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` :
  1. Pour chacun des contrôleurs (`Ctrl` et `CtrlVV`) et pour chaque nombre de défaillance (de 0 à 3), est créé un fichier `tank.alt` . Ce fichier est l'agrégation de l'ensemble des fichiers AltaRica du répertoire `Alt`.
  2. Les fichiers `test.alt` et `test.spe` sont créés en remplaçant les différentes *placeholders* des fichiers `tank.alt` et `test.spe` par leurs valeurs correspondantes – nombre de pannes, nom du contrôleur, etc...
  3. Les résultats de la commande `arc` sont ensuite écrits dans les fichiers `Res/*.res` correspondants. Cette procédure est répétée jusqu'à stabilisation des résultats (ou 5 fois maximum dans le cas échéant).
  4. Une fois stabilisé, les fichiers `test.alt` et `test.spe` sont recréés et exécutées mais en utilisant la version optimisée du contrôleur courant.
  5. Finalement, la commande `make` est exécutée 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 :

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

l'assertion serait donc vérifiée pour $nbFailures >= 2$.

2.2 Résultats avec le contrôleur initial `Ctrl`

2.2.1 Calcul d'un contrôleur

Avec 0 défaillance (0.5 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
 * dec21 = 351
 * dec10 = 342
 * CCoupGagnant = 1134
 */

/*
 * 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
 * dec21 = 102
 * dec10 = 68
 * CCoupGagnant = 712
 */

/*
 * Properties for node : System0FCtrl0F2I
 * # state properties : 7
 *
 * any_s = 94
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 26
 * out1 = 34
 * out2 = 34
 *
 * # trans properties : 4
 *
 * any_t = 858
```

```
 * dec21 = 102
 * dec10 = 68
 * CCoupGagnant = 712
 */
```

Interprétation des résultats Sans défaillance, le contrôleur initial ne vérifie pas l'objectif principal à savoir : garantir la non accessibilité des situations redoutées ($SR > 0$). Cependant, après projection, ces situations ne sont plus accessible ($SR = 0$). De plus le débit de la vanne aval semble avoir été maximisé ($out2 > out1 > out0$), ce qui valide le second objectif.

La stabilisation est atteinte dès la deuxième itération.

Avec 1 défaillance (0.5 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
 * dec21 = 2205
 * dec10 = 2139
 * CCoupGagnant = 4950
 */

/*
 * Properties for node : System1FCtrl1F1I
 * # state properties : 7
 *
 * any_s = 508
 * deadlock = 93
 * NC = 69
 * SR = 93
 * out0 = 120
 * out1 = 188
 * out2 = 200
 *
 * # trans properties : 4
 *
 * any_t = 5230
 * dec21 = 695
 * dec10 = 443
 * CCoupGagnant = 2941
 */

/*
 * Properties for node : System1FCtrl1F2I
 * # state properties : 7
 *
 * any_s = 508
 * deadlock = 96
 * NC = 69
 * SR = 96
```

```
 * out0 = 120
 * out1 = 188
 * out2 = 200
 *
 * # trans properties : 4
 *
 * any_t = 5161
 * dec21 = 695
 * dec10 = 439
 * CCoupGagnant = 2909
 */


/*
 * Properties for node : System1FCtrl1F3I
 * # state properties : 7
 *
 * any_s = 508
 * deadlock = 96
 * NC = 69
 * SR = 96
 * out0 = 120
 * out1 = 188
 * out2 = 200
 *
 * # trans properties : 4
 *
 * any_t = 5161
 * dec21 = 695
 * dec10 = 439
 * CCoupGagnant = 2909
 */
```

Interprétation des résultats Avec une défaillance, le contrôleur initial ne garantit pas le maintient du niveau de la cuve hors des zones critiques ($NC > 0$), cependant, le système ne peut pas se bloquer ($deadlock = 0$).

Après une première itération le contrôleur calculé à diminué le nombre d'états où le niveau de la cuve se retrouve en zone critique ($NC = 69$ au lieu de 93). En revanche le système peut se bloquer ($deadlock > 0$).

Les itérations suivantes ne modifient pas significativement les résultats.

Avec 2 défaillances (0.5 point)

Avec 2 défaillances, le nombre d'états où le niveau de la cuve se retrouve en zone critique augmente par rapport au système avec une seule défaillance ($SR = NC = 551$).

Après la première itération, le nombre d'états en situation redoutée diminue ($SR = 239$) mais introduit des situations bloquantes ($deadlock = 239$). La stabilisation est atteinte à la 4ème itération, cependant, le système ne respecte toujours par le premier objectif.

Avec 3 défaillances (0.5 point)

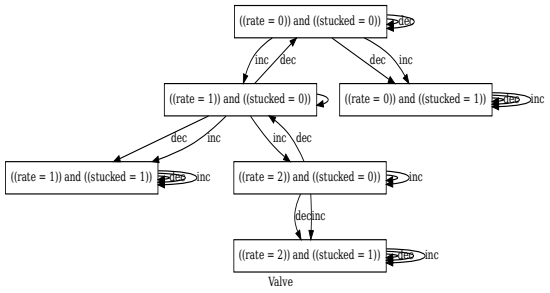
Avec 3 défaillances, le système ne respecte toujours par le premier objectifs car après stabilisation $SR = 27$, même si le nombre à diminué par rapport au système avec 2 défaillances, mais cela s'explique par le fait qu'il y a nécessairement moins d'états accessibles.

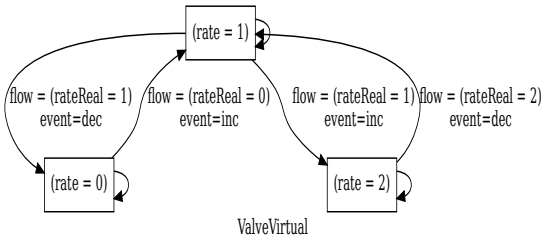
2.2.2 Bilan avec le contrôleur initial (1 point)

Le contrôleur initial ne permet de modéliser un système respectant les deux objectifs seulement si il n'y a pas de défaillances. Il ne permet donc pas de modéliser un système réaliste et robuste.

2.3 Construction d'un contrôleur initial plus performant

2.3.1 Rôle du composant `ValveVirtual`(2 points)





Le composant `ValveVirtual` est la représentation en droit d'une valve physique. Elle permet donc de simuler le fonctionnement et l'évolution d'une valve *parfaite*, c'est-à-dire, sans contrainte de l'environnement : elle ne peut donc pas tomber en panne.

Sa sémantique est la suivante :

- si les variables `rate` et `reateReal` coïncident, alors l'évènement `dec` (resp. `inc`) permet de décrémenter (resp. incrémenter) la valeur de `rate`.
- sinon, dès lors que les variables `rate` et `rateReal` se désynchronisent, cela signifie que la valve physique est tombée en panne : la valve virtuelle n'est alors plus utilisable.

2.3.2 Rôle du composant `CtrlVV` (4 points)

Le composant `CtrlVV` utilise 3 *instances* du composant `ValveVirtual` pour simuler le fonctionnement parfait (c'est à dire, sans défaillances possible) des trois `Valve` réelles du système. Pour ce faire, chacun des 27 évènements du contrôleur est synchronisé avec les évènements correspondants des `ValveVirtual`, ainsi, quand une commande du système est utilisée pour décrémenter ou incrémenter le débit d'une `Valve`, celui de la valve virtuelle du contrôleur `CtrlVV` correspondante est modifié de la même manière.

Comme expliqué dans la section précédente, si une vanne `Valve` du système tombe en panne alors les commandes `dec` et `inc` n'auront aucun effet sur cette dernière, alors que pour la vanne virtuelle `ValveVirtual` l'effet sera quand même appliqué ce qui causera une désynchronisation de la variable de flux `rateReal` et celle d'état `rate`.

Finalement, cela permet de *signifier* au système qu'une panne est survenue et ainsi permettre d'optimiser le contrôleur lors de la projection.

2.4 Résultats avec le contrôleur `CtrlVV`

2.4.1 Calcul d'un contrôleur

Avec 0 défaillance (0.5 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
```

```
 * dec21 = 213
 * dec10 = 208
 * CCoupGagnant = 548
 */

/*
 * 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
 * dec21 = 65
 * dec10 = 44
 * CCoupGagnant = 362
 */

/*
 * Properties for node : System0FCtrlVV0F2I
 * # state properties : 7
 *
 * any_s = 94
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 26
 * out1 = 34
 * out2 = 34
 *
 * # trans properties : 4
 *
 * any_t = 508
 * dec21 = 65
 * dec10 = 44
 * CCoupGagnant = 362
 */
```

Interprétation des résultats Sans défaillance, le contrôleur obtenu après stabilisation des résultats est correct car il assure les deux objectifs. A savoir, aucune situation redoutée est accessible car $SR = 0$ et le débit de la vanne aval semble avoir été maximisé car $out2 > out1 > out0$.

Avec 1 défaillance (0.5 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
 * dec21 = 910
 * dec10 = 888
 * CCoupGagnant = 1866
 */

/*
 * 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
 * dec21 = 92
 * dec10 = 64
 * CCoupGagnant = 546
 */

/*
 * Properties for node : System1FCtrlVV1F2I
 * # state properties : 7
 *
 * any_s = 232
 * deadlock = 3
 * NC = 0
 * SR = 3
 * out0 = 46
 * out1 = 104
 * out2 = 82
 *
 * # trans properties : 4
 *
 * any_t = 787
 * dec21 = 52
 * dec10 = 36
 * CCoupGagnant = 413
 */

/*
 * Properties for node : System1FCtrlVV1F3I
 * # state properties : 7
 *
 * any_s = 224
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 46
 * out1 = 104
 * out2 = 74
```

```
 *
 * # trans properties : 4
 *
 * any_t = 745
 * dec21 = 44
 * dec10 = 36
 * CCoupGagnant = 392
 */

/*
 * Properties for node : System1FCtrlVV1F4I
 * # state properties : 7
 *
 * any_s = 224
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 46
 * out1 = 104
 * out2 = 74
 *
 * # trans properties : 4
 *
 * any_t = 745
 * dec21 = 44
 * dec10 = 36
 * CCoupGagnant = 392
 */
```

Interprétation des résultats Avec une défaillance, le contrôleur obtenu après stabilisation est toujours correcte car $SR = 0$, en revanche il est difficile de se prononcer sur l'optimisation du débit de la vanne aval.

Avec 2 défaillances (0.5 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
 * dec21 = 1666
 * dec10 = 1625
 * CCoupGagnant = 2360
 */

/*
 * 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
 * dec21 = 30
 * dec10 = 27
 * CCoupGagnant = 155
 */

/*
 * Properties for node : System2FCtrlVV2F2I
 * # state properties : 7
 *
 * any_s = 2
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 2
 * out1 = 0
 * out2 = 0
 *
 * # trans properties : 4
 *
 * any_t = 4
 * dec21 = 0
 * dec10 = 0
 * CCoupGagnant = 1
 */

/*
 * Properties for node : System2FCtrlVV2F3I
 * # state properties : 7
 *
 * any_s = 2
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 2
 * out1 = 0
 * out2 = 0
 *
 * # trans properties : 4
 *
 * any_t = 4
 * dec21 = 0
 * dec10 = 0
 * CCoupGagnant = 1
 */
```

Interprétation des résultats Avec deux défaillance, le contrôleur obtenu après stabilisation est toujours correcte car $SR = 0$, en revanche le modèle ne comprends plus que deux états accessibles.

Avec 3 défaillances (0.5 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
 * dec21 = 1938
 * dec10 = 1890
 * CCoupGagnant = 2384
 */

/*
 * 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
 * dec21 = 26
 * dec10 = 16
 * CCoupGagnant = 27
 */

/*
 * Properties for node : System3FCtrlVV3F2I
 * # state properties : 7
 *
 * any_s = 2
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 2
 * out1 = 0
 * out2 = 0
 *
 * # trans properties : 4
 *
 * any_t = 4
 * dec21 = 0
 * dec10 = 0
 * CCoupGagnant = 1
 */

/*
 * Properties for node : System3FCtrlVV3F3I
 * # state properties : 7
 *
```

```
 * any_s = 2
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 2
 * out1 = 0
 * out2 = 0
 *
 * # trans properties : 4
 *
 * any_t = 4
 * dec21 = 0
 * dec10 = 0
 * CCoupGagnant = 1
 */
```

Interprétation des résultats Avec trois défaillance, comme pour deux défaillances, le contrôleur obtenu après stabilisation est toujours correcte car $SR = 0$ mais le modèle ne comprends plus que deux états accessibles.

2.4.2 Bilan avec le contrôleur CtrlVV (1 point)

Comme attendu, le contrôleur `CtrlVV` permet d'améliorer le calcul du contrôleur et prenant en compte les cas de pannes, ainsi après stabilisation peut importe le nombre de défaillance, les situations redoutées sont évitées. A partir, de deux défaillances et une fois le calcul du contrôleur stabilisé il ne reste plus que deux états possibles, effet, si deux vannes tombent en pannes il faudra nécessairement fermer la troisième pour ne pas rentrer en zone critiques.

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
   */
  {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

/*
 * Properties for node : System0FCtrlVV0F2I_Opt
 * # state properties : 7
 *
 * any_s = 49
 * deadlock = 0
 * NC = 0
```

```
 * SR = 0
 * out0 = 1
 * out1 = 14
 * out2 = 34
 *
 * # trans properties : 4
 *
 * any_t = 174
 * dec21 = 1
 * dec10 = 0
 * CCoupGagnant = 96
 */
```

Avec 1 défaillance

box

```
/*
 * Properties for node : System1FCtrlVV1F4I_Opt
 * # state properties : 7
 *
 * any_s = 191
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 21
 * out1 = 96
 * out2 = 74
 *
 * # trans properties : 4
 *
 * any_t = 580
 * dec21 = 16
 * dec10 = 14
 * CCoupGagnant = 277
 */
```

Avec 2 défaillances

box

```
/*
 * Properties for node : System2FCtrlVV2F3I_Opt
 * # state properties : 7
 *
 * any_s = 2
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 2
 * out1 = 0
 * out2 = 0
 *
 * # trans properties : 4
 *
 * any_t = 4
 * dec21 = 0
 * dec10 = 0
 * CCoupGagnant = 1
 */
```

Avec 3 défaillances

box

```
/*
 * Properties for node : System3FCtrlVV3F3I_Opt
 * # state properties : 7
 *
 * any_s = 2
 * deadlock = 0
 * NC = 0
 * SR = 0
 * out0 = 2
 * out1 = 0
 * out2 = 0
 *
 * # trans properties : 4
 *
 * any_t = 4
 * dec21 = 0
 * dec10 = 0
 * CCoupGagnant = 1
 */
```

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

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

- $SR = 0$, assure que le système ne peut pas se bloquer ($deadlock = 0$) et que le niveau de la cuve ne dépasse jamais les niveau critiques ($NC = 0$),
- $out2 > out1 > out0$, semble indiqué que le débit de la vanne aval est maximisé, c'est accentué par le fait que le débit n'est décrémenté qu'une seule fois ($dec21 = 1 \wedge dec10 = 0$).

Avec une défaillance, tous l'objectif prioritaire énoncé pour le contrôleur est respecté car $SR = 0$, cependant, il est plus difficile de savoir si le débit de la vanne aval est maximal.

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
```

Pour maximiser le débit de la vanne aval il faut que la somme des débits des vannes en amonts soit égal à celui de la vanne aval, pour éviter que le niveau de la cuve ne se retrouve en zone critique car la cuve se viderait plus vite qu'elle ne se remplierait. En particulier, il faut que les deux vannes en amonts aient un débit de 1.

Malheureusement, nous n'avons pas réussi définir un ordre sur les commandes permettant d'avoir une amélioration significative.

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)

Le contrôleur `CtrlVV` permet de s'assurer que le système ne se bloquera pas et ne pourra pas se retrouver dans une situation critique même si cela peut entrainer une perte de fonctionnalités. Au contraire du contrôleur `Ctrl` qui ne garantit pas le premier objectif en cas de panne.