The Journal of the Convergence on Culture Technology (JCCT)
Vol. 9, No. 6, pp.957-966, November 30, 2023. pISSN 2384-0358, eISSN 2384-0366

http://dx.doi.org/10.17703/JCCT.2023.9.6.957
JCCT 2023-11-115

오버슈트를 제한하는 실시간 적응형 PID 온도제어

Real-time Adaptive PID Temperature Control that limits Overshoot

남진문*

Jin Moon Nam*

요 약 본 논문에서는 새로운 실시간 적응형 PID 온도제어 기법을 제안한다. 제어대상을 표현하는 모델을 도입하여 오버슈트가 발생하지 않도록 하는 기법이다. 오버슈트의 원인이 되는 과도한 적분을 막기 위해 적분 제어량이 실시간으로 모델의 열손실을 추종하도록 적분 이득을 조정한다. 기본적으로 비례 제어는 오버슈트를 유발하지 않는다. PID 제어에서 오버슈트의 발생 원인은 적분에서 기인한다. 기존의 PID 제어는 적분이 비례 제어에 종속되고 게인이 상수로 고정되었다. 그 결과 서로에게 부정합되는 두 이득을 적용하면 과도한 오버슈트가 발생할 수 있었다. 그러나 제안하는 적응형 제어는 적분 제어량이 항상 열손실을 초과하지 않도록 능동적으로 오버슈트를 제거한다. 따라서 제안하는 기법에 따라 튜닝 실험이 필요 없는 적응형 PID 제어를 실현할 수 있다.

주요어 : 적응형 제어, PID 알고리즘, 온도제어, 오버슈트, 모델

Abstract In this paper, we propose a new real-time adaptive PID temperature control technique. This is a technique that prevents overshoot by introducing a model that represents the control object. To prevent excessive integration that causes overshoot, integral control adjusts the integral gain to track the heat loss of the model in real time. In the conventional PID control, the integration was dependent on proportional control and the gain was fixed to a constant. As a result, applying two gains that mismatch each other could cause excessive overshoot. However, the proposed adaptive control actively eliminates overshoot so that the integral control amount does not always exceed the heat loss. The cause of overshoot in PID control is integration. Basically, proportional control does not cause overshoot. Therefore, according to the proposed technique, adaptive PID control without the need for tuning experiments can be realized.

Key words : Adaptive Control, PID Algorithm, Temperature Control, Overshoot, model

Ⅰ. 서 론

PID(Proportional-Integral-Derivative) 알고리즘(algorithm)은 공학이나 산업계의 제어분야에서 많이 활용되는 중요한 기법이다. 비교적 간단하게 구현할 수 있고 제어기의 구조가 간단하지만 대상물이 목표값에 빠르고 정확하게 도달하도록 제어할 수 있으며 제어의 안정성이 우수하고 성능을 신뢰할 수 있는 것으로 알려져 있다. 이러한 장점으로 PID 알고리즘은 운동역학, 모터속도, 열역학, 온도제어 등 다양한 공학 분야에서

*정회원, 평택대학교 피어선칼리지 교수
접수일: 2023년 10월 1일, 수정완료일: 2023년 10월 20일
게재확정일: 2023년 11월 5일

Received: October 1, 2023 / Revised: October 20, 2023
Accepted: November 5, 2023
*Corresponding Author: namjm@ptu.ac.kr
Dept. of Pierson College, Pyeongtaek Univ., Korea

가장 많이 활용되는 대표적인 제어 기법이다.

1. PID 알고리즘의 동작과 우수성

다음의 수식은 PID 알고리즘의 동작원리를 표현한다. 입력변수 e(t)는 목표값(set value)과 현재값(current value)의 차이(error)를 나타내며 이를 이용하여 비례, 적분, 미분의 제어량을 계산한다. 이때 제어량 h(t)를 결정하는 제어 이득(gain) 상수를 각각 Kp, Ki, Kd로 정의한다[1][2].

$$h(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de}{dt} \qquad (1a)$$

$$e(t) = V_S - V_C(t) \qquad (1b)$$

PID 알고리즘은 제어대상(control object)의 출력에 관계하는 입력과 그 원리가 알려지지 않은 경우의 제어에 효과적일 수 있다. 즉, 제어대상의 모델(model)을 정확히 알 수 없거나 모델링의 과정에 많은 시간과 노력이 요구되는 경우에는 그 과정을 생략하고 PID 알고리즘을 간편하게 활용할 수 있다.

다수의 입력 인자가 출력에 영향을 미치는 경우에도 PID 알고리즘이 사용된다. 예를 들면 온도제어에서 온도변화를 유발하는 인자는 발열소자(heating element) 외에도 주변온도(ambient temperature)와 관계되는 열손실도 있다. 그러나 발열소자 하나만 입력 인자로 설정하는 단일 PID 알고리즘을 이용하여도 정확한 온도제어가 가능하다.

이렇게 제어대상의 동작원리를 나타내는 모델을 고려하지 않은 경우에도 PID 알고리즘이 대단히 효과적이며 모델을 반드시 요구하지 않는 것이 PID 알고리즘의 큰 장점이다. 또한 제어대상의 입출력 관계가 선형(linear)이면 우수한 제어성능과 안정성이 보장된다.

2. PID 튜닝의 필요성

PID 알고리즘의 성능을 결정하는 것은 제어 이득(게인) Kp, Ki, Kd이다. 이 상수는 제어대상의 특성에 맞추어진 최적의 값으로 정해야 한다. 모델은 제어대상의 동작 특성에 대한 정보를 가진다. 따라서 제어대상의 모델을 세우면 게인을 쉽게 구할 수 있다. 그러나 제어대상을 모델링하는 것은 복잡하고 어려운 과정일 수 있으며 모델링이 불가능한 분야도 있다.

PID 알고리즘은 모델을 대신하여 다른 방법을 통하여 게인(Kp, Ki, Kd)을 구하는 방법이 있다. 이 과정이 튜닝(tuning)이며 제어대상에서 실험을 통하여 구할 수 있다[2][3]. 제어대상에 정해진 신호를 입력하고 그 결과 파형을 분석하여 동작의 원리나 특성을 파악하는 사전 실험이다. 이 과정에서 제어대상에 맞는 게인을 구할 수 있으며 PID 알고리즘의 제어성능을 결정한다[4][5][6].

3. PID 알고리즘의 성능

다음 그림 1은 PID 알고리즘의 제어 결과를 나타낸 그림이며 PID 제어성능을 비교하여 표현한다. 하단의 두 그래프는 제어대상에 적합한 게인을 사용하였을 때 얻을 수 있는 양호한 제어 결과이다.

우측의 그래프는 오버슈트(overshoot)가 없으나 안정시간(settling time)이 길어진 상태이다. 좌측의 그림은 안정시간의 단축을 위해 약간의 오버슈트를 허용한 것이다. 그러나 그 위의 그래프는 과도한 오버슈트를 나타낸다. 이 결과는 제어대상에 적합한 게인을 구하지 못한 때문이며 다양한 원인이 있을 수 있다. 튜닝 실험 과정에서 생긴 오류이거나 튜닝 실험의 조건이 부정확하여 발생한 결과일 수 있다[2][7].

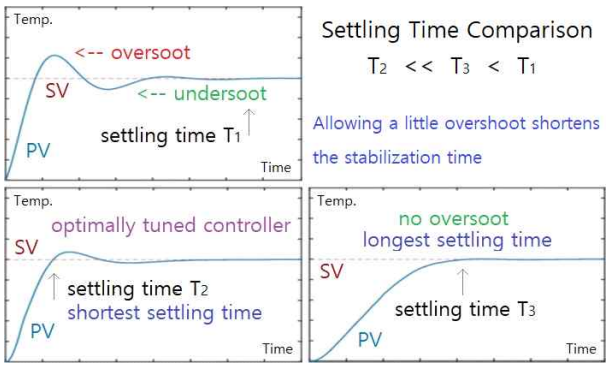


그림 1. PID 알고리즘 제어기의 성능 비교
Figure. 1. Performance comparison of PID algorithm controllers

4. 오버슈트와 안정시간

일반적으로 PID 알고리즘의 제어성능을 나타내는 항목으로 오버슈트와 언더슈트를 사용한다. 목표값(SV) 대비 각각 초과하거나 하락하여 벗어난 정도를 표현한다. 목표값의 (+/-)1% 이내를 양호한 수준으로 판단할 수 있다. 이때 주의할 점은 기울기가 반전하는 마루(ridge)와 골(valley)을 의미한다.

오버슈트/언더슈트가 감소하여 일정 수준 이내로 진입하는 순간의 시점이 안정시간이다. 예를 들면 목표값을 기준하여 오차가 (+/-)1% 이내로 진입하는 순간을 의미하며 다시 벗어나지 않아야 한다.

도달시간은 최초로 목표값에 일치하여 도달하는 순간의 시간을 의미하며 그 이후에 일정 수준을 초과하는 오버슈트가 발생하면 안정되지 못한 것이고 큰 의미를 부여하기 어렵다. 앞의 그림에서 좌측 위의 그림이 도달시간은 가장 빠르지만 안정시간은 가장 늦은 것이다.

PID 알고리즘에서 작은 오버슈트와 빠른 안정시간은 선택의 문제이며 동시에 타협의 대상이다. 제어대상의 분야에 따라서 오버슈트를 극히 제한하는 것이 최선일 수 있지만 빠른 안정시간이 우선될 수도 있다. PID 튜닝 실험을 통하여 게인 Kp, Ki, Kd를 정할 때 이를 고려하여 결정한다.

5. 적응형 제어의 필요성

다음 그림 2는 실제 PID 온도제어 실험의 결과이며 과도한 오버슈트가 발생한 모습을 나타낸다. 제어대상에 부적합한 임의의 게인(Kp, Ki, Kd)를 적용한 결과이다. 언드슈트는 급격히 축소되었으나 온도 안정시간이 약 1시간 이상으로 길어졌다. 오버슈트가 적절히 제한되었다면 20분 이내로 안정될 수도 있었다.

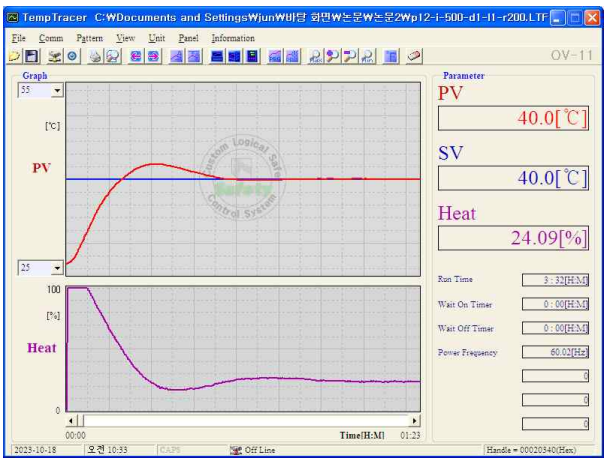


그림 2. PID 온도 제어에서 오버슈트의 발생
Figure. 2. Occurrence of overshoot in PID temperature control

앞에서 논의한 바와 같이 PID 알고리즘은 모델이 없는 분야에서 간편하게 사용할 수 있다. 이는 PID 알고리즘 수식 (1a)만 이용하여 제어함을 의미한다. 이 수식은 제어대상의 동작을 표현하지는 못한다. 따라서 PID 제어에서 게인(Kp, Ki, Kd)이 튜닝 되지 못하여 제어대상에 부적합하면 앞의 그림 2와 같이 제어성능이 극히 낮아지는 결과는 당연하다.

이와 같은 튜닝의 오류 또는 부적합한 게인의 적용으로 발생하는 오버슈트를 해결하기 위해 본 연구에서 새로운 기법을 제시한다. 이 기법은 기존의 튜닝 실험을 대체할 수 있고 변하는 환경에 맞추어 스스로 적응하는 제어를 실행할 수 있다.

II. 적응형 온도제어

이 연구에서 제시하는 적응형 제어는 제어대상의 모델을 이용한다. PID 수식에 더하여 제어대상을 정확히 기술하는 모델 수식을 같이 사용한다. 모델은 가상의 제어대상이고 PID 알고리즘은 제어 동작의 표현이다. PID 온도제어의 성능을 높이는 적응형 제어의 상세한 이론적 근거와 구현 기법은 다음에서 제시한다. 적응형 제어를 위한 모델의 도입부터 논의한다.

1. 적응형 제어를 위한 모델의 도입

먼저 제어대상이 되는 임의의 물체가 있다고 가정한다. 이때 물체의 온도변화율 dy/dt는 물체의 온도 T(t)와 주변온도 A의 차이에 비례하며 아래의 수식 (2a)로 나타낼 수 있다. 여기서 L은 비례상수이며 물체의 표면적에 관련된다. (-)부호는 열의 손실을 나타낸다.

시간 dt 동안 물체의 열손실로 변화된 열에너지는 물체의 온도변화 dy로 나타난다. 이때 물체의 질량과 비열을 각각 m과 c라고 가정하면 물체의 단위 온도 당 열에너지 용량은 (mc)이다.

다음은 물체 내부의 가열소자(heating element)에 의해 생성되는 열에너지의 히팅파워(heating power)가 H라고 가정하면 시간 dt 동안 생성되는 열에너지량은 H x dt가 된다. 이때 가열에 의한 온도 변화량을 dx라 하면 다음 수식 (2c)와 같이 나타낼 수 있다.

여기서 dz는 물체 온도변화의 총합을 나타낸다. 열손실에 의한 온도변화 dy와 열원(heat source)의 열공급에 기인한 온도변화 dx를 합한 값이다. 따라서 물체의 열손실율과 열공급에 의한 물체의 온도변화는 다음의 수식 (2f)로 나타낼 수 있다[8][9][10].

$$(m\,c)\frac{dy}{dt} = -L(T(t) - A) \qquad (2a)$$

$$(m\,c)\,dy = -L(T(t)-A)\,dt \tag{2b}$$

$$(m\,c)\,dx = H\,dt \tag{2c}$$

$$m\,c\,(dx+dy) = H\,dt - L(T(t)-A)\,dt \tag{2d}$$

$$dz = dx + dy \tag{2e}$$

$$(m\,c)\,dz = (H - L(T(t)-A))\,dt \tag{2f}$$

$$(m\,c)\frac{dz}{dt} = H - L(T(t)-A) \tag{2g}$$

2. 모델을 이용한 기존의 PID 온도제어

다음 그림 3은 온도제어를 위해 제작한 자켓(jacket)이다. 기계 장치에서 온도에 민감한 부분을 보호하기 위해 특정한 부분에 부착하여 사용한다. 자켓의 내부에 장착된 발열소자와 온도센서(temperature sensor)를 이용하여 제어기는 온도를 일정하게 유지할 수 있다.

앞의 수식 (2g)를 이용하여 자켓의 온도변화를 표현할 수 있다. 자켓의 발열소자가 발생하는 히팅파워는 수식에서 H가 되고 열손실율 상수 L은 자켓의 물리적 구조와 표면적을 종합하여 표현한다. 또한 자켓 재질의 질량 m과 비열 c가 고려되어 단위 온도당 열용량을 나타내는 상수 (mc)가 결정되며 자켓 온도의 상승속도 dz/dt를 결정한다. 즉, 수식 (2g)는 열손실과 내부의 발열소자가 온도변화에 관계하는 자켓의 모델이 된다. 따라서 앞의 모델 수식은 일반적인 온도제어 대상을 포함하여 자켓의 온도변화도 정확히 표현할 수 있다[10].

그림 3. PID 온도 제어을 위한 자켓
Figure. 3. Jacket for PID temperature control

이제 오버슈트가 발생할 수 있는 기존의 방법으로 PID 알고리즘이 제어대상의 온도를 제어하는 과정을 시뮬레이션(simulation)한다. 즉, PID제어 수식 (1a)를 모델 수식 (2g)에 대입하여 컴퓨터에서 실시하는 가상적 온도제어 실험이다. 먼저 모델 수식 (2g)를 프로그램으로 작성하면 일반적인 온도제어 대상이 컴퓨터에서 구현된다. 다음은 PID 알고리즘 수식 (1a)도 프로그램으로 구현하여 모델 (2g)의 H에 입력한다. 따라서 제어대상에 입력하는 히팅파워는 PID 알고리즘 수식 (1a)가 출력하는 제어량 h(t)가 된다. 이렇게 하면 제어대상을 목표온도까지 제어하는 과정에서 온도변화를 시뮬레이션하는 프로그램이 완성된다.

시뮬레이션에서 사용하는 제어대상은 모델 수식 (2g)에서 임의로(mc=120, L=0.25) 정하였다. 상세한 설정 조건은 아래와 같이 입력하고 60초 주기로 온도변화를 산출하여 누적된 현재온도 V_C를 구하였다. 시뮬레이션 결과의 상세한 데이터는 아래 표 1과 같다.

1. **목표온도 설정 (SV = 180℃)**
2. **주변온도 (A = 25℃)**
3. **임의의 비례 이득 입력 (Kp = 0.7)**
4. **임의의 적분 이득 입력 (Ki = 0.005)**
5. **미분 제어량 소거 (Kd = 0)**

아래 결과에서 과도한 적분의 영향으로 오버슈트가 발생하였다. 420초에 최대 14℃의 오버슈트가 발생하고 1380초 이후에 겨우 안정된 모습이다. 튜닝되지 않은 임의의 게인 Kp와 Ki를 입력하였기 때문이다. 이 오버슈트가 논문에서 제시하는 적응형 제어 기법을 적용하였을 때 제거되는지는 이후의 논의에서 비교한다.

표 1. 모델을 이용한 기존의 PID 온도제어
Table 1. Conventional PID temperature control using model

부적합 게인으로 제어 실행		A	ambient temp [℃]	25	mc [%S/℃]	heat capacity cons.
비례 Kp	0.7	SV	set value [℃]	180	L [%/℃]	heat loss constant
적분 Ki	0.005	T (= CV)	T(t)	SV - A 155	Kp [%/℃]	proportional gain
미분 Kd	0	CV	current value [℃]		Ki [%/℃/S]	integral gain
모델mc	120	z	CV - A		Kd [%S/℃]	derivative gain
모델L	0.25	t	time [S]		dt	60

		(180 조과) 오버슈트		비례항		(44.9 조과) 과도한 적분		열손실	
time	dz	CV	(mc) * dz/dt	= Kp e(t)	+	Ki ∫ e(t) dt	-	L * z	∫ e(t) dt
0	0.00	25.0	100.0	**100.0**		0.0		0.00	0.0
60	**50.00**	75.0	**87.5**	**73.5**		**31.5**		**12.50**	6300.0
120	**43.75**	118.8	69.3	**42.9**		**49.9**		23.44	9975.0
180	34.66	153.4	44.4	**18.6**		**57.9**		32.10	11570.6
240	22.18	175.6	24.6	**3.1**		**59.2**		37.65	11835.2
300	12.31	187.9	10.6	**-5.5**		**56.8**		40.72	11361.4
360	5.28	193.2	1.6	**-9.2**		**52.9**		42.04	10570.9
420	0.79	**194.0**	-3.4	**-9.8**		**48.7**		42.24	9732.7
480	-1.68	192.3	-5.4	**-8.6**		**45.0**		41.82	8995.3
540	-2.72	189.6	-5.7	**-6.7**		**42.1**		41.14	8421.4
600	-2.87	186.7	-5.0	**-4.7**		**40.1**		40.43	8019.3
660	-2.51	184.2	-3.9	-2.9		**38.8**		39.80	7767.9
720	-1.95	182.2	-2.7	-1.6		38.2		39.31	7633.2
780	-1.36	180.9	-1.7	-0.6		37.9		38.97	7580.0
840	-0.85	180.0	-0.9	0.0		37.9		38.76	7577.6
900	-0.45	**179.6**	-0.3	0.3		38.0		38.65	7602.1
960	-0.17	**179.4**	0.0	0.4		38.2		38.60	7637.2
1020	0.00	**179.4**	0.2	0.4		38.4		38.60	7672.6
1080	0.09	**179.5**	0.2	0.4		38.5		38.62	7702.7
1140	0.12	**179.6**	0.2	0.3		38.6		38.65	7725.7
1200	0.12	**179.7**	0.2	0.2		38.7		38.68	7741.4
1260	0.10	**179.8**	0.2	0.1		38.8		38.71	7750.9
1320	0.08	**179.9**	0.1	0.1		38.8		38.73	7755.8
1380	0.05	180.0	0.1	0.0		38.8		38.74	7757.5
1440	0.03	180.0	0.0	0.0		38.8		38.75	**7757.2**

표의 데이터에서 PID 알고리즘이 산출하는 히팅파워는 100%를 초과하지만 표에서는 100%로 제한하였다. 그 이유는 실제 제어대상에서 발열소자 출력값은 시뮬레이션과는 다르게 (-)음수와 (+)100% 이상은 불가능하기 때문이다. PID 알고리즘에서 게인(Kp, Ki, Kd)은 이 문제도 고려하여 결정해야 한다.

3. 적응형 제어를 위한 게인의 검출 과정

이제 논문에서 제시하는 적응형 제어 기법을 적용하는 과정에 대해 알아본다. 상세한 동작 과정은 다음의 단계에 따라 진행한다.

1. 제어대상 식별 - 열용량 상수(mc) 검출
2. 제어대상 식별 - 열손실율 상수(L) 검출
3. PID 제어 알고리즘 - 비례 게인(Kp) 검출
4. PID 제어 알고리즘 - 적분 게인(Ki) 검출
5. PID 제어 알고리즘 - 미분 게인(Kd) 검출
6. 실시간 적응형 PID 제어 적용

먼저 앞 표 1의 데이터를 이용하여 상수 mc와 L을 검출하면 수식 (2g)가 완성되어 제어대상을 특정하는 식별(Control object identification) 단계가 완성된다. 상세한 계산 과정은 이후 절에서 다룬다.

다음은 이 결과를 이용하면 제어대상에 적합한 게인(Kd, Ki, Kd)을 검출할 수 있다. 수식 (2g)에서 열용량 상수 mc는 입력되는 열에너지가 온도의 변화로 나타나는 비율을 표현한다. 이 상수가 비례제어의 강도를 결정하는 게인 Kp에 연관될 것으로 예측할 수 있다.

또한 목표온도에 도달하여 안정되면 수식 (1a)에서 비례항과 미분항의 출력은 0이 된다. 이때 상수 L이 만드는 열손실은 적분항과 같아야 한다. 즉 적분항이 열손실을 상쇄할 수 있도록 Ki를 정할 수 있으며 이 점에 착안하여 다음절에서 오버슈트의 제거에 대하여 논의한다.

본 논문에서는 미분 이득을 구하는 과정은 생략하여 Kd는 0으로 정한다. 즉, PID 제어에서 미분항을 삭제한 PI제어로 한정한다. 그 이유는 PI제어만으로도 충분한 제어성능을 구현할 수 있다.

미분항은 비례제어와 적분제어가 유발하는 온도상승율 dz/dt를 감소시키는 역할을 한다. 목표값에 도달한 정상상태에서는 제어에 대한 기여가 없으나 외란(disturbance)에 의하여 발생하는 온도변화에 대한 복원력를 강화하여 제어 안정도를 향상한다. 외란에 반응하는 강도는 선택의 문제이나 일반적으로 Kd는 온도상승율을 10% 정도 줄여 외란에 강한 제어(robust control)를 구현할 수 있다.

앞에서 기술한 바와 같이 제어대상의 온도변화 특성(mc, L)이 고려된 게인(Kd, Ki, Kd)을 구할 수 있다. 이 값은 실물에서 튜닝 실험으로 구한 결과와 동일한 효과를 가진다. 이 결과를 실물의 온도제어에 적용하면 본 논문에서 제시하는 실시간 적응형 제어가 구현된다.

4. 적응형 제어의 오버슈트 제한

일반적으로 온도제어는 오버슈트를 중요하게 다룬다. 기존의 제어에서 발생하는 오버슈트를 제거하는 새로운 기법에 대해 아래에서 상세히 논의한다.

모델 수식 (2g)는 좌측부터 온도상승율, 히팅파워, 열손실율을 각각 나타낸다. 여기서 히팅파워 H는 PID 제어 수식 (1a)을 의미한다. 이때 기존의 제어는 비례항과 적분항이 같이 온도상승(율)에 기여할 수 있으며 Kp와 Ki는 상수였다. 따라서 적분항 단독으로는 오버슈트에 대응할 수 없었다. 즉, 기존의 방법으로는 Kp에 적합하고 정확한 Ki를 미리 선정해야 하였다. 튜닝 실험이 반드시 필요했으며 중요한 이유이다.

Kp는 오버슈트의 발생에 직접의 영향은 없지만 Ki가 상수로 초기에 결정되는 기존의 제어에서는 오버슈트 발생에 간접의 영향을 미친다. Kp의 강도에 따라 온도가 상승하는 그래프의 패턴이 다르면 그 결과 적분량도 달라지기 때문이다. 따라서 Ki는 항상 Kp에 맞게 수정되어야 했지만 기능이 없었다.

온도 상승의 패턴이 변경되는 원인이 또 있다. 외부의 온도가 낮아지면 상승속도가 느려지고 적분량이 증가하지만 상수인 Ki가 변경될 수 없었다. 이 문제는 사전의 튜닝 실험으로도 해결할 수 없는 문제이다.

본 논문에서는 이 문제를 해결하기 위해 서로 독립된 게인을 제안한다. 적분항은 모델 수식 (2g)에서 열손실율만 대체하도록 정한다. 온도상승(율)에는 비례항만 기여하도록 역할을 분담한다. 또한 Kp와 Ki를 매 순간 적응하는 변수로 제안한다.

오버슈트는 목표온도에 도달하는 마지막 순간에 결정된다. 이때는 비례항의 출력은 0으로 수렴하여 Kp의 오버슈트에 대한 영향력이 없어진다. Ki는 모델 수식 (2g)가 실시간으로 제시하는 열손실율에만 맞추어 적응

하면 오버슈트는 없어진다.

앞에서 살펴본 바와 같이 다양한 이유로 적분량이 변경될 수 있다. 또한 모델 수식 (2g)가 실시간으로 제시하는 열손실율도 변한다. 따라서 매 순간 열손실율을 고려하여 Ki가 제어출력을 다시 변경하는 기능이 필요하다. 이 기능을 도입하면 변하는 환경에서도 오버슈트가 없는 적응형 제어를 구현할 수 있다.

5. 제어대상의 식별과 게인 검출

이제 적응형 제어 기법에 대하여 앞에서 제시한 내용을 수학적으로 기술한다. 먼저 제어대상을 식별하는 단계이며 다음에서 mc를 검출하는 방법을 기술한다.

수식 (2g)에서 현재온도 T(t)가 주변온도 A와 같다고 가정하면 열손실은 0이 된다. 이때 히팅파워 H가 100%의 전력(히팅파워)을 출력하면 최고의 온도상승율 (dz/dt)max가 나타난다. 앞의 표 1에서 첫 60초 구간의 기울기를 의미한다. 시간 변화분 dt와 온도 변화분 dz에 각각 60초, 50.00℃를 대입하여 다음과 같이 계산하면 열용량(mc)이 120임을 알 수 있다. 따라서 앞에서 설정한 제어대상(mc=120, L=0.25)을 정확히 검출할 수 있음이 증명된다.

단 PID 알고리즘 수식 (1a)의 히팅파워 h(t)는 시간 t에 따라 변할 수 있는 값을 의미하지만 다음과 같이 일정한 상수를 표현할 때에는 H를 사용하기로 한다. 제어대상의 현재온도 Vc는 T(t)와 같은 표현이다. 수식 (3e)는 컴퓨터 프로그램의 이산적(discrete) 표현이다.

$$T(t) = A \tag{3a}$$

$$\left[(m\,c)\frac{dz}{dt}\right]_{T(t)=A} = H \tag{3b}$$

$$\left[\frac{dz}{dt}\right]_{\max} = \frac{100}{(m\,c)} \tag{3c}$$

$$(m\,c) = 100\left[\frac{dt}{dz}\right]_{100} \approx 100\left[\frac{60}{T(t)-A}\right]_{100} = 100\,\frac{60}{50.00} = 120 \tag{3d}$$

$$(m\,c) = (H0 - L\,z0)\,\frac{dt1}{dz1} = (100 - 0.25\times 50)\,\frac{60}{43.75} = 120 \tag{3e}$$

다음 단계는 열손실율 상수 L을 검출하기 위해 앞의 수식 (2g)를 이용한다. 히팅파워 100% 출력에서 열손실이 온도 변화율을 감소시키는 관계를 수식 (4a)로 나타낼 수 있다.

앞의 표 1에서 히팅파워 출력 100%에서 첫 온도 상승분은 50.00℃이다. 주변온도와 동일한 온도에서 열손실이 없었다. 그 다음은 열손실로 인하여 상승분이 43.75℃로 감소한 것을 확인할 수 있다. 이를 이용하여 아래와 같이 계산에 적용하면 히팅파워의 12.5%가 손실되었으며 열손실율 상수는 0.25임을 알 수 있다. 따라서 앞에서 설정한 제어대상(mc=120, L=0.25)을 정확히 검출할 수 있음이 증명된다. 수식 (4e)는 컴퓨터 프로그램을 위한 이산적 표현이다.

$$(m\,c)\frac{dz}{dt} = H_{100\%} - L(T(t) - A) \tag{4a}$$

$$L(T(t) - A) = H_{100\%} - (m\,c)\frac{dz}{dt} \tag{4b}$$

$$L(T(t) - A) = 100 - 120\frac{43.75}{60} = 12.5[\%] \tag{4c}$$

$$L = \frac{12.5}{(T(t) - A)} = \frac{12.5}{50} = 0.25 \tag{4d}$$

$$L = \frac{dz1\times H1 - dz2\times H0}{dz1\times z1 - dz2\times z0} = \frac{43.75\times 92.8 - 34.66\times 100}{43.75\times 93.8 - 34.66\times 50} = 0.25 \tag{4e}$$

다음의 단계는 Kp를 검출하는 방법을 기술한다. 앞의 수식 (1a)에서 PID 알고리즘의 제어출력은 수식 (2g)의 히팅파워 H에 해당하므로 다음 수식 (5a)로 나타낼 수 있다. Kd는 0으로 가정한다. 제어의 첫 순간 (t=0)에는 온도 T(t)가 주변온도 A와 같아서 열손실항과 적분항이 0이다. 결과적으로 수식 (5d)와 같이 비례항만 온도상승에 기여하는 형태로 단순화된다.

$$(m\,c)\,\frac{dz}{dt} = Kp\;e(t) + Ki\int_0^t e(t)dt - L(T(t) - A) \tag{5a}$$

$$[T(t)]_{t=0} = A \tag{5b}$$

$$\left[\int_0^t e(t)dt\right]_{t=0} = 0 \tag{5c}$$

$$\left[(mc)\,\frac{dz}{dt}\right]_{T(0)=A} = Kp\;e(t) \tag{5d}$$

$$\left[Ki\int_0^t e(t)dt\right]_{e(t)=0,\;dz=0} = L(T(t) - A) \tag{5e}$$

온도가 목표값에 도달하여 안정상태에 진입하면 적분항은 열손실항에 수렴한다. 표 1의 하단에서 열손실 수렴상태를 확인할 수 있다. 이때는 온도변화 dz와 비

례항이 0이며 수식 (5e)를 만족한다.

또한 앞의 표에서 온도가 180℃에 도달하면 온도변화(dz)가 없는 것을 알 수 있다. 앞서 오버슈트를 제한하기 위해 적분항이 열손실율을 추종하도록 제안하였다. 수식 (5e)는 이 제안이 타당함을 나타내며 앞의 표 결과에서도 검증이 된다. 다만 제어의 모든 과정에서도 이 전제가 타당한지는 이후의 실험 및 결과 분석에서 살펴보기로 한다.

이제는 오버슈트를 제한하기 위해 적분항이 항상 열손실항에 같아지도록 수식 (5e)를 이용하여 Ki를 결정하기로 한다. 이렇게 하면 온도상승에는 비례항만 기여하도록 하는 수식 (5d)도 항상 성립하게 된다. 즉, 수식 (5e)가 모든 영역에서 항상 성립되도록 하면 수식 (5d)도 항상 모든 영역에서 성립하도록 결정할 수 있으며 결과적으로 수식 (5a)도 항상 만족하게 된다.

이렇게 하면 제어에 영향을 미치는 인자가 변하는 환경이나 부적합한 게인이 적용된 제어에서도 오버슈트가 없는 안정된 제어성능을 확보할 수 있다.

Kp를 계산하기 위해 수식 (5d)의 양변을 적분하면 아래와 같다. 적분량은 제어 과정에서 매 순간 누적하여 앞의 표에서와 같이 구할 수 있다. 적분량을 수식 (6e)에 대입하면 매 순간 적응형 Kp를 계산할 수 있다.

$$(mc)\,dz = Kp\,e(t)\,dt \tag{6a}$$

$$(mc)\int_0^z dz = Kp\int_0^t e(t)\,dt \tag{6b}$$

$$z = (T(t) - A) \tag{6c}$$

$$(mc)\,(T(t) - A) = Kp\int_0^t e(t)\,dt \tag{6d}$$

$$Kp = \frac{(m\,c)\,(T(t) - A)}{\int_0^t e(t)\,dt} = \frac{120 \times 155}{7757.2} = 2.398 \tag{6e}$$

다음 단계는 Ki를 검출하는 방법을 기술한다. 앞의 수식 (5a)에서 정상상태에 도달하여 온도상승이 멈추면 온도변화 dz가 0이 되며 다음과 같이 나타낼 수 있다. 수식 (7a)에도 앞의 표에서 구한 적분량을 대입하면 매 순간 적응형 Ki를 계산할 수 있다.

$$Ki = \frac{L(T(t) - A)}{\int_0^t e(t)\,dt} = \frac{0.25 \times 155}{7757.2} = 0.004995 \tag{7a}$$

이제까지 매 순간 적응형 게인(Kd, Ki, Kd)을 검출하는 수식에 대하여 논의하였다. 이 값은 실시간으로 연속하여 구할 수 있다.

앞의 표는 기존 방식의 PID 온도제어 결과이며 오버슈트가 발생함을 보였다. 이제 앞에서 검출한 게인을 본 논문에서 제시하는 적응형 PID 제어에 적용하여 그 효과를 알아보려 한다. 시뮬레이션을 통하여 앞의 표에서 발생한 오버슈트가 제거되는지 확인하기 위함이다.

6. 오버슈트 제거를 위한 프로그램 구현

이제 적응형 PID 제어에서 오버슈트가 제거되는지를 확인하기 위해 프로그램 구현을 통하여 검증을 실시한다. 앞의 표1에서 오버슈트가 발생한 그 제어대상(mc=120, L=0.25)을 그대로 사용한다.

단 게인은 이전의 값(Kp=0.7, Ki=0.005)를 사용하지 않고 이제까지 적응형 제어를 위해 앞에서 논의한 기법을 적용한다. 동일한 목표온도에 대한 적응형 PID 제어를 재현하였을 때 이전의 오버슈트가 제거되는지 확인하기 위함이다.

앞에서 논의한 게인(Kd, Ki, Kd)을 검출하는 수식을 코드로 구현한 프로그램의 일부를 다음에 나타내었다.

```
//=============================================================================================//
if( RoomTemp == 0 ) {                    //제어시작 온도 T(0)에서는 열손실의 초기값은 0
   RoomTemp = GetRoomTemp();
   HeatLossRate_L = 0;
   TempErrorIntegral = 0;
}
while( dt == 16 ) {
   HeatPower0 = HeatPower1;              //이전의 히팅파워 값 저장
   HeatPower1 = HeatPower2;              //이전의 히팅파워 값 저장
   HeatPower2 = GetHeatPower();          //현재의 히팅파워 값 읽음
   delay_ms(16000);                      //16초에 한번 실행
   PresentTemp0 = PresentTemp1;          //이전의 현재온도 값 저장
   PresentTemp1 = PresentTemp2;          //이전의 현재온도 값 저장
   PresentTemp2 = GetPresentTemp();      //현재의 온도센서 값 읽음
   TempChange1 = TempChange2;            //이전의 온도변화 값 저장
   TempChange2 = GetTempChange();        //현재의 온도변화 값 읽음
   MeasuredTempSlewRate = TempChange2 / dt;
//=============================================================================================//
//실제 플랜에 PID제어(히팅)를 적용하여 얻어난 실제 온도변화가 가장 모델에서도 나타나도록 상수(HeatCapacity_mc, HeatLossRate_L)를 먼저 찾는다
//---------------------------------------------------------------------------------------------//
   if( TempChange2 != 0 )
      HeatCapacity_mc = (HeatPower1 - (HeatLossRate_L * (PresentTemp1 - RoomTemp))) * dt / TempChange2;
//계이시작 온도 T(0)에서는 PresentTemp와 RoomTemp가 동일하다고 가정하여 (HeatPower * dt / TempChange)만으로 열용량 계산
   Numerator = (TempChange1 * (PresentTemp1 - RoomTemp) - TempChange2 * (PresentTemp0 - RoomTemp));
   if ( Numerator != 0 )
      HeatLossRate_L = ((TempChange1 * HeatPower1) - (TempChange2 * HeatPower0)) / Numerator;
//---------------------------------------------------------------------------------------------//
//앞절의 열손실율 상수를 알아낸 다음, 현재 히팅파워에서 열손실을 제외한 온도 상승에 기여하는 히팅량을 계산하고, 비열을 적용하면 온도변화도 추정할 수 있다
   HeatPowerUp = (HeatPower2 - (HeatLossRate_L * (PresentTemp2 - RoomTemp)));  //온도 상승에 기여한 히팅량
   TempChange3 = HeatPowerUp * dt / HeatCapacity_mc;          //찾은 모델에서 다음 n=3 단계의 온도변화 추정
   MaxTempSlewRate = MeasuredTempSlewRate * 100 / HeatPowerUp;        //HeatPower 100%의 온도변화율
//=============================================================================================//
//PID제어를 Gain(Kp, Ki, Kd)의 가상 오토튜닝,          //Kp항 발열량은 온도상승에 기여하고, Ki항은 열손실에 수렴하도록 추정함
//---------------------------------------------------------------------------------------------//
   TempErrorIntegral += Error * dt;
   Estimated_Kp = (HeatCapacity_mc * (PresentTemp2 - RoomTemp) / TempErrorIntegral);
   Estimated_Ki = (HeatLossRate_L * (PresentTemp2 - RoomTemp) / TempErrorIntegral);
//  Estimated_Ki = (HeatLossRate_L * (PresentTemp2 - RoomTemp) / TempErrorIntegral)*110/100; //110%-OS조절
//  Estimated_Kd = HeatCapacity_mc / 10;
//---------------------------------------------------------------------------------------------//
//  Kp = (Estimated_Kp + (Kp * 99)) / 100;
//  Ki = (Estimated_Kp + (Ki * 99)) / 100;
//  Kd = (Estimated_Kp + (Kd * 99)) / 100;
//---------------------------------------------------------------------------------------------//
   Kp_Accuracy = (100 * Estimated_Kp) / Kp;    //Kp_Accuracy = 90~110이면 최적의 Kp가 적용되어 제어되는 상태
   Ki_Accuracy = (100 * Estimated_Ki) / Ki;    //Ki_Accuracy = 90~110이면 최적의 Ki가 적용되어 제어되는 상태
   Kd_Accuracy = (100 * Estimated_Kd) / Kd;    //Kd_Accuracy = 90~110이면 최적의 Kd가 적용되어 제어되는 상태
}
//=============================================================================================//
```

그림 4. PID 제어 이득 검출을 위한 프로그램 코드
Figure. 4. Program code for PID control gain detection

이 프로그램에서 비례제어 게인 Kp는 앞의 수식 (6e)를 이용하여 구하고 있다. Ki도 동일한 방법으로 수식 (7a)를 이용하여 구한다. 미분 제어는 앞에서 기술한 바와 같이 기본적으로 제어성능에 관여하는 정도가 낮다. 본 논문에서는 PI제어로 가정하고 Kd에 대한 자세

한 기술은 생략한다.

7. 적응형 PID 제어 결과의 분석

다음의 표 2는 앞에서 작성한 프로그램을 샐행한 결과이며 적응형 PID 제어의 효과를 나타낸다. 이 결과는 앞의 표1과는 다르게 오버슈트가 전혀 발생하지 않았다. 420초에서 목표온도 180℃에 도달하여 안정되어 있다. 420초 순간에 최대 14℃의 오버슈트가 발생한 이전의 결과에 비해서 제어성능이 훨씬 향상되었다. 동일한 제어대상에서 기존의 제어 대비 적응형 제어가 향상된 결과를 보인다는 검증이다. 모델 수식 (2g)에 mc와 L에 각각 120와 0.25를 동일하게 적용한 결과이다.

다른 점은 표1에서 적용한 게인은 임의의 값(Kp=0.7, Ki=0.005)이었다. 즉, 제어대상(mc=120, L=0.25)에는 부적합한 게인에 해당하여 과도한 오버슈트의 원인이 되었다.

그러나 표2는 적응형 제어 기법에서 실시간 검출하여 변하는 게인을 적용한 결과이다. 표 2의 좌측에서 Kp와 Ki가 매 순간 변하는 것을 확인할 수 있으며 이로 인하여 두 게인이 서로에게 적합한 관계로 적응하는 결과로 이어진다.

또 다른 차이를 표의 우측에서 발견할 수 있다. 적분항이 항상 열손실을 추종하고 있다. 본 논문에서 제시한 적응형 제어에서는 오버슈트를 제한하기 위해 적분항이 항상 열손실과 같도록 Ki가 능동적으로 적응한 결과이다.

