數值方法 HW04

407420082 機械四 葉育賢

前言

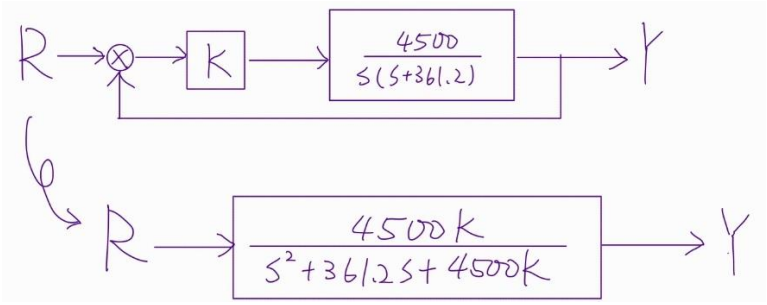
尤拉方法是一個雖然簡單，卻功能強大的演算法，能藉由簡單的數值方法，計算一階常微分方程。

本次實驗，我們要模擬控制系統的行為，並藉尤拉方法計算系統的響應。

研究方法

1. 方塊圖的化簡

   藉由方塊圖的化簡規則，經過串聯與負回授後，即可得到下圖的轉移函數。



2. 轉移函數，微分方程轉換

   藉由輸入輸出的關係式

$$\mathrm{T} = \frac{Y}{R} = \frac{4500K}{s^2 + 361.2s + 4500K}$$

   我們能夠以交叉相乘得到

$$(s^2 + 361.2s + 4500K)Y = 4500K4500K$$

   接著對兩邊同取反拉普拉斯轉換

$$y'' + 361.2y' + 4500Ky = 4500Ky$$

   便能得到時域的微分方程式

3. 狀態空間轉換

令

$y = x_1$，$y' = x_2$，$y'' = \dot{x_2}$

得

$\dot{x_1} = x_2$

$\dot{x_2} = -4500\mathrm{K}x_1 - 361.2x_2 + 4500\mathrm{K}r$

寫成矩陣的形式為

$$\begin{bmatrix}\dot{x_1}\\ \dot{x_2}\end{bmatrix} = \begin{bmatrix}0 & 1\\ -4500K & -361.2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 4500K\end{bmatrix}r$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

整理成簡化的形式

$\dot{X} = AX + Br$

$y = CX$

便可以藉由尤拉方法求解

4. 尤拉方法求解

帶入尤拉公式可得

$X_{n+1} = X_n + h(AX_n + Br)$

並以此求得結果 y

$y_n = [1 \quad 0]X_n$

程式概述

為了建立轉移函數，我將轉移函數做成一個物件，裡面包含所有轉移函數運作的方法。包含展示目前轉移函數的 show()，還有串聯並聯和負回授等等方塊圖化簡的工具，並有多項式相加和相乘的方法以實作以上工具。

```
class TransferFunction:
    def show(self):···

    def poly_mul(self,p1,p2):···

    def poly_plus(self,p1,p2):···

    def Series(self,G2):···

    def Parallel(self,G2):···

    def Nfeedback(self,H):···

    def __init__(self,*args):···
```

接著為了建立系統，我建構了 System 的物件，並有子物件 HWsystem 和 SIMsystem 和 SIMsystem_withC 繼承 System 物件。
因為考量到不同的系統有不同的方塊圖架構，因此以 polymorphism 的方式在子物件實作內容不同的 setTransferFunction。

```
class System:
    def __init__(self):...

    def TF_SS(self):...

    def setInput(self,t):...

    def f(self,X,t):...

    def EulerMethod(self):...

    def run(self):...
class HWsystem(System):
    def setTransferFunction(self):...

class SIMsystem(System):
    def setTransferFunction(self):...

    def f(self,X,t):...

class SIMsystem_withC(System):
    def setTransferFunction(self):...

    def f(self,X,t):...
```

如下圖，在不同的系統，有不同的建構轉移函數的方式。並且藉由 TransferFunction 物件裡的方法去建構轉移函數。

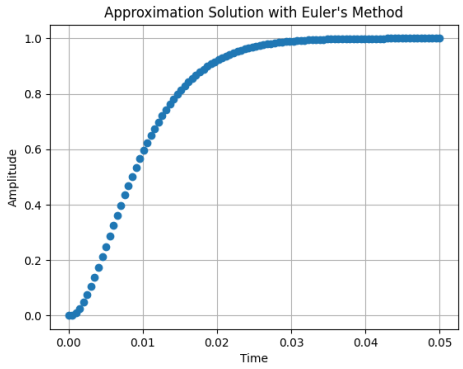
```
class HWsystem(System):
    def setTransferFunction(self):
        self.end=0.05
        print('Input G:')
        G=TransferFunction()
        H=TransferFunction(1,1)
        print('Input k')
        k=float(input())
        K=TransferFunction(k)
        G.Series(K)
        G.Nfeedback(H)
        self.T=G

class SIMsystem(System):
    def setTransferFunction(self):
        self.end=6
        print('Input G:')
        G=TransferFunction()
        self.T=G
```

最後是尤拉方法的程式實作。

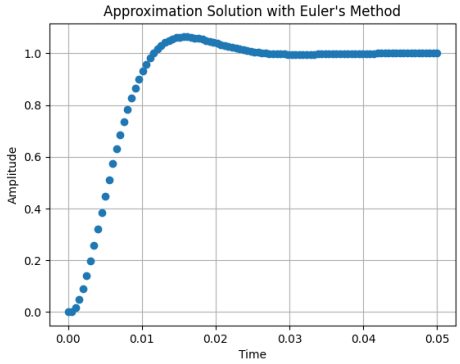
```
def EulerMethod(self):
    X0=np.zeros(self.num_x)
    start=0
    end=self.end
    n=100
    h=(end-start)/n
    t=np.linspace(start,end,n)
    y=np.zeros(n)
    X=np.zeros([n,2])
    X[0,:] = X0
    self.setInput(t)

    for i in range(1,n):
        X[i,:] = X[i-1,:]+h*self.f(X[i-1],self.U[i-1])
        y[i]=self.C.dot(X[i,:])

    plt.plot(t,y,'o')
    plt.xlabel("Time")
    plt.ylabel("Amplitude")
    plt.title("Approximation Solution with Euler's Method")
    plt.grid()
    plt.show()
```

研究結果

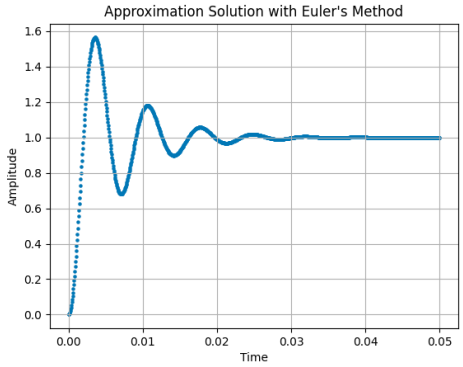
K=8



K=15

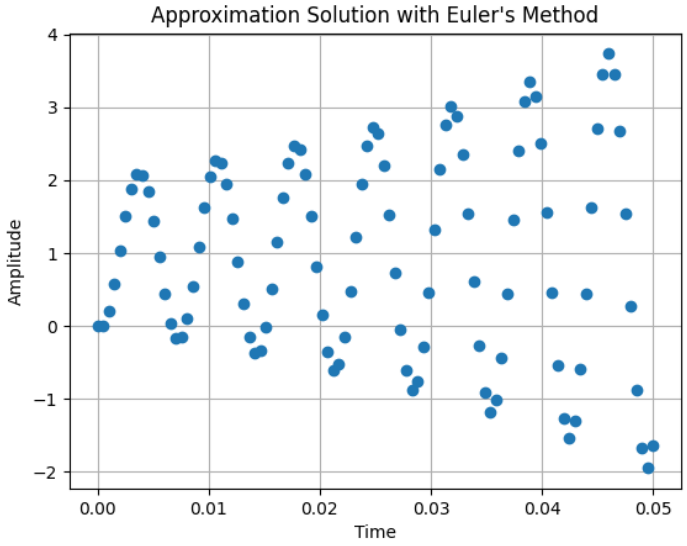


K=180



討論

經由這些模擬，我們能夠探討 K 值對系統的影響，並熟悉了由拉方法的使用。從數學與控制的觀點而言，K 值的大小會影響到阻尼係數，進而導致過阻尼，臨界阻尼，欠阻尼的影響。例如 K 值為 8 時即為過阻尼，系統沒有 overshoot，而 K 值為 180 時，系統為欠組尼，有很大的 overshoot。
此外，對於尤拉法的使用，若我們取用的步進太大，則近似的解會和精確解有誤差，容易造成發散的結果，如下圖，因此在使用尤拉法時，步進的選用也是非常重要的要點。



結論

經由以上實作，我們能感受到尤拉方法強大的功能，並能觀察到以數值方法模擬的效果非常成功。並且能藉由這個方式觀察 k 值對於控制系統的影響。

參考資料

https://github.com/EloneSampaio/Numerical-Methods-First_Order_DE/blob/master/euler_method.py