I. Introduction :

Ce chapitre présente la conception et la simulation d'un modèle de système de commande de climatisation pour un véhicule Toyota CHR hybride, réalisé dans l'environnement Simulink de MATLAB. Cette partie de notre travail s'inscrit dans le cadre d'un projet plus vaste visant à explorer et à développer des solutions pour minimiser la consommation énergétique du système de climatisation tout en assurant un confort thermique optimal pour les occupants .Dans cette section spécifique, nous nous concentrons sur l'intégration d'une approche hybride combinant une commande traditionnelle de type PID (Proportionnel-Intégral-Dérivé) et un système basé sur l'intelligence artificielle, spécifiquement un réseau neuronal récurrent (RNN), appliquée au système de climatisation. L'étude de ce modèle de simulation permet d'analyser le comportement de cette architecture de commande et d'évaluer son potentiel, notamment au regard de l'efficacité énergétique du système de climatisation. Ce modèle de simulation sert ainsi de plateforme pour étudier l'impact de différentes stratégies de commande sur la performance du système climatique du véhicule.

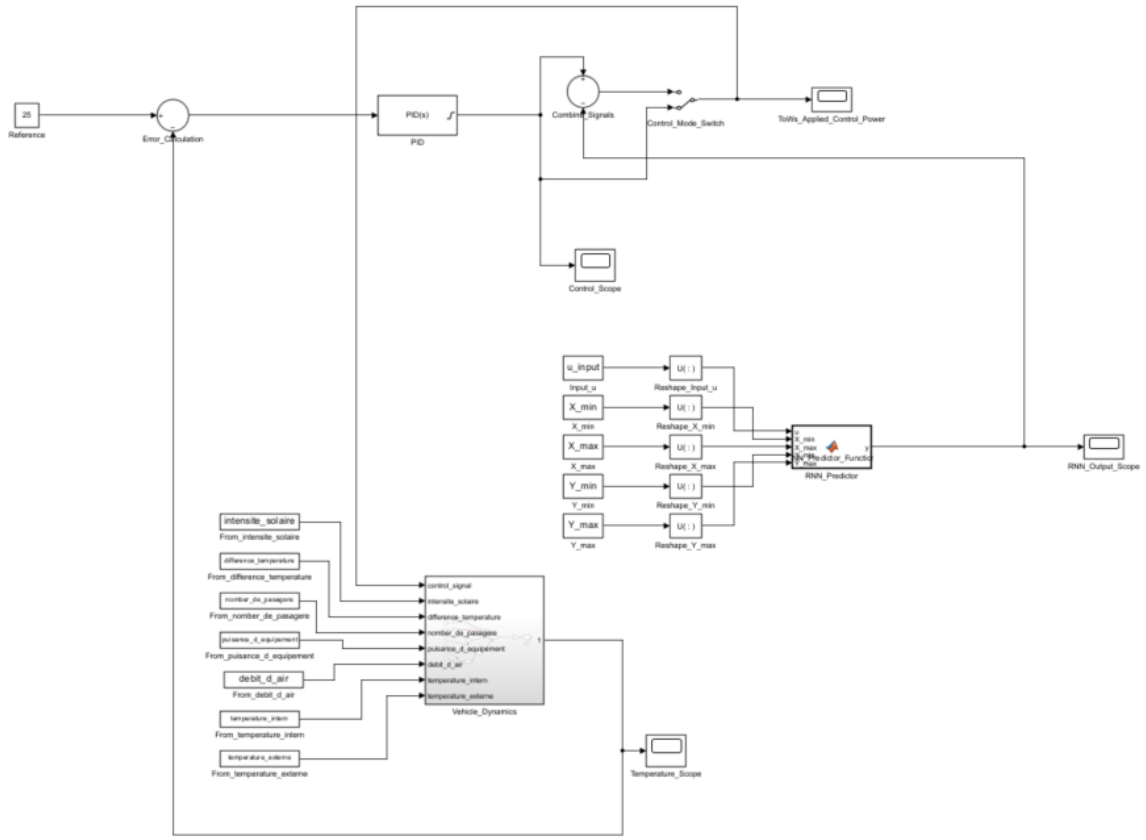
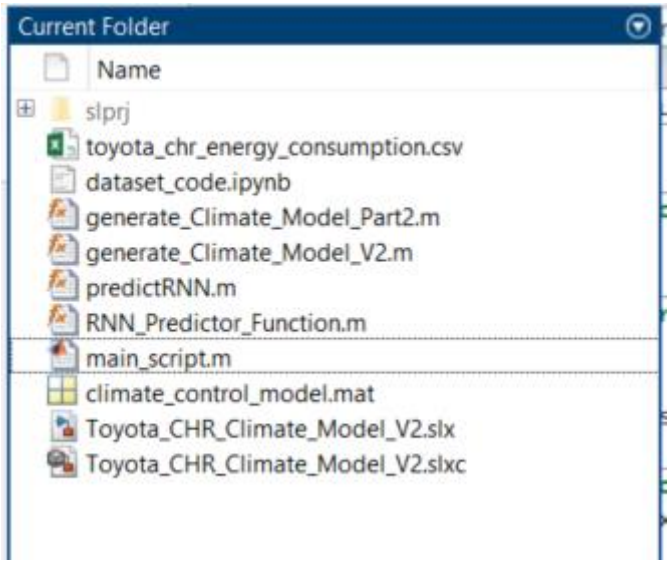


Figure 1 : Toyota_CHR_Climate_Model_V2.slx

Notre projet repose sur un modèle Simulink nommé Toyota_CHR_Climate_Model_V2.slx, généré à l'aide de scripts MATLAB (generate_Climate_Model_V2.m generate_Climate_Model_Part2.m) et s'appuyant sur des données réelles d'utilisation du véhicule (toyota_chr_energy_consumption.csv).

Notre projet est combiné de 8 fichiers principaux :

- main_script.m.
- generate_Climate_Model_V2.m,
- generate_Climate_Model_Part2.m,
- RNN_Predictor_Function.m,
- Toyota_CHR_Climate_Model_V2.slx.
- predictRNN
- climate_control_model
- toyota_chr_energy_consumption.csv



II. Description des Composants Clés de la Simulation :

Le modèle Simulink est composé de plusieurs blocs et sous-systèmes principaux :

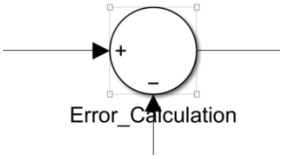
II.1. Bloc Reference :

Fournit la température de consigne souhaitée à l'intérieur du véhicule (25°C).

Figure 2: Bloc Reference.

II.2. Bloc Error_Calculation :

Calcule la différence (l'erreur) entre la température de consigne et la température réelle mesurée à l'intérieur du véhicule. Cette erreur est l'entrée principale du contrôleur PID.



II.3. Bloc PID :

Représente le contrôleur Proportionnel-Intégral-Dérivé traditionnel. Il génère un signal de commande basé sur l'erreur de température pour tenter de la ramener à la consigne.

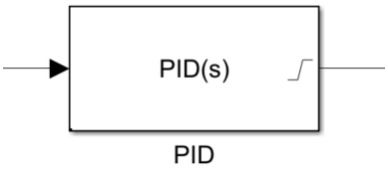
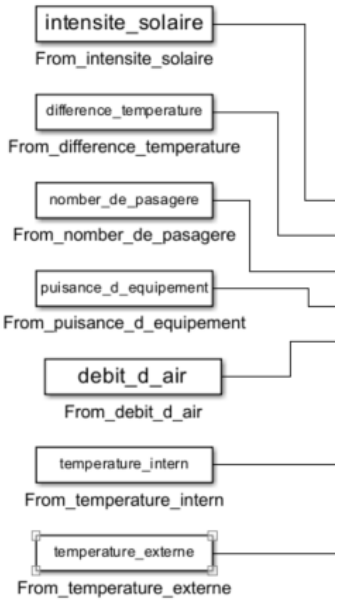


Figure 3 : PID.

II.4. Blocs From Workspace :



Ces blocs injectent les données temporelles des différents facteurs physiques (intensité solaire, température externe/interne, nombre de passagers, débit d'air, etc.) lues depuis le fichier CSV et préparées dans l'espace de travail MATLAB.

II.5. Bloc RNN_Predictor / RNN_Predictor_Function :

Ce bloc contient la logique du réseau neuronal RNN entraîné. Il prend en entrée les facteurs physiques actuels (éventuellement d'autres données comme l'historique), effectue une normalisation interne, utilise le modèle RNN pour générer une prédiction (par exemple, de la puissance requise ou du comportement futur), puis dé-normalise le résultat.

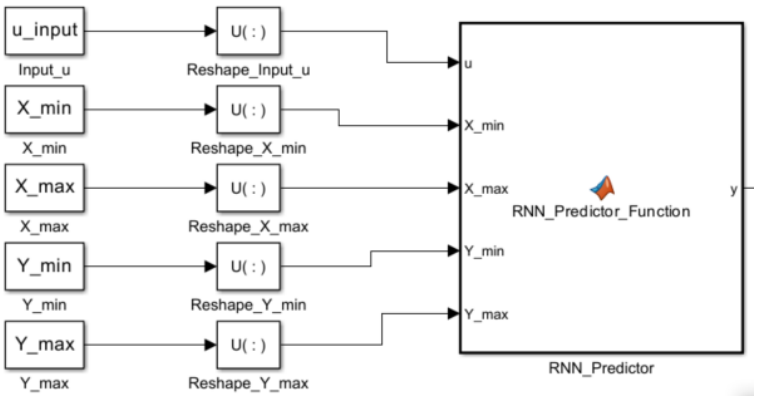


Figure 4:Bloc RNN_Predictor.

II.6. Bloc Combine Signals :

Combine le signal de commande généré par le PID traditionnel avec la prédiction ou le signal généré par le RNN. La méthode de combinaison (somme, pondération, etc.) définit la nature exacte de la commande hybride.

II.7. Bloc Control_Mode_Switch :

Un interrupteur manuel qui permet de sélectionner quel signal de commande sera appliqué au système physique : soit le signal du PID seulement, soit le signal combiné du PID et du RNN. Ce bloc est essentiel pour pouvoir comparer les deux modes de fonctionnement (traditionnel vs. hybride/AI).

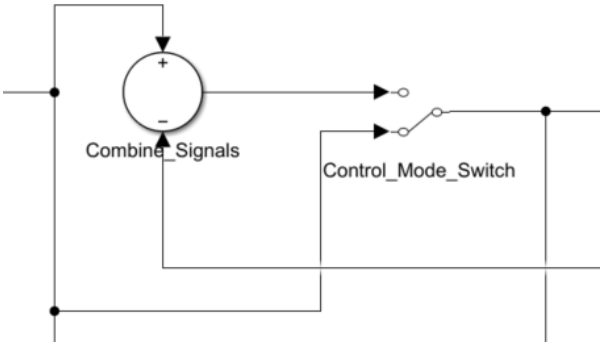


Figure 5:Bloc Combine Signals et Control_Mode_Switch

II.8. Sous-système Vehicle_Dynamics :

Ce sous-système complexe modélise le comportement thermique de l'habitacle du véhicule. Il reçoit les facteurs physiques externes (solaire, parois, passagers, air, etc.) ainsi que le signal de commande (puissance appliquée) et calcule l'évolution de
La température intérieure. Il inclut des paramètres physiques (surfaces vitrées, coefficients de transmission, sources de chaleur, capacité thermique de l'air, etc.).

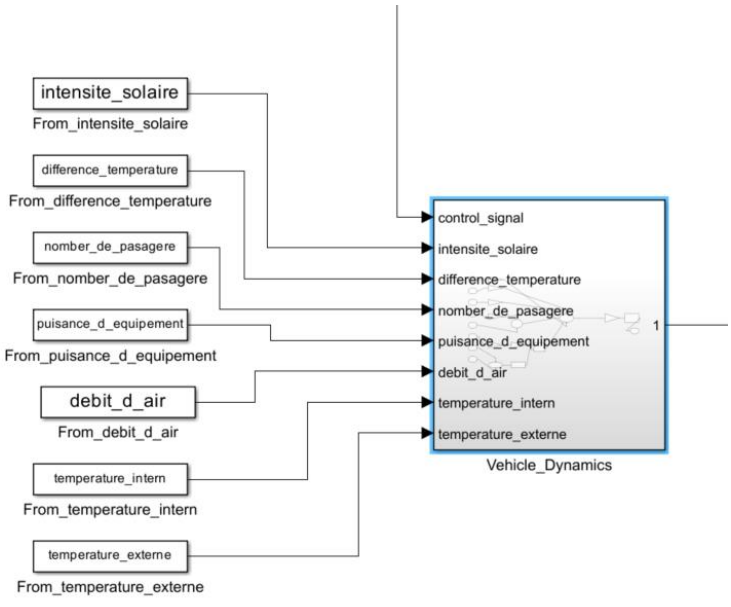


Figure 6 : Sous-système Vehicle_Dynamics

II.9. Blocs Scope :

Ces blocs permettent de visualiser graphiquement l'évolution des différents signaux au cours de la simulation (température intérieure, signal de commande, sortie du RNN.

III. **Préparation des Données** :

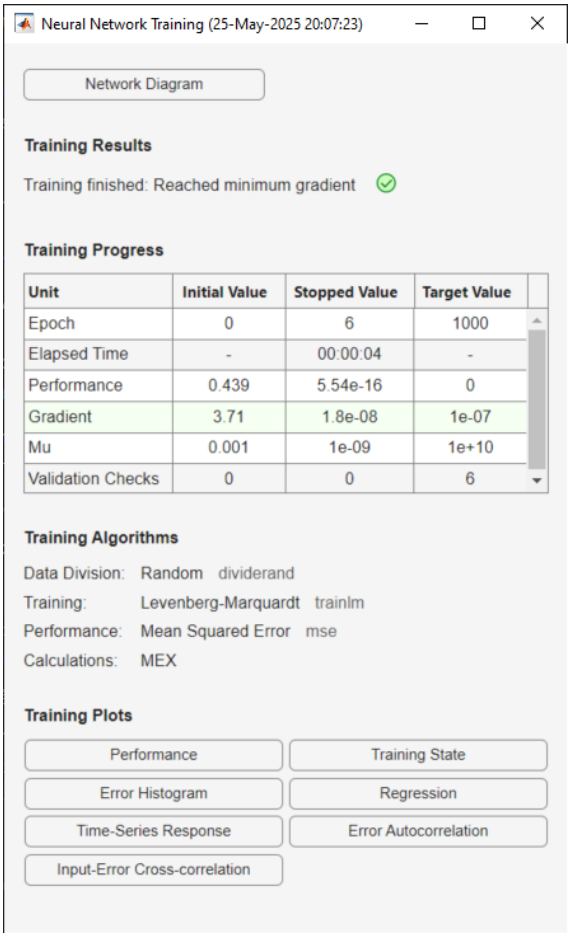
Les données historiques de consommation énergétique et des différents facteurs influençant le climat (températures, humidité, occupation, etc.) sont chargées et prétraitées. Une étape cruciale est la normalisation des données pour les adapter à l'entraînement du réseau neuronal. Les données d'entrée (`X`), de sortie (`Y`), ainsi que les valeurs min/max pour la normalisation (`X_min`, `X_max`, `Y_min`, `Y_max`), sont préparées.

```
%% === Étape 2 : Normalisation et injection dans Simulink ===
disp('Etape 2/5 : Normalisation des données...');
X_min = min(X, [], 2);
X_max = max(X, [], 2);
Y_min = min(Y);
Y_max = max(Y);

% Normalisation
X_norm = (X - X_min) ./ (X_max - X_min);
Y_norm = (Y - Y_min) / (Y_max - Y_min);

X_seq = con2seq(X_norm);
Y_seq = con2seq(Y_norm);

% Injection au format From Workspace (Simulink)
assignin('base', 'u_input', struct('time', 0, 'signals', struct('values', X(:,1)', 'dimensions', 16)));
assignin('base', 'X_min', struct('time', 0, 'signals', struct('values', X_min', 'dimensions', 16)));
assignin('base', 'X_max', struct('time', 0, 'signals', struct('values', X_max', 'dimensions', 16)));
assignin('base', 'Y_min', struct('time', 0, 'signals', struct('values', Y_min, 'dimensions', 1)));
assignin('base', 'Y_max', struct('time', 0, 'signals', struct('values', Y_max, 'dimensions', 1)));

disp('Paramètres injectés dans Simulink.');

```

Figure 7 : la normalisation des données.

IV. **Entraînement du Réseau Neuronal (RNN)** :

Un réseau neuronal récurrent est entraîné à partir des données préparées. Ce RNN a pour rôle d'apprendre les relations complexes entre les facteurs environnementaux, l'utilisation du véhicule, et la consommation énergétique/besoin en puissance de la climatisation. Le modèle entraîné est sauvegardé dans climate_control_model.mat.



Le réseau neuronal a terminé son entraînement avec succès, atteignant un gradient minimum en seulement 6 époques. Ceci a été réalisé très rapidement, avec un temps écoulé d'environ 4 secondes. La performance finale, mesurée par l'Erreur Quadratique Moyenne, est extrêmement basse à 5.54e-16. Cette faible valeur indique que le réseau a très bien appris et modélisé les données d'entraînement. L'algorithme d'entraînement utilisé était Levenberg-Marquardt ('trainlm').

V. Génération du Modèle Simulink :

Les scripts `generate_Climate_Model_V2.m` et `generate_Climate_Model_Part2.m` construisent dynamiquement le modèle Simulink. Ils ajoutent et connectent les différents blocs représentant les composants du système et la physique du véhicule. La fonction MATLAB générée pour l'inférence du RNN (`predictRNN.m`, utilisée dans le bloc `RNN_Predictor_Function`) est également intégrée.

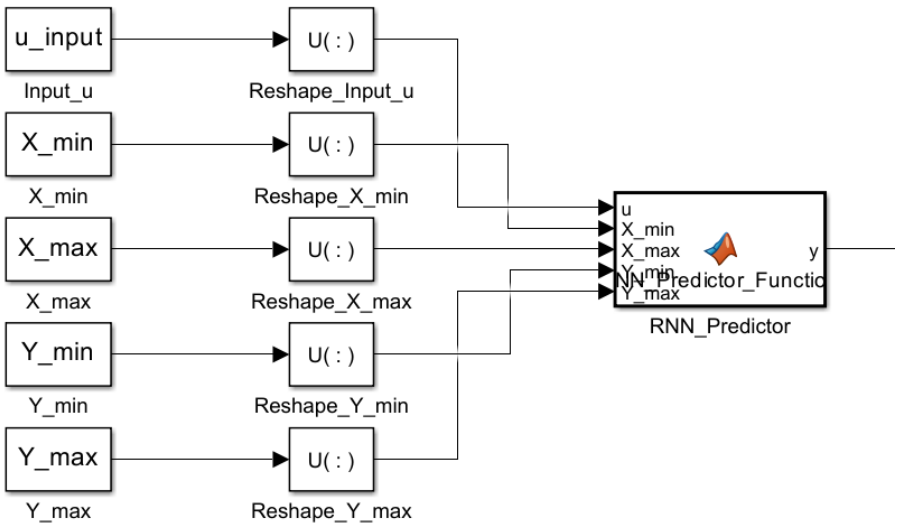


Figure 9: le block RNN_Predictor_Function.

```
function y = RNN_Predictor_Function(u, X_min, X_max, Y_min, Y_max)
%#codegen

%% Initializations for code generation
y = 0; % Initialize output

%% Protection against division by zero
epsilon = 1e-6;
range_x = X_max(:) - X_min(:);
range_x(abs(range_x) < epsilon) = epsilon;

%% Input Normalization (mapminmax from training)
x_norm = (u(:) - X_min(:)) ./ range_x;

%% Persistent delay states for RNN
persistent ai1 ai2
if isempty(ai1)
    ai1 = zeros(20, 2); % Layer 1 delay states
end
if isempty(ai2)
    ai2 = zeros(10, 2); % Layer 2 delay states
end

%% Call RNN prediction function
[y_norm, ai1, ai2] = predictRNN(x_norm, ai1, ai2);

%% Denormalize output
y = double(y_norm) * (Y_max - Y_min) + Y_min;

y = max(min(y, 1), -1);

end
```

Figure 8: le code de block RNN_Predictor_Function.

VI. Simulation :

Le modèle Simulink est exécuté pour simuler le comportement du système de climatisation sur une période donnée, en utilisant les données réelles comme entrées. Pendant la simulation, les signaux clés sont surveillés et visualisés via des blocs "Scope".

VII. Fonctionnement Étape par Étape de la Simulation et Analyse Potentielle :

La simulation peut être exécutée dans deux modes principaux, sélectionnés via le bloc Control_Mode_Switch, pour analyser l'impact de l'approche hybride :

VII.1. Mode "PID Seulement" :

L'interrupteur est configuré pour laisser passer uniquement le signal généré par le contrôleur PID traditionnel.

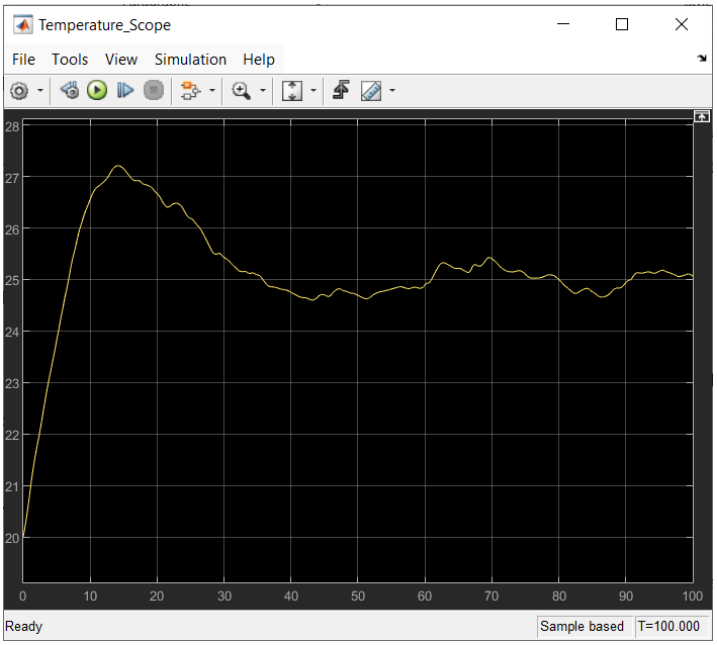


Figure 10 : scope de Température.

VII.2. Mode "PID + AI(RNN) :

L'interrupteur est configuré pour laisser passer le signal combiné (ou modifié) par l'intégration de la sortie du RNN (via le bloc Combine Signals). La simulation est lancée à nouveau avec les mêmes conditions d'entrée (les mêmes données physiques).

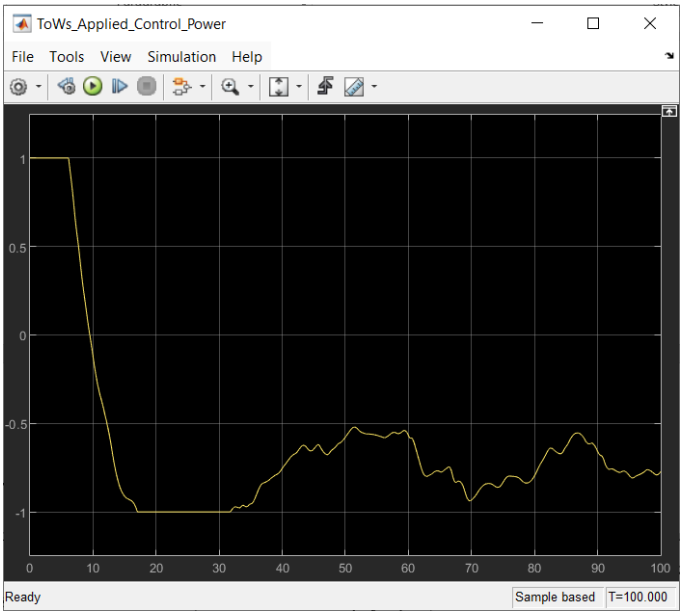


Figure 11: signal de control.

L'analyse des graphiques affichés par le Scope pour les deux modes permet de visualiser et de comparer la puissance consommée par le système sous l'effet de la commande traditionnelle versus la commande hybride. L'examen du Scope 1 confirme quant à lui la capacité du système à maintenir la température intérieure désirée dans les deux configurations. Cette comparaison visuelle offre un aperçu du potentiel de l'approche basée sur l'IA pour influencer le comportement du système, notamment en termes d'efficacité.

VIII. Conclusion :

Ce chapitre a détaillé la structure et le fonctionnement du modèle Simulink représentant le système de climatisation hybride pour la Toyota CHR. En intégrant un contrôleur PID traditionnel et un réseau neuronal RNN entraîné sur des données réelles, le modèle permet de simuler et de comparer différentes stratégies de commande. La visualisation des signaux clés

via les blocs Scope, en particulier la puissance appliquée, offre un aperçu de l'impact de l'approche hybride sur le comportement du système. Bien que ce chapitre se concentre sur la description du modèle de simulation, cette plateforme constitue un outil précieux pour analyser et potentiellement valider des améliorations de l'efficacité énergétique du système de climatisation dans le cadre de notre projet global.

Le bloc PID modélise un contrôleur Proportionnel-Intégral-Dérivé, une méthode de commande largement utilisée dans l'automatisation et la régulation. Son fonctionnement repose sur le calcul d'un signal de commande basé sur l'erreur** actuelle entre la sortie mesurée (température intérieure) et la consigne désirée. La composante **Proportionnelle (P)** réagit directement à l'amplitude de l'erreur présente. La composante **Intégrale (I)** accumule les erreurs passées, ce qui aide à éliminer l'erreur statique résiduelle sur le long terme. Enfin, la composante **Dérivée (D)** anticipe l'erreur future en se basant sur la vitesse de changement de l'erreur actuelle, ce qui permet d'améliorer la réactivité du système et de réduire les dépassements. Le signal de commande final généré par le bloc PID est la somme pondérée de ces trois actions (P, I, et D), visant à ramener l'erreur à zéro et à stabiliser la sortie (la température) à la valeur de consigne.**