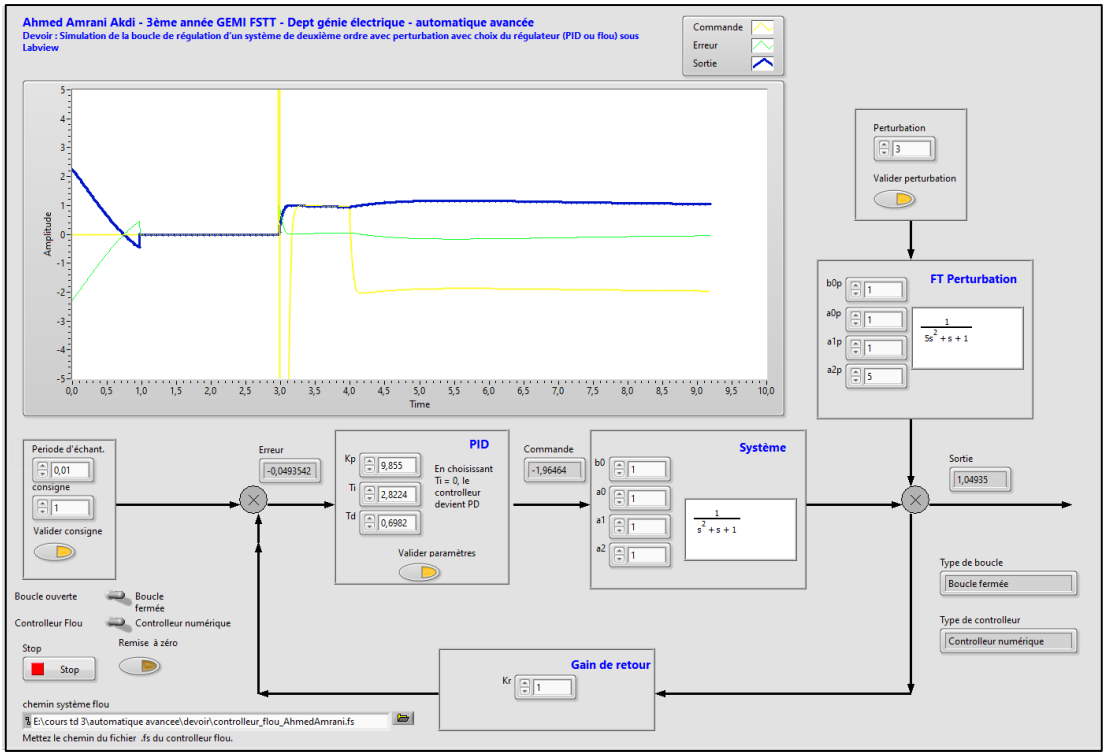
UNIVERSITÉ ABDELMALEK ESSAÂDI
FACULTÉ DES SCIENCES ET TECHNIQUES
TANGER

Département Génie Electrique

Devoir : Simulation de la boucle de régulation d'un système de deuxième ordre avec perturbation avec choix du régulateur (PID ou flou) sous LabVIEW

Module : Automatique avancée

3ème année cycle d'ingénieur GEMI



Réalisé par :

AMRANI AKDI Ahmed

Date :

06/02/2021

Encadré par :

Monsieur Amami

Table des matières

I- But du devoir :

Le but du devoir est la simulation de la régulation d'un système de deuxième ordre de fonction de transfert :

$$G(p) = \frac{b_0}{a_0 + a_1 p + a_2 p^2}$$

Avec un régulateur PID ou flou sous LabVIEW.

II- Mise en œuvre du système :

- ## Relation récurrente du système :

L'expression du système peut être écrite comme :

$$G(p) = \frac{b_0}{a_2 * (p - p_1) * (p - p_2)}$$

Où :

$$p_{1,2} = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0 a_2}}{2a_2}$$

D'une autre part, le système échantillonné et bloqué avec un bloqueur d'ordre zéro a comme transformé en z :

$$G(z) = \frac{z-1}{z} * TZ\left(\frac{G(p)}{p}\right)$$

Selon la nature des pôles, on distingue trois cas :

– ### Pôle double réel :

Dans ce cas l'expression du système devient :

$$G(p) = \frac{b_0}{a_2 * (p - p_1)^2}$$

Avec :

$$p_1 = \frac{-a_1}{2a_2}$$

On aura donc :

$TZ\left(\frac{G(p)}{p}\right) = \sum_{pôles} Res \frac{G(p)}{p(1-e^{pT}z^{-1})}$

$Res(0) = \lim_{p\to 0} p * \frac{1}{p} * \frac{b_0}{a_2 * (p-p_1)^2} * \frac{1}{(1-e^{pT}z^{-1})} = \frac{b_0}{a_2 p_1^2} * \frac{z}{z-1}$

$Res(p_1) = \frac{1}{1!} \lim_{p\to p_1} \frac{d}{dp}\left[(p-p_1)^2 * \frac{1}{p} * \frac{b_0}{a_2 * (p-p_1)^2} * \frac{1}{(1-e^{pT}z^{-1})}\right] = -\frac{b_0}{a_2 p_1^2}\frac{z}{z-z_1} + \frac{b_0}{a_2 p_1}\frac{Tz_1 z}{(z-z_1)^2}$

Avec $z_1 = e^{Tp_1}$

D'où :

$$TZ\left(\frac{G(p)}{p}\right) = Res(0) + Res(p_1) = \frac{Az + B}{(z-z_1)^2}$$

Avec :

$$A = \frac{b_0}{a_2 p_1^2}(z_1 + p_1 T z_1 + 1 - 2b_0 z_1)$$

$$B = \frac{b_0}{a_2 p_1^2}(-b0 z_1 - b_0 p_1 T z_1 + b_0 z_1^2)$$

Finalement, la relation récurrente du système échantillonné et bloqué dans le cas d'un pôle double est :

$$y[n] = Ae[n-1] + Be[n-2] + 2\,z_1 y[n-1] - z_1^2\, y[n-2];$$

– Deux Pôles réels :

On aura donc :

$$TZ\left(\frac{G(p)}{p}\right) = \sum_{pôles} Res \frac{G(p)}{p(1-e^{pT}z^{-1})}$$

$$Res(0) = \lim_{p\to 0} p * \frac{1}{p} * \frac{b_0}{a_2(p-p_1)(p-p_2)} * \frac{1}{(1-e^{pT}z^{-1})} = \frac{b_0}{a_2 p_1 p_2} * \frac{z}{z-1}$$

$$Res(p_1) = \lim_{p\to p_1} (p-p_1) * \frac{1}{p} * \frac{b_0}{a_2(p-p_1)(p-p_2)} * \frac{1}{(1-e^{pT}z^{-1})} = \frac{b_0}{a_2 p_1 p_2 (p_1-p_2)} \frac{z}{(z-z_1)}$$

$$Res(p_2) = \lim_{p\to p_2} (p-p_2) * \frac{1}{p} * \frac{b_0}{a_2(p-p_1)(p-p_2)} * \frac{1}{(1-e^{pT}z^{-1})} = \frac{b_0}{a_2 p_1 p_2 (p_2-p_1)} \frac{z}{(z-z_2)}$$

Avec :

$$p_{1,2} = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0 a_2}}{2a_2}$$

$$z_1 = e^{Tp_1}$$

$$z_2 = e^{Tp_2}$$

D'où

$$TZ\left(\frac{G(p)}{p}\right) = Res(0) + Res(p_1) + Res(p_2) = A\frac{Bz + C}{z^2 - z(z_1+z_2) + z_1 z_2}$$

Avec :

$$A = \frac{b_0}{a_2 p_1 p_2}$$

$$B = (-z_2 p_1 + p_2 z_1 - p_2 + p_1)/(p_1 - p_2)$$

$$C = ((p_1 - p_2) z_1 z_2 + p_2 z_2 - p_1 z_1)/(p_1 - p_2)$$

Finalement, la relation récurrente du système échantillonné et bloqué dans le cas de deux pôles réels est :

$$y[n] = ABe[n-1] + ACe[n-2] + y[n-1](z_1 z_2) - y[n-2] z_1 z_2;$$

– Deux pôles complexes :

La relation récurrente dans le cas de deux pôles réels distincts est valable pour le cas de deux pôles complexes, cependant le « formula node » de LabVIEW n'admet pas des variables complexes, d'où la nécessité de simplifier les expressions, en particulier, des coefficients A, B et C en utilisant :

$$p_{1,2} = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0 a_2}}{2a_2} = -\frac{a_1}{2a_2} \pm j\sqrt{4a_0 a_2 - a_1^2} = a + bj$$

$$z_{1,2} = e^{Tp_{1,2}} = e^{T(a \pm bj)} = e^{aT} e^{\pm Tbj} = Re^{\pm \alpha j} = c \pm dj$$

Et on trouve :

$$A = \frac{b_0}{a_2(a^2 + b^2)}$$

$$B = 1 - c + a * \frac{d}{b}$$

$$C = R^2 - a * \frac{d}{b} - c$$

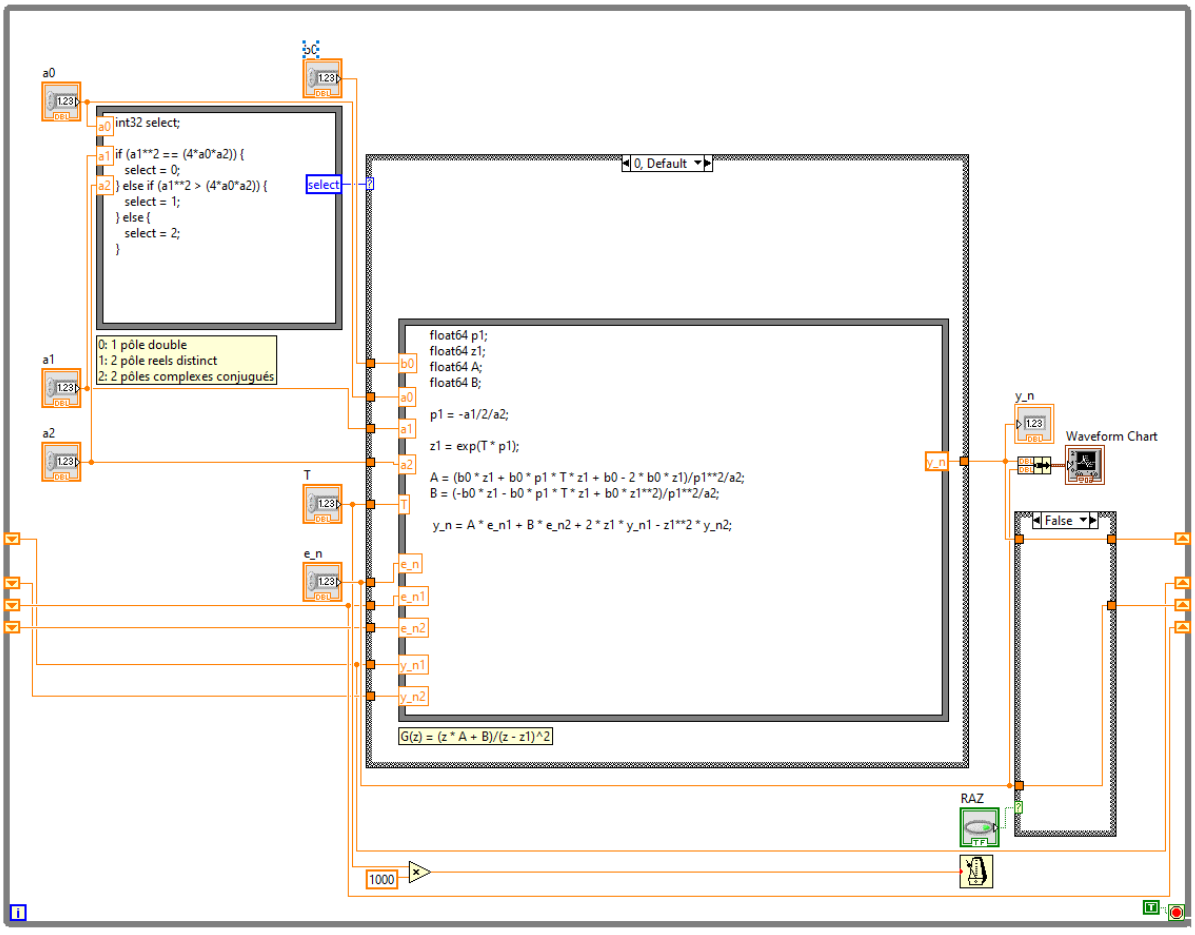
La relation récurrente devient la même que dans le cas des deux pôles réels distinct et est :

