DGL-Systeme mit Sage

Definieren und lösen des DGL-Systems

In [2]:
```
reset()
```

Wir betrachten das folgende DGL-System:

x' = x+y
y' = x-y

In [2]:
```
t = var('t')
x=function('x')(t)
y=function('y')(t)
deq1=diff(x,t)==x+y
deq2=diff(y,t)==x-y
syst=[deq1,deq2]
```

```
Der Sage-Befehl, um die DGL-Systems zu lösen, ist:

desolve_system(des, vars, ics=None, ivar=None)

INPUT:
des : Liste der Gleichungen des Systems

vars : Liste der abhängige Variablen

ics : (optional) Liste der Anfangswerte. Ob ics definiert ist, man muss
fur jede Varabeln Anfangswerte spezifizieren. [x0, y1(x0), y2(x0), ...]

ivar : (optional) unabhängige Variable, man muss sie spezifizieren wenn es
mehrere unabhängige Variablen gibt.
```

In [3]:
```
desolve_system(syst, [x,y])
```

Out[3]:
```
[x(t) == 1/2*sqrt(2)*(x(0) + y(0))*sinh(sqrt(2)*t) + cosh(sqrt(2)*t)*x
(0),
 y(t) == 1/2*sqrt(2)*(x(0) - y(0))*sinh(sqrt(2)*t) + cosh(sqrt(2)*t)*y
(0)]
```

Obs: Die algemeine Lösung hängt von x(0) und y(0) ab. Wenn wir in der algemeine Lösung die Konstanten C1 und C2 haben möchten, müssen wir die Anfangswerte x(0)=C1 und y(0)=C2 benutzen.

In [4]:
```
C1,C2=var('C1,C2')
desolve_system(syst, [x,y],[0,C1,C2])
```

Out[4]:
```
[x(t) == 1/2*sqrt(2)*(C1 + C2)*sinh(sqrt(2)*t) + C1*cosh(sqrt(2)*t),
 y(t) == 1/2*sqrt(2)*(C1 - C2)*sinh(sqrt(2)*t) + C2*cosh(sqrt(2)*t)]
```

Graphische Darstellung der Lösungen

Wenn wir die Lösungen graphisch darstellen möchten, wählen wir verschiedene Werten für die Integrationskonstanten C1, C2 aus. As Erstens, bilden wir aus der Lösung eine Funktion, die von den Variablen t, C1, C2 abhängt.

In [5]:
```
sol=desolve_system(syst, [x,y],[0,C1,C2])
sol_x(t,C1,C2)=sol[0].rhs()
sol_x
```

Out[5]:
```
(t, C1, C2) |--> 1/2*sqrt(2)*(C1 + C2)*sinh(sqrt(2)*t) + C1*cosh(sqrt(2)*
t)
```

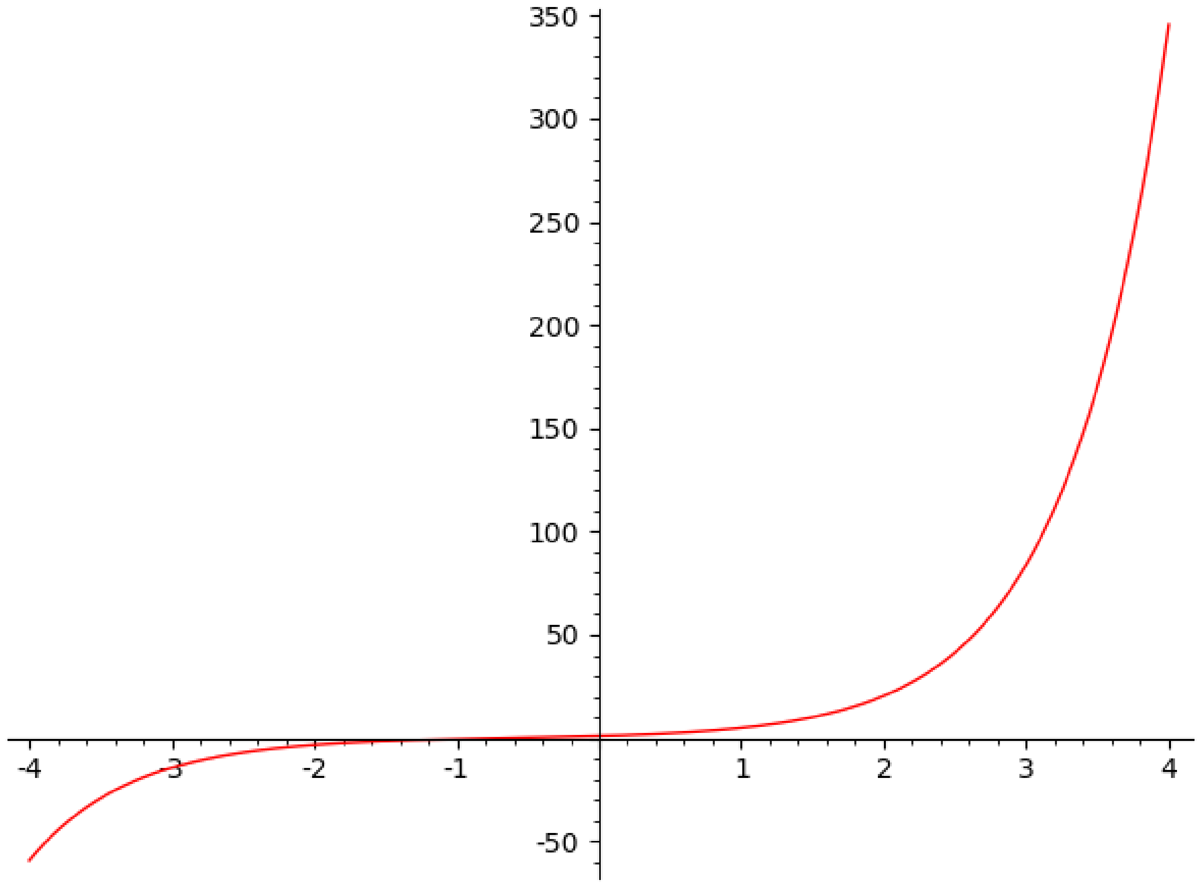
In [6]:
```
sol_y(t,C1,C2)=sol[1].rhs()
sol_y
```

Out[6]:
```
(t, C1, C2) |--> 1/2*sqrt(2)*(C1 - C2)*sinh(sqrt(2)*t) + C2*cosh(sqrt(2)*
t)
```

Wir setzen C1=1 und C2=1 ein.

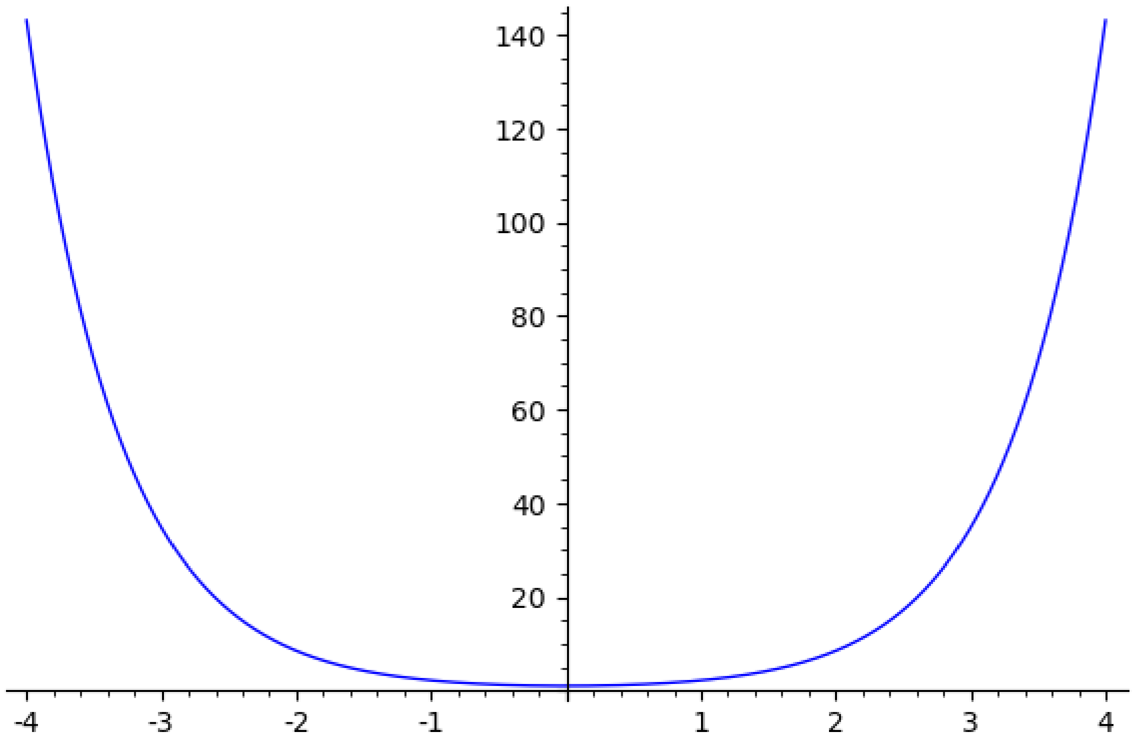
In [7]:
```
plot(sol_x(t,1,1),t,-4,4,color='red')
```

Out[7]:

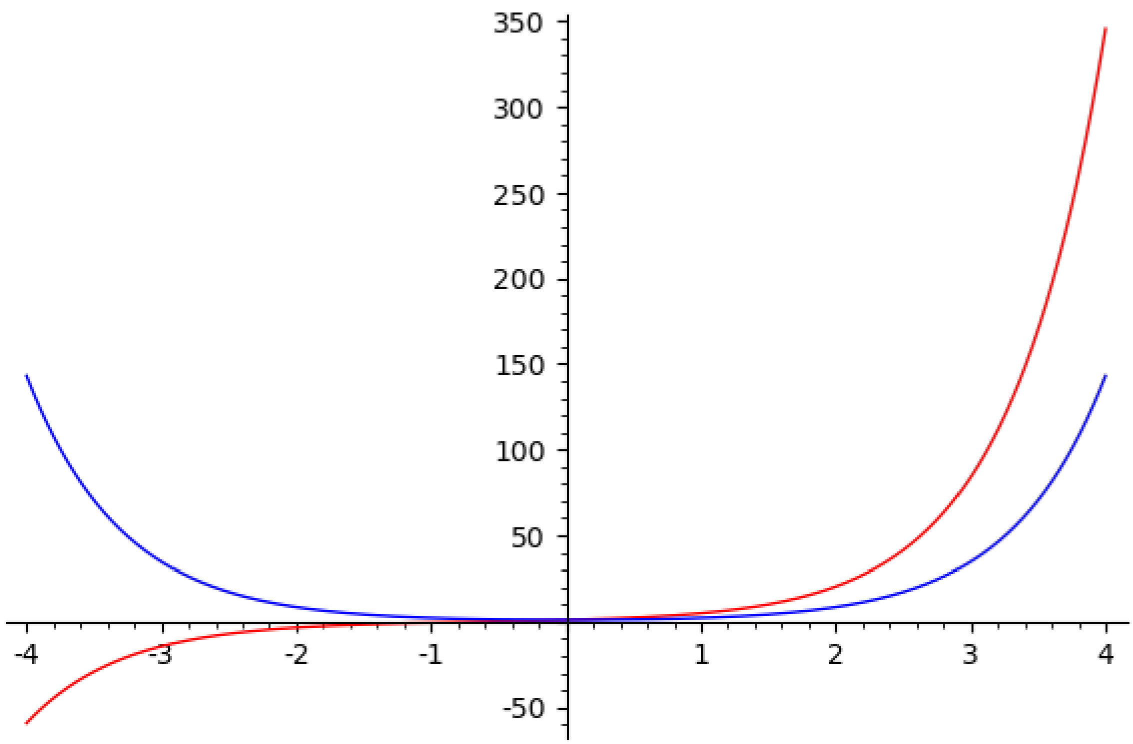


In [8]:
```
plot(sol_y(t,1,1),t,-4,4,color='blue')
```

Out[8]:



In [9]:
```
g1=plot(sol_x(t,1,1),t,-4,4,color='red')
g2=plot(sol_y(t,1,1),t,-4,4,color='blue')
show(g1+g2)
```



Anfangswertprobleme für DGL-Systeme

Wenn wir die Lösung folgendes Problem

x' = x+y
y' = x-y

x(0) = 1

y(0) = 0

berechenen möchten, müssen wir die Anfangswebedingungen (wie bei den Differentialgleichungen) definieren.

In [10]:
```
t = var('t')
x=function('x')(t)
y=function('y')(t)
deq1=diff(x,t)==x+y
deq2=diff(y,t)==x-y
syst=[deq1,deq2]
vars=[x,y]
in_cond=[0,1,0]
```

In [11]:
```
sol=desolve_system(syst, vars,in_cond)
sol
```

Out[11]:
```
[x(t) == 1/2*sqrt(2)*sinh(sqrt(2)*t) + cosh(sqrt(2)*t),
 y(t) == 1/2*sqrt(2)*sinh(sqrt(2)*t)]
```

Um die Losung graphisch darzustellen, werden wir die Funktionen, die von der Variable t abhängen, bilden. Dann benutzen den "plot"-Befehl.

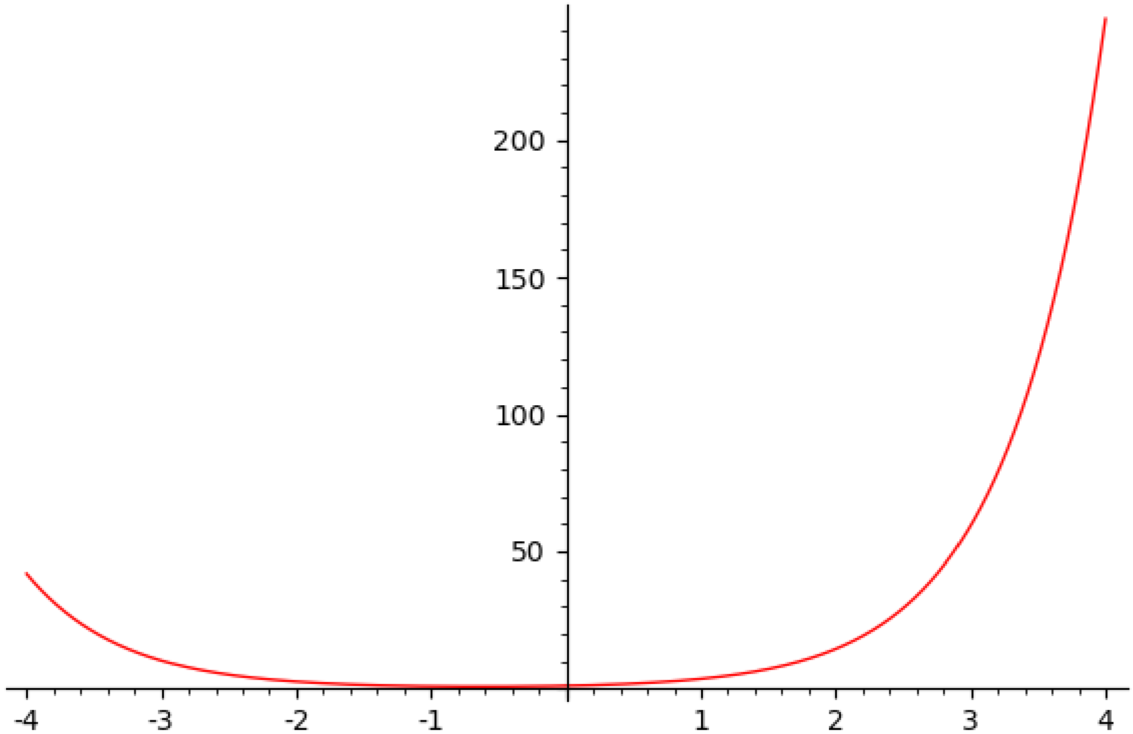
In [12]:
```
sol_x(t)=sol[0].rhs()
sol_x
```

Out[12]:
```
t |--> 1/2*sqrt(2)*sinh(sqrt(2)*t) + cosh(sqrt(2)*t)
```

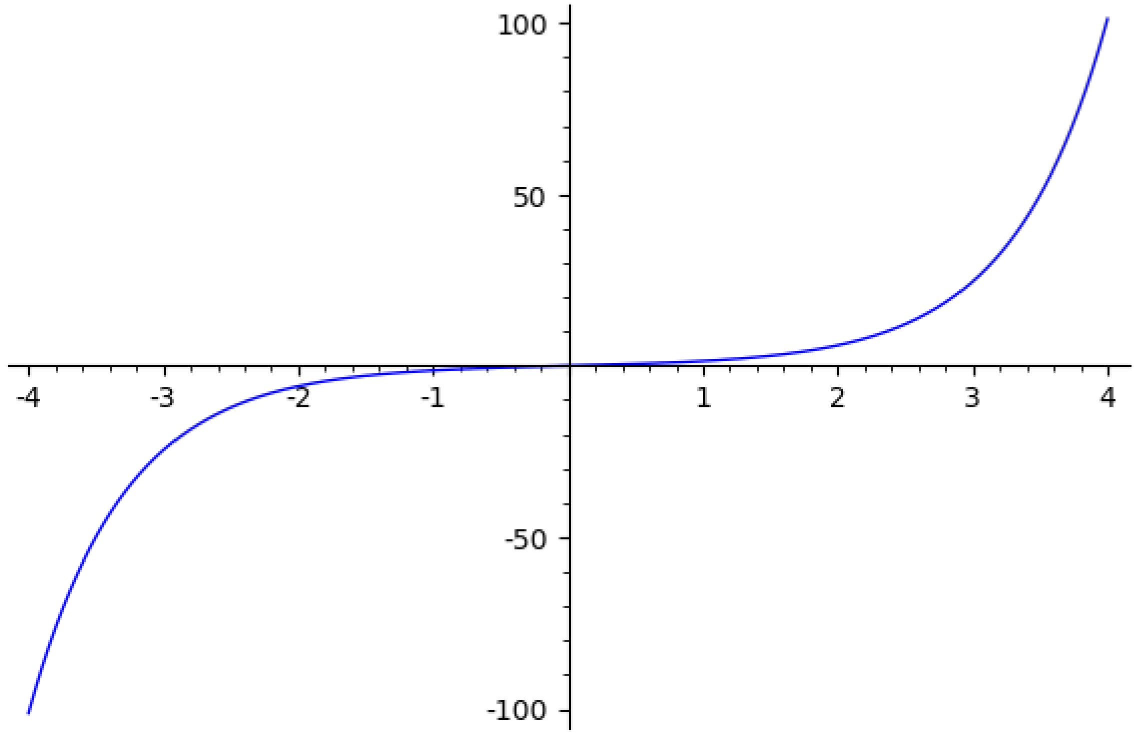
In [13]:
```
sol_y(t)=sol[1].rhs()
sol_y
```

Out[13]:
```
t |--> 1/2*sqrt(2)*sinh(sqrt(2)*t)
```

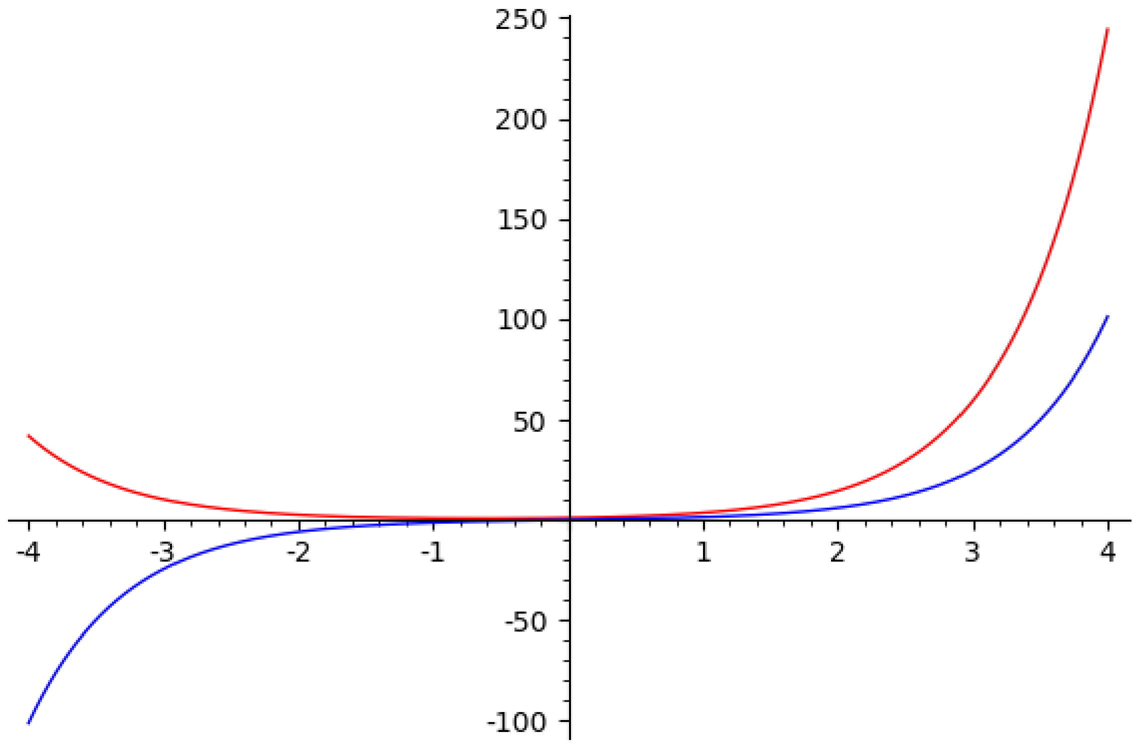
In [14]:
```
plot(sol_x(t),t,-4,4,color='red')
```
Out[14]:



In [15]:
```
plot(sol_y(t),t,-4,4,color='blue')
```
Out[15]:



In [16]:
```
g1=plot(sol_x(t),t,-4,4,color='red')
g2=plot(sol_y(t),t,-4,4,color='blue')
show(g1+g2)
```



Richtungsfeld. Phasenporträt.

Fur ein zweiter Ordnung lineares DGL-System (zwei Gleichungen - System), das Phasenportrait ist eine representative Menge der Losungen, die graphisch als Lösungen in parametrischer Form (von Parameter t) dargestellt in der Ebene xOy sind. (x, y) = (x(t), y(t)), -∞ < t < ∞. In diesem Zusammenhang, die kartesische Ebene heisst die Phasenebene. Die Kurven, die in der parametrischer Form dargestellt sind, heissen Trajektorien oder Orbits. Das Phasenportrait ist ein Instrument um das Langzeitverhalten der Lösungen des Systems zu visualisieren.

Das Richtungsfeld oder das Neigungsfeld ist die Menge der Neigunge am der Systemstrajektoren. Ein Phasenportrait ist ein graphisches Instrument um die Lösungen für ein angegebenes System in der Zukunft zu visualisiren.

Algemenin, für ein System das, in der folgende Form gegeben ist:

x' = f1(x,y) y' = f2(x,y)

können wir den Befehl plot_vector_field([f1(x,y),f2(x,y)], [x,a,b], [y,c,d]) benutzen.

Für dieses System:

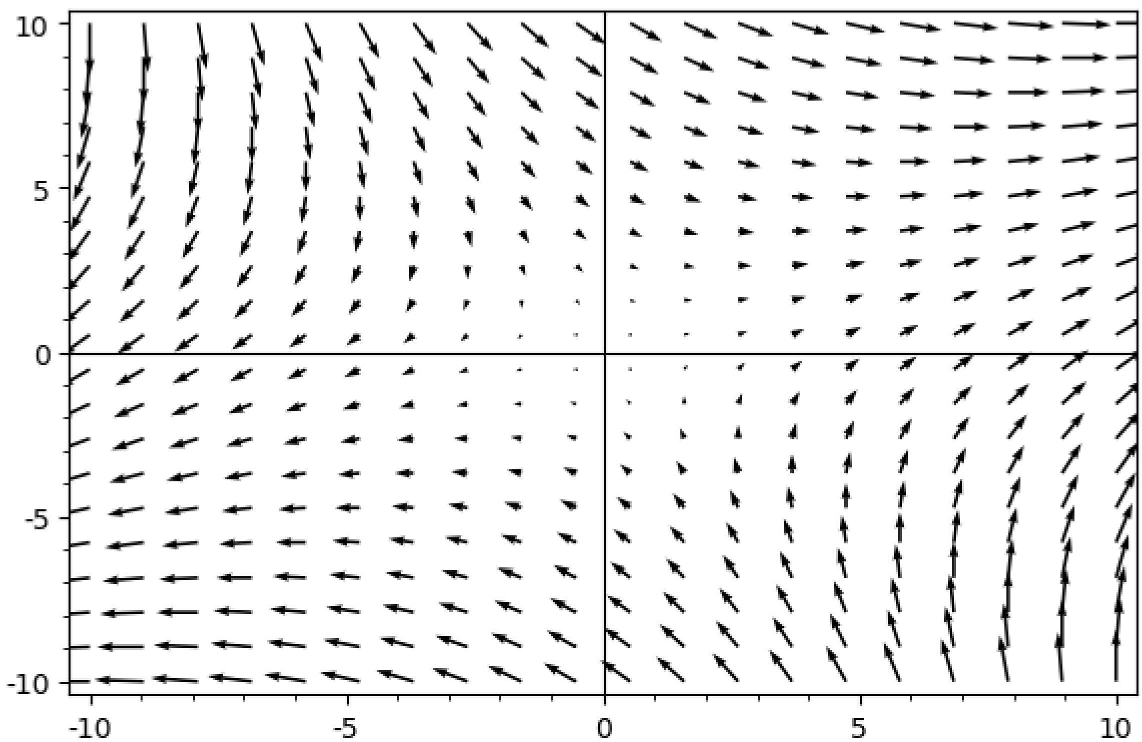
x' = x+y
y' = x-y

werden wir das Phasenporträit darstellen:

In [17]:
```
f1(u,v)=u+v
f2(u,v)=u-v
```

In [18]:
```
plot_vector_field( [f1(u,v),f2(u,v)], [u,-10,10], [v,-10,10] )
```

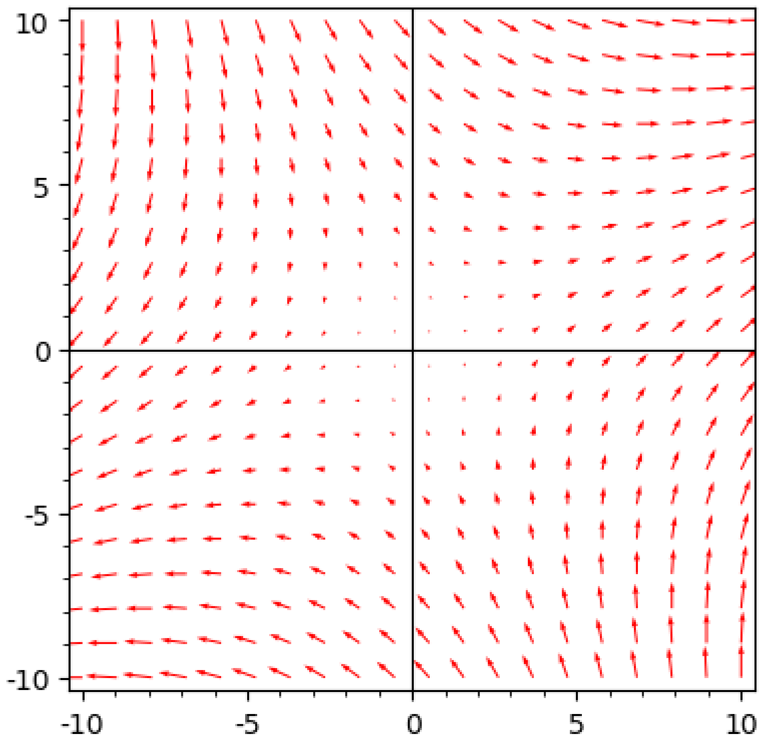
Out[18]:



Ob wir die dieselbe Koordinaten an den beiden Achsen haben möchten, benutzen wir die Option aspect_ratio = 1.

In [19]:
```
plot_vector_field( [f1(u,v),f2(u,v)], [u,-10,10], [v,-10,10], color='red' ,
```

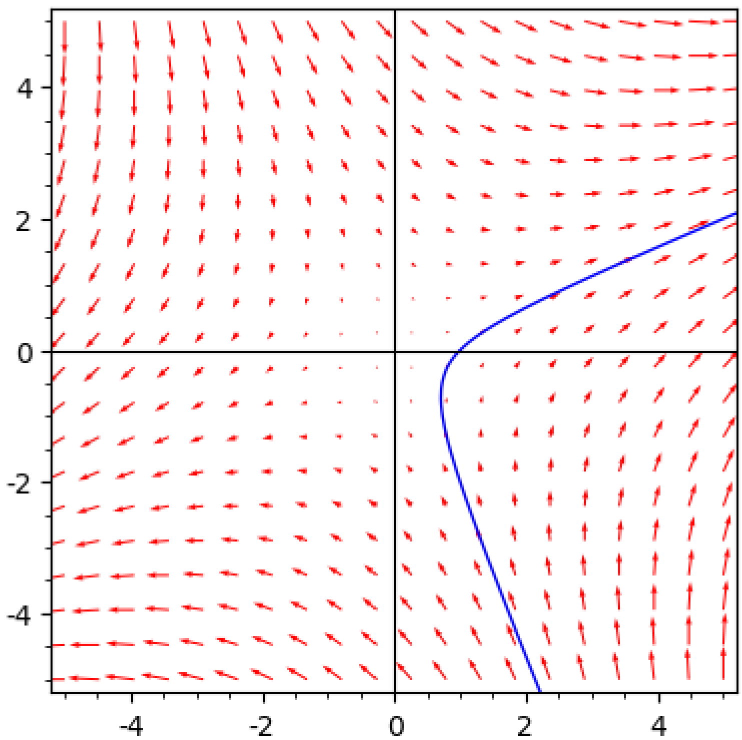
Out[19]:



Ob wir auch das Orbit entsprechend der Lösung des Systems, die die Anfangswerte x(0)=1 und y(0)=0 erfüllt, (das heisst, wir stellen das Orbit das durch den Punkt (1,0) läuft dar), benutzen wir den Befehl parametric_plot.

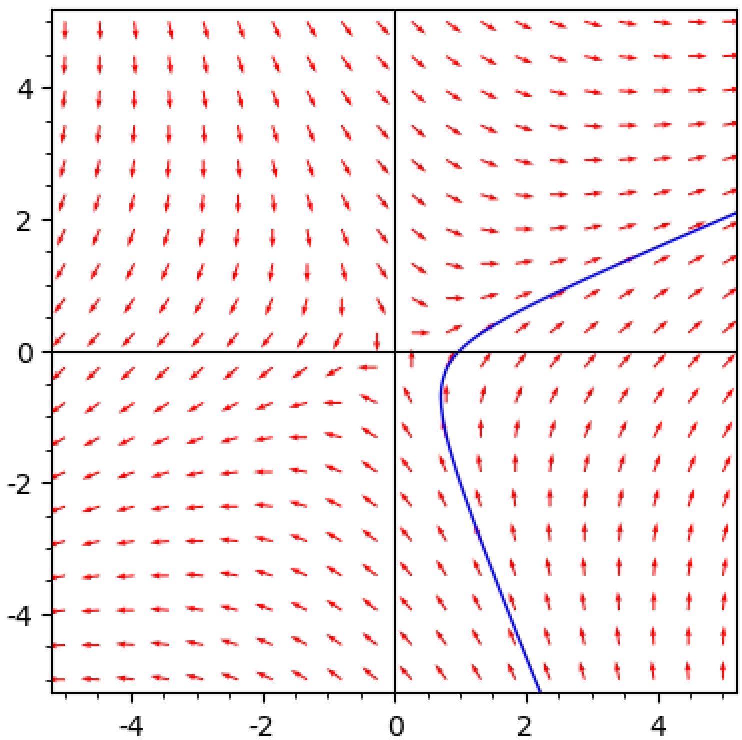
In [20]:
```
sol=desolve_system(syst, vars,in_cond)
sol_x(t)=sol[0].rhs()
sol_y(t)=sol[1].rhs()
```

In [21]:
```
g1=plot_vector_field( [f1(u,v),f2(u,v)], [u,-5,5], [v,-5,5],color='red' ,as
g2=parametric_plot((sol_x(t), sol_y(t)), (t, -10, 10),color='blue')
g=g1+g2
g.show(xmin=-5,xmax=5,ymin=-5,ymax=5)
```



Wenn die Pfeilen zu klein sind, können wir das Verhalten um den Punkt (0,0) sehen. Dann, werden wir das Vektor, das graphisch represäntieren wird, mit ihre Länge skalieren.

In [22]:
```
n=sqrt(f1(u,v)^2+f2(u,v)^2)
g1=plot_vector_field( [f1(u,v)/n,f2(u,v)/n], [u,-5,5], [v,-5,5],color='red'
g2=parametric_plot((sol_x(t), sol_y(t)), (t, -10, 10),color='blue')
g=g1+g2
g.show(xmin=-5,xmax=5,ymin=-5,ymax=5)
```



Wenn wir mehrere Trajektorien darstellen möchten, müssen wir für jeden Anfangswert eine Darstellung generieren.

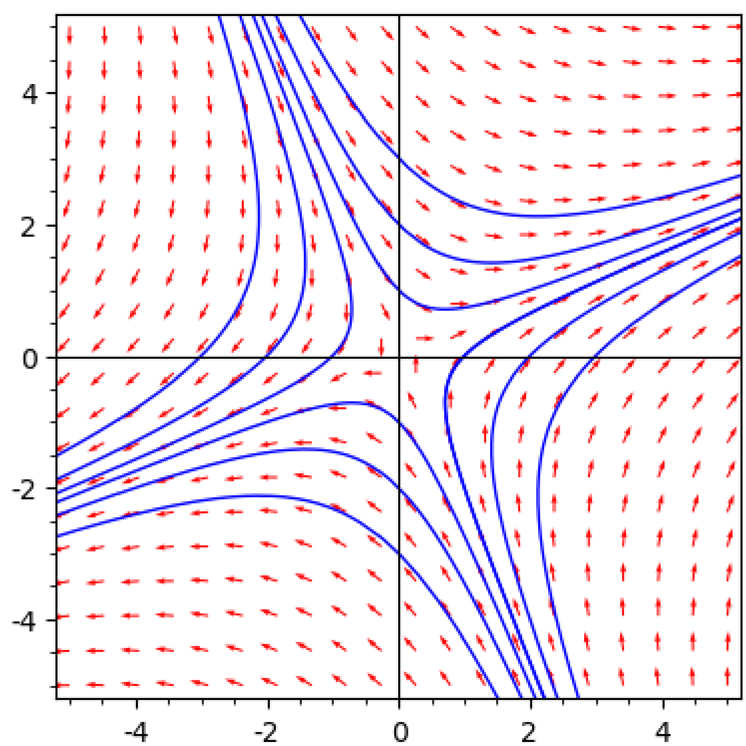
Lasst uns die Orbits für die folgenden Punkte darstellen (entsprechend den Anfangswerten x(0)=x0, y(0)=y0):

(1,0), (2,0), (3,0), (-1,0), (-2,0), (-3,0), (0,1), (0,2), (0,3), (0,-1), (0,-2), (0,-3)

In [23]:

```
g1=plot_vector_field( [f1(u,v),f2(u,v)], [u,-5,5], [v,-5,5],color='red' ,as
for k in [1..3]:
    in_cond=[0,k,0]
    sol=desolve_system(syst, vars,in_cond)
    sol_x(t)=sol[0].rhs()
    sol_y(t)=sol[1].rhs()
    g1=parametric_plot((sol_x(t), sol_y(t)), (t, -10, 10),color='blue')
    in_cond=[0,-k,0]
    sol=desolve_system(syst, vars,in_cond)
    sol_x(t)=sol[0].rhs()
    sol_y(t)=sol[1].rhs()
    g2=parametric_plot((sol_x(t), sol_y(t)), (t, -10, 10),color='blue')
    in_cond=[0,0,k]
    sol=desolve_system(syst, vars,in_cond)
    sol_x(t)=sol[0].rhs()
    sol_y(t)=sol[1].rhs()
    g3=parametric_plot((sol_x(t), sol_y(t)), (t, -10, 10),color='blue')
    in_cond=[0,0,-k]
    sol=desolve_system(syst, vars,in_cond)
    sol_x(t)=sol[0].rhs()
    sol_y(t)=sol[1].rhs()
    g4=parametric_plot((sol_x(t), sol_y(t)), (t, -10, 10),color='blue')
    g=g+g1+g2+g3+g4

g.show(xmin=-5,xmax=5,ymin=-5,ymax=5)
```



In [ ]:

In [ ]: