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

Nom1 Nom2 Nom3

Chapitre 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 de 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

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`, $nbFailures >= (V[0].fail + V[1].fail + V[2].fail)$.
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.

Chapitre 2

Le rapport

2.1 Rôle de la constante `nbFailures` (2 points)

La constante nbFailures est utilisée pour limiter le nombre de configurations atteignables. Elle correspond au nombre de valves qui peuvent être en panne en même temps.

$nbFailures >= (V[0].fail + V[1].fail + V[2].fail)$ correspond au fait que le nombre de valve qui tombent en panne doit toujours être inférieur à la valeur de la constante nbFailure, car cette ligne se trouve dans la section *assert* du code.

2.2 Résultats avec le contrôleur initial `Ctrl`

2.2.1 Calcul d'un contrôleur

Avec 0 défaillance (1 point)

```
/*
 * Properties for node : System0FCtrl
 * # state properties : 7
 *
 * any_s = 247
 * deadlock = 0
 * NC = 86
 * SR = 86
 * out0 = 80
 * out1 = 83
 * out2 = 84
 *
 * # trans properties : 4
 *
 * any_t = 3472
 * CtrlCanControl = 27
 * CCoupGagnant = 1134
 * CCoupGagnantUtile = 712
 */

/*
 * Properties for node : System0FCtrl0F1I
 * # state properties : 7
 *
 * any_s = 94
 * deadlock = 0
 * NC = 0
```

```
 * SR = 0
 * out0 = 26
 * out1 = 34
 * out2 = 34
 *
 * # trans properties : 4
 *
 * any_t = 858
 * CtrlCanControl = 27
 * CCoupGagnant = 712
 * CCoupGagnantUtile = 712
 */
```

Interprétation des résultats : Contrôleur sans débit optimisé Le controlleur peut controller en évitant les blocages car $deadlock = 0$. Cependant, il ne peut pas assurer qu'on évitera les situations redoutés, car on peut voir que $SR = 86$, ce qui indique qu'il y a 86 états dans lesquels le controlleurs se trouvera dans une situation redoutée.

Interprétation des résultats : Contrôleur avec débit optimisé Lorsqu'on optimise le débit aval, il est possible de contrôller en évitant tout blocages ou toute situation critique. En effet, la variable $SR = 0$. On peut voir que dans ce contrôleur on a un total de 96 états (any_s) et 858 transitions (any_t). On peut aussi voir, via $CtrlCanControl$, qu'il y a 27 transisitions possibles différentes à partir de l'état initial du contrôleur.

Avec 1 défaillance (1 point)

```
/*
 * Properties for node : System1FCtrl
 * # state properties : 7
 *
 * any_s = 958
 * deadlock = 0
 * NC = 329
 * SR = 329
 * out0 = 300
 * out1 = 326
 * out2 = 332
 *
 * # trans properties : 4
 *
 * any_t = 19540
 * CtrlCanControl = 81
 * CCoupGagnant = 4950
 * CCoupGagnantUtile = 3043
 */

/*
 * Properties for node : System1FCtrl1F1I
 * # state properties : 7
 *
 * any_s = 508
 * deadlock = 96
 * NC = 69
 * SR = 96
 * out0 = 120
 * out1 = 188
```

```
 * out2 = 200
 *
 * # trans properties : 4
 *
 * any_t = 5173
 * CtrlCanControl = 81
 * CCoupGagnant = 2909
 * CCoupGagnantUtile = 2909
 */
```

Interprétation des résultats : contrôleur sans débit optimisé Avec 1 défaillance, on remarque qu'on peut pas contrôler en évitant les situations rédoutées car SR est égale à l'union de *deadlock* et de NC. Donc dans notre cas, on a SR qui partage les mêmes états avec NC.

Conclusion : le controleur peut contrôler car la variable CtrlCanControl est positive, mais on ne peut pas éviter de tomber dans des situations redoutées.

Interprétation des résultats : contrôleur avec débit optimisé Avec l'optimisation du débit 'aval', on remarque qu'on a moins d'états à gérer donc c'est plus facile pour le contrôleur de controler.

Cependant, on s'apperçoit dorénavant qu'on peut avoir des blocages à cause de la variable $deadlock = 96$, ce qui revient à dire qu'on ne peut pas contrôler sans éviter de tomber dans des situations redoutées car :

NC et *deadlock* partagent 69 états et donc on risque d'augmenter les chances de tomber dans des états bloquants.

Conclusion : Malgré le fait qu'on ait optimisé le débit d'aval, on a constaté qu'on ne pourrait pas contrôler sans éviter de tomber dans SR ou dans *deadlock*. Aussi, on peut bien voir que *CCoupGagnant* a beaucoup moins de transitions(2909).

Avec 2 défaillances (1 point)

```
/*
 * Properties for node : System2FCtrl
 * # state properties : 7
 *
 * any_s = 1627
 * deadlock = 0
 * NC = 551
 * SR = 551
 * out0 = 506
 * out1 = 553
 * out2 = 568
 *
 * # trans properties : 4
 *
 * any_t = 44608
 * CtrlCanControl = 117
 * CCoupGagnant = 7533
 * CCoupGagnantUtile = 4654
 */

/*
 * Properties for node : System2FCtrl2F1I
 * # state properties : 7
 *
 * any_s = 782
 * deadlock = 239
 * NC = 107
```

```
 * SR = 239
 * out0 = 192
 * out1 = 306
 * out2 = 284
 *
 * # trans properties : 4
 *
 * any_t = 6901
 * CtrlCanControl = 112
 * CCoupGagnant = 2868
 * CCoupGagnantUtile = 2868
 */
```

Interprétation des résultats : Contrôleur sans débit optimisé Ce controlleur peut aussi s'éxécuter en évitant les blocages car $deadlock = 0$. Cependant, il ne peut pas assurer qu'on évitera les situations redoutés. Ici, $SR = 551$. On a donc beaucoup d'états dans lesquels on ne souhaite pas aller. Ceci est dû au fait qu'on peut avoir deux valves défaillantes. Si une d'entre elle est celle de la sortie, on ne peut alors plus vider l'eau de la cuve et on atteint facilement les situations redoutées.

Interprétation des résultats : Contrôleur avec débit optimisé Lorsqu'on optimise le débit aval, on diminue le nombre de situations redoutées. Elle passent de 551 à 239. Cependant, on peut bloquer le système car $deadlock = 239$. On peut noter que tous les niveaux critiques sont des situations où l'on se bloque, sinon on aurait $SR > deadlock$ (par l'absurde, si il existe un état dans NC différent de tous ceux de $deadlock$ alors, $|NC \cup deadlock| > |deadlock|$)

Avec 3 défaillances (1 point)

```
/*
 * Properties for node : System3FCtrl
 * # state properties : 7
 *
 * any_s = 1832
 * deadlock = 0
 * NC = 617
 * SR = 617
 * out0 = 570
 * out1 = 622
 * out2 = 640
 *
 * # trans properties : 4
 *
 * any_t = 57696
 * CtrlCanControl = 125
 * CCoupGagnant = 7908
 * CCoupGagnantUtile = 5029
 */

/*
 * Properties for node : System3FCtrl3F1I
 * # state properties : 7
 *
 * any_s = 240
 * deadlock = 112
 * NC = 0
 * SR = 112
 * out0 = 48
```

```
 * out1 = 120
 * out2 = 72
 *
 * # trans properties : 4
 *
 * any_t = 1568
 * CtrlCanControl = 64
 * CCoupGagnant = 343
 * CCoupGagnantUtile = 343
 */
```

Interprétation des résultats : Contrôleur sans débit optimisé Dans ce contrôleur, il n'est pas possible de se trouver dans un état bloquant ($deadlock = 0$). Il est néanmoins possible de se trouver dans un des nombreux états qui sont des niveaux critiques. Avec $any\_s = 1832$ et $NC = 617$, on peut voir que près d'un tier des états sont des situations critiques.

Interprétation des résultats : Contrôleur avec débit optimisé A l'inverse du contrôleur sans optimisation de débit, il n'est pas possible de se retrouver dans des situations ou le niveau de l'eau atteint un niveau critique ($NC = 0$), mais il est possible que le systeme se bloque dans un des 112 états qui sont dans *deadlock*. Environ un état sur deux sont des états bloquants dans ce système. Il parait donc difficile de contrôler sans jamais se retouver dans un état appartenant à *deadlock*.

2.2.2 Calcul des contrôleurs optimisés (2 points)

2.3 Rôle des composants `ValveVirtual` et `CtrlVV` (4 points)

2.4 Résultats avec le contrôleur initial `CtrlVV`

2.4.1 Calcul d'un contrôleur

Avec 0 défaillance (1 point)

```
/*
 * Properties for node : System0FCtrlVV
 * # state properties : 7
 *
 * any_s = 247
 * deadlock = 0
 * NC = 86
 * SR = 86
 * out0 = 80
 * out1 = 83
 * out2 = 84
 *
 * # trans properties : 4
 *
 * any_t = 1863
 * CtrlCanControl = 8
 * CCoupGagnant = 548
 * CCoupGagnantUtile = 362
 */

/*
 * Properties for node : System0FCtrlVV0F1I
 * # state properties : 7
 *
```

```
 * any_s = 94
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 26
 * out1 = 34
 * out2 = 34
 *
 * # trans properties : 4
 *
 * any_t = 508
 * CtrlCanControl = 8
 * CCoupGagnant = 362
 * CCoupGagnantUtile = 362
 */
```

Interprétation des résultats

Avec 1 défaillance (1 point)

```
/*
 * Properties for node : System1FCtrlVV
 * # state properties : 7
 *
 * any_s = 1201
 * deadlock = 0
 * NC = 413
 * SR = 413
 * out0 = 350
 * out1 = 463
 * out2 = 388
 *
 * # trans properties : 4
 *
 * any_t = 8370
 * CtrlCanControl = 20
 * CCoupGagnant = 1866
 * CCoupGagnantUtile = 1186
 */

/*
 * Properties for node : System1FCtrlVV1F1I
 * # state properties : 7
 *
 * any_s = 316
 * deadlock = 16
 * NC = 0
 * SR = 16
 * out0 = 68
 * out1 = 138
 * out2 = 110
 *
 * # trans properties : 4
 *
 * any_t = 1076
 * CtrlCanControl = 17
 * CCoupGagnant = 546
```

```
 * CCoupGagnantUtile = 546
 */
```

Interprétation des résultats

Avec 2 défaillances (1 point)

```
/*
 * Properties for node : System2FCtrlVV
 * # state properties : 7
 *
 * any_s = 2398
 * deadlock = 0
 * NC = 812
 * SR = 812
 * out0 = 651
 * out1 = 1005
 * out2 = 742
 *
 * # trans properties : 4
 *
 * any_t = 15894
 * CtrlCanControl = 26
 * CCoupGagnant = 2360
 * CCoupGagnantUtile = 1529
 */

/*
 * Properties for node : System2FCtrlVV2F1I
 * # state properties : 7
 *
 * any_s = 274
 * deadlock = 70
 * NC = 0
 * SR = 70
 * out0 = 52
 * out1 = 130
 * out2 = 92
 *
 * # trans properties : 4
 *
 * any_t = 725
 * CtrlCanControl = 12
 * CCoupGagnant = 155
 * CCoupGagnantUtile = 99
 */
```

Interprétation des résultats

Avec 3 défaillances (1 point)

```
/*
 * Properties for node : System3FCtrlVV
 * # state properties : 7
 *
 * any_s = 2889
 * deadlock = 0
```

```
 * NC = 970
 * SR = 970
 * out0 = 764
 * out1 = 1253
 * out2 = 872
 *
 * # trans properties : 4
 *
 * any_t = 18776
 * CtrlCanControl = 27
 * CCoupGagnant = 2384
 * CCoupGagnantUtile = 1553
 */

/*
 * Properties for node : System3FCtrlVV3F1I
 * # state properties : 7
 *
 * any_s = 210
 * deadlock = 97
 * NC = 0
 * SR = 97
 * out0 = 36
 * out1 = 114
 * out2 = 60
 *
 * # trans properties : 4
 *
 * any_t = 565
 * CtrlCanControl = 8
 * CCoupGagnant = 27
 * CCoupGagnantUtile = 27
 */
```

Interprétation des résultats

2.4.2 Calcul des contrôleurs optimisés (2 points)

2.5 Conclusion (2 points)