

---

# ECLSS シミュレーション開発

# VHABによるモデリング

有人宇宙基地に特化した化学プロセスシミュレーター(Matlab言語)



FIG 4. Matter Store

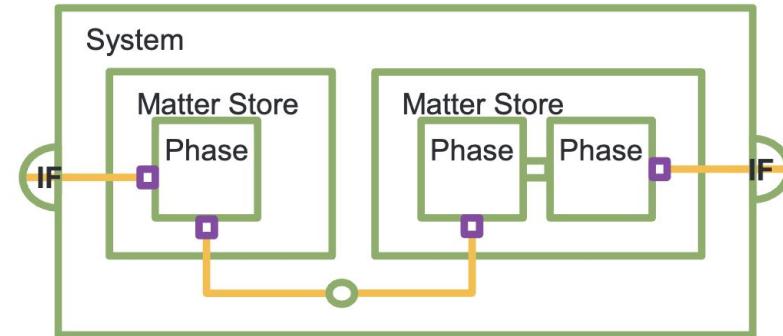


FIG 8. V-HAB System with Branches

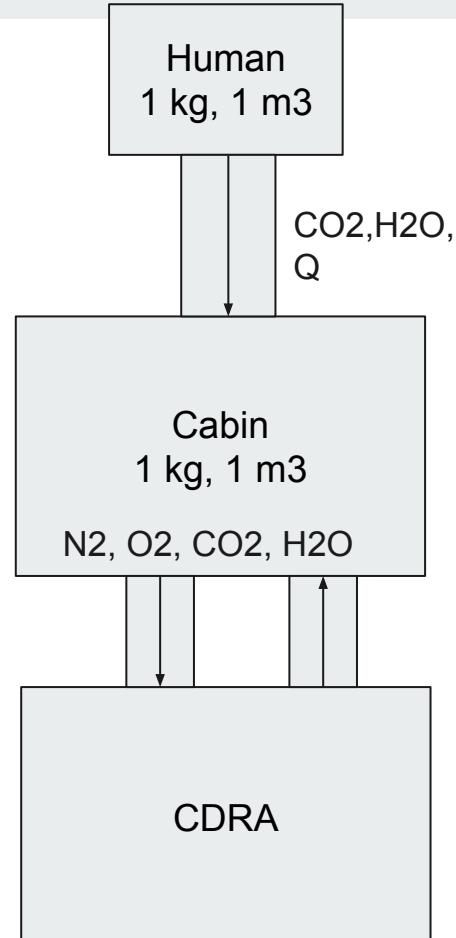
# マイルストーン

① キャビン(制御装置なし)のモデル化

- ・クルーによる一定のCO<sub>2</sub>, H<sub>2</sub>O, Heat供給
- ・ppCO<sub>2</sub>, Humidity, Temperature 単調増加(大気圧は微量増加)

② CDRAのモデル化

③ OSSのプロトタイプ作成

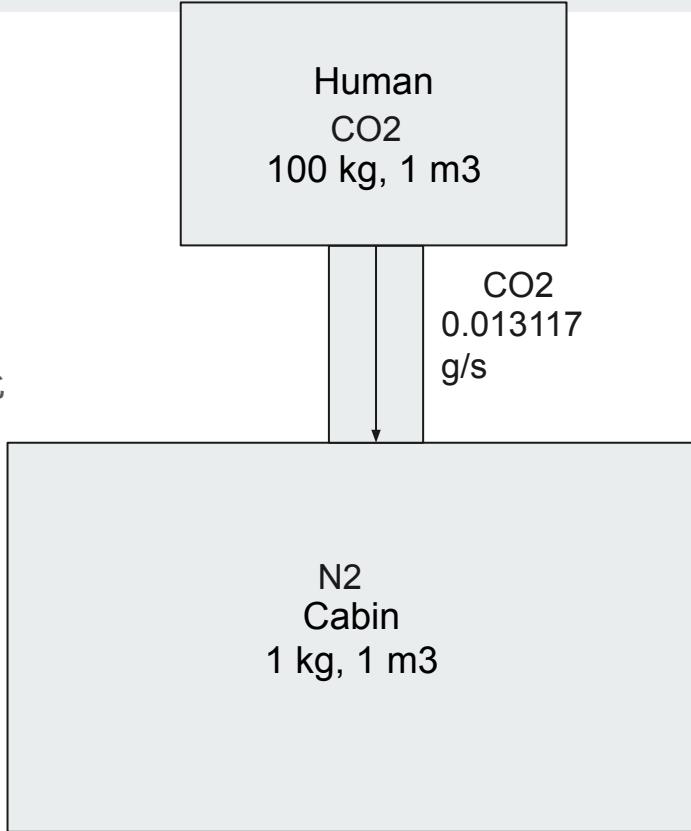


## ①-1 シンプルなキャビンのモデル化

Humanストアからキャビンストアに対する二酸化炭素の移動をモデル化

$$PV=nRT$$

二酸化炭素モル質量 44g/mol



# コード

```
classdef ExampleHERA < vsys
properties (SetAccess = protected, GetAccess = public)

end

methods
    function this = ExampleHERA(oParent, sName)
        this@vsys(oParent, sName, 30);%exec()関数を使用する時間ステップを定義。この例では30秒が使用されており、これはexec()関数が30秒ごとに呼び出される
        eval(this.oRoot.oCfgParams.configCode(this));
    end

    function createMatterStructure(this)
        createMatterStructure@vsys(this);

        %ストアの定義
        matter.store(this, 'Cabin', 1);
        matter.phases.gas(this.toStores.Cabin, 'CabinAir', struct('N2', 1), 1, 293.15);

        matter.store(this, 'Human', 1);
        matter.phases.gas(this.toStores.Human, 'HumanCO2', struct('CO2', 1), 1, 293.15);

        %プランチの定義
        matter.branch(this, this.toStores.Human.toPhases.HumanCO2, {}, this.toStores.Cabin.toPhases.CabinAir, 'CO2_Human_to_Cabin');
    end

    function createSolverStructure(this)
        createSolverStructure@vsys(this);

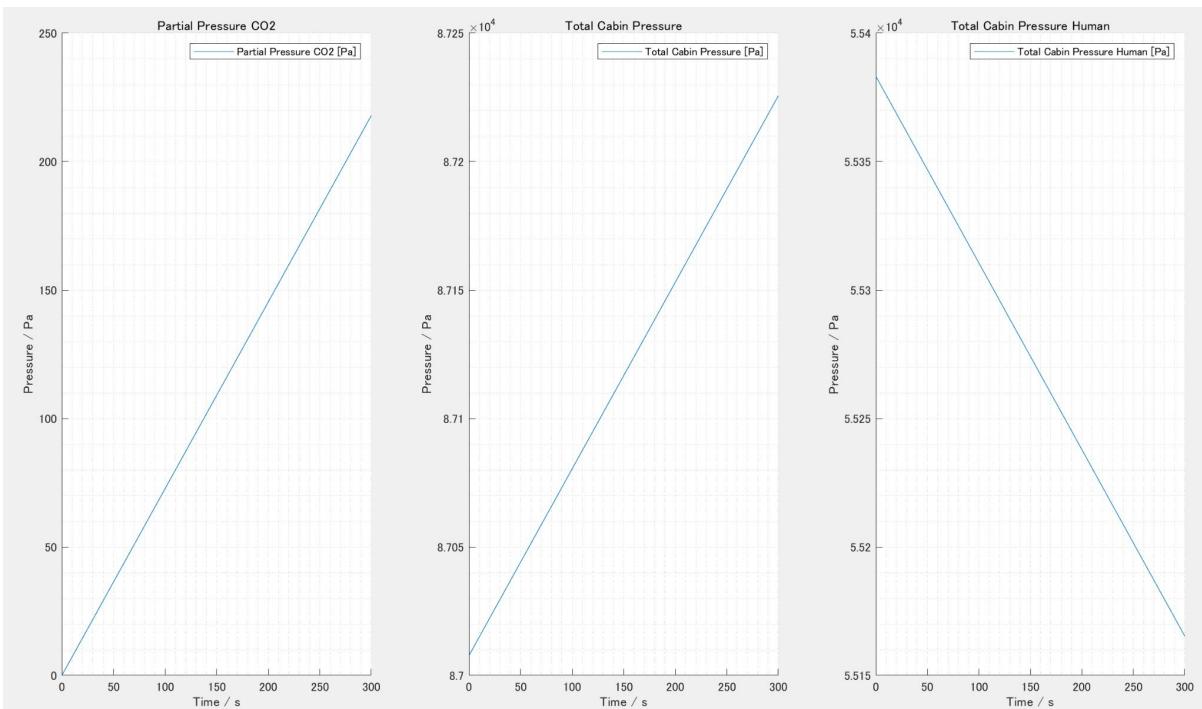
        solver.matter.manual.branch(this.toBranches.CO2_Human_to_Cabin');
        %this.toBranches.CO2_Human_to_Cabin.oHandler.setVolumetricFlowRate(0.01); %分岐を通じて一定の容積流量を確保
        this.toBranches.CO2_Human_to_Cabin.oHandler.setFlowRate(0.01); %分岐を通じて一定の容積流量を確保 0.000013117kg/s

        % 热ソルバー
        this.setThermalSolvers();
    end
end

methods (Access = protected)
    function exec(this, ~)
        exec@vsys(this);
    end
end
end
```



# シミュレーション結果



---

## ①-2 P2Pを用いたキャビンのモデル化

同一ストア間のphaseとphaseどうしの物質移動

→時間切れでギブアップ

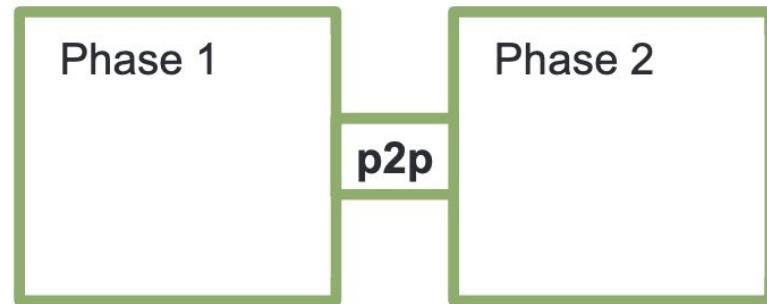


FIG 6. Phase-to-phase processor

## ② CDRAのモデル化

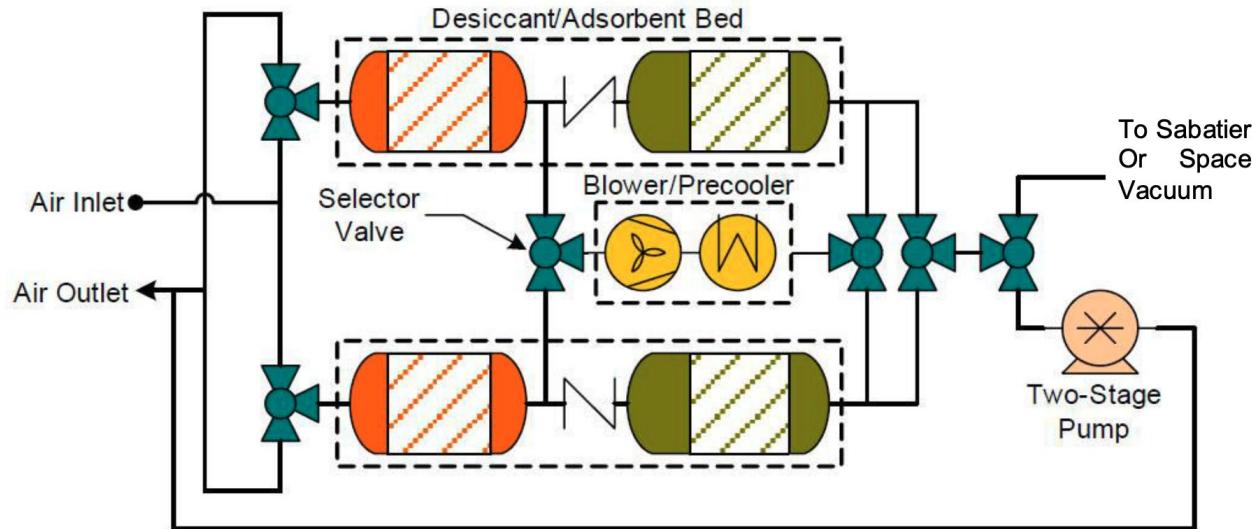


Figure 1. Integration of the CDRA Components<sup>9</sup>

# On-Orbit ISS ECLS Hardware Distribution as of February 2010

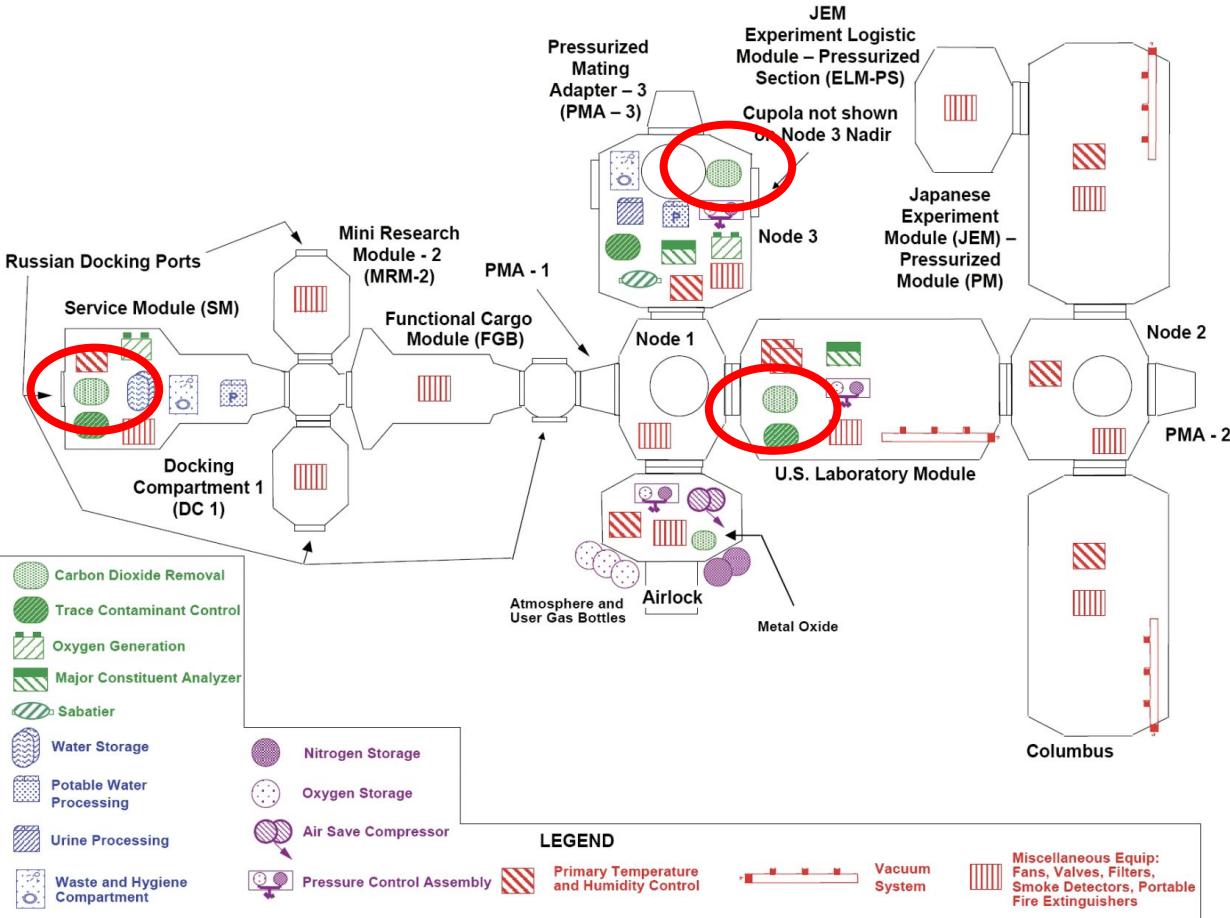
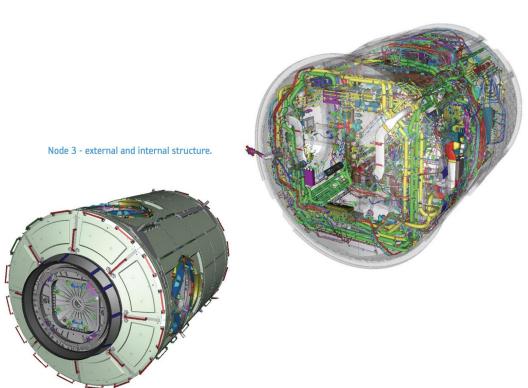
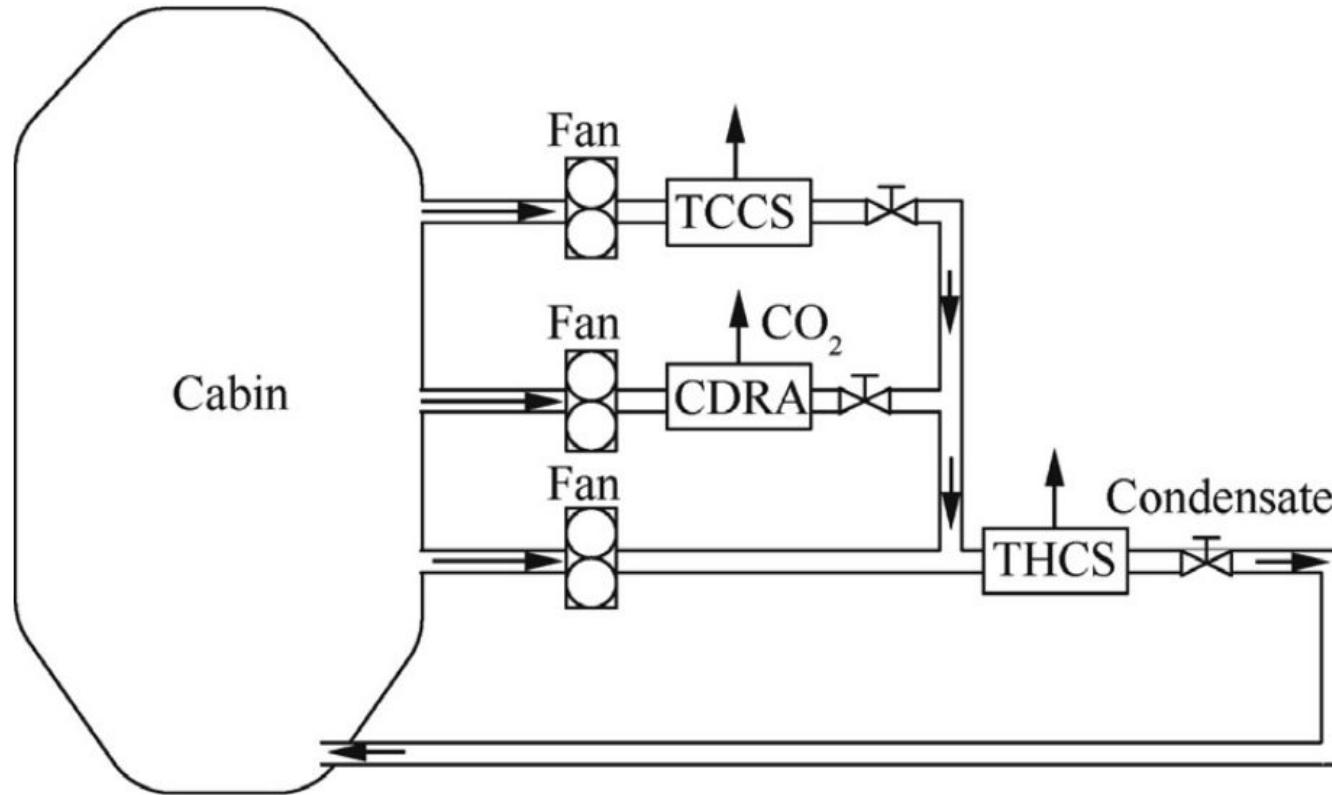


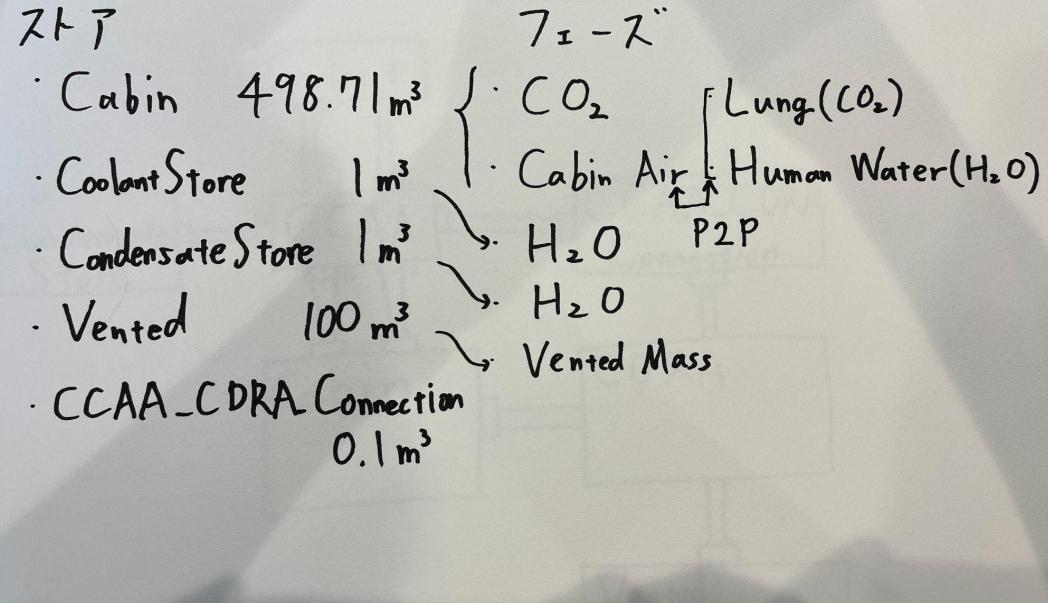
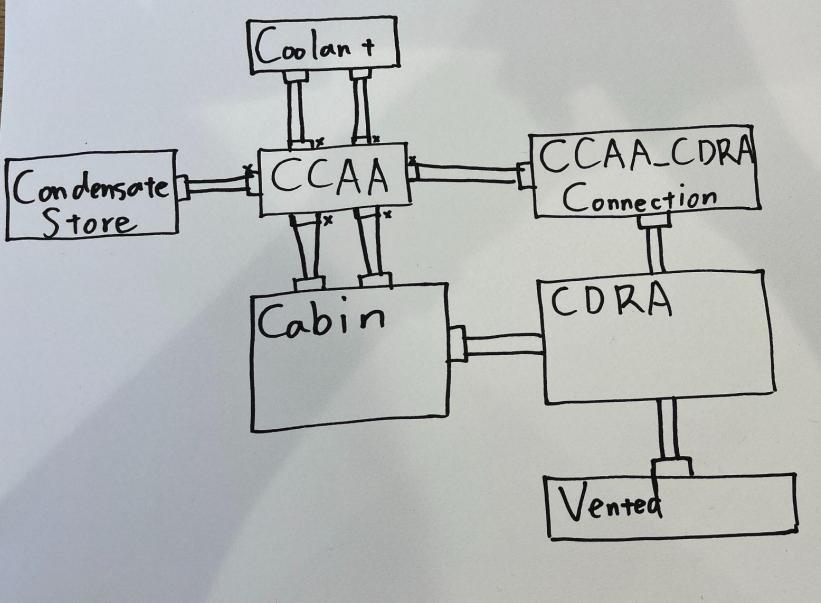
Diagramme: NASA/ISEC



