Correcteurs Analogiques et Numériques

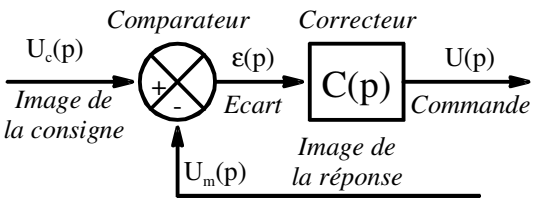
1- Correction d'un asservissement et exemples de correcteurs

1.1- Principe de la correction d'un asservissement

Le principe de tout asservissement ou régulation est de mesurer la réponse d'un système (grandeur physique asservie ou régulée) et de comparer cette mesure à la consigne à laquelle le système doit répondre. On ne compare pas forcément directement ces 2 grandeurs physiques de même type (même unité) mais 2 autres grandeurs qui sont des images de la consigne et de la réponse.

Ces 2 grandeurs physiques sont $u_c(t)$ l'image de la consigne (issue d'un adaptateur) et $u_m(t)$ image de la réponse (issue d'un capteur).

Quoiqu'il en soit ces deux images sont également (comme la consigne et la réponse) deux grandeurs de même type et donc de même unité.



La différence entre ces 2 grandeurs, issue du comparateur, est l'écart : $\varepsilon(t) = u_c(t) - u_m(t)$.

Ensuite cet écart $\varepsilon(t)$ est traité par le correcteur qui fournit une commande u(t) au système.

Deux principaux type de correction

La correction analogique se fait le plus souvent à l'aide d'Amplificateurs Linéaires Intégrés (ALI). Dans ce cas les images de la consigne et de la réponse ($u_c(t)$ et $u_m(t)$) sont des tensions en Volt. La commande (u(t)) est également une tension.

Dans certains cas (plus rares) ce peut également être des intensités.

La correction numérique se fait le plus souvent avec des microcontrôleurs qui par le traitement de nombres entiers (integer) ou décimaux (float) fournissent une commande qui est également un nombre entier (integer) ou décimal (float).

Dans certains cas la commande (comparateur + correcteur) peut recevoir des grandeurs analogiques (tension ou intensité). Dans ce cas les images de la consigne et de la réponse ($u_c(t)$ et $u_m(t)$) sont converties en entiers ou décimaux par un convertisseur analogique numérique : CAN.

De même, la commande peut être convertie en un signal analogique (souvent une tension) par un convertisseur numérique analogique : CNA.

1.2- Principaux types de correcteurs

Les principaux correcteurs utilisés sont les suivants :

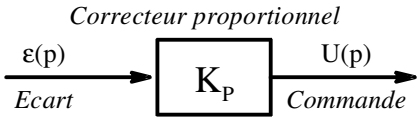
- ☞ Correcteur proportionnel : Très simple mais pas toujours très performant notamment pour la précision.
- ☞ Correcteur intégral : Très simple mais donnant un système assez souvent lent ou alors instable.
- ☞ Correcteur proportionnel intégral : Assez simple avec de bonnes performances mais parfois trop lent
- ☞ Correcteur à avance de phase : Permettant une commande rapide et stable mais pas toujours précise.
- ☞ Correcteur proportionnel intégral dérivé : Très performant : Commande rapide, précise et stable
- ☞ Correcteur à retard de phase : Plutôt très rare

Il est également possible d'améliorer les performances avec une boucle interne à l'asservissement. Comme par exemple une boucle de vitesse (tachymétrique) sur l'actionneur (Moteur à CC). On a alors deux corrections (souvent 2 correcteurs proportionnels et PI) ce qui donne une correction proche d'une correction PID.

2- Correcteur proportionnel

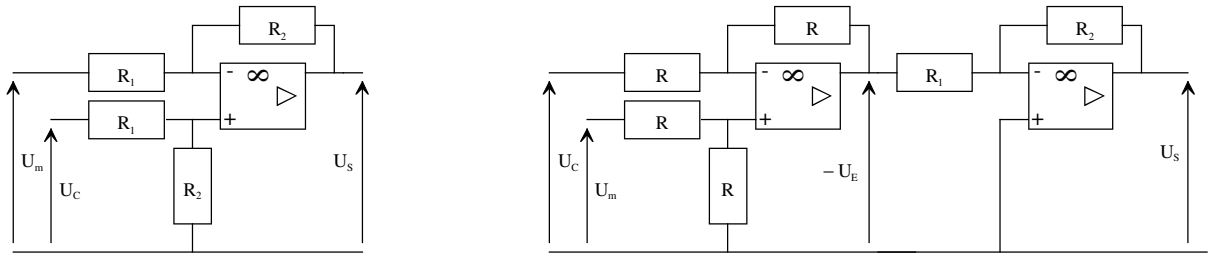
2.1- principe

La commande en sortie du correcteur est proportionnelle de gain K_P à l'écart en entrée. Sa fonction de transfert est donc :



2.2- Correction électronique analogique (Exemples)

Comparateur amplificateur Ou Comparateur + Proportionnel inverseur



$$u_S(t) = \frac{R_2}{R_1}.(u_C(t) - u_m(t)) \qquad C(p) = K_P = \frac{R_2}{R_1}$$

$$u_S(t) = \frac{R_2}{R_1}.(u_C(t) - u_m(t)) \qquad C(p) = K_P = \frac{R_2}{R_1}$$

2.3- Correction numérique

Pour donner l'algorithme de correction, on adopte les variables suivantes :

- Cons : Variable mémorisant l'image de la consigne $u_c(t)$ à la date t
- Mes : Variable mémorisant l'image de la réponse $u_m(t)$ à la date t
- Ec : Variable mémorisant l'écart $\varepsilon(t) = u_c(t) - u_m(t)$ à la date t
- ComB : Variable mémorisant la grandeur à la sortie du correcteur (Commande brute)
- Com : Variable donnant la commande au système. En général Com = ComB mais pas systématiquement par exemple cette commande peut être saturée (On a alors une variable de saturation) ou discrétisée (Com peut être un entier (integer) et ComB un décimal (float).

Boucle d'asservissement

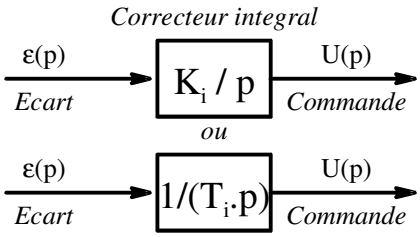
```
tant que Asservissement :          # Asservissement : booléen vrai si on asservi le système
    Cons – Mes → Ec                # On calcule l'écart
    Kp × Ec → ComB                 # On calcule la commande brute
    si ComB > Sat  alors  Sat → Com        # Sat : variable mémorisant la saturation
    sinon si ComB < – Sat  alors  – Sat → Com   # Idem
    sinon ComB → Com               # Si Com est un entier et ComB un float on a : int(ComB) → Com
    temporisation (Te)             # Te est le temps d'échantillonnage pour une durée de boucle nulle.
fin tant que
```

3- Correcteur intégral

3.1- principe

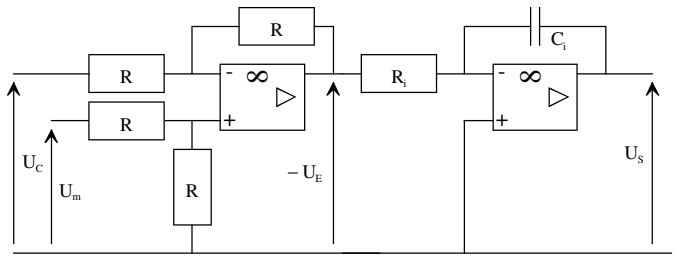
La commande en sortie du correcteur est proportionnelle de gain K_I à l'intégrale dans le temps de l'écart en entrée.

