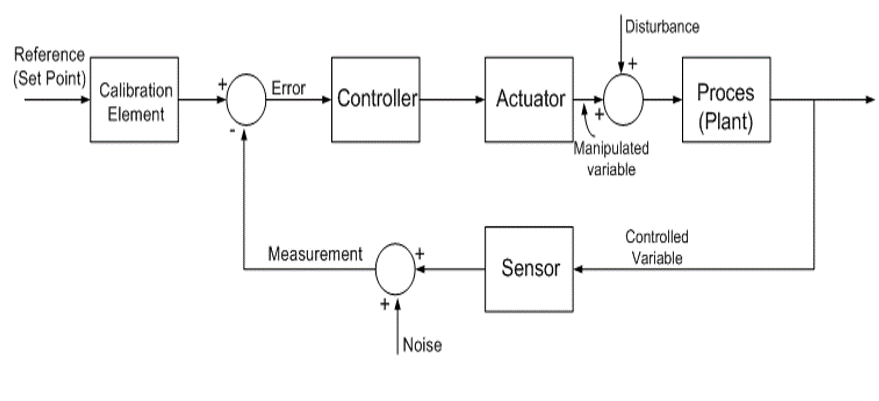
* 1. **PID 제어 시스템을 이용한 공정제어**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | PID 제어 시스템의 이해 |
| [방법] | PID 제어 시스템의 조율 |
| [응용] | 딥러닝을 적용한 PID 제어 시스템의 조율 |
| [요약] | * PID 제어 시스템의 이해 * 기존 PID 제어 시스템의 조율 방법 * 딥러닝을 이용한 PID 제어 시스템의 조율 |

### **[이론] PID 제어 시스템**

### **공정제어의 개요**

**공정제어**(process control)란 온도, 압력, 유량 등 공정의 상태를 측정하여, 해당 상태가 우리가 원하는 조건으로 도달될 수 있도록 공정의 변수를 조작하는 행위를 일컫는다. 제어시스템은 제어기를 중심으로, 제어기가 조절하는 조작 변수(manipulated variable; MV), 제어기가 제어하고자 대상이 되는 제어 변수(controlled variable; CV), 도달하고자 하는 설정 변수(목표값, set variable; SV), 그리고 제어 변수에 영향을 미치나 제어기가 조작할 수 없는 외란 변수(disturbance variable; DV)로 구성된다. 그림 1에 보인 바와 같이, 제어기는 센서를 이용하여 공정의 상태인 제어 변수 값을 측정한 후, 이를 설정 변수와 비교하여, 그 차이가 최소화되도록 조작 변수의 값을 결정한다.



**그림 1. 피드백 제어 시스템의 블록 흐름도 예시**

### **PID 제어의 이해**

PID 제어는 산업계에서 가장 널리 이용되는 제어기법으로, 계산량이 적고 그 적용이 쉽다는 장점이 있다. PID 제어는 P 제어(proportional control; 비례 제어), I 제어(integral control; 적분 제어), 그리고 D 제어(derivative control; 미분 제어)로 구성된다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

**그림 2. PID 제어 시스템의 블록 흐름도**

**P 제어**는 현 시점의 제어오차에 비례하는 제어동작을 취한다. P 제어는 계산이 간단하여 응답이 빠르다는 장점이 있으나 잔류오차(offset)가 발생한다는 단점이 있다. 여기에서, 잔류 오차란 시간의 흐름에 따라 설정 변수와 제어 변수의 차이가 감소하지 않고 유지된 채 지속되는 오차를 가리킨다.

**I 제어**는 시간의 흐름에 따라 계속해서 제어오차를 누적하고, 누적된 제어오차에 비례하여 제어동작을 산출한다. 이러한 제어오차의 누적과 이를 통한 제어동작은, 제어오차가 0이 될 때까지 지속되므로, P 제어의 단점인 잔류 오차 문제를 해결할 수 있다. 다만, I 제어의 경우, 충분한 제어오차가 누적되어야 큰 제어동작을 산출하므로, 제어 응답이 느리다는 단점을 지닌다.

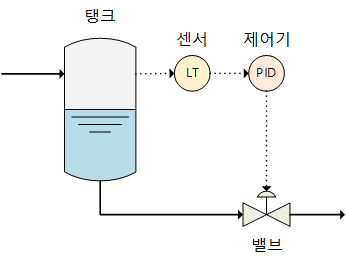
**D 제어**는 제어오차의 기울기에 비례하는 제어동작을 산출한다. D 제어는 P 제어와 마찬가지로 잔류오차를 발생시키며, 미분(즉, 제어오차의 기울기)에 기반하여 제어 동작을 산출하므로, 높은 주파수를 갖는 노이즈에 민감하다는 단점을 지닌다.일반적으로, D 제어는 그 특성상, P 제어나 I 제어와 달리 단독으로 사용하지 않는다.

### **인공지능 기반의 PID 제어**

PID 제어는 비례 이득, 적분 시간, 미분 시간으로 구성된 조율 변수에 따라 그 성능이 결정된다. 오랜 기간 동안 이러한 PID 제어의 최적 조율 변수 값을 결정하기 위한 많은 방법들이 제안되었으나, 그 적용 과정이 복잡하고 많은 시행착오 과정을 동반하여 널리 적용되지 않고 있다. 본 장에서는 인공지능 학습을 통해 PID 제어 시스템을 조율하고자 한다.

### **[문제]**

**수조에 찬 물의 수위를 PID 제어 시스템을 통해 제어하고자 한다. 관련 데이터는 ‘matplotlib.pyplot’를 불러와 이용할 때, PID 제어 시스템에 딥러닝을 적용해 PID 제어 시스템의 KP, KI, KD를 조율하라. 반복 계산 횟수는 100,000번으로 하고, 절댓값 오차를 사용한다. 초기의 조건은 KP=1.0, KI=0.5, KD=0.5이고, 오차축적 횟수는 100, 학습률(learning rate)은 0.001, 초기 오차는 10이다.**



### **[방법] PID 제어 시스템의 조율**

#### PID 제어 시스템의 조율 방법 중 가장 대표적인 지글러-니콜스 조율(Ziegler-Nichols tuning) 방법에 대해 설명하여라.

1. 지글러-니콜스 조율은 1942년 Ziegler와 Nichols에 의해 제안된 방법 중 하나로, 경계안정상태의 특성에 근거한 제어기 조율 방법 중 하나이다. 지글러-니콜스 조율은 먼저 I 제어와 D 제어를 끄고 P 제어기만 켠 상태 또는 KI에 무한에 가까운 수를 넣고, KD에는 0을 입력한 상태에서 진행한다. 이 상태에서 그림 3과 같이 공정 출력에 지속적인 진동 현상이 나타날 때까지 KC를 작은 값에서부터 서서히 증가시킨다.

텍스트, 하늘, 다른이(가) 표시된 사진

자동 생성된 설명

**그림 3. 공정 출력의 지속적인 진동 현상**

KC가 작을 때에는 목표값에 수렴하다가 KC가 점점 커지면서 오차에 민감해지기 때문에 그림 3과 같은 규칙적인 진동이 관측되기 시작한다. 만약 이보다 KC가 더 커지면 공정 출력은 발산한다. 규칙적인 진동의 주기를 Pu(ultimate period)라고 하며, 이때의 KC는 KCu(ultimate controller gain)이라고 한다. 마지막으로, KCu을 표 1에 제시된 표에 대입하여 제어 시스템을 조율한다.

테이블이(가) 표시된 사진

자동 생성된 설명

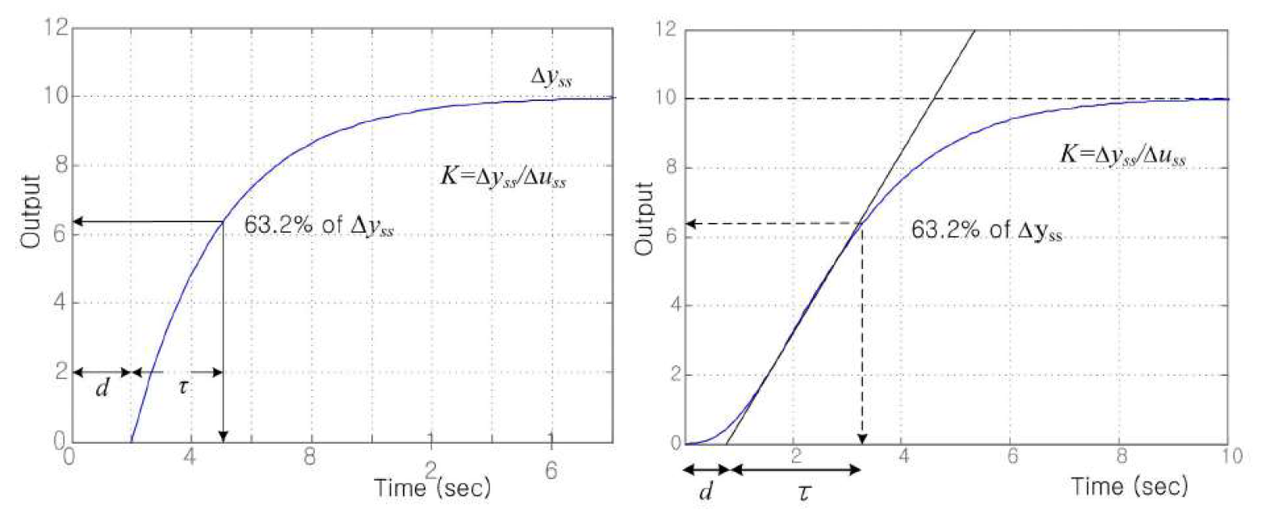
**표 1. 제어 시스템의 지글러-니콜스 조율 값**

#### 또 다른 PID 제어 시스템 조율 방법인 1차 시간지연(first-order plus dead time) 모델의 전제 조건은 무엇이며, 조율하는 방법을 설명하여라.

1. 1차 시간지연 모델은 식 (4)와 같이 나타낼 수 있다. 식 (4)는 시간 t를 라플라스 변환하여 나타낸 식이다.

식 (4)

식 (4)에서 d와 τ는 그림 4를 통해 알 수 있다. d는 지연 시간(dead time)으로, 제어기를 통해 제어출력이 나올 때까지의 시간을 의미한다. 실제 공정에서는 대부분 2차 함수와 같은 모습이 나타나는데, 이 경우 변곡점에서 접선을 그린 후 0 초부터x축과 만나는 점까지의 시간을 지연 시간으로 정의한다. τ는 시정수를 의미하며 제어 출력이 정상 상태의 제어 출력의 63.2%인 지점의 시간을 말한다.

