胡成成 41724260 通信1701

1.问题：求系统的零状态响应

$$\frac{d^2}{dt^2}y(t)+3\frac{d}{dt}y(t)+2y(t)=e^{-t}u(t),$$

$$y(0_-)=1,\quad y'(0_-)=2$$

2.数学方法理论求解

首先用高数知识求解非齐次常系数微分方程

当 $t \geq 0$ 时.

解齐次常系数微分方程：

令 $y(t)=e^{\lambda t}$ 代入 $\frac{d^2y(t)}{dt^2}+3\frac{dy(t)}{dt}+2y(t)=0$

得：$\lambda^2+3\lambda+2=0$

有单根 $\lambda=-2$ 和 $\lambda=-1$

∴有通解：$\tilde{y}=C_1e^{-2t}+C_2e^{-t}$

令特解 $y^*=A\cdot te^{-t}$ 代入原方程：

$$A(-2e^{-t}+te^{-t})+3A(e^{-t}-te^{-t})+2Ate^{-t}=e^{-t}$$

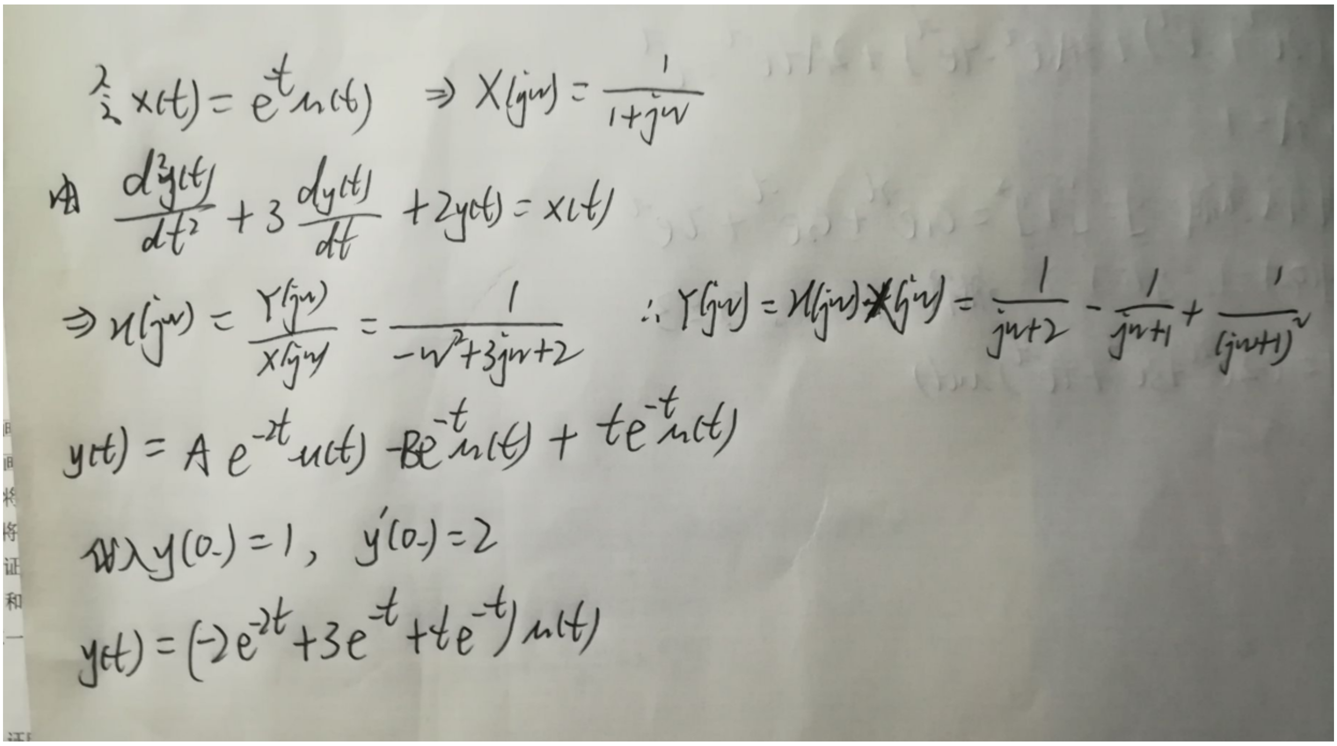
$\Rightarrow A=1$

∴非齐次通解：$y=\tilde{y}+y^*=C_1e^{-2t}+C_2e^{-t}+te^{-t}$

又 $y(0_-)=1$, $y'(0_-)=2$ 解得 $C_1=-2$, $C_2=3$

∴ $y=(-2e^{-2t}+3e^{-t}+te^{-t})u(t)$