$$y[n] = ABe[n-1] + ACe[n-2] + y\{n-1\}2c - y[n-2]R^2;$$

- Implémentation sur LabVIEW du système :

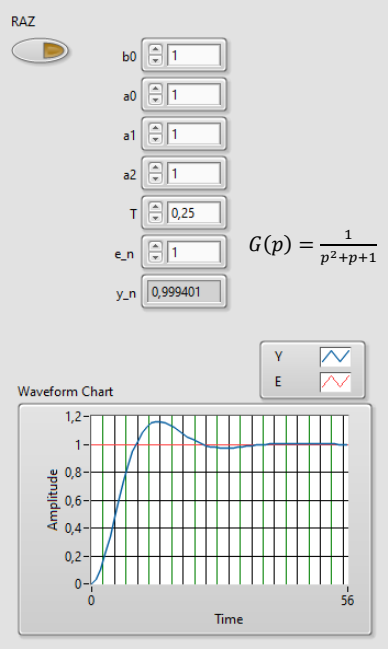
Pour implémenter l'algorithme du système on utilisera l'élément « formula node » pour calculer les coefficients et la valeur actuelle de la sortie, pour obtenir les valeurs précédentes de l'entrée et la sortie ($e[n-1]$, $e[n-2]$, $y[n-1]$, et $y[n-2]$) on utilise des « shift registers », le choix de la relation se fera grâce à une structure « case » et le sélecteur sera la sortie d'un « formula node » dans lequel on compare le discriminant de l'expression du pôle, finalement, tout le système sera mis dans une boucle cadencé par le pas d'échantillonnage.

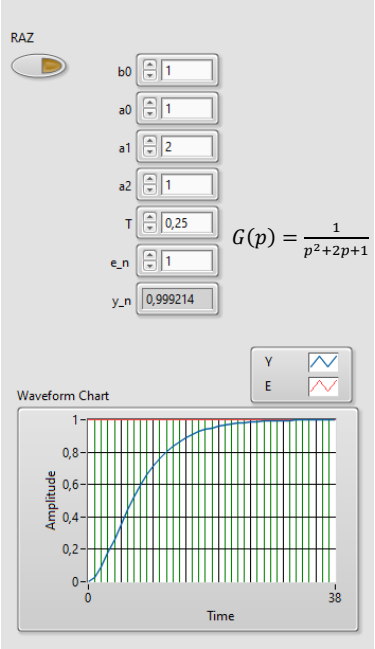
Voici le « back panel » :

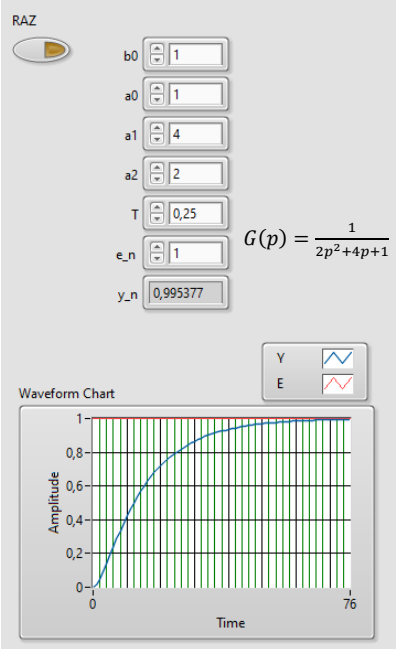


- Simulation de chaque cas :

Maintenant, on choisit une fonction de transfert pour chaque cas et on la simule, voici les résultats qui sont corrects :







III- Mise en œuvre du régulateur PID :

- Fonction de transfert en Z du régulateur et relation récurrente :

Le correcteur PID échantillonné et bloqué à l'aide d'un bloqueur zéro a comme fonction de transfert en z :

$$C(z)=\frac{z-1}{z}TZ\left(\frac{G(p)}{p}\right)=\frac{z-1}{z}TZ\left(\frac{K_p}{p}\left(1+\frac{1}{T_i p}+T_d p\right)\right)$$

- Action proportionnelle (en utilisant la transformé en z d'un échelon) :

$$C_p(z)=\frac{z-1}{z}TZ\left(\frac{K_p}{p}\right)=\frac{z-1}{z}K_p\frac{z}{z-1}=K_p$$

D'où la relation récurrente :

$$u_p[n]=K_p\varepsilon[n]$$

- Action intégrale (en utilisant la transformé en z d'une rampe) :

$$C_i(z)=\frac{z-1}{z}TZ\left(\frac{K_p}{T_i p^2}\right)=\frac{z-1}{z}K_p\frac{T}{T_i}\frac{z}{(z-1)^2}=\frac{K_p T}{T_i}\frac{1}{z-1}$$

D'où la relation récurrente :

$$u_i[n]=u_i[n-1]+\frac{K_p T}{Ti}\varepsilon[n]$$

- Action dérivée (en utilisant la transformé en z d'une impulsion) :

$$C_d(z) = \frac{z-1}{z} TZ(K_p T_d) = \frac{z-1}{z} K_p T_d$$

D'où la relation récurrente :

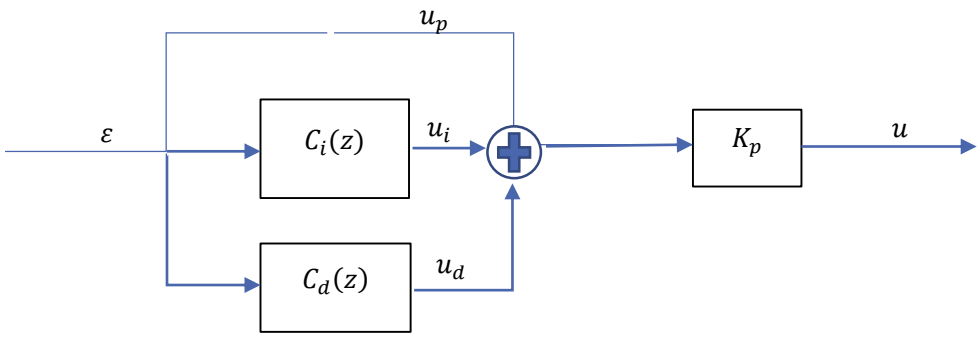
$$u_d[n] = K_p T_d(\varepsilon[n] - \varepsilon[n-1])$$

- Régulateur complet :

$$C(z) = C_p(z) + C_i(z) + C_d(z) = K_p\left(1 + \frac{T}{T_i}\frac{1}{z-1} + T_d\frac{z-1}{z}\right)$$

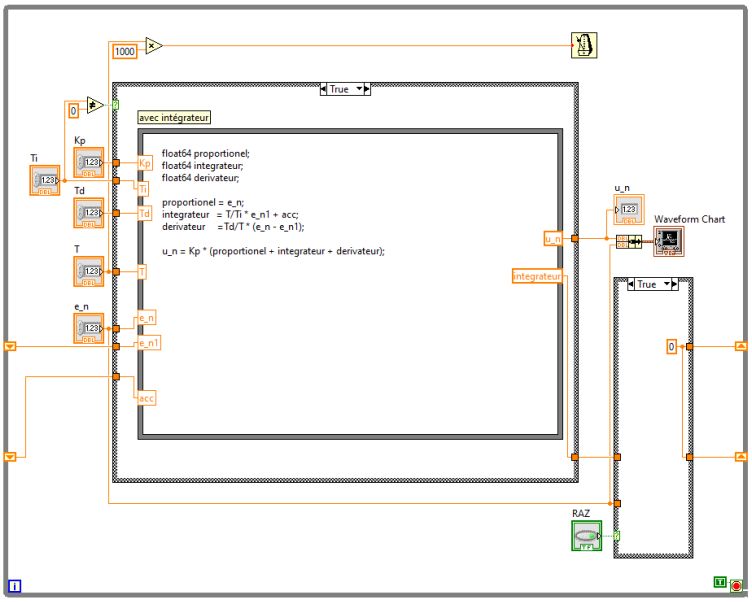
$$u[n] = u_p[n] + u_i[n] + u_d[n] = K_p\left(\varepsilon[n] + u_i[n-1] + \frac{T}{Ti}\varepsilon[n] + T_d(\varepsilon[n] - \varepsilon[n-1])\right)$$

Le schéma bloc du régulateur est le suivant :



- Implémentation sur LabVIEW du régulateur :

Comme dans le cas du système du deuxième ordre, on a une boucle cadencée par le pas d'échantillonnage, dans laquelle on a une structure « case » qui selon la constante T_i sélectionnera un correcteur PID ou PD, à l'intérieur de celle-ci, on trouve un « formula node » dans lequel on réalise les calculs de la relation récurrente, finalement, on utilise des registres de décalage pour accéder aux variables $u_i[n-1]$ et $\varepsilon[n-1]$. Voici le « back panel » :



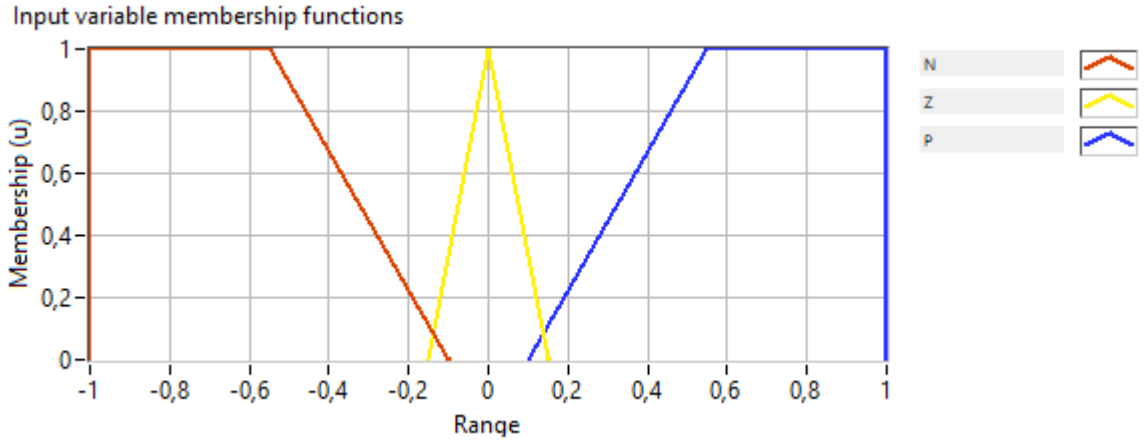
IV- Mise en œuvre du régulateur flou :

- Caractéristiques du régulateur flou :

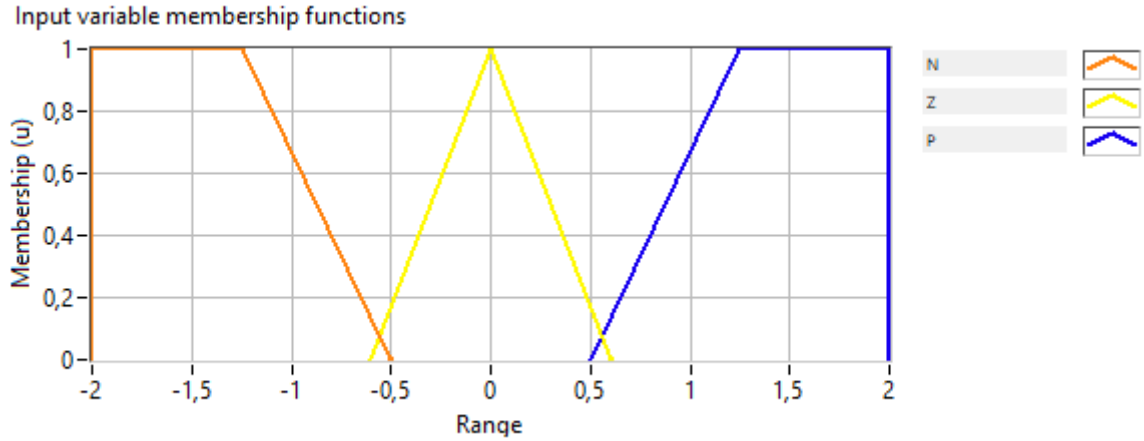
– Entrées :

On aura deux entrée l'erreur qu'on normalise par rapport à la consigne (on divise par la consigne) et sa dérivée (la différence entre deux échantillon consécutifs de l'erreur normalisé).

Donc, pour la variable linguistique « erreur », on a l'univers de discours = [-1, 1], cette variable appartient aux ensembles flous N (négative), Z (zéro) et P (positive), avec les fonctions d'appartenances suivantes :

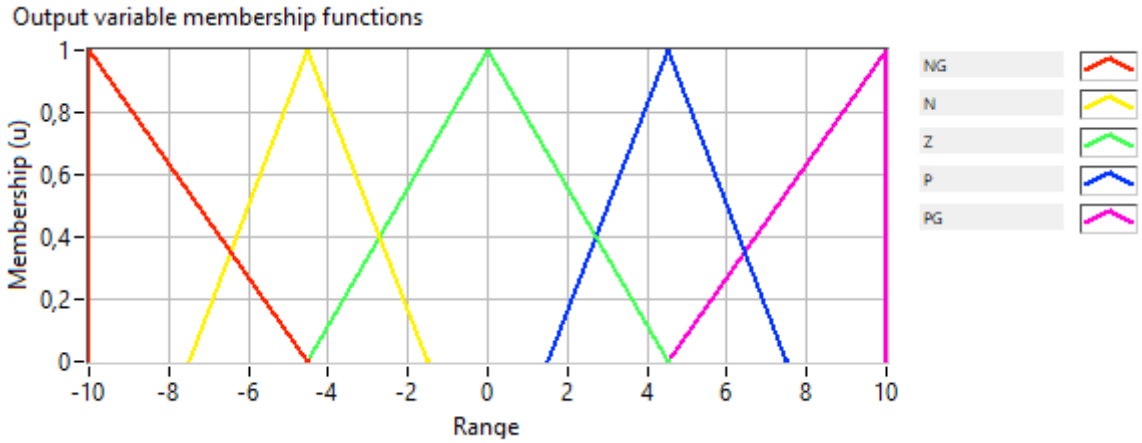


Pour la variable linguistique « dérivée de l'erreur », on a l'univers de discours = [-2, 2] (la plus grande différence est 2 = 1 – (-1)), cette variable appartient aux ensembles flous N (négative), Z (zéro) et P (positive), avec les fonctions d'appartenances suivantes :



– Sortie :

La sortie est la commande, donc la variable linguistique « commande », on a l'univers de discours = [-10, 10] (on a comparé avec la plage de sortie de la commande du correcteur PID et on a pris cette intervalle), cette variable appartient aux ensembles flous NG (négative grande), N (négative), Z (zéro), P (positive) et PG (positive grande), avec les fonctions d'appartenances suivantes :



La technique de défuzzification prise est le centre de gravité.

Pour les fonctions d'appartenance des entrées et de sortie, leurs allures ont été prise par essais et erreurs, normalement pour chaque système ces fonctions d'appartenance seront différentes.

– Base de règles :

On a les règles d'inférences suivantes selon la méthode d'inférence MIN/MAX :

$\Delta\varepsilon\backslash\varepsilon$	N	Z	P
N	PG	P	Z
Z	P	Z	N
P	Z	N	NG

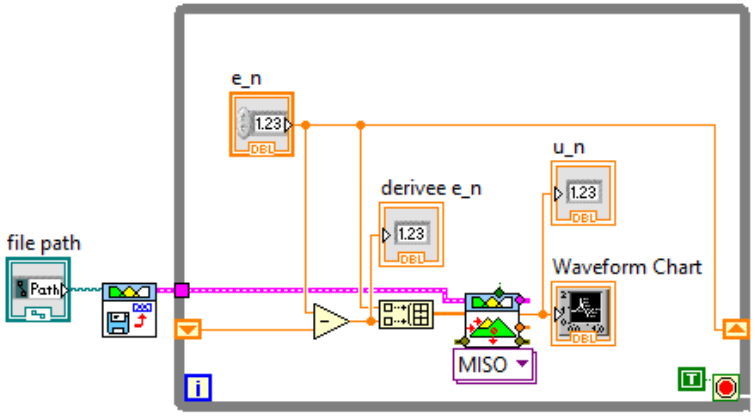
On explique un cas, les autres suivent le même raisonnement :

- Si ε est N ET $\Delta\varepsilon$ est N, alors u est PG :

Cela se traduit en : si la sortie est inférieure à l'entrée et elle est en train de diminuer, alors la commande doit être positive grande afin de remonter la sortie et diminuer l'erreur et rendre positive sa dérivée.

- Implémentation du régulateur flou sur LabVIEW :

L'implémentation du régulateur flou sur LabVIEW est directe, on utilise le « fuzzy system designer » pour concevoir le système, puis on l'introduit dans le VI. On charge le système flou, puis on alimente le contrôleur flou avec les entrées en utilisant un constructeur de tableaux et on reçoit la sortie, pour la dérivée de l'erreur, on utilise un registre de décalage afin de soustraire deux échantillon consécutifs, le tout est mis dans une boucle cadencée avec le pas d'échantillonnage. Voici le « back panel » :



V- Mise en œuvre de la boucle de régulation – montage final :

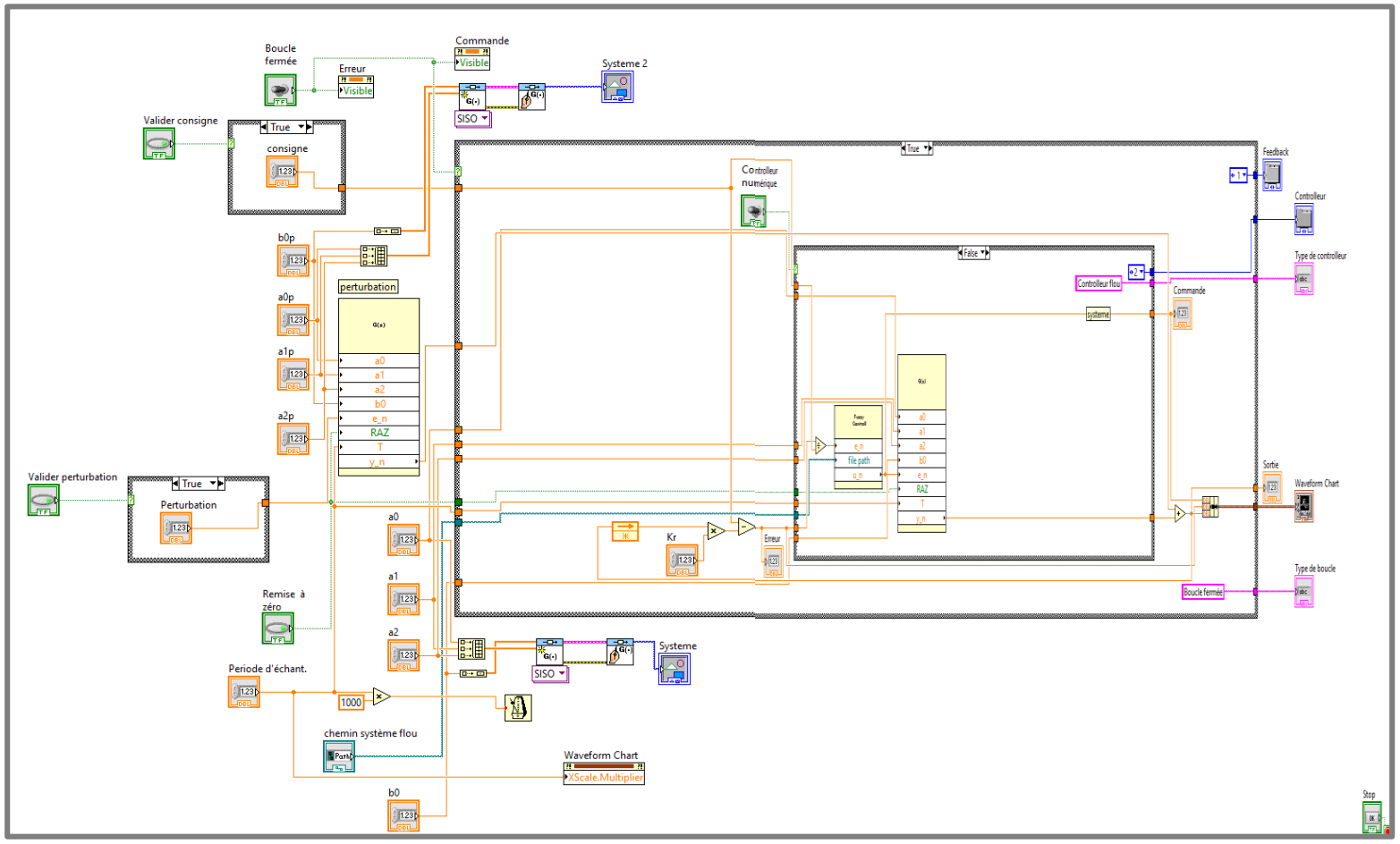
- Back panel :

Après avoir réalisé les sous-VI des contrôleurs et du système, on passe à l'implémentation de la boucle de régulation avec perturbation, choix du contrôleur, et choix de ou bien boucle ouverte ou boucle fermée.

On a une boucle cadencée par le pas d'échantillonnage, dans laquelle on lit les inputs de l'utilisateur et on a deux structures « case » imbriquées, la première est pour choisir le type de boucle, et la deuxième pour choisir le type de contrôleur, dans chaque cas, on a notre système connecté ou non avec le contrôleur, et un message affichant le type de boucle et le type de régulateur choisis.

Au dehors de la structure, on trouve la perturbation implémentée avec le sous-VI du système de deuxième ordre.

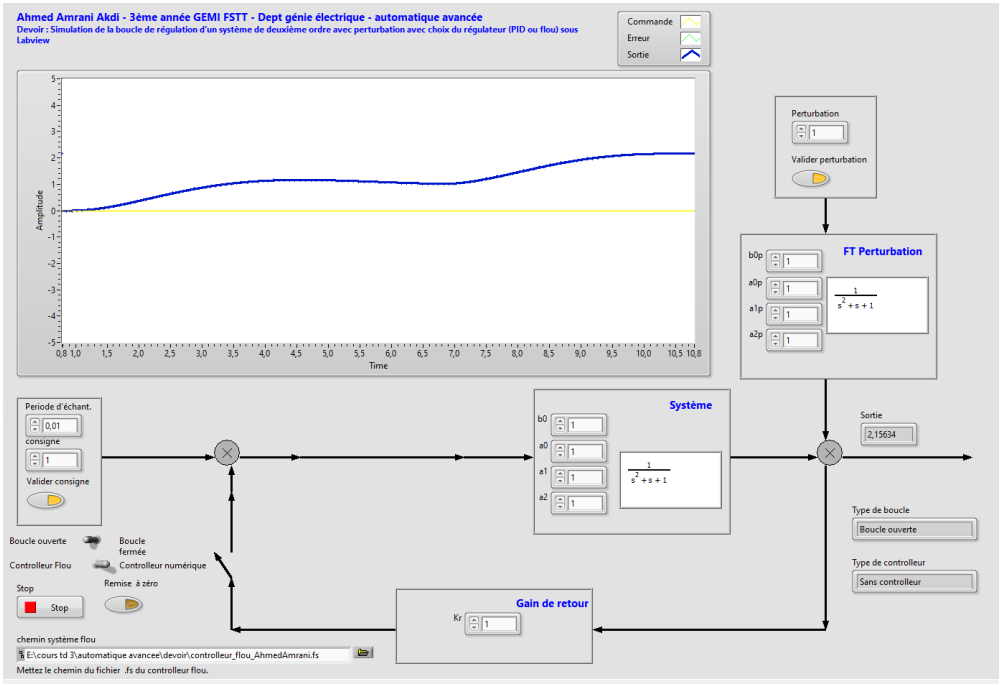
Finalement, on utilise des « property node » pour rendre visible ou invisibles les indicateurs qu'on nécessite dans chaque cas, pour changer le schéma de la boucle, on utilise des « tab control » transparents contrôlés par des constantes mises dans les structures « case », et pour ajuster le pas temporel du graphe avec le pas d'échantillonnage. Voici le « back panel » :



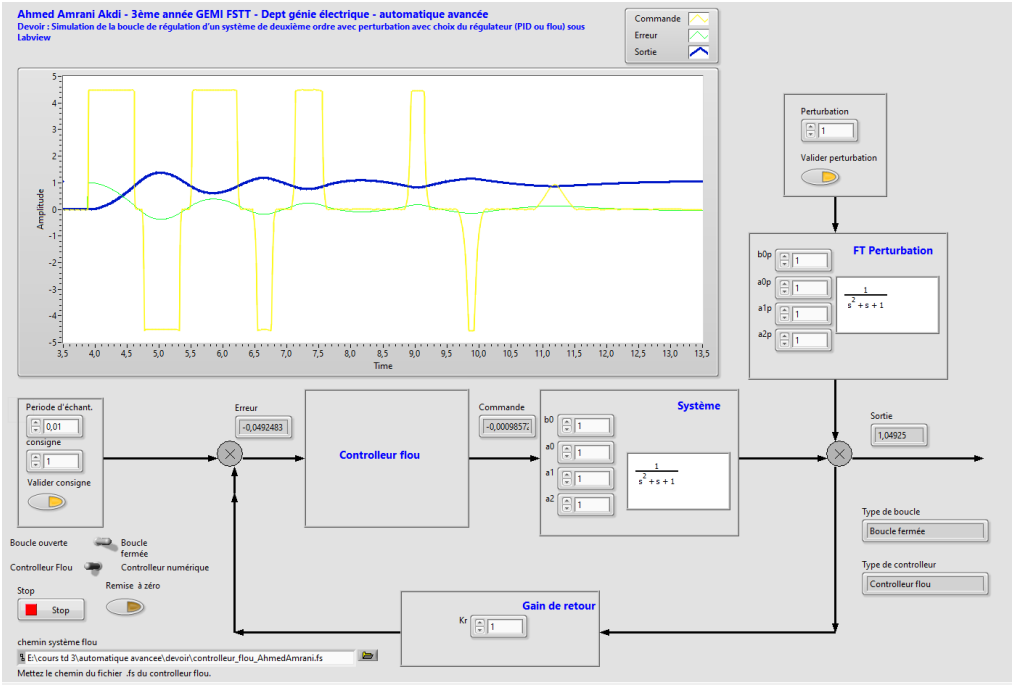
- Front panel

Pour les trois cas on simule le système $G(p) = \frac{1}{p^2+p+1}$ avec consigne échelon unitaire et avec une perturbation qui suit la même fonction que le système et qui s'ajoute après que le système atteint le régime permanent.

– Boucle ouverte :



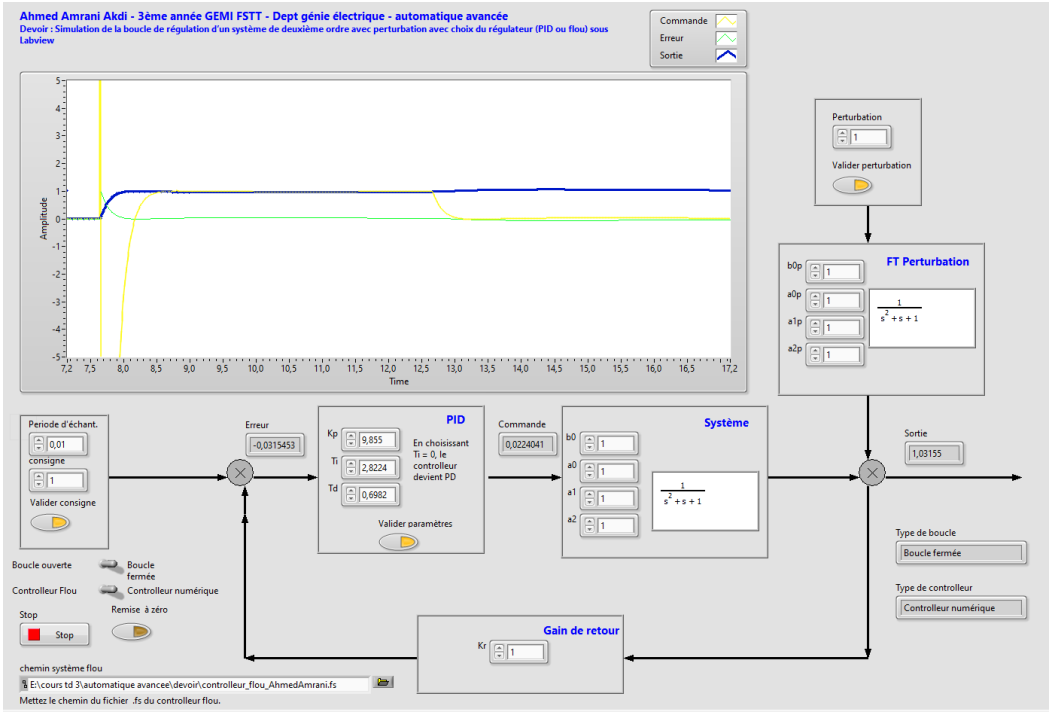
– Boucle fermé et régulateur flou :



Le système oscille au début, mais se stabilise et rejette la perturbation.

– Boucle fermé et régulateur PID :

Le réglage du régulateur a été réalisé à l'aide de Simulink, afin de comparer et de vérifier l'application. Voici donc, le résultat :



On voit que le régulateur rejette les perturbations.

VI- Annexe 1 – Code Matlab :

Voici le code Matlab utilisé pour vérifier les équations récurrentes trouvées. Il trace la réponse du système à un échelon de trois manière, en utilisant l'expression continue du système, en utilisant l'expression discrète trouvée à l'aide de la fonction de Matlab c2d (« convert to discret »), et en utilisant la relation récurrente trouvée.

```
T = 0.25;
N = 100;
b0 = 1;
a0 = 1;
a1 = 2;
a2 = 1;

% Trace réponse impulsionnelle de G(p) et son équivalent numérique G(z)
% avec période d'échantillonnage T pour N points.
% G(p) = b0/(a0 + a1*p + a2*p*p)
% G(z) = (z-1)/z * TZ(G(p)/p) %% E --->|BOZ|-->|G(p)|---> Y
% Ahmed Amrani Akdi

p1 = (-a1 + sqrt(a1^2 - 4*a0*a2))/2/a2;
p2 = (-a1 - sqrt(a1^2 - 4*a0*a2))/2/a2;
z1 = exp(T * p1);
z2 = exp(T * p2);

N_ = N + 2;
y_n = zeros(N_, 1);
e_n = ones(N_, 1);
e_n(1) = 0; e_n(2) = 0;
t_n = T * (1:N)';

% numérique
% pôle double
if p1 == p2
    % G(z) = (z * A + B)/(z - z1)^2
    A = (b0 * z1 + b0 * p1 * T * z1 + b0 - 2 * b0 * z1)/p1^2/a2;
    B = (-b0 * z1 - b0 * p1 * T * z1 + b0 * z1^2)/p1^2/a2;

    for k = 3:N_
        y_n(k) = A * e_n(k - 1) + B * e_n(k - 2) ...
            + 2 * z1 * y_n(k - 1) - z1^2 * y_n(k - 2);
    end
% deux pôles réels distincts ou bien deux pôles complexes conjugués
else
    % G(z) = A * (z * B + C)/(z * z - z * (z1 + z2) + z1 * z2)
    A = b0/p1/p2/a2;
    B = (-z2 * p1 + p2 * z1 - p2 + p1)/(p1 - p2);
    C = z1 * z2 + (p2 * z2 - p1 * z1)/(p1 - p2);

    for k = 3:N_
        y_n(k) = A * B * e_n(k - 1) + A * C * e_n(k - 2) ...
            + y_n(k - 1) * (z1 + z2) - y_n(k - 2) * z1 * z2;
    end
end
scatter(t_n, y_n(3:N_), 'b');

% en continue
z = b0;
p = [a2, a1, a0];
```

```
G_p = tf(z, p);
hold on;
step(G_p, 'g');

% fonction de conversion continu -> numérique

G_z = c2d(G_p, T, 'zoh'); % zoh = zero order hold = bloqueur d'ordre zéro
hold on;
step(G_z, 'r');
```

Voici un exemple d'éxecution: