원자력 전산해석 및 응용

최종 보고서

Improvement of a code for Passive Autocatalytic Recombiner (PAR)

2020.06.24.

1조

김지현, 송준혁, 최규정, TA DUY LONG

목차

제 1 장. 개 요

2011년, 후쿠시마 원자력 발전소에서 쓰나미로 인해 전원완전상실로 인해 냉각수 순환 및 격납건물의 파손으로 인해 방사능 물질이 누출되는 사고가 발상하였다. 이에 소내 전원이 상실되어도 냉각 및 핵연료 피복재의 산화로 인해 발생하는 수소를 제거해줄 수 있는 피동시스템의 역할이 재조명되었다. PAR는 소내 전원이 상실되어도 자연대류 및 촉매반응으로 인해 전원이 없이도 수소를 제거해 줄 수 있는 피동형 시스템이다. 피동형수소재결합기가 중대사고 시 사고완화 역할을 적절히 수행할 수 있는지 예측하기 위해서는 이 시스템의 수소 제거 능력을 평가하는 것이 필요하다. 이를 평가하기 위하여 작년 프로젝트에서는 PAR를 모사하여 화학반응 및 지배방정식을 수치해석적 방법으로 풀어내고자 하였다. 본 프로젝트는 작년의 프로젝트를 향상시켜 PAR를 모사하는 전산코드를 개선하는 것이 목적이다. 이를 개선하기 위해 총 4가지 부분에서의 개선점을 발견하였고 다음과 같다.

첫째, 화학반응을 다양화하여 촉매반응을 정확히 모사하려고 했고, 각 화학반응에 필요한 실험값들을 다양하게 사용할 수 있도록 하였다.

둘째, 지배방정식을 풀 수 있는 수치해석적 방법을 수정하였다. 이전 프로젝트의 알고리즘을 수정하고 time advancing scheme을 이용하여 화학반응이 일어나는 매우 짧은 시간과 유체의 흐름이 일어나는 시간스텝을 적절히 조정하여 이전 프로젝트에서 발생한 문제점을 해결하려고 하였다.

셋째, 각 species의 열역학적 특성에 대한 데이터를 체계적으로 구축하여 다양한 조건에서도 계산할 수 있도록 하였다. 또한, 열역학 테이블에 있는 데이터 사이의 값들을 도출하기 위하여 더 정확한 보간법을 도입하였다.

넷째, 프로그램 실행속도를 더욱 빠르게 개선하였고 사용자가 쉽게 사용할 수 있도록 편리한 GUI를 도입하였다. 이를 통해 계산속도를 단축시키며 컴퓨터의 메모리를 더욱 효율적으로 사용할 수 있게되었다.

제 2 장. 개선 사항

제 1 절. 화학반응식

PAR의 내부에서는 다양한 화학반응이 발생한다. 작년의 경우 대표적인 7가지 화학반응이 일어난다고 가정하였다. 이번 프로젝트에서는 총 30가지의 모든 화학반응을 고려하여 계산을 수행하였다. 화학반응에서 각 species 별 반응율을 계산하여 이후 species equation을 푸는데 사용한다. 화학반응율은 특정 species의 생성율과 감소율을 가지고 구할 수 있으며 아래의 식(1)과 같다.

$$\dot{\omega}_k = \frac{d\omega_k}{dt} = Production\ rate\ - Reduction\ rate \tag{1}$$

본 보고서에서 고려한 species는 총 9가지이며 다음과 같다; H, H_2, O, O_2, OH, H_2O, HO_2, $H2O_2$, N_2. 또한 30가지의 화학반응식과 화학반응율 상수를 구하기 위한 상수값을 아래의 표 1에 나타내었다. 화학반응율 상수는 아래의 식(2)와 같이 구한다.

$$k = AT^n e^{-E/RT} \tag{2}$$

여기서 A의 단위는 [mole-cm-Kelvin-sec], E의 단위는 [kJoule/mole]이고 R은 기체상수를 나타낸다. 각 화학반응에 사용되는 상수값은 실제 실험을 통하여 얻어지며 본 프로젝트에서는 GRI 3.0 실험데이터를 사용하였다. 해당 실험에서 누락된 데이터는 다른 실험 데이터를 이용하였다 [1].

표 1. Chemical Reaction Mechanisms

	Chemical Reactions	A	n	E
	H_2/O_2 reactions			
1	$H + O_2 = O + OH$	2.65×10^{16}	-0.67	71.30
	$O + OH = H + O_2$	-	-	-
2	$O + H_2 = H + OH$	3.87×10^{4}	2.70	26.19
3	$H_2 + OH = H_2O + H$	2.16×10^{8}	1.51	14.35
4	$OH + OH = O + H_2O$	3.57×10^{4}	2.40	-8.83
	$O + H_2O = OH + OH$	-	-	-
5	$H_2 + O_2 = OH + OH$	-	-	-
	H_2O_2 dissoc.-recombination			
6	$H + H + M = H_2 + M$	1.00×10^{18}	-1.00	0.00
	$H_2 + M = H + H + M$	-	-	-
7	$H + H + H_2 = 2H$	9.00×10^{16}	-0.60	0.00
8	$H + H + H_2O = H_2 + H_2O$	6.00×10^{19}	-1.25	0.00
9	$O + O + M = O_2 + M$	1.20×10^{17}	-1.00	0.00
10	$O + H + M = OH + M$	5.00×10^{17}	-1.00	0.00
11	$H + OH + M = H_2O + M$	2.20×10^{22}	-2.00	0.00
	HO_2 formation-consumption			
12	$H + O_2 + M = HO_2 + M$	2.80×10^{18}	-0.86	0.00
13	$H + 2O_2 = HO_2 + O_2$	2.08×10^{19}	-1.24	0.00
14	$H + O_2 + H_2O = HO_2 + H_2O$	1.13×10^{19}	-0.76	0.00
15	$H + O_2 + N_2 = HO_2 + N_2$	2.60×10^{19}	-1.24	0.00
16	$HO_2 + H = H2 + O2$	4.48×10^{13}	0.00	4.47
17	$HO_2 + H = OH + OH$	8.40×10^{13}	0.00	2.66
18	$HO_2 + H = H_2O + O$	3.97×10^{12}	0.00	2.81
19	$HO_2 + O = OH + O_2$	2.00×10^{13}	0.00	0.00
20	$HO_2 + OH = H_2O + O_2$	1.45×10^{13}	0.00	-2.09
	H_2O_2 formation-consumption			
21	$2HO_2 = H_2O_2 + O_2$	1.30×10^{11}	0.00	-6.82
22	$H_2O_2 + M = OH + OH + M$	-	-	-
23	$OH + OH + M = H_2O_2 + M$	2.30×10^{18}	-0.90	-7.12
24	$H_2O_2 + H = H_2O + OH$	1.00×10^{13}	0.00	15.06
25	$H_2O_2 + H = H_2 + HO_2$	1.21×10^{7}	2.00	21.76
26	$H_2O_2 + O = OH + HO_2$	9.63×10^{6}	2.00	16.74
27	$H_2O_2 + OH = H_2O + HO_2$	2.00×10^{12}	0.00	1.79

각 species 별 생성율과 감소율은 아래의 표 2에 정리하였다.

표 2. Species 별 반응

Species	Production reaction	Reduction reaction
H	2, 3, 28, 30	1, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 24, 25
H_2	6, 7, 8, 16, 25	2, 3, 5, 30
O	1, 4, 18	2, 9, 10, 19, 26, 28
O_2	9, 13, 16, 19, 20, 21, 28	5, 12, 13, 14, 15
OH	1, 2, 5, 10, 17, 19, 22, 24, 26	4, 11, 20, 23, 27, 28
H_2O	3, 4, 8, 11, 14, 18, 20, 21, 24, 29	8, 14, 29
HO_2	12, 13, 14, 15, 25, 26, 27	16, 17, 18, 19, 20, 21
H_2O_2	23	22, 24, 25, 26, 27
N_2	15	15

제 2 절. 수치해석 방법

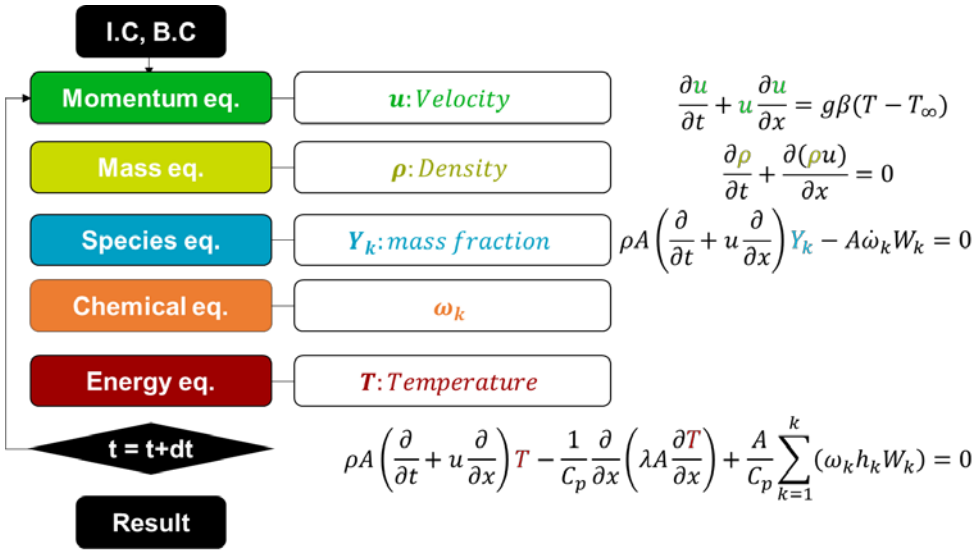

그림 1. 전체적인 알고리즘

위 그림은 지배방정식을 풀기위한 전체적인 알고리즘을 나타낸다.

제 1 항. Momentum equation

비선형 방정식을 풀기 위하여 forward-time scheme 을 사용하였다.

1단계 (solve the prediction equation, u_i^*)

$$\frac{u_i^*-u_i^n}{\Delta t}+u_i^n\frac{u_i^n-u_{i-1}^n}{\Delta x}=g\beta(T_i^n-T_\infty) \tag{3}$$

식(3)은 차분식을 나타내며, prediction velocity의 해는 다음과 같이 식(4)를 통해 구할 수 있다.

$$u_i^*=g\beta(T_i^n-T_\infty)\Delta t+u_i^n\left(1-\frac{\Delta t}{\Delta x}[u_i^n-u_{i-1}^n]\right) \tag{4}$$

2단계 (solve the correction equation, u_i^{n+1})

$$\frac{1}{2}\left[\frac{u_i^{n+1}-u_i^*}{\Delta t}+\frac{u_i^{n+1}-u_i^n}{\Delta t}\right]+u_i^n\frac{u_i^n-u_{i-1}^n}{\Delta x}=g\beta(T_i^n-T_\infty) \tag{5}$$

식(5)은 차분식을 나타내며, 다음 시간 스텝의 correction velocity의 해는 다음과 같이 식(6)를 통해 구할 수 있다.

$$u_i^{n+1}=\frac{u_i^*}{2}+g\beta(T_i^n-T_\infty)\Delta t+u_i^n\left(\frac{1}{2}-\frac{\Delta t}{\Delta x}[u_i^n-u_{i-1}^n]\right) \tag{6}$$

Forward-time backward-space scheme 을 사용하여 momentum equation을 모든 mesh에 대하여 동시에 풀 수 있다.

제 2 항. Continuity equation

Momentum equation 으로부터 구한 velocity (u) 를 사용하여 continuity equation을 semi-implicit scheme 중 Crank-Nicholson 방법을 통하여 ρ^{n+1} 의 값을 구할 수 있다. 첫 번째 단계는 explicit scheme을 사용하고 두 번째 단계는 implicit scheme을 사용한다.

$$\left(1+\frac{\Delta t}{2\Delta x}u_j^{n+1}\right)\rho_j^{n+1}=\frac{\Delta t}{2\Delta x}u_{j-1}^{n+1}\rho_{j-1}^{n+1}+\left(1-\frac{\Delta t}{2\Delta x}u_j^{n}\right)\rho_j^{n}+\frac{\Delta t}{2\Delta x}u_{j-1}^{n}\rho_{j-1}^{n} \tag{7}$$

제 3 항. Species equation

Species equation은 predictor-corrector scheme을 사용하여 풀었다.

1단계 (solve the predictor, Y_j^*)

$$\frac{Y_j^*-Y_j^n}{\Delta t}=-u_j^n\frac{Y_j^n-Y_{j-1}^n}{\Delta x}+\frac{A\dot{\omega}_k W_k}{\rho_j^n A} \tag{8}$$

식 (8)은 Y_j^*를 풀기위한 차분식이며, Y_j^*는 식(9)로 구할 수 있다.

$$\frac{1}{\Delta t}Y_j^*=\left(\frac{1}{\Delta t}-\frac{u_j^n}{\Delta x}\right)Y_j^n+\frac{u_j^n}{\Delta x}Y_{j-1}^n+\frac{A\dot{\omega}_k W_k}{\rho_j^n A} \tag{9}$$

2단계 (solve the corrector, Y_j^{n+1})

$$\frac{Y_j^{n+1}-Y_j^n}{\Delta t}=-\frac{1}{2}\left(u_j^n\frac{Y_j^n-Y_{j-1}^n}{\Delta x}-\frac{A\dot{\omega}_k W_k}{\rho_j^n A}\right)-\frac{1}{2}\left(u_j^n\frac{Y_j^*-Y_{j-1}^*}{\Delta x}-\frac{A\dot{\omega}_k W_k}{\rho_j^{n+1} A}\right) \tag{10}$$

식 (10)은 Y_j^{n+1}를 풀기위한 차분식이며, Y_j^{n+1}는 식(11)로 구할 수 있다.

$$Y_j^{n+1}=Y_j^*-\Delta t\{\frac{u_j^n}{\Delta x}\left(Y_j^*-Y_{j-1}^*\right)+\frac{\dot{\omega}W_k}{\rho_j^n}\} \tag{11}$$

제 4 항. Energy equation

지배방정식에서 diffusion velocity를 무시하고 간단화하였다. 차분방법으로는 backward-time centered-space method (BTCS scheme)을 사용하였고, 이차 미분에만 적용하였다. 일차 미분에는 backward-time backward-space scheme을 사용하였다. 이는 식 (12)에 표현하였다.

$$\rho A\left(\frac{\partial}{\partial t}+u\frac{\partial}{\partial x}\right)\mathrm{T}-\frac{1}{c_p}\frac{\partial}{\partial x}\left(\lambda A\frac{\partial T}{\partial x}\right)+\frac{A}{C_p}\sum_{k=1}^{k}(\omega_k h_k W_k)=0 \tag{12}$$

$$\rho_j^{n+1}A\frac{T_j^{n+1}-T_j^n}{\Delta t}+\rho_j^{n+1}Au_j^{n+1}\frac{T_j^{n+1}-T_{j-1}^n}{\Delta x}-\frac{\lambda A}{C_p}\frac{T_{j+1}^{n+1}-2T_j^{n+1}+T_{j-1}^{n+1}}{\Delta x^2}+\frac{A}{C_p}\sum_{k=1}^{k}(\omega_k h_k W_k)=0 \tag{13}$$

식(11)은 차분식을 표현하며 식 (14)는 해를 구하기위한 식이다.

$$\left(-\frac{\rho_j^{n+1}Au_j^{n+1}}{\Delta x}-\frac{\lambda A}{C_p\Delta x^2}\right)T_{j-1}^{n+1}+\left(\frac{\rho_j^{n+1}A}{\Delta t}+\frac{\rho_j^{n+1}Au_j^{n+1}}{\Delta x}+\frac{2\lambda A}{C_p\Delta x^2}\right)T_j^{n+1}+\left(-\frac{\lambda A}{C_p\Delta x^2}\right)T_{j+1}^{n+1}$$
$$=\frac{\rho_j^{n+1}A}{\Delta t}T_j^n-\frac{A}{C_p}\sum_{k=1}^{K}\omega_k h_k W_k \tag{14}$$

모든 연립 선형 방정식은 LU 분해법을 사용하여 해를 구했다.

제 3 절. 열역학적 특성

작년의 프로젝트에서 사용한 7가지 화학반응식에서는 7가지 화학식 H2, O, O2, OH, H2O, HO2, N2를 이용하였다. 하지만 열역학적 값은 7가지 중 H2, O2, H2O 세가지의 값만 사용하였으며 나머지 화학식에 대해서는 열역학적 값으로 0을 지정하였다. 이 경우 엔탈피를 0으로 두었기 때문에 화학반응식으로 인해 발생하는 열을 감소시키는 결과를 나타낼 수 있으며 또한 온도의 전도율과 비열도 energy equation에 영향을 미칠 수 있다. 따라서 각 기체에 해당하는 값을 추가하였으며 이번 team project에서는 반응식을 7개에서 30개로 확장하였기에 그에 필요한 H2O2의 열역학 값 또한 고려하였다. 초기에는 CO2에 관한 화학반응식도 고려하였기 때문에 그에 필요한 CO, CO2의 열역학 값도 조사하였으며 코드에서 사용되지는 않지만 함수에 값을 입력해 두어서 이후 화학반응식을 추가한다면 이용할 수 있도록 하였다.

열역학 값은 주로 NIST의 fluid graph에서 가져왔다. PAR 모델은 1기압의 등압상태로 가정할 수 있기 때문에 NIST 사이트에서 'isobaric' 옵션을 선택하면 원하는 온도에서 열역학적 값을 얻을 수 있다. 그래프에서 얻을 수 있는 다양한 값들 중에서 프로젝트에 사용되는 값은 엔탈피, 비열(열용량), 열전도율의 세가지 특성이다. 각 기체의 밀도는 이상기체방정식과 기체의 화학식량을 이용해서 계산하였기 때문에 따로 자료를 구하지 않았다. 온도범위는 작년의 프로젝트에서 시뮬레이션 결과 나타난 온도 중 최고온도를 기준으로 삼았다. 500K를 초기 온도로 두었을 때 약 1100K까지 온도가 상승하는 결과를 확인하였으며 따라서 여유 있게 1500K까지 열역학적 값을 조사하였다. 시작점은 물의 끓는점인 398K부터 의미가 있다고 생각하였기에 400K으로 설정하였으며 100K마다 값을 조사하여 테이블로 정리하였다. 하지만 NIST의 fluid graph에는 기체마다 열역학적 값을 나타낸 온도 범위가 다르기 때문에 기체별로 1500K까지의 열역학적 값을 조사할 필요가 있었다. 예를 들어 N2의 경우 1500K 이상의 값도 얻을 수 있지만, CO는 500K, H2와 O2는 1000K까지만 나타나 있다. 따라서 1500K 까지의 열역학 데이터 표를 작성하기 위해 NIST에서 화학식 별로 제공되는 'Gas phase thermochemistry data'를 참고하였다. 이 데이터 표에서는 온도별로 ΔH, C_p가 나타나 있으며, fluid graph에서 등압조건을 주고 얻었던 데이터와 비교했을 때 엔탈피의 변화와 열용량이 거의 같은 값을 나타냈기 때문에 해당하는 값을 이용해서 1500K까지 엔탈피와 열용량을 표로 작성하였다.

열전도율의 경우 위에서 사용한 'Gas phase thermochemistry data'에 나타나 있지 않았고 조사 결과 따로 값을 찾지 못했기 때문에 저온에서 얻어진 데이터로부터 선형 계산을 통해 외삽하여 값을 입력하였다. 이것은 데이터를 일차적으로 선형계산한 것이 아니라 데이터의 변화량을 조사하고, 해당 변화량의 변화를 또 조사하여 (마치 n차 미분값과 같이) 그 값이 선형성을 보이는 n번째 값에 대해 선형으로 값을 추측하여 외삽하였다. 예를 들어 데이터의 증가량이 어떤 값을 기준으로 진동하는 경우 이를 평균내서 $f(T+\Delta T)-f(T)$의 값이 일정하다 가정하였지만 O2의 경우에는 변화량이 일정하게 증가하였기에 $f(T+\Delta T)-2f(T)+f(T-\Delta T)$ 의 값이 일정하다고 가정하여 계산하였다.

H와 O의 경우 기체상태로 불안정하기 때문에 조사결과 값을 얻을 수 없었으며, 프로그램에서는 이들의 열역학 값을 0으로 처리하였다. H2O2와 OH, HO2의 경우는 'Gas phase thermochemistry data'에서는 엔탈피 변화량과 열용량을 얻을 수 있었지만 fluid graph는 NIST에서 얻을 수 없었기 때문에, 400K의 엔탈피 값을 적당한 값으로 가정하여 ΔH를 이용해 엔탈피를 구하였다. 또한 이들의 열전도율도 fluid graph가 없어 얻지 못했기 때문에 비슷한 화학식량을 가진 O2와 H2O에서 구한 열전도율을 그대로 입력하였다. 이러한 열역학 값은 함수에 새로운 값을 입력해주면 쉽게 코드에 적용할 수 있다. 조사한 9가지의 기체의 열역학적 값을 아래 표3에 나타냈다.

표 3. 열역학적 특성값

H2(K)	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)	CO2	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)
400	10.886	29.189	0.23406	400	26.285	41.447	0.025143
500	13.809	29.256	0.2805	500.33	30.612	44.7	0.033518
600	16.738	29.33	0.32809	600.67	35.236	47.381	0.041607
700	19.677	29.46	0.37631	699.83	40.047	49.59	0.049283
800	22.632	29.653	0.42517	800.17	45.12	51.459	0.056722
900	25.61	29.908	0.47467	900.5	50.363	53.023	0.063834
1000	28.616	30.222	0.5248	999.67	55.688	54.318	0.07055
1100	31.656	30.58	0.57556	1100	61.194	55.42	0.077042
1200	34.736	30.99	0.62695	1200	66.774	56.35	0.08322
1300	37.856	31.42	0.67897	1300	72.454	57.14	0.089084
1400	41.016	31.86	0.73162	1400	78.194	57.83	0.094635
1500	44.226	32.3	0.7849	1500	84.014	58.4	0.099871
H2O	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)	CO	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)
400	49.187	36.198	0.02702	400	15.371	29.365	0.033106
449.33	50.953	35.597	0.031115	450	16.844	29.554	0.03621
500.62	52.78	35.703	0.035926	499.83	18.323	29.806	0.039262
599.79	56.357	36.513	0.046345	600	21.333	30.47	0.045314
700.42	60.085	37.598	0.058015	700	24.423	31.17	0.051262
799.58	63.871	38.774	0.070332	800	27.573	31.88	0.057106
900.21	67.835	40.027	0.083494	900	30.793	32.55	0.062846
1000.8	71.927	41.304	0.097202	1000	34.083	33.18	0.068482
1100	76.085	42.554	0.11115	1100	37.423	33.73	0.074014
1200.6	80.43	43.789	0.12567	1200	40.823	34.2	0.079442
1300	84.86	44.94	0.14019	1300	44.263	34.55	0.084766
1400	89.41	46.06	0.15471	1400	47.733	34.93	0.089986
1500	94.07	47.11	0.16923	1500	51.243	35.22	0.095102
OH	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)	O2	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)
400	42.02706	29.65	0.02702	400	11.701	30.132	0.034689

500	44.97706	29.51	0.035926	500	14.762	31.108	0.04275
600	47.92706	29.52	0.046345	600	17.923	32.103	0.050738
700	50.88706	29.67	0.058015	700	21.179	32.992	0.058499
800	53.86706	29.92	0.070332	800	24.517	33.742	0.065933
900	56.87706	30.27	0.083494	900	27.923	34.363	0.072997
1000	59.92706	30.67	0.097202	1000	31.386	34.873	0.079685
1100	63.00706	31.12	0.11115	1100	34.896	35.29	0.086003
1200	66.14706	31.59	0.12567	1200	38.446	35.66	0.091951
1300	69.32706	32.05	0.14019	1300	42.026	35.99	0.097529
1400	72.54706	32.5	0.15471	1400	45.646	36.28	0.102737
1500	75.82706	32.95	0.16923	1500	49.286	36.55	0.107575
H2O2	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)	HO2	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)
400	18.5736	48.65	0.034689	400	5.772001	37.43	0.034689
500	23.6336	52.51	0.04275	500	9.632001	39.68	0.04275
600	29.0436	55.5	0.050738	600	13.702	41.68	0.050738
700	34.7236	57.92	0.058499	700	17.952	43.45	0.058499
800	40.6136	59.9	0.065933	800	22.382	45.03	0.065933
900	46.6836	61.55	0.072997	900	26.952	46.42	0.072997
1000	52.9136	62.95	0.079685	1000	31.662	47.66	0.079685
1100	59.2736	64.17	0.086003	1100	36.482	48.75	0.086003
1200	65.7436	65.27	0.091951	1200	41.412	49.73	0.091951
1300	72.3236	66.3	0.097529	1300	46.422	50.6	0.097529
1400	79.0036	67.33	0.102737	1400	51.522	51.39	0.102737
1500	85.7936	68.42	0.107575	1500	56.702	52.11	0.107575
N2	H(kJ/mol)	Cp(J/mol*K)	λ(W/m*K)				
400	11.638	29.273	0.032205				
500	14.58	29.594	0.038143				
600	17.564	30.118	0.043917				
700	20.607	30.761	0.049605				
800	23.717	31.439	0.055197				
900	26.894	32.096	0.060666				
1000	30.135	32.703	0.065991				
1100	33.433	33.248	0.07116				
1200	36.782	33.729	0.076173				
1300	40.177	34.152	0.081038				
1400	43.611	34.522	0.085763				
1500	47.08	34.846	0.090361				

프로그램에서는 위의 데이터를 기반으로 원하는 온도에서 3가지 열역학 값을 구할 수 있는 함수를 작성해야 한다. 이를 위해서는 일반적으로 보간법이나 데이터 피팅(Fitting) 함수가 필요하다. 작년 프로젝트를 조사한 결과 데이터를 8차 다항식을 이용해서 피팅하여 사용했으며 이를 그래프로 나타내면 그림2와 같다.

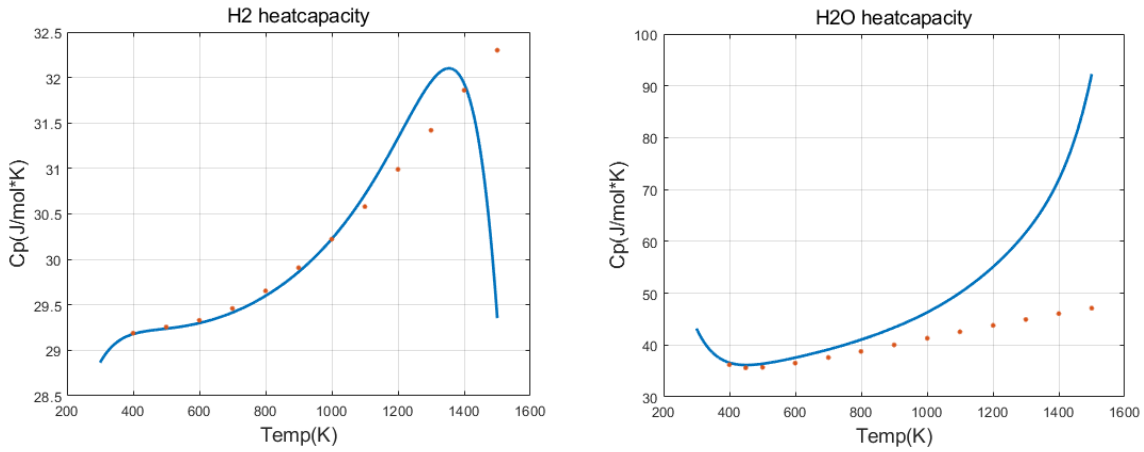

그림 2. 8차 다항식을 이용한 데이터 피팅

그래프에서 나타난 빨간 점이 실제 데이터 값이며, 파란 선은 이를 피팅한 8차 다항식이다. 그래프를 통해 특정구간에서는 실제 데이터와 비슷하지만 해당 구간을 벗어나면 급격하게 그래프가 휘어지며 큰 오차를 발생시킬 수 있다는 것을 확인할 수 있다. 또한 데이터를 피팅하는 함수의 특성 상 조사한 데이터를 함수가 정확히 지나지 않는다는 단점도 가지고 있다. 따라서 우리 팀에서는 데이터 피팅 대신 데이터 사이의 값을 구하는 보간법을 사용하기로 하였으며, Monotone cubic interpolation(MCI)를 이용하기로 하였다. 이 함수는 Hermite polynomial을 이용하는 Cubic Hermite spline에 대해 몇 가지 조건을 추가하여 데이터의 단조성을 보존하도록 하는 보간법이다. 열역학적 데이터의 경우 일반적으로 단조 증가하거나 단조 감소하는 경우가 많다는 것을 확인하였는데, 일반적인 spline 보간법의 경우 데이터가 진동할 위험이 크기 때문에 단조성을 보존 가능한 MCI가 적절하다고 판단하였다. MCI는 다음과 같이 적용할 수 있다.

우선 각 데이터를 $(x_k, y_k),\ k = 1, \dots, n$ 이라 하면, 데이터 사이의 기울기는 다음과 같다.

$$\delta_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}, \qquad for\ k = 1, \dots, n-1$$

기울기의 평균값 m_k는 다음과 같이 정의된다.

$$m_k = \frac{\delta_{k-1} + \delta_k}{2}, \qquad for\ k = 2, \dots, n-1$$

$$m_1 = \delta_1, \qquad and \qquad m_n = \delta_{n-1}$$

이때 (1) 인접한 δ에 대해 $\delta_{k-1}\delta_k < 0$ 이면 (반대 부호를 가짐) $m_k = 0$ 이다. 또한 (2) $\delta_k = 0$ 이면 $m_k = m_{k+1} = 0$ 이다. 새롭게 정의되는 변수는 다음과 같으며

$$\alpha_k = m_k/\delta_k$$

$$\beta_k = m_{k+1}/\delta_k$$

이에 대해 (3) $\alpha_k < 0$이면 $m_k = 0$, $\beta_k < 0$이면 $m_{k+1} = 0$ 이다. 또한 (4) $\alpha_k^2 + \beta_k^2 > 9$ 이면 다음과 같다.

$$m_k = \tau_k \alpha_k \delta_k \quad and \quad m_{k+1} = \tau_k \beta_k \delta_k, \qquad for \quad \tau_k = \frac{3}{\sqrt{\alpha_k^2 + \beta_k^2}}$$

이 4가지 조건을 통해 기존의 Cubic Hermite spline을 이용한 보간법이 개선되어 새로운 보간 함수가 단

조성을 가지게 되는 결과를 나타난다. 위의 변수를 이용해 결과적으로 보간 함수를 얻을 수 있다.

$$f_{interpolated}(x) = y_k \cdot h_{00}(t) + \Delta \cdot m_k \cdot h_{10}(t) + y_{k+1} \cdot h_{01}(t) + \Delta \cdot m_{k+1} \cdot h_{11}(t)$$

$$\Delta = x_{k+1} - x_k, \qquad t = \frac{x - x_k}{\Delta}$$

여기서 함수 h_{ii}는 cubic Hermite spline의 기저 함수(basis functions)이며, 다음과 같이 정의된다.

$$\mathrm{h}_{00}(t) = 2t^3 - 3t^2 + 1, \qquad h_{10}(t) = t^3 - 2t^2 + t,$$

$$\mathrm{h}_{01}(t) = -2t^3 + 3t^2, \qquad h_{11}(t) = t^3 - t^2$$

위의 식을 이용해 C++코드로 함수를 작성하였다. 몇 가지 데이터에 대해 보간 함수가 정상 작동하는 것을 확인하였으며, 이후 C#으로 작성한 프로그램에 코드를 변환하여 입력하였다. 새로운 열역학 데이터와 보간법으로 인해, 우리 팀의 프로그램은 작년 프로젝트와 비교하여 보다 정확한 열역학 값이 코드에 적용될 것이라 기대할 수 있다.

제 4 절. 프로그램 개선 및 GUI

지난 학기에 다른 조들의 프로그램들을 분석해보면 GUI(Graphical User Interface) 기능을 이용하기 위해서 PYTHON, IPLOT, ROOT 등과 같이 별도의 프로그램들을 실행하여야만 했다. 또한, GUI 기능을 추가하면 결과를 얻기까지 걸리는 총 실행시간이 현저하게 저하되는 현상이 발생하였다. 이를 해결하기 위해서 C# .Net Framework를 사용하는 것으로 하였다. C# .Net Framework는 GUI를 구현하는 것이 매우 용이하며 Interface와 Module간 데이터 연동이 우수하여 실행속도가 저하되는 현상을 방지할 수 있기 때문이다. 또한, C++은 언어의 특성상 Memory에 직접적으로 접근할 수 있기에 프로그래밍에 주의가 필요하다. 예를 들어 동적 할당과 동적 할당된 메모리를 해제하는 부분이 그 예이다. C#은 Garbage Collection 기능으로 자동 메모리 관리자 기능으로 메모리에 대한 문제를 해결할 수 있다.

프로그램은 6개의 Form Class와 1개의 일반 Class로 구성되어 있다. Form Class는 Interface, Interface가 동작하도록 하는 Method, 사용자 정의 Method로 구성되어 있다. 일반 Class는 C++, Java와 같은 객체지향언어에서 사용되는 Class의 개념과 동일하다.

제 1 항. SplashForm.cs

로딩화면이며 프로그램이 시작할 때 잠깐 나왔다가 사라진다.

Method	설명
SplashForm()	생성자이며 Interface가 실행될 수 있도록 함
SplashForm_Load(object,EventArgs)	SplashForm이 화면에 나타날 때, loadingThread실행
LoadingThread()	Thread를 통해서 loading을 진행
LoadingStep(int)	progressBar에 loading이 되는 것을 보여줌
SplashFormClose()	loading이 완료되면 SplashForm을 닫음

LoadingProgressDelegate(int)	loading이 진행되는 과정을 progressBar에 보여주기 위해서 delegate를 이용
CloseDelegate()	loading이 완료되면 delegate를 이용

제 2 항. MainForm.cs

SimulationForm을 Tab형식으로 화면에 보여주는 역할을 한다. New Simulation 메뉴를 누르면 여러 시뮬레이션을 동작시킬 수 있다.

Method	설명
MainForm()	생성자이며 Interface가 실행될 수 있도록 함
TsmiNewSimulation_Click(object,EventArgs)	New Simulation 메뉴를 눌렀을 때, 실행되는 method로 simulation 1 ~ simulation N과 같은 양식으로 SimulationForm 생성
TsmiClose_Click(object,EventArgs)	Close메뉴를 눌렀을 때, 실행되는 method로 프로그램을 종료함

제 3 항. SimulationForm.cs

Simulation Class와 SimulationInputForm, SimulationChartForm 사이에서 Controller 역할을 수행한다. 시뮬레이션을 수행할 수 있도록 Simulation형 변수, 사용자로부터 초기 온도 등과 같은 입력 값들을 받아올 수 있도록 하는 변수들을 선언했다. 또한, 결과를 출력하는 Viewer 역할도 수행한다.

Method	설명
SimulationForm()	생성자이며 Interface가 실행될 수 있도록 함
SimulationForm_Load(object,EventArgs)	폼이 화면에 처음 나타날 때, 사용자에게 입력 값들이 입력되지 않았음을 알려줌
TsmiInputValues_Click(object,EventArgs)	Input Values 메뉴를 눌렀을 때, 실행되는 method로 SimulationInputForm을 호출함
TsmiRunSimulation_Click(object,EventArgs)	Run Simulation 메뉴를 눌렀을 때, 실행되는 method로 시뮬레이션이 실행됨
TsmiViewChart_Click(object,EventArgs)	View Chart 메뉴를 눌렀을 때, 실행되는 method로 SimulationChartForm을 호출함
IsReadyDone()	입력 값에 대한 입력 여부를 사용자에게 알려줌

ShowResult()	velocity, temperature, H2 Rate의 결과를 출력하는 각각의 method를 호출
ShowResultPart1()	velocity 결과를 출력하는 method
ShowResultPart2()	temperature 결과물 출력하는 method
ShowResultPart3()	H2 Rate 결과를 출력하는 method

제 4 항. SimulationInputForm.cs

사용자로부터 온도, 수소 농도 등 초기값을 입력받아서 설정하는 form이다. 입력값들은 SimulationForm으로 넘긴다.

Method	설명
SimulationInputForm()	생성자이며 Interface가 실행될 수 있도록 함
BtnOK_Click(object,EventArgs)	입력값을 추출하여서 SimulationForm으로 값을 넘김
BtnCancel_Click(object,EventArgs)	Form을 닫음

제 5 항. SimulationChartForm.cs

SimultaionForm으로부터 velocity, temperature, H2 Rate, timeStep, spaceStep, dt 값들을 넘겨받는다. 받은 값들을 가지고 velocity, temperature, H2 Rate에 대한 결과를 그래프로 출력한다.

Method	설명
SimulationChartForm()	생성자이며 Interface가 실행될 수 있도록 함. 이 생성자는 시뮬레이션이 실행되지 않았을 때, 사용자에게 그냥 Form을 보여주기 위함
SimulationChartForm(double[,],double[,],double[,,],int,int,double)	생성자이며 Interface가 실행될 수 있도록 하며 SimulationForm으로부터 값들을 넘겨받을 수 있도록 함
TsmiClose_Click(object,EventArgs)	Form을 닫음
TsmiInputValues_Click(object,EventArgs)	SimulationChartInputForm을 호출함
ShowResult()	입력받은 timeStep에 대하여 그래프로 결과를 출력할 수 있도록 함

제 6 항. SimulationChartInputForm.cs

SimulationChartForm에서 timeStep에 대한 결과를 보여준다. SimulationChartInputForm은 필요한 timeStep을 입력받을 수 있도록 하는 Form이다.

Method	설명
SimulationChartInputForm()	생성자이며 Interface가 실행될 수 있도록 함.
BtnOK_Click(object,EventArgs)	입력값을 추출하여서 SimulationChartForm으로 값을 넘김
BtnCancel_Click(object,EventArgs)	Form을 닫음

제 7 항. Simulation.cs

Mass, Momentum, Energy, Species, Thermodynamic Properties 등 여러 계산과 반응들이 수행된다. 시뮬레이션을 수행하기 위해서 필요한 변수들은 생성자와 내부함수로 초기화 및 대입이 이루어진다.

Method	설명
Simulation(int,int,double,double)	SimulationForm으로부터 값을 넘겨받으며 변수 초기화 및 대입
InitSetting(double,double,double,double)	SimulationForm으로부터 값을 넘겨받으며 변수 초기화 및 대입
Run(int,double)	여러 반응과 계산식들을 호출하는 최상위 Method
CalculateGoverningEquation1(int)	Momentum, Continuity를 계산하도록 해당 Method를 호출
CalculateGoverningEquation2(int)	Energy계산과 온도를 업데이트
CalculateChemicalReaction(int,int)	분자별로 화학반응식을 계산
CalculateSpecies(int)	30가지 화학반응이 수행되는 Method
CalculateHeatCapacity(int,int)	Heat Capacity를 계산
CalculateEachCapacity(int,int,int)	MonotonicCubicHermiteSpline으로 species의 Heat Capacity를 계산
CalculateThermalConductivity(int,int)	Thermal Conductivity를 계산
CalculateEachThermalConductivity(int,int,int)	MonotonicCubicHermiteSpline으로 species의 Thermal Conductivity를 계산

CalculateEnthalpy(int,int)	Enthalpy를 계산
CalculateEachEnthalpy(int,int,int)	MonotonicCubicHermiteSpline으로 species의 Enthalpy를 계산
LUdecomposition(double[,],double[],int)	LU 분해법이 이루어지는 함수
GetH2()	H2 Rate에 대한 Getter Method
GetU()	velocity에 대한 Getter Method
GetTemperature()	temperature에 대한 Getter Method
MonotonicCubicHermiteSpline(double,double[],double[])	MonotonicCubicHermiteSpline 계산 수행
H00(doulbe), H10(doulbe), H01(doulbe), H11(doulbe)	MonotonicCubicHermiteSpline 계산을 수행하는데 필요한 보조 식
Momentum(int)	Momentum을 계산
Continuity(int)	Continuity를 계산
Species(int)	Species를 계산

프로그램을 더 발전시키기 위해서 접근한 GUI 개선과 실행 속도 향상이라는 목표를 달성하기 위해서 최대한으로 프로그래밍을 진행하였다. GUI는 사람마다 선호하는 것이 다르기 때문에 좋은 UI/UX를 구성하는 것은 무리가 있었다. 하지만 CLI보다는 확실히 더 좋은 UI/UX이기에 진행하고자 했던 목표에는 어느 정도 달성했다고 생각한다. 또한, CLI에서는 결과를 그래프로 출력하는 것이 쉽지 않아서 다른 프로그램들을 이용하는 경우들이 있는데 본 프로젝트는 그래프를 결과를 출력하는 것까지 가능하다. 다른 프로그램과의 연동이 없이 한 프로그램 내에서 입력부터 결과까지 이루어지므로 인간오류의 확률을 낮추어 주고 결과를 얻기까지 소요되는 총 실행시간을 줄일 수 있다.

개선된 Numerical Method, Thermodynamic Properties, Chemical Reaction에 대해서 프로그래밍을 진행할 때, 개선되고 변경된 사항들이 정상적인 결과를 나타내는지 확인하지 않은 것은 프로그래머가 하여야 할 일을 제대로 하지 않았다고 생각한다. 충분히 좋은 결과를 얻었을 수 있었을 거라고 생각했지만 할 일을 제대로 하지 않아서 결과를 얻지 못했다고 생각한다.