

# Master 1, Conceptions Formelles

## Projet du module ALTARICA

### Synthèse (assistée) d'un contrôleur du niveau d'une cuve

Alain Griffault

Jeudi 1 mars 2018

## 1 Cahier des charges

Le système que l'on souhaite concevoir est composé :

- d'un réservoir contenant **toujours** suffisamment d'eau pour alimenter l'exploitation,
- d'une cuve,
- de deux canalisations parfaites amont reliant le réservoir à la cuve, et permettant d'amener l'eau à la cuve,
- d'une canalisation parfaite aval permettant de vider l'eau de la cuve,
- chaque canalisation est équipée d'une vanne commandable, afin de réguler l'alimentation et la vidange de la cuve,
- d'un contrôleur.

### 1.1 Détails techniques

#### 1.1.1 La vanne

Les vannes sont toutes de même type, elles possèdent trois niveaux de débits correspondant à trois diamètres d'ouverture : 0 correspond à la vanne fermée, 1 au diamètre intermédiaire et 2 à la vanne complètement ouverte. Les vannes sont commandables par les deux instructions `inc` et `dec` qui respectivement augmente et diminue l'ouverture. Malheureusement, la vanne est sujet à défaillance sur sollicitation, auquel cas le système de commande devient inopérant, la vanne est désormais pour toujours avec la même ouverture.

#### 1.1.2 La Cuve

Elle est munie de *nbSensors* capteurs (au moins quatre) situés à *nbSensors* hauteurs qui permettent de délimiter *nbSensors* + 1 zones. La zone 0 est comprise entre le niveau 0 et le niveau du capteur le plus bas ; la zone 1 est comprise entre ce premier capteur et le second, et ainsi de suite.

Elle possède en amont un orifice pour la remplir limité à un débit de 4, et en aval un orifice pour la vider limité à un débit de 2.

#### 1.1.3 Le contrôleur

Il commande les vannes avec les objectifs suivants ordonnés par importance :

1. Le système ne doit pas se bloquer, et le niveau de la cuve ne doit jamais atteindre les zones 0 ou *nbSensors*.
2. Le débit de la vanne aval doit être le plus important possible.

On fera également l'hypothèse que les commandes ne prennent pas de temps, et qu'entre deux pannes et/ou cycle *temporel*, le contrôleur à toujours le temps de donner au moins un ordre. Réciproquement, on fera l'hypothèse que le système à toujours le temps de réagir entre deux commandes.

#### 1.1.4 Les débits

Les règles suivantes résument l'évolution du niveau de l'eau dans la cuve :

- Si ( $amont > aval$ ) alors au temps suivant, le niveau aura augmenté d'une unité.
- Si ( $amont < aval$ ) alors au temps suivant, le niveau aura baissé d'une unité.
- Si ( $amont = aval = 0$ ) alors au temps suivant, le niveau n'aura pas changé.
- Si ( $amont = aval > 0$ ) alors au temps suivant, le niveau pourra :
  - avoir augmenté d'une unité,
  - avoir baissé d'une unité,
  - être resté le même.

## 2 L'étude

### 2.1 Rappel méthodologique

Comme indiqué en cours, le calcul par point fixe du contrôleur est exact, mais l'opération de projection effectuée ensuite peut perdre de l'information et générer un contrôleur qui n'est pas satisfaisant. Plus précisément, le contrôleur ALTARICA généré :

- ne garanti pas la non accessibilité des *Situations Redoutées*.
- ne garanti pas l'absence de *nouvelles situations de blocages*.

Dans le cas où il existe toujours *des situations de blocages ou redoutées*, vous pouvez au choix :

1. Corriger manuellement le contrôleur calculé (sans doute très difficile).
2. Itérer le processus du calcul du contrôleur jusqu'à stabilisation du résultat obtenu.
  - Si le contrôleur obtenu est sans blocage et sans situation redoutée, il est alors correct.
  - Si le contrôleur obtenu contient toujours des blocages ou des situations redoutées, c'est que le contrôleur initial n'est pas assez performant, mais rien de garanti que l'on soit capable de fournir ce premier contrôleur suffisamment performant.

**Remarque :** Pour vos calculs, vous pouvez utiliser au choix les commandes :

- `altarica-studio xxx.alt xxx.spe`
- `arc -b xxx.alt xxx.spe`
- `make` pour utiliser le fichier GNUmakefile fourni.

### 2.2 Le travail à réaliser

Avant de calculer les contrôleurs, vous devez répondre aux questions suivantes.

1. Expliquez le rôle de la constante `nbFailures` et de la contrainte, présente dans le composant `System`,  $nbFailures \geq (V[0].fail + V[1].fail + V[2].fail)$ .
2. Expliquez le rôle du composant `ValveVirtual` et de son utilisation dans le composant `CtrlVV`, afin de remplacer le composant `Ctrl` utilisé en travaux dirigés.

L'étude consiste à étudier le système suivant deux paramètres :

1. `nbFailures` : une constante qui est une borne pour le nombre de vannes pouvant tomber en panne.
2. Le contrôleur initial qui peut être soit `Ctrl`, soit `CtrlVV`.

Pour chacun des huit systèmes étudiés, vous devez décrire votre méthodologie pour calculer les différents contrôleurs et répondre aux questions suivantes :

1. Est-il possible de contrôler en évitant les blocages et les situations critiques ?
2. Si oui, donnez quelques caractéristiques de ce contrôleur, si non, expliquez pourquoi.
3. Est-il possible de contrôler en optimisant le débit aval et en évitant les blocages et les situations critiques ?
4. Si oui, donnez quelques caractéristiques de ce contrôleur, si non, expliquez pourquoi.

### 3 Conditions générales

- Vous pouvez utiliser les fichiers sources de l'archive `FD-2017-2018-M1-CC-sujet.tgz`.
- **Groupes de trois étudiants maximum acceptés.**
- Le rapport est constitué de l'archive `FD-2017-2018-M1-CC-rapport.tgz` généré par la commande `make`. Il sera envoyé par mail à `Alain.Griffault@labri.fr`
- Date limite de remise du travail : **vendredi 16 mars 2018.**