HTP50cork 코드 개발 보고서

1D 비정상 열전달 및 열분해(Pyrolysis) 수치해석 + GUI 응용 프로그램

문서번호	HTP50-DEV-RPT-001
버전	v1.0
작성일	2026-02-22
작성자	(작성자명 기입)
소속/제출처	(소속/제출처 기입)

제출용 문서

결재

작성	검토	승인	비고

개정 이력

버전	날짜	작성자	변경 내용
v1.0	2026-02-22	(작성자)	초안 작성(이론 중심 + GUI 설명 + 부록 구성)

Contents

List of Figures

List of Tables

Chapter 1

개발 목적 및 개요

1.1 프로젝트 개요

HTP50cork는 **코르크(P50) + 금속 기판** 이중층 구조의 **1차원 비정상(Transient) 열전달**을 수치적으로 해석하고, **열분해(Pyrolysis)로 인한 비가역 물성 변화**를 포함하여 열응답을 예측하는 데스크톱 GUI 응용 프로그램이다.

본 프로젝트의 설계 의도는 다음과 같다.

- 열전달 지배방정식을 공간 이산화하여 ODE 시스템으로 구성(Method of Lines).
- 열분해의 비가역성을 상태 변수 $T_{\max}$로 추적하여 물성치 동결(Freeze)을 구현.
- 강성(Stiff) 문제에 대응하기 위해 BDF 기반 내재적 시간 적분 및 희소 야코비안(sparsity)을 적용.
- 사용자가 재료/경계 데이터를 CSV로 로드 및 편집하고, 결과를 그래프로 확인/내보내기할 수 있는 GUI 제공.

1.2 개발 범위

1.2.1 물리 모델

- 해석 차원: 1D (두께 방향 x)
- 구조: 코르크층(두께 L_1) + 금속층(두께 L_2)
- 경계조건:
  - 표면($x = 0$): 대류 열전달 $q_{\text{conv}} = h(t)\,[T_r(t) - T_s]$
  - 후면($x = L_1 + L_2$): 단열(adiabatic), $\partial T/\partial x = 0$
- 열분해 모델:

- Simple 모드: C_p, k 비가역 동결(냉각 시 $T_{\max}$ 기준)
- Advanced 모드: Simple + 질량 프로파일 기반 밀도 감소 $\rho(T_{\max})$

1.2.2 소프트웨어 범위

- GUI: 입력(Geometry/Pyrolysis/데이터), 실행, 진행률, 결과 시각화(3개 그래프)
- 데이터: CSV 로드/편집, 사용자 기본값(JSON) 저장
- 결과: CSV/PNG 내보내기

1.3 해석 영역 개념도

그림 1.1는 이중층 구조, 격자 분할 및 경계조건을 개념적으로 나타낸 것이다.

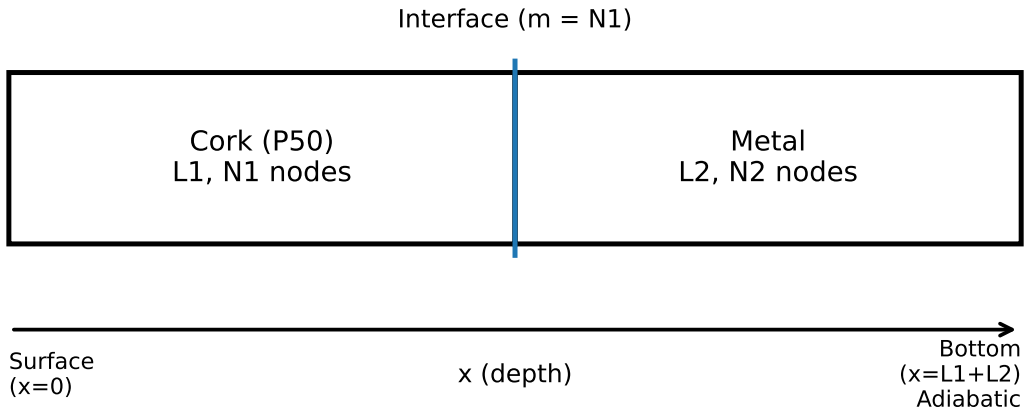

Figure 1.1: 코르크–금속 이중층 1D 해석 영역 및 경계조건 개념도

Chapter 2

수치해석 이론 및 모델 정식화

2.1 지배 방정식

본 해석은 온도 의존 물성치($\rho(T)$, $C_p(T)$, $k(T)$)를 고려한 1차원 비정상 열전도 방정식을 사용한다.

$$\rho(T)\, C_p(T)\, \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k(T) \frac{\partial T}{\partial x} \right) \tag{2.1}$$

2.2 격자 및 이산화(Method of Lines)

해석 영역은 코르크층(L_1, N_1 노드)과 금속층(L_2, N_2 노드)으로 구성되며, 각 영역은 균일 격자로 분할한다.

$$\Delta x_1 = \frac{L_1}{N_1}, \qquad \Delta x_2 = \frac{L_2}{N_2}, \qquad N = N_1 + N_2$$

계면 노드 인덱스는 $m = N_1$로 정의한다.

공간 이산화 이후, 각 노드 온도 $T_i(t)$에 대해 ODE 형태로 변환하여 시간 적분을 수행한다(Method of Lines).

2.3 경계/내부/계면 노드 에너지 보존식

2.3.1 표면 경계 노드(대류 + 내부 전도)

표면 노드($i = 1$ 또는 구현 상 $i = 0$)는 대류 열유속과 내부 전도 열유속의 균형으로 표현된다.

$$\rho_1 C_{p,1}(T_1) \left(\frac{\Delta x_1}{2} \right) \frac{dT_1}{dt} = h(t)\,(T_r(t) - T_1) + \bar{k}_{1\to 2} \frac{T_2 - T_1}{\Delta x_1} \tag{2.2}$$

2.3.2 내부 노드(코르크/금속 공통 형태)

내부 노드는 좌/우 인접 노드로의 전도 플럭스를 이용해 다음과 같이 표현된다.

$$\rho C_p(T_i)\,\Delta x\,\frac{dT_i}{dt} = \bar{k}_{i-1\to i}\frac{T_{i-1}-T_i}{\Delta x} + \bar{k}_{i\to i+1}\frac{T_{i+1}-T_i}{\Delta x} \tag{2.3}$$

2.3.3 이종 재료 계면 노드

계면은 코르크와 금속의 물성이 동시에 영향을 주므로, 계면 제어체적의 열용량을 양측 half-cell의 합으로 구성한다.

$$\left(\rho_1 C_{p,1}\frac{\Delta x_1}{2} + \rho_2 C_{p,2}\frac{\Delta x_2}{2}\right)\frac{dT_m}{dt} = \bar{k}_{m-1\to m}\frac{T_{m-1}-T_m}{\Delta x_1} + \bar{k}_{m\to m+1}\frac{T_{m+1}-T_m}{\Delta x_2} \tag{2.4}$$

2.3.4 후면 단열 경계 노드

후면($x = L_1 + L_2$)은 단열 조건이므로 열유속이 0이며, 내부 전도만 반영된다.

$$\rho_2 C_{p,2}(T_N)\left(\frac{\Delta x_2}{2}\right)\frac{dT_N}{dt} = \bar{k}_{N-1\to N}\frac{T_{N-1}-T_N}{\Delta x_2} \tag{2.5}$$

2.4 유효 열전도도(조화 평균)

노드 계면의 유효 열전도도는 온도 의존성을 고려해 조화 평균을 사용한다.

$$\bar{k}_{i\to i+1} = \frac{2k(T_i)k(T_{i+1})}{k(T_i)+k(T_{i+1})} \tag{2.6}$$

2.5 해석상의 단위

기본 단위계는 SI를 사용한다(길이 m, 시간 s, 온도 °C 또는 K, k는 W/(m K), C_p는 J/(kg K), ρ는 kg/m^3).

Chapter 3

열분해(Pyrolysis) 모델(이력 및 비가역성)

3.1 열분해 모델링의 핵심: 상태 변수 $T_{\max}$

열분해는 비가역 반응이므로, 현재 온도 T뿐 아니라 과거 최대 도달 온도 $T_{\max}$를 추적해야 한다.

$$T_{\max}(t) = \max_{0 \le \tau \le t} \{T(\tau)\} \tag{3.1}$$

3.2 $T_{\max}$의 ODE화(수치적 매끄러움 확보)

$T_{\max}$를 단순 max 연산으로 갱신하면 미분 불연속이 발생할 수 있으므로, 수치 안정성을 위해 ODE 형태로 연립하여 풀이한다. 핵심 아이디어는 (1) 온도가 상승할 때만, (2) $T > T_{\max}$ 일 때만 $T_{\max}$가 증가하도록 하는 것이다.

$$\frac{dT_{\max}}{dt} = \underbrace{\max\left(0, \frac{dT}{dt}\right)}_{\text{ramp}} \cdot \underbrace{\sigma(T - T_{\max})}_{\text{smooth switch}} \tag{3.2}$$

여기서 $\sigma(\cdot)$는 시그모이드(로지스틱) 게이트 함수이며, 불연속 `if-else`를 대체하여 강성 증가를 완화한다.

3.3 물성치 비가역 동결(Property Freezing)

3.3.1 동결 계수(게이트) 정의

코르크 P50은 열분해 임계 온도 T_{crit}를 초과한 후 냉각 단계로 들어가면, 물성치가 virgin 상태로 회복되지 않는다. 이를 모사하기 위해 다음의 동결 계수 λ를 정의한다.

$$\lambda = \sigma(T_{\max} - T_{\text{crit}})\ \sigma(T_{\max} - T) \tag{3.3}$$

첫 번째 게이트는 "열분해가 시작되었는가"(임계 온도 초과), 두 번째 게이트는 "현재 냉각 중인가"를 판단한다.

3.3.2 유효 물성치 결정

동결 계수로 최종 유효 물성치 ϕ_{eff}를 선형 보간하여 정의한다.

$$\phi_{\text{eff}}(T, T_{\max}) = (1-\lambda)\,\phi_{\text{table}}(T) + \lambda\,\phi_{\text{table}}(T_{\max}) \tag{3.4}$$

여기서 $\phi \in \{k,\, C_p\}$이다. 즉, $\lambda \simeq 1$인 경우(열분해 후 냉각) k, C_p는 $T_{\max}$에서의 값으로 고정된다.

3.4 밀도 감소 모델(Advanced 모드)

3.4.1 Simple 모드

Simple 모드에서는 밀도 변화를 무시하고 ρ를 상수(혹은 테이블 값)로 둔다.

$$\rho(T) = \rho_{\text{virgin}} \quad (\text{constant}) \tag{3.5}$$

3.4.2 Advanced 모드: 질량 프로파일 기반 $\rho(T_{\max})$

Advanced 모드에서는 열분해 가스 방출에 따른 질량 손실을 고려한다. 고정 격자(fixed-grid) 가정 하에서 부피 변화가 없으므로, 질량 변화가 곧 밀도 변화로 해석된다.

$$\rho_{\text{eff}}(T_{\max}) = \rho_{\text{virgin}} \times \frac{M(T_{\max})}{100} \tag{3.6}$$

여기서 $M(T_{\max})$는 TGA 기반 정규화 질량(%) 프로파일이다.

3.5 최종 연립 시스템

열분해 모델을 포함하면 최종 상태는 다음과 같이 $2N$ 차원의 ODE 시스템이 된다.

$$\frac{d}{dt}\begin{bmatrix}\mathbf{T}\\ \mathbf{T}_{\max}\end{bmatrix} = \mathbf{F}(t, \mathbf{T}, \mathbf{T}_{\max}) \tag{3.7}$$

Chapter 4

수치 해법 및 안정성 설계

4.1 강성(Stiffness)과 시간 적분법 선택

열분해 모델은 임계 온도 부근에서 물성 변화가 급격하고, 계면에서 물성 대비가 커서 시간 스케일이 분리되기 쉽다. 따라서 명시적(explicit) 방법은 매우 작은 시간 스텝을 요구할 수 있으며, 본 프로젝트는 강성 문제에 적합한 BDF(Backward Differentiation Formula) 기반 내재적(implicit) 적분을 사용한다.

4.2 BDF 기반 ODE 솔버 구성

시간 적분은 SciPy `solve_ivp`의 `method='BDF'`를 사용한다. 기본 설정 예시는 다음과 같다.

- 상대 오차 허용: `rtol = 1e-6`
- 절대 오차 허용: `atol = 1e-8`
- 초기 스텝: `first_step = 1e-4 s`

4.3 희소 야코비안(sparsity) 제공

1D 열전달 이산화는 국소 결합(local coupling) 특성으로 인해 야코비안 구조가 삼중대각에 가깝다. 따라서 야코비안의 비제로 구조(sparsity pattern)를 미리 제공하면, 내부 뉴턴 반복 및 야코비안 근사 비용을 절감할 수 있다.

4.4 보간/외삽 및 안정성 안전장치

4.4.1 재료 물성 보간

재료 물성 테이블은 선형 보간으로 구성되며, 테이블 범위 밖에서는 선형 외삽을 허용한다.

4.4.2 경계조건 보간

$h(t)$ 및 $T_r(t)$는 선형 보간을 사용하되, 시간 범위 밖에서는 최초/최종 값을 유지하도록 클램프(clamp)한다.

4.4.3 하한값(guard) 적용

수치적 발산/0-division을 방지하기 위해 다음과 같은 하한값을 적용한다.

$$k \geq 10^{-3}, \qquad C_p \geq 1.0 \text{ (또는 no-pyro에서 } 500), \qquad \rho \geq 10^{-20}$$

Chapter 5

GUI 구성 및 기능 설명

5.1 GUI 레이아웃 개요

프로그램은 좌측 사이드바 입력 패널과 우측 3개 그래프 영역으로 구성된다(그림 5.1).

