6. Problemas de autovalores e autovetores

Sistemas de duas equações diferenciais lineares de 1ª ordem homogêneos com coeficientes constantes

Type *Markdown* and LaTeX: α^2

Exemplo 1, p.304

In [5]:

```
import sympy as sp
sp.init_printing()
```

In [42]:

```
t = sp.symbols('t', real = True)
c1, c2 = sp.symbols('c1, c2', real = True)
x1 = sp.Function('x1')(t)
x2 = sp.Function('x2')(t)

A = sp.Matrix([[1,1],[4,1]])
A
```

Out[42]:

$$\begin{bmatrix} 1 & 1 \\ 4 & 1 \end{bmatrix}$$

In [43]:

```
A[0,0]
```

Out[43]:

$$1$$

Para encontrar os autovalores de uma matriz, use `eigenvals` . Retorna um dicionário de pares `aulovalor:multiplicidade` .

In [44]:

```
A.eigenvals()
```

Out[44]:

$$\{-1 : 1,\ 3 : 1\}$$

In [45]:

```
r1 = list(_.keys())[0]
r2 = list(_.keys())[1]
print(r1,r2)
```

```
3 -1
```

Para encontrar os autovetores de uma matriz, use `eigenvects`. Retorna uma lista de tuplas da forma `(autovalor,multiplicidade, autovetor)`.

In [46]:

```
A.eigenvects()
```

Out[46]:

$$\left[\left(-1,\ 1,\ \left[\begin{matrix}-\frac{1}{2}\\ 1\end{matrix}\right]\right),\ \left(3,\ 1,\ \left[\begin{matrix}\frac{1}{2}\\ 1\end{matrix}\right]\right)\right]$$

In [47]:

```
v1 = _[0][2][0]
v2 = _[1][2][0]
print(v1,v2)
```

```
Matrix([[-1/2], [1]]) Matrix([[1/2], [1]])
```

In [48]:

```
x = c1*v1*sp.exp(r1*t) + c2*v2*sp.exp(r2*t)
x
```

Out[48]:

$$\left[\begin{matrix}-\frac{c_1 e^{3t}}{2} + \frac{c_2 e^{-t}}{2}\\ c_1 e^{3t} + c_2 e^{-t}\end{matrix}\right]$$

In [49]:

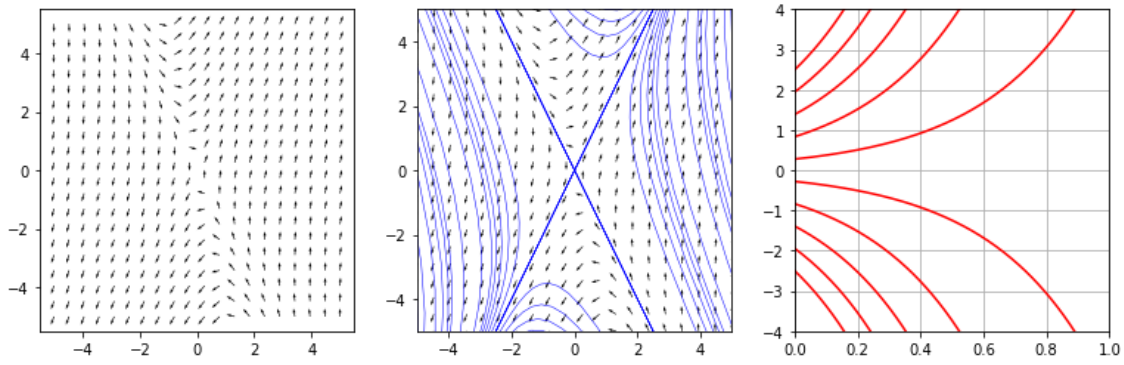
```
f1 = sp.lambdify((t,c1,c2), x[0])
f2 = sp.lambdify((t,c1,c2), x[1])
```

