REPORT

13주차 과제

과 목 명	|	회전기기
담당교수	|	홍선기 교수님
학 과	|	시스템제어공학과
학 번	|	20210710
이 름	|	맹지우
제 출 일	|	2023.05.31.

유도전동기 시뮬레이션

맹지우_20210710
호서대학교 시스템제어공학과
(H.P: 010-9332-6526, E-mail : 20210710@vision.hoseo.edu)

1. 코드 구현

```
import numpy as np              # NumPy 라이브러리는 수학적인 연산
import math                     # math 라이브러리는 수학 함수 사용
import matplotlib.pyplot as plt  # 그래프를 그리기 위하여 Matplotlib의 pyplot 라이브러리 호출
```

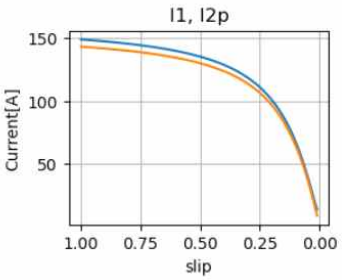
```
# 파라미터 값
V1 = 127                    # 입력 전압
R1 = 0.344                  # 1차측 저항
R2p = 0.147                 # 2차측 저항
X2p = 0.498                 # 2차측 리액턴스
X1 = 0.224                  # 1차측 리액턴스
Xm = 12.6                   # 자화 리액턴스
Prot = 50                   # 출력
Rc = V1 ** 2 / Prot         # 철손 저항
ns = 1200                   # 동기 속도
```

```
# 계산 범위 설정
s = np.arange(1,0, - 0.01)              # 슬립 값 범위 설정
nr = (1 - s) * ns                       # 회전 속도 설정
wr = 2 * math.pi / 60 * nr              # 각속도 설정

# 모터 파라미터
Z1 = R1 + 1j * X1                       # 1차측 임피던스
Zmc = 1j * Rc * Xm / (Rc + 1j * Xm)     # 유도기 회로 임피던스
Z2p = R2p / s + 1j * X2p                # 2차측 임피던스
Zred = Zmc * Z2p / (Zmc + Z2p)          # 감소된 임피던스
Ztot = Z1 + Zred                        # 전체 임피던스
```

```
# 값 계산
i1 = V1 / Ztot              # 1차측 전류
I1 = np.abs(i1)             # 1차측 전류 크기
e1 = V1-i1 * Z1             # 유도 전압
i2p = e1 / Z2p              # 2차측 전류
I2p = np.abs(i2p)           # 2차측 전류 크기
im = e1 / (1j * Xm)         # 자화 전류
Im = np.abs(im)             # 자화 전류 크기
ic = e1 / Rc                # 철손 전류
Ic = np.abs(ic)             # 철손 전류 크기
ipi = im + ic               # 자화 및 철손 전류
Ipi = np.abs(ipi)           # 자화 및 철손 전류 크기

P2i = 3 * np.real(I2p ** 2 * R2p / s)      # 2차측 입력 전력
Pout = (1 - s) * P2i                       # 유도기 출력 전력
Pin = 3 * np.real(V1 * np.conjugate(i1))   # 유도기 입력 전력
Eff = Pout / Pin                           # 효율

PF = np.zeros(100)          # 전력 인수 배열 초기화
Tout = np.zeros(100)        # 유도기 토크 배열 초기화

for ss in range(len(s)):
    PF[ss] = math.cos(math.atan2(np.imag(i1[ss]), np.real(i1[ss])))    # 전력 계산
    Tout[ss] = P2i[ss] / ns * 60 / (2 * math.pi)                       # 토크 계산
```

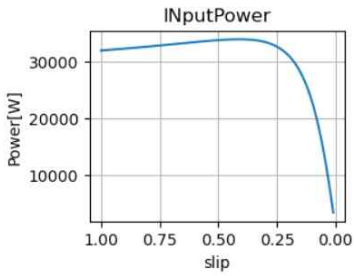
```
# 첫번째 그림 : 1차, 2차 전류
plt.subplot(224)          # 전체적인 크기 조절로, 숫자가 클수록 전체적인 크기가 작아짐. 즉, 111이면 221보다 크기가 커짐
plt.plot(s, I1)           # slip에 따른 1차 전류 그래프
plt.plot(s, I2p)          # slip에 따른 2차 전류 그래프
plt.gca().invert_xaxis()  # x축 반전
plt.grid(True)            # 그리드 표시
plt.title('I1, I2p')      # 그래프 제목 지정
plt.xlabel('slip')        # x축 레이블
plt.ylabel('Current[A]')  # y축 레이블
```

```
Text(0, 0.5, 'Current[A]')
```