표 2. 적응형 PID 온도제어의 시뮬레이션 결과
Table 2. Simulation results of adaptive PID control

적응형 제어		[S]	목표[℃]	현재온도[℃]	변화[℃]	합계[%]		비례항[%]		적분항[%]		열손실[%]
Kp	Ki	time	SV	CV (= T)	dz	(mc)*dz/dt	=	Kp e(t)		Ki ∫ e(t) dt	=	L * (T - A)
0.645	0.000000	0	180	25.0	0.0	100.0	=	100.0		0.0	=	0.0
0.952	0.001984	60	180	75.0	50.0	100.0	=	100.0		12.5	=	12.5
1.128	0.002350	**120**	180	125.0	50.0	62.0	=	62.0		23.4	=	23.4
1.332	0.002774	**180**	180	156.0	31.0	31.9	=	31.9		32.1	=	32.1
1.527	0.003181	240	180	172.0	16.0	12.2	=	12.2		37.6	=	37.6
1.721	0.003584	300	180	178.1	6.1	3.3	=	3.3		40.7	=	40.7
1.909	0.003977	360	180	179.7	1.6	0.5	=	0.5		42.0	=	42.0
2.083	0.004340	**420**	180	**180.0**	0.3	0.0	=	0.0		42.2	=	42.2
2.232	0.004649	480	180	**180.0**	0.0	(0.0)	=	0.0		41.8	=	41.8
2.345	0.004885	540	180	**180.0**	0.0	0.0	=	0.0		41.1	=	41.1
2.420	0.005041	600	180	**180.0**	0.0	(0.0)	=	0.0		40.4	=	40.4
2.459	0.005123	660	180	**180.0**	0.0	0.0	=	0.0		39.8	=	39.8
2.472	0.005150	720	180	**180.0**	0.0	(0.0)	=	0.0		39.3	=	39.3
2.468	0.005141	780	180	**180.0**	0.0	0.0	=	0.0		39.0	=	39.0
2.455	0.005115	840	180	**180.0**	0.0	(0.0)	=	0.0		38.8	=	38.8

이 분석를 통해서 Ki는 Kp에 맞추어 정해야 한다는 것이 중요하다는 것을 알 수 있다. 또한 결과의 비교를 통해 본 논문에서 제시하는 실시간 적응형 제어가 오버슈트 대응에 훨씬 효과적임을 알 수 있다.

지금까지 가상의 제어대상 모델을 이용하는 시뮬레이션을 통해 본 논문에서 제시하는 실시간 적응형 제어 기법에 대하여 이론적으로 논의하였다. 다음에서 실험을 통하여 검증을 실시한다.

III. 실험 및 결과 분석

앞에서 실시한 컴퓨터 시뮬레이션을 통하여 실시간 적응형 PID 제어 기법에 대한 이론적 논의의 타당함이 입증되었다. 이제 실제 제어대상을 이용한 온도제어 실험을 실시하고 그 성능을 측정한다. 실험에는 PID 온도제어용으로 제작된 그림 3의 자켓들을 사용하였다.

아래 그림 5는 적응형 PID 제어의 성능을 검증하기 위하여 자켓을 이용한 실험의 결과를 나타낸다. 실험에서 사용한 자켓은 1번, 2번, 3번이다. PID 온도 제어기는 본 논문에서 제시한 적응형 알고리즘을 탑재하였다. 적응형 제어는 튜닝이 필요 없으며 실제로 제어기에서 튜닝 버튼을 제거하였다.

제어기에는 목표온도를 3가지 설정할 수 있으며 SV1, SV2, SV3에 기억한다. 설정온도는 사용자가 변경할 수 있으나 각각에는 제어의 특징이 있다. SV1를 사용하면 오버슈트가 제한되도록 하였고 SV3는 온도 도달시간의 단축을 우선하도록 적분 게인 Ki를 조정하였다. SV2는 중간의 특징을 가진다.

제어 분야에 따라서 오버슈트의 제한이 중요한 경우는 SV1를 선택할 수 있다. 반면에 빠른 도달시간이 필요한 경우는 SV3를 선택할 수 있다. 이 특화된 제어 특성(전략)을 활용하면 튜닝이 필요 없는 적응형 제어의 활용성을 크게 높일 수 있다.

1. 적응형 제어의 실험 결과

실험에 사용한 자켓들은 온도 및 전기적 특성, 크기, 형태 등에서 매우 상이하며 상세한 사양을 다음 그림 5에 나타내었다. 자켓 #1, #2, #3의 목표온도가 다르며 전기적 사양에 따라 각각 90℃, 190℃, 150℃로 정해진다. 발열용량은 순서대로 110W, 279W, 300W이다.

각 자켓은 SV1, SV2, SV3를 이용하는 3번의 온도실험에 사용된다. 온도실험 순서는 자켓 #1 90℃에서 3번(SV1, SV2, SV3), 자켓 #2 190℃에서 3번(SV1, SV2, SV3), 자켓 #3 150℃에서 3번(SV1, SV2, SV3)이다. 오버슈트와 안정시간을 측정하여 비교한다.

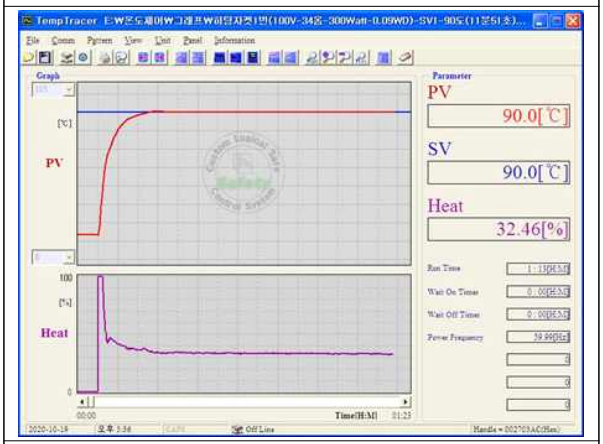
이 실험의 목적은 서로 상이한 제어대상(#1, #2, #3)

에서도 제시한 적응형 제어 기법이 효과적인지 실험으로 검증하기 위함이다. 또한 서로 다른 제어 특성(SV1, SV2, SV3)를 적용하였을 때도 제어성능을 동일하게 유지하는지를 검증한다.

상세한 온도제어 실험의 결과를 그래프로 아래에 나타내었으며 온도제어 결과는 모두 우수한 결과를 보인다. 사전에 튜닝을 실시하지 않은 실험이며 목표온도, 도달시간, 오버슈트 등의 자세한 실험 결과를 다음에 나타내었다.

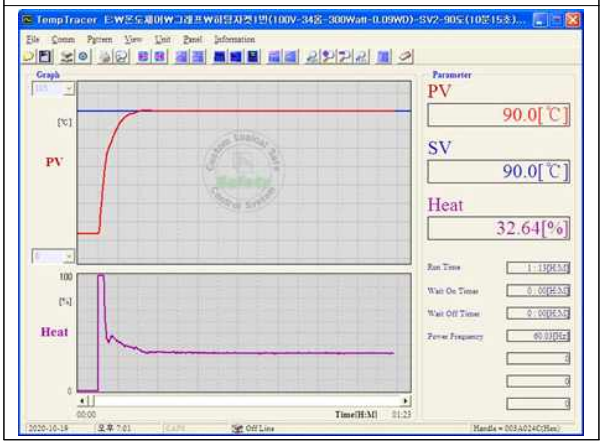
Temperature Control of Jacket #1 using SV1

- Heating jacket : No.1 (100V, 34Ω, 300Watt, 0.09W/cm²)
- PID selection : SV1 setting (provided in the controller)
- Set temperature : 90°C
- Rise time to setting : 11min. 51sec.
- Overshoot : 0.3°C
- Program version : Pr5.1
- Ambient temp. : 24°C



Temperature Control of Jacket #1 using SV2

- Heating jacket : No.1 (100V, 34Ω, 300Watt, 0.09W/cm²)
- PID selection : SV2 setting (provided in the controller)
- Set temperature : 90°C
- Rise time to setting : 10min. 15sec.
- Overshoot : 0.3°C
- Program version : Pr5.1
- Ambient temp. : 24°C



Temperature Control of Jacket #1 using SV3

- Heating jacket : No.1 (100V, 34Ω, 300Watt, 0.09W/cm²)
- PID selection : SV3 setting (provided in the controller)
- Set temperature : 90°C
- Rise time to setting : 8min. 18sec.
- Overshoot : 1.0°C
- Program version : Pr5.1
- Ambient temp. : 24°C

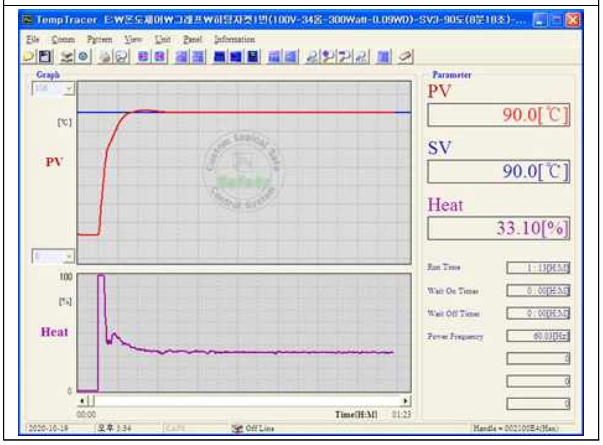


그림 5. 자켓 1번의 온도실험 결과
Figure. 5. Temperature experiment of jacket No. 1

2. 적응형 제어 실험 결과의 비교 분석

앞에서 실시한 실험의 결과에서 제어성능을 나타내는 데이터를 추출하려 비교표로 다음 표 3에 나타내었다. 튜닝을 실시하지 않은 상태에서 단 1대의 동일한 PID 제어기로 실시한 실험의 결과이다.

제어기에는 실험을 위하여 특성이 다른 3가지의 제어 전략이 탑재되어 있으며 순서대로 적용하여 온도실험을 실시하였다. 3종의 자켓에서 총 9번의 실험 결과이며 안정시간과 오버슈트를 비교하였다.

오버슈트는 최소 0.2℃, 최고 0.4℃로 안정되어 있다. 제어 시간의 단축을 우선하는 SV3에서는 오버슈트가 최대 1.1℃까지 높아진다.

목표온도의 안정시간은 최소 8분 18초, 최대 11분 15초를 나타내었으며 모두 양호한 결과를 얻었다. 다만 오버슈트의 제한을 우선하는 SV1에서는 최대 15분 58초까지 길어진 결과를 보이지만 양호한 결과이다.

표 3. 자켓 온도실험 결과의 분석
Table 3. Jacket temperature test results

Rise time (sec), Overshoot(℃)

PID settings (Rise time)	SV1 (Slow)	SV2 (Medium)	SV3 (Fast)
Jacket #1 (90℃ – 0.09W/cm²)	11m 51s (+0.3℃)	10m 15s (+0.3℃)	08m 18s (+1.0℃)
Jacket #2 (190℃ – 0.50W/cm²)	13m 45s (+0.2℃)	11m 26s (+0.2℃)	9m 49s (+0.2℃)
Jacket #3 (150℃– 0.20W/cm²)	15m 58s (+0.4℃)	13m 18s (+0.4℃)	11m 15s (+1.1℃)

3가지 자켓에서 모두 공통으로 나타난 특징은 SV3를 이용하면 안정시간이 빨라지면서 약간의 오버수트가 나타난다. 적응형 게인 Ki를 정하는 과정에서 빠른 안정시간을 위한 계산이 적용된 결과이다. 반대로 SV1를 사용하면 오버슈트가 극히 제한된다. 이 특성은 사용자의 필요에 따라 선택할 수 있는 기능이다.

결론적으로 본 논문에서 제안하는 실시간 적응형 PID 제어 기법을 적용하여 상이한 3종 자켓에서 튜닝을 하지 않은 상태로도 오버슈트가 제한되거나 온도의 안정시간이 빠른 우수한 제어성능을 실현할 수 있었다.

Ⅳ. 결 론

본 논문에서 실시간 적응형 PID 온도제어 기법을 제시하였다. 기존의 제어에서 발생하는 오버슈트의 원인을 분석하여 새로운 대안을 제시하였다.

오버슈트는 적분 이득의 부조화(mismatch)에 의하여 발생한다. 비례 이득이 주도하여 결정되는 온도상승의 궤적을 따라 적분량이 결정된다. 최종 목표온도에 도달하면서 이 적분량이 열손실에 수렴하도록 적분 이득을 정해야 한다. 그러나 기존의 제어에서는 비례 이득에 부적합한 적분 이득을 다시 조정하는 수단이 없으며 오버슈트/언더슈트의 결과를 초래하였다. 또한 제어대상에 부적합한 제어이득을 적용하거나 PID 튜닝이 제한되는 경우에는 제어성능의 저하를 초래한다.

제안한 적응형 제어 기법은 제어대상을 기술하는 모델을 내부에 탑재하고 있다. 이 모델을 이용하여 제어대상의 열손실과 온도 특성을 스스로 검출할 수 있다. 제어대상에 적합한 PID 제어이득을 검출할 수 있으며 결과적으로 오버슈트를 억제하거나 허용범위 이내로 제한하여 목표온도 도달시간을 단축할 수도 있다. 따라서 온도제어에 영향을 주는 인자가 변하는 경우에는 제안한 적응형 제어가 대단히 효과적이다.

다만 논문에서 제안하는 적응형 제어 기법은 열전파 경로가 복잡한 경우에 발생하는 시간 지연의 문제에는 한계가 있다. 추후 시간의 인자를 포함하는 PID 온도제어 연구도 계획한다.

References

[1] Jan Jantzen, “Tuning of fuzzy PID controllers,” Denmark. Tech. Report, no. 98-H 871 (fpid), pp. 1-22, 30 Sep. 1998.

[2] B. Nagaraj, S. Subha, B. Rampriya, “Tuning Algorithms for PID Controller Using Soft Computing Techniques,” *IJCSNS International Journal of Computer Science and Network Security*, Vol. 8, No. 4, pp. 278-281, April 2008.

[3] Kiam Heong Ang, Gregory Chong, Yun Li, “PID Control System Analysis, Design, and Technology,” *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, Vol. 13, No. 4, pp. 559-576, July 2005. DOI: 10.1109/TCST.2005.847331

[4] Karl J. Astrom and T. Hagglund, “*PID Controllers: Theory, design and tuning*,” Instrument society of America, 2nd edition, 1995.

[5] Katsuhiko Ogata, "*Modern Control Engineering," Third edition*, Prentice - Hall Inc., 1997.

[6] Benjamon C. Kuo, "*Digital Control Systems," Sencond edition*, Saunders Colledges Publishing, 1992.

[7] Jens Graf, "*PID Control: Ziegler-Nichols Tuning*," Createspace Independent Pub., 2013.

[8] A. Ortega and J. Juan Rosales, "Newton's law of cooling with fractional conformable derivative," *Revista Mexicana de F´ısica*, Vol. 64, pp. 172-175, March 2018. DOI: https://doi.org/10.31349/RevMexFis.64.172

[9] Charles L. Phillips, H. Troy Nagle, "*Digital Control System Analysis and Design*," Third edition, Prentice-Hall Inc., 1995.

[10] Jin-moon Nam, "Model Identification of Heating Jacket with Heat Loss Characteristics and Verification through Temperature Experiment," *Journal of The Institute of Electronics and Information Engineers*, Scheduled to be published in December 2023.