利用信号与系统中冲激响应求解验证



令 $x(t)=e^{-t}u(t) \Rightarrow X(j\omega)=\frac{1}{1+j\omega}$

由 $\frac{d^2y(t)}{dt^2}+3\frac{dy(t)}{dt}+2y(t)=x(t)$

$\Rightarrow H(j\omega)=\frac{Y(j\omega)}{X(j\omega)}=\frac{1}{-\omega^2+3j\omega+2}$ $\therefore Y(j\omega)=H(j\omega)X(j\omega)=\frac{1}{j\omega+2}-\frac{1}{j\omega+1}+\frac{1}{(j\omega+1)^2}$

$y(t)=Ae^{-2t}u(t)-Be^{-t}u(t)+te^{-t}u(t)$

代入 $y(0_-)=1,\ y'(0_-)=2$

$y(t)=(-2e^{-2t}+3e^{-t}+te^{-t})u(t)$

3.利用MATLAB求解验证

```
y=dsolve('D2y+3*Dy+2*y=exp(t)','y(0)=1','Dy(0)=2','t')
```

得出结果：

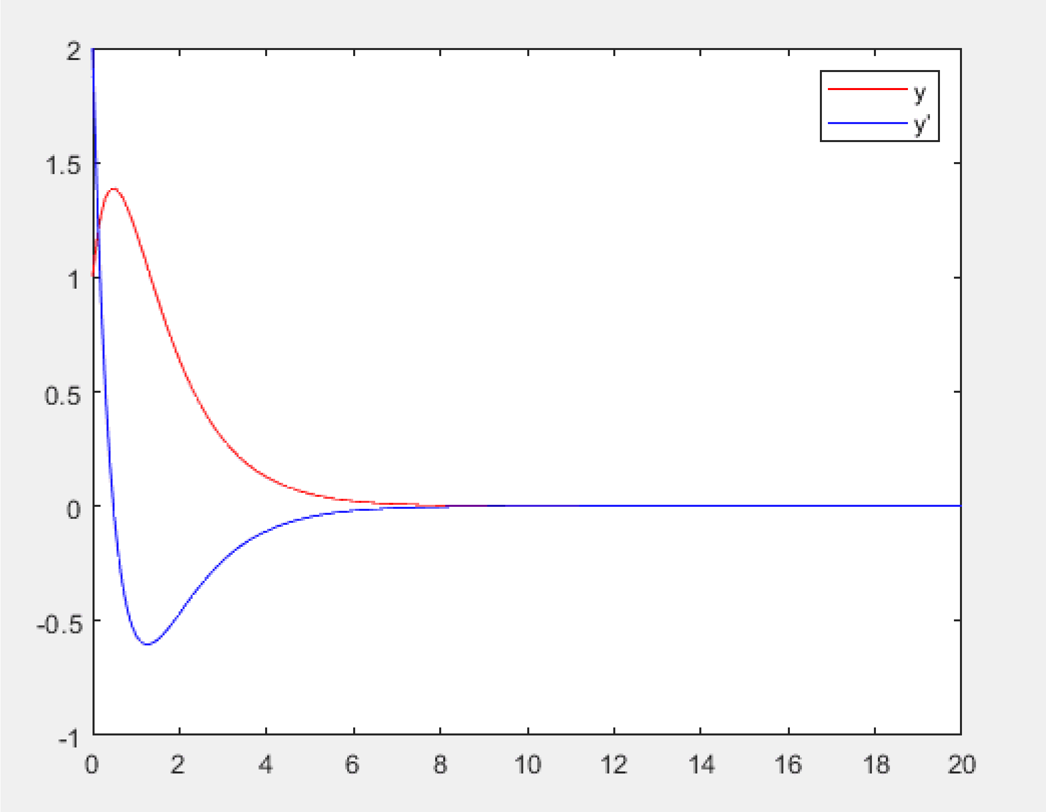
```
y =


                                (t - 2 exp(-t) + 3) exp(-t)
```

根据结果检验，上述手动计算与实际计算机得出结果一致。

```
t=0:0.1:20;
y = (t - 2 .*exp(-t) + 3) .*exp(-t);
y1=-exp(-t) .*(t - 2 .*exp(-t) + 3) + exp(-t).* (1 + 2.* exp(-t));
plot(t,y,'r-',t,y1,'b-'),legend('y','y'')
```

用MATLAB模拟图像结果：



4.利用Python求解该方程

通过上述计算，我们利用Python求解系统的零状态响应：

库函数准备

```
scipy
sympy
matplotlib
numpy
```

运行：jupyter

利用sympy进行符号解法

```
from sympy import *
init_printing()
#定义符号常量x 与 f(x)
x = Symbol('x')
f = symbols('f', cls=Function)
#用diffeq代表微分方程： f''(x) + 3f'(x) + 2f(x) = exp(-x)
diffeq = Eq(f(x).diff(x, x) + 3*f(x).diff(x) + 2*f(x), exp(-x))
#调用dsolve函数,返回一个Eq对象，hint控制精度
print(dsolve(diffeq, f(x)))
```

得到符号解，输出如下

```
Eq(f(x), (C1 + C2*exp(-x) + x)*exp(-x))
```

在带入初始松弛条件:

```
C1=-2
C2=3
```

结果与我们计算结果一致。

利用Numpy和Scipy进行数值解法

```
import matplotlib.pyplot as plt
from scipy import linspace,exp
from scipy.integrate import odeint, solve_bvp, solve_ivp
import numpy as np

'''
    为了兼容solve_ivp的参数形式，微分方程函数定义的参数顺序为(t,y)，因此使用odeint函数时需要使参数tfirst=True
    二阶甚至高阶微分方程组都可以变量替换成一阶方程组的形式，再调用相关函数进行求解，因此编写函数的时候，不同于一阶微分方程，二阶或者高阶微分方程返回的是低阶到高阶组成的方程组，

'''



def fvdp1(t,y):
    '''
    要把y看出一个向量，y = [dy0,dy1,dy2,...]分别表示y的n阶导，那么
    y[0]就是需要求解的函数，y[1]表示一阶导，y[2]表示二阶导，以此类推
    '''
    dy1 = y[1]      # y[1]=dy/dt，一阶导
    dy2 = -3 * y[1] - 2 * y[0] + exp( -1 * t )
    # y[0]是最初始，也就是需要求解的函数
    # 注意返回的顺序是[一阶导， 二阶导]，这就形成了一阶微分方程组
    return [dy1,dy2]

# 或者下面写法更加简单
def fvdp2(t,y):
    '''
    要把y看出一个向量，y = [dy0,dy1,dy2,...]分别表示y的n阶导
    对于二阶微分方程，肯定是由0阶和1阶函数组合而成的，所以下面把y看成向量的话，y0表示最初始的函数，也就是我们要求解的函数，y1表示一阶导，对于高阶微分方程也可以以此类推
    '''
    y0, y1 = y
    # y0是需要求解的函数，y1是一阶导
    # 返回的顺序是[一阶导， 二阶导]，这就形成了一阶微分方程组
    dydt = [y1, -3*y1-2*y0+exp(-t)]

    return dydt
```

```
def solve_second_order_ode():
    '''
    求解二阶ODE
    '''
    t2 = linspace(0,20,1000)
    tspan = (0, 20.0)
    y0 = [1.0, 2.0] # 初值条件
    # 初值[2,0]表示y(0)=2,y'(0)=0
    # 返回y，其中y[:,0]是y[0]的值，就是最终解，y[:,1]是y'(x)的值
    y = odeint(fvdp1, y0, t2, tfirst=True)

    y_ = solve_ivp(fvdp2, t_span=tspan, y0=y0, t_eval=t2)

    plt.subplot(211)
    y1, = plt.plot(t2,y[:,0],label='y')
    y1_1, = plt.plot(t2,y[:,1],label='y‘')
    plt.legend(handles=[y1,y1_1])

    plt.subplot(212)
    y2, = plt.plot(y_.t, y_.y[0,:],'g--',label='y(0)')
    y2_2, = plt.plot(y_.t, y_.y[1,:],'r-',label='y(1)')
    plt.legend(handles=[y2,y2_2])

    plt.show()

solve_second_order_ode()
```

结果：