In [50]:

```
import numpy as np
import matplotlib.pyplot as plt
```

In [51]:

```
x1,x2 = np.meshgrid(np.linspace(-5,5,20),
                    np.linspace(-5,5,20))

fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(13, 4))

u1 = x1 + x2
u2 = 4*x1 + x2
u1 = u1/np.sqrt(u1**2+u2**2)
u2 = u2/np.sqrt(u1**2+u2**2)
axs[0].quiver(x1, x2, u1, u2)


ti = np.linspace(-10, 10, 200)
for c1 in np.linspace(-10, 10, 11):
    for c2 in np.linspace(-10, 10, 11):
        axs[1].plot(f1(ti,c1,c2), f2(ti,c1,c2), 'b-', linewidth=0.5)
axs[1].quiver(x1, x2, u1, u2)
axs[1].set_xlim([-5,5])
axs[1].set_ylim([-5,5])

ti = np.linspace(0, 1, 200)
for c1 in np.linspace(-5, 5, 10):
    for c2 in np.linspace(0, 1, 1):
        axs[2].plot(ti, f1(ti,c1,c2), 'r')
        #axs[2].plot(ti, f2(ti,c1,c2), 'b')
axs[2].set_xlim([0,1])
axs[2].set_ylim([-4,4])
plt.grid()

plt.show()
```



In [58]:

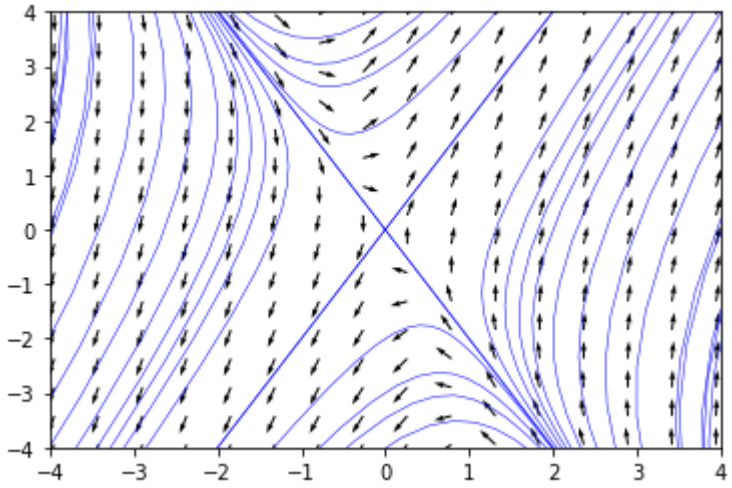
```
u1 = sp.Function('u1')
u2 = sp.Function('u2')
u = A*sp.Matrix([u1(t),u2(t)])
u
```

Out[58]:

$$\begin{bmatrix} 1.0\,u_1(t) + 1.0\,u_2(t) \\ 4.0\,u_1(t) + 1.0\,u_2(t) \end{bmatrix}$$

In [60]:

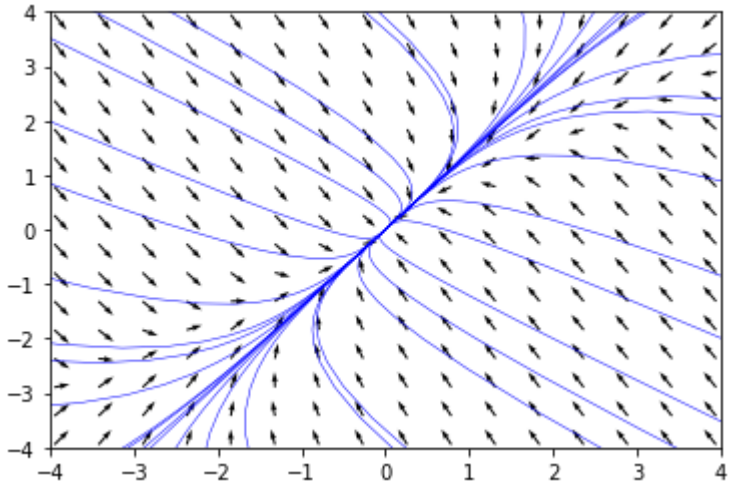
```
eqs = [sp.Eq(sp.Derivative(u1(t), t), u[0]),
       sp.Eq(sp.Derivative(u2(t), t), u[1])]

ti = np.linspace(-10, 10, 200)

for c1 in np.linspace(-10,10, 11):
    for c2 in np.linspace(-10,10, 11):
        f1, f2 = sp.dsolve(eqs, [u1(t), u2(t)], ics={u1(0): c1, u2(0): c2})
        uu1 = sp.lambdify(t, f1.rhs)
        uu2 = sp.lambdify(t, f2.rhs)
        plt.plot(uu1(ti), uu2(ti), 'b-', linewidth=0.5)

x1,x2 = np.meshgrid(np.linspace(-5,5,20),
                    np.linspace(-5,5,20))

A = np.array(A, dtype=np.float64)
v1 = A[0,0]*x1 + A[0,1]*x2
v2 = A[1,0]*x1 + A[1,1]*x2

v1 = v1/np.sqrt(v1**2+v2**2)
v2 = v2/np.sqrt(v1**2+v2**2)
plt.quiver(x1, x2, v1, v2)

plt.xlim([-4,4])
plt.ylim([-4,4])

plt.show()
```



Exemplo 2, p.306

In [44]:

```
eqs = [sp.Eq(sp.Derivative(u1(t), t), -3*u1(t)+np.sqrt(2)*u2(t)),
       sp.Eq(sp.Derivative(u2(t), t),  np.sqrt(2)*u1(t)-2*u2(t))]

ti = np.linspace(-10, 10, 200)


for c1 in np.linspace(-10,10, 6):
    for c2 in np.linspace(-10,10, 6):
        f1, f2 = sp.dsolve(eqs, [u1(t), u2(t)], ics={u1(0): c1, u2(0): c2})
        uu1 = sp.lambdify(t, f1.rhs)
        uu2 = sp.lambdify(t, f2.rhs)
        plt.plot(uu1(ti), uu2(ti), 'b-', linewidth=0.5)

x1,x2 = np.meshgrid(np.linspace(-5,5,20),
                    np.linspace(-5,5,20))

v1 = -3*x1+np.sqrt(2)*x2
v2 = np.sqrt(2)*x1-2*x2

v1 = v1/np.sqrt(v1**2+v2**2)
v2 = v2/np.sqrt(v1**2+v2**2)
plt.quiver(x1, x2, v1, v2)

plt.xlim([-4,4])
plt.ylim([-4,4])

plt.show()
```



Autovalores complexos

Exemplo 1, p.312

In [45]:

```
A = sp.Matrix([[-1/2,1],[-1,-1/2]])
```

In [46]:

```
A.eigenvals()
```

Out[46]:

$$\{-0.5 - 1.0i : 1,\ -0.5 + 1.0i : 1\}$$

In [47]:

```
A.eigenvects()
```

Out[47]:

$$\left[\left(-0.5 - 1.0i,\ 1,\ \left[\begin{matrix}-0.707106781186548i\\-0.707106781186548\end{matrix}\right]\right),\ \left(-0.5 + 1.0i,\ 1,\ \left[\begin{matrix}0.\\0.\end{matrix}\right]\right.\right.$$

In [48]:

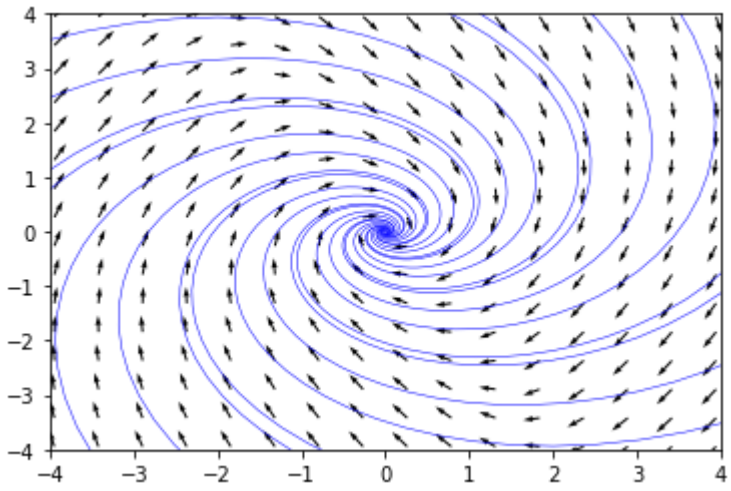
```
t = sp.symbols('t', real = True)
c1, c2 = sp.symbols('c1, c2', real = True)
```

