminar defects in the form of the perfectly bonding and sliding without friction bonds. This is good but to fully analyse a composite material we should also look at defects *within* the layer such as matrix or reinforcement cracking. One way this can be done is to reduce the stiffness properties

of individual layers [2]. The accuracy of this is not known but in future works one might wish to consider the effects of both inter and intralaminar defects simultaneously.

Another area in which future works can delve into is the practical applicability of the model in this thesis. The considered material can be applied to any applications in which large deformation might take place such as tyres, force absorbers, hoses or maybe even fishing rods. Moreover, compressive stresses can occur at cracks or holes in the composite even if it is under tensile loading [1].

Depending on the application, materials will have to be specifically chosen for the matrix and reinforcement. For example, tyres used in tractors in a building site might use a reinforcement material such as steel to increase resistance to impact due to rocks and holes. On the other hand, tyres used in large aircrafts may use light materials along with a low material thickness ratio to reduce weight. There are many other factors to be considered when choosing materials to be used in a tyre such as the temperature and speed at which the tyre is operated at.

However, once a list of materials has been chosen one may compare these materials using the written Matlab program. If one requires a large critical shortening factor (i.e. a low critical strain) one may choose to have a high material constant ratio (fig. 13) or a high thickness ratio (fig. 15). The cost of the materials should also be taken in to account. Knowing that the rate of increase of critical shortening factor slows down at high material constants, one should consider if the increase in cost is worth the small increase in critical shortening factor. Additionally, if one requires a very high accuracy of critical shortening factor (such as in applications in which lives are at risk) where the upper and lower bounds are required to be as close to together as possible, one should choose a high material constant ratio and a low layer thickness ratio.

As mentioned in the introduction the compressive strength of composites are 30-40% lower than the tensile strength making it the limiting factor of many applications. If the compressive strength of composites were to be improved the ceiling of what can be achieved can be pushed further upwards.

Appendix A: Matlab function for perfectly bonded under equi-biaxial loading

```
function [s1,s2,s3,s4]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm)
am1=zeros(1,length(ar1)); %Preallocate arrays for the matrix
normalised wavelength for the four modes.
am2=zeros(1,length(ar2)); %This is just to make the code run faster
am3=zeros(1,length(ar3));
am4=zeros(1,length(ar4));
s1=zeros(1,length(ar1)); %Preallocate arrays for the shortening factor
s2=zeros(1,length(ar2));
s3=zeros(1,length(ar3));
s4=zeros(1,length(ar4));

for i=1:length(ar1); %Mode 1
    am1(i)=(hm/hr).*ar1(i);  %Equation 20 to reduce the number of
variables to 2
    %The corresponding characteristic equation:
    f=@(s1) -s1^-3*(1+s1^6)^2*(1-cr/cm)^2*tanh(ar1(i).*s1^-
3)*tanh(am1(i).*s1^-3)-4*s1^3*(1-
cr/cm)^2*tanh(ar1(i))*tanh(am1(i))+(2-
(1+s1^6)*(cr/cm))^2*tanh(ar1(i).*s1^-3)*tanh(am1(i))+(1+s1^6-
2*cr/cm)^2*tanh(ar1(i))*tanh(am1(i)*s1^-3)+(1-
s1^6)^2*(cr/cm)*(tanh(ar1(i))*tanh(ar1(i).*s1^-
3)+tanh(am1(i))*tanh(am1(i)*s1^-3));
    s1(i)=fzero(f,0.8); %Find the shortening factor
end
for i=1:length(ar2); %Mode 2
    am2(i)=(hm/hr).*ar2(i);
    f=@(s2) -s2^-3*(1+s2^6)^2*(1-cr/cm)^2*tanh(ar2(i).*s2^-
3)*coth(am2(i).*s2^-3)-4*s2^3*(1-
cr/cm)^2*tanh(ar2(i))*coth(am2(i))+(2-
(1+s2^6)*(cr/cm))^2*tanh(ar2(i).*s2^-3)*coth(am2(i))+(1+s2^6-
2*cr/cm)^2*tanh(ar2(i))*coth(am2(i)*s2^-3)+(1-
s2^6)^2*(cr/cm)*(tanh(ar2(i))*tanh(ar2(i).*s2^-
3)+coth(am2(i))*coth(am2(i)*s2^-3));
    s2(i)=fzero(f,0.8);
end
for i=1:length(ar3) %Mode 3
    am3(i)=(hm/hr).*ar3(i);
    f=@(s3) -s3^-3*(1+s3^6)^2*(1-cr/cm)^2*coth(ar3(i).*s3^-
3)*coth(am3(i).*s3^-3)-4*s3^3*(1-
cr/cm)^2*coth(ar3(i))*coth(am3(i))+(2-
(1+s3^6)*(cr/cm))^2*coth(ar3(i).*s3^-3)*coth(am3(i))+(1+s3^6-
2*cr/cm)^2*coth(ar3(i))*coth(am3(i)*s3^-3)+(1-
s3^6)^2*(cr/cm)*(coth(ar3(i))*coth(ar3(i).*s3^-
3)+coth(am3(i))*coth(am3(i)*s3^-3));
    s3(i)=fzero(f,0.6);
end
for i=1:length(ar4) %Mode 4
    am4(i)=(hm/hr).*ar4(i);
    f=@(s4) -s4^-3*(1+s4^6)^2*(1-cr/cm)^2*coth(ar4(i).*s4^-
3)*tanh(am4(i).*s4^-3)-4*s4^3*(1-
cr/cm)^2*coth(ar4(i))*tanh(am4(i))+(2-
(1+s4^6)*(cr/cm))^2*coth(ar4(i).*s4^-3)*tanh(am4(i))+(1+s4^6-
2*cr/cm)^2*coth(ar4(i))*tanh(am4(i)*s4^-3)+(1-
s4^6)^2*(cr/cm)*(coth(ar4(i))*coth(ar4(i).*s4^-
3)+tanh(am4(i))*tanh(am4(i)*s4^-3));
    s4(i)=fzero(f,0.6);
end
```

Appendix B: Matlab function for sliding layers under equi-biaxial loading

```
function [s1,s2,s3,s4]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm)
am1=zeros(1,length(ar1)); %Preallocate arrays for the matrix
normalised wavelength for the four modes.
am2=zeros(1,length(ar2)); %This is just to make the code run faster
am3=zeros(1,length(ar3));
am4=zeros(1,length(ar4));
s1=zeros(1,length(ar1)); %Preallocate arrays for the shortening factor
for he four modes.
s2=zeros(1,length(ar2));
s3=zeros(1,length(ar3));
s4=zeros(1,length(ar4));

for i=1:length(ar1); %Mode 1
    am1(i)=(hm/hr).*ar1(i);  %Equation 20 to reduce the number of
variables to 2
    %The corresponding characteristic equation:
    f=@(s1) 4*s1^3*((cr/cm)*tanh(ar1(i))+tanh(am1(i)))-
(1+s1^6)^2*((cr/cm)*tanh(ar1(i)/s1^3)+tanh(am1(i)/s1^3));
    s1(i)=fzero(f,0.6091); %Find the shortening factor
end
for i=1:length(ar2); %Mode 2
    am2(i)=(hm/hr).*ar2(i);
    f=@(s2) 4*s2^3*((cr/cm)*tanh(ar2(i))+coth(am2(i)))-
(1+s2^6)^2*((cr/cm)*tanh(ar2(i)/s2^3)+coth(am2(i)/s2^3));
    s2(i)=fzero(f,0.6091);
end
for i=1:length(ar3) %Mode 3
    am3(i)=(hm/hr).*ar3(i);
    f=@(s3) 4*s3^3*((cr/cm)*coth(ar3(i))+coth(am3(i)))-
(1+s3^6)^2*((cr/cm)*coth(ar3(i)/s3^3)+coth(am3(i)/s3^3));
    s3(i)=fzero(f,0.6091);
end
for i=1:length(ar4) %Mode 4
    am4(i)=(hm/hr).*ar4(i);
    f=@(s4) 4*s4^3*((cr/cm)*coth(ar4(i))+tanh(am4(i)))-
(1+s4^6)^2*((cr/cm)*coth(ar4(i)/s4^3)+tanh(am4(i)/s4^3));
    s4(i)=fzero(f,0.6091);
end
```