**그림 4. (a) 1차 함수와 (b) 2차 함수의 반응응답곡선**

1차 시간지연모델의 전제 조건은 ‘0.1<d/τ<1.0’이다. d/τ<0.1일 때에는 지연 시간이 무시되는 1차 공정에 근접한다. 1차 공정의 경우 KC를 크게 할수록 제어가 잘된다. 반면, d/τ>1.0에서는 지연시간이 매우 커 일반적인 PID 제어 시스템으로는 제어성능에 한계가 생기므로 dead time compensator(ex. Smith predictor)가 필요하다.

|  |  |  |  |
| --- | --- | --- | --- |
| 제어기 | KC | KI | KD |
| P | (τ/KPd) | - | - |
| PI | 0.9(τ/KPd) | 3.33d | - |
| PID | 1.2(τ/KPd) | 2.0d | 0.5d |

**표 2. 제어 시스템의 1차 시간지연 모델 조율 값**

마지막으로, 1차 시간지연 모델로부터 지글러-니콜스 조율에 필요한 Kcu와 Pu를 구하여 지글러-니콜스 조율을 적용한다. Kcu와 Pu를 쉽게 계산하기 위해 표2와 같이 KP, d, τ값으로도 나타낼 수 있다.

### **[응용] 딥러닝을 통한 이산화탄소 배출량 예측**

#### ‘matplotlib.pyplot’을 불러오고, 탱크의 수위를 PID제어기를 정의하여라.

1. 먼저, 그림 5와 같이 윈도우 명령 프롬프트에 pip을 활용해 matplotlib을 컴퓨터에 설치한다. 그리고 아래의 절차에 따라 탱크의 수위와 PID 제어 시스템을를 정의한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| import matplotlib.pyplot as plt   #데이터베이스 불러오기 |

**그림 5.** matplotlib.pyplot **설치하기**

명령 프롬프트에 ‘python -m pip install -U matplotlib’를 입력하여 matplotlib.pyplot를 설치한다.

먼저, 파이썬에 설치한 matplotlib.pyplot를 불러온다.

|  |
| --- |
| class Liquid:      def \_\_init\_\_(self, error):          self.current\_error = error          self.last\_action = 0      def take\_action(self, action):          self.current\_error += 0.1 \* action          self.last\_action = action  class PID:      err\_sum = 0      old\_err = 0      def pid(self, current, goal, kp, ki, kd):          err = goal - current          self.err\_sum += err          delta\_err = err - self.old\_err          self.old\_err = err          return kp\*err + ki\*self.err\_sum + kd\*delta\_err |

위의 코드와 같이 탱크의 수위는 목표값과 실제 탱크의 수위만큼의 차인 error를 가져와 제어동작을 취함으로써 수위를 조절한다. 이때, ‘current’는 현재 탱크의 수위, ‘goal’은 제어기의 목표값을 나타내며, kp, ki, kd는 각 제어기의 조율 인자(KP, KI, KD)를 의미한다.

#### PID 제어 시스템을 조율하기 위한 함수를 정의해라

1. PID 제어 시스템 조율에 필요한 함수는 아래의 절차대로 정의한다.

|  |
| --- |
| class Derivative:      def \_\_init\_\_(self):          self.last\_x = 0          self.last\_y = 0      def get\_gradient(self, x, y):          d = (y - self.last\_y) / (x - self.last\_x)          self.last\_x = x          self.last\_y = y          return d |

위의 코드와 같이 미분 클래스를 정의한다. 미분 클래스는 기울기를 통해 조율 인자를 최적화할 때 사용된다.

|  |
| --- |
| class Train:      kp = 1.0; ki = 0.5; kd = 0.5      goal = 0      episode\_length = 100      learning\_rate = 0.001      def \_\_init\_\_(self):          self.dp = Derivative()          self.di = Derivative()          self.dd = Derivative()          self.step = 0          self.last\_loss = 0 |

위의 코드는 제어 시스템의 조율을 위해 해당 모델을 훈련시키기 위해 훈련 클래스를 정의하는 코드이다. 가장 먼저 문제에 주어진 초기 조건들을 입력하고 변수들을 정의한다. 목표값과 실제 탱크의 수위의 차이는 0이 되는 것이 이상적이므로 goal은 0으로 설정하고, 오차축적 횟수(episode\_length)는 100, 학습률은 0.001로 설정한다.

|  |
| --- |
| def abs\_mean(self, list):          sum = 0          for i in list:              sum += abs(i)          return sum / len(list)      def loss(self):          liquid = Liquid(10)          pid = PID()          error = []          for i in range(self.episode\_length):              error.append(liquid.current\_error)              liquid.take\_action(pid.pid(liquid.current\_error, self.goal, self.kp, self.ki, self.kd))          return self.abs\_mean(error)      def optimize(self):          self.kp = self.kp - self.learning\_rate \* self.dp.get\_gradient(self.kp, self.loss())          self.ki = self.ki - self.learning\_rate \* self.di.get\_gradient(self.ki, self.loss())          self.kd = self.kd - self.learning\_rate \* self.dd.get\_gradient(self.kd, self.loss())          self.last\_loss = self.dd.last\_y          print("step={}, kp={}, ki={}, kd={}, loss={}".format(self.step, self.kp, self.ki, self.kd, self.last\_loss))          self.step += 1 |

위의 코드는 훈련 클래스에서 정의된 함수들이다. ‘abs\_mean’은 문제에서 정의된 것처럼 오차를 구할 때 사용할 절댓값 평균 함수를 정의한다. ‘loss’는 for문을 통한 반복 계산을 통해 탱크의 수위를 보정하며 실제 탱크의 수위와 목표값의 차이(error)를 축적하는 함수이다. ‘loss’에서 구해진 error는 abs\_mean 함수를 통해 절댓값 평균으로 출력된다. 마지막으로, ‘optimize’는 앞서 정의한 함수들을 통해 최종적으로 조율 인자들을 최적화함으로써 PID 제어 시스템을 조율한다. 또한 이때, 뒤쪽의 그림 6과 같이 반복 계산을 하는 각 단계의 값들을 볼 수 있도록 설정했다.

#### 앞서 정의한 함수들을 통해 KP, KI, KD 값을 최적화하고 이를 도식화하라.

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':      loss = []      kp = []      ki = []      kd = []      train = Train()      for j in range(100000):          train.optimize()          loss.append(train.last\_loss)          kp.append(train.kp)          ki.append(train.ki)          kd.append(train.kd) |

1. 아래와 같은 절차를 통해 조율 인자들을 최적화할 수 있다. 또한 그 결과는 그림 6과 7을 통해 각각 확인할 수 있다.

먼저, 위의 코드와 같이 계산된 데이터를 담을 배열을 생성하고, Train의 인스턴스를 저장할 변수(train)을 생성한다. 또한 앞서 정의한 ‘optimize’함수를 사용해 조율 인자들을 최적화한다. 이때 반복 횟수는 문제에서 언급한 것처럼 100000회로 설정한다.

텍스트이(가) 표시된 사진

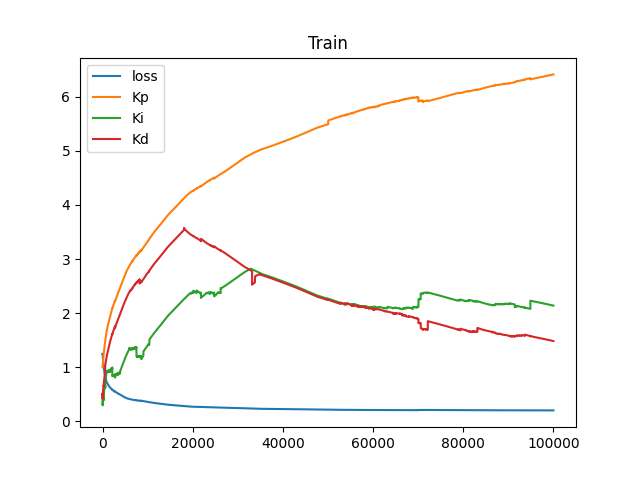
자동 생성된 설명

**그림 6. 조율 인자 최적화 결과 예시**

그림 6은 각 반복 횟수에 따른 최적화 결과를 나타낸다.

|  |
| --- |
| plt.plot(loss, label="loss")      plt.plot(kp, label="Kp")      plt.plot(ki, label="Ki")      plt.plot(kd, label="Kd")      plt.legend()      plt.title("Train")      plt.savefig("train.png") |

위의 코드는 그림 6에서 나타낸 최적화 과정을 도식화하는 코드이다. 위의 코드를 실행하면 그림 7과 같은 결과를 얻을 수 있다.



**그림 7. 조율 인자 도식화 코드 및 결과**

### **[결론]**

PID는 가장 기본적인 제어 시스템 중 하나로, 공정의 목표 값과 실제 데이터 간의 차이가 0이 되도록 조정하며 제어한다. 본 장에서는 딥러닝 기법을 적용하여 PID 제어 시스템을 튜닝했다. 그 결과, 반복 계산을 100,000회 반복하자 오차(loss)가 0으로 거의 수렴한 것을 확인할 수 있다. 더 많은 횟수를 반복한다면 더 정확하게 조율된 PID 제어 시스템을 얻을 수 있을 것이다.

### **[학습 결과]**

* 학습 내용

기본적인 PID 제어 시스템의 이해와 인공지능과의 접목이 필요한 이유 이해.

* 학습 결과 확인하기

대표적인 PID 제어 시스템 조율 방법 익히기.

* 학습 결과 응용하기

본 장의 학습내용에 기반해 실제 공정데이터를 이용해 딥러닝을 적용하여 PID 제어기를 조율하는데 응용.

* 1. **인공신경망 모델 기반 예측 제어를 이용한 공정제어**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | 모델 예측 제어(MPC)의 이해 |
| [방법] | 인공신경망 모델 기반 예측 제어 |
| [응용] | 인공신경망 모델 기반 예측 제어를 이용한 CSTR 제어 |
| [요약] | * 모델 기반 예측 제어의 이해 * Simulink를 이용한 인공신경망 모델 기반 예측 제어 구축 * 인공신경망 모델 기반 예측 제어를 이용한 CSTR 제어 |

### **[이론] 모델 예측 제어를 이용한 다변수 공정의 제어**

다변수 공정이란 입출력 변수가 두 개 이상인 공정을 의미하며, MIMO (multi-input multi-output) 공정이라 부른다. 이와 대비되는 개념으로, 입출력 변수가 각각 하나만 존재하는 단변수 공정 (single-input single-output; SISO)이 있다. 증류탑을 비롯한 화학공학에서 우리가 다루는 대부분의 공정들은 다변수 공정에 해당하며, 변수들 간에 상호간섭이 존재한다. 그림 1을 예로 들면, 공정입력(u1 및 u2)이 변하는 경우 상호간섭(G12 및 G21)이 발생하여, 각각의 공정 입력이 여러 개의 공정 출력(y1 및 y2)에 영향을 미치게 된다. 이러한 상호간섭은 단변수 제어인 PID 제어의 성능을 저하시키는 요인으로 작용하며, 상호간섭이 심각할 경우 단변수 제어로는 좋은 제어성능을 기대하기 어렵다.

한편, 실제 공정에서는 공정 입력 및 공정 출력 변수들에 제약조건이 존재한다. PID 제어의 경우, 알고리즘 특성 상 이러한 제약조건을 고려할 수 없는 어려움으로 인해, low selector 또는 high selector의 도입을 통한 override 제어루프의 구성 또는 clamping 등의 방법을 통해 제어시스템을 구성하나, 그 설계가 복잡하고 성능에 한계를 지닌다.

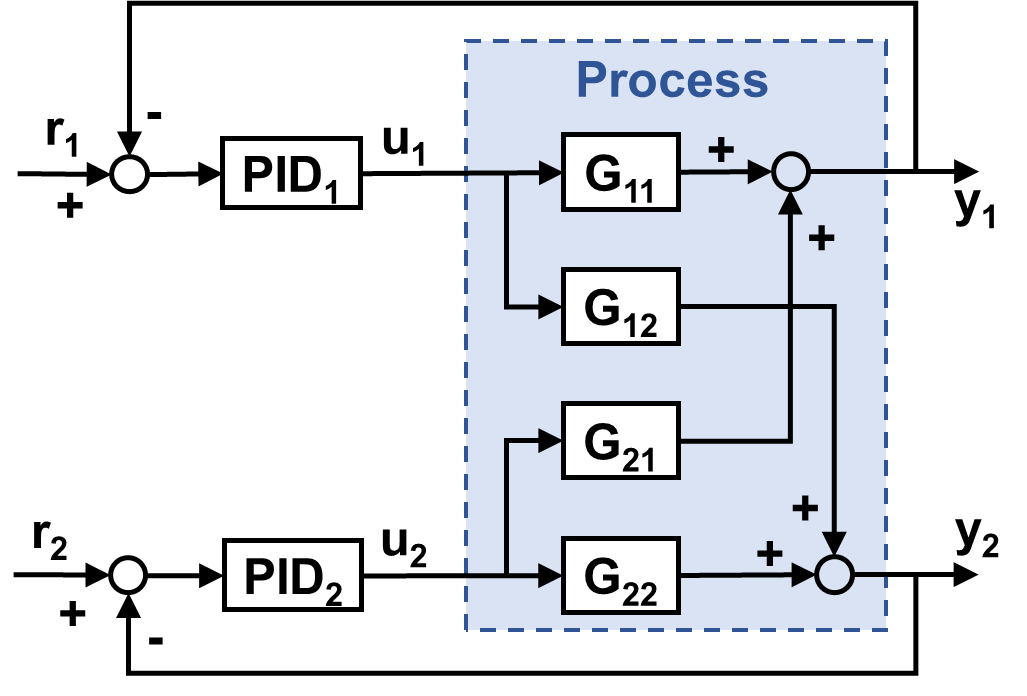


그림1. PID 제어를 이용한 다변수 공정의 제어

다변수 공정을 여러 개의 단변수 제어로 조작하는 것에 대한 어려움과 성능의 한계를 극복하기 위한 좋은 대안으로 다변수 제어가 있다. 연속공정에 대한 대표적인 다변수 제어로서, 그림 2에 보인 바와 같이 모델 예측 제어(model predictive control)가 있다.

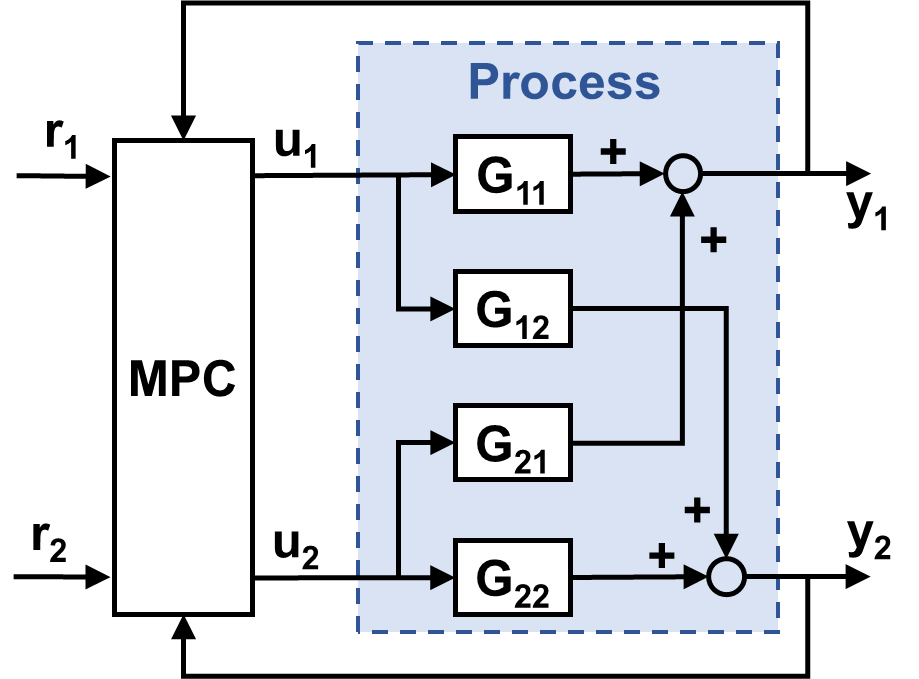


그림 2. 모델 예측 제어를 이용한 다변수 공정의 제어

그림 3은 모델 예측 제어의 동작 원리를 나타낸다. 모델 예측 제어는 실제 공정의 거동을 잘 모사할 수 있는 공정 모델(가상 공정)을 통해, 공정 입력의 변화에 따른 공정 출력을 예측하고, 이를 통해 최적의 공정 입력 시나리오(즉, 단시간 내에 제어오차를 최소화할 수 있는 공정 입력의 변화)를 도출함으로써, 공정을 제어한다. 제어기가 결정하는 공정 입력 값이, 모델을 통해 예측된 공정 출력에 의해 좌우되므로, 제어기의 성능은 공정 모델의 정확성에 크게 의존한다.

