# Differentialgleichungen mit Sage

## Definieren und Lösen Differentialgleichungen erster Ordnung

Sei die Differentialgleichung

y'(x)= 2*y(x)

Diese Differentialgleichung wird in Sage wie folgendes definiert:

In [1]:
```
x=var('x')
y=function('y')(x)
eqd=diff(y,x)==2*y
desolve(eqd,y)
```

Out[1]:
```
_C*e^(2*x)
```

```
In Sage benutzen wir den Befehl desolve, um eine Differentialgleichung zu
lösen.

desolve(equation, variable, ics = ..., ivar = ..., show_method = ...,
contrib_ode = ...)

wobei:

equation eine Differentialgleichung ist. Die Gleichheit wird durch das
Symbol == repräsentiert;

variable die abhängige Variable ist, d.h., y als y(x);

ics ist optional und es steht für Anfangswerte. Für eine
Differentialgleichung erster Ordnung, schreibt man [x0,y0] und für eine
Differentialgleichung zweiter Ordnung, schreibt man [x0,y0,x1,y1] oder
[x0,y0,y0'];

ivar ist optional und es steht fur die unabhängige Variable, d.h., x in
y(x). Es muss angegeben werden wenn es da mehr als eine unabhängige
Variable oder Parameter gemäß y gibt;

show_method is ein optional boolean auf falsch gesetzt. Wenn true ist,
Sage gibt ein Paar "[solution, method]" zurück, wobei method der String
ist, der die Methode beschreibt, die verwendet wurde, um eine Lösung zu
erhalten. Die Methode kann einen der folgenden: linear, separable, exact,
homogeneous, bernoulli, generalized homogeneous sein;

contrib_ode is ein optional boolean auf falsch gesetzt. Wenn true ist,
desolve lässt um Clairaut, Lagrange, Riccati und etwa andere Gleichungen
zu lösen. Dies kann eine lange Zeit dauern und ist somit standardmäßig
ausgeschaltet.
```

Wenn wir die verwendete Methode sehen möchten, die Sage benutzt hat um die Losung zu finden, können wir die Option show_method = True hinfügen. Die Antwort wird als Liste angegeben,

In [2]:
```
desolve(eqd,y,show_method=True)
```

Out[2]:
```
[_C*e^(2*x), 'linear']
```

## Darstellung der Lösung in expliziter Form

Es gibt zwei Wege um die Lösungen für eine gegebene Differentialgleichung darzustellen. Die erste ist um einen Wert für die Integrationskonstante im Lösungsausdruck zu ersetzen und danach mit dem plot-Befehl darstellen. Die zweite Methode ist um die Lösung als eine zwei-Variablen Funktion zu definieren, eins ist die Variable x und das zweite ist die Integrationskonstante; der Graph wird durch plot-Befehl für einen angegebenen Wert für die Integrationskonstante erhalten.

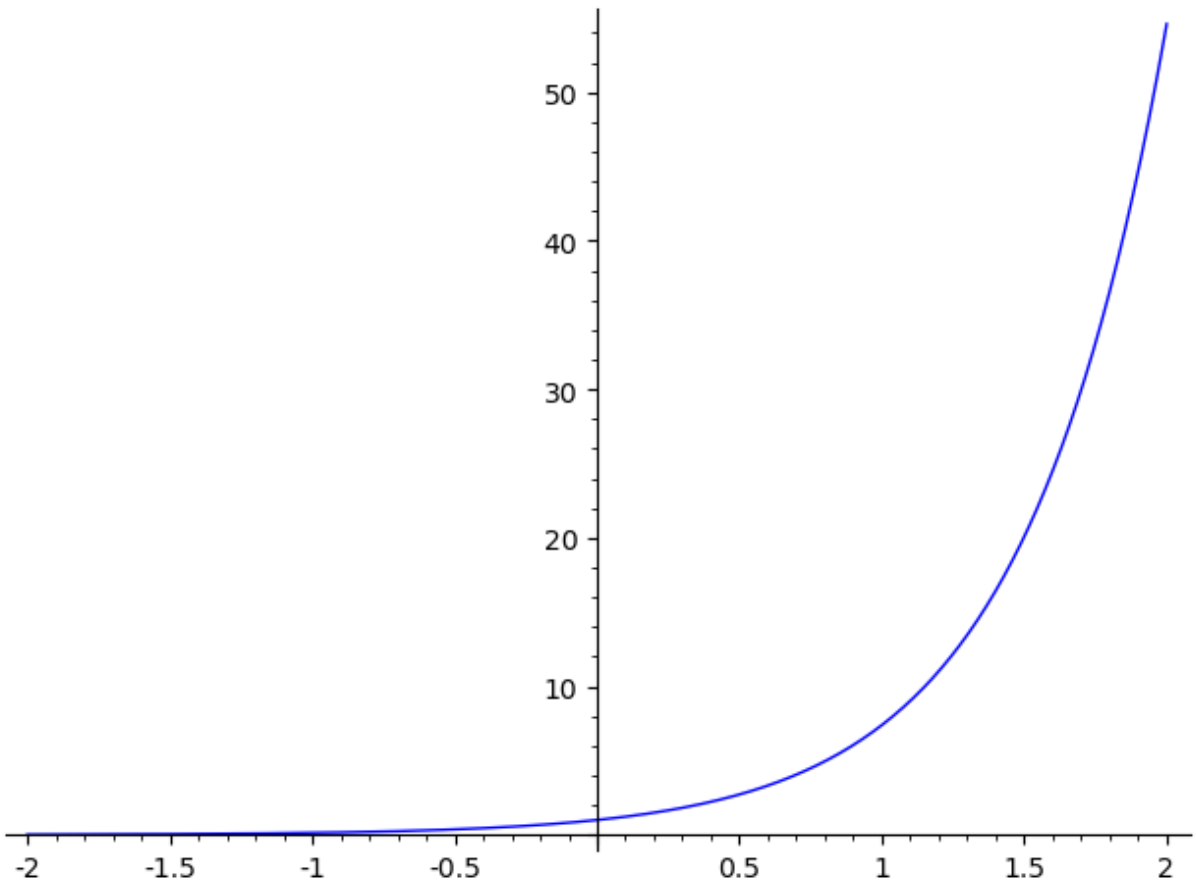
In [3]:
```
sol=desolve(eqd,y)
_C=var('_C')
sol.substitute(_C==1)
```

Out[3]:
```
e^(2*x)
```

In [4]:
```
sol1=sol.substitute(_C==1)
```

In [5]:
```
plot(sol1,x,-2,2)
```

Out[5]:

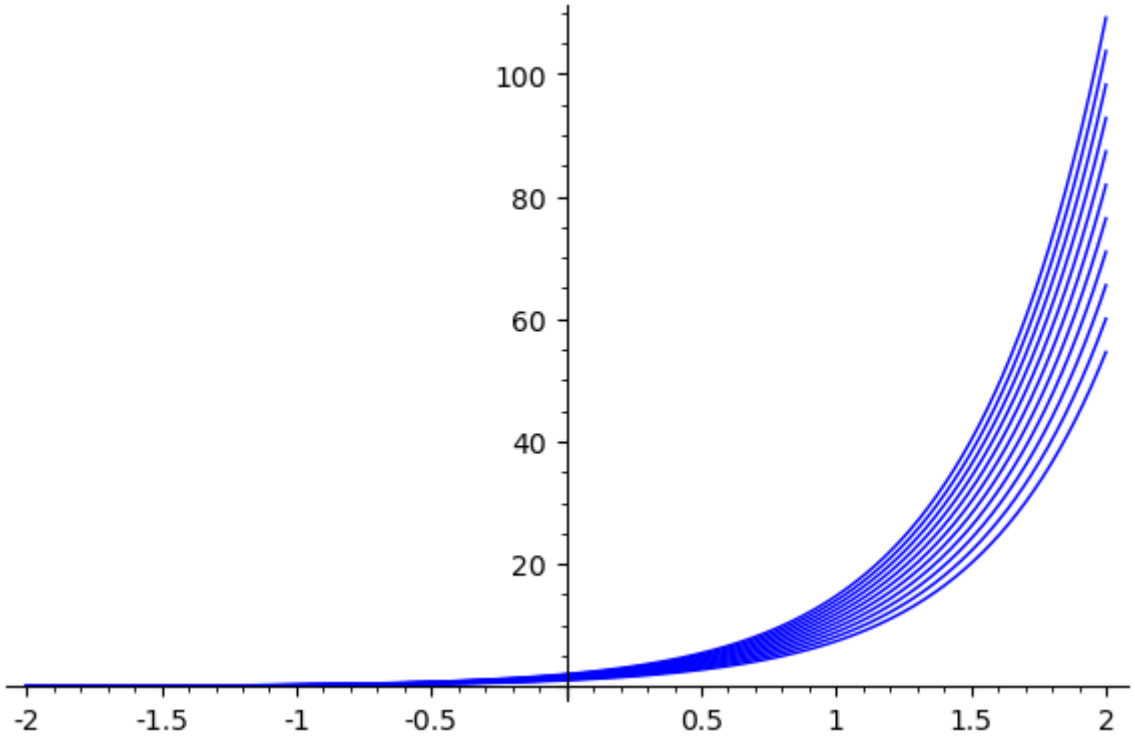


Wenn wir mehr als eine Lösung darstellen möchten, können wir wie folgendes machen: z.B., lassen Sie uns die Lösungen entwerfen, die den konstanten Werten

_C=1, 1.1, 1.2,..., 2

der Integration entsprechen.

In [6]:
```
sol=desolve(eqd,y)
sol1=sol.substitute(_C==1)
g=plot(sol1,x,-2,2)
for i in [11..20]:
    sol1=sol.substitute(_C==i/10)
    g=g+plot(sol1,x,-2,2)
show(g)
```



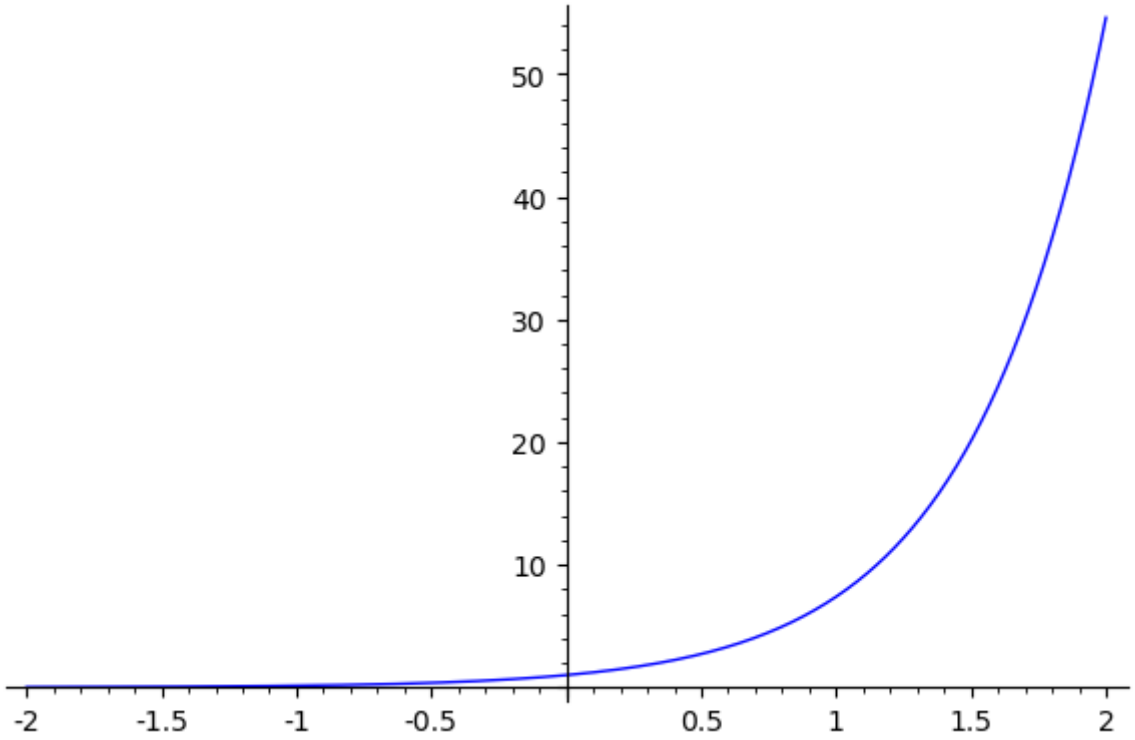

Die zweite Methode besteht darin, die Lösung als eine Funktion zu definieren, die von x und die Integrationskonstante _C

In [7]:
```
sol(x,_C)=desolve(eqd,y)
sol(x,_C)
```

Out[7]:
```
_C*e^(2*x)
```

In [8]:
```
plot(sol(x,1),x,-2,2)
```

Out[8]:




Die Lösungsgraphen entsprechen den Integrationskonstanten

_C = 1, 1.1, 1.2,..., 2

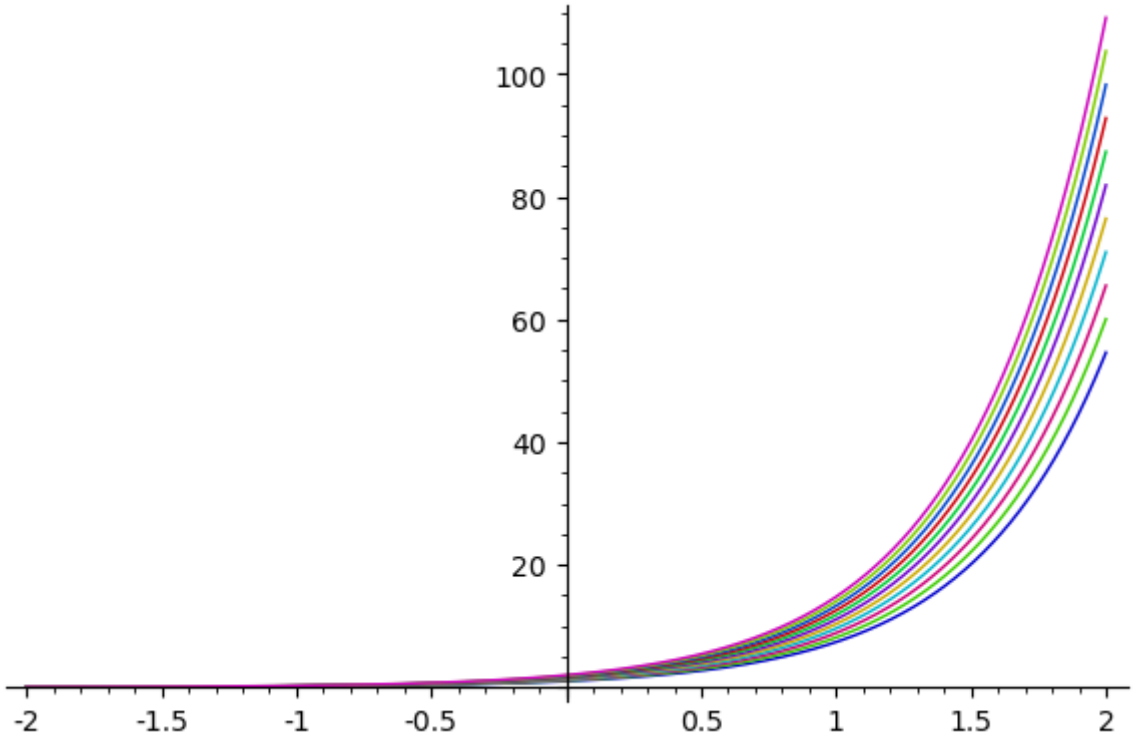
erhalten werden, die die Liste der entsprechenden Lösungsfunktionen und den plot-Befehl definieren.

In [9]:
```
list_f=[sol(x,i/10) for i in [10..20]]
list_f
```

Out[9]:
```
[e^(2*x),
 11/10*e^(2*x),
 6/5*e^(2*x),
 13/10*e^(2*x),
 7/5*e^(2*x),
 3/2*e^(2*x),
 8/5*e^(2*x),
 17/10*e^(2*x),
 9/5*e^(2*x),
 19/10*e^(2*x),
 2*e^(2*x)]
```

In [10]:
```
plot(list_f,x,-2,2)
```

Out[10]:




## Darstellung der Lösung in impliziter Form

Für den Fall, dass die Lösung in impliziter Form erhalten wird, müssen wir den implicit_plot Befehl benutzen um die entsprechenden Graphen zu erhalten. Z.B., lassen Sie uns die Differentialgleichung

(3*y(x)^2+exp(x))* y'(x)+exp(x)*(y(x)+1)+cos(x) = 0

betrachten.

In [11]:
```
x=var('x')
y=function('y')(x)
eqd=(3*y^2+exp(x))*diff(y,x)+exp(x)*(y+1)+cos(x)==0
desolve(eqd,y)
```

Out[11]:
```
y(x)^3 + e^x*y(x) + e^x + sin(x) == _C
```

Beachten Sie, dass in der Lösung impliziter Form erscheine beiden y(x) und x auf der linken Seite. Zuerst konstruieren wir die implizite Lösungsform, in diesem Fall den Ausdruck von der linken Seite des Lösungsausdrucks, in dem wir y(x) durch eine Variable ersetzen müssen (wir bezeichnen dies durch yy).

In [12]:
```
sol=desolve(eqd,y)
sol
```

Out[12]:
```
y(x)^3 + e^x*y(x) + e^x + sin(x) == _C
```

In [13]:
```
yy=var('yy')
sol.substitute(y(x)==yy)
```

Out[13]:
```
yy^3 + yy*e^x + e^x + sin(x) == _C
```

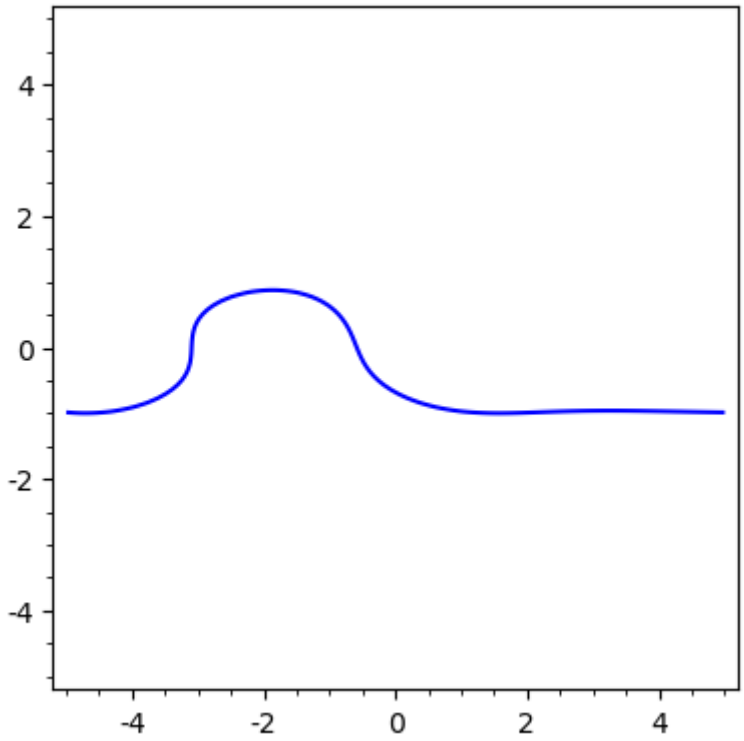
In [14]:
```
_C=var('_C')
f(x,yy,_C)=sol.substitute(y(x)==yy)
f(x,yy,_C)
```

Out[14]:
```
yy^3 + yy*e^x + e^x + sin(x) == _C
```

Für die Integrationskonstante _C=0 erhalten wir folgende Grafik:

In [15]:
```
implicit_plot(f(x,yy,0),(x,-5,5),(yy,-5,5))
```

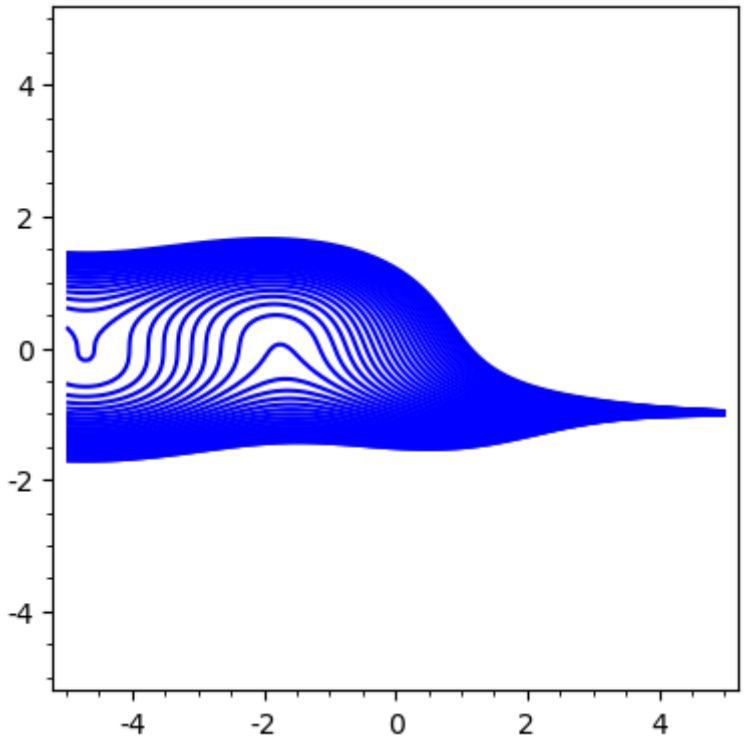
Out[15]:




Wenn wir mehr Lösungen grafisch darstellen wollen, indem wir jeden Graphen einer Variablen zuordnen, kombinieren wir sie, indem wir den Befehl show hinzufügen.

If we want to graph more solutions we assing each graph to a variable, we combine them by adding and we use show command. Zum Beispiel verwenden wir bei den Werten c =-4, -19/5, ...,-1/5, 0, 1/5, 2/5, ..., 4 die for-Schleife, um die Graphen zu generieren:

In [16]:
```
g=implicit_plot(f(x,yy,-4),(x,-5,5),(yy,-5,5))
for i in [-20..20]:
    g1=implicit_plot(f(x,yy,i/5),(x,-5,5),(yy,-5,5))
    g=g+g1
show(g)
```




## Definieren und lösen Differentialgleichung zweiter Ordnung. Darstellung der Lösungen.

Eine Differentialgleichung zweiter Ordnung ist in der gleichen Weise definiert, wie wir es im Falle einer Differentialgleichung erster Ordnung getan haben, zusätzlich wird der Begriff y'' in diesem Fall erscheinen. Betrachten wir die folgende Differentialgleichung zweiter Ordnung:

y''+3y'+2y=1+x^2

In [17]:
```
x=var('x')
y=function('y')(x)
eqd=diff(y,x,2)+3*diff(y,x)+2*y==1+x^2
desolve(eqd,y)
```

Out[17]:
```
1/2*x^2 + _K1*e^(-x) + _K2*e^(-2*x) - 3/2*x + 9/4
```

Beachten Sie, dass die Lösung von zwei Integrationskonstanten abhängt. Wenn wir einige Lösungen zeichnen müssen, müssen wir den Integrationskonstanten einige Werte zuweisen. Wie wir im Fall der ersten Ordnung Differentialgleichungslösungen gesehen, die wir auf zwei Arten tun können, ist die erste Methode, im Lösungsausdruck die entsprechenden Werte für _K1 und _K2 zu ersetzen oder, die zweite Methode, um den Lösungsausdruck als Funktion von drei Variablen x, _K1, _K2 zu konstruieren.

