In [1]:
```
import numpy as np
```

In [2]:
```
np.zeros(3)
```

Out[2]:
```
array([0., 0., 0.])
```

In [3]:
```
# the above code was just used to test to see if the notebook was working
```

In [4]:
```
# this entry is testing how to use latex within jupyter notebooks
#$$ Q1: Construct the Products A\dotB and B\dotA, with A and B being:$$

A = np.array([[1,2,3],[4,5,6],[7,8,9]])
B = np.array([[10,11,12],[13,14,15],[16,17,18]])
```

In [5]:
```
%%latex
Q1: Construct the Products $A \cdot B$ and $B \cdot A$, with $A$ and $B$ being
$$
A = \begin{bmatrix}
        1 & 2 & 3 \\
        4 & 5 & 6\\
        7 & 8 & 9 \\
        \end{bmatrix},
B = \begin{bmatrix}
        10 & 11 & 12 \\
        13 & 14 & 15\\
        16 & 17 & 18 \\
        \end{bmatrix}
$$
```

Q1: Construct the Products \$A \cdot B\$ and \$B \cdot A\$, with \$A\$ and \$B\$ being \$\$ A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6\\ 7 & 8 & 9 \\ \end{bmatrix}, B = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15\\ 16 & 17 & 18 \\ \end{bmatrix} \$\$

In [6]:
```
# using sympy in this section. not sure how to print these nicely.
from sympy import *
A = Matrix([[1,2,3],[4,5,6],[7,8,9]])
B = Matrix([[10,11,12],[13,14,15],[16,17,18]])
```

In [7]:
```
print("A*B = ")
A*B
```

```
A*B =
```

Out[7]: \$\displaystyle \left[\begin{matrix}84 & 90 & 96\\201 & 216 & 231\\318 & 342 & 366\end{matrix}\right]\$

In [8]:
```
print("B*A = ")
B*A
```

```
B*A =
```

Out[8]: $\displaystyle \left[\begin{matrix}138 & 171 & 204\\174 & 216 & 258\\210 & 261 & 312\end{matrix}\right]$

In [9]:
```
%%latex
Q2: Using the matrix $M$ defined below, determine $M^T$, the symmetric part of $
$$
M = \begin{bmatrix}
        4 & 5 & -5 \\
        -1 & 3 & -1\\
        7 & 1 & 1 \\
        \end{bmatrix}
$$
```

Q2: Using the matrix M defined below, determine M^T, the symmetric part of M, and the skew-symmetic part of M $$ M = \begin{bmatrix} 4 & 5 & -5 \\ -1 & 3 & -1\\ 7 & 1 & 1 \\ \end{bmatrix} $$

In [10]:
```
M = Matrix([[4,5,-5],[-1,3,-1],[7,1,1]])
M_transpose = M.T
```

In [11]:
```
%%latex
The symmetric part of $M$, $M_S$, is defined as
$$M_S = \frac{1}{2}(M + M_T)$$
And the skew-symmetic part of $M$, $M_K$, is defined as
$$M_K = \frac{1}{2}(M - M_T)$$
```

The symmetric part of M, M_S, is defined as $$M_S = \frac{1}{2}(M + M_T)$$ And the skew-symmetic part of M, M_K, is defined as $$M_K = \frac{1}{2}(M - M_T)$$

In [12]:
```
M_S = (1/2)*(M + M_transpose)
M_K = (1/2)*(M - M_transpose)
```

In [13]:
```
print("Symmetric part of M = ")
M_S
```

```
Symmetric part of M =
```

Out[13]: $\displaystyle \left[\begin{matrix}4.0 & 2.0 & 1.0\\2.0 & 3.0 & 0\\1.0 & 0 & 1.0\end{matrix}\right]$

In [14]:
```
print("Skew-symmetric part of M = ")
M_K
```

```
Skew-symmetric part of M =
```

Out[14]: $\displaystyle \left[\begin{matrix}0 & 3.0 & -6.0\\-3.0 & 0 & -1.0\\6.0 & 1.0 & 0\end{matrix}\right]$

In [15]:
```
%%latex
Q3: Show that the trace of the product $A \cdot B = 0$ if $A$ is symmetric and $
```

Q3: Show that the trace of the product $A \cdot B = 0$ if A is symmetric and B is skew-symmetric

In [16]:
```
# use different variables to show that it is ANY symmetric and skew-symmetric ma
a1,b1,c1,d1,e1,f1,a2,b2,c2,d2,e2,f2 = symbols('a1 b1 c1 d1 e1 f1 a2 b2 c2 d2 e2
A = Matrix([[a1,b1,c1],[b1,d1,e1],[c1,e1,f1]])
B = Matrix([[0,b2,c2],[-1*b2,0,e2],[-1*c2,-1*e2,0]])
print("Generic symmetric matrix A = ")
print(pretty(A))
print("Generic skew-symmetric matrix B = ")
print(pretty(B))
```

```
Generic symmetric matrix A =
⎡a₁  b₁  c₁⎤
⎢          ⎥
⎢b₁  d₁  e₁⎥
⎢          ⎥
⎣c₁  e₁  f₁⎦
Generic skew-symmetric matrix B =
⎡ 0   b₂   c₂⎤
⎢            ⎥
⎢-b₂   0   e₂⎥
⎢            ⎥
⎣-c₂  -e₂  0 ⎦
```

In [17]:
```
%%latex
Note that for a matrix to be skew-symmetric, its main diagonal must be equal to
$A^T = -A$ is the definition of a skew-symmetric matrix, and $x = -x$ iff $x = 0
We will now compute the trace of $A \cdot B$, which is show to be 0 for generic
symmetric and skew-symmetric matricies.
```

Note that for a matrix to be skew-symmetric, its main diagonal must be equal to zero because $A^T = -A$ is the definition of a skew-symmetric matrix, and $x = -x$ iff $x = 0$ $\forall$ x $\in R$ We will now compute the trace of $A \cdot B$, which is show to be 0 for generic symmetric and skew-symmetric matricies.

In [18]:
```
# showing that the tr(A*B) = 0
ab = A*B
#Trace(ab) does not work; sympy does not like computing the trace of variables i
#trace of a matrix = sum of the main diagonals
trace = ab[0,0] + ab[1,1] + ab[2,2]
print("The trace is equal to: " + str(trace))
```

```
The trace is equal to: 0
```

In [19]:
```
%%latex
Q4: Find the derivative of $A(t)$, where
$$
A(t) = \begin{bmatrix}
        t & t^2 & sin(\omega t) \\
        cosh(t) & ln(t) & 17t\\
        1/t & 1/(t^2) & ln(t^2) \\
        \end{bmatrix}
$$
```

Q4: Find the derivative of $A(t)$, where $$ A(t) = \begin{bmatrix} t & t^2 & sin(\omega t) \\ cosh(t) &

ln(t) & 17t\\ 1/t & 1/(t^2) & ln(t^2) \\ \end{bmatrix} $$

In [20]:
```
t,w = symbols('t,w')
A = Matrix([[t,t**2,sin(w*t)],[cosh(t),ln(t),17*t],[1/t,1/(t**2),ln(t**2)]])
A_dx = A.diff(t)
print("The derivative of A(t) is:")
A_dx
```

```
The derivative of A(t) is:
```

Out[20]: $\displaystyle \left[\begin{matrix}1 & 2 t & w \cos{\left(t w \right)}\\\sinh{\left(t \right)} & \frac{1}{t} & 17\\- \frac{1}{t^{2}} & - \frac{2}{t^{3}} & \frac{2}{t}\end{matrix}\right]$

In [21]:
```
%%latex
Q5: Find the eigenvalues and eigenvectors of the matrix $G$;
$$
G = \begin{bmatrix}
        10 & 50 & -50 \\
        50 & 100 & 10\\
        -50 & 10 & 100 \\
        \end{bmatrix}
$$
```

Q5: Find the eigenvalues and eigenvectors of the matrix G; $$ G = \begin{bmatrix} 10 & 50 & -50 \\ 50 & 100 & 10\\ -50 & 10 & 100 \\ \end{bmatrix} $$

In [22]:
```
G = Matrix([[10,50,-50],[50,100,10],[-50,10,100]])
#G_np = np.array([[1,2,3],[2,4,5],[3,5,6]])
eigen = list(G.eigenvals().keys())
eigen_vec = G.eigenvects()
#eig_np,x = np.linalg.eig(G_np)
```

In [23]:
```
#eig_np
print("Eigenvalues of G:")
print(float(eigen[0]))
print(float(eigen[1]))
print(float(eigen[2]))
print("Corresponding Eigenvectors of G:")
print(pretty(eigen_vec[0][2][0]))
print(pretty(eigen_vec[1][2][0]))
print(pretty(eigen_vec[2][2][0]))
```

```
Eigenvalues of G:
110.0
-31.240384046359605
131.2403840463596
Corresponding Eigenvectors of G:
⎡0⎤
⎢ ⎥
⎢1⎥
⎢ ⎥
⎣1⎦
⎡4   √66⎤
⎢─ + ───⎥
⎢5    5 ⎥
⎢       ⎥
```

```
   -1

    1
 4    √66
 ─ - ────
 5     5

   -1

    1
```

In [24]:

```
%%latex
Q6: Determine $G^{-1}$ then show that the eigenvalues of $G^{-1}$ are the invers
```

Q6: Determine $G^{−1}$ then show that the eigenvalues of $G^{−1}$ are the inverse of the eigenvalues of G.

In [25]:

```
#G_np_inv = np.linalg.inv(G_np)
#G_np_inv_eig,z = np.linalg.eig(G_np_inv)
G_inv = G.inv()
G_inv
```

Out[25]: $\displaystyle \left[\begin{matrix}- \frac{9}{410} & \frac{1}{82} & - \frac{1}{82}\\\frac{1}{82} & \frac{3}{902} & \frac{13}{2255}\\- \frac{1}{82} & \frac{13}{2255} & \frac{3}{902}\end{matrix}\right]$

In [26]:

```
G_inv_eig = list(G_inv.eigenvals().keys())
for i in range(0,3):
    print("Inverse of eigenvalue " + str(i+1) + " of G and the corresponding eig
    print(float(1/eigen[i]))
    print(float(G_inv_eig[i]))
print("Clearly, the above values are equal")
```

```
Inverse of eigenvalue 1 of G and the corresponding eigenvalue of G_inv:
0.00909090909090909
0.00909090909090909
Inverse of eigenvalue 2 of G and the corresponding eigenvalue of G_inv:
-0.03200984976740478
-0.03200984976740478
Inverse of eigenvalue 3 of G and the corresponding eigenvalue of G_inv:
0.007619605864965757
0.007619605864965757
Clearly, the above values are equal
```

In [27]:

```
#^ need to figure out how to index these; definitely are inverses of each other.
```

In [28]:

```
%%latex
Q7 For the matrix $T$ with the following Cartesian components
$$
T = \begin{bmatrix}
        1 & 0 & 0 \\
        0 & \frac{11}{4} & -\frac{\sqrt{3}}{4}\\
        0 & -\frac{\sqrt{3}}{4} & \frac{9}{4} \\
        \end{bmatrix}
$$
```

Q7 For the matrix T with the following Cartesian components $$ T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{11}{4} & -\frac{\sqrt{3}}{4}\\ 0 & -\frac{\sqrt{3}}{4} & \frac{9}{4} \\ \end{bmatrix} $$

In [29]:

```
# a) Find the eigenvalues and eigenvectors
# b) Find the angle between the first eigenvector and the x1 access
# c) That the eigenvectors form an orthonormal basis

T = np.array([[1,0,0],[0,11/4,-1*np.sqrt(3)/4],[0,-1*np.sqrt(3)/4,9/4]])
# eigenvalues and eigenvectors
T_eig,T_eigVec = np.linalg.eig(T)
# need to order these
print("Eigenvalues:")
print(np.sort(T_eig)[::-1])
# order these by size
print("First eigenvector:")
print(T_eigVec[1])
print("Second eigenvector:")
print(T_eigVec[2])
print("Third Eigenvector:")
print(T_eigVec[0])
# angle between first eigenvalue and x1 axis
theta_x1 = np.arccos(T_eigVec[1][0]/(np.sqrt(T_eigVec[1][0]**2 + T_eigVec[1][1]*
print("Angle between first eigenvector and the x1 axis = " + str(theta_x1))
# Show that the eigenvectors form an orthonormal basis
print("Showing that all the eigenvectors are unit vectors:")
for j in range(0,3):
    print("norm of eigenvector " + str(T_eigVec[j]) + ": " + str(np.linalg.norm(
# show that they are all orthogonal
print("the inner product of all combinations of eigenvectors is 0")
print(np.dot(T_eigVec[0],T_eigVec[1]))
print(np.dot(T_eigVec[0],T_eigVec[2]))
print(np.dot(T_eigVec[1],T_eigVec[2]))
print("---hence, the eigenvectors form an orthonormal basis---")
```

```
Eigenvalues:
[3. 2. 1.]
First eigenvector:
[-0.5       -0.8660254  0.       ]
Second eigenvector:
[-0.8660254  0.5        0.       ]
Third Eigenvector:
[0. 0. 1.]
Angle between first eigenvector and the x1 axis = 2.0943951023931957
Showing that all the eigenvectors are unit vectors:
norm of eigenvector [0. 0. 1.]: 1.0
norm of eigenvector [-0.5       -0.8660254  0.       ]: 1.0
norm of eigenvector [-0.8660254  0.5        0.       ]: 1.0
the inner product of all combinations of eigenvectors is 0
0.0
0.0
0.0
---hence, the eigenvectors form an orthonormal basis---
```

In [30]:

```
%%latex
2. The Stress tensor $\\$
Q8: The state of stress at a certain point in a body is given by
$$
T = \begin{bmatrix}
        1 & 2 & 3 \\
        2 & 4 & 5\\
```

```
        3 & 5 & 0 \\
        \end{bmatrix} {MPa}
$$
On each of the coordinate planes (with normals $e_1$,$e_2$,$e_3$),
determine the normal stress and the magnitude of the
total shear stress.
$\\$
Answer: the normal stress on each coordinate plane $e_i$ is equivalent to $T_{ii
and the shear stress is equivalent to the magnitude of $T_{ij}$, where $i \neq j
Hence, we have as follows:
```

2. The Stress tensor $\\$ Q8: The state of stress at a certain point in a body is given by $$ T = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5\\ 3 & 5 & 0 \\ \end{bmatrix} {MPa} $$ On each of the coordinate planes (with normals e_1,e_2,e_3), determine the normal stress and the magnitude of the total shear stress. $\\$ Answer: the normal stress on each coordinate plane e_i is equivalent to T_{ii}, and the shear stress is equivalent to the magnitude of T_{ij}, where $i \neq j$. Hence, we have as follows:

In [31]:

```
T = Matrix([[1,2,3],[2,4,5],[3,5,0]])
print("Answer to Question 8:")
for i in range(0,3):
    print("On the plane e" + str(i+1) + ":")
    print("Normal stress = "  + str(T[i,i]))
    print("Magnitude of Total Shear Stress = " + str(sqrt(T[i,i-1]**2. + T[i,i-2
```

```
Answer to Question 8:
On the plane e1:
Normal stress = 1
Magnitude of Total Shear Stress = 3.60555127546399
On the plane e2:
Normal stress = 4
Magnitude of Total Shear Stress = 5.38516480713450
On the plane e3:
Normal stress = 0
Magnitude of Total Shear Stress = 5.83095189484530
```

In [32]:

```
%%latex
Q9: The state of stress at a certain point in a body is given by
$$
T = \begin{bmatrix}
        2 & -1 & 3 \\
        -1 & 4 & 0\\
        3 & 0 & -1 \\
        \end{bmatrix} {MPa}
$$
For the plane defined by its normal n = $2e_1 + 2e_2 + e_3$, find the traction (
and the magnitude of the normal and shear stress.
```

Q9: The state of stress at a certain point in a body is given by $$ T = \begin{bmatrix} 2 & -1 & 3 \\ -1 & 4 & 0\\ 3 & 0 & -1 \\ \end{bmatrix} {MPa} $$ For the plane defined by its normal n = $2e_1 + 2e_2 + e_3$, find the traction (or stress) vector, and the magnitude of the normal and shear stress.

In [ ]:

In [33]:
```
# Q9 Answer:
print("Q9: Answer")
n = Matrix([2,2,1])
T = Matrix([[2,-1,3],[-1,4,0],[3,0,-1]])
# t = T*n
t = T*n
print("Traction vector t = ")
print(pretty(t))
# magnitude of normal stress on this plane = t dot n
normal_stress = t.dot(n)
# magnitude of shear stress on this plane = magnitude(t  - normal_stress*n)
shear_stress = t - normal_stress*n
shear_stress_norm = shear_stress.norm()
print("Normal Stress = " + str(normal_stress))
print("Shear Stress = ")
print(pretty(shear_stress))
```

```
Q9: Answer
Traction vector t =
⎡5⎤
⎢ ⎥
⎢6⎥
⎢ ⎥
⎣5⎦
Normal Stress = 27
Shear Stress =
⎡-49⎤
⎢   ⎥
⎢-48⎥
⎢   ⎥
⎣-22⎦
```

In [34]:
```
%%latex
Q10: Given $T_{11} = 1$ MPa and $T_{22} = -1$ MPa, with all other $T_{ij} = 0$ a
continuum, show that the only plane in which the traction vector is zero is
the plane whose normal is $e_3$.
```

Q10: Given $T_{11} = 1$ MPa and $T_{22} = -1$ MPa, with all other $T_{ij} = 0$ at a point in a continuum, show that the only plane in which the traction vector is zero is the plane whose normal is e_3.

In [35]:
```
T = Matrix([[1,0,0],[0,1,0],[0,0,0]])
# define arbitrary constants that give the components of an arbitrary normal vec
x, y, z = symbols('x y z')
n = Matrix([x,y,z])
t = T*n
print("Q10 Answer:")
print(pretty(t))
print("Regardless of the choice of plane, the only plane in which the tractionve
```

```
Q10 Answer:
⎡x⎤
⎢ ⎥
⎢y⎥
⎢ ⎥
⎣0⎦
Regardless of the choice of plane, the only plane in which the tractionvector is
zero
is the plane e3 (the third entry in the vector.)
```

In [36]:

```
%%latex
3. The Strain tensor $\\$
Q11: The strain tensor at a point is given by
$$
\epsilon = \begin{bmatrix}
        1 & -3 & \sqrt{2} \\
        -3 & 1 & -\sqrt{2} \\
        \sqrt{2} & -\sqrt{2} & 4 \\
        \end{bmatrix}
$$
Determine the extensional (i.e. normal) and shear strains in the direction
$n = (1/2)e_1 - (1/2)e_2 + (1/\sqrt{2})e_3$
```

3. The Strain tensor $\\$ Q11: The strain tensor at a point is given by $$ \epsilon = \begin{bmatrix} 1 & -3 & \sqrt{2} \\ -3 & 1 & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & 4 \\ \end{bmatrix} $$ Determine the extensional (i.e. normal) and shear strains in the direction $n = (1/2)e_1 - (1/2)e_2 + (1/\sqrt{2})e_3$

In [37]:

```
# are these supposed to magnitudes, or vectors?
e = Matrix([[1,-3,sqrt(2)],[-3,1,-1*sqrt(2)],[sqrt(2),-1*sqrt(2),4]])
n = Matrix([1/2,-1/2,1/sqrt(2)])
e_n = e*n
e_n_normal = e_n.dot(n)
#test = Transpose(n)*e*n
print("Normal Strain in the given direction n = " + str(e_n_normal))
shear_strain = e_n - e_n_normal*n
shear_strain_norm = shear_stress.norm()
print("Shear Strain along the given direction n =" + str(pretty(shear_strain_no
```

```
Normal Strain in the given direction n = 6.00000000000000
Shear Strain along the given direction n =√5189
```

In [38]:

```
%%latex
Q12:  A displacement field is given by $u_1 = 3x_1x_2^2$, $u_2 = 2x_3x_1,$ and $
Determine the strain tensor $\epsilon$.
```

Q12: A displacement field is given by $u_1 = 3x_1x_2^2$, $u_2 = 2x_3x_1,$ and $u_3=x^{2}_3 − x_1x_2$. Determine the strain tensor ϵ.

In [39]:

```
x1,x2,x3 = symbols('x1 x2 x3')
u = Matrix([3*x1*x2**2,2*x3*x1,x3**2 - x1*x2])
# normal strains
exx = diff(u[0],x1)
eyy = diff(u[1],x2)
ezz = diff(u[2],x3)
# shear strains
exy = (1/2)*(diff(u[0],x2) + diff(u[1],x1))
exz = (1/2)*(diff(u[0],x3) + diff(u[2],x1))
eyz = (1/2)*(diff(u[1],x3) + diff(u[2],x2))
# assembly of the strain tensor
e = Matrix([[exx,exy,exz],[exy,eyy,eyz],[exz,eyz,ezz]])
print("Strain Tensor e:")
e
```

```
Strain Tensor e:
```

Out[39]: $\displaystyle \left[\begin{matrix}3 x_{2}^{2} & 3.0 x_{1} x_{2} + 1.0 x_{3} & - 0.5 x_{2}\\3.0 x_{1} x_{2} + 1.0 x_{3} & 0 & 0.5 x_{1}\\- 0.5 x_{2} & 0.5 x_{1} & 2 x_{3}\end{matrix}\right]$

In [40]:

```
%%latex
Q13: A displacement field is given by $x_1 = X_1+AX_3, x_2 = X_2, and x_3 = X_3-
Calculate the volume change ΔV and show that ΔV≈0 if A is a very small constant.
```

Q13: A displacement field is given by $x_1 = X_1+AX_3, x_2 = X_2, and x_3 = X_3−AX_1$. Calculate the volume change ΔV and show that ΔV≈0 if A is a very small constant.

In [41]:

```
%%latex
Note: as given in class, one way to look at volume change is the following:
    $\Delta V = \frac{V_{def} - V_{init}}{V_{init}} = \epsilon_{11} + \epsilon_{
```

Note: as given in class, one way to look at volume change is the following: $\Delta V = \frac{V_{def} - V_{init}}{V_{init}} = \epsilon_{11} + \epsilon_{22} + \epsilon_{33}$

In [42]:

```
%%latex
We will instead use the deformation gradient F to calculate the change in volume
$$
    F_{ij} = \frac{\partial{x_i}}{\partial{X_j}}
$$
Following this formula, we find that in this case,
$$
F = \begin{bmatrix}
        1 & 0 & A \\
        0 & 1 & 0 \\
        -A & 0 & 1 \\
        \end{bmatrix}
$$
```

We will instead use the deformation gradient F to calculate the change in volume; $$ F_{ij} = \frac{\partial{x_i}}{\partial{X_j}} $$ Following this formula, we find that in this case, $$ F = \begin{bmatrix} 1 & 0 & A \\ 0 & 1 & 0 \\ -A & 0 & 1 \\ \end{bmatrix} $$

In [43]:

```
A = symbols('A')
F = Matrix([[1,0,A],[0,1,0],[-A,0,1]])
# the determinant of F, the deformation gradient, is equal to the scale factor o
J = det(F)
print("General formula for the change in volume of the given displacement field:
J
```

```
General formula for the change in volume of the given displacement field:
```

Out[43]: $\displaystyle A^{2} + 1$

In [44]:

```
%%latex
If A is a very small constant, A << 1, thus we can state
$A^2 + 1 \approx 1$
Thus, as the scale factor of the volume change is 1, we can state that $\Delta V
```

If A is a very small constant, A << 1, thus we can state $A^2 + 1 \approx 1$ Thus, as the scale factor of the volume change is 1, we can state that $\Delta V \approx 0.$

In [ ]:

```

```

In [ ]:

```

```