(b) Simplified ventilation pipe network

**Fig. 1** Distribution and flow diagram of ECLSS.

# CDRAデモンストレーション



CCAA

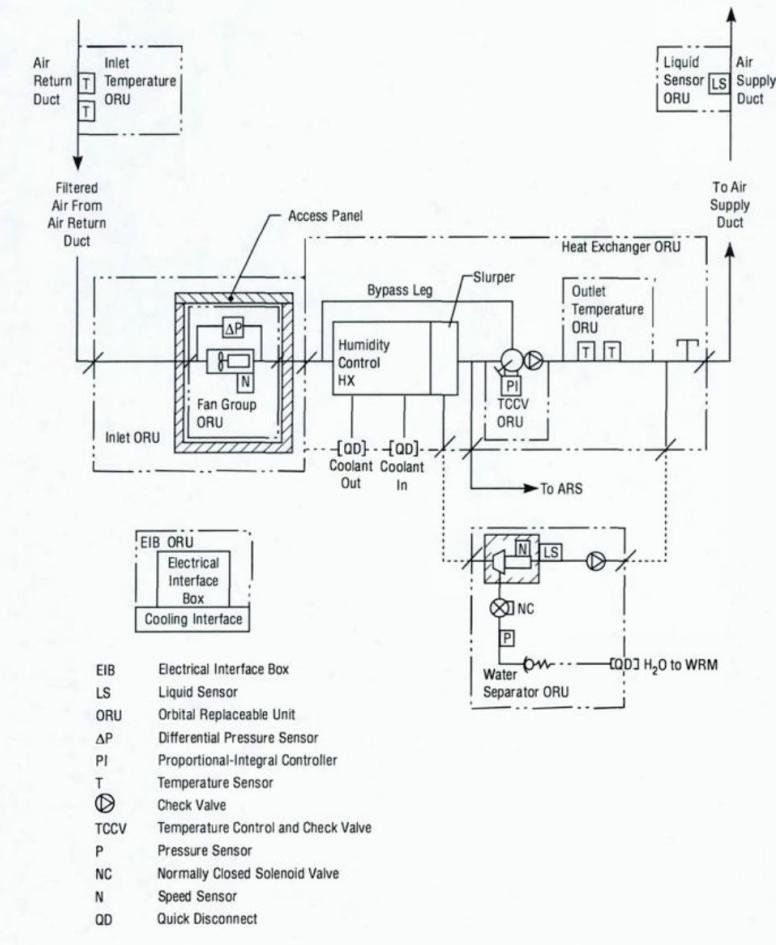
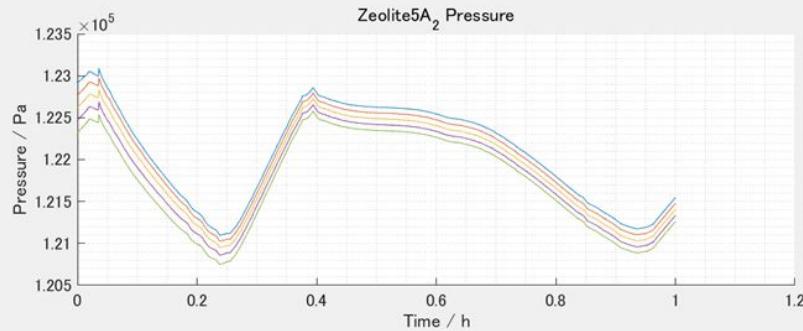
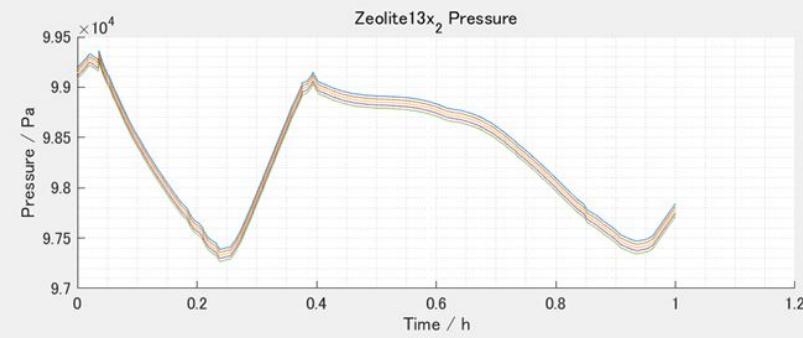
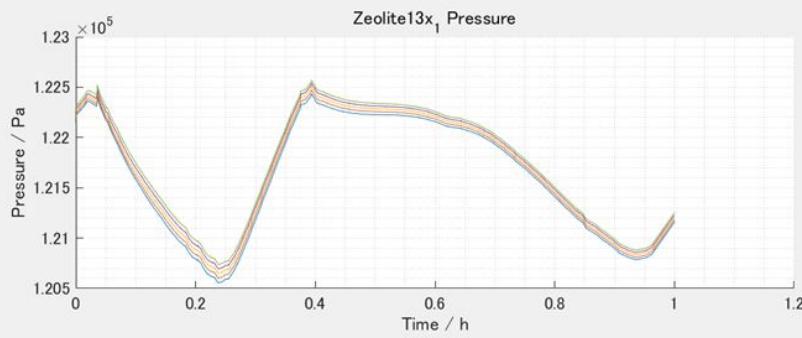
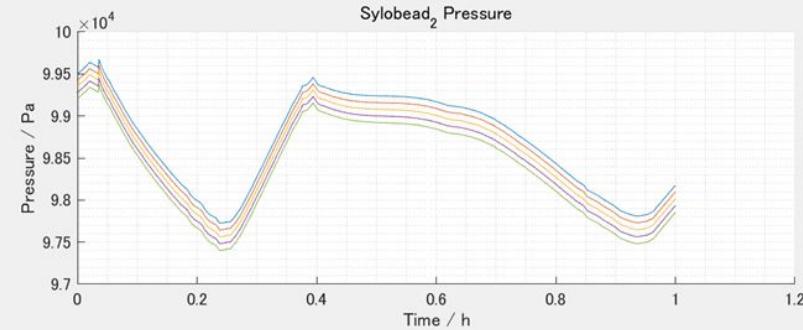
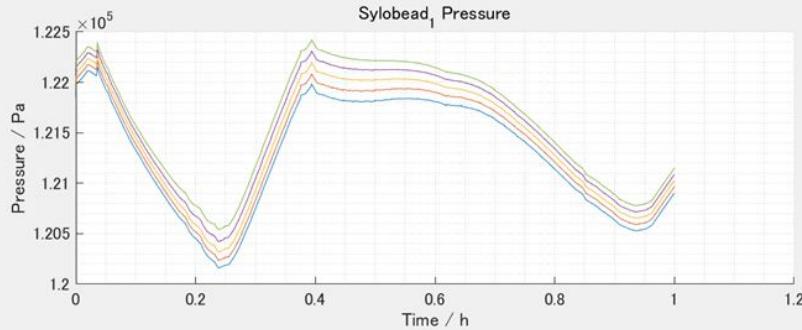
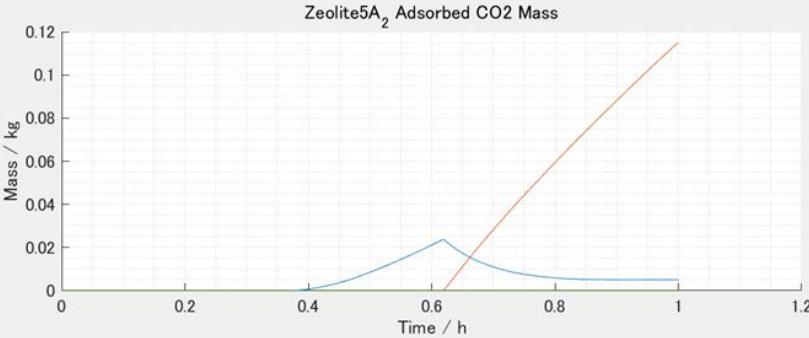
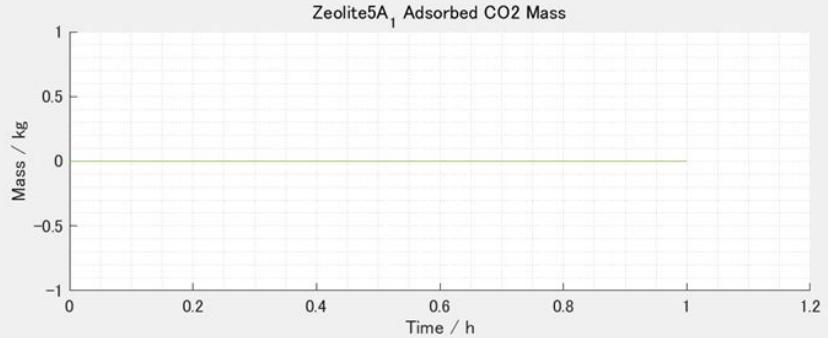
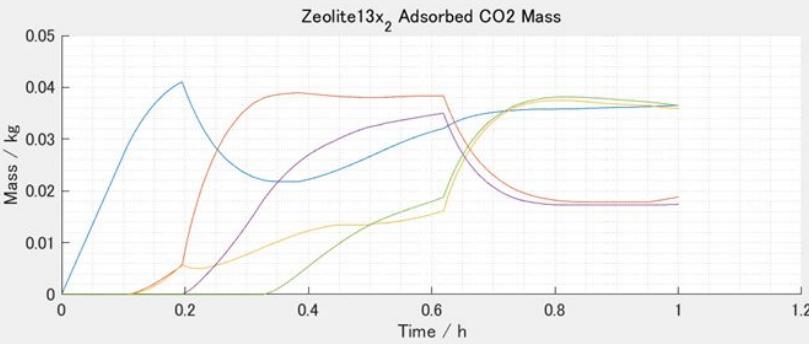
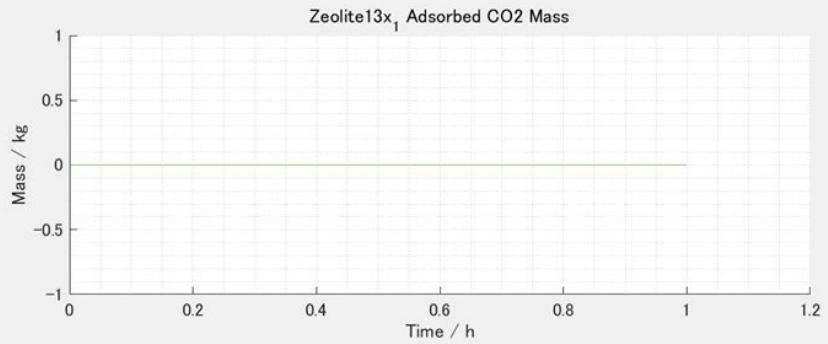
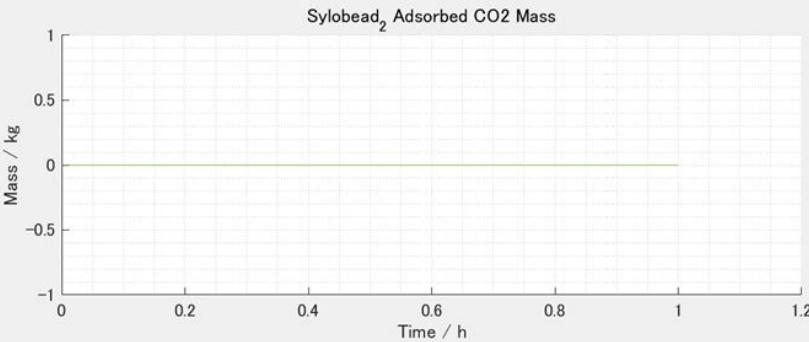
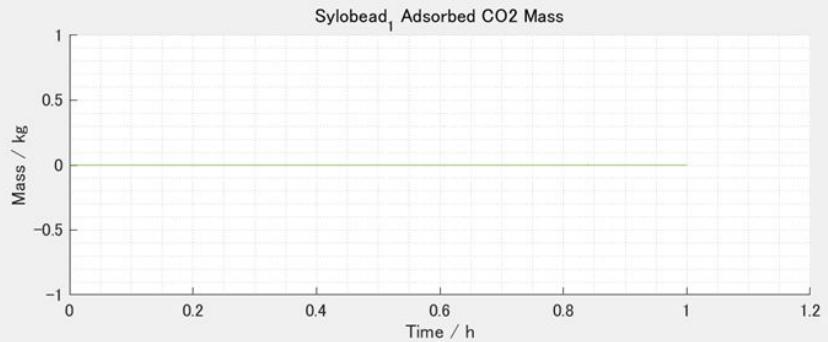
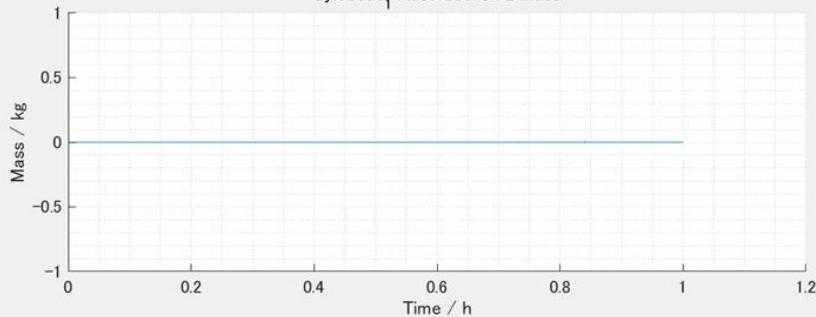


FIG 11. Common Cabin Air Assembly Process Schematic [21]