Appendix C: Matlab function for perfectly bonded under uniaxial loading

```
Function [s1,s2,s3,s4]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,
hr,hm);
am1=zeros(1,length(ar1)); %Preallocate arrays for the matrix
normalised wavelength for the four modes.
am2=zeros(1,length(ar2)); %This is just to make the code run faster
am3=zeros(1,length(ar3));
am4=zeros(1,length(ar4));
s1=zeros(1,length(ar1)); %Preallocate arrays for the shortening factor
for he four modes.
s2=zeros(1,length(ar2));
s3=zeros(1,length(ar3));
s4=zeros(1,length(ar4));

for i=1:length(ar1);  %Mode 1
```

```
    am1(i)=(hm/hr).*ar1(i); %Equation 20 to reduce the number of
variables to 2
    %The corresponding characteristic equation:
    f=@(s1) -s1^-2*(1+s1^4)^2*(1-cr/cm)^2*tanh(ar1(i)*s1^-
2)*tanh(am1(i)*s1^-2)-4*s1^2*(1-
(cr/cm))^2*tanh(ar1(i))*tanh(am1(i))+(2-
(1+s1^4)*(cr/cm))^2*tanh(ar1(i)*s1^-2)*tanh(am1(i))+(1+s1^4-
2*(cr/cm))^2*tanh(ar1(i))*tanh(am1(i)*s1^-2)+(1-
s1^4)^2*(cr/cm)*(tanh(ar1(i))*tanh(ar1(i)*s1^-
2)+tanh(am1(i))*tanh(am1(i)*s1^-2));
    s1(i)=fzero(f,0.8); %Find the shortening factor
end
for i=1:length(ar2);  %Mode 2
    am2(i)=(hm/hr).*ar2(i);
    f=@(s2) -s2^-2*(1+s2^4)^2*(1-cr/cm)^2*tanh(ar2(i)*s2^-
2)*coth(am2(i)*s2^-2)-4*s2^2*(1-
(cr/cm))^2*tanh(ar2(i))*coth(am2(i))+(2-
(1+s2^4)*(cr/cm))^2*tanh(ar2(i)*s2^-2)*coth(am2(i))+(1+s2^4-
2*(cr/cm))^2*tanh(ar2(i))*coth(am2(i)*s2^-2)+(1-
s2^4)^2*(cr/cm)*(tanh(ar2(i))*tanh(ar2(i)*s2^-
2)+coth(am2(i))*coth(am2(i)*s2^-2));
    s2(i)=fzero(f,0.8);
end
for i=1:length(ar3) %Mode 3
    am3(i)=(hm/hr).*ar3(i);
    f=@(s3) -s3^-2*(1+s3^4)^2*(1-cr/cm)^2*coth(ar3(i).*s3^-
2)*coth(am3(i).*s3^-2)-4*s3^2*(1-
cr/cm)^2*coth(ar3(i))*coth(am3(i))+(2-
(1+s3^4)*(cr/cm))^2*coth(ar3(i).*s3^-2)*coth(am3(i))+(1+s3^4-
2*cr/cm)^2*coth(ar3(i))*coth(am3(i)*s3^-2)+(1-
s3^4)^2*(cr/cm)*(coth(ar3(i))*coth(ar3(i).*s3^-
2)+coth(am3(i))*coth(am3(i)*s3^-2));
    s3(i)=fzero(f,0.3);
end
for i=1:length(ar4)  %Mode 4
    am4(i)=(hm/hr).*ar4(i);
    f=@(s4) -s4^-2*(1+s4^4)^2*(1-cr/cm)^2*coth(ar4(i).*s4^-
2)*tanh(am4(i).*s4^-2)-4*s4^2*(1-
cr/cm)^2*coth(ar4(i))*tanh(am4(i))+(2-
(1+s4^4)*(cr/cm))^2*coth(ar4(i).*s4^-2)*tanh(am4(i))+(1+s4^4-
2*cr/cm)^2*coth(ar4(i))*tanh(am4(i)*s4^-2)+(1-
s4^4)^2*(cr/cm)*(coth(ar4(i))*coth(ar4(i).*s4^-
2)+tanh(am4(i))*tanh(am4(i)*s4^-2));
    s4(i)=fzero(f,0.3);
end
```

Appendix D: Matlab function for sliding layers under uniaxial loading

```
function [s1,s2,s3,s4]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,
hr,hm)
am1=zeros(1,length(ar1)); %Preallocate arrays for the matrix
normalised wavelength for the four modes.
am2=zeros(1,length(ar2)); %This is just to make the code run faster
am3=zeros(1,length(ar3));
am4=zeros(1,length(ar4));
s1=zeros(1,length(ar1)); %Preallocate arrays for the shortening factor
for he four modes.
s2=zeros(1,length(ar2));
s3=zeros(1,length(ar3));
s4=zeros(1,length(ar4));
```

```
for i=1:length(ar1); %Mode 1
    am1(i)=(hm/hr).*ar1(i);  %Equation 20 to reduce the number of
variables to 2
    %The corresponding characteristic equation:
    f=@(s1) 4*s1^2*((cr/cm)*tanh(ar1(i))+tanh(am1(i)))-
(1+s1^4)^2*((cr/cm)*tanh(ar1(i)/s1^2)+tanh(am1(i)/s1^2));
    s1(i)=fzero(f,0.8); %Find the shortening factor
end
 for i=1:length(ar2); %Mode 2
    am2(i)=(hm/hr).*ar2(i);
    f=@(s2) 4*s2^2*((cr/cm)*tanh(ar2(i))+coth(am2(i)))-
(1+s2^4)^2*((cr/cm)*tanh(ar2(i)/s2^2)+coth(am2(i)/s2^2));
    s2(i)=fzero(f,0.8);
end
for i=1:length(ar3) %Mode 3
    am3(i)=(hm/hr).*ar3(i);
    f=@(s3) 4*s3^2*((cr/cm)*coth(ar3(i))+coth(am3(i)))-
(1+s3^4)^2*((cr/cm)*coth(ar3(i)/s3^2)+coth(am3(i)/s3^2));
    s3(i)=fzero(f,0.6);
end
for i=1:length(ar4) %Mode 4
    am4(i)=(hm/hr).*ar4(i);
    f=@(s4) 4*s4^2*((cr/cm)*coth(ar4(i))+tanh(am4(i)))-
(1+s4^4)^2*((cr/cm)*coth(ar4(i)/s4^2)+tanh(am4(i)/s4^2));
    s4(i)=fzero(f,0.6);
end
```

Appendix E: Main matlab code part 1

```
function varargout = test(varargin)
% TEST M-file for test.fig
%      TEST, by itself, creates a new TEST or raises the existing
%      singleton*.
%      H = TEST returns the handle to a new TEST or the handle to
%      the existing singleton*.
%      TEST('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TEST.M with the given input
arguments.
%      TEST('Property','Value',...) creates a new TEST or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before test_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to test_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% InputCR the above text to modify the response to help test
% Last Modified by GUIDE v2.5 11-Apr-2012 16:19:13
% Begin initialization code - DO NOT INPUTCR
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @test_OpeningFcn, ...
                   'gui_OutputFcn',  @test_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT INPUTCR

% --- Executes just before test is made visible.
function test_OpeningFcn(hObject, ~, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to test (see VARARGIN)
% Choose default command line output for test
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
global k
clc
k=0; %used later to make the table work
% UIWAIT makes test wait for user response (see UIRESUME)
% uiwait(handles.figure);
% --- Outputs from this function are returned to the command line.
function varargout = test_OutputFcn(~, ~, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
function InputCR_Callback(~, ~, ~)
% hObject    handle to InputCR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of InputCR as text
%        str2double(get(hObject,'String')) returns contents of InputCR
as a double
% --- Executes during object creation, after setting all properties.
function InputCR_CreateFcn(hObject, ~, ~)
% hObject    handle to InputCR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: InputCR controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function InputHR_Callback(hObject, ~, ~)
% hObject    handle to InputHR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of InputHR as text
%        str2double(get(hObject,'String')) returns contents of InputHR
as a double
% --- Executes during object creation, after setting all properties.
function InputHR_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to InputHR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function InputCM_Callback(hObject, eventdata, handles)
% hObject    handle to InputCM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of InputCM as text
%        str2double(get(hObject,'String')) returns contents of InputCM
as a double
% --- Executes during object creation, after setting all properties.
function InputCM_CreateFcn(hObject, eventdata, handles)
% hObject    handle to InputCM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function InputHM_Callback(hObject, eventdata, handles)
% hObject    handle to InputHM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of InputHM as text
%        str2double(get(hObject,'String')) returns contents of InputHM
as a double
% --- Executes during object creation, after setting all properties.
function InputHM_CreateFcn(hObject, eventdata, handles)
% hObject    handle to InputHM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on selection change in popupmenu2.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
%        popupmenu1
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
 if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
 end
% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu2
% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
 if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
 end
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
as a double
% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu3
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu3
% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(~, ~, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Clear all the input boxes
set(handles.InputCR,'String','')
set(handles.InputCM,'String','')
set(handles.InputHR,'String','')
set(handles.InputHM,'String','')
set(handles.edit5,'String','');
set(handles.edit6,'String','');
set(handles.edit7,'String','');
```