In [18]:
```
sol=desolve(eqd,y)
sol
```

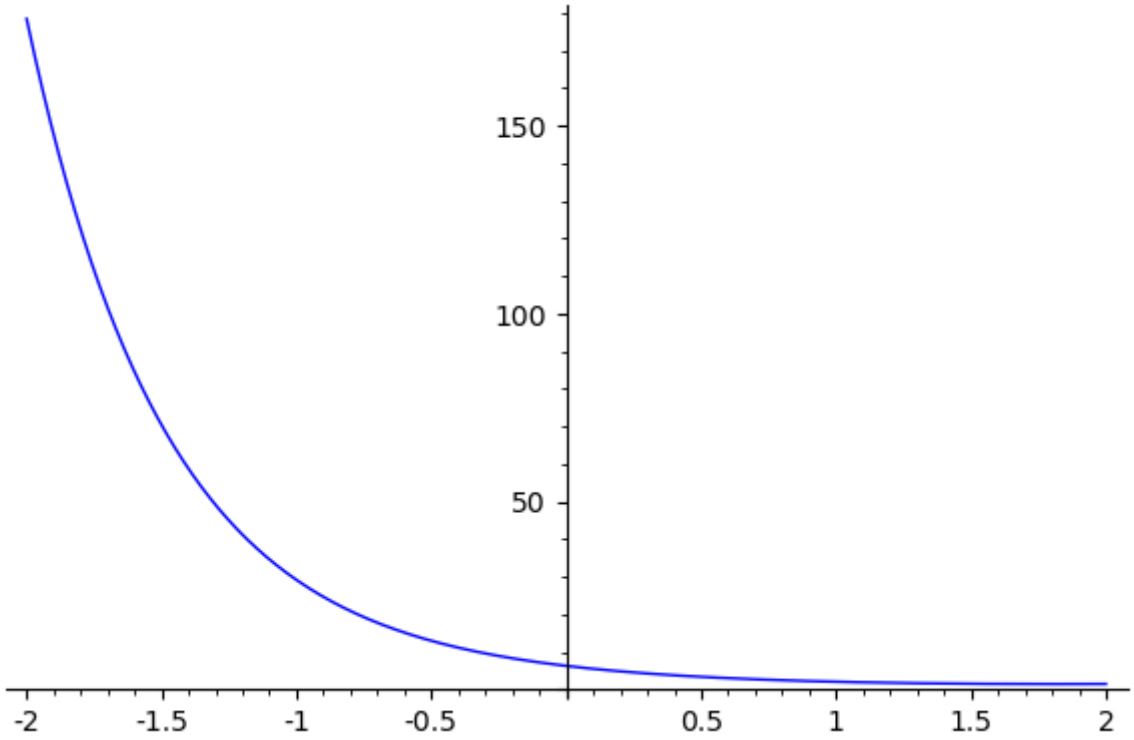
Out[18]:
```
1/2*x^2 + _K1*e^(-x) + _K2*e^(-2*x) - 3/2*x + 9/4
```

In [19]:
```
_K1,_K2=var('_K1,_K2')
sol.substitute(_K1==1,_K2==3)
```

Out[19]:
```
1/2*x^2 - 3/2*x + e^(-x) + 3*e^(-2*x) + 9/4
```

In [20]:
```
sol1=sol.substitute(_K1==1,_K2==3)
plot(sol1,x,-2,2)
```

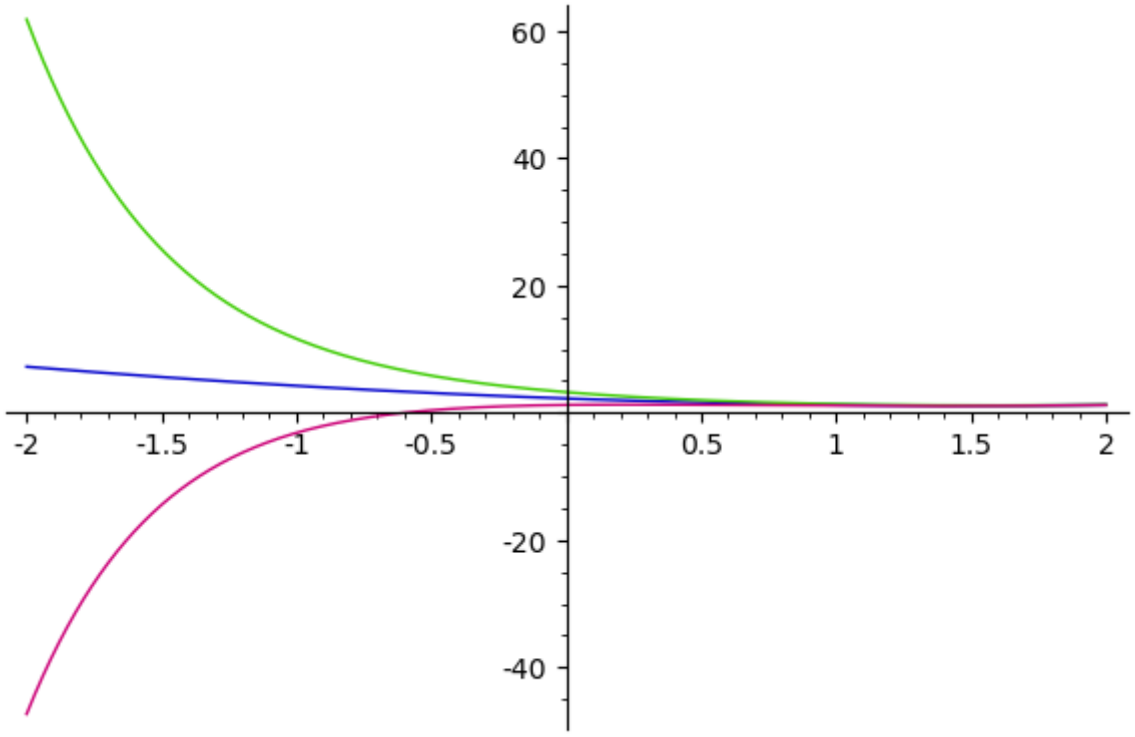
Out[20]:




Wenn wir verschiedene Lösungen vergleichen wollen, die unterschiedlichen Werten von _K1 und _K2 entsprechen, können wir die folgende Reihe schreiben:

In [21]:
```
sol1=sol.substitute(_K1==0,_K2==0)
sol2=sol.substitute(_K1==0,_K2==1)
sol3=sol.substitute(_K1==0,_K2==-1)
plot([sol1,sol2,sol3],x,-2,2)
```

Out[21]:



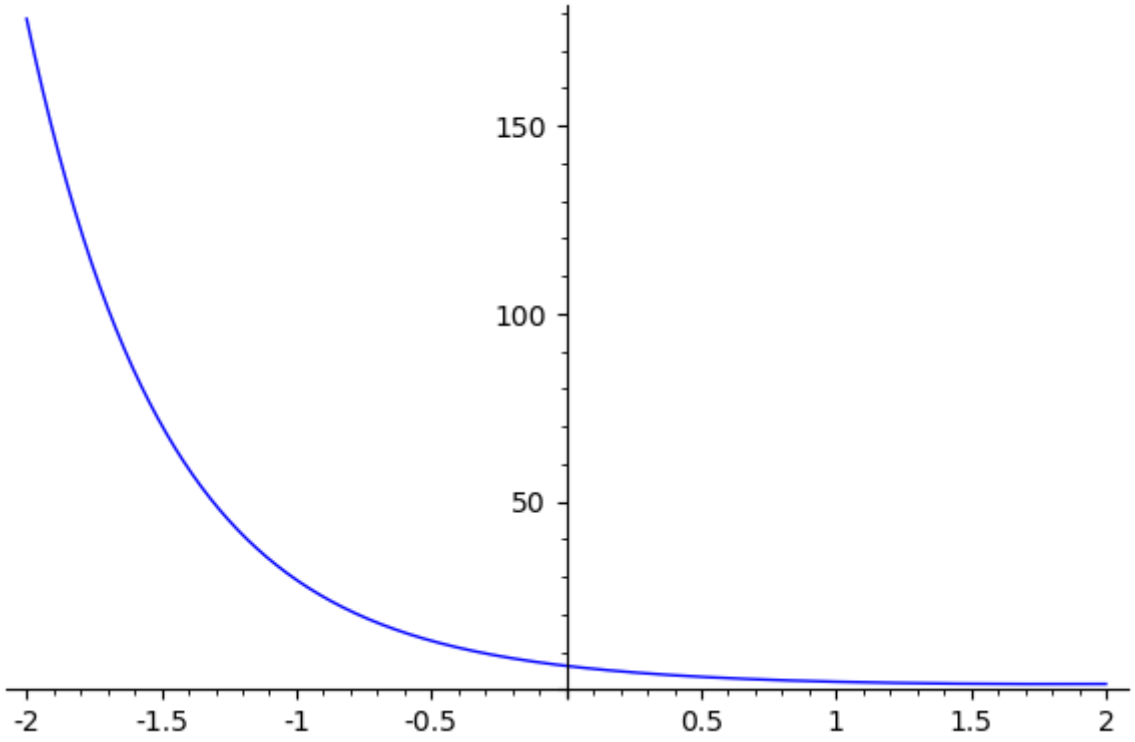

Wir können dieselbe Definition der Lösung als Funktion erhalten, die von drei Variablen x, _K1, _K2 abhängt

In [22]:
```
_K1,_K2=var('_K1,_K2')
sol(x,_K1,_K2)=desolve(eqd,y)
sol(x,_K1,_K2)
```

Out[22]:
```
1/2*x^2 + _K1*e^(-x) + _K2*e^(-2*x) - 3/2*x + 9/4
```

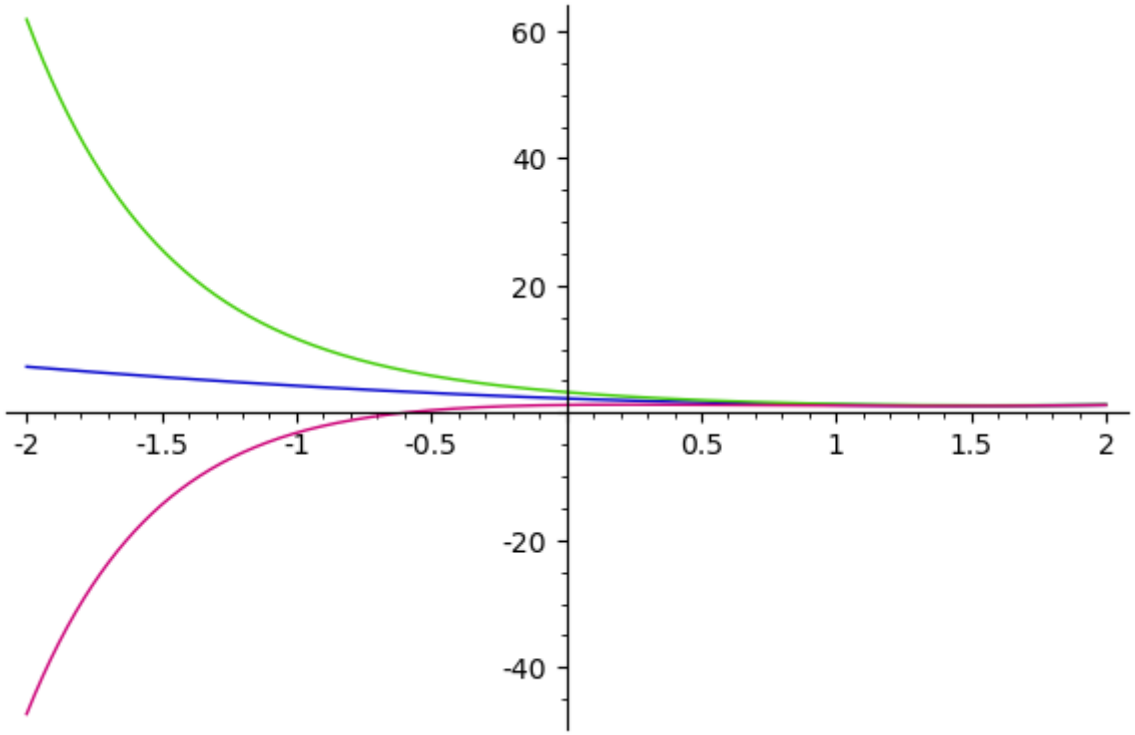
In [23]:
```
plot(sol(x,1,3),x,-2,2)
```

