f-22-jupyter-salt

April 22, 2021

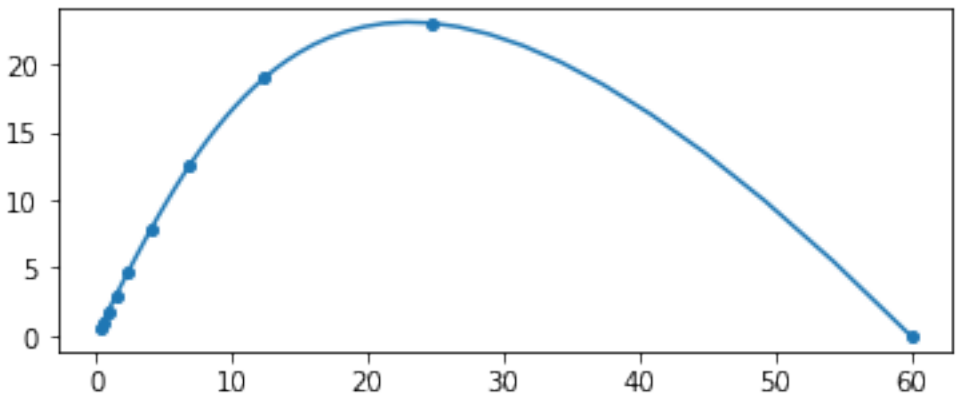
[1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
v0 = np.array([1.0, 2.0])[:, np.newaxis]
v1 = np.array([1.0, -2.0])[:, np.newaxis]
lambda0 = -1.0 / 20.0
lambda1 = -3.0 / 20.0
```

[3]:
```
t = np.linspace(0, 100, 100)
løsning = (30 * v0 * np.exp(lambda0 * t)
           + 30 * v1 * np.exp(lambda1 * t))
```

[4]:
```
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.plot(*løsning,
        marker='o', markevery=10, markersize=4)
```

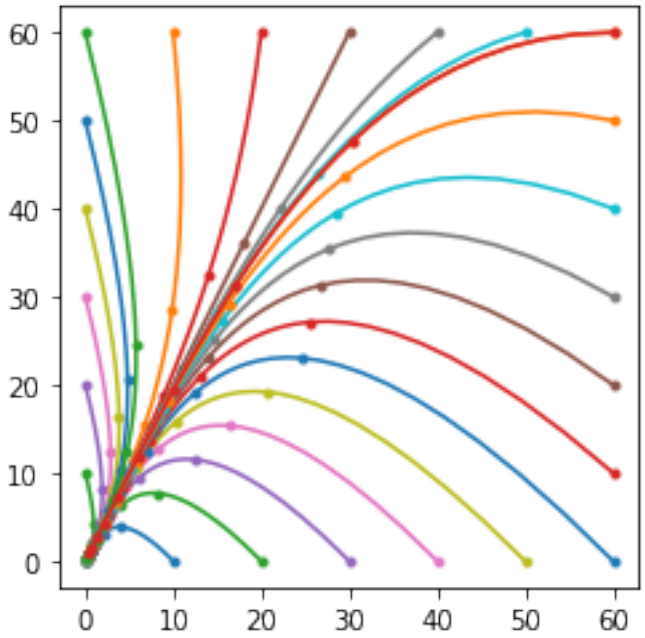
[4]:
```
[<matplotlib.lines.Line2D at 0x106fc8ca0>]
```



[5]:
```
v = np.hstack([v0, v1])

def sol(r0, r1, v, lambda0, lambda1, t):
    c = np.linalg.solve(v, np.array([r0, r1])[:, np.newaxis])
```

```
    return (c[0] * v[:, [0]] * np.exp(lambda0 * t)
            + c[1] * v[:, [1]] * np.exp(lambda1 * t))
```

[6]:
```
t = np.linspace(0, 100, 100)

marks = dict(marker='o', markevery=10, markersize=3)

fig, ax = plt.subplots()
ax.set_aspect('equal')
for s in np.linspace(10, 60, 6):
    ax.plot(*sol(s, 0, v, lambda0, lambda1, t), **marks)
    ax.plot(*sol(s, 60, v, lambda0, lambda1, t), **marks)
for s in np.linspace(10, 60, 6):
    ax.plot(*sol(0, s, v, lambda0, lambda1, t), **marks)
    ax.plot(*sol(60, s, v, lambda0, lambda1, t), **marks)
```



[7]:
```
# kredsløb eksemplet
```

[8]:
```
a_mat = np.array([[-2.0, -2.5],
                  [10.0, -2.0]])
```

[9]:
```
# karakteristisk poly a x^2 + b x + c
```

[10]:
```
a = 1.0
```

[11]:
```
b = 2.0 * (-2.0) * (-1.0)
```

[12]:
```
c = (-2.0 * -2.0) - (-2.5 * 10.0)
```

[13]:
```
a, b, c
```

[13]:
```
(1.0, 4.0, 29.0)
```

[14]:
```
# rødder (-b +/- sqrt(b^2 - 4ac))/(2a) = -b/2a +/- sqrt(discriminant) / 2a
```

[15]:
```
diskriminant = b**2 - 4*a*c
```

[16]:
```
diskriminant
```

[16]:
```
-100.0
```

[17]:
```
-b/(2*a)
```

[17]:
```
-2.0
```

[18]:
```
# +/- i gange
```

[19]:
```
10 / (2*a)
```

[19]:
```
5.0
```

[20]:
```
lambda0 = -2.0 + 5.0j
lambda1 = -2.0 - 5.0j
```

[21]:
```
# find en egenvektor v0
```

[22]:
```
b_mat = a_mat - lambda0 * np.eye(2)
```

[23]:
```
b_mat
```

[23]:
```
array([[ 0. -5.j, -2.5+0.j],
       [10. +0.j,  0. -5.j]])
```

[24]:
```
b_mat[0, :] /= b_mat[0,0]
```

[25]:
```
b_mat
```

[25]:
```
array([[ 1.-0.j , -0.-0.5j],
       [10.+0.j ,  0.-5.j ]])
```

[26]:
```
b_mat[1, :] += - b_mat[1,0] * b_mat[0, :]
```

[27]:
```
b_mat
```

[27]:
```
array([[ 1.-0.j , -0.-0.5j],
       [ 0.+0.j ,  0.+0.j ]])
```

[ ]: