Determination Of Extreme Shear Stresses In Teaching Mechanics Using Freely Available Computer Tools

Rado Flajs

Abstract—In the present paper the extreme shear stresses with the corresponding planes are established using the freely available computer tools like the Gnuplot, Sage, R, Python and Octave. In order to support these freely available computer tools, their strong symbolical and graphical abilities are illustrated.

The nature of the stationary points obtained by the Method of Lagrangian Multipliers can be determined using freely available computer symbolical tools like Sage.

The characters of the stationary points can be explained in the easiest way using freely available computer graphical tools like Gnuplot, Sage, R, Python and Octave. The presented figures improve the understanding of the problem and the obtained solutions for the majority of students of civil or mechanical engineering.

Keywords—engineering, continuum mechanics, extreme shear stresses, Gnuplot, Sage, R, Python, Octave

I. INTRODUCTION

In the study of continuum mechanics in engineering and physics a great attention is devoted to the consideration of the normal and shear stresses. Shear stress \( \tau \equiv \sigma_{nt} \) is defined by the projection of the stress vector \( \vec{\sigma}_n \) to the corresponding plane. The physical meaning of the stress vector, normal and shear stresses are seen from Fig. 1. The square of the shear stress can be obtained by Pitagora theorem

\[
\sigma_{nt}^2 = \vec{\sigma}_n \cdot \vec{\sigma}_n - (\vec{\sigma}_n \cdot \vec{\sigma}_n)^2.
\]  

Using the Lagrange identity

\[
(\vec{a} \times \vec{b}) \cdot (\vec{c} \times \vec{d}) = (\vec{a} \cdot \vec{c}) (\vec{b} \cdot \vec{d}) - (\vec{b} \cdot \vec{c}) (\vec{a} \cdot \vec{d})
\]

we get

\[
\sigma_{nt}^2 = \vec{\sigma}_n \cdot \vec{\sigma}_n - (\vec{\sigma}_n \cdot \vec{\sigma}_n)^2 = (\vec{\sigma}_n \cdot \vec{\sigma}_n) (\vec{\sigma}_n \cdot \vec{\sigma}_n) - (\vec{\sigma}_n \cdot \vec{\sigma}_n)^2 = (\vec{\sigma}_n \times \vec{\sigma}_n) \cdot (\vec{\sigma}_n \times \vec{\sigma}_n) = \|\vec{\sigma}_n \times \vec{\sigma}_n\|^2.
\]

Applying the principal coordinate system \((x_1, x_2, x_3)\) and referring to the principal basis \(\vec{e}_1, \vec{e}_2, \vec{e}_3\) the stress vector \(\vec{\sigma}_n\) can be expressed by the linear combination

\[
\vec{\sigma}_n = \sigma_{11} e_{1} \vec{e}_1 + \sigma_{22} e_{2} \vec{e}_2 + \sigma_{33} e_{3} \vec{e}_3,
\]

where \(\sigma_{11}, \sigma_{22}, \sigma_{33}\) are principal stresses and \(e_{1}, e_{2}, e_{3}\) are directional cosines of the unit normal at the arbitrary plane \([1], [2]\). According to that we can write

\[
\vec{\sigma}_n \times \vec{\sigma}_n = \begin{vmatrix}
\vec{e}_1 & \vec{e}_2 & \vec{e}_3 \\
\sigma_{11} e_{1} & \sigma_{22} e_{2} & \sigma_{33} e_{3} \\
e_{1} & e_{2} & e_{3}
\end{vmatrix}
\]

and obtain

\[
\sigma_{nt}^2 = (\sigma_{11} - \sigma_{22})^2 e_{1}^2 e_{2}^2 e_{3}^2 + (\sigma_{33} - \sigma_{11})^2 e_{1}^2 e_{2}^2 e_{3}^2 + (\sigma_{22} - \sigma_{33})^2 e_{2}^2 e_{3}^2 e_{3}^2.
\]  

It is worth to note that Eq. (3) presents the component form of the equation \(|\sigma_{nt}| = \|\vec{\sigma}_n \times \vec{\sigma}_n\|\) in the principal coordinate system.

One of the interesting problems is to find the extreme shear stresses with the corresponding planes. The extreme shear stresses play the important role in Tresca yielding criteria \([1], [2], [3], [4], [5], [6], [7]\).

Using Eq. (3) we have to mathematically determine the directional cosines \(e_{1}, e_{2}, e_{3}\) of the unit normal to the planes where the extreme shear stresses appear. Since the sum of directional cosines squares equals one, we will search the maximum of the square of the shear stresses under the condition \(g(e_{1}, e_{2}, e_{3}) = e_{1}^2 + e_{2}^2 + e_{3}^2 - 1 = 0\). Using the Method of Lagrangian Multipliers \([8]\) we can find the stationary points of the Lagrangian function \(L(e_{1}, e_{2}, e_{3}, \lambda) = \sigma_{nt}^2(e_{1}, e_{2}, e_{3}) + \lambda g(e_{1}, e_{2}, e_{3})\). This method requires the usage of the differential calculus. The classification of the stationary points can be quite cumbersome and is usually
omitted in the undergraduate study of mechanics. The Method of Lagrangian Multipliers can be naturally explained by the usage of the freely available graphics tools like Gnuplot [9], Sage [10], R [11], Python [12] and Octave [13]. The characteristics of the obtained stationary points (saddle points or Extrema), which usually cause a bit of confusion, can be determined immediately from the obtained pictures.

II. CLASSIFICATION OF THE STATIONARY POINTS, OBTAINED BY THE METHOD OF LAGRANGIAN MULTIPLIERS, USING SAGE SYMBOLIC TOOLS

The type of the stationary point (saddle points or Extrema) should be determined from the eigenvalues of Hessian second derivative matrix of Lagrangian function $L$ as follows:

$$H = \begin{bmatrix}
\frac{\partial^2 L}{\partial x^2} & \frac{\partial^2 L}{\partial x \partial y} & \frac{\partial^2 L}{\partial x \partial z} \\
\frac{\partial^2 L}{\partial y \partial x} & \frac{\partial^2 L}{\partial y^2} & \frac{\partial^2 L}{\partial y \partial z} \\
\frac{\partial^2 L}{\partial z \partial x} & \frac{\partial^2 L}{\partial z \partial y} & \frac{\partial^2 L}{\partial z^2}
\end{bmatrix}$$

(4)

at stationary points

$$(e_{n1}, e_{n2}, e_{n3}, \lambda) = \left( \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0, -\sigma_{11} \sigma_{22} \right), \quad (5a)$$

$$(e_{n1}, e_{n2}, e_{n3}, \lambda) = \left( 0, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, -\sigma_{11} \sigma_{33} \right), \quad (5b)$$

$$(e_{n1}, e_{n2}, e_{n3}, \lambda) = \left( 0, 0, \frac{\sqrt{2}}{2}, -\sigma_{22} \sigma_{33} \right). \quad (5c)$$

The calculation can be performed using the following procedure in Sage 4.6.2 as follows:

# function value
f = F.subs(x=s[x], y=s[y], z=s[z], lam=s[lam])
# Hessian matrix
h = Hesse(F,x,y,z).subs(x=s[x], y=s[y], z=s[z], lam=s[lam])
# eigenvectors of Hessian matrix
l = h.eigenvalues()
e = h.eigenvectors_right()
# simplify
if f != 0:
    f = f.factor()
for i in range(len(l)):
    if l[i] != 0:
        l[i] = l[i].factor()
print k, s, f, l
# print k, s, f, l, e

Applying this program one can obtain the following results, where for each obtained solution the values in the first, the second, the third and the fourth rows present index of the obtained solution, the stationary point, the value of Lagrangian function $L$ and the eigenvalues of Hessian matrix at those point, respectively:

1. $$(y: 0, lam: -s_{11}^2, x: -1, z: 0)$$
   
2. $$(y: 0, lam: -s_{11}^2, x: 1, z: 0)$$
   
3. $$(y: 1, lam: -s_{22}^2, x: 0, z: 0)$$
   
4. $$(y: -1, lam: -s_{22}^2, x: 0, z: 0)$$
   
5. $$(y: -1/2*sqrt(2), lam: -s_{11}*s_{22}, x: -1/2*sqrt(2), z: 0)$$
   
6. $$(y: 1/2*sqrt(2), lam: -s_{11}*s_{22}, x: 1/2*sqrt(2), z: 0)$$
   
7. $$(y: -1/2*sqrt(2), lam: -s_{11}*s_{22}, x: -1/2*sqrt(2), z: 0)$$
   
8. $$(y: 1/2*sqrt(2), lam: -s_{11}*s_{22}, x: 1/2*sqrt(2), z: 0)$$
   
9. $$(y: 0, lam: -s_{33}^2, x: 0, z: -1)$$
   
10. $$(y: 0, lam: -s_{33}^2, x: 0, z: 1)$$

# print k, s, f, l, e

# simple test
sig11, sig22, sig33 = var('sig11, sig22, sig33')
x, y, z, lam = var('x, y, z, lam')
F = funF(sig11, sig22, sig33, x, y, z, lam)
# all stationary points
S = solve([D(F,x)==0,D(F,y)==0,D(F,z)==0,D(F,lam)==0],
x, y, z, lam,solution_dict=True)
# stationary points analysis
n = len(S)
for (k,s) in zip(range(1,n+1),S):
Fig. 2. Graphical explanation of the characters of the stationary points, saddle points or Extrema, obtained by the Method of Lagrangian Multipliers.

A. Graphical explanation of the character of the stationary shear stress points obtained by the Method of Lagrangian Multipliers using graphical tools on two dimensional example

A nice explanation of the two dimensional stress state is presented in [14]. Let us consider the case \( \sigma_{11} = \sigma_{22} < \sigma_{33} \). Equation (3) simplifies into

\[
\sigma_{nt}^2 = (\sigma_{11} - \sigma_{22})^2 e_{n1}^2 + (\sigma_{33} - \sigma_{11})^2 e_{n1}^2 e_{n3}^2 + (\sigma_{22} - \sigma_{33})^2 e_{n2}^2 e_{n3}^2 + (\sigma_{11} - \sigma_{33})^2 e_{n1}^2 e_{n2}^2.
\]  

Consequently we get

\( \sigma_{nt} = \pm (\sigma_{11} - \sigma_{33}) e_{n3} \sqrt{e_{n1}^2 + e_{n2}^2}. \)  

Employing the abbreviation \( x = e_{n1}, y = e_{n2}, z = e_{n3}, r = \sqrt{x^2 + y^2} \) we maximize the function \( f(r, z) = (\sigma_{33} - \sigma_{11}) r z \) under the condition \( g(r, z) = r^2 + z^2 - 1 = 0 \). Using the Method of Lagrangian Multipliers, we can find four stationary points \( (r, z) = \left( \pm \frac{\sqrt{2}}{2}, \pm \sqrt{2} \right) \) (in the Fig. 3 denoted by grey bullets) where the gradients \( \nabla f \) and \( \nabla g \) coincide. The gradients are perpendicular to the curves \( g(x, y) = 0 \) and \( f = \tau_{\text{max}} \). All stationary points are Extrema, since the unit circle touches the curve \( f(x, y) = 1 \).

B. Graphical explanation of the character of the stationary shear stress points obtained by the Method of Lagrangian Multipliers using graphical tools on two dimensional example

A nice graphical explanation of the two dimensional stress state is presented in [14]. Let us consider the case \( \sigma_{11} = \sigma_{22} < \sigma_{33} \). Equation (3) simplifies into

\[
\sigma_{nt}^2 = (\sigma_{11} - \sigma_{22})^2 e_{n1}^2 + (\sigma_{33} - \sigma_{11})^2 e_{n1}^2 e_{n3}^2 + (\sigma_{22} - \sigma_{33})^2 e_{n2}^2 e_{n3}^2 + (\sigma_{11} - \sigma_{33})^2 e_{n1}^2 e_{n2}^2.
\]  

Consequently we get

\( \sigma_{nt} = \pm (\sigma_{11} - \sigma_{33}) e_{n3} \sqrt{e_{n1}^2 + e_{n2}^2}. \)  

Employing the abbreviation \( x = e_{n1}, y = e_{n2}, z = e_{n3}, r = \sqrt{x^2 + y^2} \) we maximize the function \( f(r, z) = (\sigma_{33} - \sigma_{11}) r z \) under the condition \( g(r, z) = r^2 + z^2 - 1 = 0 \). Using the Method of Lagrangian Multipliers, we can find four stationary points \( (r, z) = \left( \pm \frac{\sqrt{2}}{2}, \pm \sqrt{2} \right) \) (in the Fig. 3 denoted by grey bullets) where the gradients \( \nabla f \) and \( \nabla g \) coincide. The gradients are perpendicular to the curves \( g(x, y) = 0 \) and \( f = \tau_{\text{max}} \). All stationary points are Extrema, since the unit circle (graph of the function \( g = 0 \)) only touches (not intersects) the graph of the function \( f = \tau_{\text{max}} \) at these points.
Extrema

Saddle points

C. Graphical explanation of the character of the stationary shear stress points obtained by the Method of Lagrangian Multipliers using graphical tools on a three-dimensional example

In three-dimensional case, where \( \sigma_{11} < \sigma_{22} < \sigma_{33} \), the situation is a bit more complicated, so we will use the computer graphical tools in order to classify the character of the stationary points directly from the obtained picture. Employing the principal coordinate system \((x, y, z)\) and using the Method of Lagrangian Multipliers one can obtain 12 stationary points, namely \((\pm \frac{\sqrt{2}}{2}, \pm \frac{\sqrt{2}}{2}, 0)\), \((\pm \frac{\sqrt{2}}{2}, 0, \pm \frac{\sqrt{2}}{2})\), \((0, \pm \frac{\sqrt{2}}{2}, \pm \frac{\sqrt{2}}{2})\). We will construct the unit sphere, i.e. the graph of the function \(g(x, y, z) = x^2 + y^2 + z^2 - 1 = 0\), and three surfaces corresponding the equations:

\[
\tau(x, y, z) \equiv \sigma_{nt}(x, y, z) = \frac{\sigma_{33} - \sigma_{22}}{2}, \quad (11a)
\]
\[
\tau(x, y, z) \equiv \sigma_{nt}(x, y, z) = \frac{\sigma_{33} - \sigma_{11}}{2}, \quad (11b)
\]
\[
\tau(x, y, z) \equiv \sigma_{nt}(x, y, z) = \frac{\sigma_{22} - \sigma_{11}}{2}. \quad (11c)
\]

Since the unit sphere intersects the graph of the functions \((11a)\) and \((11c)\), the corresponding stationary points \((\pm \frac{\sqrt{2}}{2}, \pm \frac{\sqrt{2}}{2}, 0)\) and \((0, \pm \frac{\sqrt{2}}{2}, \pm \frac{\sqrt{2}}{2})\) are saddle points. In the second case the unit sphere touches the graph of the function \((11b)\), so the corresponding stationary points \((\pm \frac{\sqrt{2}}{2}, 0, \pm \frac{\sqrt{2}}{2})\), are Extrema.

All these facts can be seen easily from Figs. 4, 5, 6 and 7 and can be obtained from graphical programming tools like Sage 10, Octave 3.2.4 and Python 3.6.5 and obtained from the obtained picture. Employing the principal coordinate system \((x, y, z)\) and using the Method of Lagrangian Multipliers one can obtain 12 stationary points, namely \((\pm \frac{\sqrt{2}}{2}, \pm \frac{\sqrt{2}}{2}, 0)\), \((\pm \frac{\sqrt{2}}{2}, 0, \pm \frac{\sqrt{2}}{2})\), \((0, \pm \frac{\sqrt{2}}{2}, \pm \frac{\sqrt{2}}{2})\). We will construct the unit sphere, i.e. the graph of the function \(g(x, y, z) = x^2 + y^2 + z^2 - 1 = 0\), and three surfaces corresponding the equations:

\[
\tau(x, y, z) \equiv \sigma_{nt}(x, y, z) = \frac{\sigma_{33} - \sigma_{22}}{2}, \quad (11a)
\]
\[
\tau(x, y, z) \equiv \sigma_{nt}(x, y, z) = \frac{\sigma_{33} - \sigma_{11}}{2}, \quad (11b)
\]
\[
\tau(x, y, z) \equiv \sigma_{nt}(x, y, z) = \frac{\sigma_{22} - \sigma_{11}}{2}. \quad (11c)
\]

Since the unit sphere intersects the graph of the functions \((11a)\) and \((11c)\), the corresponding stationary points \((\pm \frac{\sqrt{2}}{2}, \pm \frac{\sqrt{2}}{2}, 0)\) and \((0, \pm \frac{\sqrt{2}}{2}, \pm \frac{\sqrt{2}}{2})\) are saddle points. In the second case the unit sphere touches the graph of the function \((11b)\), so the corresponding stationary points \((\pm \frac{\sqrt{2}}{2}, 0, \pm \frac{\sqrt{2}}{2})\), are Extrema.

All these facts can be seen easily from Figs. 4, 5, 6 and 7 and can be obtained from graphical programming tools like Sage 10, Octave 3.2.4 and Python 3.6.5 using the corresponding programming codes presented in the Appendix. The principal axes in Figs. 4, 5, 6 and 7, are denoted with indices 1, 2 and 3.

The stationary points in Figs 4(b), 5(b), 6(b), and 7(b), are Extrema, since the unit ball touches the graph of the function \(f = \tau_{\text{max}}\) at these points. The stationary points in Figs. 4(a), 5(a), 6(a), 7(a) and 4(c), 5(c), 6(c), 7(c) are saddle points, since the unit ball intersects the graph of the function \(f = c\) for \(c = \frac{\sigma_{33} - \sigma_{22}}{2}\) and \(c = \frac{\sigma_{22} - \sigma_{11}}{2}\), respectively. The corresponding programming codes written in Sage 4.6.2, Octave 2.10.1, Python and Octave 3.2.4 are presented in the Appendix.

IV. CONCLUSION

In the paper the extreme shear stresses with the corresponding planes were determined. At first step, the nature of the...
stationary points obtained by the Method of Lagrangian Multipliers is established symbolically using the freely available symbolic tool Sage [10]. Next, the new simple graphical procedure to establish the character of the stationary points is presented, employing the freely available graphical tools like Sage [10], R [11], Python [12] and Octave [13]. The same quality of graphical presentation could be achieved as with the commercial ones like MATLAB and Mathematica using the same length of programming code. The authors found all these tools helpful in their teaching process. The presented figures help the students of civil or mechanical engineering to improve their understanding of the problem and the obtained solutions.

V. APPENDIX: APPLIED GRAPHICS PROGRAMMING CODES

The surfaces shown in Figs. 3, 4, 5, 6 and 7 were obtained by the usage of the GnuPlot [9], Sage [10], R [11], Python [12] and Octave [13] program codes presented below. The Octave graphics is principally based on GnuPlot graphics, which is well suitable for two dimensional presentation, but less suitable for three dimensional plots. In order to avoid this inconvenience, the Octave vrml package [15] was included. Applying this package, high quality three dimensional graphics plot can be constructed in standardized vrml or wrl format [16], by readable using well known programming tools like view3dscene [17] or FreeWRL [18].

A. GnuPlot 4.4

```plaintext
# notation according to article: x = r, y = z
reset
# principal stresses
sig1 = 1.0 # sig2 = 1.0
sig3 = 4.0
tau_max = abs(sig3 - sig1)/2 # extreme shear stress
circle(x, y) = sqrt(x**2+y**2) - tau_max
set terminal wxt
set size square
set xlabel "r"; set ylabel "z"
staylorpoints = 
"(x,-1.5,-1.5, y,-1.5, 1.5, 2x,-1.5, 1.5, 2, x, 1.5, 2)"
pplot = plot(tau(x,y,sig1,sig2,sig3),
"(x,-1.5,-1.5, y,1.5,"
"taumax, abs(c(sig3,sig1,sig2)-c(sig2,sig3))/2)"
"(x,-2, 2, y,-2, 2)"
"(x,2, 2, y,-2, 2)"
"plot_points=100, contour=tau(isp), color=c, opacity=1"
def plotTaulsConstant(c, sig1,sig2,sig3, isp=0):
tau_max = map(abs,c(sig3,sig2)-c(sig2,sig3))/2
return implcit_plot3d(tau(x,y,z,sig1,sig2,sig3),
"(x,-2, 2, y,-2, 2, z,-2, 2)"
"plot_points=100, contour=tau(isp),
"color=c, opacity=1"

def plotStationaryPoints(p, c):
return point(p, rgbcolor=c, pointsize=150)
def plotStationaryArrow(p1, p2, c):
return arrow(p1,p2, rgbcolor=c, width=2, arrowsize=5)
def plotStationaryText(t, p):
return text3d(t(p[0],p[1]), fontsize=80, color='black')
def plotCoordinateSystem(c):
COORD = map(abs,[0,0,0], [1,0,0], [0,1,0])
return (sum(arrow((0,0,0),COORD[1], rgbcolor=c, 
width=2, arrowsize=5) for i in [0..2]) + 
sum(text3d(i+1,COORD[i]*1.05,fontsize=12, 
color=c) for i in [0..2]))
deftauplot(sig1=1,sig2=2,sig3=3, isp=0):
set terminal color
set view3dscene[17] or view3dscene[18]
# stationary point colors
col = ['green', 'blue', 'yellow']
# stationary points annotations
text = ['I', 'II', 'III']
# plot of the unit ball isosurface
B1 = plotUnitBall('blue')
# plot of the isosurface tau = const
T = plotTaulsConstant('red',sig1,sig2,sig3, isp)
# stationary points for isp=1
SPI = matrix([[0,-1,1],[0,1,-1],[0,1,1]])
# stationary points for isp=2
SP2 = matrix([[-1,0,1],[1,0,-1],[1,0,1]])
# stationary points for isp=3
SP3 = matrix([[-1,0,-1],[1,0,1],[1,-1,0],[1,1,0]])
# all stationary points
SP = [SPI, SP2, SP3]
# stationary points annotation
# points
S = plotStationaryPoints(SP[isp],col[isp])
# text
TS = sum(plotStationaryText(t,x,SP[isp],col[isp]))
# for i in [0..3])
# gradients
AS = plotStationaryArrow(SP[isp],x,SP[isp],x, col[isp])
for i in [0..3])
# plot Cartesian coordinate system
with basis e1, e2 and e3
# axes + text
CS = plotCoordinateSystem('black')
# join all figure parts
F = T+B1+5*S+T*S+CS
# tau is const + unit sphere
+ stationary points + coor system
return F
```

B. Sage 4.6.2

```plaintext
def ball(x,y,z):
return sqrt(x^2+y^2+z^2)
def tau(x,y,z, sig1=1,sig2=2,sig3=3):
return ball(sig1-sig2)*x*y, (sig1-sig3)*x*z, (sig2-sig3)*y*z)
def plotUnitBall(c):
```

C. R 2.10.1

```plaintext
# source('tauplot.R'); f = tauplot(1,2,4,1)
library(rgl) # base 3d graphics library
library(misc3d) # miscellaneous 3d graphics library
library(rgl) # base 3d graphics library
library(misc3d) # miscellaneous 3d graphics library
ball = function(x,y,z) { sqrt(x^2+y^2+z^2) }
tau = function(x,y,z,sig1,sig2,sig3):
ball((sig1-sig2)*x*y,(sig1-sig3)*x*z,(sig2-sig3)*y*z)

plotUnitBall = function(balls='blue'){
for i in [0..3])
# plot Cartesian coordinate system
with basis e1, e2 and e3
# axes + text
CS = plotCoordinateSystem('black')
# join all figure parts
F = T+B1+5*S+T*S+CS
# tau is const + unit sphere
+ stationary points + coor system
return F
```
D. Octave 3.2.4

function s = tauplot2(sig11, sig22, sig33, isp) % call: tauplot(sig11, sig22, sig33, isp) % function plots the tau surface % taumax with the unit ball (inserted) % input parameters: % sig11, sig22, sig33 are the principal stresses % isp is the index of the plot group of % the stationary points
    if nargin < 3, sig11 = 1; sig22 = 2; sig33 = 3; end
    if nargin < 4, isp=1; end
    % ball
    R = @(x,y,z) sqrt((x.^2 + y.^2 + z.^2)); % tau max with the unit ball (inserted)
    tau = @(x,y,z) sig11.*x.*z + sig22.*y.*z + sig33.*x.*y;
    % stationary (extreme) shear stresses
    % stationary shear stresses
    taup = abs(tau([2 3 1])) - abs(tau([3 1 2]))/2; % max shear stress
    taumax = max(abs(tausp));
    if taumax == 0, error(’Hidrostatic stress state.’); end
    % domain plot points
    xmax = 2; ymax = 2; zmax = 2; d = 1/0.01:1;
    X = d.*xmax; Y = d.*ymax; Z = d.*zmax;
    X,Y,Z = meshgrid(X,Y,Z);
    % vrml plot of the unit ball
    R = @(X,Y,Z) isosurface(X,Y,Z,tau,X,Y,Z,taumax(isp),T);
    s2 = isosurface(X,Y,Z,tau,X,Y,Z,taumax(isp),T);
    % vrml plot of the stationary/extreme shear directions
    s22 = sqrt(2)/2;
    for i = 1:4
        stxt = {'I','II','III'};
        sp{i} = vrml_arrowAB(spt{i}*0,2*spt{i},[0 1 1]);
        eg{i} = vrml_textT(stxt{isp},[1 1 1]);
    end
    % vrml plot of the stationary/text points
    bfs = vrml_faces(V', F', “col”, [1 0 0],’tran’,0.0);
    s1 = vrml_faces(V’, F’, “col”, [0 0 1],’tran’,0.0);
    for i = 1:4
        spx{i} = vrml_textT(stxt{isp},2*spt{i});
        sp{i} = vrml_arrowAB(spt{i}*0,2*spt{i},[0 1 1]);
        egx{i} = vrml_textT(stxt{isp},[1 1 1]);
    end
    % vrml background
    bkg = vrml_Background(“skyColor”,[1 1 1]);
    % join vrml plots together
    fr = vrml_frame([0 0 0],[0 0 pi],’scale’,0.5);
    save_vrml(’tauplot.wrl’,’nobg’,s1,s2, ... 
        egx{1},egx{2},egx{3}, ... 
        spx{1},spx{2},spx{3}, spx{4}, ... 
    end
    % vrml plot of the stationary/text points
    bfs = vrml_faces(V’, F’, “col”, [1 0 0],’tran’,0.0);
    s1 = vrml_faces(V’, F’, “col”, [0 0 1],’tran’,0.0);
    for i = 1:4
        spx{i} = vrml_textT(stxt{isp},2*spt{i});
        sp{i} = vrml_arrowAB(spt{i}*0,2*spt{i},[0 1 1]);
        eg{i} = vrml_textT(stxt{isp},[1 1 1]);
    end
    % vrml background
    bkg = vrml_Background(“skyColor”,[1 1 1]);
    % join vrml plots together
    fr = vrml_frame([0 0 0],[0 0 pi],’scale’,0.5);
    save_vrml(’tauplot.wrl’,’nobg’,s1,s2, ... 
        egx{1},egx{2},egx{3}, ... 
        egx{1},egx{2},egx{3}, ... 
        spx{1},spx{2},spx{3}, spx{4}, ... 
    end
    % vrml plot of the stationary/text points
    bfs = vrml_faces(V’, F’, “col”, [1 0 0],’tran’,0.0);
    s1 = vrml_faces(V’, F’, “col”, [0 0 1],’tran’,0.0);
    for i = 1:4
        spx{i} = vrml_textT(stxt{isp},2*spt{i});
        sp{i} = vrml_arrowAB(spt{i}*0,2*spt{i},[0 1 1]);
        eg{i} = vrml_textT(stxt{isp},[1 1 1]);
    end
    % vrml background
    bkg = vrml_Background(“skyColor”,[1 1 1]);
    % join vrml plots together
    fr = vrml_frame([0 0 0],[0 0 pi],’scale’,0.5);
    save_vrml(’tauplot.wrl’,’nobg’,s1,s2, ... 
        egx{1},egx{2},egx{3}, ... 
        egx{1},egx{2},egx{3}, ... 
        spx{1},spx{2},spx{3}, spx{4}, ... 
    end
    % vrml plot of the stationary/text points
    bfs = vrml_faces(V’, F’, “col”, [1 0 0],’tran’,0.0);
    s1 = vrml_faces(V’, F’, “col”, [0 0 1],’tran’,0.0);
    for i = 1:4
        spx{i} = vrml_textT(stxt{isp},2*spt{i});
        sp{i} = vrml_arrowAB(spt{i}*0,2*spt{i},[0 1 1]);
        eg{i} = vrml_textT(stxt{isp},[1 1 1]);
    end
    % vrml background
    bkg = vrml_Background(“skyColor”,[1 1 1]);
    % join vrml plots together
    fr = vrml_frame([0 0 0],[0 0 pi],’scale’,0.5);
    save_vrml(’tauplot.wrl’,’nobg’,s1,s2, ... 
        egx{1},egx{2},egx{3}, ... 
        egx{1},egx{2},egx{3}, ... 
        spx{1},spx{2},spx{3}, spx{4}, ... 
    end
    % vrml plot of the stationary/text points
    bfs = vrml_faces(V’, F’, “col”, [1 0 0],’tran’,0.0);
    s1 = vrml_faces(V’, F’, “col”, [0 0 1],’tran’,0.0);
    for i = 1:4
        spx{i} = vrml_textT(stxt{isp},2*spt{i});
        sp{i} = vrml_arrowAB(spt{i}*0,2*spt{i},[0 1 1]);
        eg{i} = vrml_textT(stxt{isp},[1 1 1]);
    end
    % vrml background
    bkg = vrml_Background(“skyColor”,[1 1 1]);
    % join vrml plots together
    fr = vrml_frame([0 0 0],[0 0 pi],’scale’,0.5);
    save_vrml(’tauplot.wrl’,’nobg’,s1,s2, ... 
        egx{1},egx{2},egx{3}, ... 
        egx{1},egx{2},egx{3}, ... 
        spx{1},spx{2},spx{3}, spx{4}, ... 
    end
}
E. Python

```python
import numpy as np
from mayavi import mlab
import numpy as np

# sphere
def R(x, y, z):
    # sphere
    # plot of the surface tau(x,y,z) = c
    return sqrt(x*x+y*y+z*z)

# plot line
def plotline(x0, y0, z0, x1, y1, z1):
    a = (x1-x0)/(y1-y0)
b = (y1-y0)/(x1-x0)
c = (x1-x0)*(y1-y0)-d
f = abs(a-b)+abs(a-c)+abs(b-c) + abs(np.sign(a)-1)
return f

d = sqrt((x1-x0)**2 +(y1-y0)**2 +(z1-z0)**2)

# plot plane
def plotplane(x0, y0, z0, (x1, y1, z1)):
    a = (x-x0)/(y-y0)
b = (y-y0)/(x-x0)
c = (x-x0)*(y-y0)-d
f = abs(a-b)+abs(a-c)+abs(b-c) + abs(np.sign(a)-1)
return f

d = sqrt((x1-x0)**2 +(y1-y0)**2 +(z1-z0)**2)

# plot arrow
def plotarrow(x0, y0, z0, x1, y1, z1):
    a = 0.35
    b = 1.0-a
    x = a*x0 + b*x1
    y = a*y0 + b*y1
    z = a*z0 + b*z1
d = R(xm-x0,ym-y0,zm-z0)

# plot points
def plotpoints(x0, y0, z0, s=1):
    a = abs(a-b) + abs(a-c) + abs(b-c) + abs(np.sign(a)-1)

# plot unitsphere
def plotunitsphere(col=(0.0, 0.0, 1.0)):
    # plot unit sphere
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot constau
def plotconstau(sig11=1, sig22=2, sig33=3, c=1, col=(1.0, 0.0, 0.0)):
    # plot of the surface tau(x,y,z) = c
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot cone
def plotcone(x0, y0, z0, x1, y1, z1, s=1):
    # plot cone
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot pline
def plotcline(x0, y0, z0, x1, y1, z1):
    # plot pline
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot pline
def plotpline(x0, y0, z0, x1, y1, z1):
    # plot pline
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot text3d
def plottext3d(a, b, c, t):
    # plot text3d
    mlab.text3d(a, b, c, t, scale=5)

# plot arrow
def plotarrow(x0, y0, z0, x1, y1, z1):
    a = 0.35
    b = 1.0-a
    x = a*x0 + b*x1
    y = a*y0 + b*y1
    z = a*z0 + b*z1
d = R(xm-x0,ym-y0,zm-z0)

# plot plane
def plotplane(x0, y0, z0, x1, y1, z1, s=1):
    # plot plane
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot constau
def plotconstau(sig11=1, sig22=2, sig33=3, c=1, col=(1.0, 0.0, 0.0)):
    # plot of the surface tau(x,y,z) = c
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot cone
def plotcone(x0, y0, z0, x1, y1, z1, s=1):
    # plot cone
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot pline
def plotpline(x0, y0, z0, x1, y1, z1):
    # plot pline
    return R((sig11-sig22)*x*y, (sig22-sig33)*y*z, (sig11-sig33)*x*z)

# plot text3d
def plottext3d(a, b, c, t):
    # plot text3d
    mlab.text3d(a, b, c, t, scale=5)
```

G = [(2.0,0,0),(0,2.0,0),(0,0,2.0)]
plotunitsphere()
plotconstau(sig11,sig22,sig33,tausp[isp])
for sp in SP[isp]:
    plotsphere(sp,0.1)
    plottext3d(textcoor(tuple(1.2*NA(sp))),
        txtSP[isp])
    plotarrow(sp,tuple(2*NA(sp)))
for (e,g,txt) in zip(E,G,txtG):
    plottext3d(textcoor(tuple(1.2*NA(g))),txt)
    plotarrow(e,g)

# mlab.show()
mlab.draw()
name = ['Fig1.png', 'Fig2.png', 'Fig3.png']
mlab.savefig(name[isp])
mlab.close()
return 1

ACKNOWLEDGMENT

The work was partially supported by the Slovenian Research Agency through the grant P2-0260. The support is gratefully acknowledged.

REFERENCES