In [49]:

```
eqs = [sp.Eq(sp.Derivative(u1(t), t), (A*sp.Matrix([u1(t),u2(t)])).row(0)[0]),
       sp.Eq(sp.Derivative(u2(t), t), (A*sp.Matrix([u1(t),u2(t)])).row(1)[0])]
f1, f2 = sp.dsolve(eqs, [u1(t), u2(t)])
```

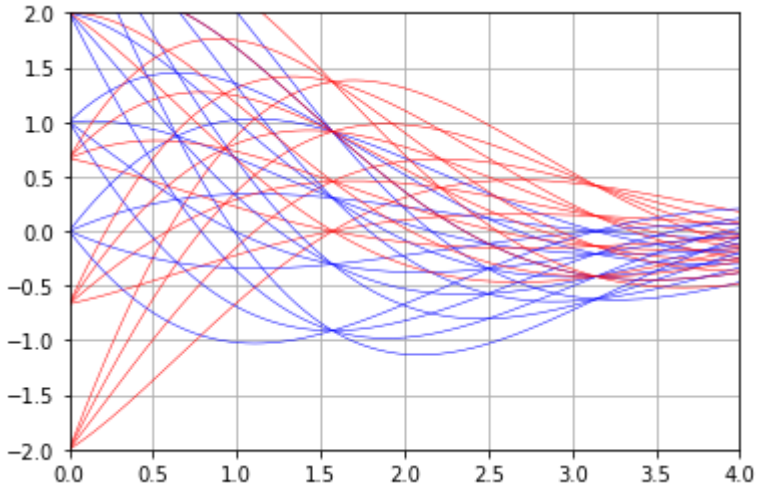
In [50]:

```
ti = np.linspace(-10, 10, 200)

for c1 in np.linspace(-5,5, 4):
    for c2 in np.linspace(-5,5,4):
        f1, f2 = sp.dsolve(eqs, [u1(t), u2(t)], ics={u1(0): c1, u2(0): c2})
        uu1 = sp.lambdify(t, f1.rhs)
        uu2 = sp.lambdify(t, f2.rhs)
        plt.plot(uu1(ti), uu2(ti), 'b-', linewidth=0.5)

x1,x2 = np.meshgrid(np.linspace(-5,5,20),
                    np.linspace(-5,5,20))

v1 = -1/2*x1+x2
v2 = -x1-1/2*x2

v1 = v1/np.sqrt(v1**2+v2**2)
v2 = v2/np.sqrt(v1**2+v2**2)
plt.quiver(x1, x2, v1, v2)

plt.xlim([-4,4])
plt.ylim([-4,4])

plt.show()
plt.show()
```



In [51]:

```
ti = np.linspace(0, 10, 200)
for c1 in np.linspace(-2,2, 4):
    for c2 in np.linspace(0,3,4):
        f1, f2 = sp.dsolve(eqs, [u1(t), u2(t)], ics={u1(0): c1, u2(0): c2})
        uu1 = sp.lambdify(t, f1.rhs)
        uu2 = sp.lambdify(t, f2.rhs)
        plt.plot(ti, uu1(ti), 'r-', linewidth=0.5)
        plt.plot(ti, uu2(ti), 'b-', linewidth=0.5)

plt.xlim([0,4])
plt.ylim([-2,2])
plt.grid()
```



Autovalores repetidos

Exemplo 1, p.312

In [52]:

```
A = sp.Matrix([[1,-1],[1,3]])
A.eigenvals()
```

Out[52]:

$$\{2 : 2\}$$

In [53]:

```
A.eigenvects()
```

Out[53]:

$$\left[\left(2,\ 2,\ \left[\begin{matrix}-1\\1\end{matrix}\right]\right)\right]$$

In [57]:

```
eqs = [sp.Eq(sp.Derivative(u1(t), t), (A*sp.Matrix([u1(t),u2(t)])).row(0)[0]),
       sp.Eq(sp.Derivative(u2(t), t), (A*sp.Matrix([u1(t),u2(t)])).row(1)[0])]
f1, f2 = sp.dsolve(eqs, [u1(t), u2(t)])
```

In [58]:

```
f1
```

Out[58]:

$$\mathrm{u}_1(t) = (-C_1 - C_2 t + C_2)\, e^{2t}$$

In [59]:

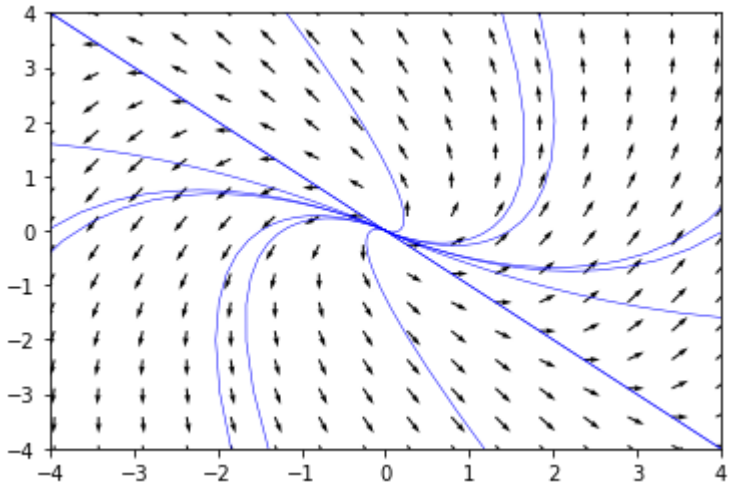
```
f2
```

Out[59]:

$$\mathrm{u}_2(t) = (C_1 + C_2 t)\, e^{2t}$$

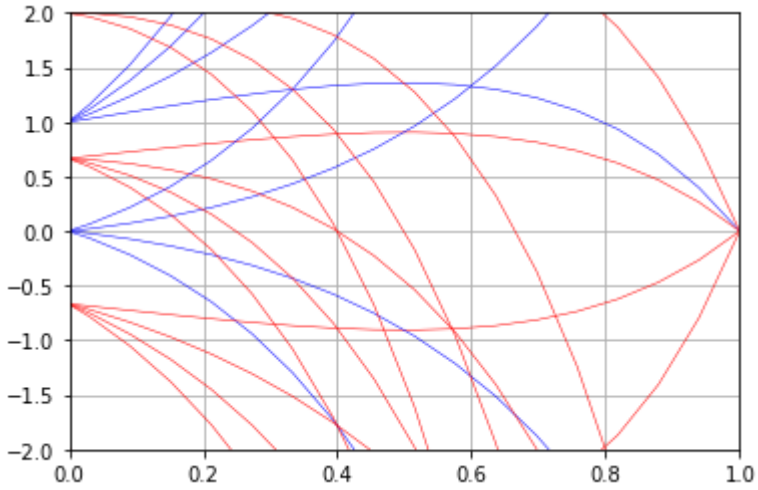
In [69]:

```
ti = np.linspace(-10, 10, 200)

for c1 in np.linspace(-5,5, 4):
    for c2 in np.linspace(-5,5,4):
        f1, f2 = sp.dsolve(eqs, [u1(t), u2(t)], ics={u1(0): c1, u2(0): c2})
        uu1 = sp.lambdify(t, f1.rhs)
        uu2 = sp.lambdify(t, f2.rhs)
        plt.plot(uu1(ti), uu2(ti), 'b-', linewidth=0.5)

x1,x2 = np.meshgrid(np.linspace(-5,5,20),
                    np.linspace(-5,5,20))



x1,x2 = np.meshgrid(np.linspace(-5,5,20),
                    np.linspace(-5,5,20))

v1 = x1-x2
v2 = x1+x2

v1 = v1/np.sqrt(v1**2+v2**2)
v2 = v2/np.sqrt(v1**2+v2**2)
plt.quiver(x1, x2, v1, v2)

plt.xlim([-4,4])
plt.ylim([-4,4])

plt.show()
plt.show()
```



In [82]:

```
ti = np.linspace(-3, 3, 100)
for c1 in np.linspace(-2,2, 4):
    for c2 in np.linspace(0,3,4):
        f1, f2 = sp.dsolve(eqs, [u1(t), u2(t)], ics={u1(0): c1, u2(0): c2})
        uu1 = sp.lambdify(t, f1.rhs)
        uu2 = sp.lambdify(t, f2.rhs)
        plt.plot(ti, uu1(ti), 'r-', linewidth=0.5)
        plt.plot(ti, uu2(ti), 'b-', linewidth=0.5)

plt.xlim([0,1])
plt.ylim([-2,2])
plt.grid()
```



Type *Markdown* and LaTeX: α^2