Sa fonction de transfert est donc :



3.2- Correction électronique analogique (Exemple)

Comparateur
+
Intégrateur
Inverseur



$$U_S(t) = \frac{1}{R_i.C_i} \cdot \int U_E(t).dt$$

$$C(p) = \frac{K_i}{p}$$

$$K_i = \frac{1}{T_i} = \frac{1}{R_i.C_i}$$

3.5- Correction numérique

Domaine de Laplace : ⇒

Domaine temporel :

Après discrétisation : ⇒

Pour donner l'algorithme de correction, on adopte les variables suivantes :

- Date & Date_prec : Variables mémorisant les dates t & t−dt
- dt : Variable mémorisant la durée d'échantillonage : dt = Date − Date_prec
- Cons : Variable mémorisant l'image de la consigne $u_c(t)$ à la date t
- Mes : Variable mémorisant l'image de la réponse $u_m(t)$ à la date t
- Ec : Variable mémorisant l'écart $\varepsilon(t) = u_c(t) - u_m(t)$ à la date t
- ComB & ComB_Prec : Variables mémorisant la grandeur à la sortie du correcteur aux dates t & t−dt
- Com : Variable donnant la commande au système. En général Com = ComB mais pas systématiquement par exemple cette commande peut être saturée (On a alors une variable de saturation) ou discrétisée (Com peut être un entier (integer) et ComB un décimal (float).

Boucle d'asservissement

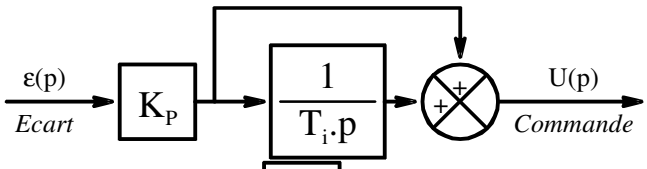
```
0 , 0 → ComB , ComB_Prec
Horloge → Date                  # La date est donnée par l'horloge du microprocesseur
tant que Asservissement :       # Asservissement : booléen vrai si on asservi le système
    Date → Date_Prec            # L'ancienne date (t) est mémorisée dans la variable Date_prec (t−dt)
    Horloge → Date              # La date est à nouveau donnée par l'horloge du micropocesseur
    Date − Date_prec → dt       # On calcul la durée d'échantillonage (peut être différente de Te)
    Cons − Mes → Ec             # On calcule l'écart
    ComB → ComB_prec            # On mémorise la commande brute à la date t−dt qui était ComB à t−dt
                                # On calcule la nouvelle commande brute
    si ComB > Sat  alors  Sat → ComB        # Si saturation inutile d'intégrer davantage
    sinon si ComB < − Sat  alors  − Sat → ComB   # Si saturation inutile d'intégrer davantage
    sinon ComB → Com            # Si Com est un entier et ComB un float on a : int(ComB) → Com
    temporisation (Te)          # Te est le temps d'échantillonnage pour une durée de boucle nulle.
fin tant que
0 , 0 → ComB , ComB_Prec        # A la fin de l'asservissement on réinitialise
Horloge → Date                  # La date est donnée par l'horloge du micropocesseur
```

4- Correcteur proportionnel intégral

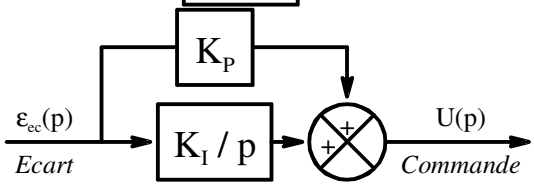
4.1- principe

La commande en sortie du correcteur est en partie proportionnelle à l'écart et en partie proportionnelle à l'intégrale dans le temps de l'écart

Cas 1 (Correction analogique)

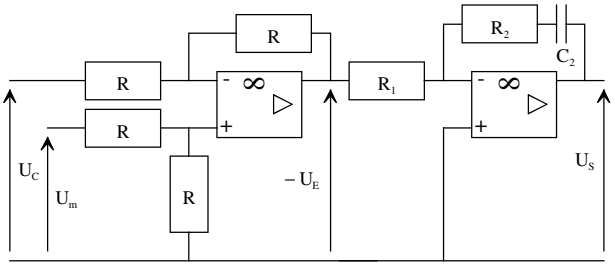


Cas 2 (Correction numérique)



4.2- Correction électronique analogique (Exemple)