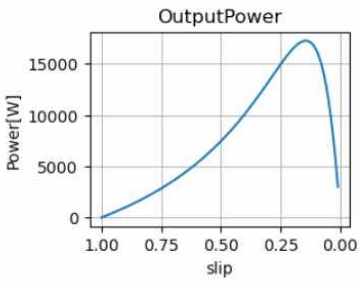

```
# 두번째 그림 - 유도기 입력(전력)
plt.subplot(224)          # 전체적인 크기 조절로, 숫자가 클수록 전체적인 크기가 작아짐. 즉, 111이면 221보다 크기가 커짐
plt.plot(s, Pin)          # slip에 따른 유도기 입력 전력 그래프
plt.gca().invert_xaxis()  # x축 반전
plt.grid(True)            # 그리드 표시
plt.title('INputPower')   # 그래프 제목
plt.xlabel('slip')        # x축 레이블
plt.ylabel('Power[W]')    # y축 레이블
```

```
Text(0, 0.5, 'Power[W]')
```



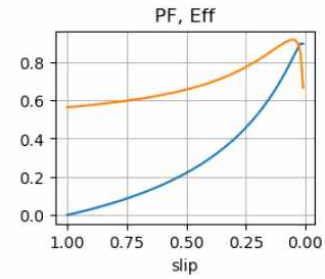

```
# 세번째 그림 - 유도기 출력(전력)
plt.subplot(224)          # 전체적인 크기 조절로, 숫자가 클수록 전체적인 크기가 작아짐. 즉, 111이면 221보다 크기가 커짐
plt.plot(s, Pout)         # slip에 따른 유도기 출력 전력 그래프
plt.gca().invert_xaxis()  # x축 반전
plt.grid(True)            # 그리드 표시
plt.title('OutputPower')  # 그래프 제목
plt.xlabel('slip')        # x축 레이블
plt.ylabel('Power[W]')    # y축 레이블
```

```
Text(0, 0.5, 'Power[W]')
```



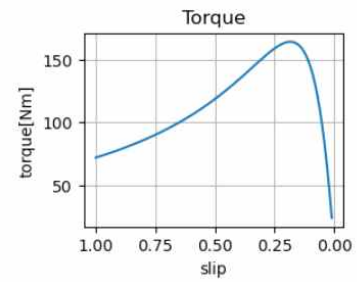
```
# 네번째 그림 - 효율
plt.subplot(224)          # 전체적인 크기 조절로, 숫자가 클수록 전체적인 크기가 작아짐. 즉, 111이면 221보다 크기가 커짐
plt.plot(s, Eff)          # slip에 따른 효율 그래프
plt.plot(s, PF)           # slip에 따른 전력 인수 그래프
plt.gca().invert_xaxis()  # x축 반전
plt.grid(True)            # 그리드 표시
plt.title('PF, Eff')      # 그래프 제목
plt.xlabel('slip')        # x축 레이블
plt.ylabel('')            # y축 레이블
```

```
Text(0, 0.5, '')
```



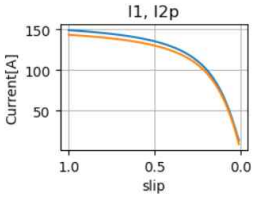

```
# 다섯번째 그림 - 유도기 토크
plt.subplot(224)          # 전체적인 크기 조절로, 숫자가 클수록 전체적인 크기가 작아짐. 즉, 111이면 221보다 크기가 커짐
plt.plot(s, Tout)         # slip에 따른 유도기 토크 그래프
plt.gca().invert_xaxis()  # x축 반전
plt.grid(True)            # 그리드 표시
plt.title('Torque')       # 그래프 제목
plt.xlabel('slip')        # x축 레이블
plt.ylabel('torque[Nm]')  # y축 레이블
```

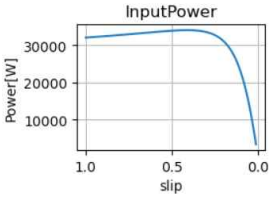
```
Text(0, 0.5, 'torque[Nm]')
```

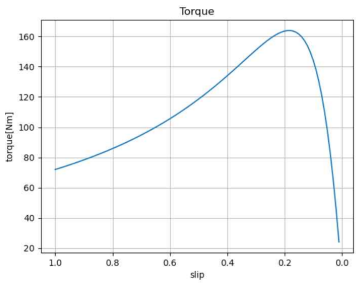


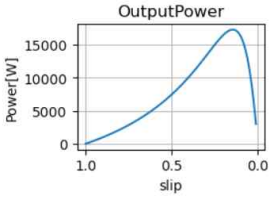
2. 회전자 저항값 변경

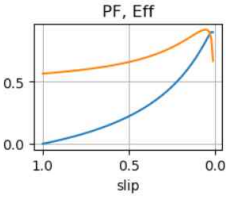
1) 회전자 저항값 기존



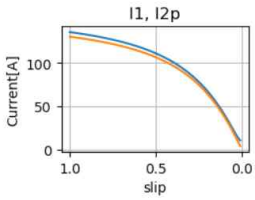


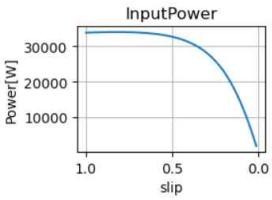


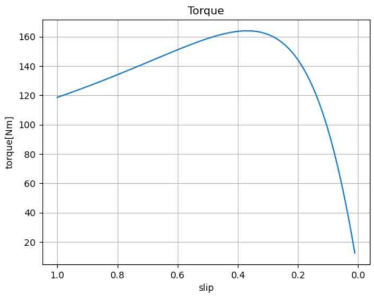


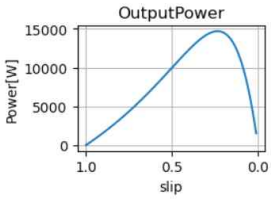


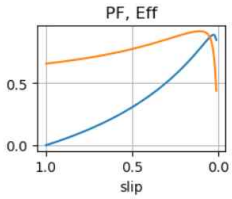
2) 회전자 저항값 2배



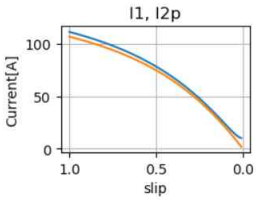


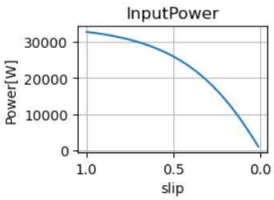


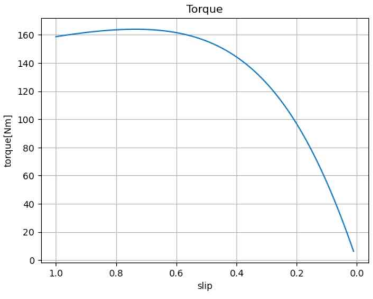


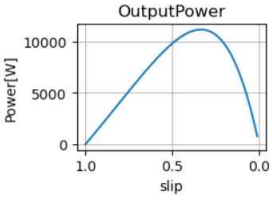


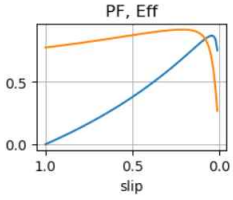
3) 회전자 저항값 4배











3. 결론 및 소감

회전자 저항을 2배로 늘릴 경우, 회전자 저항이 증가하면 회전자 측 전류가 감소하기 때문에 2차 측 전류 I2p는 감소하게 된다. 유도기 입력 전력 Pin은 회전자 전류 I2p의 제곱에 회전자 저항 R2p를 곱한 값이므로, I2p의 감소로 인해 Pin도 감소한다. 유도기 출력 전력 Pout은 P2i에 (1 − s)를 곱한 값이고, P2i는 I2p의 제곱에 R2p를 곱한 값이므로, I2p가 감소하게 되니 Pout도 감소한다. 효율 Eff은 Pout을 Pin으로 나눈 값으로, Pout의 감소로 인해 Eff도 감소한다. 유도기 토크 Tout는 P2i를 동기 속도 ns로 나눈 값이기에 P2i의 감소로 인해 Tout도 감소한다. 자화 전류 Im과 철손 전류 Ic는 유도 전압 E1을 각각 자화 리액턴스 Xm과 철손 저항 Rc로 나눈 값으로, 회전자 저항의 증가로 인해 전류의 크기가 감소하므로 Im과 Ic도 감소한다. 총 전류 Ipi는 자화 전류 Im과 철손 전류 Ic를 더한 값이니까 Ipi는 Im과 Ic보다 작아진다. 4배로 늘릴 경우, 2배와 똑같게 값들이 감소하며 2배보다는 더 크게 감소하는 형태를 볼 수 있다. 그래프를 살펴보면, 저항값이 증가하는 형태로 갈수록 전류 I1과 I2p의 그래프 형상은 굴곡이 있던 형태에서 점점 직선의 형태로 변화하게 된다. Pin 또한 굴곡졌던 형태에서 회전자 저항의 값이 증가할수록 직선의 형태로 변화하는 것을 알 수 있다. 반대로 Pout과 효율 Eff, 전력 인수 PF는 회전자 저항값이 증가할수록 곡선이 굴곡지는 형태로 변하고 있다. 마지막으로 토크 값을 살펴보면, 처음과 달리 저항값이 증가하면서 점차 완만한 형태로 변한다는 것을 볼 수 있다. 즉, 회전자 저항 값의 증가로 인해 회전자 전류, 입력 전력, 출력 전력, 효율, 유도기 토크, 자화 전류, 철손 전류, 총 전류 등이 모두 감소하게 됩니다.

이번 학기에 부전공에서 파이썬 언어를 사용하다 보니까 Anaconda를 많이 사용했었는데, 지금까지 텍스트 마이닝, 머신러닝 등 데이터 처리를 할 수 있겠다는 생각만 했지, 이렇게 전공 교과목에서 유도전동기 시뮬레이션을 할 수 있을 거라고는 생각조차 하지 못했다. 그런데 이번 기회에 이렇게 과제로서 새로운 것을 시도해볼 수 있는 기회가 생겨서 좋았고, 지금까지 데이터 처리만 생각하고 파이썬을 다루었었는데, 회전 기기를 공부하면서도 파이썬을 이용할 수 있게 되어 이 방향으로도 많이 사용해볼 것 같다.