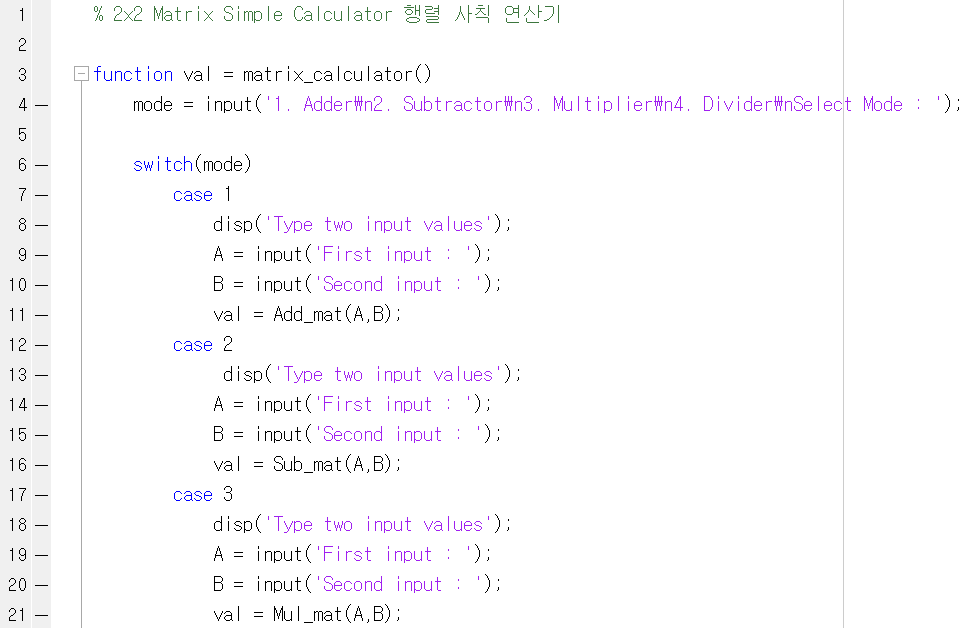


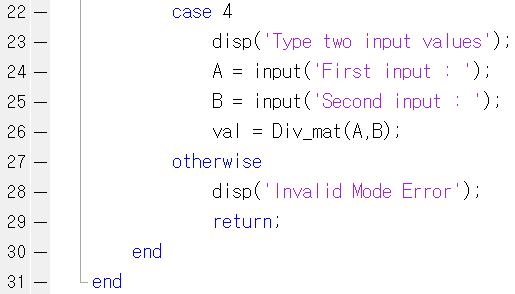
디지털 신호처리 설계

**10주차 MatLab 과제**

정보통신공학과 12171788 박지은

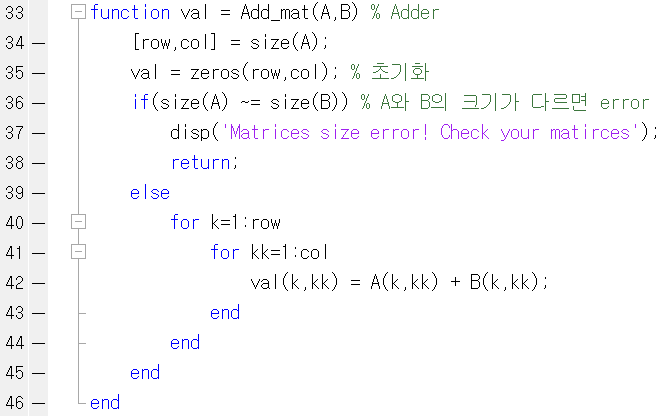
**1) 2x2 Matrix** **Calculator**

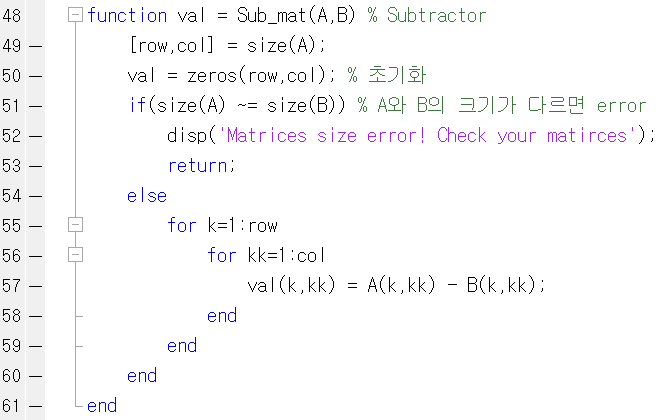




2x2의 기본 행렬 연산기를 만들어주었다. **Main function**으로 matrix\_calculator를 만들어주었고, **Sub function**으로는 덧셈, 뺄셈, 곱셈, 나눗셈에 대한 4가지 함수를 만들어 호출해주었다.