Comparateur
+
Proportionnel
Intégral
Inverseur



$$u_S(t) = \frac{R_2}{R_1}.\left(u_E(t) + \frac{1}{R_2.C_2}\int u_E(t).dt\right)$$

$$C(p) = \frac{K.(1 + T_i.p)}{T_i.p}$$

$$K = \frac{R_2}{R_1} \qquad T_i = R_2.C_2$$

4.3- Correction numérique

Domaine de Laplace :

⇒

Si $K_p = K_i.T_i$ ⇒

Domaine temporel :

Après discrétisation :

⇒

Pour donner l'algorithme de correction, on adopte les variables suivantes :

- Date & Date_prec : Variables mémorisant les dates t & t–dt
- dt : Variable mémorisant la durée d'échantillonage : dt = Date – Date_prec
- Cons : Variable mémorisant l'image de la consigne $u_c(t)$ à la date t
- Mes : Variable mémorisant l'image de la réponse $u_m(t)$ à la date t
- Ec & Ec_Prec : Variables mémorisant les écart $\varepsilon(t) = u_c(t) - u_m(t)$ aux dates t & t–dt
- ComB & ComB_Prec : Variables mémorisant la grandeur à la sortie du correcteur aux dates t & t–dt
- Com : Variable donnant la commande au système. En général Com = ComB mais pas systématiquement par exemple cette commande peut être saturée (On a alors une variable de saturation) ou discrétisée (Com peut être un entier (integer) et ComB un décimal (float).

Boucle d'asservissement

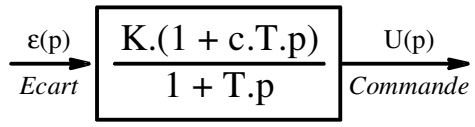
```
0 , 0 → ComB , ComB_Prec        # Initialisation des commandes brutes
0 , 0 → Ec , Ec_prec            # Initialisation des variables écarts
Horloge → Date                  # La date est donnée par l'horloge du microprocesseur
tant que Asservissement :       # Asservissement : booléen vrai si on asservi le système
    Date → Date_Prec            # L'ancienne date (t) est mémorisée dans la variable Date_prec (t–dt)
    Horloge → Date              # La date est à nouveau donnée par l'horloge du micropocesseur
    Date – Date_prec → dt       # On calcul la durée d'échantillonage (peut être différente de Te)
    Ec → Ec_Prec                # L'ancien écat est mémorisé dans la variable Ec_prec
    Cons – Mes → Ec             # On calcule le nouvel écart
    ComB → ComB_prec            # On mémorise la commande brute à la date t–dt qui était ComB à t–dt
                                # On calcule la nouvelle commande brute

    si ComB > Sat  alors  Sat → ComB          # Si saturation inutile d'intégrer davantage
    sinon si ComB < – Sat  alors  – Sat → ComB # Si saturation inutile d'intégrer davantage
    ComB → Com                  # Si Com est un entier et ComB un float on a : int(ComB) → Com
    temporisation (Te)          # Te est le temps d'échantillonnage pour une durée de boucle nulle.
fin tant que
0 , 0 → ComB , ComB_Prec        # A la fin de l'asservissement on réinitialise
0 , 0 → Ec , Ec_prec            # Initialisation des variables écarts
Horloge → Date                  # La date est donnée par l'horloge du micropocesseur
```

5- Correcteur à avance de phase

5.1- principe

Ce correcteur a une fonction de transfert du premier ordre généralisé.

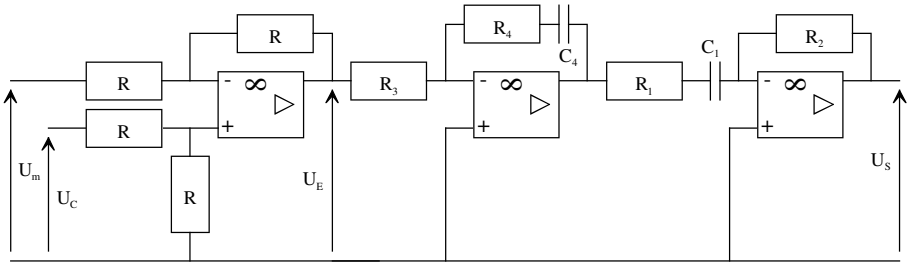
Correcteur à avance de phase



5.2- Correction électronique analogique (Exemple)

Sa conception analogique se compose comme le produit d'un circuit Proportionnel Intégral (PI) et d'un Proportionnel Dérivé Filtré (PDF) : $C(p) = \frac{K_a.(1 + T_a.p)}{p} . \frac{K_b.p}{1 + T_b.p}$

Comparateur
+
Proportionnel Intégral Inverseur
+
Proportionnel Dérivé Filtré Inverseur



$$C(p) = \frac{1 + R_4.C_4.p}{R_3.C_4.p} . \frac{R_2.C_1.p}{1 + R_1.C_1.p} = \frac{\frac{R_2.C_1}{R_3.C_4} . \left(1 + \frac{R_4.C_4}{R_1.C_1} . R_1.C_1 .p\right)}{1 + R_1.C_1 .p}$$

$$\mathbf{K = \frac{R_2.C_1}{R_3.C_4}} \qquad \mathbf{T = R_1.C_1} \qquad c = \mathbf{\frac{R_4.C_4}{R_1.C_1}}$$

5.3- Correction numérique

Domaine de Laplace :

⇒

Domaine temporel :

Après discrétisation :

⇒

⇒

Pour donner l'algorithme de correction, on adopte les variables suivantes :

- Date & Date_prec : Variables mémorisant les dates t & t−dt
- dt : Variable mémorisant la durée d'échantillonage : dt = Date − Date_prec
- Cons : Variable mémorisant l'image de la consigne $u_c(t)$ à la date t
- Mes : Variable mémorisant l'image de la réponse $u_m(t)$ à la date t
- Ec & Ec_Prec : Variables mémorisant les écart $\varepsilon(t) = u_c(t) - u_m(t)$ aux dates t & t−dt
- ComB & ComB_Prec : Variables mémorisant la grandeur à la sortie du correcteur aux dates t & t−dt
- Com : Variable donnant la commande au système. En général Com = ComB mais pas systématiquement par exemple cette commande peut être saturée (On a alors une variable de saturation) ou discrétisée (Com peut être un entier (integer) et ComB un décimal (float).

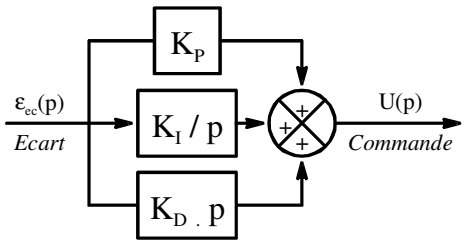
Boucle d'asservissement

```
0 , 0 → ComB , ComB_Prec        # Initialisation des commandes brutes
0 , 0 → Ec , Ec_prec            # Initialisation des variables écarts
Horloge → Date                  # La date est donnée par l'horloge du microprocesseur
tant que Asservissement :       # Asservissement : booléen vrai si on asservi le système
    Date → Date_Prec        # L'ancienne date (t) est mémorisée dans la variable Date_prec (t−dt)
    Horloge → Date          # La date est à nouveau donnée par l'horloge du micropocesseur
    Date − Date_prec → dt       # On calcul la durée d'échantillonage (peut être différente de Te)
    Ec → Ec_Prec            # L'ancien écat est mémorisé dans la variable Ec_prec
    Cons − Mes → Ec         # On calcule le nouvel écart
    ComB → ComB_prec        # On mémorise la commande brute à la date t−dt qui était ComB à t−dt
                            # On calcule la nouvelle commande brute

    si ComB > Sat  alors  Sat → Com             # Si saturation la commande est la saturation
    sinon si ComB < − Sat  alors  − Sat → Com   # Si saturation la commande est la saturation
    sinonComB → Com         # Si Com est un entier et ComB un float on a : int(ComB) → Com
    temporisation (Te)      # Te est le temps d'échantillonnage pour une durée de boucle nulle.
fin tant que
0 , 0 → ComB , ComB_Prec        # A la fin de l'asservissement on réinitialise
0 , 0 → Ec , Ec_prec            # Initialisation des variables écarts
Horloge → Date                  # La date est donnée par l'horloge du microprocesseur
```

6- Correcteur proportionnel intégral dérivé (PID)

6.1- principe

La commande en sortie du correcteur est en partie proportionnelle à l'écart, en partie à l'intégrale dans le temps de l'écart et à la dérivée temporelle de cet écart.



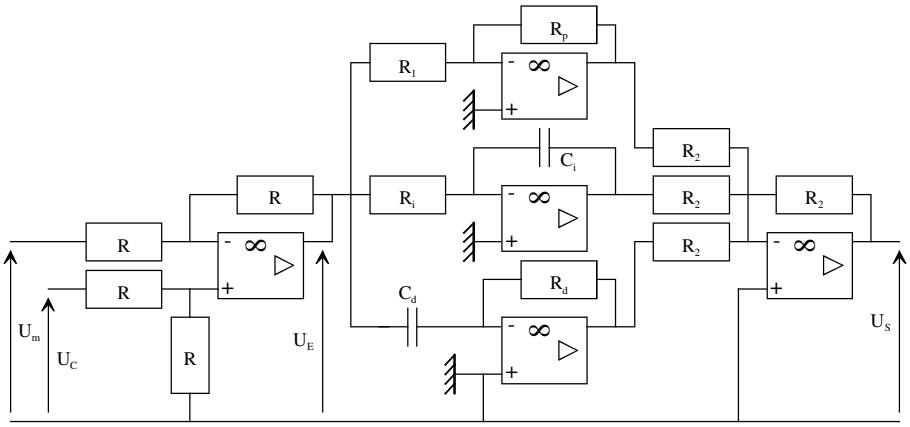
$$\mathbf{C(p) = \frac{K.\left(1 + \frac{2.\xi}{\omega_0}.p + \frac{p^2}{\omega_0^2}\right)}{p}} \qquad \textbf{Avec :}$$

Cas particulier : $K_P = 2.\sqrt{K_I.K_D}$ Dans ce cas : $\xi = 1$ En posant : $T = \sqrt{\frac{K_D}{K_I}}$ On a alors :

Avec :

6.2- Correction électronique analogique (Exemple)

Comparateur
+
[Inverseur
+
Intégrateur Inverseur
+
Dérivateur Inverseur] en //
+
Sommateur Amplificateur Inverseur



$$U_S(t) = \frac{R_p}{R_1}.U_E(t) + \frac{1}{R_i.C_i}\int U_E(t).dt + R_d.C_d.\frac{d\,U_E(t)}{dt}$$

$$U_S(p) = \left(\frac{R_p}{R_1} + \frac{1}{R_i.C_i}.\frac{1}{p} + R_d.C_d.p\right).U_E(p)$$

$$\mathbf{C(p) = K_P + \frac{K_I}{p} + K_D.p}$$

Avec : $\mathbf{K_P = \frac{R_p}{R_1}}$ $\mathbf{K_I = \frac{1}{R_i.C_i}}$ $\mathbf{K_D = R_d.C_d}$

6.3- Correction numérique

Domaine de Laplace : $\mathbf{C(p) = \frac{U(p)}{\varepsilon(p)} = \frac{K_i.\left(1 + \frac{K_P}{K_i}.p + \frac{K_D}{K_i}.p^2\right)}{p}}$

$$\Rightarrow \quad \mathbf{p.U(p) = K_i.\varepsilon(p) + K_P.p.\varepsilon(p) + K_D.p^2.\varepsilon(p)}$$

Domaine temporel : $\mathbf{\frac{d\,u(t)}{dt} = K_i.\varepsilon(t) + K_P.\frac{d\,\varepsilon(t)}{dt} + K_D.\frac{d^2\varepsilon(t)}{dt^2} = K_i.\varepsilon(t) + K_P.\dot{\varepsilon}(t) + K_D.\frac{d\,\dot{\varepsilon}(t)}{dt}}$

Soit Après discrétisation :

$$\frac{\mathbf{u(t) - u(t-dt)}}{\mathbf{dt}} = \mathbf{K_i.\varepsilon(t) + K_p.\dot{\varepsilon}(t) + K_D.}\ \frac{\mathbf{\dot{\varepsilon}(t) - \dot{\varepsilon}(t-dt)}}{\mathbf{dt}}$$

$$\Rightarrow \quad \mathbf{u(t) = K_i.\varepsilon(t).dt + u(t-dt) + K_p.\dot{\varepsilon}(t).dt + K_D.(\dot{\varepsilon}(t) - \dot{\varepsilon}(t-dt))}$$

Pour donner l'algorithme de correction, on adopte les variables suivantes :

- ☞ Date & Date_prec : Variables mémorisant les dates t & t–dt
- ☞ dt : Variable mémorisant la durée d'échantillonage : dt = Date – Date_prec
- ☞ Cons : Variable mémorisant l'image de la consigne $u_c(t)$ à la date t
- ☞ Mes : Variable mémorisant l'image de la réponse $u_m(t)$ à la date t
- ☞ Ec & Ec_Prec : Variables mémorisant les écarts $\varepsilon(t) = u_c(t) - u_m(t)$ aux dates t & t–dt
- ☞ VEc & VEc_Prec : Variables mémorisant les variations des écarts aux dates t & t–dt
- ☞ ComB & ComB_Prec : Variables mémorisant la grandeur à la sortie du correcteur aux dates t & t–dt
- ☞ Com : Variable donnant la commande au système. En général Com = ComB mais pas systématiquement par exemple cette commande peut être saturée (On a alors une variable de saturation) ou discrétisée (Com peut être un entier (integer) et ComB un décimal (float).

Boucle d'asservissement

```
0 , 0 → ComB , ComB_Prec        # Initialisation des commandes brutes
0 , 0 → Ec , Ec_prec            # Initialisation des variables écarts
0 , 0 → VEc , VEc_prec          # Initialisation des variables variations des écarts
Horloge → Date                  # La date est donnée par l'horloge du microprocesseur
tant que Asservissement :       # Asservissement : booléen vrai si on asservi le système
    Date → Date_Prec            # L'ancienne date (t) est mémorisée dans la variable Date_prec (t–dt)
    Horloge → Date              # La date est à nouveau donnée par l'horloge du micropocesseur
    Date – Date_prec → dt       # On calcul la durée d'échantillonage (peut être différente de Te)
    Ec → Ec_Prec                # L'ancien écart est mémorisé dans la variable Ec_prec
    Cons – Mes → Ec             # On calcule le nouvel écart
    VEc → VEc_Prec              # L'ancienne variation d'écart est mémorisé dans la variable VEc_Prec
    (Ec – Ec_Prec)/dt → VEc     # On calcule la nouvelle variation de l'écart
    ComB → ComB_prec            # On mémorise la commande brute à la date t–dt qui était ComB à t–dt
                                # On calcule la nouvelle commande brute
    Ki×Ec×dt + ComB_Prec + Kp×VEc×dt + KD×(VEc – VEc_Prec) → ComB
                                # Si saturation inutile d'intégrer davantage
    si ComB > Sat  alors  ComB_Prec + Kp×VEc×dt + KD×(VEc – VEc_Prec) → ComB
                                # Si saturation inutile d'intégrer davantage
    sinon si ComB < – Sat  alors  ComB_Prec + Kp×VEc×dt + KD×(VEc – VEc_Prec) → ComB
    si ComB > Sat  alors  Sat → Com          # Si saturation la commande est la saturation
    sinon si ComB < – Sat  alors  – Sat → Com    # Si saturation la commande est la saturation
    sinonComB → Com             # Si Com est un entier et ComB un float on a : int(ComB) → Com
    temporisation (Te)          # Te est le temps d'échantillonnage pour une durée de boucle nulle.
fin tant que
0 , 0 → ComB , ComB_Prec        # A la fin de l'asservissement on réinitialise
0 , 0 → Ec , Ec_prec            # Initialisation des variables écarts
Horloge → Date                  # La date est donnée par l'horloge du microprocesseur
```