Vorbemerkung:

Dieses *Mathematica*- Notebook behandelt nicht die im CA-Rundbrief diskutierte Erwärmung einer Zugbremsscheibe, sondern ein verwandtes Problem, nämlich die Wärmeleitung in einem (dünnen) Stab. Obwohl dieses (Lehrbuch-) Beispiel weniger komplex ist, zeigt es bereits alle Eigenschaften des vollen Problems. Der unten dargestellte Fall (feste Vorgabe der Temperatur an den Stab-Enden) kann auf kompliziertere Randbedingungen erweitert werden. Die Darstellung erfolgt in der Standard -*Mathematica*- Notation (nur diese ist programmiertechnisch eindeutig). Alle Erläuterungen sind als Kommentare eingefügt.

Die Umsetzung in die PDF-Datei erfolgte aus dem *Mathematica*-Menu. Leider ist die Formatierung an einigen Stellen suboptimal.

D. Hackenbracht

Viele Vorgänge werden durch partielle Differentialgleichungen (PDE) beschrieben. Die gesuchte Funktion hängt dabei von mehreren unabhängigen Variablen ab. Die wichtigsten linearen Vertreter dieser PDE sind:

- die Wärmeleitgleichung (allgemeiner: die Diffusionsgleichung)
- die Wellengleichung
- die Potentialgleichung (auch Laplace-Gl. genannt) .

Neben Anfangsbedingungen (Vorgabe der Funktion zur Zeit $t = 0$) müssen auch Randbedingungen gestellt werden (Vorgabe der Funktion oder ihrer räumlichen Ableitung am Rande des untersuchten Gebietes). Die exakte Behandlung (im Sinne einer einfachen Lösungsformel) ist meistens nicht möglich. Oft benötigt man (unendliche) Reihen, um die Lösung darzustellen. Aber auch dies geht nur in relativ einfachen Fällen.

Im Folgenden wird die Wärmeleitung in einem Stab untersucht. Der Einfachheit halber gehen wir davon aus, dass die Temperatur nur in der Stab- Längsrichtung (die hier die z-Richtung sein soll) variiert. Die Temperatur T ist dann eine Funktion von t und z . Man spricht dann auch von der [räumlich] eindimensionalen Wärmeleitgleichung. Die Herleitung der Wärmeleitgleichung findet man in einem Physikbuch, z.B. Alonso/Finn, Fundamental University Physics, Vol. 2 . Außerdem soll der Stab - außer an seinen Enden - thermisch isoliert sein.

A) Die Vorbereitung

In[1]:=
```
Remove["Global`*"]
```

Remove::rmnsm : There are no symbols matching "Global`*". ≫

In[2]:=
```
(* Die Materialdaten sind die für
 reines Kupfer  [Quelle: Deutsches Kupferinstitut] ;
   eigentlich hängen diese Daten auch von der Temperatur ab,
   aber dass soll im Folgenden vernachlässigt werden;

   λ ist die Wärmeleitfähigkeit in W/( K m) , cp die spezifische Wärme in J /(kg K) ,
   ρ die Dichte in kg/m^3  *)
```

In[3]:=
```
Materialdaten = {λ → 395 , cp → 380 , ρ → 8960 }
```

Out[3]=
```
{λ → 395, cp → 380, ρ → 8960}
```

In[4]:=
```
(* die relevanten geometrischen Daten,
d.h. hier linker und rechter Rand des Stabes, Einheit Meter *)
```

In[5]:=
```
z0 = 0;
```

In[6]:=
```
zn = 1;
```

In[7]:=

In[8]:=
```
(* die Bestandteile der Wärmeleitgleichung : *)
```

In[9]:=
```
(* die partielle Ableitung der Temperatur T  nach der Zeit  t : *)
```

In[10]:=
```
Tpunkt[z_, t_] = D[T[z, t], t]
```

Out[10]=
$$T^{(0,1)}[z, t]$$

In[11]:=
```
(* die 2. partielle Ableitung der Temperatur T  nach der Raumkoordinate z :  *)
```