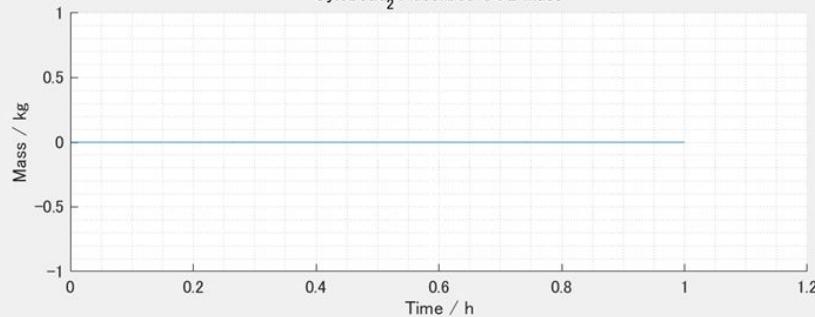




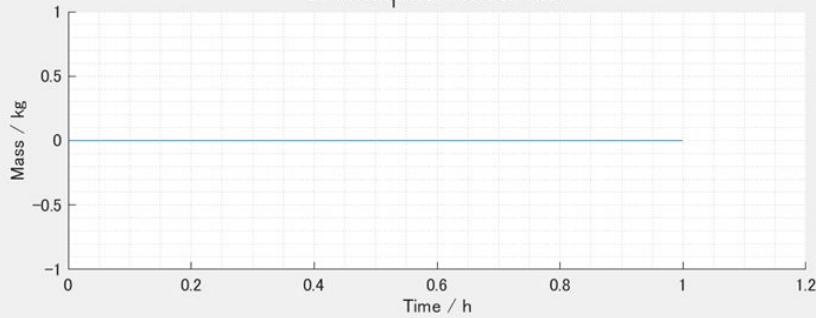
Sylobead<sub>1</sub> Adsorbed CO<sub>2</sub> Mass



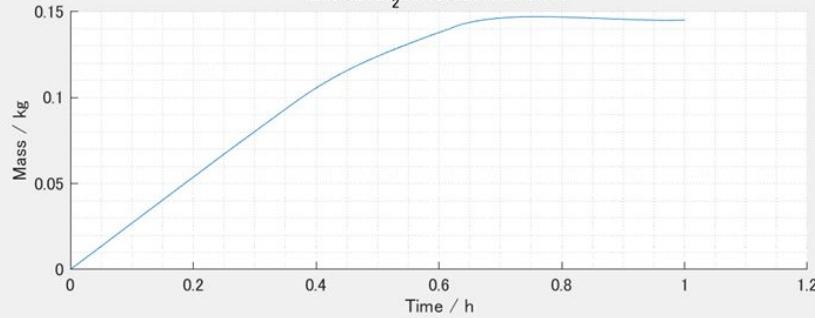
Sylobead<sub>2</sub> Adsorbed CO<sub>2</sub> Mass



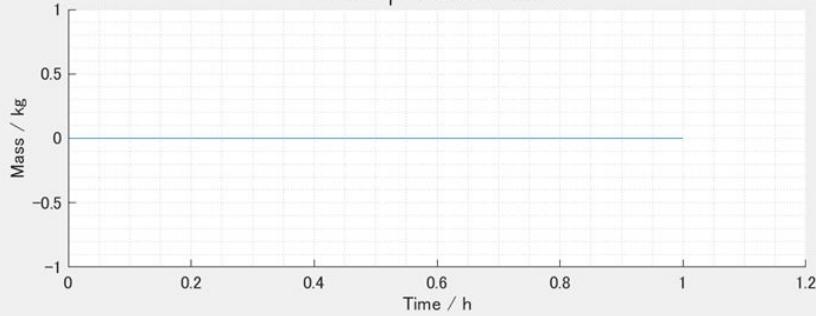
Zeolite13x<sub>1</sub> Adsorbed CO<sub>2</sub> Mass



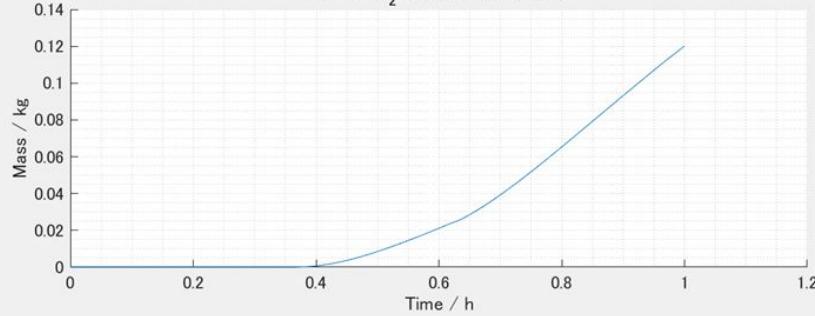
Zeolite13x<sub>2</sub> Adsorbed CO<sub>2</sub> Mass

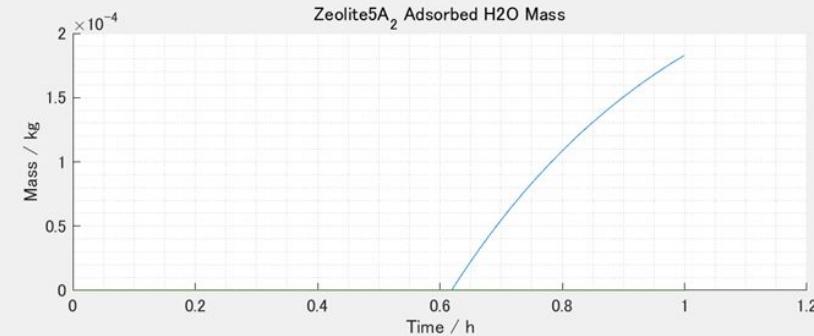
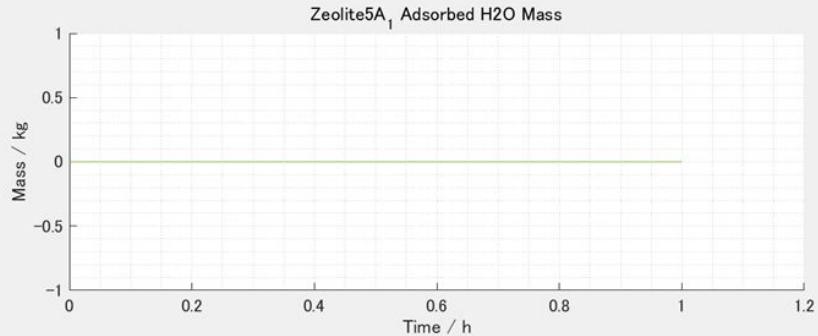
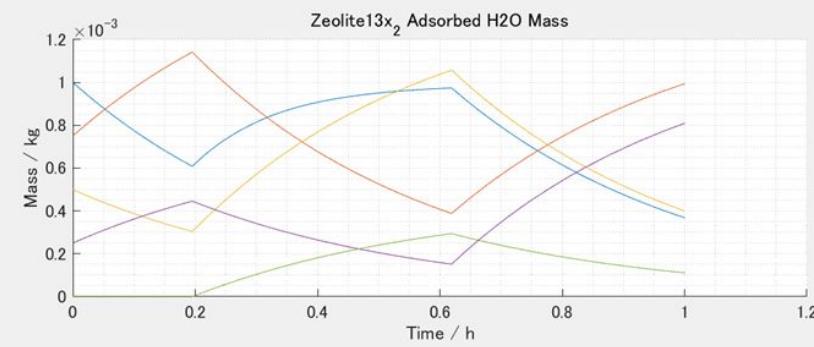
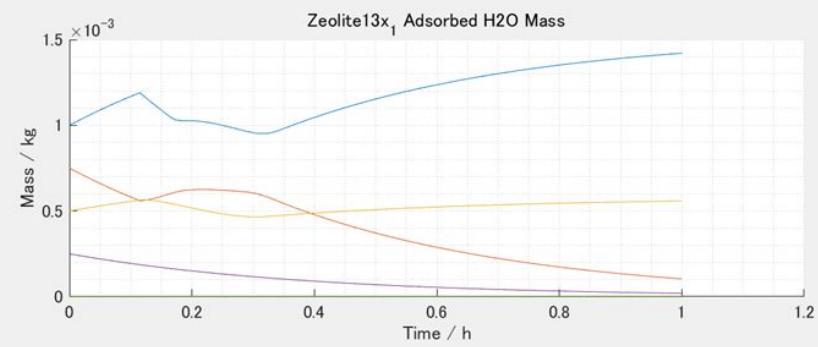
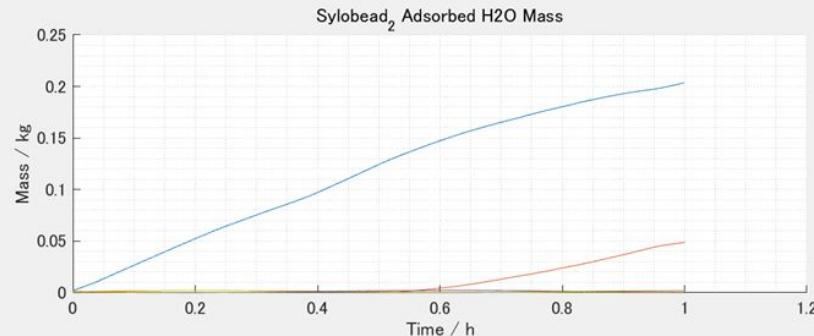
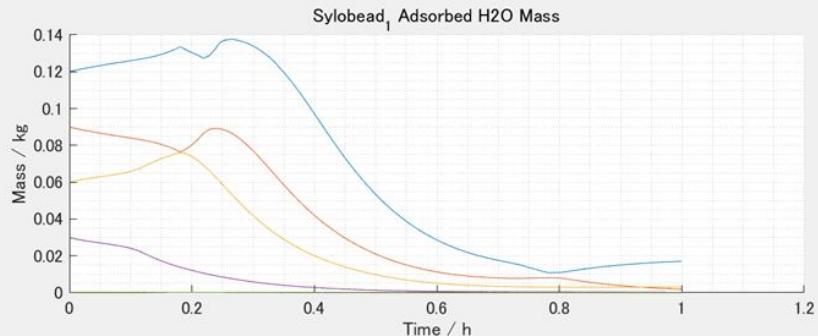


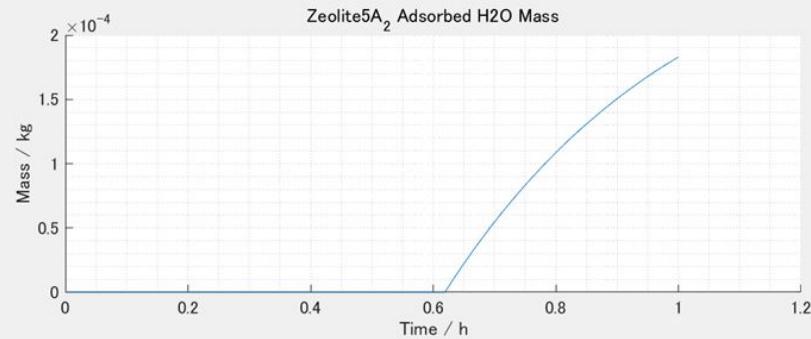
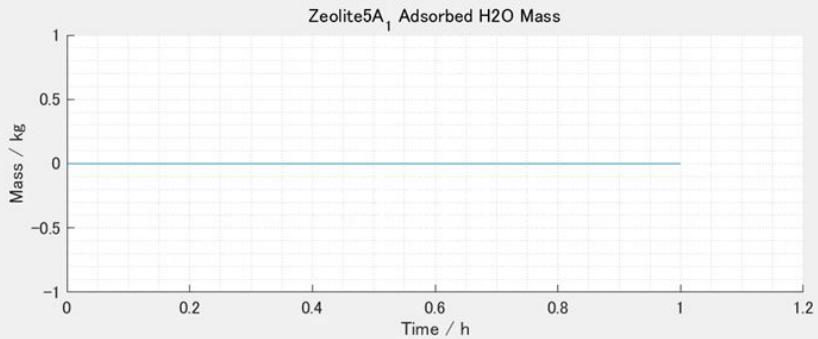
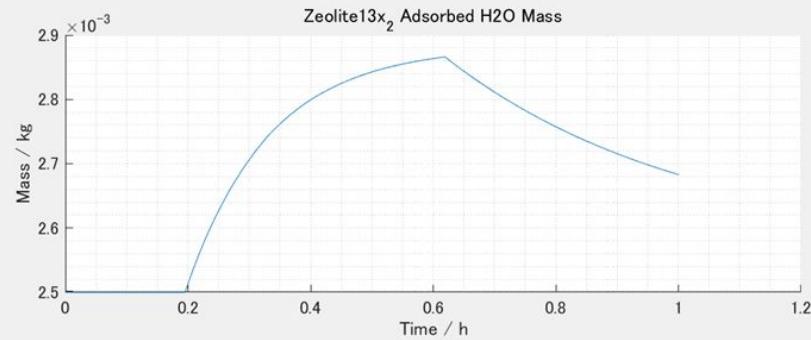
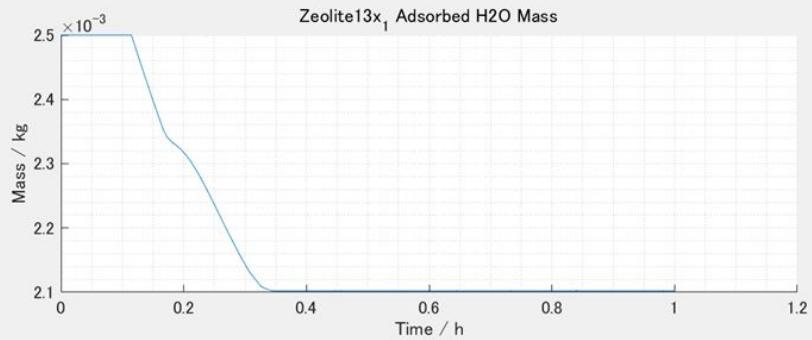
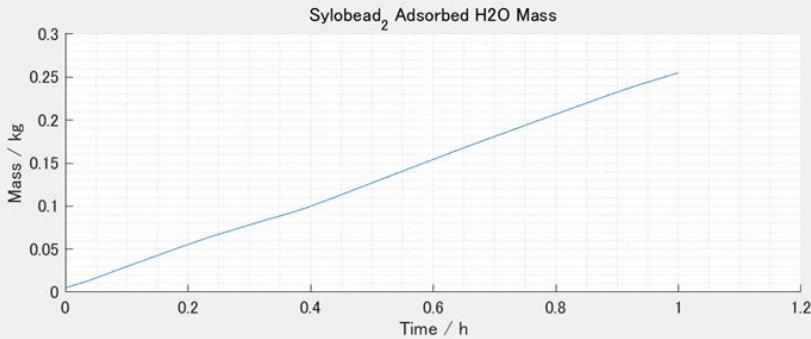
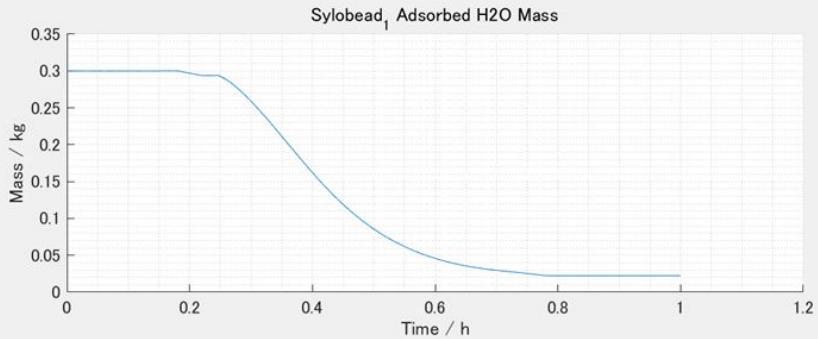
Zeolite5A<sub>1</sub> Adsorbed CO<sub>2</sub> Mass

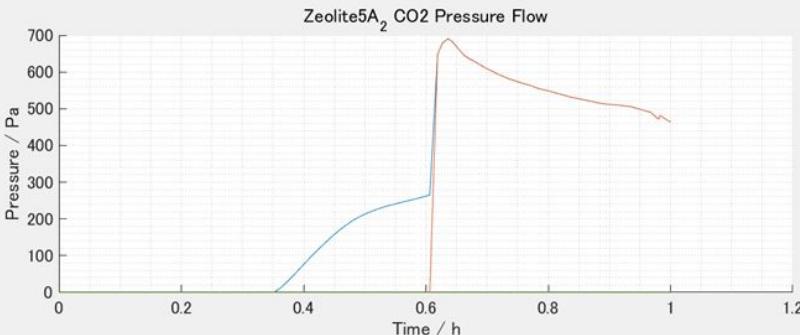
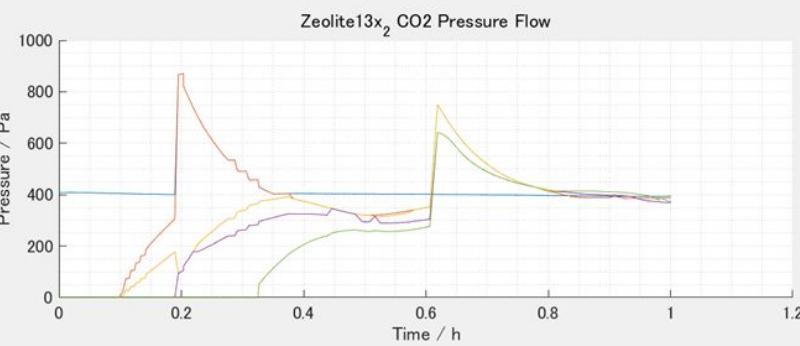
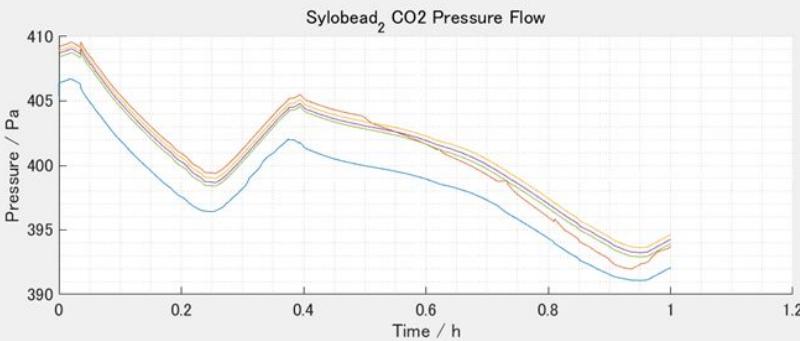
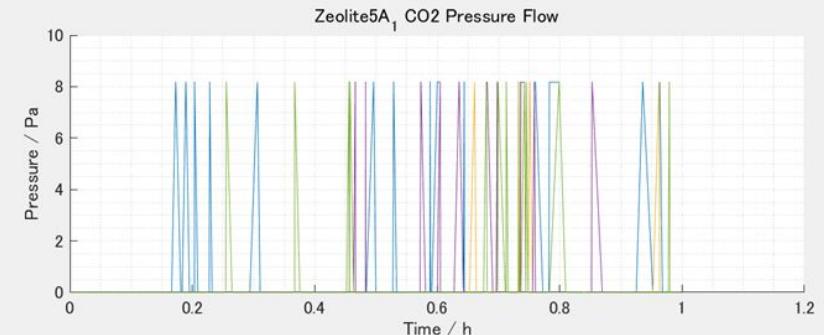
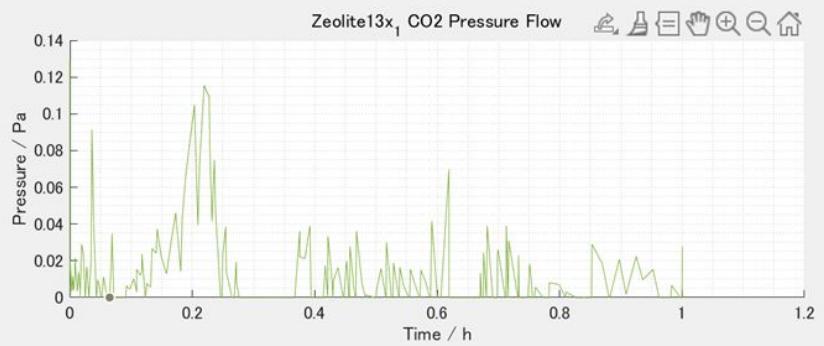
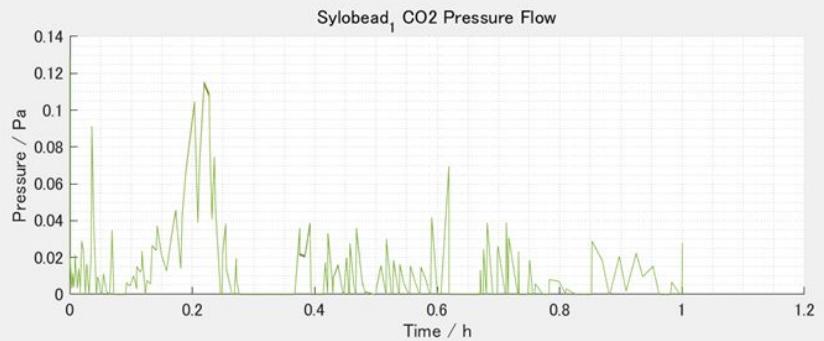


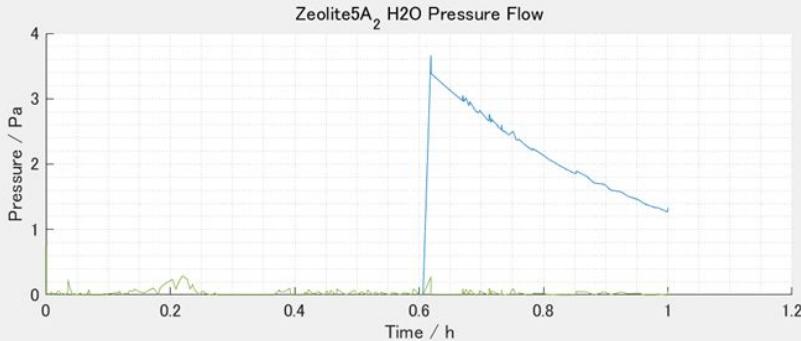
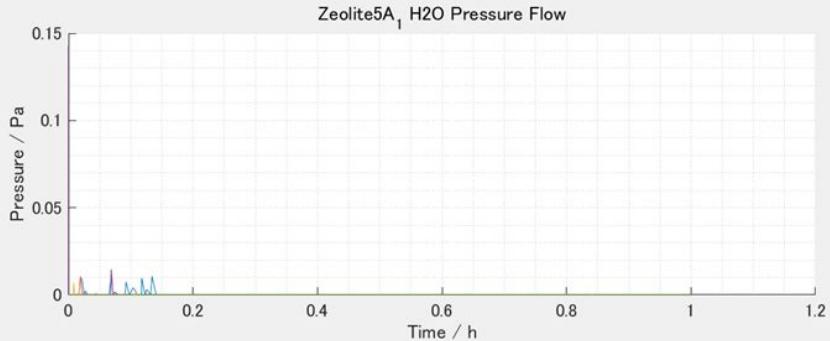
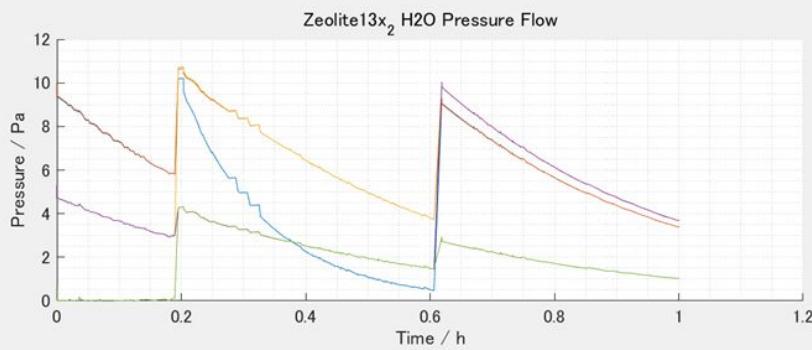
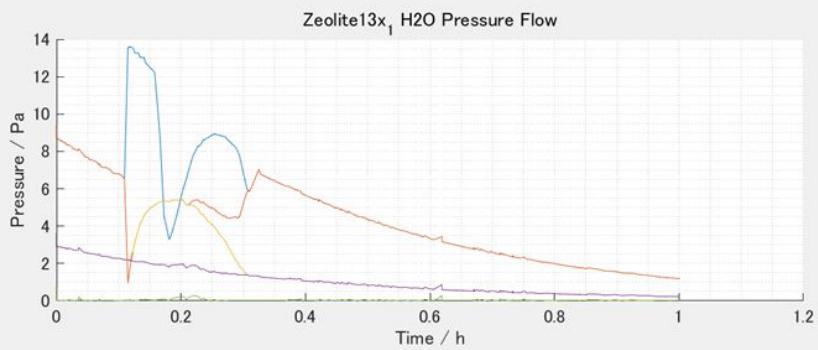
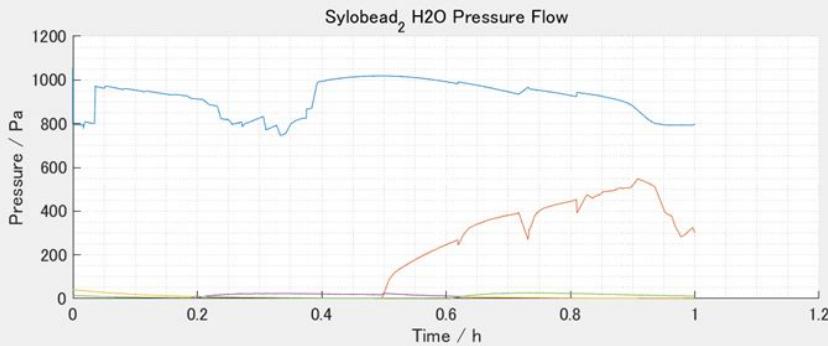
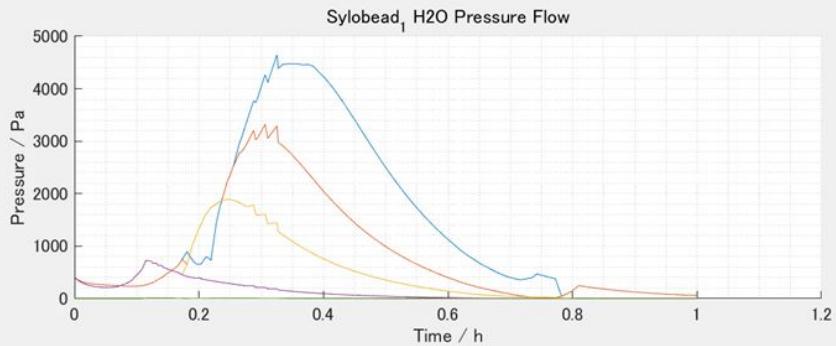
Zeolite5A<sub>2</sub> Adsorbed CO<sub>2</sub> Mass

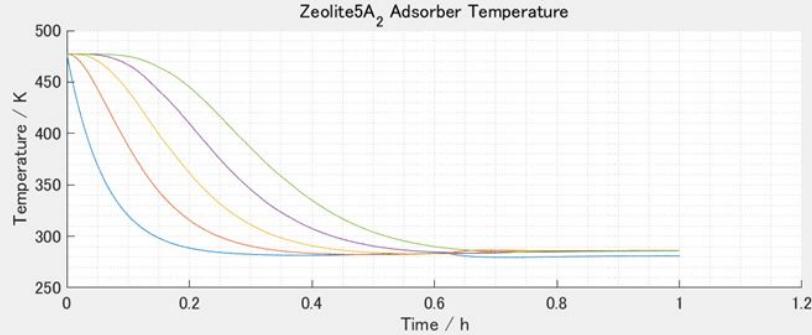
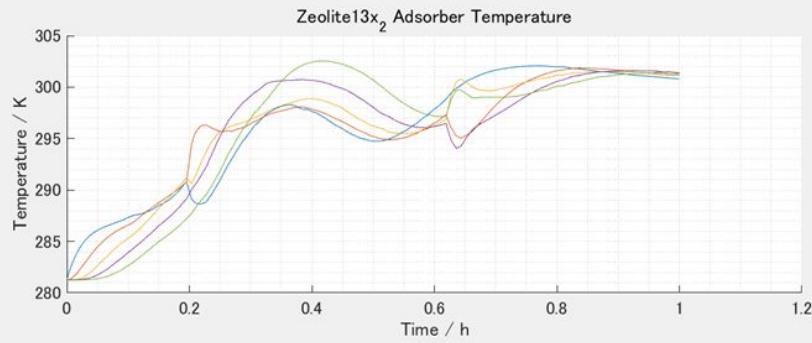
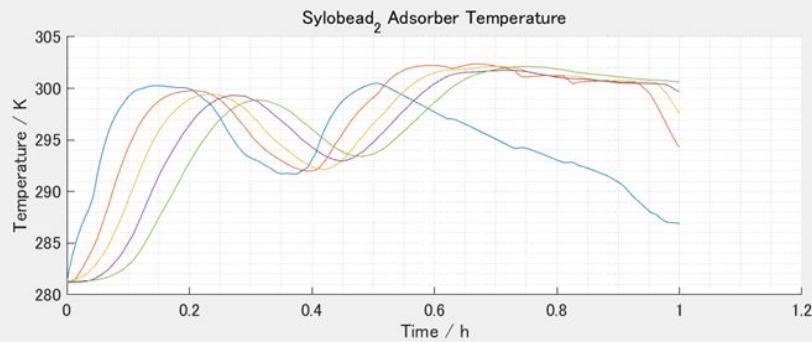
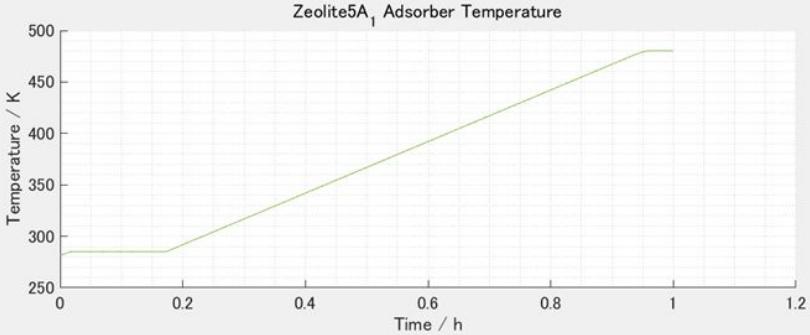
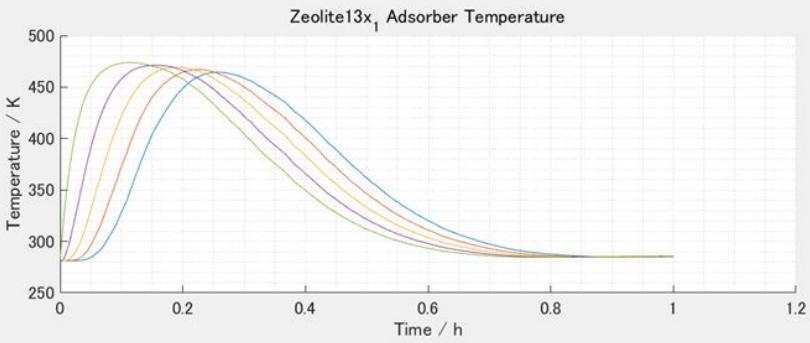
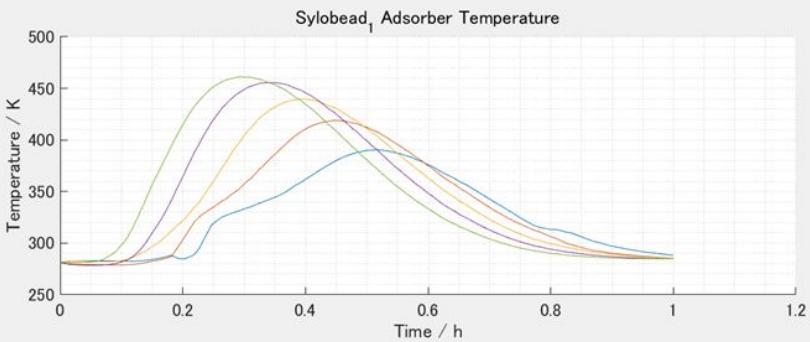


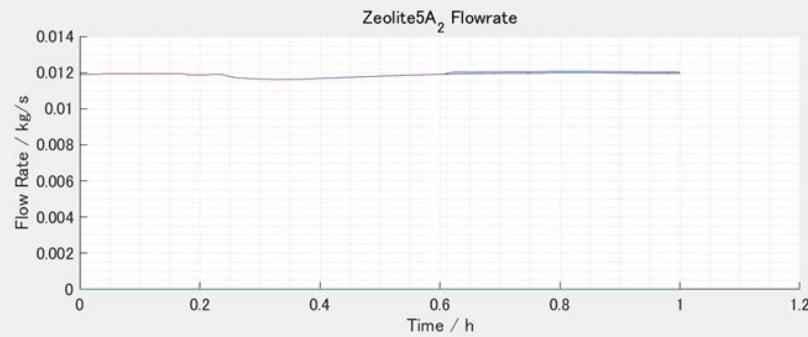
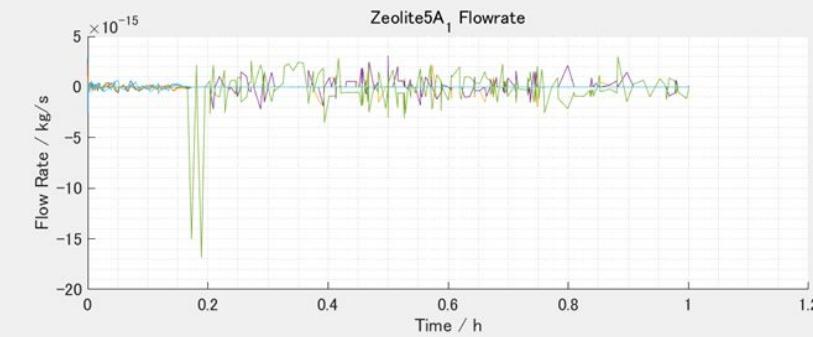
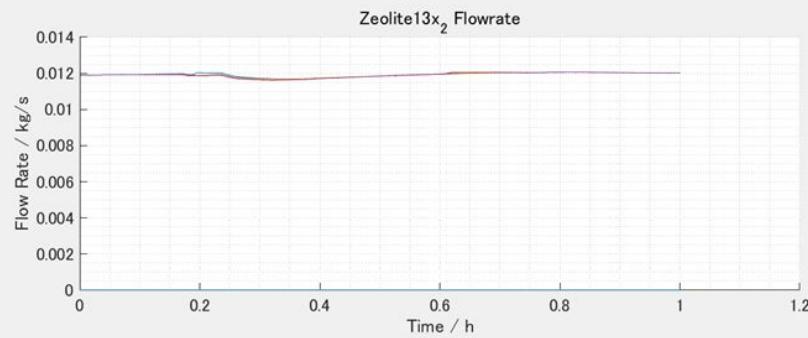
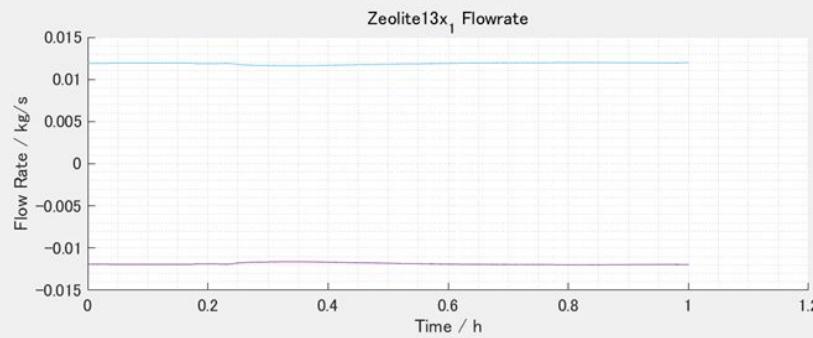
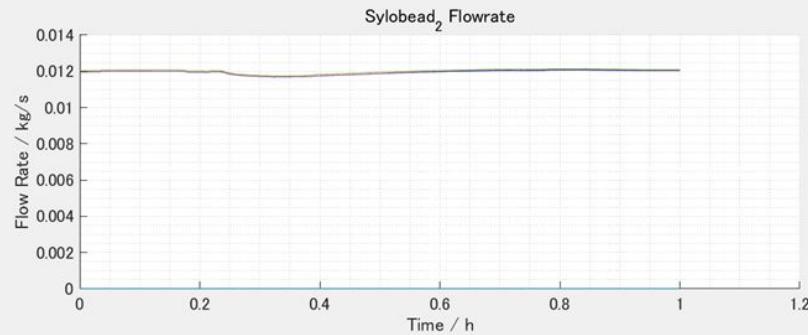
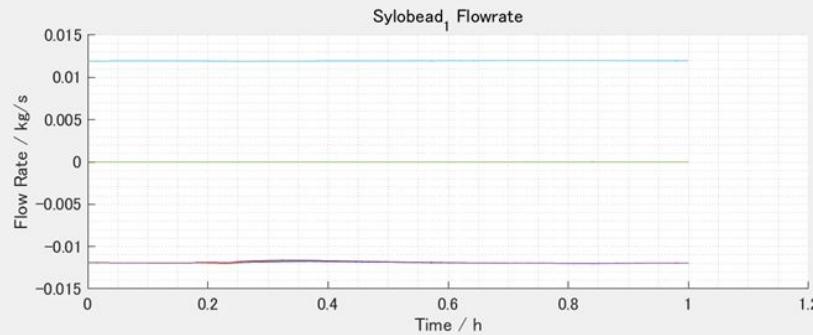


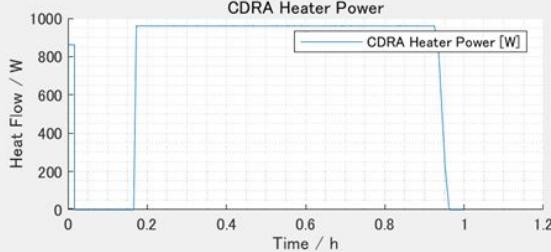
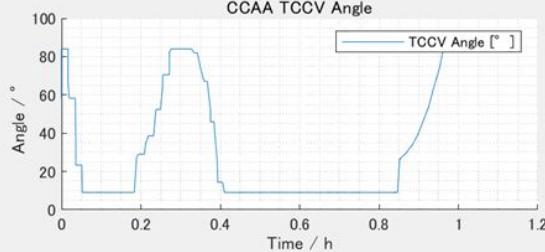
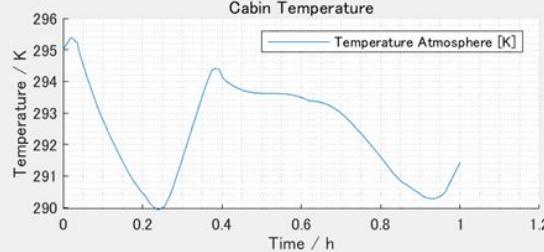
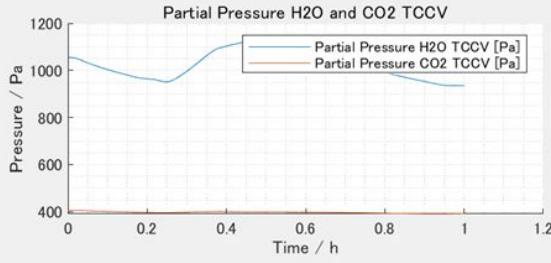
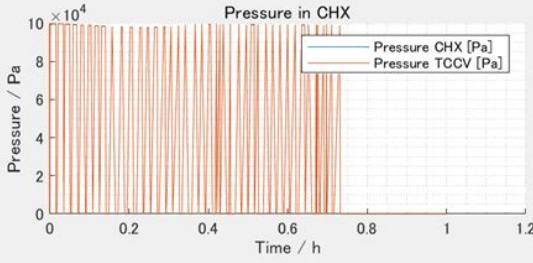
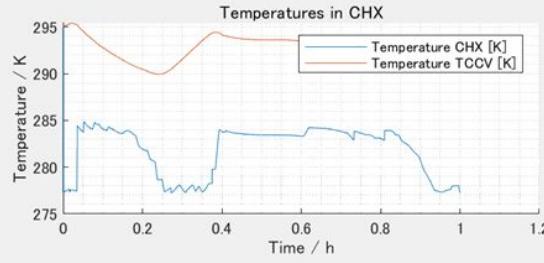
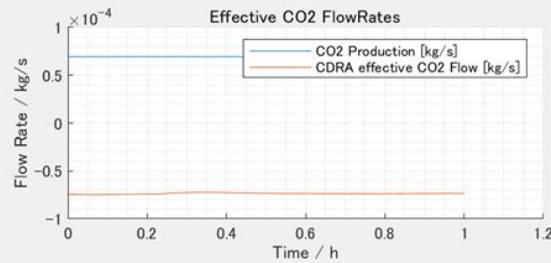
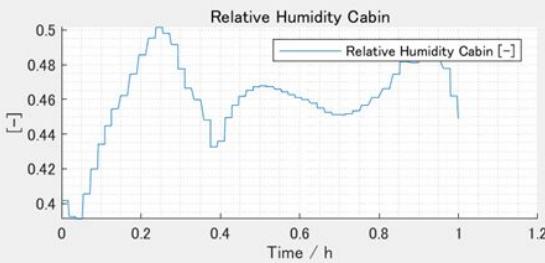
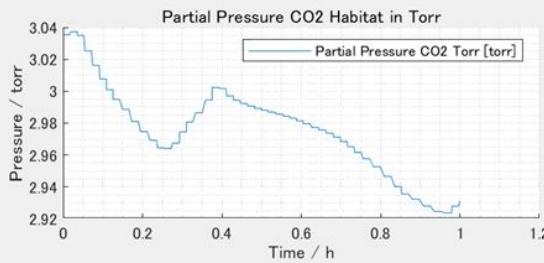
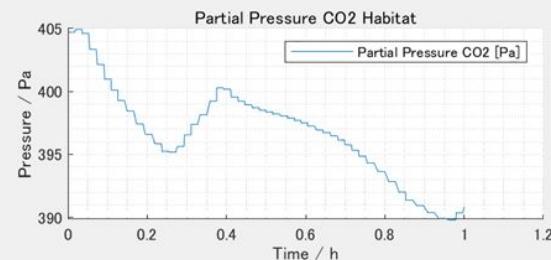
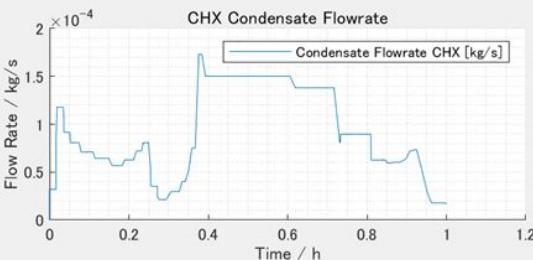
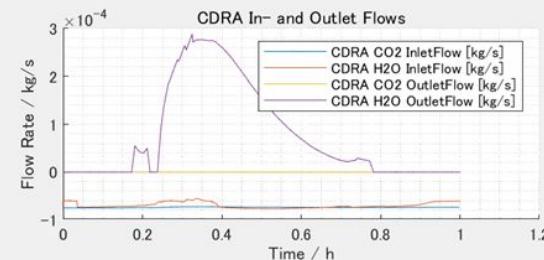


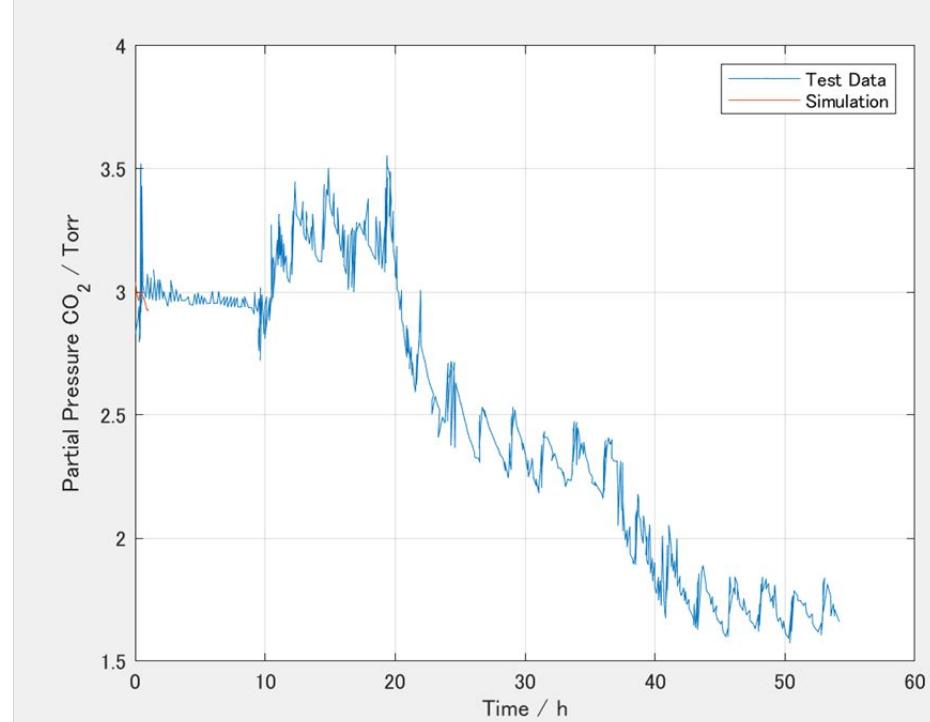
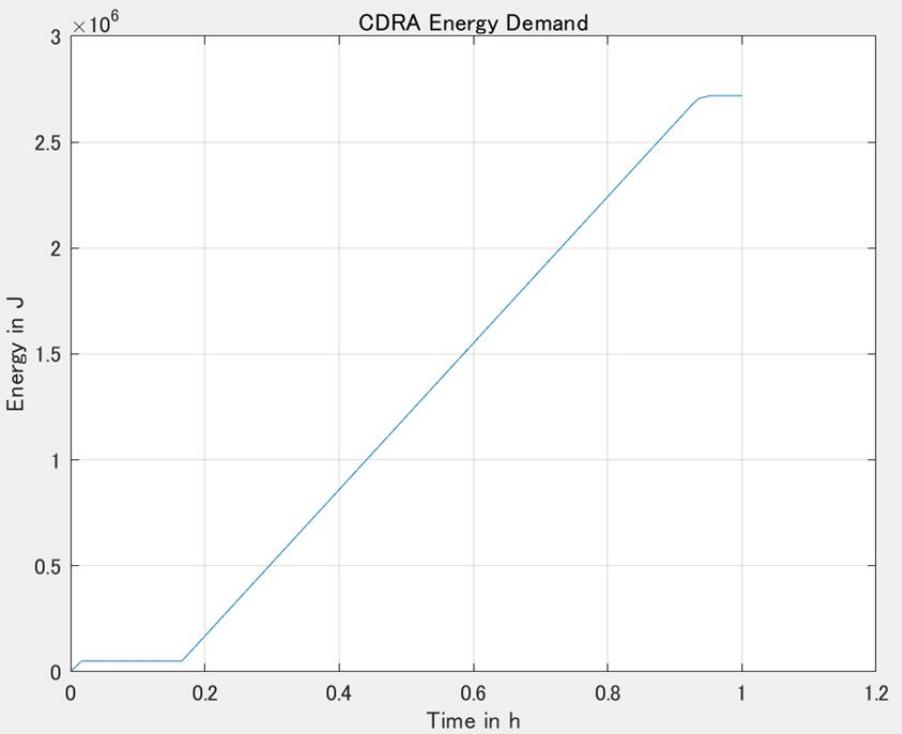








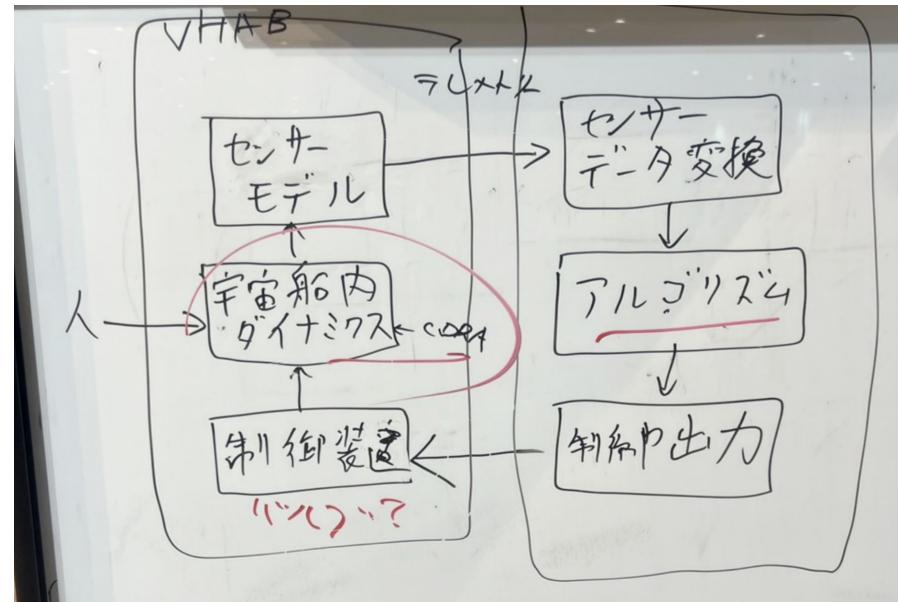
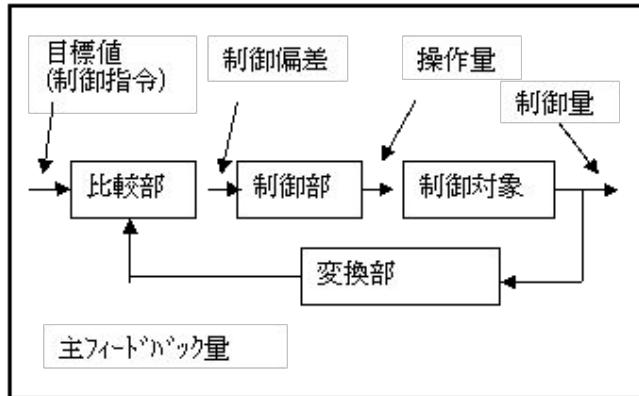


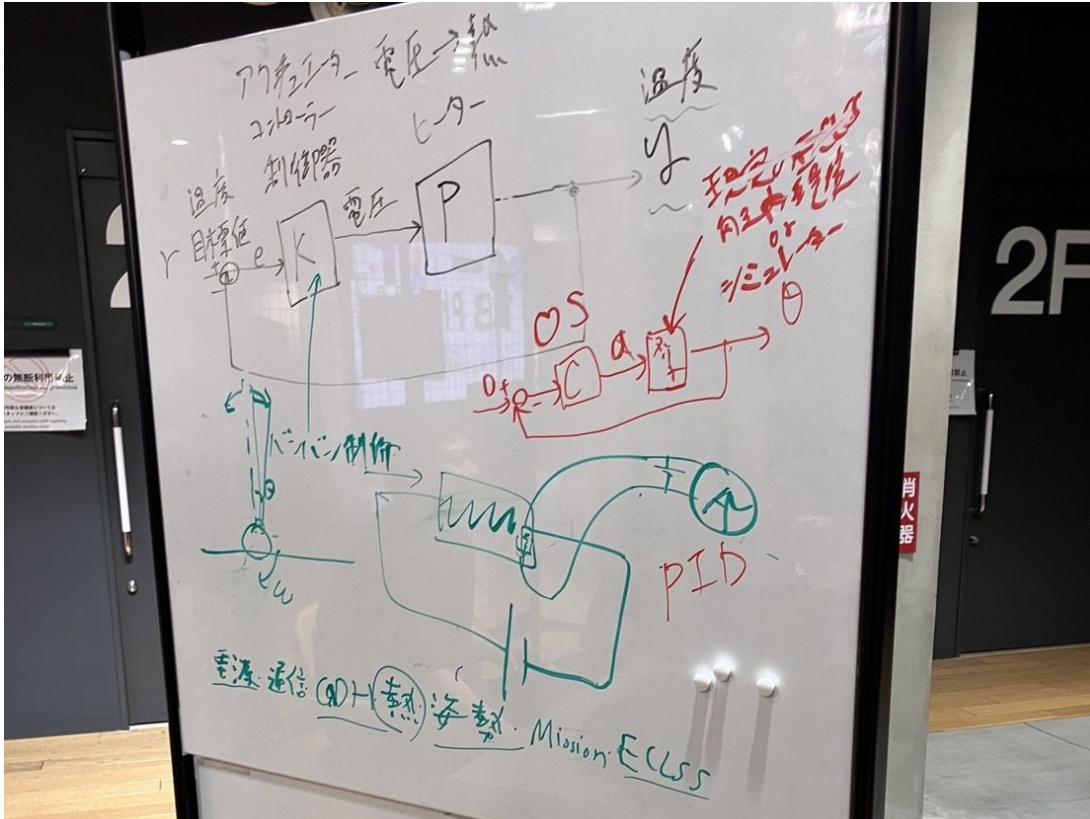


---

# ECLSS OS開発

# ECLSS OSの果たす役割 一制御系とはー





---

## ECLSS制御の現状調査

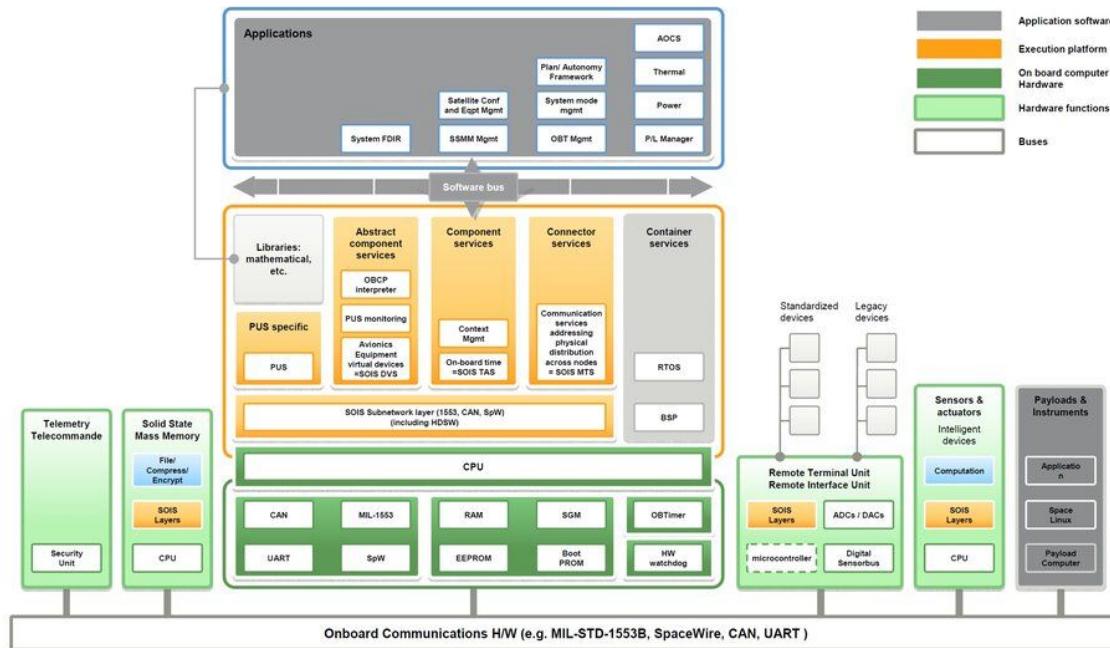
今のECLSSはどれくらい自動で動いているのか？

→クルーが自動で制御している部分は？

→地上局が遠隔で制御している部分は？

→軌道上コンポが自動的にモードや制御量を切り替えている部分は？

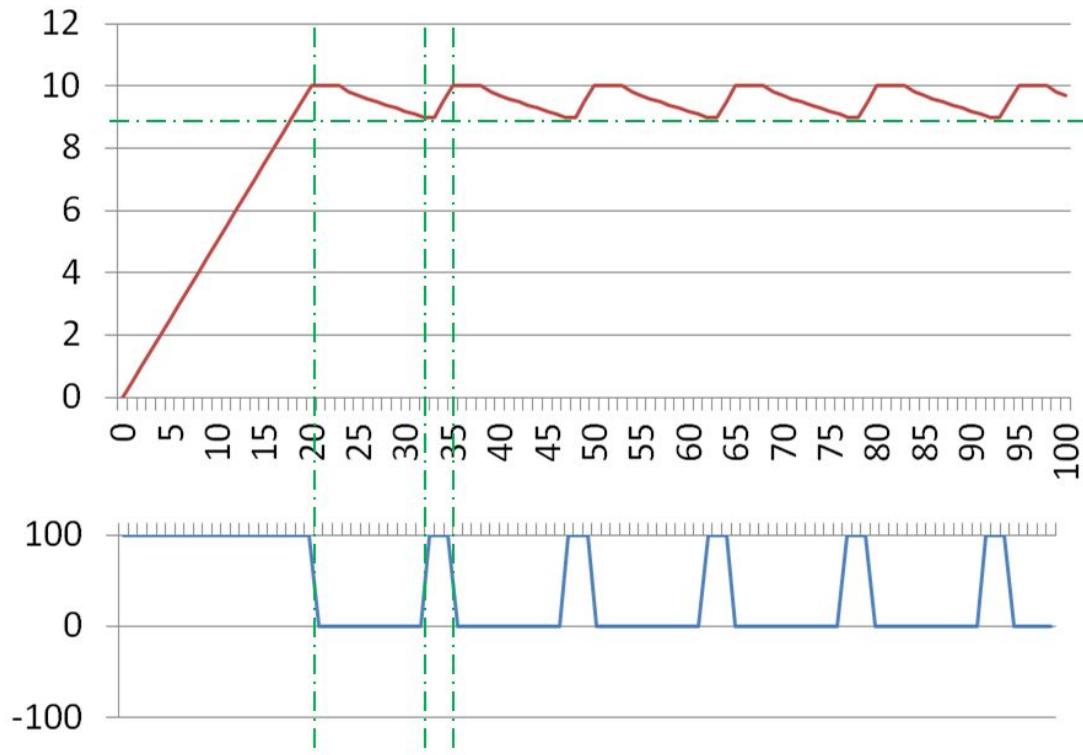
# ISS・Space ShuttleのOS (Onboard Computer)



# (ISS想定)

| 機能         | 制御                           | イベント                      | OS         | クルー                  | 地上局    |
|------------|------------------------------|---------------------------|------------|----------------------|--------|
| CDRA       | 自動 ON/OFF                    | ppCO <sub>2</sub> 閾値逸脱    | 異常検知・制御指令  | モニタリング<br>動作中の違和感を報告 | モニタリング |
| OGS 酸素発生装置 | 自動 ON/OFF                    | ppO <sub>2</sub> 閾値逸脱     |            |                      |        |
| 空気循環       | 流量制御                         | 気圧閾値逸脱                    |            |                      |        |
| 水再生        | 多段階プロセス                      | 水タンクの水位モニター               |            |                      |        |
| CDRA・除去タンク | 手動 ON/OFF                    | 故障・メンテナンス                 | 異常検知・クルー通知 | 修復・交換作業              | モニタリング |
| CDRA       | 機器出力レベル手動                    | 非定常状態                     |            |                      |        |
|            | 吸着剤交換                        | 吸着性能低下通知                  |            |                      |        |
| TCCS       | フィルター交換                      | 吸着性能低下通知                  |            |                      |        |
| 全体システム     | 制御ループの設定<br>動作スケジュール・パラメータ変更 | 高度な診断・問題特定(センサー有・総合的判断あり) | -          | -                    | 高度な診断  |

人工衛星の熱制御にも使われる  
**Bang-Bang制御**



# main.m

```
% インスタンス作成
cdra = CDRACo2Control; % CDRA制御インスタンス
sensor = CO2Sensor; % CO2センサーインスタンス

% シミュレーション設定
numSteps = 50; % シミュレーションステップ数

% データ保存用
co2Values = zeros(1, numSteps); % CO2濃度の履歴
cdraStates = zeros(1, numSteps); % CDRAのON/OFF状態の履歴

% シミュレーションループ
for i = 1:numSteps
    % センサーが次のCO2濃度を生成
    sensor = sensor.generateNextCO2(cdra.CDRAState, cdra.ppCO2UpperLimit, cdra.ppCO2LowerLimit);
    currentCO2 = sensor.getCO2();

    % CO2濃度を用いてCDRAを制御
    cdra = cdra.updateState(currentCO2);

    % データを保存
    co2Values(i) = currentCO2;
    cdraStates(i) = cdra.CDRAState; % true/false を 1/0 に変換して保存

    % 結果を表示
    fprintf('ステップ %d - CO2濃度: %d Pa, CDRA状態: %s\n', i, currentCO2, cdra.getState());
end

% グラフ1: CO2濃度
figure;
plot(1:numSteps, co2Values, '- ', 'LineWidth', 1.5);
xlabel('ステップ');
ylabel('CO2濃度 (Pa)');
title('時間経過によるCO2濃度の変化');
ylim([190 500]);
grid on;

% グラフ2: CDRAのON/OFF状態
figure;
stairs(1:numSteps, cdraStates, 'LineWidth', 1.5);
xlabel('ステップ');
ylabel('CDRAの状態 (1=ON, 0=OFF)');
title('時間経過によるCDRA状態の変化');
ylim([-0.5 1.5]);
grid on;
```

# CDRACo2Control.m

```
classdef CDRACo2Control
    properties
        ppCO2UpperLimit = 400; % CO2がこの値を超えたならCDRAをON (Pa)
        ppCO2LowerLimit = 200; % CO2がこの値を下回ったらCDRAをOFF (Pa)
        CDRAstate = false;      % CDRAの状態 (false: OFF, true: ON)
    end

    methods
        % コンストラクタ
        function obj = CDRACo2Control(torigerOn, torigerOff)
            if nargin > 0
                obj.ppcO2UpperLimit = torigerOn;
                obj.ppcO2LowerLimit = torigerOff;
            end
        end

        % CDRAの状態を更新
        function obj = updateState(obj, currentCO2)
            if currentCO2 >= obj.ppcO2UpperLimit
                obj.CDRAstate = true; % ON
            elseif currentCO2 <= obj.ppcO2LowerLimit
                obj.CDRAstate = false; % OFF
            end
        end

        % 現在の状態を文字列で返す
        function state = getState(obj)
            if obj.CDRAstate
                state = 'ON';
            else
                state = 'OFF';
            end
        end
    end
end
```

# CO2Sensor.m

```
classdef CO2Sensor
    properties
        currentCO2 = 350; % 初期CO2濃度 (Pa)
    end

    methods
        % 現在のCO2濃度を取得
        function co2 = getCO2(obj)
            co2 = obj.currentCO2;
        end

        % CO2濃度を設定
        function obj = setCO2(obj, newCO2)
            if newCO2 < 200 || newCO2 > 500
                error('CO2 concentration must be between 200 and 500 Pa.');
            end
            obj.currentCO2 = newCO2;
        end

        % CO2濃度を制御状態に基づいて更新
        function obj = generateNextCO2(obj, cdraState, ppCO2UpperLimit, ppCO2LowerLimit)
            % 制御状態がONの場合
            if cdraState
                % OFFになるまでCO2を下げる
                obj.currentCO2 = obj.currentCO2 - randi([10, 20]); % ランダムに減少
                obj.currentCO2 = max(obj.currentCO2, ppCO2LowerLimit - 10); % 下限を超えない
            else
                % ONになるまでCO2を上げる
                obj.currentCO2 = obj.currentCO2 + randi([10, 20]); % ランダムに増加
                obj.currentCO2 = min(obj.currentCO2, ppCO2UpperLimit + 10); % 上限を超えない
            end
        end
    end
end
```

