

1 Library laden

Laden von tVar und Festlegen eines Zahlenformates

```
require("../tVar.lua")
tVar.numFormat = "%.3f"
```

2 U-Wert Berechnung

Berechnung des U-Wertes eines Bauteiles mit 2 Schichten.

```
-- Variable Rse mit dem Wert 0.04 und der Latex Darstellung r_{se}
rse = tVar:New(0.04,"r_{se}")
rsi = tVar:New(0.13,"r_{si}")

d1 = tVar:New(20,"d_{1}")
lambda1 = tVar:New(0.035,"\\lambda_{1}")

d2 = tVar:New(10,"d_{2}")
lambda2 = tVar:New(0.5,"\\lambda_{2}")

R = rse+d1/lambda1+d2/lambda2+rsi

-- R:bracR() erzeugt Rundeklammern und den Term R
U=R:bracR()^(-1)
-- Bei der Ausgabe wird zuerst der Name der Var. angezeigt. Festlegen auf U
U:setName("U") = "U"
```

Output mit U:outFull():

$$U = \left(r_{se} + \frac{d_1}{\lambda_1} + \frac{d_2}{\lambda_2} + r_{si} \right)^{-1} = \left(0.040 + \frac{10.000}{0.035} + \frac{10.000}{0.500} + 0.130 \right)^{-1} = 0.003 \quad (1)$$

Output mit U:outHalf():

$$U = \left(r_{se} + \frac{d_1}{\lambda_1} + \frac{d_2}{\lambda_2} + r_{si} \right)^{-1} = 0.003 \quad (2)$$

Output mit U:outVar():

$$U = 0.003 \quad (3)$$

3 Vergleichsspannung

Berechnung der Vergleichsspannung bei gegebenem σ_x und τ_{xy}

```
sigma_x = tVar:New(11.4,"\\sigma_{x}")
tau_xy = tVar:New(2.43,"\\tau_{xy}")
```

```
-- tVar.sqrt([tVar],[Number])
sigma_v = tVar.sqrt(sigma_x^2+3*tau_xy^2,2)
sigma_v:setName("\\sigma_{v}")
```

Output mit `sigma_v:outFull()`:

$$\sigma_v = \sqrt{\sigma_x^2 + 3.000 \cdot \tau_{xy}^2} = \sqrt{11.400^2 + 3.000 \cdot 2.430^2} = 12.152 \quad (4)$$

3.1 Resultierende Kraft

Berechnen der resultierenden Kraft auf einen Kreisquerschnitt.

```
r = tVar.New(10,"r")
```

```
A=r^2*tVar.PI
F=sigma_v*A
F:setName("F")
```

Output mit `F:outFull()`:

$$F = \sigma_v \cdot r^2 \cdot \pi = \sqrt{11.400^2 + 3.000 \cdot 2.430^2} \cdot 10.000^2 \cdot \pi = 3817.710 \quad (5)$$

Ausführen mit `sigma_v:fix()`:

```
-- fix() löscht die Berechnungsschritte einer Var., damit wird bei jedem Aufruf
    der Var. nur mehr der Name angezeigt
sigma_v:fix()
A=r^2*tVar.PI
F=sigma_v*A
F:setName("F")
```

Output mit `F:outFull()`:

$$F = \sigma_v \cdot r^2 \cdot \pi = 12.152 \cdot 10.000^2 \cdot \pi = 3817.710 \quad (6)$$

4 Matrizen

4.1 Addition und Subtraktion

```
-- Definition durch 2-Dimensionales Array
A=tMat.New({{10,2,5,3},{2,4,3,1}}, "a_{1}")
B=tMat.New({{3,1,4,1},{2,4,5,10}}, "b_{1}")

-- CRLF() fuegt eine Zeilenumbruch ein und achtet auf Klammern
C = A + B:CRLF() -A
C:setName("C")
```

Output mit C:outFull():

$$\mathbf{C} = \mathbf{a}_1 + \mathbf{b}_1 - \mathbf{a}_1 = \begin{pmatrix} 10.000 & 2.000 & 5.000 & 3.000 \\ 2.000 & 4.000 & 3.000 & 1.000 \end{pmatrix} + \begin{pmatrix} 3.000 & 1.000 & 4.000 & 1.000 \\ 2.000 & 4.000 & 5.000 & 10.000 \end{pmatrix} - \begin{pmatrix} 10.000 & 2.000 & 5.000 & 3.000 \\ 2.000 & 4.000 & 3.000 & 1.000 \end{pmatrix} = \begin{pmatrix} 3.000 & 1.000 & 4.000 & 1.000 \\ 2.000 & 4.000 & 5.000 & 10.000 \end{pmatrix} \quad (7)$$