In[12]:=
```
Tablzz[z_, t_] = D[T[z, t], {z, 2}]
```

Out[12]=
$$T^{(2,0)}[z, t]$$

In[13]:=
```
(* die Wärmeleitgleichung... *)
```

In[14]:=
```
dgl = Tpunkt[z, t] == λ Tablzz[z, t] / (cp ρ)
```

Out[14]=
$$T^{(0,1)}[z, t] == \frac{\lambda\, T^{(2,0)}[z, t]}{cp\, \rho}$$

In[15]:=
```
(* ...mit den Materialdaten: *)
```

In[16]:=
```
dgl = dgl /. Materialdaten
```

Out[16]=
$$T^{(0,1)}[z, t] == \frac{79\, T^{(2,0)}[z, t]}{680\,960}$$

In[17]:=

```
(* die zusätzlichen Bedingungen: *)

(* Anfangstemperatur in °C *)
```

In[18]:=
```
T0 = 20;
```

In[19]:=
```
(* 1.
   als Ausgangslage/Anfangsbedingung zur Zeit t=0
  nehmen wir eine homogene Temperaturverteilung im Stab an: *)
```

In[20]:=
```
anfangsbedingung = T[z, 0] == T0;
```

In[21]:=
```
(* 2a.
   am linken Rand wird die Temperatur zur Zeit t=0
  plötzlich auf 100 °C angehoben und auf diesem Wert festgehalten: *)
```

In[22]:=
```
randbedingungLinks = T[0, t] == 100;
```

In[23]:=
```
(* 2b.
   am rechten Rand wird die Temperatur auf dem Wert T0 festgehalten:*)
```

In[24]:=
```
randbedingungRechts = T[zn, t] == T0;
```

In[25]:=

```
(* alle Gleichungen zusammen: *)
```

In[26]:=
```
gl = {dgl, anfangsbedingung, randbedingungLinks , randbedingungRechts}
```

Out[26]=
$$\left\{T^{(0,1)}[z, t] == \frac{79\, T^{(2,0)}[z, t]}{680\,960}, T[z, 0] == 20, T[0, t] == 100, T[1, t] == 20\right\}$$

B) Die Berechnung der Lösung

In[27]:=

```
(* alle Gl. werden dem Mathematica- Lösungsbefehl übergeben;
   DSolve löst Differentialgleichungen exakt! *)
```

In[28]:=
```
ls = DSolve[ gl, T[z, t], {z, t}]
```

Out[28]=
$$\text{DSolve}\left[\left\{T^{(0,1)}[z, t] == \frac{79\, T^{(2,0)}[z, t]}{680\,960}, T[z, 0] == 20, T[0, t] == 100, T[1, t] == 20\right\}, T[z, t], \{z, t\}\right]$$

In[29]:=
```
(* aus den o.g. Gründen kann Mathematica dieses Problen nicht exakt lösen;
   also ist eine numerische Behandlung erforderlich;
   NDSolve ermittelt numerisch eine approxiamtive Lösung der Differentialgleichung *)
```

In[30]:=
```
(* Anfangs- und Endezeit des Simulationsintervalls, in Sekunden: *)
```

In[31]:=
```
t0 = 0;
```

In[32]:=
```
tn = 1000;
```

In[33]:=
```
ls = NDSolve[ gl, T[z, t], {z, z0, zn}, {t, t0, tn}]
```

```
NDSolve::ibcinc : Warning: Boundary and initial conditions are inconsistent. >>
```

Out[33]=
```
{{T[z, t] → InterpolatingFunction[{{0., 1.}, {0., 1000.}}, <>][z, t]}}
```

In[34]:=
```
(* die Aufbereitung des Ergebnisses : *)
```

In[35]:=
```
lsf = ls // Flatten;
```

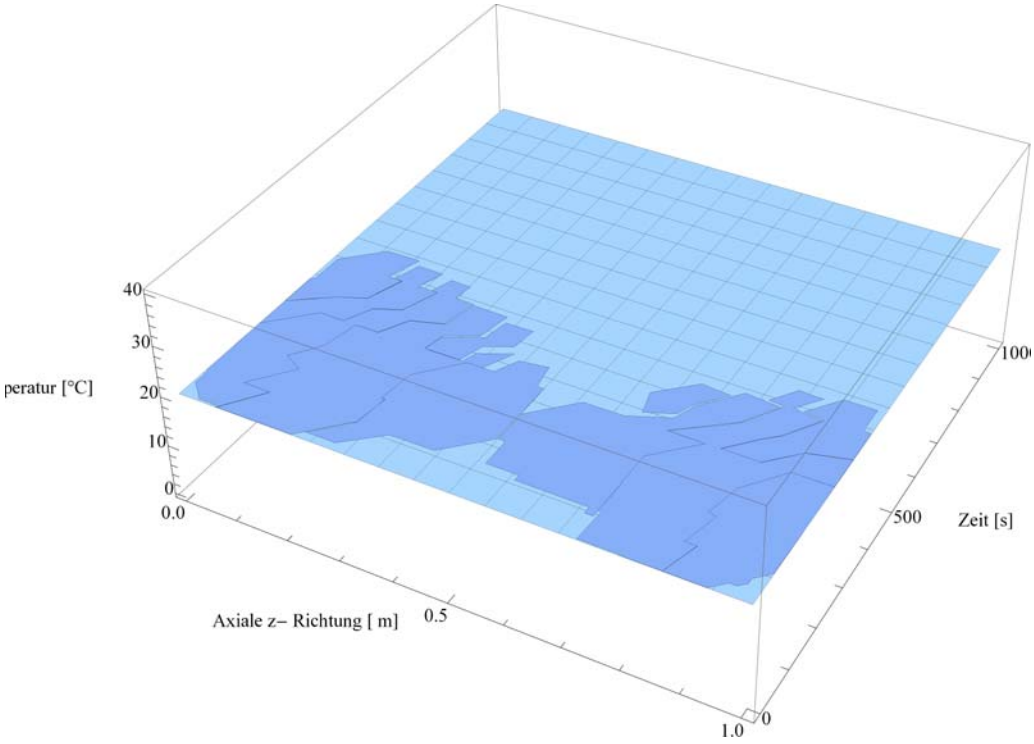
In[36]:=
```
TL[z_, t_] = T[z, t] /. lsf
```

Out[36]=
```
InterpolatingFunction[{{0., 1.}, {0., 1000.}}, <>][z, t]
```

In[37]:=
```
(* die Gesamtsicht: *)
```

In[38]:=
```
Plot3D[TL[z, t], {z, z0, zn}, {t, t0, tn},
 AxesLabel → {"Axiale z- Richtung [ m]\t", "\tZeit [s]", "Temperatur [°C]\t"}]
```

Out[38]=



In[39]:=

```
(* Es ist gar nichts passiert !!

Die obige Warnung muss man also sehr ernst nehmen;
" Boundary and initial conditions are inconsistent " trifft ja auch zu:
  über die Anfangsbedingung geben wir am linken Rand T[0,0] == T0 vor,
  über die Randbedingung aber T[0,0]== 100 !
  Dieser plötzliche Anstieg der Temperatur ist ja auch unphysikalisch. Man muss
  ihn also anders modellieren, also die Temperatur allmählich hochfahren *)
```

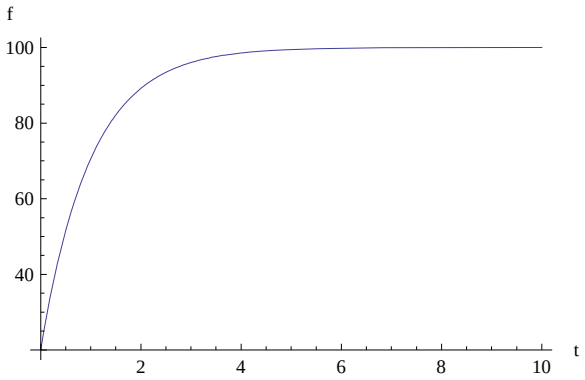
C) Korrektur des Modells und erneute Berechnung der Lösung