Appendix F: Main matlab code part 2

```
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(~, ~, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
global k

%get what the user wants to be plotted from the drop down menus
loading=get(handles.popupmenu3,'Value');
var=get(handles.popupmenu1,'Value');
bond=get(handles.popupmenu2,'Value');

%get the variables the user has entered
cr= str2double(get(handles.InputCR,'String'));
cm= str2double(get(handles.InputCM,'String'));
hr= str2double(get(handles.InputHR,'String'));
hm= str2double(get(handles.InputHM,'String'));
minar=str2double(get(handles.edit5,'String'));
maxar=str2double(get(handles.edit6,'String'));
incar=str2double(get(handles.edit7,'String'));

%set the normalised wavelength, one for each mode(but only need one
tbh)
ar1=minar:incar:maxar;
ar2=minar:incar:maxar;
ar3=minar:incar:maxar;
ar4=minar:incar:maxar;

%see if any of the input boxes are 0 for error messages
emptyCR=isempty(get(handles.InputCR,'String'));
emptyCM=isempty(get(handles.InputCM,'String'));
emptyHR=isempty(get(handles.InputHR,'String'));
emptyHM=isempty(get(handles.InputHM,'String'));
emptyminar=isempty(get(handles.edit5,'String'));
emptymaxar=isempty(get(handles.edit6,'String'));
%make sure the min.wavelength is smaller than the the max. wavenlength
if minar >= maxar
    msgbox('The minimum normalised wavelength has to be smaller than
the maximum normalised wavelength')
    return
end


if loading==1 && var==1 && bond==1 %Plot Uniaxial, Normalized
Wavelength, Perfect
    %Error message if input boxes haven't been entered
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
        return
    end
    %Get the data from the function

[s1,s2,s3,s4]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
```

```
    figure
    plot(ar1,s1,'r*',ar2,s2,'b',ar3,s3,'ks',ar4,s4,'g');
    legend('Mode 1','Mode 2','Mode 3','Mode 4','Location','SouthEast')
    title(['Uniaxial Compression Perfectly Bonded
Cr/Cm=',num2str(cr/cm),'   hr/hm=',num2str(hr/hm)])
    xlabel('Normalized Wavelength')
    ylabel('Shortening Factor')

    %calculate and display the crit. shortening factor and crit.
wavelength
    k=k+1;
    row = get(handles.uitable1,'Data');
    sall=[s1 s2];
    arall=[ar1 ar2];
    [CritShort,locationofCritShort]=max(sall);
    Critar=arall(locationofCritShort);
    row{k,1}='Uniaxial Compression';
    row{k,2}='Perfect';
    row{k,3}=cr/cm;
    row{k,4}=hr/hm;
    row{k,5}=CritShort;
    row{k,6}=Critar;
    set(handles.uitable1,'Data',row)


elseif loading==1 && var==1 && bond==2  %Uniaxial, Normalised
Wavelength, Sliding
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
        return
    end
    if minar<0.1
            questminar = questdlg('A value of greater or equal to 0.1
for normalised wavelngth is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
            switch questminar
                case 'OK'
                case 'Continue Anyway'

[s1,s2,s3,s4]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

                    figure

plot(ar1,s1,'r*',ar2,s2,'b',ar3,s3,'ks',ar4,s4,'g');
                    legend('Mode 1','Mode 2','Mode 3','Mode
4','Location','SouthEast')
                    title(['Uniaxial Compression Sliding Layers
Cr/Cm=',num2str(cr/cm),'   hr/hm=',num2str(hr/hm)])
                    xlabel('Normalized Wavelength')
                    ylabel('Shortening Factor')

                    k=k+1;
                    row = get(handles.uitable1,'Data');
                    sall=[s1 s2];
                    arall=[ar1 ar2];
                    [CritShort,locationofCritShort]=max(sall);
```

```
                    Critar=arall(locationofCritShort);
                    row{k,1}='Uniaxial Compression';
                    row{k,2}='Sliding Layers';
                    row{k,3}=cr/cm;
                    row{k,4}=hr/hm;
                    row{k,5}=CritShort;
                    row{k,6}=Critar;
                    set(handles.uitable1,'Data',row)
            end
    else

[s1,s2,s3,s4]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

        figure
        plot(ar1,s1,'r*',ar2,s2,'b',ar3,s3,'ks',ar4,s4,'g');
        legend('Mode 1','Mode 2','Mode 3','Mode
4','Location','SouthEast')
        title(['Uniaxial Compression Sliding Layers
Cr/Cm=',num2str(cr/cm),'   hr/hm=',num2str(hr/hm)])
        xlabel('Normalized Wavelength')
        ylabel('Shortening Factor')

        k=k+1;
        row = get(handles.uitable1,'Data');
        sall=[s1 s2];
        arall=[ar1 ar2];
        [CritShort,locationofCritShort]=max(sall);
        Critar=arall(locationofCritShort);
        row{k,1}='Uniaxial Compression';
        row{k,2}='Sliding Layers';
        row{k,3}=cr/cm;
        row{k,4}=hr/hm;
        row{k,5}=CritShort;
        row{k,6}=Critar;
        set(handles.uitable1,'Data',row)
    end


elseif loading==1 && var==1 && bond==3   %Uniaxial, Normalised
Wavelength, Both
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
        return
    end
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
normalised wavelngth is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'

[s1perf,s2perf]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1slide,s2slide]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

                figure
```

```
plot(ar1,s1perf,'r',ar1,s1slide,'k',ar2,s2perf,'b*',ar2,s2slide,'g*')
                legend('Mode 1 Perfectly Bonded','Mode 1 Sliding
Layers','Mode 2 Perfectly Bonded','Mode 2 Sliding Layers
','Location','SouthWest')
                title(['Uniaxial Compression
Cr/Cm=',num2str(cr/cm),'   hr/hm=',num2str(hr/hm)])
                xlabel('Normalized Wavelength')
                ylabel('Shortening Factor')
                 %axis([0.17 0.8 0.8 1])
        end
    else

[s1perf,s2perf]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1slide,s2slide]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

        figure

plot(ar1,s1perf,'r',ar1,s1slide,'k',ar2,s2perf,'b*',ar2,s2slide,'g*')
        legend('Mode 1 Perfectly Bonded','Mode 1 Sliding Layers','Mode
2 Perfectly Bonded','Mode 2 Sliding Layers ','Location','SouthWest')
        title(['Uniaxial Compression    Cr/Cm=',num2str(cr/cm),'
hr/hm=',num2str(hr/hm)])
        xlabel('Normalized Wavelength')
        ylabel('Shortening Factor')
        %axis([0.17 0.8 0.8 1])
    end


elseif loading==1 && var==2 && bond==1 %Uniaxial, Material Constant
ratio, Perfect
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values in thickness ratio and normalised
wavelength input boxes.')
        return
    end
    c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
    c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
    scrit=zeros(1,length(c));
    for i=1:length(c)
        cr=c(i);
        cm=1;
        [s1,s2]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
        scrit(i)=max([s1 s2]);
    end
    figure
    plot(c,scrit)
    title(['Uniaxial Compression Perfectly Bonded
hr/hm=',num2str(hr/hm)])
    xlabel('Ratio of Material Constants')
    ylabel('Critical Shortening Factor')


elseif loading==1 && var==2 && bond==2  %Uniaxial, Material constant
ratio, Sliding
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
```

```
        msgbox('Please enter values in thickness ratio and normalised
wavelength input boxes.')
        return
    end
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
normalised wavelngth is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'
                c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});

c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
                scrit=zeros(1,length(c));
                for i=1:length(c)
                    cr=c(i);
                    cm=1;

[s1,s2]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
                    scrit(i)=max([s1 s2]);
                end
                figure
                plot(c,scrit)
                title(['Uniaxial Compression Sliding Layers
hr/hm=',num2str(hr/hm)])
                xlabel('Ratio of Material Constants')
                ylabel('Critical Shortening Factor')
        end
    else
        c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
        c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
        scrit=zeros(1,length(c));
        for i=1:length(c)
            cr=c(i);
            cm=1;

[s1,s2]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
            scrit(i)=max([s1 s2]);
        end
        figure
        plot(c,scrit)
        title(['Uniaxial Compression Sliding Layers
hr/hm=',num2str(hr/hm)])
        xlabel('Ratio of Material Constants')
        ylabel('Critical Shortening Factor')
    end


elseif loading==1 && var==2 && bond ==3  %Uniaxial, Material Constant
Ratio, Both
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values in thickness ratio and normalised
wavelength input boxes.')
        return
    end
    if minar<0.1
```

```
        questminar = questdlg('A value of greater or equal to 0.1 for
normalised wavelngth is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'
                c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});

c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
                scritperf=zeros(1,length(c));
                scritslide=zeros(1,length(c));
                for i=1:length(c)
                    cr=c(i);
                    cm=1;

[s1perf,s2perf,s3perf,s4perf]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,
cr,cm,hr,hm);

[s1slide,s2slide,s3slide,s4slide]=UniaxialSlidingLayers(ar1,ar2,ar3,ar
4,cr,cm,hr,hm);
                    scritperf(i)=max([s1perf s2perf s3perf s4perf]);
                    scritslide(i)=max([s1slide s2slide s3slide
s4slide]);
                end
                figure
                plot(c,scritperf,'r',c,scritslide,'k')
                legend('Perfectly Bonded','Sliding
Layers','Location','SouthEast')
                title(['Uniaxial Compression
hr/hm=',num2str(hr/hm)])
                xlabel('Ratio of Material Constants')
                ylabel('Critical Shortening Factor')
                %axis([0 200 0 1])
        end
    else
        c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
        c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
        scritperf=zeros(1,length(c));
        scritslide=zeros(1,length(c));
        for i=1:length(c)
            cr=c(i);
            cm=1;

[s1perf,s2perf,s3perf,s4perf]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,
cr,cm,hr,hm);

[s1slide,s2slide,s3slide,s4slide]=UniaxialSlidingLayers(ar1,ar2,ar3,ar
4,cr,cm,hr,hm);
            scritperf(i)=max([s1perf s2perf s3perf s4perf]);
            scritslide(i)=max([s1slide s2slide s3slide s4slide]);
        end
        figure
        plot(c,scritperf,'r',c,scritslide,'k')
        legend('Perfectly Bonded','Sliding
Layers','Location','SouthEast')
        title(['Uniaxial Compression   hr/hm=',num2str(hr/hm)])
        xlabel('Ratio of Material Constants')
        ylabel('Critical Shortening Factor')
        %axis([0 200 0 1])
```

```
    end


elseif loading==1 && var==3 && bond ==1  %Uniaxial, Thickness Ratio,
Perfectly
    if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater or equal to 0 in
material constant and normalised wavelength input boxes.')
        return
    end
    h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
    h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
    scrit=zeros(1,length(h));
    for i=1:length(h)
        hr=h(i);
        hm=1;
        [s1,s2]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
        scrit(i)=max([s1 s2]);
    end
    figure
    plot(h,scrit)
    title(['Uniaxial Compression Perfectly Bonded
cr/cm=',num2str(cr/cm)])
    xlabel('Ratio of Thickness')
    ylabel('Critical Shortening Factor')


elseif loading==1 && var==3 && bond ==2  %Uniaxial, Thickness Ratio,
Sliding
    if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
        return
    end
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
normalised wavelngth is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'
                h1=inputdlg({'Min. thickness ratio','Max. thickness
ratio ','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});

h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
                scrit=zeros(1,length(h));
                for i=1:length(h)
                    hr=h(i);
                    hm=1;

[s1,s2]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
                    scrit(i)=max([s1 s2]);
                end
                figure
                plot(h,scrit)
                title(['Uniaxial Compression Sliding Layers
cr/cm=',num2str(cr/cm)])
                xlabel('Ratio of Thickness')
```

```
                ylabel('Critical Shortening Factor')
        end
    else
        h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
        h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
        scrit=zeros(1,length(h));
        for i=1:length(h)
            hr=h(i);
            hm=1;

[s1,s2]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
            scrit(i)=max([s1 s2]);
        end
        figure
        plot(h,scrit)
        title(['Uniaxial Compression Sliding Layers
cr/cm=',num2str(cr/cm)])
        xlabel('Ratio of Thickness')
        ylabel('Critical Shortening Factor')
    end


elseif loading==1 && var==3 && bond ==3   %Uniaxial, Thickness Ratio,
Both
     if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
        return
    end
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
normalised wavelngth is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'
                h1=inputdlg({'Min. thickness ratio','Max. thickness
ratio ','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});

h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
                scritperf=zeros(1,length(h));
                scritslide=zeros(1,length(h));
                for i=1:length(h)
                    hr=h(i);
                    hm=1;

[s1perf,s2perf,s3perf,s4perf]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,
cr,cm,hr,hm);

[s1slide,s2slide,s3slide,s4slide]=UniaxialSlidingLayers(ar1,ar2,ar3,ar
4,cr,cm,hr,hm);
                    scritperf(i)=max([s1perf s2perf s3perf s4perf]);
                    scritslide(i)=max([s1slide s2slide s3slide
s4slide]);
                end
                figure
                plot(h,scritperf,'r',h,scritslide,'k')
                legend('Perfectly Bonded','Sliding
Layers','Location','SouthEast')
```

```
                title(['Equi-Biaxial Compression
cr/cm=',num2str(cr/cm)])
                xlabel('Ratio of Thickness')
                ylabel('Critical Shortening Factor')
        end
    else
        h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
        h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
        scritperf=zeros(1,length(h));
        scritslide=zeros(1,length(h));
        for i=1:length(h)
            hr=h(i);
            hm=1;

[s1perf,s2perf,s3perf,s4perf]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,
cr,cm,hr,hm);

[s1slide,s2slide,s3slide,s4slide]=UniaxialSlidingLayers(ar1,ar2,ar3,ar
4,cr,cm,hr,hm);
            scritperf(i)=max([s1perf s2perf s3perf s4perf]);
            scritslide(i)=max([s1slide s2slide s3slide s4slide]);
        end
        figure
        plot(h,scritperf,'r',h,scritslide,'k')
        legend('Perfectly Bonded','Sliding
Layers','Location','SouthEast')
        title(['Equi-Biaxial Compression   cr/cm=',num2str(cr/cm)])
        xlabel('Ratio of Thickness')
        ylabel('Critical Shortening Factor')
    end


elseif loading==2 && var==1 && bond ==1 %Biaxial, Normalized
Wavelength, Perfectly
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
    else
        [s1,s2,s3,s4]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

        figure
        plot(ar1,s1,'r*',ar2,s2,'b',ar3,s3,'ks',ar4,s4,'g');
        legend('Mode 1','Mode 2','Mode 3','Mode
4','Location','SouthEast');
        %axis([0 0.8 0.2 1.05])
        title(['Equi-Biaxial Compression Perfectly Bonded
Cr/Cm=',num2str(cr/cm),'   hr/hm=',num2str(hr/hm)])
        xlabel('Normalized Wavelength')
        ylabel('Shortening Factor')

        k=k+1;
        row = get(handles.uitable1,'Data');
        sall=[s1 s2 s3 s4];
        arall=[ar1 ar2 ar3 ar4];
        [CritShort,locationofCritShort]=max(sall);
        Critar=arall(locationofCritShort);
        row{k,1}='Equi-Biaxial Compression';
        row{k,2}='Perfect';
```

```
        row{k,3}=cr/cm;
        row{k,4}=hr/hm;
        row{k,5}=CritShort;
        row{k,6}=Critar;
        set(handles.uitable1,'Data',row)
    end


elseif loading==2 && var==1 && bond==2  %Biaxial, Normalized,
Wavelength, Sliding
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater or equal to 0 in all
input boxes.')
    else
        if minar<0.1
            questminar = questdlg('A value of greater or equal to 0.1
for normalised wavelngth is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
            switch questminar
                case 'OK'
                case 'Continue Anyway'

[s1,s2,s3,s4]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

                    figure

plot(ar1,s1,'r*',ar2,s2,'b',ar3,s3,'ks',ar4,s4,'g');
                    title(['Equi-Biaxial Compression Sliding Layers
Cr/Cm=',num2str(cr/cm),'   hr/hm=',num2str(hr/hm)])
                    legend('Mode 1','Mode 2','Mode 3','Mode
4','Location','SouthEast');
                    xlabel('Normalized Wavelength')
                    ylabel('Shortening Factor')
                    %axis([0.1 0.8 0.3 1])

                    k=k+1;
                    row = get(handles.uitable1,'Data');
                    sall=[s1 s2 s3 s4];
                    arall=[ar1 ar2 ar3 ar4];
                    [CritShort,locationofCritShort]=max(sall);
                    Critar=arall(locationofCritShort);
                    row{k,1}='Uniaxial Compression';
                    row{k,2}='Sliding Without Priction';
                    row{k,3}=cr/cm;
                    row{k,4}=hr/hm;
                    row{k,5}=CritShort;
                    row{k,6}=Critar;
                    set(handles.uitable1,'Data',row)
            end
        else
            [s1,s2,s3,s4]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

            figure
            plot(ar1,s1,'r*',ar2,s2,'b',ar3,s3,'ks',ar4,s4,'g');
            title(['Equi-Biaxial Compression Sliding Layers
Cr/Cm=',num2str(cr/cm),'   hr/hm=',num2str(hr/hm)])
            legend('Mode 1','Mode 2','Mode 3','Mode
4','Location','SouthEast');
```

```
            xlabel('Normalized Wavelength')
            ylabel('Shortening Factor')
            %axis([0.1 0.8 0.3 1])

            k=k+1;
            row = get(handles.uitable1,'Data');
            sall=[s1 s2 s3 s4];
            arall=[ar1 ar2 ar3 ar4];
            [CritShort,locationofCritShort]=max(sall);
            Critar=arall(locationofCritShort);
            row{k,1}='Equi-Biaxial Compression ';
            row{k,2}='Sliding Without Friction';
            row{k,3}=cr/cm;
            row{k,4}=hr/hm;
            row{k,5}=CritShort;
            row{k,6}=Critar;
            set(handles.uitable1,'Data',row)
        end
    end


elseif loading==2 && var==1 && bond==3  %Biaxial, Normalized
Wavelength, Both
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater or equal to 0 in all
input boxes.')
    else
        if minar<0.1
            questminar = questdlg('A value of greater or equal to 0.1
for normalised wavelngth is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
            switch questminar
                case 'OK'
                case 'Continue Anyway'

[sperf1,sperf2]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[sslide1,sslide2]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

                    figure

plot(ar1,sperf1,'r',ar1,sslide1,'k',ar2,sperf2,'o',ar2,sslide2,'*')
                    legend('Mode 1 Perfectly Bonded','Mode 1 Sliding
Layers','Mode 2 Perfectly Bonded','Mode 2 Sliding Layers
','Location','SouthWest')
                    title(['Equi-Biaxial Compression
Cr/Cm=',num2str(cr/cm),'   hr/hm=',num2str(hr/hm)])
                    xlabel('Normalized Wavelength')
                    ylabel('Shortening Factor')
                    %axis([0.17 0.8 0.8 1])
            end
        else
         [sperf1,sperf2]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[sslide1,sslide2]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

            figure
```

```
plot(ar1,sperf1,'r',ar1,sslide1,'k',ar2,sperf2,'b',ar2,sslide2,'g')
            legend('Mode 1 Perfectly Bonded','Mode 1 Sliding
Layers','Mode 2 Perfectly Bonded','Mode 2 Sliding Layers
','Location','SouthWest')
            title(['Equi-Biaxial Compression Cr/Cm=',num2str(cr/cm),'
hr/hm=',num2str(hr/hm)])
            xlabel('Normalized Wavelength')
            ylabel('Shortening Factor')
            %axis([0.17 0.8 0.8 1])
        end
    end


elseif loading==2 && var==2 && bond ==1  %Biaxial, Material Constant
Ratio, Perfectly
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater or equal to 0 in half
thickness and normalised wavelength input boxes.')