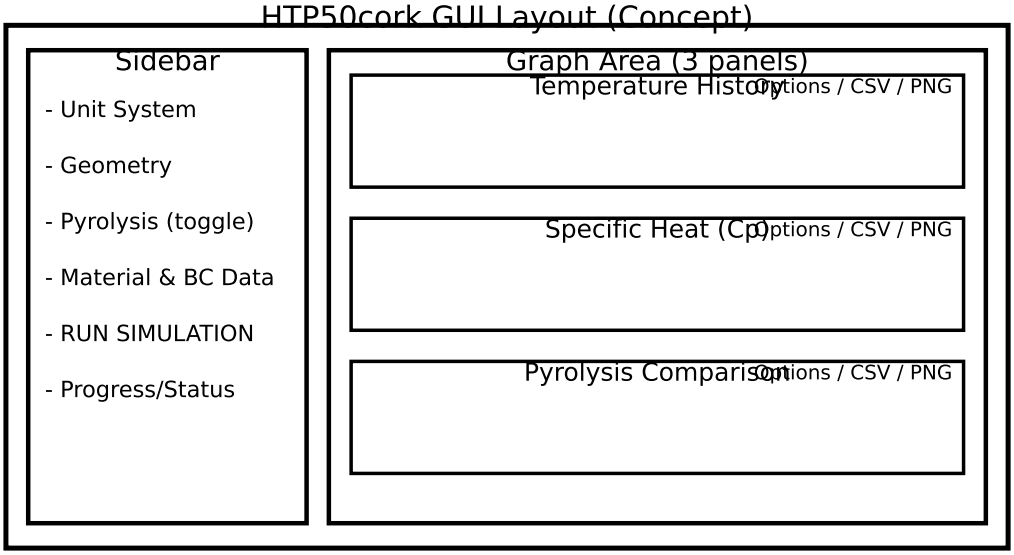

Figure 5.1: GUI 레이아웃 개념도

5.2 사이드바 입력 패널

5.2.1 입력 항목

사이드바는 크게 (1) Unit System, (2) Geometry, (3) Pyrolysis, (4) Material & BC Data로 구성된다. 표 5.1는 주요 입력과 기능을 정리한 것이다.

Table 5.1: 사이드바 주요 UI 요소 및 기능

구성 요소	기능
Unit System	단위계 선택(SI). 기본값 자동 적용 및 일부 파라미터 자동 갱신
Geometry	$L_1, L_2, N_1, N_2, t_{final}, T_{init}$ 입력
Pyrolysis: Density Change	ON: Advanced(밀도 감소 포함), OFF: Simple(Cp/k 동결만)
Pyrolysis: T_critical	열분해 동결 활성화 임계 온도(T_{crit}) 설정
Material & BC Data	Cork/Metal 물성 및 $h(t), T_r(t)$ 데이터 CSV 로드/편집
RUN SIMULATION	입력 검증 후 해석 실행(열분해 포함/미포함 2회)
Progress/Status	진행률 및 상태(Ready/Solving/Done/Error) 표시

5.3 그래프 영역 및 버튼

그래프는 3개 섹션으로 구성된다.

1. Temperature History: 온도 이력(노드별)
2. Specific Heat (Cp): 비열 변화(열분해 동결 영향 확인)
3. Pyrolysis Comparison: 열분해 포함(실선) vs 미포함(점선) 비교

각 그래프 섹션에는 공통적으로 **Options/CSV/PNG** 버튼이 제공된다.

- Options: 표시 노드/색상/레이블 설정(팝업)
- CSV: 해당 그래프 데이터 내보내기
- PNG: 해당 그래프 이미지(고해상도) 내보내기

Temperature History에는 추가로 **Peak Temp** 버튼이 있으며, 선택 노드의 최고 온도와 도달 시간을 요약 표시한다.

5.4 실행 흐름(스레딩/진행률)

GUI 프리징을 방지하기 위해 계산은 워커 스레드에서 수행되며, 메인 스레드는 일정 주기(100 ms)로 솔버의 현재 시간(`t_current`)을 폴링하여 진행률을 업데이트한다. 그림 5.2는 실행 흐름을 개념적으로 나타낸다.