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

그림 3. 모델 예측 제어기의 동작 원리

전통적으로, 이러한 공정 모델은 부공간 모델인식(subspace model identification) 기법 등을 이용하여, 공정 입력과 공정 출력의 상호관계, 즉, 공정 입력의 변화에 따른 공정 출력의 거동을 잘 표현할 수 있도록 제작되었으나, 최근에는 인공지능 기법을 통해, 더욱 성능이 개선된 새로운 기법들이 제안되고 있다.

본 장에서는 매트랩을 통해 신경망 모델 기반의 모델 예측 제어를 구축함으로써, 성능이 뛰어난 다변수 제어 설계 능력을 함양하는 것을 목적으로 한다.

### **[문제]**

**매트랩을 이용하여 인공신경망 모델 기반 예측 제어를 구축하고 이를 통해 흐름 w1(t)를 조정하여 제품농도를 유지하도록CSTR을 제어하라.**

**- 이때 입력농도는 Cb1=24.9, Cb2=0.1, CSTR 시스템의 동적 모델의 상수 k1과 k2는 각각 1이다.**

**- 또한, 계산의 복잡함을 피하기 위해 w2(t)는 0.1로 가정하고 탱크의 수위 h(t)는 제어되지 않는다고 한다.**

### **[방법] 인공신경망 모델 기반 예측 제어를 이용한 CSTR 제어**

#### 인공신경망 모델 기반 예측 제어의 첫 번째 단계는 ‘시스템 식별(system identification)’이다. 이 단계에 대해 설명하라.

1. 시스템 식별은 그림 4과 같이 플랜트의 순방향 동적 요소를 표현하도록 신경망을 훈련시키는 것을 말한다. 플랜트 출력값과 신경망 출력값 사이의 예측 오차는 신경망 훈련 신호로 사용된다.

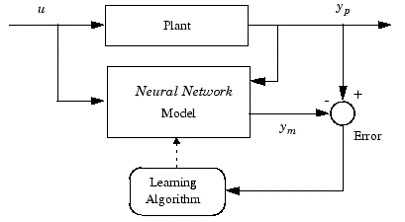


그림4. 인공신경망 훈련 개요

인공신경망 모델은 이전의 직전 입력값과 직전 플랜트 출력값을 사용하여 플랜트 출력값의 미래의 값을 예측한다. 인공신경망 모델의 구조는 그림5와 같이 표현된다.

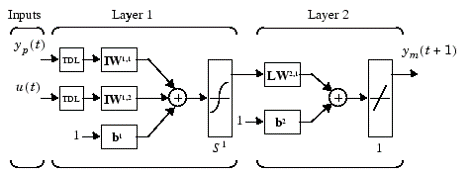


그림5. 인공신경망 모델 개요

#### 인공신경망 모델 기반 예측 제어의 두 번째 단계는 ‘예측 제어(predictive control)’이다. 이 단계에 대해 설명하라.

1. 모델 예측 제어 방법은 이동구간(receding horizon) 기법(SoHa96)을 기반으로 한다. 신경망 모델은 지정된 시간 지평에 대해 플랜트 응답을 예측한다. 예측값은 수치 최적화 프로그램이 지정된 지평에서 다음과 같은 성능 조건을 최소화하는 제어 신호를 파악할 때 사용된다.

텍스트이(가) 표시된 사진

자동 생성된 설명 식 (1)

수식 1. 수치 최적화 수식

여기서 N1, N2, Nu는 추적 오차와 제어 증분이 계산되는 지평을 정의한다. u’ 변수는 잠정적 제어 신호이고 yr은 원하는 응답, ym은 신경망 모델 응답이다. ρ는 제어 증분의 제곱합이 성능 지수에서 차지하는 비중을 결정한다.

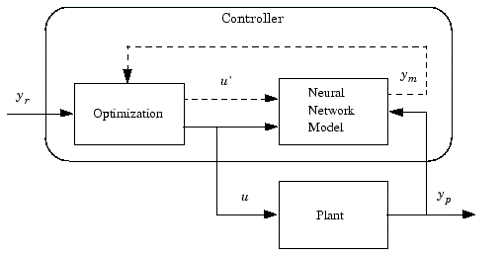


그림6. 모델 예측 제어 과정의 블록 흐름도

그림 6는 모델 예측 제어 과정을 나타낸 것이다. 제어기는 신경망 플랜트 모델과 최적화 블록으로 구성된다. 최적화 블록은 J를 최소화하는 u’의 값을 결정하며, 그런 다음 최적의 u가 플랜트에 입력된다.

#### CSTR(연속 교반 탱크 반응기; continuous stirred-tank reactor)를 도식화하고, 동적 모델을 나타내어라.

1. CSTR을 도식화하면 그림 7와 같이 나타낼 수 있다.

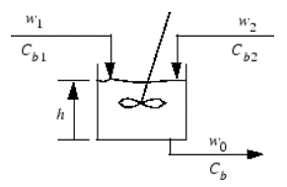


그림 7. 도식화한 CSTR

그림 7를 동적 모델로 나타내면 식 (2)과 식 (3)와 같이 나타낼 수 있다.

식 (2)

식 (3)

여기서 h(t)는 액체 수위이고, Cb(t)는 이 과정의 출력에서의 제품 농도이고, w1(t)는 농축된 공급 용액 Cb1의 유량이고, w2(t)는 희석된 공급 용액 Cb2의 유량이다.

### **[응용] 인공신경망 모델 기반 예측 제어를 이용한 CSTR 제어**

#### 매트립에서 ‘Simulink’ 기능을 활용해 공정 식별(plant identification)을 진행하라.

1. 공정 식별은 그림 8~10와 같은 절차를 통해 할 수 있다.