4가지 행렬 연산 함수는 **switch문**을 사용해 case별로 숫자 키 1, 2, 3, 4를 이용해 호출한다. 각 case별로 먼저 2가지 행렬을 입력 받는다. input을 사용해 사용자의 입력을 요청하고 return해주었다. 입력 값은 변수 A, B에 각각 저장하고, 각 연산의 sub function으로 연산하여 결과 값을 변수 val로 반환하였다.

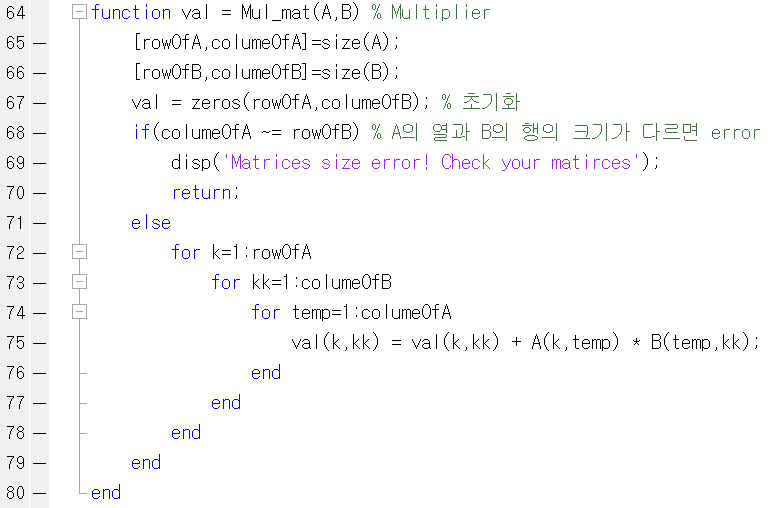


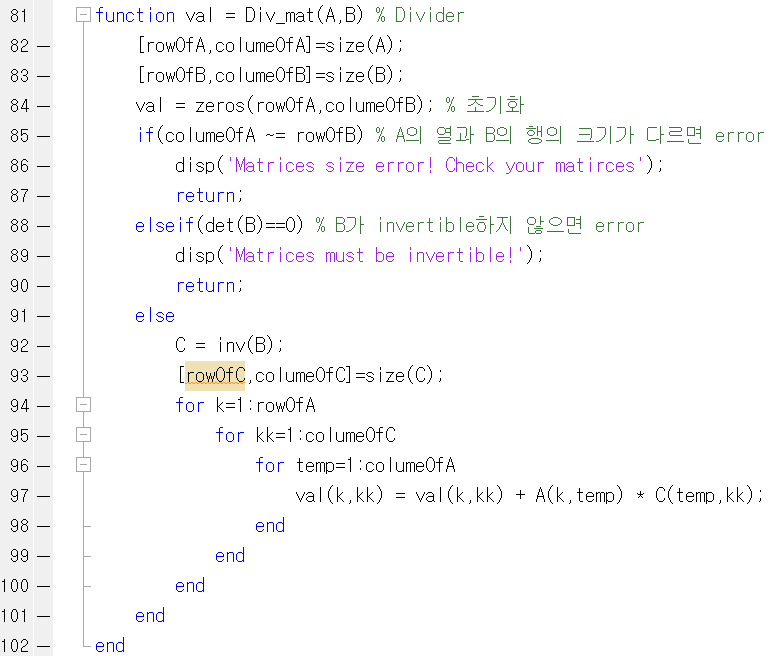


먼저 덧셈과 뺄셈에 관한 행렬 연산 함수이다. 행렬의 덧셈과 뺄셈은 두 행렬의 크기가 같아야 하므로 크기 비교가 필요하다.

제일 먼저 결과로 반환할 val값을 초기화해주었다. **size**함수를 사용해 row와 col값에 행렬 A의 size를 저장하고, val는 A의 크기와 동일하게 생성되어 0으로 초기화된다.

그 다음 **if문**을 사용하여 행렬의 크기가 다른 경우 에러 메시지가 출력되도록 하였다. 그 외에는 모두 연산이 가능하므로 **이중 for문**을 사용하여 같은 자리에 있는 원소들을 더하거나 뺀다. 첫번째 for문에는 1부터 차례대로 행이 증가하고, 두번째 for문에는 열이 증가하여 ( k , kk )의 위치에 맞게 원소들이 계산된다.



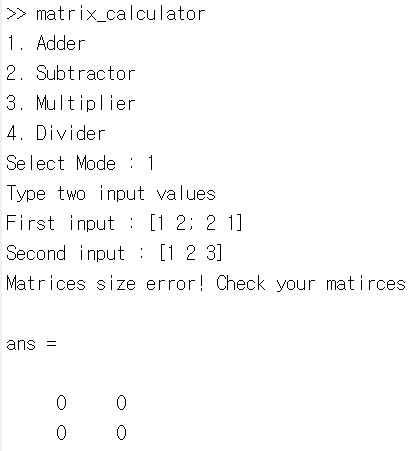
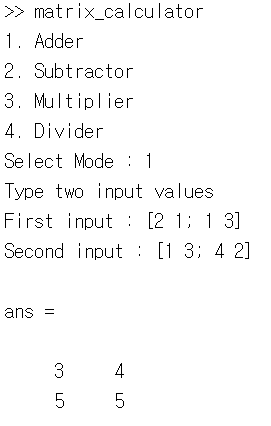


곱셈과 나눗셈에 관한 연산 함수도 마찬가지의 알고리즘을 사용하여 계산해주었고, 경우에 맞게 조건문을 추가해주었다. 행렬의 곱셈과 나눗셈은 앞에 오는 행렬의 열의 크기와 뒤에 오는 행렬의 행의 크기가 같아야 한다. 먼저 **rowOfA,B**와 **columeOfA,B**라는 변수를 만들어 A와 B의 행과 열의 크기를 각각 저장해준다. 또한 A의 열의 크기, B의 행의 크기가 각각 val의 행과 열의 크기가 되므로 그에 맞게 생성하여 초기화해준다. 따라서 if문으로 A의 열의 크기와 B의 행의 크기가 같지 않으면 에러를 출력한다.

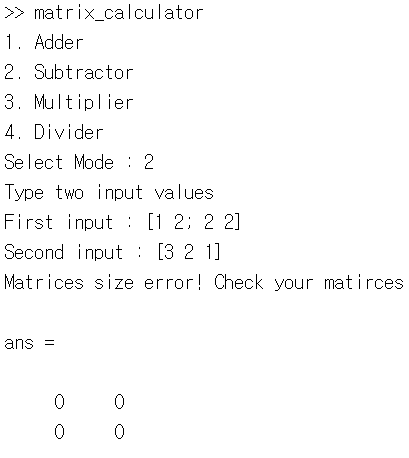
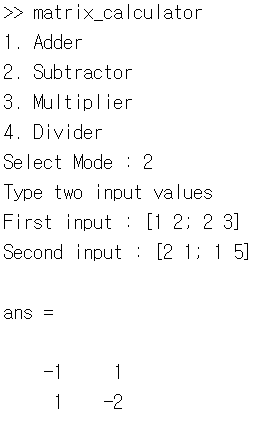
또한 행렬의 곱셈 연산은 덧셈과 뺄셈처럼 같은 자리에 있는 원소끼리의 곱셈이 아니기 때문에 삼중 for문을 사용하였다. 따라서 A의 첫번째 행과 B의 첫번째 열을 곱한 원소가 val의 첫번째 열의 원소가 된다.

나눗셈 연산은 행렬 B의 역행렬과의 곱셈 연산을 하는 것이기 때문에 행렬 B가 invertible해야 한다. 행렬이 invertible할 조건은 행렬의 determinant가 0이 아닌 것이다. 따라서 elseif로 det(B)가 0일 때 에러를 출력한다. 행렬 B의 역행렬이 존재할 때, inv(B)를 행렬 C에 저장하여 A / B^-1의 연산이 A \* C로 바뀌어서 곱셈일 때와 마찬가지로 계산된다.

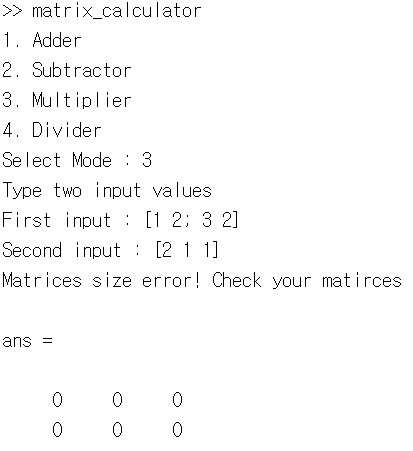
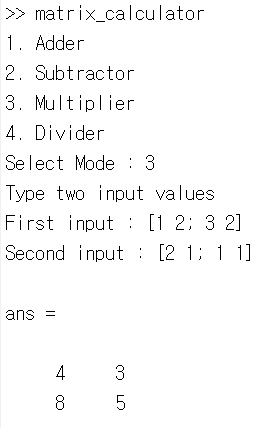
<Adder>



<Subtractor>

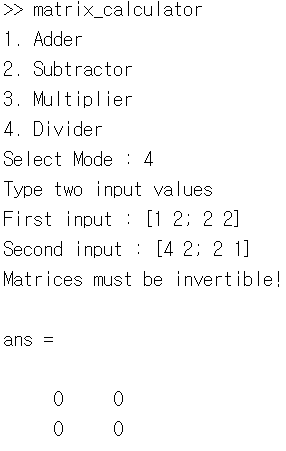
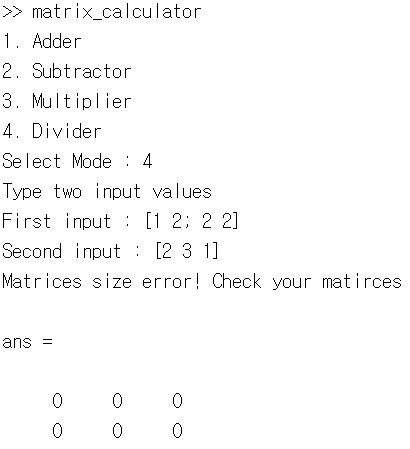
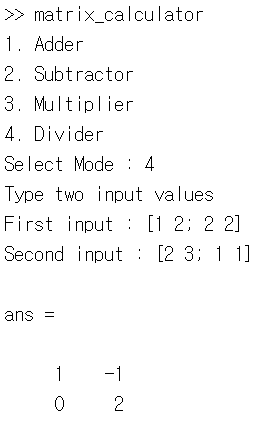


<Multiplier>



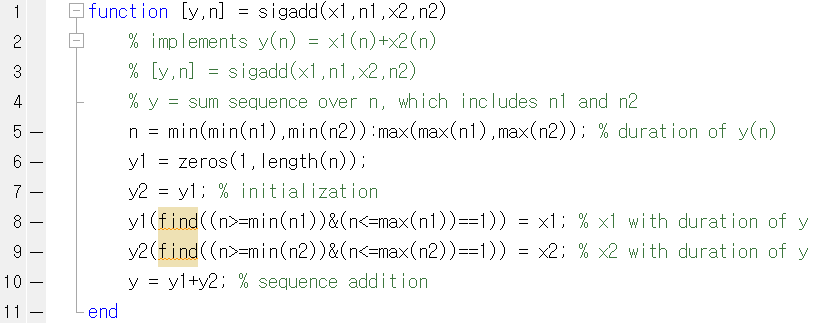
덧셈, 뺄셈의 연산 결과 2x2의 행렬끼리의 연산은 제대로 되지만, 2x2와 3x1처럼 크기가 다른 경우는 error가 출력된다. 또한 곱셈도 마찬가지로 앞 행렬의 열과 뒷 행렬의 행의 크기가 다르면 error를 출력한다.

<Divider>

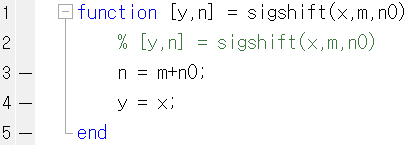


나눗셈의 경우는 곱셈과 마찬가지로 앞 행렬의 열과 뒷 행렬의 행의 크기가 다르면 error를 출력한다. 또한 추가적으로 뒷 행렬이 invertible하지 않을 때(det=0) error가 출력되는 것을 확인할 수 있다.

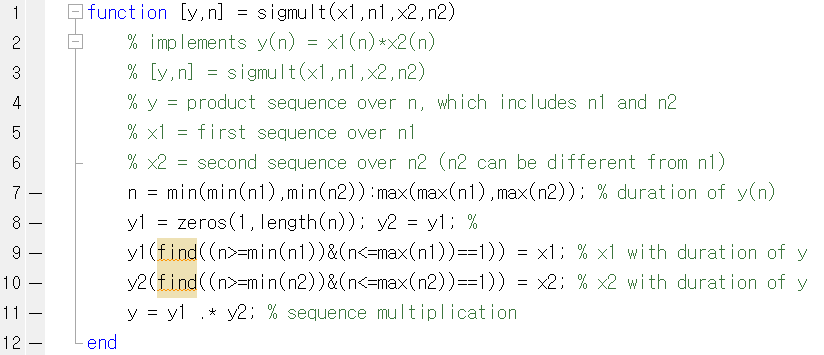
**2) Discrete-time Signals #Problem 1**



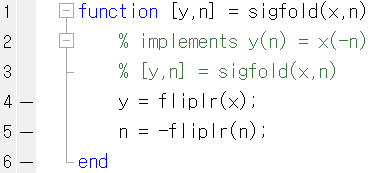
**덧셈 (y(n) = x1(n)+x2(n))**에 대한 함수이다. 함수를 받고 n의 구간 내에서 두 함수를 더해준다.



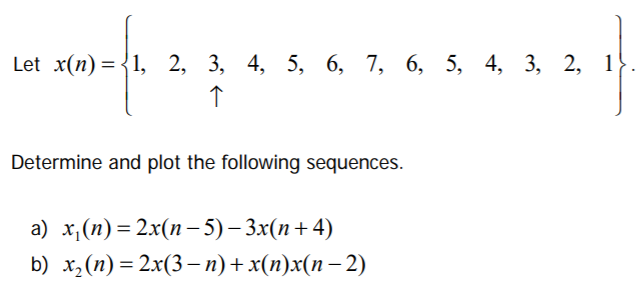
**평행이동 (y(n) = x(n-m))**에 대한 함수이다. 입력 값만큼 지연하여 출력한다.

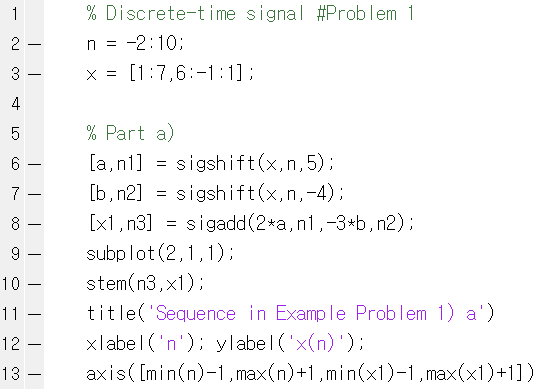


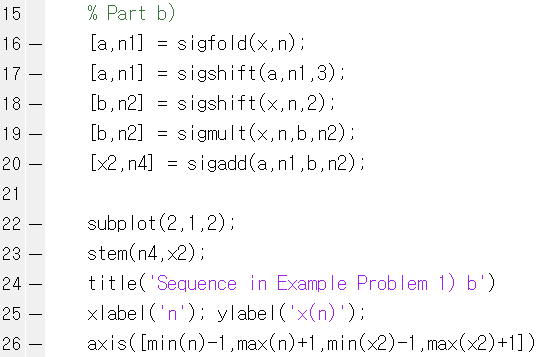
**곱셈 (y(n) = x1(n)\*x2(n))**에 대한 함수이다. 함수를 받고 n의 구간 내에서 두 함수를 곱해준다.



**대칭이동 (y(n) = x(-n))**에 대한 함수이다. n에 대하여 부호를 대칭시킨다.

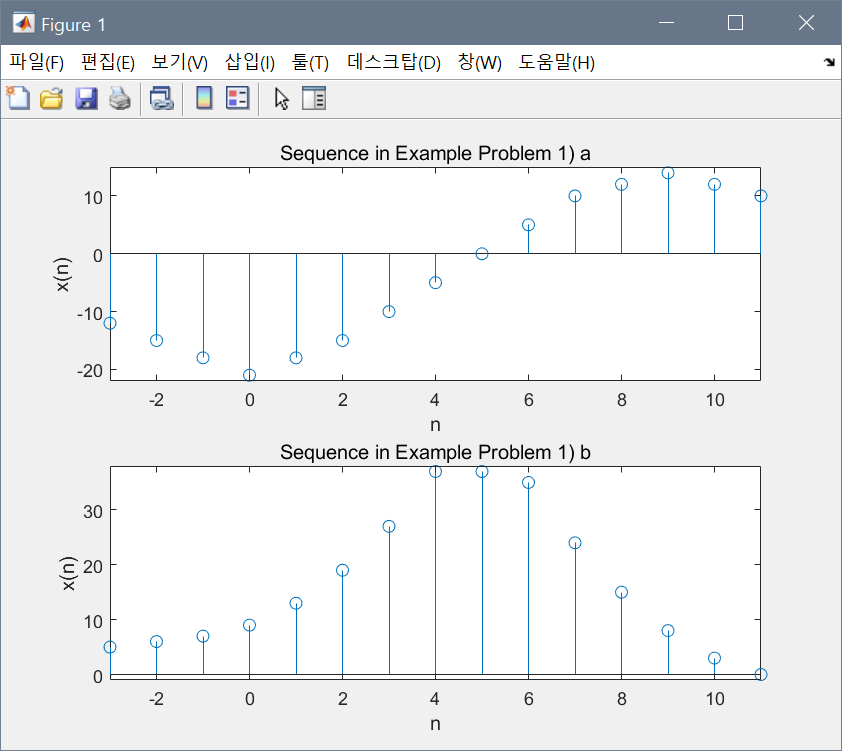






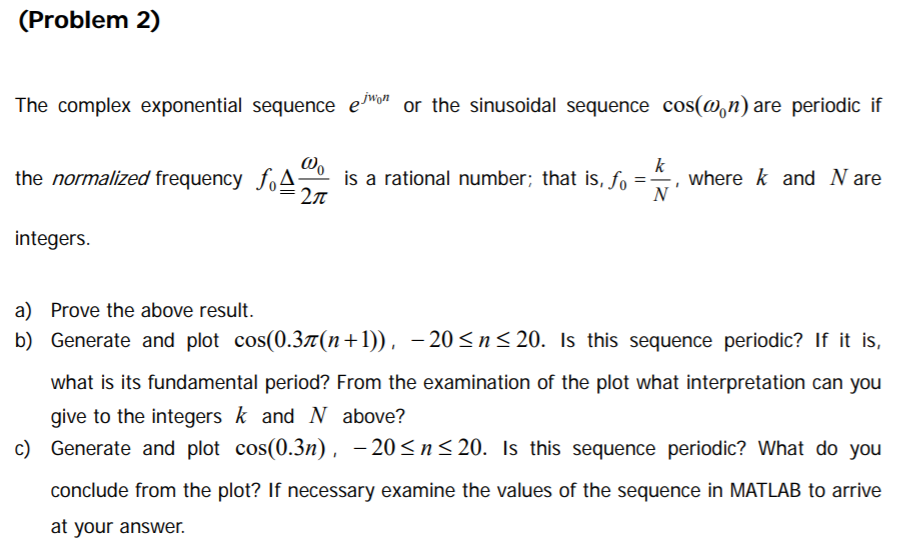
n = 0일 때, x(n) = 3이기 때문에 n = -2:10으로, x = [1:7, 6:1:1]로 설정해주었다.

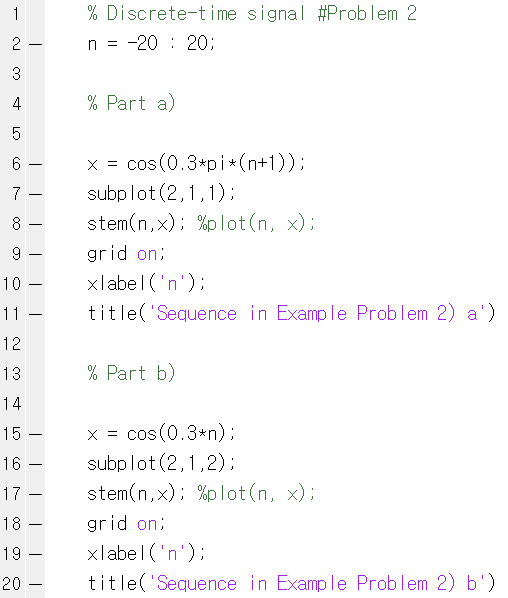
1. x1(n)은 x(n)을 5만큼 shift하고 -4만큼 shift한 것의 합이기 때문에 **sigshift**함수와 **sigadd**함수를 사용하여 표현해주었다.
2. x2(n)은 x(n)을 fold한 다음 shift한 것들의 곱과 합이기 때문에 **sigfold, sigshift, sigadd, sigmult**함수를 모두 사용하여 표현해주었다.



축의 범위는 x(n)과 n이 최대, 최소일 때로 정해주었기 때문에 그래프가 딱 맞게 출력되었다. 또한 discrete한 signal을 표현해주기 위해 **stem**함수를 이용하였다.

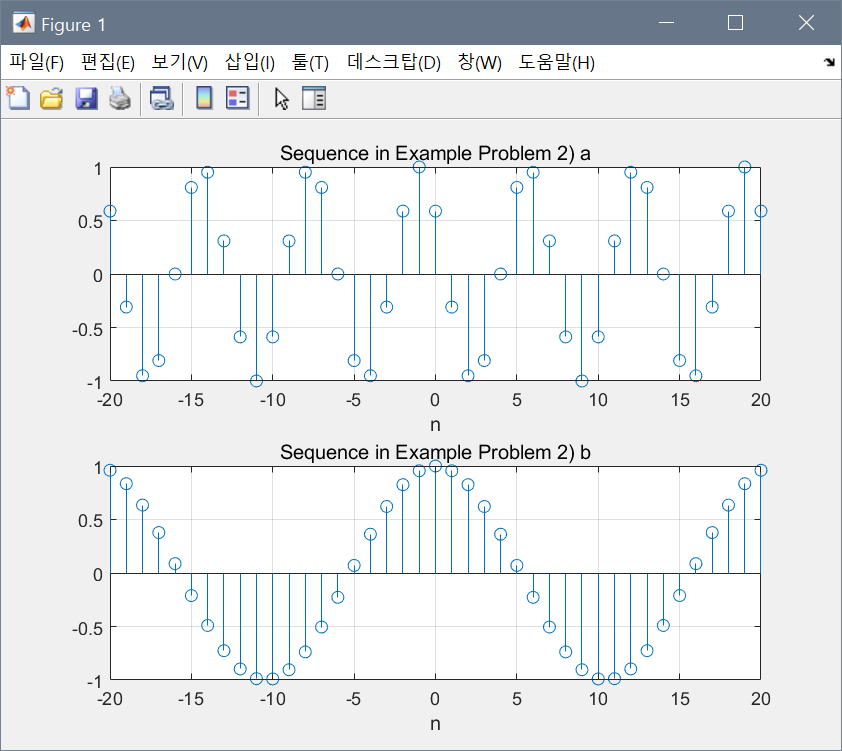
**3) Discrete-time Signals #Problem 2**





n = -20:20으로 설정해주었다.

1. cos(0.3π (n +1)) , − 20 ≤ n ≤ 20
2. cos(0.3n) , − 20 ≤ n ≤ 20



a) 함수의 주기가 3/5이기 때문에 정수(기약분수)로 표현이 가능하다. 따라서 이 함수는 주기 함수이다.

b) 함수의 주기가 3/5π이기 때문에 정수로 표현이 불가능하다. 따라서 이 함수는 주기 함수가 아니다. Discrete한 값들을 자세히 보면 x = -19일 때 x(n) = 0.8347이고, x = 1일 때 x(n) = 0.9553이기 때문에 주기 함수가 아닌 것을 확인할 수 있다.

**고찰)**

이번 시간에는 Matlab으로 행렬의 기본 연산과 discrete signal의 표현에 대해 실습하였다. 행렬에 대한 기본적인 함수와 matlab에서 사용하는 function과 subfunction을 공부하여 2x2의 간단한 행렬에 대한 사칙 연산기를 만들 수 있었다. 또한 discrete-time signal에 대한 sequence와 그 표현 방법을 익힐 수 있었다.