In[40]:=
```
(* also:
   am linken Rand wird die Temperatur zur Zeit t=0
  allmählich auf 100 °C angehoben und auf diesem Wert festgehalten;
   dies geschieht hier mit einer geeigneten "Anschalt"-Funktion f(t) : *)
```

In[41]:=
```
f[t_] := T0 + 80 * (1 - Exp[-t])
```

In[42]:=
```
Plot[f[t], {t, t0, t0 + 10}, PlotRange → All, AxesLabel → {" t \t", " f \t"}]
```

Out[42]=



In[43]:=
```
(* die e-Funktion klingt so schnell ab,
   dass die Temperatur nach ca 5s den gewünschten konstanten Wert 100°C hat;

   korrigierte Randbedingung: *)
```

In[44]:=
```
randbedingungLinks = T[0, t] == f[t]
```

Out[44]=
$$T[0, t] == 20 + 80\left(1 - e^{-t}\right)$$

In[45]:=
```
(* neue Rechnung: *)
```

In[46]:=
```
Clear[T]
```

In[47]:=
```
gl = {dgl, anfangsbedingung, randbedingungLinks , randbedingungRechts}
```

Out[47]=
$$\left\{T^{(0,1)}[z, t] == \frac{79\, T^{(2,0)}[z, t]}{680\,960}, T[z, 0] == 20, T[0, t] == 20 + 80\left(1 - e^{-t}\right), T[1, t] == 20\right\}$$

In[48]:=
```
ls = NDSolve[ gl, T[z, t], {z, z0, zn}, {t, t0, tn}]
```

Out[48]=
```
{{T[z, t] → InterpolatingFunction[{{0., 1.}, {0., 1000.}}, <>][z, t]}}
```

In[49]:=
```
(* die Aufbereitung des Ergebnisses : *)
```

In[50]:=
```
lsf = ls // Flatten;
```

In[51]:=
```
TL[z_, t_] = T[z, t] /. lsf
```

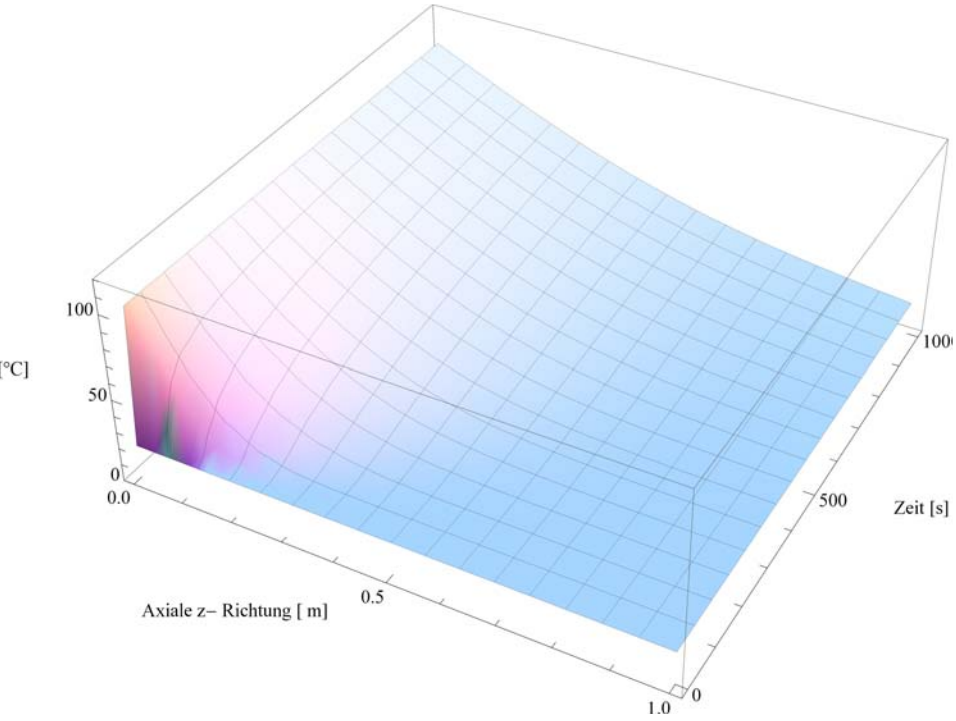
Out[51]=
```
InterpolatingFunction[{{0., 1.}, {0., 1000.}}, <>][z, t]
```

D) Graphische Darstellung der Lösung

In[52]:=
```
(* die Gesamtsicht: *)
```

In[53]:=
```
Plot3D[TL[z, t], {z, z0, zn}, {t, t0, tn}, PlotRange → {0, 120},
 AxesLabel → {"Axiale z- Richtung [ m]\t", "\tZeit [s]", "Temperatur [°C]\t"}]
```

Out[53]=



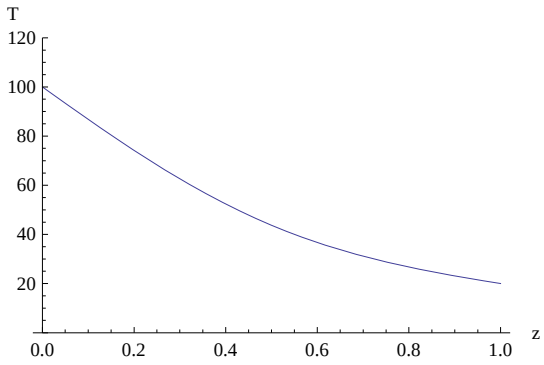
In[54]:=

```
(* Momentaufnahme: das Temperaturprofil längs des Stabes am Ende der Simulation: *)
```

In[55]:=
```
Plot[TL[z, tn], {z, z0, zn}, PlotRange → {0, 120}, AxesLabel → {" z \t", " T \t"},
 PlotLabel → "räumlicher Temperaturverlauf längs des Stabes am Ende der Simulation"]
```

ılicher Temperaturverlauf längs des Stabes am Ende der Simul

Out[55]=



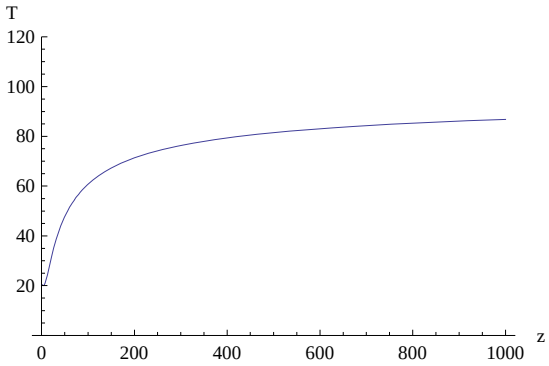
In[56]:=

```
(* der zeitliche Temperaturverlauf an einer festen Stelle, hier z= 0.1: *)
```

In[57]:=
```
Plot[TL[0.1, t], {t, t0, tn}, PlotRange → {0, 120}, AxesLabel → {" z \t", " T \t"},
 PlotLabel → "zeitlicher Temperaturverlauf in der Nähe des linken Randes"]
```

zeitlicher Temperaturverlauf in der Nähe des linken Randes

Out[57]=

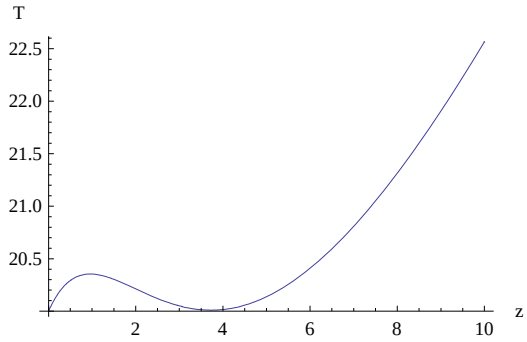


In[58]:=
```
(* daraus ein Detail: *)
```

In[59]:=
```
Plot[TL[0.1, t], {t, t0, tn / 100}, AxesLabel → {" z \t", " T \t"},
 PlotLabel → "anfänglicher Temperaturverlauf in der Nähe des linken Randes"]
```

anfänglicher Temperaturverlauf in der Nähe des linken Randes

Out[59]=



Die unschöne "Delle" im Temperaturverlauf ist, wie man so sagt, ein "numerisches Artefakt" ;
die numerische Stabilität der Lösung muss - in jedem Fall - durch eine genauere Rechnung geprüft werden

E) Kontrolle: eine genauere Rechnung

In[61]:=
```
ls = NDSolve[ gl, T[z, t], {z, z0, zn}, {t, t0, tn},
              MaxStepSize → {zn / 100,  tn / 1000},
              MaxSteps → Infinity,  AccuracyGoal → 11]
```

Out[61]=
```
{{T[z, t] → InterpolatingFunction[{{0., 1.}, {0., 1000.}}, <>][z, t]}}
```

In[62]:=
```
lsf = ls // Flatten;
```

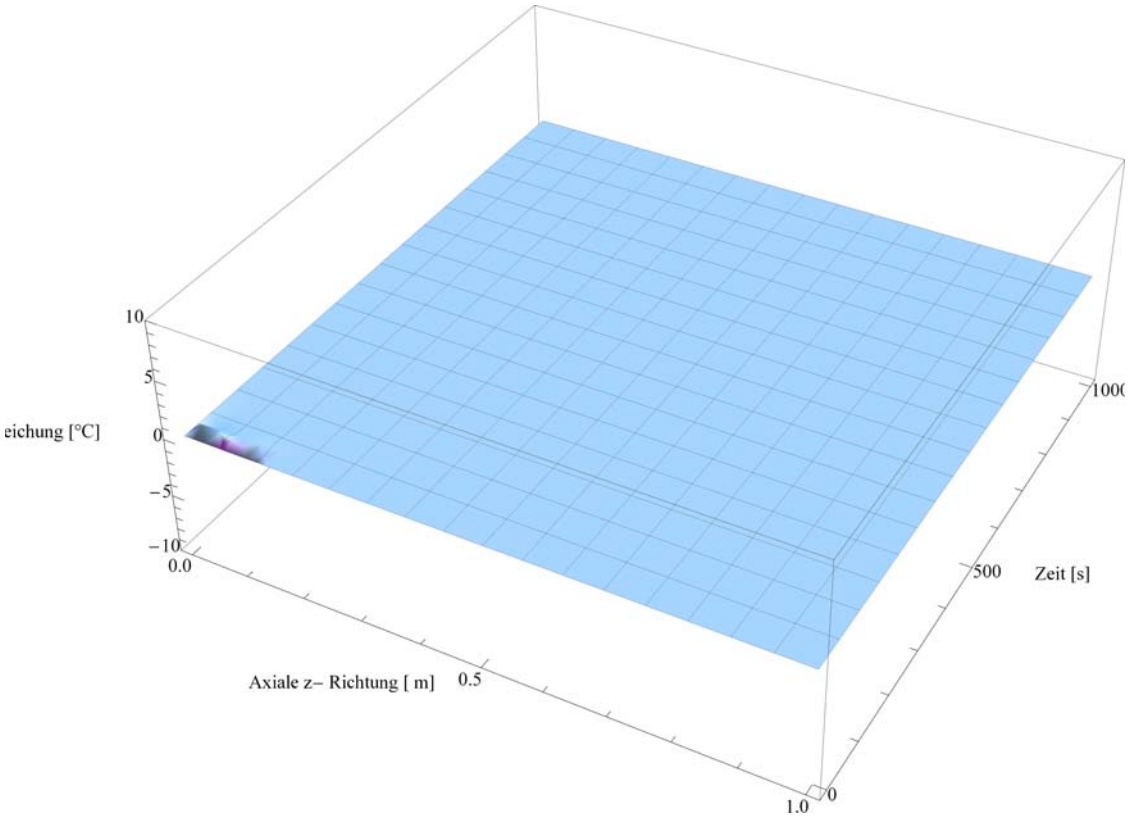
In[63]:=
```
TLneu[z_, t_] = T[z, t] /. lsf
```

Out[63]=
```
InterpolatingFunction[{{0., 1.}, {0., 1000.}}, <>][z, t]
```

In[64]:=
```
(* der direkte Vergleich beider Lösungen: *)
```

In[65]:=
```
Plot3D[TLneu[z, t] - TL[z, t], {z, z0, zn}, {t, t0, tn}, PlotRange → {-10, 10},
 AxesLabel → {"Axiale z- Richtung [ m]\t", "\tZeit [s]", "Abweichung [°C]\t"}]
```

Out[65]=



Das sieht gut aus; insgesamt ist die Lösung also numerisch stabil

In[67]:=

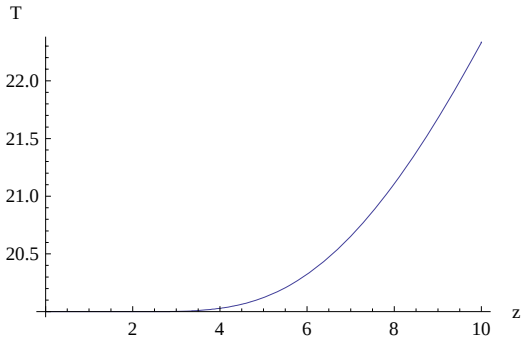
In[68]:=

```
(* das kritische Detail: *)
```

In[69]:=
```
Plot[TLneu[0.1, t], {t, t0, tn / 100}, AxesLabel → {" z \t", " T \t"},
 PlotLabel → "anfänglicher Temperaturverlauf in der Nähe des linken Randes"]
```

anfänglicher Temperaturverlauf in der Nähe des linken Randes

Out[69]=



Die Delle ist weg - oder ?

```
(* zur Kontrolle einige Zahlenwerte: *)
```

In[71]:=
```
Table[TLneu[0.1, t], {t, 0, 4, 0.1}]
```

```
{20.`, 20.000000000000966`, 20.000000000025647`, 19.999999999517808`, 19.99999999629978`,
 19.999999989834823`, 19.999999993613375`, 20.000000045193175`, 20.000000188381946`,
 20.000000456863607`, 20.000000844994958`, 20.000001206084377`, 20.00000116792606`,
 20.000000150884556`, 19.99999735913098`, 19.999992004817617`, 19.999983355475646`,
 19.999970974433463`, 19.999955138404804`, 19.999937193092485`, 19.999919837129394`,
 19.999908839314052`, 19.99991053646658`, 19.999933968371465`, 19.999993722600973`,
 20.000102933422667`, 20.000283349862844`, 20.00055473438746`, 20.000944419842586`,
 20.001481556486766`, 20.002198099940426`, 20.003134090854655`, 20.00432447360984`,
 20.005810760257454`, 20.007640509875024`, 20.0098566230452`, 20.012507133396706`,
 20.01564229416877`, 20.01930998803875`, 20.023559541074608`, 20.028440480534698`}
```

Dieser Verlauf ist akzeptabel, ansonsten muss man noch genauer rechnen.

In[73]:=

In[74]:=
```
(* direkter Vergleich beider Lösungen: *)
```

In[75]:=
```
Plot[TLneu[0.1, t] - TL[0.1, t], {t, t0, tn / 100}, AxesLabel → {" z \t", " Abweichung \t"}]
```

Abweichung

Out[75]=