4.2 Multiplikation and Division

```
A=tMat:New({{10,2,5,3},{2,4,3,1}}, "a_{1} ")
B=tMat:New({{3,1},{4,2},{5,8},{2,9}}, "b_{1}")
D=tMat:New({{5,12.2},{3,22}}, "D")

-- Uebergabe Paramter fuer CRLF wird ab ende der Zeile und am Anfang der
naechsten Angezeigt
C = A:CRLF("\ \cdot") * (B*2) - D/2
C=-C:bracR()
C:setName("C")
```

Output mit C:outFull():

$$\mathbf{C} = - \left(\mathbf{a}_1 \cdot \mathbf{b}_1 \cdot 2.000 - \frac{\mathbf{D}}{2.000} \right) = - \left(\begin{pmatrix} 10.000 & 2.000 & 5.000 & 3.000 \\ 2.000 & 4.000 & 3.000 & 1.000 \end{pmatrix} \cdot \begin{pmatrix} 3.000 & 1.000 \\ 4.000 & 2.000 \\ 5.000 & 8.000 \\ 2.000 & 9.000 \end{pmatrix} \cdot 2.000 - \frac{\begin{pmatrix} 5.000 & 12.200 \\ 3.000 & 22.000 \end{pmatrix}}{2.000} \right) = \begin{pmatrix} -135.500 & -155.900 \\ -76.500 & -75.000 \end{pmatrix} \quad (8)$$

4.3 Transponieren

```
A=tMat:New({{10,2,5,3},{2,4,3,1}}, "a_{1}")
C = A:T()
C:setName("C")
```

Output mit C:outFull():

$$\mathbf{C} = \mathbf{a}_1^\top = \begin{pmatrix} 10.000 & 2.000 & 5.000 & 3.000 \\ 2.000 & 4.000 & 3.000 & 1.000 \end{pmatrix}^\top = \begin{pmatrix} 10.000 & 2.000 \\ 2.000 & 4.000 \\ 5.000 & 3.000 \\ 3.000 & 1.000 \end{pmatrix} \quad (9)$$

4.4 Determinante und Inverse

```
A_2=tMat:New({{10,2,5},{2,4,3},{7,4,3}}, "a_{2}")
C = A_2:Det()
Inv = A_2:Inv() * A_2:CRLF("=")
```

```
C:setName("C")
Inv:setName("Inv")
```

Output mit C:outFull():

$$C = |\mathbf{a}_2| = \left| \begin{pmatrix} 10.000 & 2.000 & 5.000 \\ 2.000 & 4.000 & 3.000 \\ 7.000 & 4.000 & 3.000 \end{pmatrix} \right| = -70.000 \quad (10)$$

Output mit Inv:outFull():

$$\begin{aligned} \text{Inv} &= \mathbf{a}_2^{-1} \cdot \mathbf{a}_2 = \begin{pmatrix} 10.000 & 2.000 & 5.000 \\ 2.000 & 4.000 & 3.000 \\ 7.000 & 4.000 & 3.000 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 10.000 & 2.000 & 5.000 \\ 2.000 & 4.000 & 3.000 \\ 7.000 & 4.000 & 3.000 \end{pmatrix} = \\ &= \begin{pmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ -0.000 & -0.000 & 1.000 \end{pmatrix} \end{aligned} \quad (11)$$

5 Vektor Operationen

5.1 Addition und Subtraktion

```
v_1=tVec:New({10,2,7},"v_{1}")
v_2=tVec:New({3,1,2},"v_{2}")
v_3 = (v_1+v_2):bracR()-v_2
v_3:setName("v_{3}")
```

Output mit v_3:outFull():

$$\vec{v}_3 = (\vec{v}_1 + \vec{v}_2) - \vec{v}_2 = \left(\begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix} + \begin{pmatrix} 3.000 \\ 1.000 \\ 2.000 \end{pmatrix} \right) - \begin{pmatrix} 3.000 \\ 1.000 \\ 2.000 \end{pmatrix} = \begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix} \quad (12)$$

Nur den Wert Ausgeben mit v_3:out(): $\begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix}$

5.2 Skalarprodukt und Produkt mit einem Skalar

Ist zur Zeit nur mit R3 Vektoren möglich.

```
v_1=tVec:New({10,2,7},"v_{1}")
v_2=tVec:New({3,1,2},"v_{2}")
v_3 = (v_1*2):bracR()*v_2
v_3:setName("v_{3}")
```

Output mit `v_3:outFull()`:

$$v_3 = (\vec{v}_1 \cdot 2.000) \cdot \vec{v}_2 = \left(\begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix} \cdot 2.000 \right) \cdot \begin{pmatrix} 3.000 \\ 1.000 \\ 2.000 \end{pmatrix} = 92.000 \quad (13)$$

5.3 Kreuzprodukt

Ist zur Zeit nur mit *R3* Vektoren möglich.

```
v_1=tVec:New({10,2,7},"v_{1}")
v_2=tVec:New({3,1,2},"v_{2}")
v_3 = v_1:crossP(v_2)
v_3:setName("v_{3}")
```

Output mit `v_3:outFull()`:

$$\vec{v}_3 = \vec{v}_1 \times \vec{v}_2 = \begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix} \times \begin{pmatrix} 3.000 \\ 1.000 \\ 2.000 \end{pmatrix} = \begin{pmatrix} -3.000 \\ 1.000 \\ 4.000 \end{pmatrix} \quad (14)$$

6 Befehlsreferenz

Wenn nicht anders angegeben, sind alle Befehle entsprechen der Hierarchie **tVar** \Rightarrow **tMat** \Rightarrow **tVec** vererbt.

tVar	
<code>tVar.numFormat = [format]</code>	Definiert das Ausgabe-Zahlenformat [latex]. <code>tVar.numFormat = "%.3f"</code>
<code>tVar:New([value],[latex])</code>	Erzeugt eine neue Variable mit dem Wert [value] und der Latex Darstellung [latex]. <code>rse= tVar:New(0.04,"r_{se}")</code>
<code>[tVar]:setName([latex])</code>	Legt die Latex Darstellung für einen berechneten Wert fest. <code>U:setName("U")</code>
<code>[tVar]:bracR()</code>	Umschließt die Formeldarstellung der Variable [tVar] mit runden Klammer. <code>U = R:bracR()</code>
<code>tVar.sqrt([tVar],[number])</code>	[numbert]te Wurzel aus [tVar] <code>sigma_v = tVar.sqrt(sigma_x^2+3*tau_xy^2,2)</code>

<code>tVar.PI</code>	π <code>A=r^2*tVar.PI</code>
<code>[tVar]:fix([latex])</code>	Bereinigt die Variable von Berechnungsschritten, damit wird bei einer Rechenoperation nicht mehr die Gleichung verwendet sondern die Bezeichnung. <code>[latex]</code> Latex Darstellung (Optional) <code>A:fix()</code>
<code>[tVar]:CRLF([string])</code>	Macht einen Zeilenumbruch an der Stelle in der Formel. <code>[string]</code> Optional. Zeichen wird am Ende der Zeile und am Anfang der neuen Zeile eingefügt. <code>C = A + B:CRLF("-") - A</code>
<code>[tVar]:outFull([bool])</code>	Ausgabe: Name=Gleichung=Gleichung mit Werten=Ergebnis. <code>[bool]</code> nummeriert für true. <code>U:outFull(true)</code>
<code>[tVar]:outHalf([bool])</code>	Ausgabe: Name=Gleichung=Ergebnis. <code>[bool]</code> nummeriert für true. <code>U:outHalf(true)</code>
<code>[tVar]:outVar([bool],[bool])</code>	Ausgabe: Name=Ergebnis. <code>[bool](1)</code> nummeriert für true. <code>[bool](2)</code> ohne environment für false. <code>U:outVar(false,true)</code>
<code>[tVar]:out()</code>	Ausgabe: Ergebnis (Inline). <code>U:out()</code>

tMat	
<code>tMat:New([value],[latex])</code>	Erzeugt eine neue Matrix-Variable mit dem Wert <code>[value]</code> und der Latex Darstellung <code>[latex]</code> . <code>A_2=tMat:New({{10,2,5},{2,4,3},{7,4,3}},"a_{2}")</code>
<code>[tMat]:T()</code>	Matrix Transponieren <code>C=A:T()</code>
<code>[tMat]:Det()</code>	Matrix Determinante <code>C=A:Det()</code>
<code>[tMat]:Inv()</code>	Matrix Inverse <code>C=A:Inv()</code>
tVec	
<code>tVec:New([value],[latex])</code>	Erzeugt eine neue Vektor-Variable mit dem Wert <code>[value]</code> und der Latex Darstellung <code>[latex]</code> . <code>v_1=tVec:New({10,2,7},"v_{1}")</code>
<code>[tVec]:crossP([tVec])</code>	Kreuzprodukt zweier Vektoren <code>v_3=v_1:crossP(v_2)</code>