    else
        c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
        c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
        scrit=zeros(1,length(c));
        for i=1:length(c)
            cr=c(i);
            cm=1;

[s1,s2,s3,s4]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
            scrit(i)=max([s1 s2 s3 s4]);
        end
        figure
        plot(c,scrit)
        title(['Equi-Biaxial Compression Perfectly Bonded
hr/hm=',num2str(hr/hm)'])
        xlabel('Ratio of Material Constants')
        ylabel('Critical Shortening Factor')
        %axis([0 200 0 1])
    end


elseif loading==2 && var==2 && bond ==2  %Biaxial, Material Constant
Ratio, Sliding
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater or equal to 0 in half
thickness and normalised wavelength input boxes.')
    else
        if minar<0.1
            questminar = questdlg('A value of greater or equal to 0.1
for Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
            switch questminar
                case 'OK'
                case 'Continue Anyway'
                    c1=inputdlg({'Min. material constant','Max.
material constant','Material constant
increment'},'',1,{'1','100','5'});

c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
```

```
                    scrit=zeros(1,length(c));
                    for i=1:length(c)
                        cr=c(i);
                        cm=1;

[s1,s2,s3,s4]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
                        scrit(i)=max([s1 s2 s3 s4]);
                    end
                    figure
                    plot(c,scrit)
                    title(['Equi-Biaxial Compression Sliding Layers
hr/hm=',num2str(hr/hm)])
                    xlabel('Ratio of Material Constants')
                    ylabel('Critical Shortening Factor')
                    %axis([0 200 0.75 1])
            end
        else
            c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
            c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
            scrit=zeros(1,length(c));
            for i=1:length(c)
                cr=c(i);
                cm=1;

[s1,s2,s3,s4]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
                scrit(i)=max([s1 s2 s3 s4]);
            end
            figure
            plot(c,scrit)
            title(['Equi-Biaxial Compression Sliding Layers
hr/hm=',num2str(hr/hm)])
            xlabel('Ratio of Material Constants')
            ylabel('Critical Shortening Factor')
            %axis([0 200 0.75 1])
        end
    end


elseif loading==2 && var==2 && bond==3  %Biaxial, Material Constant
Ratio, Both
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater or equal to 0 in half
thickness and normalised wavelength input boxes.')
    else
        if minar<0.1
            questminar = questdlg('A value of greater or equal to 0.1
for Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
            switch questminar
                case 'OK'
                case 'Continue Anyway'
                    c1=inputdlg({'Min. material constant','Max.
material constant','Material constant
increment'},'',1,{'1','100','5'});

c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
                    scritperf=zeros(1,length(c));
                    scritslide=zeros(1,length(c));
                    for i=1:length(c)
```

```
                        cr=c(i);
                        cm=1;

[s1perf,s2perf,s3perf,s4perf]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr
,hm);

[s1slide,s2slide,s3slide,s4slide]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,
hr,hm);
                        scritperf(i)=max([s1perf s2perf s3perf
s4perf]);
                        scritslide(i)=max([s1slide s2slide s3slide
s4slide]);
                    end
                    figure
                    plot(c,scritperf,'r',c,scritslide,'k')
                    legend('Perfectly Bonded','Sliding
Layers','Location','SouthEast')
                    title(['Equi-Biaxial Compression
hr/hm=',num2str(hr/hm)])
                    xlabel('Ratio of Material Constants')
                    ylabel('Critical Shortening Factor')
                    %axis([0 200 0 1])
            end
        else
            c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
            c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
            scritperf=zeros(1,length(c));
            scritslide=zeros(1,length(c));
            for i=1:length(c)
                cr=c(i);
                cm=1;

[s1perf,s2perf,s3perf,s4perf]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr
,hm);

[s1slide,s2slide,s3slide,s4slide]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,
hr,hm);
                scritperf(i)=max([s1perf s2perf s3perf s4perf]);
                scritslide(i)=max([s1slide s2slide s3slide s4slide]);
            end
            figure
            plot(c,scritperf,'r',c,scritslide,'k')
            legend('Perfectly Bonded','Sliding
Layers','Location','SouthEast')
            title(['Equi-Biaxial Compression
hr/hm=',num2str(hr/hm)])
            xlabel('Ratio of Material Constants')
            ylabel('Critical Shortening Factor')
            %axis([0 200 0 1])
        end
    end


elseif loading==2 && var==3 && bond ==1  %Biaxial, Thickness Ratio,
Perfectly
    if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater or equal to 0 in
material constant and normalised wavelength input boxes.')
    else
```

```
        h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
        h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
        scrit=zeros(1,length(h));
        for i=1:length(h)
            hr=h(i);
            hm=1;

[s1,s2,s3,s4]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
            scrit(i)=max([s1 s2 s3 s4]);
        end
        figure
        plot(h,scrit)
        title(['Equi-Biaxial Compression Perfectly Bonded
cr/cm=',num2str(cr/cm)])
        xlabel('Ratio of Thickness')
        ylabel('Critical Shortening Factor')
    end


elseif loading==2 && var==3 && bond ==2  %Biaxial, Thickness Ratio,
Sliding
    if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater or equal to 0 in
material constant and normalised wavelength input boxes.')
    else
        if minar<0.1
            questminar = questdlg('A value of greater or equal to 0.1
for Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
            switch questminar
                case 'OK'
                case 'Continue Anyway'
                    h1=inputdlg({'Min. thickness ratio','Max.
thickness ratio ','Thickness ratio
increment'},'',1,{'0.01','0.4','0.01'});

h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
                    scrit=zeros(1,length(h));
                    for i=1:length(h)
                        hr=h(i);
                        hm=1;

[s1,s2,s3,s4]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
                        scrit(i)=max([s1 s2 s3 s4]);
                    end
                    figure
                    plot(h,scrit)
                    title(['Equi-Biaxial Compression Sliding Layers
cr/cm=',num2str(cr/cm)])
                    xlabel('Ratio of Thickness')
                    ylabel('Critical Shortening Factor')
            end
        else
            h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
            h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
            scrit=zeros(1,length(h));
            for i=1:length(h)
                hr=h(i);
```

```
                hm=1;

[s1,s2,s3,s4]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
                scrit(i)=max([s1 s2 s3 s4]);
            end
            figure
            plot(h,scrit)
            title(['Equi-Biaxial Compression Sliding Layers
cr/cm=',num2str(cr/cm)])
            xlabel('Ratio of Thickness')
            ylabel('Critical Shortening Factor')
        end
    end


elseif loading==2 && var==3 && bond==3 %Biaxial, Thickness Ratio, Both
    if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values in material constant and
normalised wavelength input boxes.')
    else
        if minar<0.1
           questminar = questdlg('A value of greater or equal to 0.1
for Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
            switch questminar
                case 'OK'
                case 'Continue Anyway'
                    h1=inputdlg({'Min. thickness ratio','Max.
thickness ratio ','Thickness ratio
increment'},'',1,{'0.01','0.4','0.01'});

h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
                    scritperf=zeros(1,length(h));
                    scritslide=zeros(1,length(h));
                    for i=1:length(h)
                        hr=h(i);
                        hm=1;

[s1perf,s2perf,s3perf,s4perf]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr
,hm);

[s1slide,s2slide,s3slide,s4slide]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,
hr,hm);
                        scritperf(i)=max([s1perf s2perf s3perf
s4perf]);
                        scritslide(i)=max([s1slide s2slide s3slide
s4slide]);
                    end
                    figure
                    plot(h,scritperf,'r',h,scritslide,'k')
                    legend('Perfectly Bonded','Sliding
Layers','Location','SouthEast')
                    title(['Equi-Biaxial Compression
cr/cm=',num2str(cr/cm)])
                    xlabel('Ratio of Thickness')
                    ylabel('Critical Shortening Factor')
            end
        else
            h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
```

```
            h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
            scritperf=zeros(1,length(h));
            scritslide=zeros(1,length(h));
            for i=1:length(h)
                hr=h(i);
                hm=1;

[s1perf,s2perf,s3perf,s4perf]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr
,hm);

[s1slide,s2slide,s3slide,s4slide]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,
hr,hm);
                scritperf(i)=max([s1perf s2perf s3perf s4perf]);
                scritslide(i)=max([s1slide s2slide s3slide s4slide]);
            end
            figure
            plot(h,scritperf,'r',h,scritslide,'k')
            legend('Perfectly Bonded','Sliding
Layers','Location','SouthEast')
            title(['Equi-Biaxial Compression
cr/cm=',num2str(cr/cm)])
            xlabel('Ratio of Thickness')
            ylabel('Critical Shortening Factor')
         end
     end




elseif loading==3 && var==1 && bond==1 %Uni and Bi, Normalized
Wavelength, Perfectly