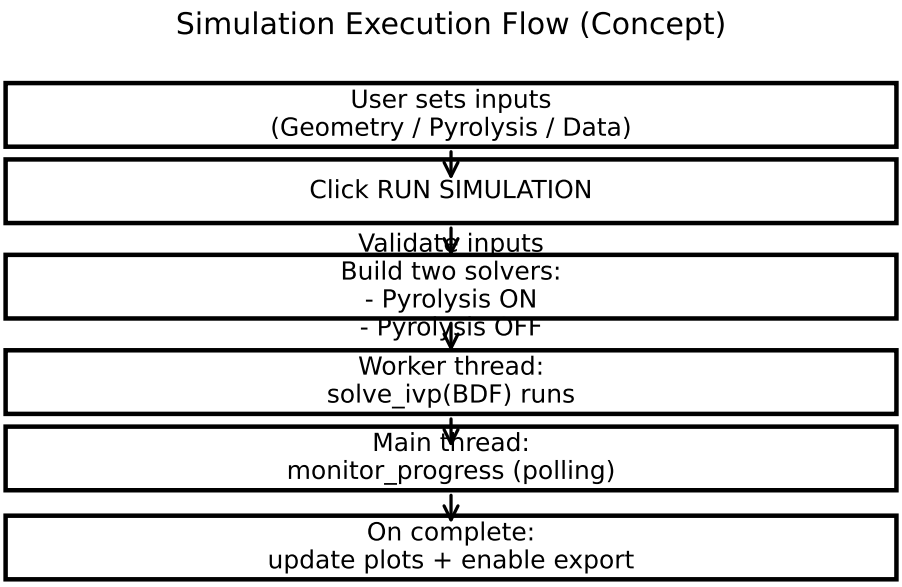

Figure 5.2: 시뮬레이션 실행 흐름(개념도)

5.5 데이터 편집 팝업(Table Editor)

재료/경계 데이터는 내장 Table Editor로 편집할 수 있으며, 주요 기능은 다음과 같다.

- Add Row / Delete Row: 데이터 행 추가/삭제
- Plot Graph: 현재 데이터의 그래프 미리보기
- Save Default: 사용자 기본값(JSON)으로 영구 저장
- Save & Close: 현재 세션에 적용 후 닫기

5.6 그래프 옵션 팝업(Graph Options)

Graph Options 창에서는 라인 단위로 (노드/색상/레이블)을 설정한다. 노드 지정은 숫자 외에도 `surf`, `mid/int`, `bot` 등의 별칭을 허용한다.

Chapter 6

한계 및 향후 개선 방향

6.1 모델링 한계

- 복사(radiation) 열전달은 기본 모델에 포함되어 있지 않다(대류 중심).
- 삭마(ablation)로 인한 표면 후퇴(moving boundary)는 fixed-grid 가정으로 단순화되어 있다.
- 열분해 반응 kinetics(Arrhenius 반응률 등)는 상태 변수 $T_{\max}$ 기반의 준현상학적(phenomenological) 모델로 대체되어 있다.

6.2 소프트웨어 개선 제안

- 모듈 분리: 물리 엔진/GUI/데이터 I/O를 패키지 구조로 분리하여 유지보수성 향상
- 테스트: 회귀 테스트(기본 입력 대비 결과 스냅샷) 및 단위 테스트 도입
- 입력 검증: 물리적 범위/단위 정합성 검증 강화
- 확장: 복사 경계조건, 반응열/가스 생성, 이동 경계 모델 등 단계적 확장

Appendix A

(부록 A) 코드 아키텍처

A.1 계층 구조 개요

본 소프트웨어는 물리 엔진(Physics), 데이터 관리(Data), 프레젠테이션(GUI) 계층으로 책임을 분리한다. 그림 A.1는 계층 구조를 개념적으로 나타낸다.

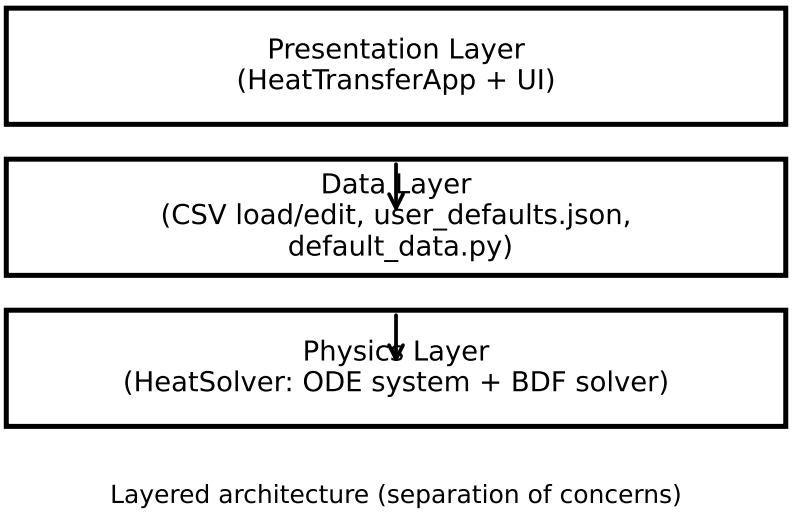

Figure A.1: 계층 기반 아키텍처(개념도)

A.2 핵심 클래스 책임

A.2.1 HeatSolver (Physics Engine)