Out[23]:



In [24]:
```
plot([sol(x,0,0),sol(x,0,1),sol(x,0,-1)],x,-2,2)
```

Out[24]:




# Die Lösung der Anfangswertprobleme. Darstellung der Lösung des Anfangswertproblems

## Anfangswertprobleme für Differentialgleichungen erster Ordnung

```
Der Befehl, um die Lösung für eine gegebene AWP für eine
Differentialgleichung erster Ordnung zu finden, ist:

desolve(equation, variable, ics = ...)

und

ics steht für Anfangswertbedingung. Für eine DGL erster Ordnung schreibt
man ics=[x0,y0] für eine gegebene y(x0)
```

Z.B, wenn wir die Lösung für das folgende AWP

y'=2y

y(0)=1

finden, haben wir:

In [25]:
```
x=var('x')
y=function('y')(x)
eqd=diff(y,x)==2*y
desolve(eqd,y,ics=[0,1])
```

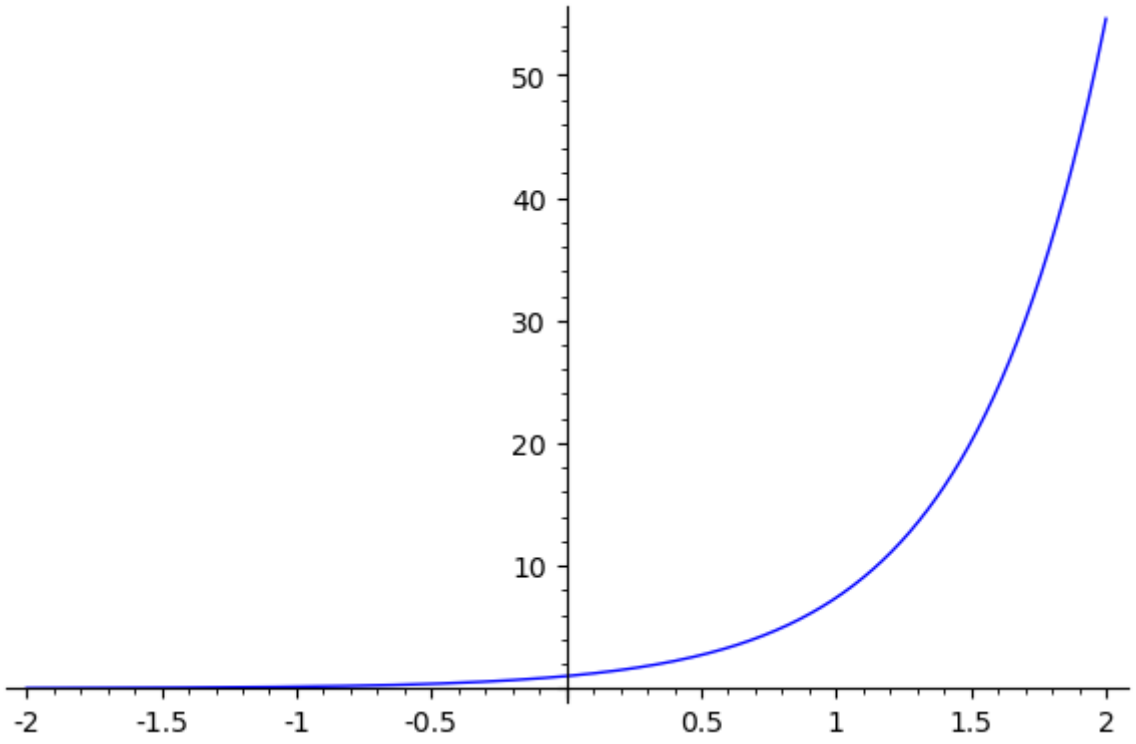
Out[25]:
```
e^(2*x)
```

In [26]:
```
sol=desolve(eqd,y,ics=[0,1])
sol
```

Out[26]:
```
e^(2*x)
```

In [27]:
```
plot(sol,x,-2,2)
```

Out[27]:




In dem Fall, in dem die Differentialgleichung von einem Parameter abhängt, ist es interessant, die Abhängigkeit der Lösung in Bezug auf diesem Parameter zu studieren. Z.B, lassen Sie uns die Abhängigkeit der AWP

y'=k*y

y(0)=1

Lösung in Bezug auf dem Parameter k untersuchen.

```
Wichtige Bemerkung: Wenn die Differentialgleichung von einem Parameter im
desolve-Befehl abhängt, müssen wir die Liste der Variablen angeben, dh,
wenn wir die Lösung y=y(x) finden möchten, verwenden wir den desolve-
Befehl wie folgt:

desolve(eqd, [y,x])
```

In [28]:
```
x,k=var('x,k')
y=function('y')(x)
eqd=diff(y,x)==k*y
desolve(eqd,[y,x],ics=[0,1])
```

Out[28]: e^(k*x)

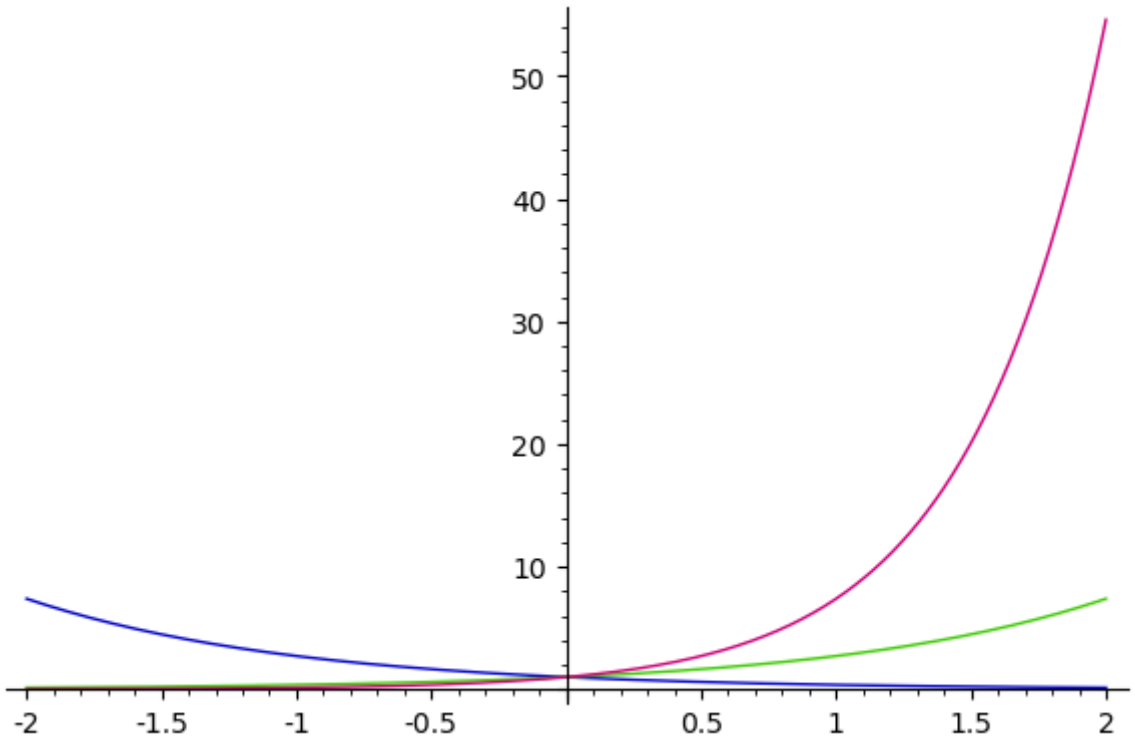
In [29]:
```
sol(x,k)=desolve(eqd,[y,x],ics=[0,1])
sol(x,k)
```

Out[29]: e^(k*x)

Die Abhängigkeit der IVP-Lösung in Bezug auf den Parameter k zu untersuchen, bedeutet, Lösungen für defferente Werte von k zu plotten

In [32]:
```
plot([sol(x,-1),sol(x,1),sol(x,2)],x,-2,2)
```

Out[32]:




In [ ]:

## Anfangswertprobleme für eine Differentialgleichung zweiter Ordnung

```
Der Lösungsbefehl, um die Lösung für eine gegebene IVP für eine
Differentialgleichung zweiter Ordnung zu finden, lautet:

desolve(equation, variable, ics = ...)

wo:

ics=[x0,y0,y1] fur einen gegebenen Anfangswert

y(x0)=y0

y'(x0)=y1
```

Lassen Sie uns die Lösung des IVP

y''+3y'+2y=1+x^2

y(0)=1

y'(0)=1

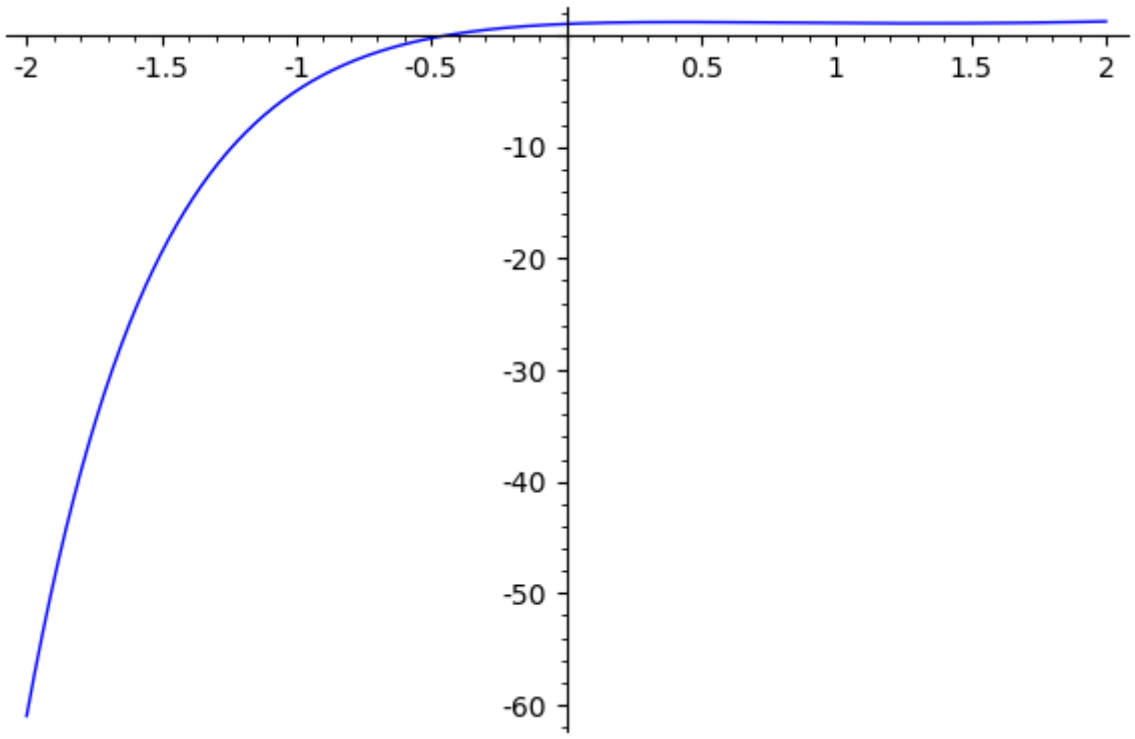
finden.

In [33]:
```
x=var('x')
y=function('y')(x)
eqd=diff(y,x,2)+3*diff(y,x)+2*y==1+x^2
desolve(eqd,y,ics=[0,1,1])
```

Out[33]:
```
1/2*x^2 - 3/2*x - 5/4*e^(-2*x) + 9/4
```

In [34]:
```
sol=desolve(eqd,y,ics=[0,1,1])
plot(sol,x,-2,2)
```

Out[34]:




In [ ]:

In [ ]: