1 Library laden

Laden von `tVar` und Festlegen eines Zahlenformates

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

2 U-Wert Berechnung

Berechnung des U-Wertes eine 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 Therm R
U=R:bracR()^(-1)
-- Bei der Ausgabe wird zuerst der Name der Var. angezeigt. Festlegen auf U
U:setName("U") = "U"
U:setUnit("\\frac{m^2K}{W}")
```

Output mit `U:outFull()`:

$$U = \left(r_{se} + \frac{d_1}{\lambda_1} + \frac{d_2}{\lambda_2} + r_{si}\right)^{-1} = \left(0.040 + \frac{10.000}{0.035} + \frac{10.000}{0.500} + 0.130\right)^{-1} = 0.003 \ \frac{m^2K}{W} \tag{1}$$

Output mit `U:outHalf()`:

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

Output mit `U:outVar()`:

$$U = 0.003 \ \frac{m^2K}{W} \tag{3}$$

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()`:

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

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()`:

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

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()`:

$$F = \sigma_v \cdot r^2 \cdot \pi = 12.152 \cdot 10.000^2 \cdot \pi = 3817.710 \tag{6}$$

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 Klammergroeszen
C = A + B:CRLF() -A
C:setName("C")
```

Output mit `C:outFull()`:

$$\mathbf{C} = \mathbf{a_1} + \mathbf{b_1} - \mathbf{a_1} = \begin{pmatrix} 10.000 & 2.000 & 5.000 & 3.000 \\ 2.000 & 4.000 & 3.000 & 1.000 \end{pmatrix} + \begin{pmatrix} 3.000 & 1.000 & 4.000 & 1.000 \\ 2.000 & 4.000 & 5.000 & 10.000 \end{pmatrix} - \begin{pmatrix} 10.000 & 2.000 & 5.000 & 3.000 \\ 2.000 & 4.000 & 3.000 & 1.000 \end{pmatrix} = \begin{pmatrix} 3.000 & 1.000 & 4.000 & 1.000 \\ 2.000 & 4.000 & 5.000 & 10.000 \end{pmatrix} \tag{7}$$

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()`:

$$\mathbf{C} = -\left(\mathbf{a_1} \cdot \mathbf{b_1} \cdot 2.000 - \frac{\mathbf{D}}{2.000}\right) = -\left(\begin{pmatrix} 10.000 & 2.000 & 5.000 & 3.000 \\ 2.000 & 4.000 & 3.000 & 1.000 \end{pmatrix} \cdot \begin{pmatrix} 3.000 & 1.000 \\ 4.000 & 2.000 \\ 5.000 & 8.000 \\ 2.000 & 9.000 \end{pmatrix} \cdot 2.000 - \frac{\begin{pmatrix} 5.000 & 12.200 \\ 3.000 & 22.000 \end{pmatrix}}{2.000}\right) = \begin{pmatrix} -135.500 & -155.900 \\ -76.500 & -75.000 \end{pmatrix} \tag{8}$$

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()`:

$$\mathbf{C} = \mathbf{a_1}^\top = \begin{pmatrix} 10.000 & 2.000 & 5.000 & 3.000 \\ 2.000 & 4.000 & 3.000 & 1.000 \end{pmatrix}^\top = \begin{pmatrix} 10.000 & 2.000 \\ 2.000 & 4.000 \\ 5.000 & 3.000 \\ 3.000 & 1.000 \end{pmatrix} \tag{9}$$

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()`:

$$C = |\mathbf{a_2}| = \left| \begin{pmatrix} 10.000 & 2.000 & 5.000 \\ 2.000 & 4.000 & 3.000 \\ 7.000 & 4.000 & 3.000 \end{pmatrix} \right| = -70.000 \tag{10}$$

Output mit `Inv:outFull()`:

$$\mathbf{Inv} = \mathbf{a_2}^{-1} \cdot \mathbf{a_2} = \begin{pmatrix} 10.000 & 2.000 & 5.000 \\ 2.000 & 4.000 & 3.000 \\ 7.000 & 4.000 & 3.000 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 10.000 & 2.000 & 5.000 \\ 2.000 & 4.000 & 3.000 \\ 7.000 & 4.000 & 3.000 \end{pmatrix} =$$
$$= \begin{pmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ -0.000 & -0.000 & 1.000 \end{pmatrix} \tag{11}$$

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()`:

$$\vec{v_3} = (\vec{v_1} + \vec{v_2}) - \vec{v_2} = \left(\begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix} + \begin{pmatrix} 3.000 \\ 1.000 \\ 2.000 \end{pmatrix} \right) - \begin{pmatrix} 3.000 \\ 1.000 \\ 2.000 \end{pmatrix} = \begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix} \tag{12}$$

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()`:

$$v_3 = (\vec{v_1} \cdot 2.000) \cdot \vec{v_2} = \left(\begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix} \cdot 2.000 \right) \cdot \begin{pmatrix} 3.000 \\ 1.000 \\ 2.000 \end{pmatrix} = 92.000 \tag{13}$$

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()`:

$$\vec{v_3} = \vec{v_1} \times \vec{v_2} = \begin{pmatrix} 10.000 \\ 2.000 \\ 7.000 \end{pmatrix} \times \begin{pmatrix} 3.000 \\ 1.000 \\ 2.000 \end{pmatrix} = \begin{pmatrix} -3.000 \\ 1.000 \\ 4.000 \end{pmatrix} \tag{14}$$

6 Globale Parameter

Folgende globalen Parameter können über:

```
tVar.[befehl] = ""
```

oder lokal für eine Variable z.B.:

```
U.[befehl] = ""
```

festgelegt werden. Mögliche Befehle mit default Werten sind:

```
numFormat = "%.3f"
mathEnviroment = "align"
debugMode = "off"
outputMode = "RES" --oder RES, RES_EQ, RES_EQ_N
numeration = true
```

6.1 Beispiel 1

Globale Definition:

```
tVar.outputMode = "RES"
tVar.numeration = false
```

Output mit `U:print()`:

$$U = 0.003 \ \frac{m^2 K}{W}$$

6.2 Beispiel 2

Globale Definition:

```
tVar.outputMode = "RES_EQ"
tVar.numeration = true
```

Output mit `U:print()`:

$$U = \left(r_{se} + \frac{d_1}{\lambda_1} + \frac{d_2}{\lambda_2} + r_{si} \right)^{-1} = 0.003 \ \frac{m^2 K}{W} \tag{15}$$

7 Befehlsreferenz

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

tVar	
`tVar.numFormat = [format]`	Definiert das Ausgabe-Zahlenformat `[latex]`. `tVar.numFormat = "%.3f"`
`tVar:New([value],[latex])`	Erzeugt eine neue Variable mit dem Wert `[value]` und der Latex Darstellung `[latex]`. `rse= tVar:New(0.04,"r_{se}")`
`[tVar]:setName([latex])`	Legt die Latex Darstellung für einen berechneten Wert fest. `U:setName("U")`
`[tVar]:setUnit([latex])`	Legt die Latex Darstellung für die Einheit fest. `U:setUnit("mm")`
`[tVar]:bracR()`	Umschließt die Formeldarstellung der Variable [tVar] mit runden Klammer. `U = R:bracR()`
`tVar.sqrt([tVar],[number])`	[numbert]te Wurzel aus [tVar] `sigma_v = tVar.sqrt(sigma_x^2+3*tau_xy^2,2)`
`[tVar].debugMode = "on"||"off"`	Ausgabe der generierten LaTeX-Kommandos Global: `tVar.debugMode = "on"`
`[tVar].mathEnviroment = [string]`	Festlegen der Math-Umgebung. Default: align. Wenn dann keine Umgebung.
`tVar.PI`	π `A=r^2*tVar.PI`
`[tVar]:fix([latex])`	Bereinigt die Variable von Berechnungsschritten, damit wird bei einer Rechenoperation nicht mehr die Gleichung verwendet sondern die Bezeichnung. [latex] Latex Darstellung (Optional) `A:fix()`
`[tVar]:CRLF([string])`	Macht einen Zeilenumbruch an der Stelle in der Formel. [string] Optional. Zeichen wird am Ende der Zeile und am Anfang der neune Zeile eingefügt. `C = A + B:CRLF("-")- A`

`[tVar]:outFull([bool],[bool])`	Ausgabe: Name=Gleichung=Gleichung mit Werten=Ergebnis. [bool] nummeriert für true. [bool](2) ohne enviroment für false. `U:outFull(true)`
`[tVar]:outHalf([bool],[bool])`	Ausgabe: Name=Gleichung=Ergebnis. [bool] nummeriert für true. [bool](2) ohne enviroment für false. `U:outHalf(true)`
`[tVar]:outVar([bool],[bool])`	Ausgabe: Name=Ergebnis. [bool](1) nummeriert für true. [bool](2) ohne enviroment für false. `U:outVar(false,true)`
`[tVar]:out()`	Ausgabe: Ergebnis (Inline). `U:out()`
tMat	
`tMat:New([value],[latex])`	Erzeugt eine neue Matrix-Variable mit dem Wert `[value]` und der Latex Darstellung `[latex]`. `A_2=tMat:New({{10,2,5},{2,4,3},{7,4,3}},"a_{2}")`
`[tMat]:T()`	Matrix Transponieren `C=A:T()`
`[tMat]:Det()`	Matrix Determinante `C=A:Det()`
`[tMat]:Inv()`	Matrix Inverse `C=A:Inv()`
tVec	
`tVec:New([value],[latex])`	Erzeugt eine neue Vektor-Variable mit dem Wert `[value]` und der Latex Darstellung `[latex]`. `v_1=tVec:New({10,2,7},"v_{1}")`
`[tVec]:crossP([tVec])`	Kreuzprodukt zweier Vektoren `v_3=v_1:crossP(v_2)`