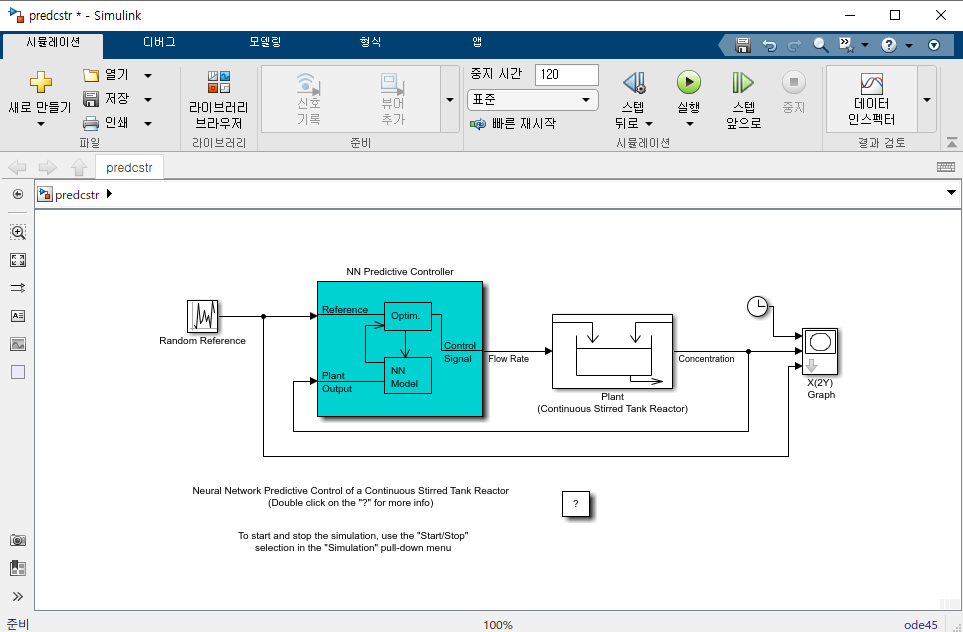


그림8. 실행된 Simulink 모델

매트랩을 실행하여 명령창에 ‘predcstr’을 입력한다. 입력한 명령을 실행하면 그림8과 같이 Simulink 편집기가 실행된다. Plant 블록에는 Simulink CSTR 플랜트 모델이 포함되어 있다. 파랗게 표시된 NN Predictive Controller 블록신호에서 ‘Control Signal은 Plant 모델의 입력에 연결되고, ‘Plant Output’은 Plant 블록 출력에 연결된다. 그리고 ‘Reference’는 Random Reference 신호에 연결된다.

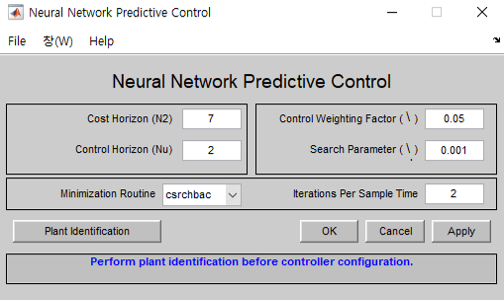


그림9. 모델 예측 제어기 설정창

NN Predictive Controller 블록을 더블 클릭하면 그림 9과 같이 모델 예측 제어기 설정창이 열린다. 이 창에서는 N2와 Nu를 변경할 수 있고, ρ도 정의할 수 있다. 이때, N1은 1로 고정되어 있다. α는 최적화를 위해 필요한 성능 감소분을 결정한다. 또한, 최적화 알고리즘이 사용할 선형 최소화 루틴(Minimization Routine)과 각 샘플 시간에 최적화 알고리즘을 몇 회 반복하여 수행할지 횟수(Iterations Per Sample Time)을 설정할 수 있다.

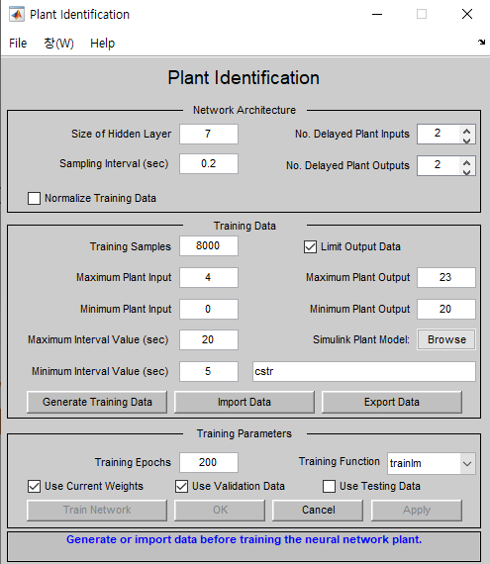


그림10. 인공신경망 플랜트 모델 설정창

플랜트 모델은 미래의 플랜트 출력값을 예측한다. 최적화 알고리즘은 이러한 예측값을 사용하여 미래의 성능을 최적화하는 제어 입력값을 결정한다. 플랜트 모델 신경망에는 앞에서 본 것처럼 하나의 은닉 계측이 있다. 이 창에서 은닉 계층의 크기, 지연 입력값과 지연 출력값의 개수, 훈련 함수를 선택한다.

#### 공정 식별 과정을 통해 생성한 인공 신경망 플랜트 모델을 훈련한 후 시뮬레이션을 실행하여 플랜트 출력값과 기준 신호를 비교하여라.

1. 인공 신경망 플랜트 모델의 훈련은 아래의 그림 11~13를 통해 이루어진다.

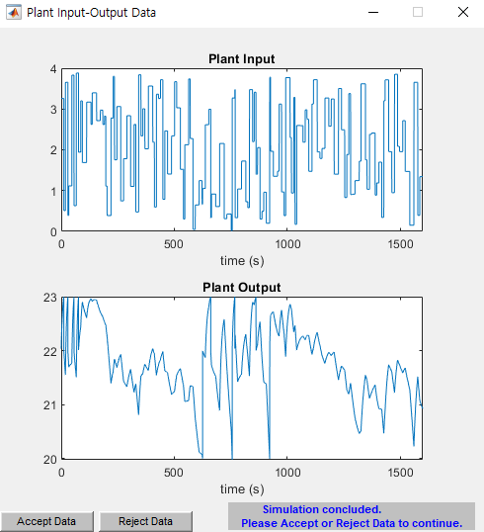


그림11. 인공신경망 플랜트 모델 훈련 결과창

그림10에서 ‘훈련데이터 생성(Generate Training Data)’을 클릭한다. 그림 11와 같이 Simulink 플랜트 모델에 일련의 무작위 스텝 입력값을 적용하여 훈련 데이터를 생성한다. 또한 이를 바탕으로 그림 11의 아래와 같이 잠정적 훈련 데이터가 표시된다. 이 데이터를 통해 미래의 공정 거동을 잘 예측한다고 판단되면 ‘Accept data’를 선택하고, 잘 예측하지 못한다고 판단되면 ‘Refuse Data’를 선택한 후 공정의 거동을 잘 예측할 수 있도록 조건을 변경한다.

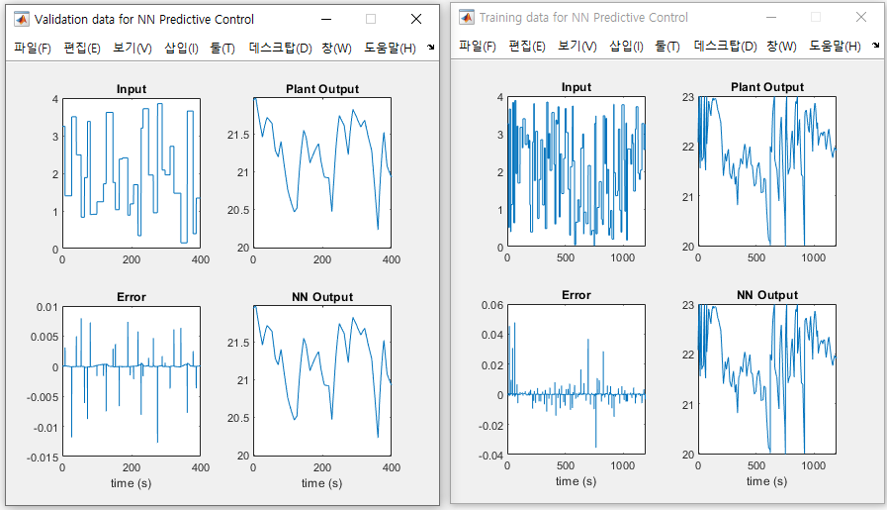


그림12. 플랜트 모델의 응답

그림 11에서 ‘Accept Data’를 클릭하고, 인공신경망 플랜트 모델 설정창(그림 10)에서 ‘Train Network’를 클릭한다. 클릭 시 플랜트 모델 훈련이 시작되고 훈련이 완료되면 그림 12과 같이 플랜트 모델의 응답이 표시된다. 이때, 검증 데이터와 테스트 데이터가 존재하는 경우 각각에 대한 별도의 플롯도 표시된다.

훈련이 완료된 후, 플랜트 모델 설정창(그림 10)에서 ‘Train Network’를 다시 선택하여 동일한 데이터 세트로 훈련을 계속할 수도 있고, ‘Erase Generated Data’를 선택하여 새로운 데이터 세트를 생성할 수도 있고, 현재 플랜트 모델을 수락하고 폐루프 시스템의 시뮬레이션을 시작할 수도 있다.

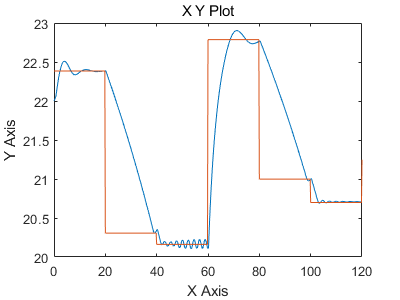


그림13. 플랜트 출력값과 기준 신호의 비교

플랜트 모델 설정창(그림 10)에서 ‘OK’를 선택하면 NN Predictive Controller 블록으로 훈련된 신경망 플랜트 모델을 불러오게 된다. 또한, 모델 예측 제어기 설정창(그림 9)에서 ‘OK’를 선택하면 NN Predictive Controller 블록으로 제어기 파라미터를 불러오게 된다. 이후 Simulink 편집기로 돌아가서 시뮬레이션을 실행하면 그림 13과 같이 플랜트 출력값과 기준 신호가 표시된다.

### **[결론]**

모델 기반 예측 제어는 고급 제어 기법 중 하나로, 공정의 거동을 잘 모사하는 모델을 생성해 이를 제어하는 방법을 통해 제어한다. 본 장에서는 인공신경망을 접목해 인공신경망 모델 기반 예측 제어를 통해 CSTR을 제어하였다. 같은 방법을 통해 보다 복잡한 공정으로 적용분야를 확대하여 공정 제어를 수행할 수 있다.

### **[학습 결과]**

* 학습 내용

모델 기반 예측 제어의 이해 및 인공신경망을 접목한 인공신경망 모델 기반 예측 제어 방법 익히기

* 학습 결과 확인하기

Simulink를 이용한 인공신경망 모델 기반 예측 제어 공정 모델 생성 및 설정하기

* 학습 결과 응용하기

인공신경망 모델 기반 예측 제어를 통해 CSTR 제어 모델 만들기