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
        return
    end
    [s1Bi,s2Bi]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1Uni,s2Uni]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

    figure
    plot(ar1,s1Bi,'b',ar2,s2Bi,'g*',ar1,s1Uni,'r',ar2,s2Uni,'ko')
    legend('Equi-Biaxial Mode 1','Equi-Biaxial Mode 2','Uniaxial Mode
1','Uniaxial Mode 2','Location','SouthEast')
    title(['Equi-Biaxial and Uniaxial Compression Perfectly Bonded
cr/cm=',num2str(cr/cm),'  hr/hm=',num2str(hr/hm)])
    ylabel('Shortening Factor')


elseif loading==3 && var==1 && bond==2   %Uni and Bi, Normalised
Wavelength, Sliding
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
        return
    end
```

```
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'
            [s1Bi,s2Bi]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1Uni,s2Uni]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

            figure

plot(ar1,s1Bi,'b',ar2,s2Bi,'g*',ar1,s1Uni,'r',ar2,s2Uni,'ko')
            legend('Equi-Biaxial Mode 1','Equi-Biaxial Mode
2','Uniaxial Mode 1','Uniaxial Mode 2','Location','SouthEast')
            title(['Equi-Biaxial and Uniaxial Compression Sliding
Layers   cr/cm=',num2str(cr/cm),'  hr/hm=',num2str(hr/hm)])
            ylabel('Shortening Factor')
        end
    else
        [s1Bi,s2Bi]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1Uni,s2Uni]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

        figure
        plot(ar1,s1Bi,'b',ar2,s2Bi,'g*',ar1,s1Uni,'r',ar2,s2Uni,'ko')
        legend('Equi-Biaxial Mode 1','Equi-Biaxial Mode 2','Uniaxial
Mode 1','Uniaxial Mode 2','Location','SouthEast')
        title(['Equi-Biaxial and Uniaxial Compression Sliding Layers
cr/cm=',num2str(cr/cm),'  hr/hm=',num2str(hr/hm)])
        ylabel('Shortening Factor')
        xlabel('Normalised Wavelngth')
    end


elseif loading==3 && var==1 && bond==3   %Uni and Bi, Normalised
Wavelength, Perfect & Sliding
    if emptyCR==1 || emptyCM==1 || emptyHR==1 || emptyHM==1 ||
emptyminar==1 || emptymaxar==1 || cr<=0 || cm<=0 || hr<=0 || hm<=0 ||
minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values of greater than 0 in all input
boxes.')
        return
    end
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'

[s1BiSlide,s2BiSlide]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1UniSlide,s2UniSlide]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr
,hm);

[s1BiPerf,s2BiPerf]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
```

```
[s1UniPerf,s2UniPerf]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr
,hm);

                figure

plot(ar1,s1BiSlide,'b',ar2,s2BiSlide,'g*',ar1,s1UniSlide,'r',ar2,s2Uni
Slide,'ko',ar1,s1BiPerf,'m',ar2,s2BiPerf,'c.',ar1,s1UniPerf,'y',ar2,s2
UniPerf,'kd')
                legend('Equi-Biaxial Sliding Mode 1','Equi-Biaxial
Sliding Mode 2','Uniaxial Sliding Mode 1','Uniaxial Sliding Mode
2','Equi-Biaxial Perfectly Bonded Mode 1','Equi-Biaxial Perfectly
Bonded Mode 2','Uniaxial Mode Perfectly Bonded 1','Uniaxial Perfectly
Bonded Mode 2','Location','SouthEast')
                title(['Equi-Biaxial and Uniaxial Compression
cr/cm=',num2str(cr/cm),'  hr/hm=',num2str(hr/hm)])
                ylabel('Shortening Factor')
                xlabel('Normalised Wavelength')
        end
    else

[s1BiSlide,s2BiSlide]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1UniSlide,s2UniSlide]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr
,hm);

[s1BiPerf,s2BiPerf]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1UniPerf,s2UniPerf]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr
,hm);

        figure

plot(ar1,s1BiSlide,'b',ar2,s2BiSlide,'g*',ar1,s1UniSlide,'r',ar2,s2Uni
Slide,'ko',ar1,s1BiPerf,'m',ar2,s2BiPerf,'c.',ar1,s1UniPerf,'y',ar2,s2
UniPerf,'kd')
        legend('Equi-Biaxial Sliding Mode 1','Equi-Biaxial Sliding
Mode 2','Uniaxial Sliding Mode 1','Uniaxial Sliding Mode 2','Equi-
Biaxial Perfectly Bonded Mode 1','Equi-Biaxial Perfectly Bonded Mode
2','Uniaxial Mode Perfectly Bonded 1','Uniaxial Perfectly Bonded Mode
2','Location','SouthEast')
        title(['Equi-Biaxial and Uniaxial Compression
cr/cm=',num2str(cr/cm),'  hr/hm=',num2str(hr/hm)])
        ylabel('Shortening Factor')
        xlabel('Normalised Wavelength')
    end


elseif loading==3 && var==2 && bond==1   %Uni and Bi, Material
Constant Ratio, Perfect
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values for thickness ratio and normalised
wavelength input boxes.')
        return
    end
    c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
    c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
    scritUni=zeros(1,length(c));
    scritBi=zeros(1,length(c));
```

```
    for i=1:length(c)
        cr=c(i);
        cm=1;

[s1Bi,s2Bi,s3Bi,s4Bi]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1Uni,s2Uni,s3Uni,s4Uni]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,c
m,hr,hm);
        scritBi(i)=max([s1Bi s2Bi s3Bi s4Bi]);
        scritUni(i)=max([s1Uni s2Uni s3Uni s4Uni]);
    end
    figure
    plot(c,scritBi,'b',c,scritUni,'g')
    legend('Equi-Biaxial Perfect','Uniaxial
Perfect','Location','SouthEast')
    title(['Equi-Biaxial and Uniaxial Compression
hr/hm=',num2str(hr/hm)])
    ylabel('Critical Shortening Factor')
    xlabel('Material Constant Ratio cr/cm')


elseif loading==3 && var==2 && bond==2   %Uni and Bi, Material
Constant Ratio, Sliding
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values for thickness ratio and normalised
wavelength input boxes.')
        return
    end
     if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'
                c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});

c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
                scritUni=zeros(1,length(c));
                scritBi=zeros(1,length(c));
                for i=1:length(c)
                    cr=c(i);
                    cm=1;

[s1Bi,s2Bi,s3Bi,s4Bi]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1Uni,s2Uni,s3Uni,s4Uni]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,
hr,hm);
                    scritBi(i)=max([s1Bi s2Bi s3Bi s4Bi]);
                    scritUni(i)=max([s1Uni s2Uni s3Uni s4Uni]);
                end
                figure
                plot(c,scritBi,'b',c,scritUni,'g')
                legend('Equi-Biaxial Sliding','Uniaxial
Sliding','Location','SouthEast')
                title(['Equi-Biaxial and Uniaxial Compression
hr/hm=',num2str(hr/hm)])
                ylabel('Critical Shortening Factor')
                xlabel('Material Constant Ratio cr/cm')
```

```
        end
    else
        c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
        c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
        scritUni=zeros(1,length(c));
        scritBi=zeros(1,length(c));
        for i=1:length(c)
            cr=c(i);
            cm=1;

[s1Bi,s2Bi,s3Bi,s4Bi]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1Uni,s2Uni,s3Uni,s4Uni]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,
hr,hm);
            scritBi(i)=max([s1Bi s2Bi s3Bi s4Bi]);
            scritUni(i)=max([s1Uni s2Uni s3Uni s4Uni]);
        end
        figure
        plot(c,scritBi,'b',c,scritUni,'g')
        legend('Equi-Biaxial Sliding','Uniaxial
Sliding','Location','SouthEast')
        title(['Equi-Biaxial and Uniaxial Compression
hr/hm=',num2str(hr/hm)])
        ylabel('Critical Shortening Factor')
        xlabel('Material Constant Ratio cr/cm')
    end


elseif loading==3 && var==2 && bond==3   %Uni and Bi, Material
Constant Ratio, Perfect and Sliding
    if emptyHR==1 || emptyHM==1 || emptyminar==1 || emptymaxar==1 ||
hr<=0 || hm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values for thickness ratio and normalised
wavelength input boxes.')
        return
    end
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'
                c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});

c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
                scritUniPerf=zeros(1,length(c));
                scritBiPerf=zeros(1,length(c));
                scritBiSlide=zeros(1,length(c));
                scritUniSlide=zeros(1,length(c));
                for i=1:length(c)
                    cr=c(i);
                    cm=1;

[s1BiPerf,s2BiPerf,s3BiPerf,s4BiPerf]=PerfectlyBonded(ar1,ar2,ar3,ar4,
cr,cm,hr,hm);

[s1UniPerf,s2UniPerf,s3UniPerf,s4UniPerf]=UniaxialPerfectlyBonded(ar1,
ar2,ar3,ar4,cr,cm,hr,hm);
```

```
[s1BiSlide,s2BiSlide,s3BiSlide,s4BiSlide]=SlidingLayers(ar1,ar2,ar3,ar
4,cr,cm,hr,hm);

[s1UniSlide,s2UniSlide,s3UniSlide,s4UniSlide]=UniaxialSlidingLayers(ar
1,ar2,ar3,ar4,cr,cm,hr,hm);

                    scritBiPerf(i)=max([s1BiPerf s2BiPerf s3BiPerf
s4BiPerf]);
                    scritUniPerf(i)=max([s1UniPerf s2UniPerf s3UniPerf
s4UniPerf]);
                    scritBiSlide(i)=max([s1BiSlide s2BiSlide s3BiSlide
s4BiSlide]);
                    scritUniSlide(i)=max([s1UniSlide s2UniSlide
s3UniSlide s4UniSlide]);
                end
                figure

plot(c,scritBiPerf,'b',c,scritUniPerf,'g.',c,scritBiSlide,'k',c,scritU
niSlide,'m*')
                legend('Equi-Biaxial Perfect','Uniaxial
Perfect','Equi-Biaxial Sliding','Uniaxial
Sliding','Location','SouthEast')
                title(['Equi-Biaxial and Uniaxial Compression
hr/hm=',num2str(hr/hm)])
                ylabel('Critical Shortening Factor')
                xlabel('Material Constant Ratio cr/cm')
        end
    else
        c1=inputdlg({'Min. material constant','Max. material
constant','Material constant increment'},'',1,{'1','100','5'});
        c=str2double(c1{1}):str2double(c1{3}):str2double(c1{2});
        scritUniPerf=zeros(1,length(c));
        scritBiPerf=zeros(1,length(c));
        scritBiSlide=zeros(1,length(c));
        scritUniSlide=zeros(1,length(c));
        for i=1:length(c)
            cr=c(i);
            cm=1;

[s1BiPerf,s2BiPerf,s3BiPerf,s4BiPerf]=PerfectlyBonded(ar1,ar2,ar3,ar4,
cr,cm,hr,hm);

[s1UniPerf,s2UniPerf,s3UniPerf,s4UniPerf]=UniaxialPerfectlyBonded(ar1,
ar2,ar3,ar4,cr,cm,hr,hm);
[s1BiSlide,s2BiSlide,s3BiSlide,s4BiSlide]=SlidingLayers(ar1,ar2,ar3,ar
4,cr,cm,hr,hm);

[s1UniSlide,s2UniSlide,s3UniSlide,s4UniSlide]=UniaxialSlidingLayers(ar
1,ar2,ar3,ar4,cr,cm,hr,hm);

            scritBiPerf(i)=max([s1BiPerf s2BiPerf s3BiPerf
s4BiPerf]);
            scritUniPerf(i)=max([s1UniPerf s2UniPerf s3UniPerf
s4UniPerf]);
            scritBiSlide(i)=max([s1BiSlide s2BiSlide s3BiSlide
s4BiSlide]);
            scritUniSlide(i)=max([s1UniSlide s2UniSlide s3UniSlide
s4UniSlide]);
        end
        figure
```

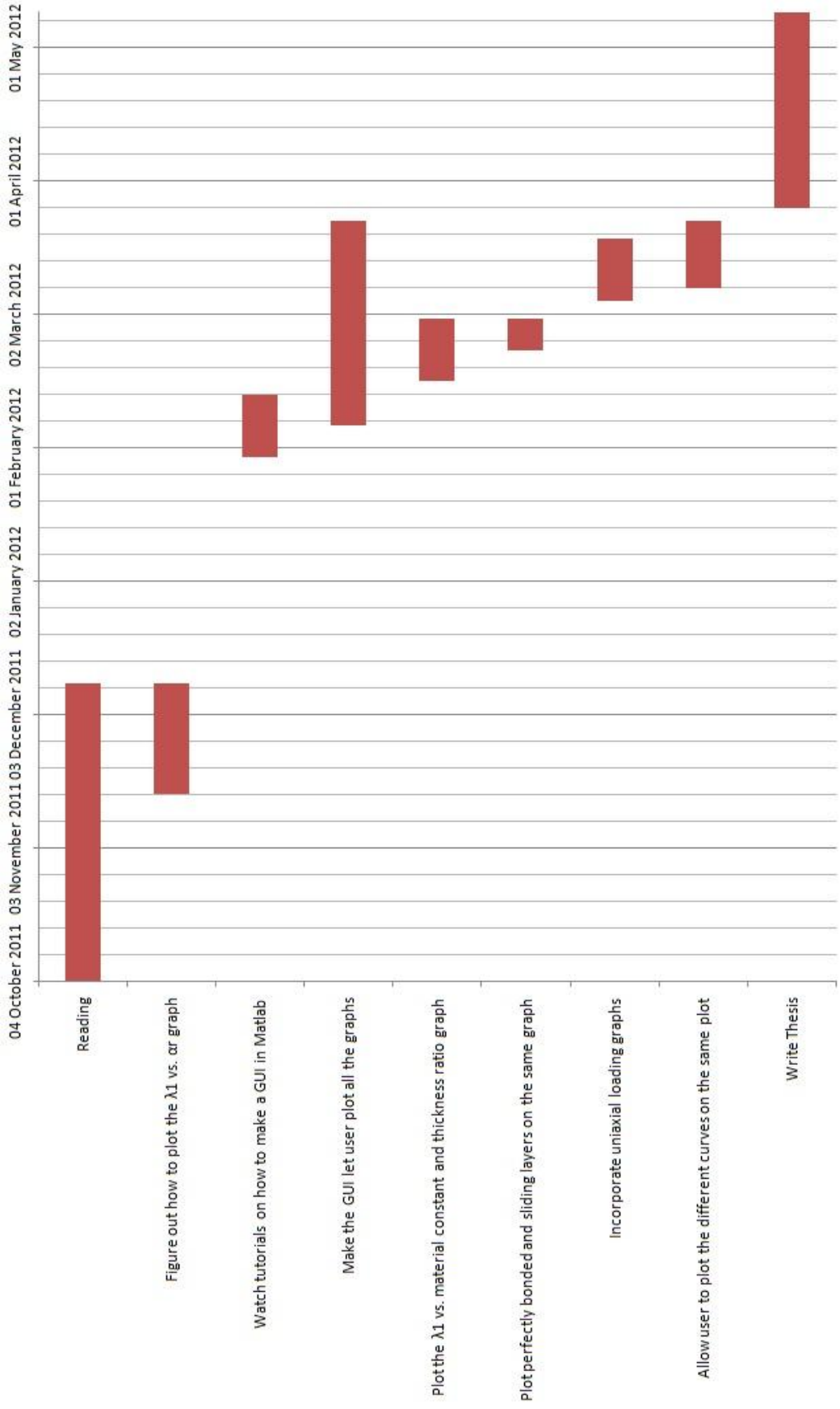
```
plot(c,scritBiPerf,'b',c,scritUniPerf,'g.',c,scritBiSlide,'k',c,scritU
niSlide,'m*')
         legend('Equi-Biaxial Perfect','Uniaxial Perfect','Equi-
Biaxial Sliding','Uniaxial Sliding','Location','SouthEast')
         title(['Equi-Biaxial and Uniaxial Compression
hr/hm=',num2str(hr/hm)])
         ylabel('Critical Shortening Factor')
         xlabel('Material Constant Ratio cr/cm')
     end


elseif loading==3 && var==3 && bond==1   %Uni and Bi, Thickness Ratio,
Perfect
    if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values in material constant and
normalised wavelength input boxes.')
        return
    end
    h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
    h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
    scritUni=zeros(1,length(h));
    scritBi=zeros(1,length(h));
    for i=1:length(h)
        hr=h(i);
        hm=1;

[s1Uni,s2Uni]=UniaxialPerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
        [s1Bi,s2Bi]=PerfectlyBonded(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
        scritUni(i)=max([s1Uni s2Uni]);
        scritBi(i)=max([s1Bi s2Bi]);
    end
    figure
    plot(h,scritBi,'b',h,scritUni,'g')
    legend('Equi-Biaxial Perfect','Uniaxial
Perfect','Location','SouthEast')
    title(['Equi-Biaxial and Uniaxial Compression
cr/cm=',num2str(cr/cm)])
    ylabel('Critical Shortening Factor')
    xlabel('Material Thickness Ratio hr/hm')

elseif loading==3 && var==3 && bond==2   %Uni and Bi, Thickness Ratio,
Sliding
    if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values in material constant and
normalised wavelength input boxes.')
        return
    end
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
        switch questminar
            case 'OK'
            case 'Continue Anyway'
                h1=inputdlg({'Min. thickness ratio','Max. thickness
ratio ','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
```

```
h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
                scritUni=zeros(1,length(h));
                scritBi=zeros(1,length(h));
                for i=1:length(h)
                    hr=h(i);
                    hm=1;

[s1Uni,s2Uni]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);

[s1Bi,s2Bi]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
                    scritUni(i)=max([s1Uni s2Uni]);
                    scritBi(i)=max([s1Bi s2Bi]);
                end
                figure
                plot(h,scritBi,'b',h,scritUni,'g')
                legend('Equi-Biaxial Sliding','Uniaxial
Sliding','Location','SouthEast')
                title(['Equi-Biaxial and Uniaxial Compression
cr/cm=',num2str(cr/cm)])
                ylabel('Critical Shortening Factor')
                xlabel('Material Thickness Ratio hr/hm')
        end
    else
        h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
        h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
        scritUni=zeros(1,length(h));
        scritBi=zeros(1,length(h));
        for i=1:length(h)
            hr=h(i);
            hm=1;

[s1Uni,s2Uni]=UniaxialSlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
            [s1Bi,s2Bi]=SlidingLayers(ar1,ar2,ar3,ar4,cr,cm,hr,hm);
            scritUni(i)=max([s1Uni s2Uni]);
            scritBi(i)=max([s1Bi s2Bi]);
        end
        figure
        plot(h,scritBi,'b',h,scritUni,'g')
        legend('Equi-Biaxial Sliding','Uniaxial
Sliding','Location','SouthEast')
        title(['Equi-Biaxial and Uniaxial Compression
cr/cm=',num2str(cr/cm)])
        ylabel('Critical Shortening Factor')
        xlabel('Material Thickness Ratio hr/hm')
    end


elseif loading==3 && var==3 && bond==3   %Uni and Bi, Thickness Ratio,
Perfect and Sliding
    if emptyCR==1 || emptyCM==1 || emptyminar==1 || emptymaxar==1 ||
cr<=0 || cm<=0 || minar<=0 || maxar<=0 || incar<=0
        msgbox('Please enter values in material constant and
normalised wavelength input boxes.')
        return
    end
    if minar<0.1
        questminar = questdlg('A value of greater or equal to 0.1 for
Minimum Normalised wavelength is needed to give an accurate
graph','Warning','OK','Continue Anyway','OK');
```

```
        switch questminar
            case 'OK'
            case 'Continue Anyway'
                h1=inputdlg({'Min. thickness ratio','Max. thickness
ratio ','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});

h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
                scritUniPerf=zeros(1,length(h));
                scritBiPerf=zeros(1,length(h));
                scritBiSlide=zeros(1,length(h));
                scritUniSlide=zeros(1,length(h));
                for i=1:length(h)
                    hr=h(i);
                    hm=1;

[s1BiPerf,s2BiPerf,s3BiPerf,s4BiPerf]=PerfectlyBonded(ar1,ar2,ar3,ar4,
cr,cm,hr,hm);

[s1UniPerf,s2UniPerf,s3UniPerf,s4UniPerf]=UniaxialPerfectlyBonded(ar1,
ar2,ar3,ar4,cr,cm,hr,hm);

[s1BiSlide,s2BiSlide,s3BiSlide,s4BiSlide]=SlidingLayers(ar1,ar2,ar3,ar
4,cr,cm,hr,hm);

[s1UniSlide,s2UniSlide,s3UniSlide,s4UniSlide]=UniaxialSlidingLayers(ar
1,ar2,ar3,ar4,cr,cm,hr,hm);

                    scritBiPerf(i)=max([s1BiPerf s2BiPerf s3BiPerf
s4BiPerf]);
                    scritUniPerf(i)=max([s1UniPerf s2UniPerf s3UniPerf
s4UniPerf]);
                    scritBiSlide(i)=max([s1BiSlide s2BiSlide s3BiSlide
s4BiSlide]);
                    scritUniSlide(i)=max([s1UniSlide s2UniSlide
s3UniSlide s4UniSlide]);
                end
                figure

plot(h,scritBiPerf,'b',h,scritUniPerf,'g.',h,scritBiSlide,'k',h,scritU
niSlide,'m*')
                legend('Equi-Biaxial Perfect','Uniaxial
Perfect','Equi-Biaxial Sliding','Uniaxial
Sliding','Location','SouthEast')
                title(['Equi-Biaxial and Uniaxial Compression
cr/cm=',num2str(cr/cm)])
                ylabel('Critical Shortening Factor')
                xlabel('Material Thickness Ratio hr/hm')
        end
    else
        h1=inputdlg({'Min. thickness ratio','Max. thickness ratio
','Thickness ratio increment'},'',1,{'0.01','0.4','0.01'});
        h=str2double(h1{1}):str2double(h1{3}):str2double(h1{2});
        scritUniPerf=zeros(1,length(h));
        scritBiPerf=zeros(1,length(h));
        scritBiSlide=zeros(1,length(h));
        scritUniSlide=zeros(1,length(h));
        for i=1:length(h)
            hr=h(i);
            hm=1;
```

```
[s1BiPerf,s2BiPerf,s3BiPerf,s4BiPerf]=PerfectlyBonded(ar1,ar2,ar3,ar4,
cr,cm,hr,hm);

[s1UniPerf,s2UniPerf,s3UniPerf,s4UniPerf]=UniaxialPerfectlyBonded(ar1,
ar2,ar3,ar4,cr,cm,hr,hm);

[s1BiSlide,s2BiSlide,s3BiSlide,s4BiSlide]=SlidingLayers(ar1,ar2,ar3,ar
4,cr,cm,hr,hm);

[s1UniSlide,s2UniSlide,s3UniSlide,s4UniSlide]=UniaxialSlidingLayers(ar
1,ar2,ar3,ar4,cr,cm,hr,hm);

            scritBiPerf(i)=max([s1BiPerf s2BiPerf s3BiPerf s4BiPerf]);
            scritUniPerf(i)=max([s1UniPerf s2UniPerf s3UniPerf
s4UniPerf]);
            scritBiSlide(i)=max([s1BiSlide s2BiSlide s3BiSlide
s4BiSlide]);
            scritUniSlide(i)=max([s1UniSlide s2UniSlide s3UniSlide
s4UniSlide]);
        end
        figure

plot(h,scritBiPerf,'b',h,scritUniPerf,'g.',h,scritBiSlide,'k',h,scritU
niSlide,'m*')
        legend('Equi-Biaxial Perfect','Uniaxial Perfect','Equi-Biaxial
Sliding','Uniaxial Sliding','Location','SouthEast')
        title(['Equi-Biaxial and Uniaxial Compression
cr/cm=',num2str(cr/cm)])
        ylabel('Critical Shortening Factor')
        xlabel('Material Thickness Ratio hr/hm')
    end

end
```

Appendix G: Gantt Chart



References

[1] Guz I.A. & Soutis C., 2005, "Compressive strength of laminated composites: on application of the continuum fracture theory". In: M.Guagliano & M.H. Aliabadi, eds. 2005. Fracture and Damage of Composites. WIT Press. Ch.1.

[2] Guz I.A. & Hermann K.P., 2003, "On the lower bounds for the critical loads under large deformations in non-linear hyperelastic composites with imperfect interlaminar adhesion" , European Journal of Mechanics A/Solids 22, pp 837-849.

[3] Soutis C. & Guz I.A., 2006, "Fracture of layered composites by internal fibre instability: effect of interfacial adhesion" The Aeronautical Journal of the Royal Aeronautical Society, March 2006, pp 185-190.

[4] Guz I.A. & Soutis C., 1999, "Compressive fracture of non-linear composites undergoing large deformations" International Journal of Solids and Structures 38, pp 3759-3770.

[5] Guz I.A., 2004, "The effect of the multi-axiality of compressive loading on the accuracy of a continuum model for layered materials" International Journal of Solids and Structures 42, pp 439-453.

[6] Menshykova M.V., Guz I.A. & Menshykov O.V, 2009, "A unified computational approach to instability of periodic laminated materials" Computer Modelling in Engineering and Sciences, vol.51, no3, pp239-259.

[7]Jones R.M, 1999, "Mechanics of Composite Materials" 2nd edition. London: Taylor and Francis Limited.

[8]Daniel A.M & Ishai O., 1994. "Engineering Mechanics of Composite Material". New York: Oxford University Press.

[9]Airbus, 2011. Composites Manufacturing and Assembly. [online] Available at: <http://www.airbus.com/work/why-join-airbus/training-and-development/composites-manufacturing-assembly/?contentId=[_TABLE%3Att_content%3B_FIELD%3Auid]%2C&cHash=22935adfac92fcbbd4ba4e1441d13383>[Accessed 6 April 2012].

[10] Boeing, 2011. Boeing 787 From the Ground Up. [online] Available at: <http://www.boeing.com/commercial/aeromagazine/articles/qtr_4_06/article_04_2.html > [Accessed 6 April 2012].