1 Lineare Bewegung der Körper

1.2 Lineare Bewegung mit Reibung

1.2.1 Rückblick

Sie haben die Begriffe *momentane Geschwindigkeit* (oder einfach **Geschwindigkeit**) und *momentane Beschleunigung* kennen gelernt. Die Geschwindigkeit ist die erste Ableitung der Weg-Zeit-funktion nach der Zeit, $v(t) = \frac{dx(t)}{dt} = x'(t)$, d.h. v gibt in jedem Augenblick die Steigung der Weg-Zeit-Kurve an.

Die *skalare* Geschwindigkeit (speed), die ein Tachometer anzeigt, ist gleich dem Betrag der Geschwindigkeit und hat die Einheit $\frac{km}{h} = .2777... \frac{m}{s}$. (Die Ausdrucksweise "Stundenkilometer" ist falsch, denn sie bedeutet $km \cdot h$.)

Die **Beschleunigung** (momentane) ist die erste Ableitung der Geschwindigkeit nach der Zeit, $a(t) = \frac{dv(t)}{dt} = x''(t)$, sie hat also die Einheit $\frac{m}{s^2}$.

Während des freien Falls (oder beim senkrechten Wurf) ohne Luftreibung ist die konstante Gravitationskraft (Gewicht), die einzige Kraft, die auf das Objekt einwirkt. Seine Bewegungsgleichungen kann man folgendermaßen zusammenfassen:

$$\begin{aligned} a &= -g = -9.8 \tfrac{m}{s^2} \\ y(t) &= y_0 + v_0 t - gt^2/2 \\ v(t) &= v_0 - gt \\ v^2 + 2g(y - y_0) &= v_0^2 \end{aligned} \qquad \text{(Freier Fall)}$$

Beispiel:

Jemand wirft einen Tennisball senkrecht nach oben ($y_0 = 0$) mit $v_0 = 15m/s$.

1. Wie lange braucht der Ball, um seine maximale Höhe zu erreichen?

2. Wie groß ist die maximale Höhe?

3. Welche Zeit braucht der Ball, um einen Punkt in 6m Höhe zu passieren? (Es gibt zwei Zeiten, denn der Ball passiert die 6m-Marke einmal beim Steigen, dann nochmals beim Fallen.)

```
reset()://befreit alle Variablen von Wertzuweisungen
g:=9.8:
y0:=0:
v0:=16:
```

```
y:=y0+v0*t-g*t^2/2:
t1:=solve(diff(y,t)=0,t)//Steigzeit
1.632653061
subs(y,t=t1[1])//Steighoehe
13.06122449
t2:=solve(y=6,t)// Zeit, um 6m zu erreichen
{0.4322088841, 2.833097238}
t2_1:=t2[1]// Zeit, um zum ersten Mal die 6m-Marke zu erreichen
0.4322088841
subs(y,t=t2_1)//Test
6.0
t2_2:=t2[2]// Zeit, um die 6m-Marke zum zweiten Mal zu erreichen
2.833097238
subs(y,t=t2_2)//Test
6.0
```

1.2.2 Berücksichtigung der Luftreibung ($v'(t) = -r \cdot v(t)$)

Die Fälle mit konstanter Beschleunigung, die wir bisher untersuchten, sind mathematisch recht einfach zu behandeln. Reale Fall-oder Flugbewegungen finden jedoch immer in einem umgebenden Medium statt, meist Luft oder Wasser. In diesem Fall erfährt der Körper zusätzlich zur Gewichtskraft eine Reibungskraft (drag). Man nimmt im einfachsten Fall an, dass die vom Medium erzeugte Beschleunigung negativ und proportional zu v ist. Wir nehmen zunächst an, dass es sich um eine *horizontale* Bewegung handelt, bei der wir die Gewichtskraft außer Acht lassen können.

Wir gehen also aus von dem Ansatz

$$a(t) = v'(t) = -rv(t) \tag{1.2-1}$$

Die Reibungskonstante r muss in jedem Fall experimentell bestimmt werden. (Für Kugeln und einfache Körper gibt es Tabellen.) Wir müssen eine Funktion suchen, die abgeleitet, sich selbst proportional ist. Vielleicht wissen Sie, dass die Exponentialfunktionen diese Eigenschaft haben. Wir notieren also

$$v(t) = v_0 e^{-rt} \tag{1.2-2}$$

Wir können dieses Resultat leicht mit MuPAD kontrollieren

```
reset():
vel:=t->v0*e^(-r*t):
diff(vel(t),t)
-1/e^(r*t)*r*v0*ln(e)
```

Da $\ln(e) = 1$, gilt $-rv_0e^{-rt} = -rv(t)$. Also genau das, was wir erwarteten.

Mathematisch ist 1.2-1 eine ODE (**O**rdinary **D**ifferential **E**quation) mit der *Anfangsbedingung* $v_0 = v(0)$. Wir wollen dieses Anfangswertproblem *vel* nennen (velocity) und schreiben

```
vel:=ode({v'(t)=-r*v(t),v(0)=v0},v(t)):
velocity:= solve(vel)
```

`velocity` ist der Name, den wir der Lösung des Problems geben. MuPAD erzeugt $\{v_0 e^{-rt}\}$. Wir werden sehr viel mit Anfangs-und Randwertproblemen zu haben und dabei immer wieder die **ode**-Funktion einsetzen.

Wer mit dem "Scientific WorkPlace" (kurz **SWP**) arbeitet (wie ich es tue), hat es ebenfalls sehr bequem. Die Lösung unserer Differentialgleichung sieht dann so aus:

$$\begin{matrix} v' + rv = 0 \\ v(0) = v_0 \end{matrix} \text{ , Exact solution is: } \{v_0 e^{-rt}\}$$

In SWP macht in Wirklichkeit MuPAD die Mathematik. Bei SWP schreibt man Diff.-Gleichung und Bedingungen getrennt in die Felder einer einspaltigen Matrix und bringt den Kursor in die Gleichung. Anschließend geht man zu *Compute > Solve ODE > exact* (oder numerisch). Vgl. das Tutorial!

Um jetzt die Weg-Zeit-Gleichung zu erhalten, lösen wir die Differentialgleichung $x'(t) = v_0 e^{-rt}$ mit der Anfangsbedingung $x(0) = 0$.

$$\begin{matrix} x' - v_0 e^{-rt} = 0 \\ x(0) = 0 \end{matrix} \text{ , Exact solution is: } \left\{\tfrac{1}{r} v_0 - \tfrac{1}{r} v_0 e^{-rt}\right\}$$

Dieses Ergebnis vereinfachen wir mit dem Befehl *Compute > Factor* von SWP

$$\tfrac{1}{r} v_0 - \tfrac{1}{r} v_0 e^{-rt} = -v_0 \tfrac{e^{-rt}-1}{r}$$

Wir wollen dieses Ergebnis folgendermaßen ausdrücken

$$x = \frac{v_0}{r}(1 - e^{-rt}) \tag{1.2-3}$$

Die entsprechenden MuPAD-Anweisungen lauten:

```
pos:=ode({x'(t)=v0*exp(-r*t),x(0)=0},x(t)):
position:=solve(pos):
factor(%[1])
```

Der Term e^{-rt} ist 1 für $t = 0$ und nimmt schnell mit der Zeit ab. Wenn t gegen ∞ strebt, geht x gegen den Grenzwert $x_{\lim} = \frac{v_0}{r}$. Mit diesem Ausdruck kann man experimentell r betimmen, indem man zu einem gegebenen v_0 das zugehörige $x_{\lim}$ bestimmt. Unter *Zeitkonstante* τ versteht man den reziproken r-Wert, d.h. $\tau := \frac{1}{r}$. Die Zeit, nach der die Geschwindigkeit auf die Hälfte ihresAnfangswertes gesunken ist, heißt *Halbwertszeit* $t_{1/2}$. Durch Lösen der Gleichung $\frac{v_0}{2} = v_0 e^{-rt}$ nach t erhalten wir $t_{1/2} = \ln(2)/r$ Sekunden. Wenn wir die Gleichung mit SWP lösen, ergibt sich

assume (real) = $\mathbb{R}$ $\frac{v_0}{2} = v_0 e^{-rt}$,

$$\text{Solution is: } \begin{cases} \left\{ \frac{1}{r} \left(\ln 2 - 2i\pi X_{11} \right) \mid X_{11} \in \{0\} \right\} & \text{if} \quad v_0 \neq 0 \wedge r \neq 0 \\ \emptyset & \text{if} \quad v_0 \neq 0 \wedge r = 0 \\ \mathbb{R} & \text{if} \quad v_0 = 0 \end{cases}$$

also ein gewaltiges Resultat, das allerdings ebenfalls gleich ist $t_{1/2} = \ln(2)/r$.

MuPAD selbst ist nicht freundlicher mit seinem Resultat. Ausweg mit mehrfachem `assume`:

```
assume(t,Type::Real):
assume(v0>0):assume(r>0):
solve(v0/2=v0*exp(-r*t),t)

{1/r*ln(2)}
```

Beispiel: Wir wollen annehmen, dass wir in einem Motorboot auf einem See fahren. Wenn die Geschwindigkeit gerade $v = v_0 = 5m/s$ beträgt, bleibt der Motor stehen und das Boot gleitet mit $r = 0.1s^{-1}$ohne Antrieb weiter. Wie weit wird es noch gleiten?

```
v0:=5:r:=0.1:
DIGITS:=2:
v:=t->v0*exp(-r*t):
t_halb:=float(ln(2)/r):
print("Halbwertszeit =", t_halb, " Sekunden");
graf:=plot::Function2d(v(t),t=0..40):
plot(graf, GridVisible=TRUE, AxesTitles= ["Zeit/s",
"v(m/s)"],Header="Boot")

"Halbwertszeit =", 6.9, " Sekunden"
```

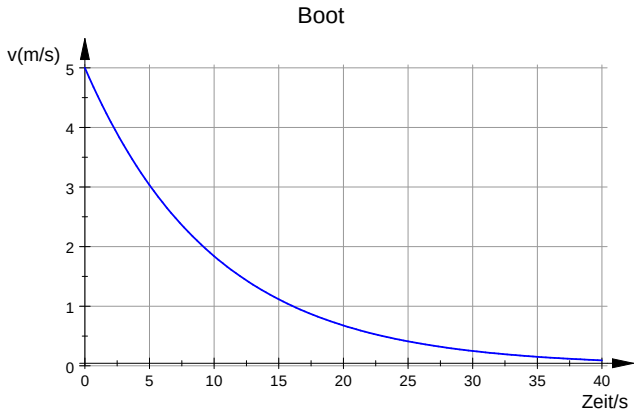


Fig. 1-1

An dieser Graphik sehen wir einige neue Techniken: Achsenbezeichnung, Überschrift und Gitternetz. (Um eine Graphik von MuPAD aus zu exportieren, speichert man sie zunächst mit *Export Graphics* (vorher ein Linksklick in die Graphik). Es erscheint ein "Guide", in dem man *Next* anklickt. Den Filetyp auswählen, z.B. **wmf** für ScientifikWorkPlace, und mit *Choose* Ort und Namen der Graphik angeben. Von SWP aus importiert man die Graphik dann mit *File > Import Picture.* Will man nur ein MuPAD-Ergebnis nach SWP kopieren, so benutzt man in MuPAD *Edit > Copy* und in SWP *Edit > paste.*

Für Neugierige erwähnen wir noch, dass man auch in früheren Zeiten in der Lage war, eine einfache Differentialgleichung wie $\frac{dv}{dt} = -rv(t)$ zu lösen (zu integrieren). Man schrieb die Gleichung mit *separierten* Variablen und integrierte beide Seiten: $\int \frac{1}{v} dv = \int k dt$. Ergebnis: $\ln v = kt + C$. Wenn wir dies nach v auflösen, erhalten wir $e^{\ln v} = e^{kt+C}$ oder $v = v(t) = e^C e^{kt} = v(0)e^{kt}$. Wir haben statt $-r$ einfach k geschrieben, wobei k positiv oder negativ sein kann. Bei $k > 0$ beschreibt die Lösung einen Wachstumsvorgang, bei $k < 0$ haben wir einen Zerfallsvorgang.

1.2.3 Weitere MuPAD-Beispiele

In seiner "graphics library" enthält MuPAD einen wahren Schatz an Graphikelementen. Mit `"?plot"` können Sie sich diese anschauen und ausprobieren. Ich will Ihnen in diesem Abschnitt nur einige der gebräuchlichsten Graphikformen zeigen und erklären.

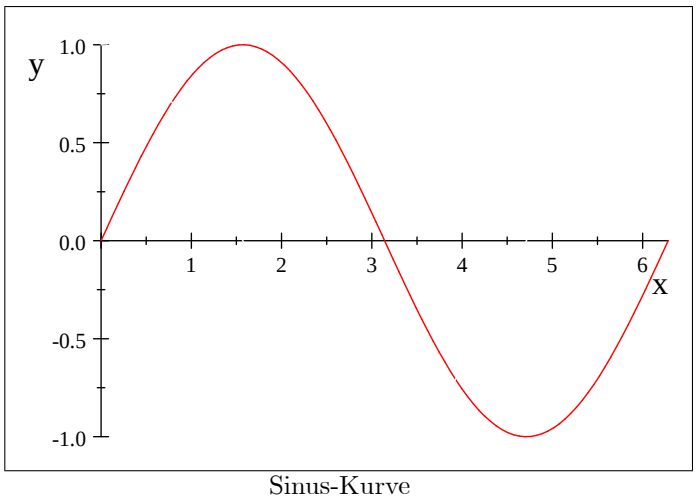
1.

Zunächst zeichnen wir die Funktion $f(x) = \sin(x)$ zwischen 0 und 2π rad (Radiant). 1 rad := $180^0/\pi$ In MuPAD zeichnen wir f mit der Anweisung

```
f:=plot::Function2d(sin(x), x=0..2*PI):

plot(f)
```

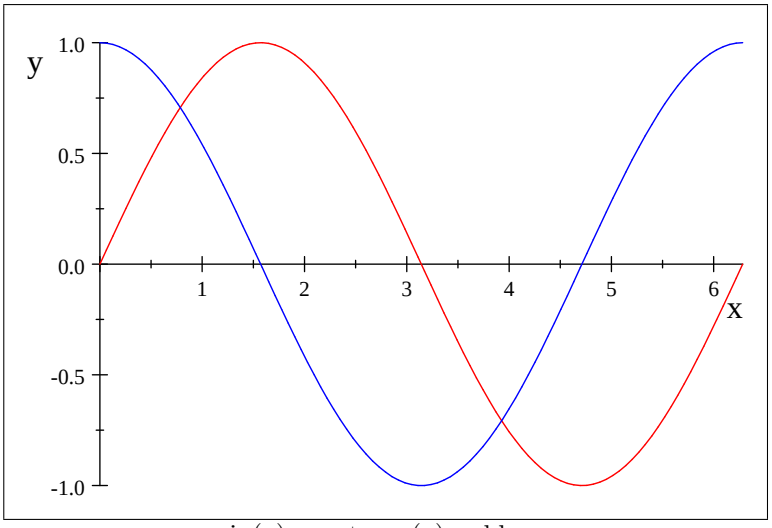
Um diesen Grafen in SWP zu zeichnen, schreiben wir einfach $\sin(x)$ *Compute > Plot 2D > Rectangular*



Sinus-Kurve

Fig.1-2

Jetzt den Kursor in den Graphen setzen und rechtsklicken, wo man dann mit *Plot Properties* Farbe (z.B. rot), Zeichenintervall usw. aussuchen kann. Für das Intervall nehmen wir 0 bis 6.283 (ca. 2π). Mit *Labeling* können wir dem Graphen auch noch einen Namen geben. Wollten wir zusätzlich $\cos(x)$ einzeich-nen, wir schreiben und markieren $\cos(x)$ und ziehen es einfach in den vorigen Graphen hinein:



sin(x) = rot; cos(x) = blau

Fig.1-3

Das funktioniert wie Hexerei. Man kann auf diese Weise noch Geraden, z. B. Tangenten, und Punkte usw. einzeichnen. Der MuPAD-Kode sieht etwas umständlicher aus:

```
f1:=plot::Function2d(sin(x), x=0..2*PI, Color=RGB::Red):
f2:=plot::Function2d(cos(x), x=0..2*PI, Color=RGB::Blue):
plot(f1,f2,Footer="sin(x)=rot, cos(x)=blau")
```

2.

Der Befehl `plotfunc2d` eignet sich besonders zum einfachen Zeichnen mehrerer Graphen:

```
plotfunc2d(cos(2*x^2), sin(4*x^2), sin(x)*cos(x), x=0..2)
```

Die Graphik wird von MuPAD vollkommen selbständig generiert:

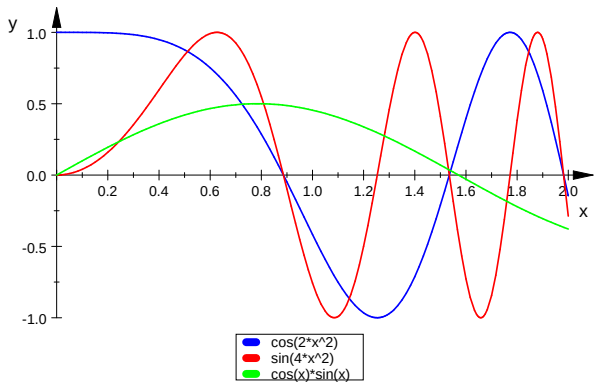


Fig.1-4

3.

Nun werden wir erneut das Boot-Beispiel betrachten und in den Graphen noch den Halbwertspunkt einzeichnen. Der MuPAD-Kode sieht folgendermaßen aus

```
v0:=5:r:=0.1:
DIGITS :=3:
thalb:=float(ln(2)/r):
v_0:=float(v0/2):
v:=t->v0*exp(-r*t):
p1:=float(v(thalb)):
f1:=plot::Function2d(v(t),t=0..40):
l1:=plot::Line2d([0,v_0],[thalb,v_0], Color=RGB::Blue):
l2:=plot::Line2d([thalb,0],[thalb,v_0], Color=RGB::Green):
```

```
p:=plot::Point2d([thalb,p1], Color = RGB::Blue):
info1:=plot::Text2d("Halbwertspunkt",[thalb,p1],
HorizontalAlignment=Left):
plot(f1,l1,l2,p,
AxesTitles= ["t/s","v/m/s"],Header="Geschw. des Bootes", info1)
```

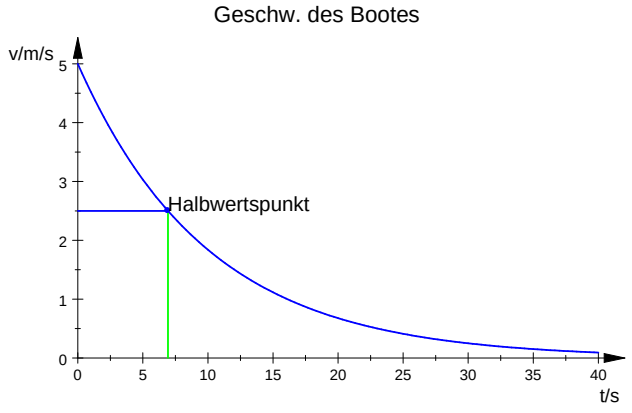


Fig.1-5

Wir haben hier einige neue Befehle benutzt: `plot::Line2d`, `plot::Point2d`, `plot::Text2d`. Nun kann man auch noch die **Tangente** einzeichnen. (Die Gleichung der Tangente an den Graphen der Funktion $y = f(x)$ im Punkt $(c, f(c))$ ist $t(x) = f'(c)(x - c) + f(c)$.) Im Programm von vorhin sind nur wenige Änderungen vorzunehmen:

```
v0:=5:r:=0.1:
DIGITS :=3:
thalb:=float(ln(2)/r):
v_0:=float(v0/2):
v:=t->v0*exp(-r*t):
  tang:=v'(thalb)*(t-thalb)+v(thalb):
  tend:=float(thalb-v(thalb)/v'(thalb)):
p1:=float(v(thalb)):
f1:=plot::Function2d(v(t),t=0..40):
  f2:=plot::Function2d(tang(t),t=0..tend,Color=RGB::Red):
l1:=plot::Line2d([0,v_0],[thalb,v_0], Color=RGB::Blue):
l2:=plot::Line2d([thalb,0],[thalb,v_0], Color=RGB::Green):
p:=plot::Point2d([thalb,p1],
  Color = RGB::Black,PointSize=3*unit::mm):
info1:=plot::Text2d("Halbwertspunkt",[thalb,p1],
HorizontalAlignment=Left):
  plot(f1,f2,l1,l2,p,
AxesTitles= ["t/s","v/m/s"],Header="Geschw. des Bootes", info1)
```

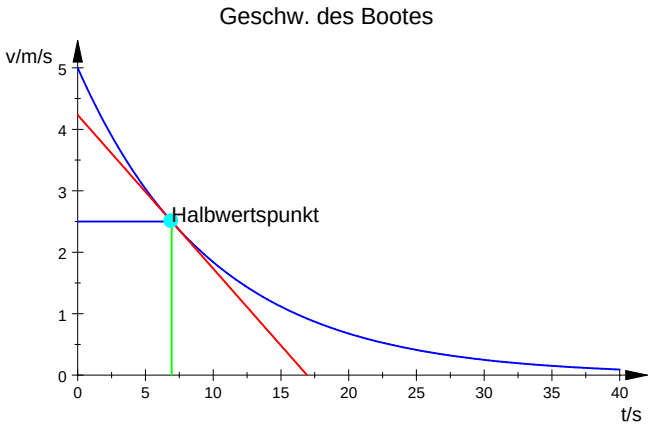


Fig.1-6

Die Punktgröße legen wir mit `PointSize=n*unit::mm` fest.

Zur **Übung** sollten Sie versuchen, den Graph der Funktion $f(x) = (x-2)^2/2+2$ zusammen mit der Tangente in $x = 3$ im Bereich $[0..6]$ zu zeichnen.

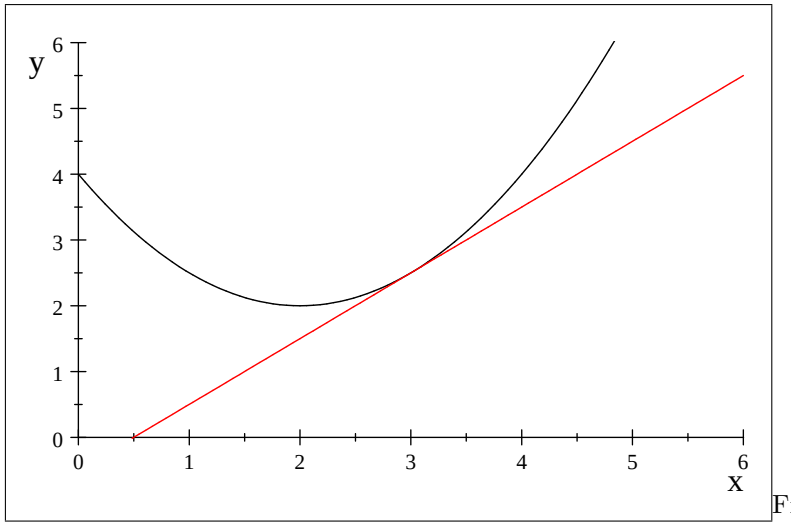


Fig.1-7

$1(x-3)+2.5$ (diese Gleichung in den Graphen zu "$(x-2)^2/2+2$" ziehen)

Wir haben die Zeichnung zunächst mit SWP erzeugt. Der Kode für MuPAD sieht so aus:

```
reset():
f:=x->(x-2)^2/2+2:
c:=3:
c1:=float(f(c)):/Dezimalzahl
```

```
t1:=f(c)+f'(c)*(x-c)://Tangente in (c;c1)
f1:=plot::Function2d(f(x),x=0..6):
g:=plot::Function2d(t1(x),x=0..6, Color=RGB::Red):
p:=plot::Point2d([c,c1], Color=RGB::Blue):
plot(f1,g,p,GridVisible=TRUE,TicksNumber=High)
```

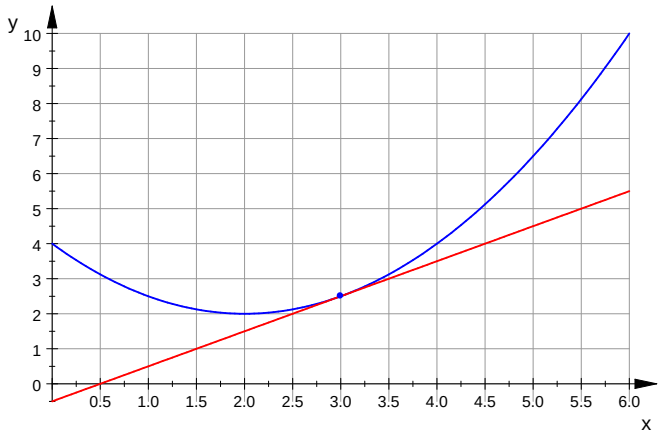


Fig.1-8

Beachten Sie den neuen Befehl `TicksNumber=High`.