- 격자 생성 및 노드 인덱싱(코르크/금속/계면)
- 재료 물성 $k(T), C_p(T), \rho(T)$ 보간 함수 초기화

- 경계조건 $h(t), T_r(t)$ 보간 함수 초기화
- 열분해 모드(Simple/Advanced) 처리: $T_{\max}$ 상태 변수, Cp/k 동결, 밀도 감소(선택)
- ODE RHS 구성 및 BDF 적분 수행

A.2.2 HeatTransferApp (GUI Controller)

- 사용자 입력 수집/검증
- 두 솔버(열분해 포함/미포함) 생성 및 워커 스레드 실행
- 진행률 표시(폴링), 그래프 갱신, CSV/PNG 내보내기
- 팝업(Table Editor, Graph Options) 관리

A.3 파일 구성(로컬 프로젝트 기준)

- `main.tex`: 보고서 메인(이 문서)
- `sections/`: 장별 텍스트
- `figures/`: 개념도(벡터 PDF)
- `code/`: 부록용 코드 발췌 및 원본(`app.py`)

Appendix B

(부록 B) 코드 발췌

B.1 HeatSolver 핵심 로직(발췌)

본 부록은 보고서 이해를 위해 핵심 로직만 발췌/요약한 것이다. 전체 구현은 code/app.py 를 참고한다.

```
"""HTP50cork - HeatSolver 발췌요약(/)주의

:
- 이 파일은 app.의py HeatSolver 구현에서 핵심 로직만 발췌하여 보고서 부록용
    으로 정리한 것입니다.
- 벡터화버퍼/ 최적화/GUI 관련 코드는 생략되어 있습니다.
- 전체 원본은 code/app.를py 참고하세요.
"""

import numpy as np
from scipy.integrate import solve_ivp
from scipy.interpolate import interp1d
from scipy.sparse import lil_matrix


class HeatSolver:
    """1D transient heat transfer (+ optional pyrolysis) solver.

    State vector:
        y = [T_0 ... T_{N-1}, Tmax_0 ... Tmax_{N-1}]  (size = 2N)
    """

    def __init__(self, params, materials_data, bc_data, enable_pyrolysis=
    True):
        self.p = params
```

```
        self.mat_data = materials_data
        self.bc_data = bc_data
        self.enable_pyrolysis = enable_pyrolysis

        # Geometry (bilayer)
        self.L1 = params['L1']          # cork thickness
        self.L2 = params['L2']          # metal thickness
        self.N1 = int(params['N1'])     # cork nodes
        self.N2 = int(params['N2'])     # metal nodes
        self.dx1 = self.L1 / self.N1
        self.dx2 = self.L2 / self.N2
        self.N = self.N1 + self.N2
        self.m = self.N1               # interface index

        self._init_interpolators()
        self.jac_sparsity = self._build_jac_sparsity()
        self.t_current = 0.0

    def _build_jac_sparsity(self):
        """Provide sparsity pattern for the BDF Jacobian approximation."""
        N = self.N
        size = 2 * N
        J = lil_matrix((size, size), dtype=int)

        # Temperature equations: tri-diagonal coupling
        for i in range(N):
            J[i, i] = 1
            if i > 0:     J[i, i-1] = 1
            if i < N-1:   J[i, i+1] = 1
            # pyrolysis affects cork-side properties via Tmax
            if i <= self.m:
                J[i, N+i] = 1

        # Tmax equations: depends on T and Tmax
        for i in range(N):
            row = N + i
            J[row, i] = 1
            if i > 0:     J[row, i-1] = 1
            if i < N-1:   J[row, i+1] = 1
            J[row, N+i] = 1

```

```
65          return J.tocsr()
66
67      def _init_interpolators(self):
68          """Create property interpolators k(T), Cp(T), rho(T), and BC h(t),
    Tr(t)."""
69
70          def create_interp(data_list):
71              arr = np.array(data_list, dtype=float)
72              x = arr[:, 0]
73              f_k   = interp1d(x, arr[:, 1], kind='linear', bounds_error=
    False, fill_value="extrapolate")
74              f_Cp  = interp1d(x, arr[:, 2], kind='linear', bounds_error=
    False, fill_value="extrapolate")
75              f_rho = interp1d(x, arr[:, 3], kind='linear', bounds_error=
    False, fill_value="extrapolate")
76              return f_k, f_Cp, f_rho
77
78          # Cork and Metal property tables
79          self.k1_func, self.Cp1_func, self.rho1_func = create_interp(self.
    mat_data['cork'])
80          self.k2_func, self.Cp2_func, self.rho2_func = create_interp(self.
    mat_data['metal'])
81
82          # Normalized mass profile M(Tmax) for Advanced mode
83          mass = np.array(self.mat_data.get('mass', []), dtype=float)
84          if len(mass) > 0:
85              self.mass_func = interp1d(mass[:, 0], mass[:, 1], kind='linear'
    ,
86                                        bounds_error=False, fill_value=(mass
    [0, 1], mass[-1, 1]))
87          else:
88              self.mass_func = lambda x: 100.0
89
90          # Boundary conditions
91          htab  = np.array(self.bc_data['h'], dtype=float)
92          Trtab = np.array(self.bc_data['Tr'], dtype=float)
93          self.h_func  = interp1d(htab[:, 0],  htab[:, 1],  kind='linear',
94                                  bounds_error=False, fill_value=(htab[0, 1],
     htab[-1, 1]))
95          self.Tr_func = interp1d(Trtab[:, 0], Trtab[:, 1], kind='linear',
96                                  bounds_error=False, fill_value=(Trtab[0, 1],
```

```
        Trtab[-1, 1]))

    @staticmethod
    def k_harm(k1, k2):
        """Harmonic mean conductivity."""
        return (2.0 * k1 * k2) / (k1 + k2 + 1e-12)

    @staticmethod
    def sigmoid(x, stiffness=100.0):
        """Smooth gate function to avoid discontinuities."""
        limit = 700.0 / (stiffness + 1e-12)
        x = np.clip(x, -limit, limit)
        return 1.0 / (1.0 + np.exp(-stiffness * x))

    def get_props1(self, T, Tmax):
        """Cork properties with optional pyrolysis effects."""
        k   = np.maximum(self.k1_func(T),   1e-3)
        Cp  = np.maximum(self.Cp1_func(T),  1.0)
        rho = np.maximum(self.rho1_func(T), 1e-20)

        if not self.enable_pyrolysis:
            return k, np.maximum(Cp, 500.0), rho

        Tcrit = float(self.p.get('T_simple', 500.0))
        mode  = str(self.p.get('pyro_mode', 'advanced')).lower()

        # Freeze Cp/k after Tmax exceeds Tcrit AND node is cooling.
        g_thr  = self.sigmoid(Tmax - Tcrit, stiffness=20.0)
        g_cool = self.sigmoid((Tmax - 0.1) - T, stiffness=20.0)
        factor = g_thr * g_cool

        kTmax  = np.maximum(self.k1_func(Tmax), 1e-3)
        CpTmax = np.maximum(self.Cp1_func(Tmax), 1.0)

        k  = k  * (1.0 - factor) + kTmax  * factor
        Cp = Cp * (1.0 - factor) + CpTmax * factor

        # Advanced: density from mass profile (fixed-grid assumption)
        if mode != 'simple':
            mass_pct = np.clip(self.mass_func(Tmax), 20.0, 100.0)  # [%]
            rho = rho * (mass_pct / 100.0)
```

```
137
138         return k, Cp, np.maximum(rho, 1e-20)
139
140     def get_props2(self, T):
141         """Metal properties."""
142         k   = np.maximum(self.k2_func(T),   1e-3)
143         Cp  = np.maximum(self.Cp2_func(T),  500.0)
144         rho = np.maximum(self.rho2_func(T), 1e-20)
145         return k, Cp, rho
146
147     def ode_system(self, t, y):
148         """Compute RHS of coupled ODE system (T and Tmax)."""
149         self.t_current = t
150         N = self.N
151         T    = y[:N]
152         Tmax = y[N:]
153
154         dTdt = np.zeros(N)
155
156         # Boundary values
157         h  = max(0.0, float(self.h_func(t)))
158         Tr = float(self.Tr_func(t))
159
160         # Properties
161         k1, Cp1, rho1 = self.get_props1(T[:self.m],   Tmax[:self.m])
162         k2, Cp2, rho2 = self.get_props2(T[self.m:])
163
164         # ---- Heat equation (schematic) ----
165         # 0) Surface: convection + conduction to node 1
166         k01 = self.k_harm(k1[0], k1[1])
167         q_cond = k01 * (T[0] - T[1]) / self.dx1
168         q_conv = h * (Tr - T[0])
169         dTdt[0] = (q_conv - q_cond) / (rho1[0] * Cp1[0] * (self.dx1 / 2))
170
171         # 1) Interior nodes (vectorized in full code; omitted here)
172         # 2) Interface: mixed heat capacity of half-cells (cork + metal)
173         # 3) Metal interior and bottom adiabatic boundary
174         # ------------------------------------------------------------
175
176         # ---- History equation: Tmax grows only when T is rising ----
177         ramp = np.maximum(0.0, dTdt)
```

```
178         gate = self.sigmoid(T - Tmax, stiffness=100.0)
179         dTmax_dt = ramp * gate
180 
181         return np.concatenate([dTdt, dTmax_dt])
182 
183     def solve(self):
184         """Time integration using SciPy BDF."""
185         t_span = (0.0, float(self.p['t_final']))
186 
187         T0    = np.ones(self.N) * float(self.p['T_init'])
188         Tmax0 = np.ones(self.N) * float(self.p['T_init'])
189         y0 = np.concatenate([T0, Tmax0])
190 
191         sol = solve_ivp(
192             self.ode_system, t_span, y0,
193             method='BDF',
194             rtol=1e-6, atol=1e-8,
195             first_step=1e-4,
196             jac_sparsity=self.jac_sparsity,
197         )
198         return sol
```

B.2 GUI 스레딩/진행률 흐름(발췌)

```
1  """HTP50cork - GUI 실행스레딩/ 흐름 발췌요약()
2 
3  - GUI 프리징을 방지하기 위해 계산(solve)을 워커 스레드에서 수행하고,
4    메인 스레드는 after() 기반 폴링으로 진행률을 업데이트한다.
5  - 전체 원본은 code/app.를py 참고.
6  """
7 
8  import threading
9  import time
10 
11 
12 class HeatTransferApp:
13     def run_simulation(self):
14         # 1) 입력값 검증 및 파라미터 구성
15         params = {
16             'L1': float(self.entries['L1'].get()),
```

```
            'L2': float(self.entries['L2'].get()),
            'N1': int(self.entries['N1'].get()),
            'N2': int(self.entries['N2'].get()),
            't_final': float(self.entries['t_final'].get()),
            'T_init': float(self.entries['T_init'].get()),
            'T_simple': float(self.entries['T_simple'].get()),
            'pyro_mode': self.pyro_mode_var.get(),
        }

        # 2) 열분해 포함미포함/ 솔버 개2 구성
        solver_pyro    = HeatSolver(params, mat_data_pyro,    self.bc_data,
    enable_pyrolysis=True)
        solver_no_pyro = HeatSolver(params, mat_data_no_pyro, self.bc_data,
    enable_pyrolysis=False)

        # 3) 워커 스레드에서 계산 실행
        thread = threading.Thread(target=self._run_solver_thread,
                                  args=(solver_pyro, solver_no_pyro))
        thread.daemon = True
        thread.start()

        # 4) 메인 스레드: 진행률 폴링
        self.monitor_progress(solver_pyro, thread)

    def monitor_progress(self, solver, thread):
        if thread.is_alive():
            t_curr = getattr(solver, 't_current', 0.0)
            t_final = solver.p['t_final']
            self.progress_bar.set(min(t_curr / t_final, 1.0))
            self.after(100, lambda: self.monitor_progress(solver, thread))
        else:
            self.progress_bar.set(1.0)

    def _run_solver_thread(self, solver1, solver2):
        start_t = time.time()
        sol1 = solver1.solve()
        sol2 = solver2.solve()
        elapsed = time.time() - start_t
        self.after(0, lambda: self._on_simulation_complete(solver1, sol1,
    sol2, elapsed))
```

Appendix C

(부록 C) 입력 데이터 형식(CSV)

C.1 데이터셋 개요

프로그램은 재료 물성 및 경계조건을 CSV로 입력받는다. 기본적으로 각 CSV는 1행 헤더 + 수치 행으로 구성된다.

C.2 CSV 헤더(권장)

표 C.1는 데이터 키별 권장 헤더를 정리한 것이다.

Table C.1: 입력 CSV 헤더 형식

데이터 키	헤더(열)
Cork (Pyrolysis)	`Temp, k, Cp, rho`
Cork (No Pyrolysis)	`Temp, k, Cp, rho`
Mass Profile (Advanced)	`Temp, MassNorm`
Metal	`Temp, k, Cp, rho`
h(t)	`Time, Value`
Tr(t)	`Time, Value`

C.3 단위 주의

- 온도: 기본 °C 기준(테이블과 GUI가 °C로 가정)
- h: W/(m^2 K), k: W/(m K), C_p: J/(kg K), ρ: kg/m^3
- 시간: s

Appendix D

(부록 D) 참고 문헌

Bibliography

[1] Internal Technical Note, 수치 해석 모델: *1*차원 비정상 열전달 및 열분해 (*1-D Transient Heat Transfer & Pyrolysis*), February 22, 2026.

[2] Project README, *HTP50cork — 1D* 비정상 열전달 시뮬레이터 (*Cork-Metal Bilayer*).

[3] Sakraker et al., *Cork-based thermal protection system: characterization and in-flight validation*, CEAS Space Journal (2022), DOI: 10.1007/s12567-021-00395-z.

[4] SciPy Developers, *scipy.integrate.solve_ivp Documentation*.