Master 1, Conceptions Formelles
Projet du module AltaRica
Synthèse (assistée) d'un contrôleur du niveau d'une cuve

Emile ROLLEY Gilles SOUTON

Chapter 1

Le sujet

1.1 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.

1.1.1 Détails techniques

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.

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.

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.

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.

1.2 L'étude

1.2.1 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 précisemment, 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 ou 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 ne garanti que l'on soit capable de fournir ce premier contrôleur suffisemment 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.

1.2.2 Le travail a réaliser

L'étude consiste à étudier le système suivant trois 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`.
3. Une éventuelle optimisation pour améliorer le débit aval.

Les questions auxquelles vous devez répondre sont dans le fichier `fichier rapport.tex`. Elles correspondent aux interrogations 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.

Vous écrirez vos réponses dans ce même fichier.

Chapter 2

Le rapport

Le rapport est sur 20 points.

2.1 Processus

2.1.1 Rôle du fichier `GNUmakefile` (1.5 points)

- lignes 1 à 87 : déclaration des variables globales nécessaires pour l'exécution des règles.
- règle `all` : exécute les règles `sources` et `tank.time` et compile les cibles
  - `sujet.pdf`, `FD-2021-2022-M1-CC-sujet.tgz`,
  - `rapport-<user>.pdf`, `FD-2021-2022-M1-CC-rapport-<user>.tgz`,
  - `corrige.pdf` et `FD-2021-2022-M1-CC-corrige.tgz`.
- règle `sources` : exécute `make all` pour les sous-répertoires `Alt` et `Spec`.
- règle `tank.time` : pour chaque controleur `Ctrl` et `CtrlVV`, ainsi que pour chaque nombre de défaillances, est créé un fichier `tank.alt`. Ce fichier est l'aggrégation de l'ensembles des fichiers AltaRica du répertoire `Alt`. TODO: Elle termine en exécutant la commande `make` dans tous les sous-répertoires : `Res Controleurs Graphs LaTeX`.

2.1.2 Rôle de la constante `nbFailures` et de l'assertion associée (0.5 point)

La constante `nbFailures` corresponds au nombre de maximum de vannes pouvant tombées en pannes simultanément (`Valve.stucked == 1`). Cette condition est assurée par l'assertion :

```
# ligne 154 tank.alt
  nbFailures >= (V[0].stucked + V[1].stucked + V[2].stucked);
```

Par exemple, si les vannes 0 et 1 tombent en pannes :

$$V[0].stucked + V[1].stucked + V[2].stucked = 1 + 1 + 0 = 2$$

2.2 Résultats avec le contrôleur initial `Ctrl`

2.2.1 Calcul d'un contrôleur

Avec 0 défaillance (0.5 point)

Interprétation des résultats

Avec 1 défaillance (0.5 point)

Interprétation des résultats

Avec 2 défaillances (0.5 point)

Interprétation des résultats

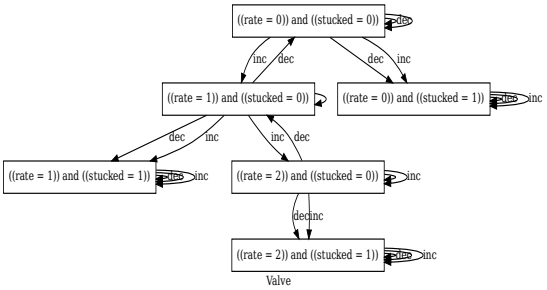
Avec 3 défaillances (0.5 point)

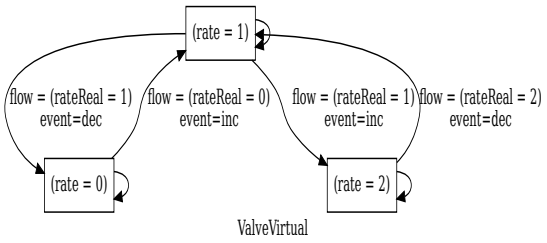
Interprétation des résultats

2.2.2 Bilan avec le contrôleur initial (1 point)

2.3 Construction d'un contrôleur initial plus performant

2.3.1 Rôle du composant `ValveVirtual`(2 points)





2.3.2 Rôle du composant `CtrlVV` (4 points)

2.4 Résultats avec le contrôleur `CtrlVV`

2.4.1 Calcul d'un contrôleur

Avec 0 défaillance (0.5 point)

Interprétation des résultats

Avec 1 défaillance (0.5 point)

Interprétation des résultats

Avec 2 défaillances (0.5 point)

Interprétation des résultats

Avec 3 défaillances (0.5 point)

Interprétation des résultats

2.4.2 Bilan avec le contrôleur CtrlVV (1 point)

2.5 Une première optimisation des contrôleurs pour améliorer le débit aval

2.5.1 Une optimisation basée sur les priorités (1 point)

```
event
  /*
   * les priorites dependent des actions sur la vanne aval
   * inc > nop > dec
```

```
   */
  /* nnn, nni, nnd, nin, nii, nid, ndn, ndi, ndd, inn, ini, ind, iin, iii, iid, idn, idi, idd,
  /* {iii, nii, ini} > {nni}; */
  /* {nni} > {dni, ndi, nnn}; */
  /* {dni, ndi, nnn, ddd} > {dnn, ndn, ddn, dnd, ndd, ddd}; */
  /* {ddi, dii, dni, idi, iii, ini, ndi, nii, nni} > */
  /*   {ddn, din, dnn, idn, iin, inn, ndn, nin, nnn}; */
  /* {ddn, din, dnn, idn, iin, inn, ndn, nin, nnn} > */
  /*   {ddd, did, dnd, idd, iid, ind, ndd, nid, nnd}; */
edon
```

L'optimisation mise en place défini un ordre de priorité sur les différentes actions sur les vannes. En particulier, sur la vanne aval : $inc > nop > dec$. Cela forcera donc à choisir les actions maximisant le débit de la vanne aval, en effet, l'action incrémentant le débit est prioritaire par rapport aux deux autres et l'action de ne pas modifier le débit est prioritaire à celle le diminuant.

2.5.2 Calcul des contrôleurs optimisés avec `CtrlVV`

Avec 0 défaillance

box

Avec 1 défaillance

box

Avec 2 défaillances

box

Avec 3 défaillances

box

2.5.3 Bilan avec la première optimisation du contrôleur CtrlVV (1 point)

Sans défaillance, tous les objectifs énnoncés pour le contrôleur sont respectés :

- $SR = 0$, assure que le system ne peut pas se bloquer ($deadlock = 0$) et que le niveau de la cuve ne dépasse jamais les niveau critiques ($NC = 0$),
- $out2 > out1 > out0$, démontre que le débit de la vanne aval est maximisé, c'est accentué par le fait que le débit n'est décrementé qu'une seule fois ($dec21 = 1 \wedge dec10 = 0$). (TODO: est-ce suffisant pour assurer que le débit aval est maximisé ?)

Avec une défaillance, tous l'objectif prioritaire énnoncé pour le contrôleur est respecté car $SR = 0$, cependant, pour l'optimisation du débit aval : TODO: comprendre CCoupGagnant.

A partir de 2 défaillances, le système ne se bloque pas et la niveau de la cuve ne dépasse pas les niveaux critiques ($SR = 0$), mais cela à un coup car le système ne possède plus que deux états ($any\_s = 2$).

2.6 Une deuxième optimisation (3 points)

Il est possible d'obtenir de meilleurs résultats que les précédents par au moins deux façons.

1. En utilisant un meilleur ordre pour les priorités entre événements.
2. En introduisant cet objectif dans le système de calcul de point fixe des actions du contrôleur.

Vous devez proposer une des deux optimisations conduisant à une solution dans laquelle le débit de la vanne aval est le moins souvent possible décrémenter, voire jamais. Pour cela, vous pouvez :

- Meilleur ordre sur les événements.
  1. Modifier le fichier `ControleursOpt/Optimisation.alt`.
  2. Faites `make`.

3. Quand vos résultats sont satisfaisants, notez les, puis copiez votre fichier dans `ControleursOpt/Optimisation-2.alt`.
4. Remettez le fichier `ControleursOpt/Optimisation.alt` d'origine.
5. Faites `make`.

- Meilleur système d'équations au point fixe.
  1. Modifier le fichier `Spec/System.spe`.
  2. Faites `make`.
  3. Quand vos résultats sont satisfaisants, notez les, puis copiez votre fichier dans `Spec/System-2.spe`.
  4. Remettez le fichier `Spec/System.spe` d'origine.
  5. Faites `make`.

Le nouvel ordre

```
event
  /*
   * les priorites dependent des actions sur la vanne aval
   * inc > nop > dec
   */
  /* {ddi, dii, dni, idi, iii, ini, ndi, nii, nni} > */
  /*   {ddn, din, dnn, idn, iin, inn, ndn, nin, nnn}; */
  /* {ddn, din, dnn, idn, iin, inn, ndn, nin, nnn} > */
  /*   {ddd, did, dnd, idd, iid, ind, ndd, nid, nnd}; */
edon
```

Le nouveau système d'équations

```
with SystemNbPannesFNomDuControleur do
  deadlock := any_s - src(any_t - self_epsilon);
  notResettable := any_s -coreach(initial,any_t);
  /* output */
  out0 := any_s & [V[2].rate=0];
  out1 := any_s & [V[2].rate=1];
  out2 := any_s & [V[2].rate=2];
  dec21 := any_t & label V[2].dec & rsrc(out2);
  dec10 := any_t & label V[2].dec & rsrc(out1);
  /* Niveaux critiques et Situations redoutees */
  NC := any_s & [T.level=0 | T.level=nbSensors];
  SR := deadlock | NC;
  EPerdu := any_s - SR;
  CGagne := any_s - SR;
  /* systeme d'equation au point fixe
  * CGagnant = CGagne & src(CCoupGagnant)
  * CCoupGagnant = CCoup & rtgt(EPerdant)
  * EPerdant = EPerdu & (src(ECoupPerdant) - src(ECoupNonPerdant))
  * ECoupPerdant = ECoup & rtgt(CGgagnant)
  * ECoupNonPerdant = ECoup & rtgt(any_s - CGagnant)
  */
  CCoup := any_t & label cmd;
  ECoup := any_t & label env;
  CCoupGagnant -= CCoup & rtgt(EPerdu & (src(ECoup & rtgt(CGagne & src(CCoupGagnant))) - src(EC
  /* Controller projection */
  /* Syntax and semantic of the "project" command
  * param 1 : les configurations projetees
  * param 2 : les transitions projetees
  *  - arg1 : projection existentielle
  *  - arg2 : projection universelle
  * param 3 : nom du noeud AltaRica genere
```

```
  * param 4 : simplification ou non des expressions booleennes
  * param 5 (optionel) : noeud de la hierarchie servant a la projection
  */
  /* Universal projection is use */
  project(any_s, (empty_t,CCoupGagnant), 'CtrlInitialNbPannesFNumIter', true,C)
       > 'Controleurs/CtrlInitialNbPannesFNumIter.alt';
  /* record results */
  show(any_s, any_t, deadlock, NC, SR , out0, out1, out2, dec21, dec10, CCoupGagnant) > 'Res/$N
done
```

2.7 Conclusion sur la synthèse de contrôleur (1 point)