hydra

hydraulics with QGIS

NT22 – Description du fichier de contrôle et de régulation

HYDRA SOFTWARE

www.hydra-software.net

contact@hydra-software.net

Version 31 : 10/01//2024

setec hydratec

Immeuble Central Seine
42-52 quai de la Rapée
75582 Paris Cedex 12
Email :
hydra@hydra.setec.fr
T : 01 82 51 64 02
F : 01 82 51 41 39

Directeur d'affaire : LPT
Responsable d'affaire : BLN
N°affaire : 01630796
Fichier : NT22_ description fichier de controle.doc

Version	Date	Etabli par	Vérifié par	Nb pages	Observations / Visa
1	01/08/2017	LPT			Notice mise à jour de la version V8
2	24/07/2017	LPT			Correction du tableau 3.1
3	10/10/2017	LPT			Création de la fonction #PID pour les pompes
4	06/11/2017	LPT			Implémentation commande Elseif et else Création de l'instruction Calculn Création de l'instruction Data
5	11/11/2017	LPT			Ajout de la commande Qc pour l'objet « constant inflow »
6	17/11/2017	LPT			Modification des données de méta commandes.
7	24/02/2018	LPT			Enrichissement des fonctionnalités : boucle DO, tableaux de chaine de caractères, accessibilité aux variables indicées d'un tableau .
8	26/05/2018	LPT			Regroupement des commandes CALCUL et CALCULN La commande SET est étendue à des variables indicées de tableau La commande SHOW est étendu à des tableaux et des variables indicées de tableaux
9	27/09/2018	LPT			Accès à une variable d'état d'un sous modèle différent du modèle en cours de traitement
10	25/11/2018	LPT			Accès à l'objet LCLDLAT (ex QDL1)
11	08/03/2019	LPT			Ajout du mot clé « TRACS » pour la commande de manœuvre rapide de certains actionneurs. Suppression dans la notice de la commande de déplacement DZ pour tous les actionneurs (cette commande n'a jamais été active dans le code)
12	28/04/2019	LPT			Ajout du mot clé SAVE
13	15/05/2019	LPT			Ajout de la variable DTLAG
14	11/11/2019	LPT			Implémentation manœuvre rapide pour les vannes S_gate et L_gate
15	31/01/2020	LPT			Pilotage d'une ouverture de brèche trapézoïdale à l'aide de l'objet LSTK
16	30/11/2020	LPT			Option EXTERNAL
17	01/05/2021	LPT			Ajout opérandes logiques – commande Séquence Commandes open et closed – désactivation commande dzs
18	09/06/2021	LPT			Rétablissement de la commande DZS
19	21/04/2022	LPT			Ajout des meta commandes $READ_EXT, $WRITE_EXT et des commandes WRITE et #GET_EXT
20	26/07/2022	LPT			Restructuration du module Dy avec : - Généralisation de l'utilisation des variables && et & - Création des sous programmes.

21	27/10/2022	LPT			Clarification notice sur l'utilisation des mots clés DT, DTLAG, TIME
22	15/11/2022	LPT			Enrichissement des déclarations de variables pour le module Dy. Création de l'instruction DATA SEQUENCE Amélioration de la gestion des sous programmes
23	15/02/2023	LPT			Ajout des instructions SELECTC…VALUEC… ENDSELECTC Ajout des fonctions #LEC_CDATE0 #DIFF_DATE #ADD_HR_TO_DATE
24	17/02/2023	LPT			Ajout de la fonction #INSTANCIATION
25	25/02/2023	LPT			Les instructions SELECTC…VALUEC… ENDSELECTC et DO…ENDDO peuvent désormais être imbriquées
26	03/04/2023	LPT			Enrichissement régulation de l'objet LPOMP
27	10/04/2023	LPT			Modification appel de sous-programme si l'argument est un tableau
28	22/05/2023	LPT			Implémentation du mode Set %var = SEQUENCE
29	20/09 :2023	LPT			Implémentation de la fonction DIM, nettoyage du code et sécurisation du traitement des sous programmes
30	08/11/2023	LPT			Implémentation de la méta commande !DEBUG et désactivation de la commande Debug ajout des commandes View_control Endview_control
31	10/01/2024	LPT			Création de la fonction #LEC_CDATE Création d'alias de type d'objet pour compatibilité AQDV

TABLE DES MATIERES

1 MOTIVATIONS 4

2 PRINCIPES 5

2.1 LES VARIABLES LOCALES 5
2.2 COMMANDES CONDITIONNELLES D'ACTIONNEURS 5
2.2.1 Actionneurs concernés 5
2.2.2 Conditions d'activation d'un actionneur 6
2.3 LES INSTRUCTIONS D'AFFECTATION ET DE CALCUL 7
2.4 LES INSTRUCTIONS CONDITIONNELLES 7
2.5 LES SOUS PROGRAMMES 7
2.6 LES METACOMMANDES 8
2.7 RENSEIGNEMENT DES PARAMETRAGES 8
2.8 EXECUTION D'UN ALGORITHME DE CONTROLE 8
2.9 SYNTHESE 8

3 DESCRIPTION DU FICHIER DE CONTROLE 10

3.1 INSTRUCTIONS DE COMMANDE D'UN ACTIONNEUR 10
3.1.1 Instructions conditionnelles 10
3.1.2 Instructions non conditionnelles 19
3.2 INSTRUCTIONS D'INITIALISATION OU D'AFFECTATION OU DE CALCUL 19
3.2.1 L'instruction DATA 19
3.2.2 L'instruction DATA SEQUENCE 20
3.2.3 L'instruction SET 21
3.2.4 L'instruction SAVE 26
3.2.5 L'instruction CALCUL ou CALCULN 26
3.2.6 Les noms de variables locales 29
3.3 LES INSTRUCTIONS DE RUPTURE DE SEQUENCE 29
3.3.1 Les instructions de fin et les commentaires 29
3.3.2 Les instructions conditionnelles IF ..THEN, ELSEIF..THEN, ELSE, ENDIF 29
3.3.3 L'instruction DO … ENDDO 30
3.3.4 Les instructions SELECTC…VALUEC…ENDSELECTC 31
3.4 LES INSTRUCTIONS DE SUIVI DES FICHIERS DE CONTROLE 33
3.4.1 Commande DEBUG 33
3.4.2 Commandes VIEW_COMMAND … ENDVIEW_COMMAND 33
3.4.3 Commande Show 34
3.4.4 La commande WRITE 34
3.5 ACTIVATION D'UN REGIME TRANSITOIRE RAPIDE 34
3.6 OPERATIONS SUR LES VARIABLES TABLEAUX 36
3.7 LES META COMMANDES 36
3.7.1 La méta commande $CASE 36
3.7.2 La méta commande $READ_EXT 37
3.7.3 La méta commande $WRITE_EXT 38
3.7.4 La méta commande $DEBUG 38

4 LES SOUS PROGRAMMES 40

4.1 PRINCIPES 40
4.1.1 Défintiion 40
4.1.2 Exemple d'agencement 40
4.1.3 Intérêt présenté par le sous-programme 41

4.1.4 Règles d'utilisation *41*
4.2 DESCRIPTION DES INSTRUCTIONS 42
4.2.1 Appel à un sous-programme *42*
4.2.2 Déclaration d'un sous-programme *42*
4.2.3 Déclaration de fin d'un sous-programme *43*
4.2.4 Déclaration de variables d'actionneurs dans une commande CALL *43*
4.3 EXEMPLE 43

5 EXEMPLES DE FICHIER DE CONTROLE 46

5.1 EXEMPLE 1 46
5.2 EXEMPLE 2 46
5.3 EXEMPLE 3 46
5.4 EXEMPLE 4 47
5.5 EXEMPLE 5 47
5.6 EXEMPLE 6 48
5.7 EXEMPLE 7 48
5.8 EXEMPLE 8 :UTILISATION DE L'INSTRUCTION DO… ENDDO 49
5.9 EXEMPLE 9 : MANIPULATION DE VARIABLES TABLEAUX 50

6 ANNEXE 1 : FONCTIONS DISPONIBLES DANS LA BIBLIOTHEQUE DU MODULE DE CONTROLE 51

6.1 MOT CLE : NTERP 51
6.2 MOT CLE : DATEVALJ 51
6.3 MOT CLE : MIN 52
6.4 MOT CLE : MAX 52
6.5 MOT CLE : MOY 52
6.6 MOT CLE : ABS 52
6.7 MOT CLE : LEC_ DATE0 53
6.8 MOT CLE : EXTRACT 53
6.9 MOT CLE: PID 53
6.10 MOT CLE : GET_EXT 54
6.11 MOT CLE : IRSTART 54
6.12 MOT CLE : LEC_CDATE0 55
6.13 MOT CLE : DIFF_DATE 55
6.14 MOT CLE : ADD_HR_TO_DATE 55
6.15 MOT CLE : INSTANCIATION 56
6.16 MOT CLE : DIM 57
6.17 MO CLE : LEC_CDATE 57

7 ANNEXE 2 : PARAMETRES MODIFIABLES POUR CHAQUE ACTIONNEUR 58

7.1 S_GATE 58
7.2 S_RWEIR 59
7.3 S_RGATE 60
7.4 L_PUMP_DERIV 62
7.5 L_WEIR 64
7.6 L_GATE 65
7.7 L_RGATE 66
7.8 CL_ZT 68
7.9 L_PUMP 69
7.10 CL_Q0 70
7.11 CASIER -STORAGE 71
7.12 LSTK 72
7.13 CL_DLAT 74

1 MOTIVATIONS

Les ouvrages hydrauliques ponctuels modélisés dans HYDRA sont paramétrés par défaut de façon à fonctionner en mode local. Les paramètres par défaut sont définis dans la boite de dialogue des différents objets disponibles dans l'interface.

La plupart des actionneurs sont fixes, d'autres sont mobiles et régulés par rapport à une consigne de déplacement, de niveau ou de débit.

Dans les situations réelles d'exploitation, ces actionneurs peuvent très bien être manœuvrés de façon différente de celle indiquée par la consigne par défaut. On distingue deux cas :

1. les états contraints : un actionneur est consigné en raison d'un événement extérieur tel qu'une mise en chômage, un incident sur le réseau ou un état de crue,
2. gestion coordonnée : on peut être amené à moduler la consigne d'un actionneur en fonction d'un algorithme de contrôle faisant intervenir tout un ensemble de paramètres d'état du système et d'actionneurs : la régulation locale n'est plus adaptée dans ce cas.

2 PRINCIPES

Le module de régulation est paramétré par un ou plusieurs fichiers de contrôle définis par l'utilisateur à l'aide d'un éditeur de texte.

Chaque fichier est structuré par un langage de programmation spécifique à Hydra qui permet d'implémenter des stratégies de contrôle complexes.

Les éléments de langage disponibles comprennent :

- les variables locales,
- les commandes conditionnelles d'actionneurs,
- les instructions d'affectation et de calcul,
- les instructions conditionnelles,
- les sous programmes,
- les méta commandes.

2.1 LES VARIABLES LOCALES

Ces variables sont de quatre types :

1. Les variables simples numériques, préfixées par le caractère « % »,
2. Les variables simples de type caractère, préfixées par le caractère « & »,
3. Les tableaux de valeurs numériques, préfixés par « %% »,
4. Les tableaux de chaine de caractères, préfixés par « && ».

Ces variables sont accessibles par toutes les instructions d'un même réseau. Les variables entre deux réseaux différents sont accessibles via l'instruction EXTERNAL.

2.2 COMMANDES CONDITIONNELLES D'ACTIONNEURS

2.2.1 Actionneurs concernés

Seul un nombre limité de d'actionneurs hydrauliques est concerné. Le tableau page suivante donne la liste de ces modules.

Mot clé	**Alias**
S_gate	**VA**
S_Rweir	**DE**
S_Rgate	**RGB**
L_pump_deriv	**QDP**
L_weir	**QMS**
L_gate	**QMV**
L_Rgate	**RGZ, RGQ**
CL_zt	**CLZT**
L_pump	**QMP**
CL_dlat	**QDL1**

Tableau 2-1 : Liste des actionneurs accessibles par le module de contrôle

Le préfixes « S », «L », « CL » désigne le type d'objet de modélisation : singularity, link ou condition à la limite.

La lettre « R » en tête du nom de l'élément précise que cet élément est déjà régulé par défaut via le paramétrage défini dans la fenêtre de saisie de l'élément.

Les alias peuvent être utilisés : ils assurent la compatibilité avec les mots clés définis dans les fichiers de contrôle tournant avec les versions antérieures d'HYDRA.

2.2.2 Conditions d'activation d'un actionneur

Une ligne d'instruction conditionnelle s'interprète comme suit :

ITYPE 'CID'

Si (Cond1) et/ou (Cond2) alors (action)

ITYPE est un des types d'actionneurs reconnus dans le tableau précédent

'CID' est le nom de l'actionneur qui est nécessairement de type désigné par itype

(Condi) est une relation de comparaison entre deux variables d'état :

condi = (VAR-A) GT/LT (VAR-B)

VAR peut être exprimé de 5 manières :

- une valeur numérique,
- une variable locale,
- un paramètre hydraulique désignant une grandeur connue à l'instant t en un point du réseau : débit, cote, variation de débit, variation de cote,
- le temps,
- le dernier pas de temps de calcul.
-

GT signifie « plus grand que »

LT signifie « plus petit que »

(action) est une commande s'exprimant comme suit :

action = PARAM VAR-action

PARAM est un des paramètres reconnus par itype (voir tableau 3.1).

VAR-ACTION peut être exprimé de 3 manières :

- valeur numérique,
- variable locale

- mot-clé DEFAUT : cette déclaration a pour effet d'activer le mode de pilotage par défaut déclaré dans le modèle de base, c'est pratique dans le cas d'un état contraint : en fin de séquence d'état contraint, le programme va automatiquement réactiver le pilotage initial en vigueur, sans que l'opérateur n'ait besoin de se rappeler les paramètres associés.

2.3 LES INSTRUCTIONS D'AFFECTATION ET DE CALCUL

On distingue :

- Les instructions d'initialisation et d'affectation : SET, DATA
- L'instruction de calcul : CALCUL ou CALCULN
- Les instructions de contrôle de syntaxe des instructions et des résultats : DEBUG et SHOW.

Les opérations d'affectation et de calcul peuvent porter sur des variables simples, des variables tableau ou des variables indicées de tableaux.

2.4 LES INSTRUCTIONS CONDITIONNELLES

Un jeu d'instructions complémentaires permet d'élargir les possibilités de programmation offertes. Ce sont :

- Les instructions de boucle et de rupture de séquence :
  - DO … ENDDO
  - IF, ELSEIF, ELSE, ENDIF
  - SELECTC, VALUEC, ENDSELECTC
- Les instructions de contrôle de syntaxe des instructions et des résultats : DEBUG et SHOW

2.5 LES SOUS PROGRAMMES

Les sous programmes sont une nouveauté, motivée par le projet GAO. Ce sont des unités de programmation autonomes permettant des définir des configurations de pilotage d'une unité fonctionnelle. Les sous programmes sont appelés par le programme principal u par un autre sous programmes. Ils sont régis par les instructions CALL, SUBROUTINE, ENDSUBR Les données peuvent être échangées via la liste d'arguments des instructions CALL et SUBROUTINE.

Cette fonctionnalité permet d'améliorer significativement la structure, la lisibilité et la maintenance d'un fichier de régulation dans les applications complexes telles que la GAO.

2.6 LES MÉTACOMMANDES

Les métacommandes sont des instructions préfixées par le caractère « $ ». On distingue les instructions suivantes :

- $CASE : définition des commandes attachées à une réseau
- $ SREAD_EXT : lecture d'un fichier externe de courbes x(t) pour une exploitation par le module de régulation.
- $ SREAD_WRITE : écriture des valeurs de variables simples dans un fichier externe à chaque pas de temps.

2.7 RENSEIGNEMENT DES PARAMÉTRAGES

Concrètement, une stratégie de contrôle est totalement renseignée par un fichier ASCII qui contient une série d'instructions rentrées séquentiellement.

L'utilisateur peut éditer ou modifier ce fichier avant toute simulation, via le menu Scénario de l'interface d'HYDRANET. En l'absence de ce fichier, le programme applique les algorithmes de contrôle local par défaut définis dans le modèle hydraulique.

2.8 EXÉCUTION D'UN ALGORITHME DE CONTRÔLE

Si le fichier de contrôle est détecté, les tâches de calcul se déroulent comme suit :

1. Phase de calcul des données
2. Chaque instruction de contrôle est lue et compilée par le programme.
3. Phase d'exécution

Au début de chaque pas de temps de calcul, le module de contrôle est appelé. Les instructions sont analysées en séquence et actualisées. Lorsqu'une instruction conditionnelle est vérifiée, le module de contrôle appelle la routine du module hydraulique concerné et celle-ci actualise le paramètre transmis par le module de contrôle. Tous les modules sont équipés de temporisation pour assurer des variations progressives d'état.

Deux remarques importantes :

- les variables locales sont définies dans un « COMMON ». Elles ne sont donc pas effacées entre deux appels au module de contrôle,
- lors d'une procédure de démarrage à « chaud », le programme conserve la mémoire du mode de pilotage de chaque actionneur géré par le module de contrôle au cours de la simulation antérieure. Cette potentialité est mise à profit dans le cadre des simulations en continu.

2.9 SYNTHÈSE

L'expérience acquise avec un grand nombre de projets opérationnels montre que le module de contrôle mis en place dans HYDRA est puissant et permet en pratique de satisfaire la

quasi-totalité des stratégies de contrôle envisageables. La syntaxe proposée assure une grande facilité de mise en œuvre ; elle rend ce module particulièrement adapté pour orienter et tester des stratégies de gestion de systèmes hydrauliques complexes dans le domaine fluvial comme dans celui des réseaux.

3 DESCRIPTION DU FICHIER DE CONTROLE

Le fichier de contrôle est défini dans le menu Scénario settings de l'IHM. On distingue :

- les instructions de commande d'actionneurs,
- les instructions d'affectation ou de calcul,
- les instructions de rupture de séquence,
- les instructions de contrôle visuel du déroulement des logiques de contrôle.

3.1 INSTRUCTIONS DE COMMANDE D'UN ACTIONNEUR

Un actionneur est un module hydraulique, dont certains paramètres sont modifiables via le fichier de contrôle. La liste des actionneurs est récapitulée dans le tableau 2.1.

3.1.1 Instructions conditionnelles

a) Syntaxe générale

Chaque groupe d'instructions conditionnelles est structuré comme suit :

Ligne	
1	Ctype 'cid' [DTREG] [value]
2	IF (cond1) [AND / OR] [(cond2)] THEN (action)
3	IF (cond1) [AND / OR] [(cond2)] THEN (action)
…	...
1 + n	IF (cond1) [AND / OR] [(cond2)] THEN (action)
Ligne blanche	

Ligne	Variable	Format	Définition
1	Ctype 'cid' DTREG value	Character Character Mot clé Float	Type d'actionneur (la liste reconnue est donnée dans les tableaux 2.1 et 3.1) Identificateur de l'objet de modélisation (entre quotes) Instruction facultative. Si le mot clé DTREG est présent, le programme lit dans « value » l'intervalle de temps de rafraîchissement de l'instruction en heures. Si DTREG est omis, l'instruction est rafraîchie à chaque pas de temps de calcul
2 à 1 + n	(cond1) (cond2) (Action)	Character Character Character	Condition logique portant sur l'état du système Condition logique portant sur l'état du système Action portant sur un paramètre associé à l'élément 'cid', dont le type est déclaré par Ctype

Remarques

- On peut séparer les mots clés (ou identificateurs) par au moins un et autant de blancs qu'on le souhaite, dans la limite d'une longueur totale de 256 caractères par ligne.
- Les lignes d'instruction à l'intérieur d'un groupe doivent être contiguës (pas de lignes blanches).
- 'L'identificateur de l'objet de modélisation est celui défini dans l'IHM. Il peut occuper jusqu'à 24 caractères.
- Les parenthèses entourant les instructions "cond1" et "cond2" ou "action" sont obligatoires si le mot clé IF est présent. Le groupe entre crochets peut être omis. S'il est présent le mot clé AND ou OR est requis.
- Si le mot clé IF est absent la ligne ne contient que l'instruction (action). On peut alors omettre les parenthèses entourant l'instruction.

- S'il existe plusieurs lignes après la ligne 1, la séquence logique est interprétée comme suit :

  if (condition11) and/or (condition12) then (action1)

  endif

  if (condition21) and/or (condition22) then (action2)

  endif

  etc …

Un groupe peut donc contenir plusieurs lignes d'instruction. Le programme traite chaque ligne en séquence à chaque pas de temps de régulation.

Il faut laisser une ligne blanche après la dernière instruction.

L'action est activée si la condition logique (condo 1) (et/ou la condition logique (cond2)) est (sont) satisfaite(s).

b) Syntaxe de déclaration d'un actionneur :

La désignation Cid de l'actionneur peut être :

- Un nom d'objet du modèle
- Une variable charactère contenant le nom de l'objet ; les formats suivants sont reconnus :
  - &cvar
  - &&ctab[2]
  - &&ctab[%i]

c) Syntaxe d'une instruction (condi)

La syntaxe d'une condition (condi) est indépendante de l'élément ou de son type. Elle est définie comme suit :

VAR-A	Opérande logique	VAR-B

L'opérande logique peut prendre les valeurs suivantes :

GT : « plus grand que »

LT « plus petit que »,

GE, : « plus grand ou égal à «

LE : « plus petit ou égal à »,

EQ : égal à »

NE : « non égal à »

VAR-A et **VAR-B** sont des variables d'état qui peuvent être exprimées de plusieurs manières :

1.Valeur numérique : Cette valeur figure explicitement sur la ligne de commande.

2.Variable locale : Cette variable (de type réelle) est déclarée sous la forme :

%var : var est un libellé de 24 caractères maxi.

Cette variable est définie par une instruction DATA, SET, CALCUL ou CALCULN (voir plus loin).

3.Paramètre hydraulique désignant une grandeur connue à l'instant t. Ce paramètre est exprimé par la juxtaposition de deux champs comme suit

parametre_hydraulique cid_ctrl

cid-ctrl est l'identificateur (pas de quotes) d'un point de contrôle figurant dans le modèle : « node », « singularity » ou « link ».

Les paramètres hydrauliques reconnus sont :

QLIAIS
ZLIAM
ZLIAV
DZLI
Z
QAM
QAV
DQ
ZAM
ZAV
DZ

Les règles d'appareillage du paramètre hydraulique et de l'identificateur sont précisées au §3.2.3.

A noter que l' identificateur doit exister **cid-ctrl** dans le modèle. S'il est désigné par une variable caractère il faut au préalable générer une variable locale %var via la commande SET, par exemple : SET %var = QLIAIS ZAM &&tab[5] et identifier **VAR_A** ou **VAR_B** par %var.

4.Temps exprimé par la variable : **TIME**

La variable TIME est un temps exprimé en heures relatives à la date de début de simulation.

5.Pas de temps exprimé par la variable : **DT**

La variable DT : Pas de temps de calcul, est exprimée en heures.

6.Pas de temps exprimé par la variable : **DTLAG**

La variable DTLAG : temps écoulé en heures entre 2 appels à la routine de régulation.

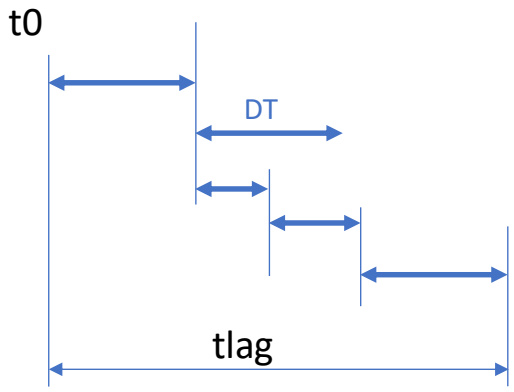
Cette variable doit être utilisée lorsque l'on souhaite calculer des cumuls de temps à partir d'un temps donné, par exemple :

Set %tag= DTLAG

Calcul %tcumul = %t0 +dtlag

%t0 est réactualisé lorsque %tcumul dépasse une valeur donnée.

Attention : DTLAG n'est pas nécessairement égal à DT. La distinction entre les 2 variables est faite en raison de la procédure de calcul avec pas de temps adaptatif dans HYDRA. Le schéma suivant clarifie la différence d'interprétation entre des deux variables :



d) Extension de l'instruction « condition » à des variables caractères

La variable VAR_A peut être une variable locale de type caractères : *&cvara.*

Il est alors nécessaire que VAR_B soit une variable de même type : &cvarb.

Si une variable est comparée à une constante il faut affectée cette constante à une variable locale pour que la comparaison soit acceptée.

e) Syntaxe d'une instruction (action)

La syntaxe d'une instruction (action) est la suivante :

PARAM VAR-action

PARAM est un paramètre reconnu par le type d'élément *ctype* (cf. tableau 3.1). Il dépend du type d'élément. L'instruction couvre les cas possibles suivants :

- pilotage d'une consigne en débit ou en cote
- déplacement d'actionneur
- consignation d'un actionneur régulable en position fixe.

VAR-action peut être exprimé de 6 manières :

1.Valeur numérique : cette valeur figure explicitement.

2.Variable locale exprimée sous la forme : % vari

Attention la variable de type %%tabi[ ] n'est pas reconnue. Il faut transformer au préalable cette variable en introduisant une variable locale comme suit :

Set %vari = %%tab[ind] ou : Set %vari = %%tab[%ind]

3.Mot clé : 'DEFAUT

La déclaration ''DEFAUT'' a pour effet d'activer le mode de pilotage par défaut, déclaré dans le paramétrage de l'élément, c'est pratique dans le cas d'un état contraint : en fin de séquence d'état contraint, le programme va automatiquement réactiver le mode initial en vigueur, sans que l'opérateur n'ait besoin de se rappeler les paramètres associés.

4. Mot clé : CLOSED

La déclaration CLOSED est applicable pour toutes les actions de type « gate » ou weir, à savoir les types : VA, DE, RGB, RGZ, RGQ, QMX, QMV, QDL1.

Elle génère la fermeture de la vanne, quelle que soit son mode de fonctionnement. Pour les seuils elle relève le seuil de 10m.

Cette commande est utile quand les actionneurs fonctionnent en tout en rien : on ne pas besoin de connaitre les cotes altimétriques pour imposer une fermeture complète.

5. Mot clé : OPEN

Cette déclaration provoque l'action inverse de CLOSED : elle ouvre totalement une vanne et abaisse totalement un seuil en position fixe.

Cette commande est utile quant un actionneur fonctionne en statique en tout ou rien : complètement ouvert ou compétemment fermé

Lorsque le mot clé DEFAUT ou CLOSED ou OPEN est déclaré le champ PARAM peut être vide : il est ignoré par le programme

6. Mot clé SEQUENCE (ou CRENEAU)

Cette instruction est suivie d'un bloc décrivant une courbe x(t), x étant le paramètre PARAM.

Le programme s'électionne la bonne valeur de x par interpolation avec le temps de calcul.

Cette disposition évite de faire suivre la déclaration de l'actionneur par une série de commandes conditionnelles dépendant du temps.

La syntaxe est la suivante :

Param SEQUENCE

{temps valeur} : temps : temps en heures relatif à la date. De la simulation.

où :

Param SEQUENCE

{date ; valeur}

Une date est exprimée sous l'un des formats suivants (les termes entre crochets sont facultatifs) :

aaaammajj

aaaammjjhrmn

aammjjhhmn

jj/mm/aaaa [hh:m:ss:cc]

aaaa/mm/jj [hh:mn:ss:cc]

jj-mm-aaaa [hh:m:ss:cc]

aaaa-mm-jj [hh:mn:ss:cc]

Exemple dans le cas où le temps est défini par une date :

RGQ 'CLI_RGQ_FICTIVE'

Qregul SEQUENCE

25/05/2016 ;2.03

26/05/2016 ;2.29

27/05/2016 ;2.40

28/05/2016 ;2.51

29/05/2016 ;2.81

30/05/2016 ;4.99

31/05/2016 ;5.07

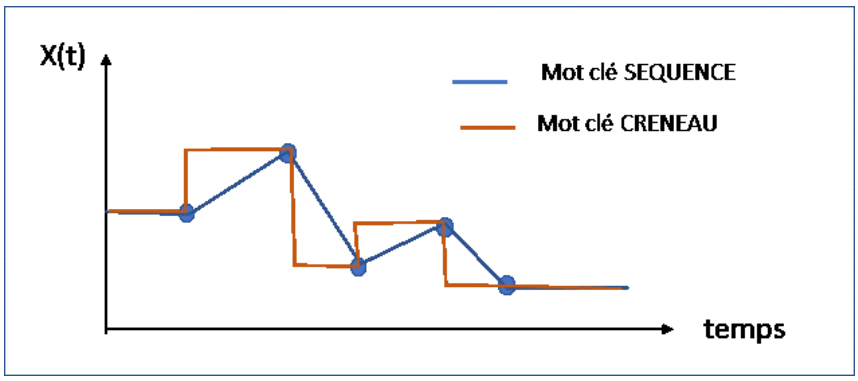
01/06/2016 ;5.96

02/06/2016 ;3.11

03/06/2016 ;1.47

Ligne blanche

Si le mot clé SEQUENCE est remplacé par le mot clé CRENEAU la syntaxe ci-dessus est inchangée mais le mode d'interpolation des données nodales est différent comme illustré ci -dessous :



f) Paramètres modifiables

Le tableau 3.1 ci-après dresse la liste des singularités concernées par le module de contrôle, ainsi que les paramètres modifiables attachés à chaque singularité.

Mot clé	**Alias**	**Paramètre modifiable (PARAM)**	**Définition**
S_gate	**VA**	**H** **ZS** **DZS**	Hauteur du seuil/radier Cote de seuil (vanne secteur) Variation de cote de seuil
S_Rweir	**DE**	**ZS** **DZS** **ZREGUL**	Cote de seuil (état contraint) Variation de cote de seuil Cote de régulation en amont immédiat vanne (mode régulation)
S_Rgate	**RGB**	**H** **ZS** **DZS** **QREGUL** **ZREGUL**	Hauteur du seuil/radier Cote de seuil Variation de cote de seuil Débit de régulation à travers la vanne (mode régulation en débit) Cote de régulation (mode régulation en cote)
L_pump_deriv	**QDP, QD**	**QPOMP** **ZPOMP** **QREGULP**	Débit maxi de pompage Cote de démarrage du pompage Débit de régulation de pompage
L_weir	**QMS**	**H** **ZS** **DZS**	Hauteur du seuil/radier Cote de seuil (vanne secteur) Variation de cote de seuil
L_gate	**QMV**	**H** **ZS** **DZS**	Hauteur du seuil/radier (pas le même effet selon le sens de d'ouverture de la vanne) Cote de seuil Variation de cote de seuil
L_Rgate	**RGZ RGQ, RG**	**H** **ZS** **DZS** **ZREGUL** **QREGUL**	Hauteur du seuil/radier (état contraint, pas le même effet selon le sens de d'ouverture de la vanne) Cote de seuil (état contraint) Variation de cote de seuil Cote de régulation au point de consigne (mode régulation) Débit de régulation à travers la vanne (mode régulation)
CL_ZT	**CLZT**	**Z0**	Cote d'eau imposée
CL_Q0	**CLQ0, HYCON**	**Q0**	
L_pump	**QMP**	**FQPOMP** **DZPOMP**	Fraction du débit total (entre 0 et 1) Translation des cotes démarrage/arrêt
Storage	**CAS**	**ZINI**	Imposition d'une cote initiale dans un casier
LSTK	**LSTK**	**alp**	Coefficient de réduction de la géométrie de la liaison
CL_DLDAT	**QDL1**	**ZS**	Cote de seuil

Tableau 3-1 : paramètres modifiables de chaque actionneur

Remarques :

- Les paramètres modifiables peuvent être classés en 3 catégories :
  - les consignes de position : hauteur d'ouverture de vanne, cote de seuil, débit de pompage,
  - les consignes de déplacement de vannes. Le déplacement dzs est appliqué sans temporisation.
  - les consignes de régulation en cote ou débit.
- Les consignes applicables dépendent du type de l'actionneur.
- Les cotes ZS imposées sont automatiquement corrigées en cas de valeurs trop basses (ex : ZS < cote radier), ou trop hautes (ex : ZS > cote voûte)
- Un ouvrage mobile qui est consigné dans un état contraint reste dans cet état jusqu'à l'émission d'une nouvelle instruction de régulation.
- Lorsque le mot clé PARAM = H : VAR-action doit être donné en mètres
- Lorsque le mot clé PARAM = ZS, Z0, ZPOMP ou ZREGUL : VAR-action doit être donné en mètres dans le référentiel du modèle de référence
- Lorsque le mot clé PARAM = QPOMP et QREGUL : VAR-action doit être donné en m^3/s
- Les fichiers annexes détaillent la signification des paramètres modifiables pour chaque actionneur.
- Le paramètre ZINI permet de modifier la cote dans un casier : utile pour adapter une condition de remplissage dans un réservoir en début de crue.

3.1.2 Instructions non conditionnelles

L'instruction : IF (cond1) [AND / OR] [(cond2)] THEN (action) peut tout à fait être remplacée par l'instruction : action.

Dans ce cas l'instruction est exécutée sans condition.

NB : ne pas entourer cette instruction de parenthèses s'il n'existe pas de condition.

3.2 INSTRUCTIONS D'INITIALISATION OU D'AFFECTATION OU DE CALCUL

Ces instructions servent à affecter des valeurs numériques et à effectuer des opérations sur les variables locales.

3.2.1 L'instruction DATA

Cette instruction sert à initialiser une valeur ou un tableau de valeurs dans une variable. La variable est initialisée dans la phase de lecture des données du fichier de contrôle. La commande est ignorée dans la phase calcul.

Elle s'écrit sur une seule ligne.

a) Initialisation d'une variable simple :}

DATA %vari = x (x : valeur réelle)

b) Initialisation d'un tableau de valeurs numériques réelles

DATA %%tabi = x1 x2 … xn (xi : valeur réelle)

c) Initialisation d'un tableau de chaines de caractéres

DATA &&ctabi = c1 c2 … cn

Le nombre de caractères de chaque chaine ci doit être compris entre 1 et 24.

Une chaine représente généralement un nom d'actionneur.

3.2.2 L'instruction DATA SEQUENCE

Lorsque le nombre de données DATA est trop important la définition des valeurs en lignes devient peu pratique, d'autant que la largeur totale d'une ligne est limitée à 150 caractères.

L'instruction DATA SEQUENCE pallie cet inconvénient avec le format suivant :

DATA SEQUENCE
var1 var2 …… Varn
Np
{val1 val2 …… valn} Np lignes

Var = tableau de valeurs %%val ou tableau de chaines de caractères : &&tab

Le nombre de variables déclarées sous la commande DATA SEQUENCE est limité à 20.

La valeur de Np peut être quelconque.

On peut définir autant de commandes DATA SEQUENCE qu'on le souhaite.

<u>Exemple :</u>

```
DATA SEQUENCE
&&pompe0   %%zstart  %%zstop  %%qregul
3
p1          101       100       0
p2          101.2     100.2     0
p3          101.4     100.4     0
```

Cette commande est équivalente aux commandes suivantes :

data = p1 p2 p3
data %%zstart = 101 101.2 101.4
data %%zstop = 100 100.2 100.4
data %%qregul = 0 0 0

3.2.3 L'instruction SET

Elle s'écrit sur une seule ligne. Les syntaxes suivantes sont reconnues :

d) Définition d'une variable contenant une valeur numérique ou une chaîne de caractères (24 caractères au plus)

SET x = VAR

La variable **x** peut être une variable simple ou la variable indicée d'un tableau. L'indice peut être lui-même une variable simple. Exemple de syntaxe licite :

%var1 %%tab [2] %%tab[%i] &ch &&ctab[3] &&ctab[%i]

L'identificateur de variable est défini par le caractère « % » suivi d'un nom quelconque de caractères au maximum.

VAR est une valeur d'état pouvant être exprimée de 7 manières :

<u>1.Valeur</u> ex : set %var1 = 39.2
Set &ch1 = chaine

<u>2.Variable locale</u> ex : set %var1 = %var13
set %var1= %%tab [3]
set %var1= %%tab [%3]
set &ch1 = &chx
set &ch1 = &&ctab[10]
set &ch1 = &&ctab[%k]

<u>3.Paramètre hydraulique</u> désignant une grandeur connue à l'instant t. Ce paramètre est exprimé par la juxtaposition de deux champs comme suit

Set = ***parametre_hydraulique*** ***cid_ctrl***

cid-ctrl est l'identificateur (pas de quotes) d'un point de contrôle figurant dans le modèle : « node », « singularity » ou « link ».

cid-ctrl peut être défini selon deux modes possibles :

- mode direct : par l'identificateur du point de contrôle
- mode indirect : par la variable &&ctab[ ] qui contient l'identificateur

Les paramètres hydrauliques reconnus sont :

QLIAIS
ZLIAM
ZLIAV
DZLI
Z
QAM
QAV
DQ
ZAM
ZAV
DZ

Les règles de cohérence suivantes s'appliquent, selon la grandeur invoquée :

i. si grandeur = **QLIAIS** : le point de contrôle est nécessairement un « link » :

- QLIAIS est alors le débit passant par le link.

  Si le link est de type uni nodal le débit est positif s'il est sortant et négatif dans le cas contraire. Par exemple pour l'objet Hy le débit associé à Qliais est négatif bien que ce soit un débit d'apport.

ii. si grandeur = **ZLIAM**, **ZLIAV**, **DZLI** : le point de contrôle est nécessairement un link :

- ZLIAM : cote du nœud amont du link
- ZLIAV : cote du nœud aval du link
- DZLI : cote différentielle ZLIAM – ZLIAV

iii. si grandeur = **Z** : le point de contrôle est nécessairement un « node ».

- **Z** est alors la cote au nœud.

iv. si grandeur = **QAM**, **QAV**, **DQ, ZAM**, **ZAV**, **DZ** : le point de contrôle est nécessairement un node appartenant à une branche (river ou manhole) ou une singularité « posée » le long d'un « reach » ou d' une « branch »:

- **QAM** : débit en amont immédiat de la singularité
- **QAV** : débit en aval immédiat de la singularité
- **DQ** : débit différentiel QAM - QAV
- **ZAM** : cote en amont immédiat singularité
- **ZAV** : cote en aval immédiat singularité
- **DZ** : cote différentielle ZAM – ZAV

Note : si le point de contrôle est un « node », la cote à ce « node » peut être indifféremment désignée par les variables, Z, ZAM ou ZAV.

Par définition un « node » » désigne indifféremment une des entités suivantes : « river_node, mandole, scorage, station node, crossroad, mesh element »

Exemples :

set %var1 = Z NOD_AB

set %var1 = QLIAIS &&pomp[2]

set %vat1 = QLIAIS &&pomp[%i] (licite à l'intérieur d'une boucle)

4.Position d'un actionneur

Il faut déclarer 2 paramètres :

- le type de la singularité tel qu'il est défini dans le tableau 3.1,
- L'identificateur de la singularité (signes ' ' entourant l'identificateur)

ex : set %var1 = S_gate 'van_32'

Cette dernière option est puissante car elle permet d'agir sur les actionneurs en fonction de l'état d'autres actionneurs.

Les valeurs renvoyées dans la variable dépendent du type de l'actionneur. Elles sont précisées ci-après :

itype	Grandeur renvoyée	PARAM correspondant
S_gate	Hauteur d'ouverture	H
S_Rweir	Cote seuil	ZS
S_Rgate	Hauteur d'ouverture	H
L_pump_deriv	Débit maxi de pompage	QPOMP
L_weir	Cote seuil	ZS
L_gate	Cote seuil	ZS
L_Rgate	Cote seuil	ZS
CL_zt	Cote d'eau	Z0

Il est possible de désigner l'actionneur par un nom de variable caractère défini dans un tableau :

ex : set %var1 = S_gate '&&ctab[3]'

ex : set %var1 = S_gate '&&ctab[%i]'

5.Temps exprimé par la variable : **TIME**

La variable TIME est un temps exprimé en heures relatives à la date de début de simulation.

6.Pas de temps exprimé par la variable : **DT**

La variable DT : Pas de temps de calcul, est exprimée en heures.

7.Pas de temps exprimé par la variable : **DTLAG**

La variable DTLAG : temps écoulé en heures entre 2 appels à la routine de régulation.

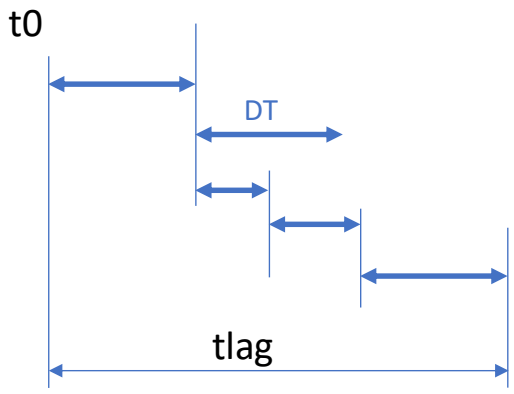
Cette variable doit être utilisée lorsque l'on souhaite calculer des cumuls de temps à partir d'un temps donné, par exemple :

Set %dtlag = DTLAG

Calcul %tcumul = %t0 +%dtlag

%t0 est réactualisé lorsque %tcumul dépasse une valeur donnée.

Attention : DTLAG n'est pas nécessairement égal à DT. La distinction entre les 2 variables est faite en raison de la procédure de calcul avec pas de temps adaptatif dans HYDRA. Le schéma suivant clarifie la différence d'interprétation entre des deux variables :



7. Option EXTERNAL

Accès à un paramètre hydraulique appartenant à un réseau « *Resj* » différent de celui « *Res_ext* » défini dans le méta commande $CASE en cours de lecture.

Cette possibilité n'est disponible que pour le cas 3 ci-dessus. La syntaxe est la suivante :

set %var1 = Z nod_ext EXTERNAL (Res_ext).

Le nœud « *nod_ext* » appartient au réseau *Res_ext.* Le mot clé « EXTERNAL » suivi du nom du réseau auquel il appartient permet d'accéder à la cote z(nod_ext), cette dernière est stockée dans la variable globale %var1 et peut donc être exploitée par le réseau « *Resi* ».

A noter que tous les paramètres d'état du réseau « *Res_ext* » (cf. §3.1.1b) sont accessibles avec cette procédure.

8.Chronique temporelle

Cette instruction est suivie d'un bloc décrivant une courbe x(t), x étant le paramètre PARAM.

Le programme s'électionne la bonne valeur de x par interpolation avec le temps de calcul.

La syntaxe est la suivante :

Set %var = SEQUENCE

{temps valeur} : temps : temps en heures relatif à la date. De la simulation.

{date ; valeur}

Une date est exprimée sous l'un des formats suivants (les termes entre crochets sont facultatifs) :

aaaammajj

aaaammjjhrmn

aammjjhhmn

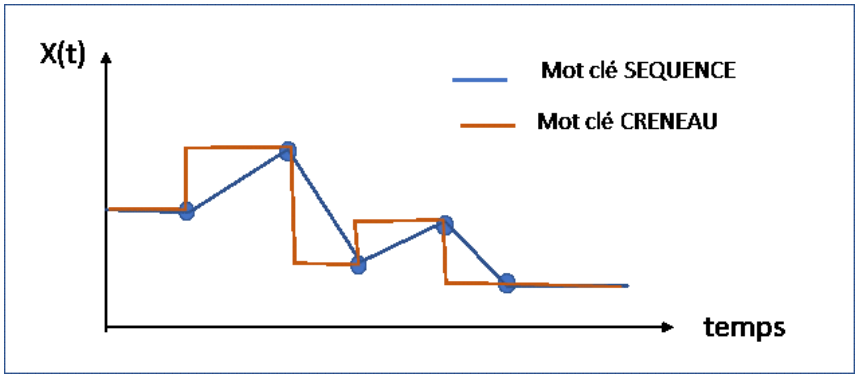
jj/mm/aaaa [hh:m:ss:cc]

aaaa/mm/jj [hh:mn:ss:cc]

jj-mm-aaaa [hh:m:ss:cc]

aaaa-mm-jj [hh:mn:ss:cc]

Si le mot clé SEQUENCE est remplacé par le mot clé CRENEAU la syntaxe ci-dessus est inchangée mais le mode d'interpolation des données nodales est différent comme illustré ci -dessous :



e) Définition d'une variable « tableau » de valeurs numériques :

SET %%tabi = x1 x2 … xn

%%tabi désigne une variable tableau. Le nombre maximum de variables tableau différentes est 100 pour chaque réseau d'un projet.

xi = valeurs numériques du tableau. On peut définir 200 valeurs au maximum.

f) Définition d'une variable « tableau » de chaines de caractères

SET &&tabi = cx1 cx2 … cxn

&&tabi désigne une variable tableau. Le nombre maximum de variables tableau différentes est 100 pour chaque réseau d'un projet.

xi = chaine de caractères du tableau. On peut définir 200 valeurs au maximum.

Contrairement à l'instruction DATA l'instruction SET est exécutée à chaque pas de temps de calcul.

3.2.4 L'instruction SAVE

L'instruction SAVE permet de sauvegarder la valeur d'une variable simple ou d'une variable « tableau » au temps *tsave* défini dans la commande SAVE du fichier X_R.cmd.

Les valeurs des variables sont stockées dans le fichier X_R_dy.ini.

En cas démarrage à chaud le contenu di ficher X_R_dy.ini est lu à la fin de la phase de lecture du fichier de contrôle et les valeurs des variables lues dans ce fichier écrasent celles initialisées dans le fichier de contrôle.

Syntaxe :

SAVE %var1

SAVE %%tab1

SAVE &ch1

SAVE &&ctab1

Cette commande est notamment utile dans le cas où une variable est initialisée en début de calcul d'une chronique avec la commande DATA. Cette valeur évolue au cours de la simulation et on peut souhaiter la conserver en fin de simulation pour en disposer au début de la simulation suivante. En l'absence de commande SAVE cette variable serait réinitialisée à la valeur déclarée par la commande DATA.

3.2.5 L'instruction CALCUL ou CALCULN

Cette commande permet d'exécuter des calculs complexes sur des variables simples ou des tableaux. Elle regroupe désormais les fonctionnalités des instructions CALCUL et CALCULN. L'un ou l'autre de ces mots clés peut être utilisé indifféremment.

On distingue 2 cas :

Cas 1 : le membre de gauche est un tableau :

Calcul d'une variable tableau à l'aide d'une fonction:

CALCUL %%tabi = #func (arg1, arg2, ….., argn)

« func » est l'identificateur d'une fonction accessible depuis la bibliothèque intégrée au module de contrôle. Le nombre et le type des arguments dans la liste d'appel dépendent de la nature de la fonction : ce peuvent être des variables simples, des valeurs numériques ou des variables tableaux. Dans le cas d'une variable tableau calculée la taille du tableau est déterminée par la nature de la fonction ou la liste d'arguments.

Calcul arithmétique sur des variables tableaux :

CALCUL %%tabi = %%tabj « opérateur » %%tabk

L'opération s'applique sur chaque variable du tableau.

Les tableaux i, j et k doivent comporter le même nombre de valeurs.

Les opérateurs reconnus sont :

- « ^ » : l'opérateur puissance
- « * » : l'opérateur multiplication
- « / » : l'opérateur division
- « + » : l'opérateur addition
- « - » : l'opérateur soustraction

Cas 2 : le membre de gauche est une variable :

La variable peut être une variable simple ou la variable indicée d'un tableau. L'indice peut être lui-même une variable simple. Exemple de syntaxe licite :

%var %%tab [2] %%tab[%i]

Calcul d'une variable tableau à l'aide d'une fonction :

CALCUL %vari = #func (arg1, arg2, ….., argn)

« func » est l'identificateur d'une fonction accessible depuis la bibliothèque intégrée au module de contrôle. Le nombre et le type des arguments dans la liste d'appel dépendent de la nature de la fonction : ce peuvent être des variables simples, des valeurs numériques ou des variables tableaux. Les fonctions implantées dans le module de contrôle sont définies à l'annexe 1.

Calcul arithmétique sur des variables :

Cette commande permet d'exécuter des opérations arithmétiques complexes en une seule ligne de commande et d'affecter le résultat à une variable locale %var.

Les opérandes sont :

- Soit une variable locale simple,

- La variable indicée d'un tableau. L'indice est déclaré entre crochets [ ]. L'indice peut être désigné par une variable.
- Soit une valeur réelle

Les opérateurs reconnus sont :

- « ^ » : l'opérateur puissance
- « * » : l'opérateur multiplication
- « /» : l'opérateur division
- « + » : l'opérateur addition
- « - » : l'opérateur soustraction

Les règles habituelles de priorité d'exécution des calculs s'appliquent à la liste ci-dessus avec l'ordre de priorité décroissante suivant : « ^ », « * », « / », « + », « - ».

Les séparateurs « (« et «) » permettent d'encapsuler des sous calculs et donc d'orienter l'ordre d'exécution des calculs.

La commande s'écrit :

CALCUL %vari = {expression arithmétique}

<u>Exemple 1</u>

Calcul %var1 = ((3+%va2) / (4. - %var3)) ^ (3/2)

Si %var2 = 2. Et %var%3 = 1. On obtient :

%var1 = 2.1517

<u>Exemple 2</u>

Calcul %var1 = 3. + 4. * 2. -1.

On obtient : %var1 = 10.

<u>Exemple 3</u>

Calcul %var1 = 3. + 4. * (2. -1.)

On obtient : %var1 = 7.

<u>Exemple 4 avec variable indicée d'un tableau :</u>

Calcul %var1 = %%tab [3] + 4. * (2. -1.)

Si %%tab [3] = 3 on obtient : %var1 = 7.

%var1 = %%tab[%i] + 4. * (2. -1.)

Si %i=3 et %%tab [3] = 3 on obtient : %var1 = 7.

<u>Attention</u> : chaque opérateur doit être précédé d'une variable ou d'une valeur numérique.

Par exemple l'expression %var = -1 + 3 n'est pas licite. Il faut écrire :

%var = 0 -1 +3 ou %var = 3 - 1

3.2.6 Les noms de variables locales

Les variables simples et les variables tableaux sont définis par des chaînes de 16 caractères au maximum, précédées des caractères « % » ou « %% » selon le type de la variable.

Afin d'assurer la compatibilité avec des versions antérieures il est possible de continuer « nommer » les variables locales comme suit :

%i.xxx

où « i » est un nombre compris entre 1 et 99 et xxx est un libellé de 8 caractères au maximum attachés à la variable %i par le signe '.'

Le libellé est ignoré par le programme, celui-ci ne reconnaissant que la variable i attachée au signe « % ». Par exemple les variables %10. toto et %10.titi sont interprétées en interne comme la même variable locale %10.

3.3 LES INSTRUCTIONS DE RUPTURE DE SÉQUENCE

3.3.1 Les instructions de fin et les commentaires

Afin d'augmenter la lisibilité du fichier de régulation, on peut introduire le signe « ! » sur n'importe qu'elle ligne : les caractères de la ligne inscrits à droite de ce signe sont ignorés.

Par ailleurs, le programme s'arrête de lire le fichier lorsqu'il rencontre une ligne commençant par les caractères « - - - - - » (un caractère «-» suffit).

La commande FIN provoque le même résultat.

3.3.2 Les instructions conditionnelles IF ..THEN, ELSEIF..THEN, ELSE, ENDIF

Elles s'écrivent sous la forme :

IF (condi) **THEN**

{ liste d'instructions }

ELSEIF (condj) **THEN**

{ liste d'instructions }

ELSEIF (condk) **THEN**

{ liste d'instructions }

ELSE

{ liste d'instructions }

ENDIF

(condi) est une instruction logique décrite au paragraphe 3.1.1.2.

Lorsque (condi) est satisfaite, la liste d'instructions qui suit est exécutée. Sinon le programme se branche sur la commande ELSEIF…THEN, ELSE ou ENDIF qui suit la commande IF……THEN.

{liste d'instructions} est constituée d'une énumération quelconque d'instructions de commandes d'actionneur, d'instructions d'affectation ou d'instruction de calcul.

Le programme accepte des instructions logiques emboîtées jusqu'à un niveau d'emboitement égal à 10.

Si une condition est satisfaite la liste d'instructions qui suit est exécutée, le programme se branche ensuite directement sur la commande ENDIF.

3.3.3 L'instruction DO … ENDDO

Cette instruction permet de répéter des instructions dans une boucle. La syntaxe est la suivante :

DO %i = %ideb %ifin [%inc]

(instructions)

ENDDO

La variable de comptage %i est initialisée à %ideb en début de boucle.

Elle est ensuite incrémentée de la valeur %inc a chaque fois que l'instruction ENDDO est atteinte. Le programme se branche alors sur l'instruction DO tant que la valeur prise par %i reste inférieure ou égale à %ifin.

<u>Commentaires :</u>

- Les variables %ideb %ifin %inc peuvent être remplacées par des valeurs entières.
- La variable %inc est facultative. Si elle n'est pas définie le programme sélectionne la valeur 1 par défaut.
- Le programme autorise des boucles DO Imbriquées jusqu'à un niveau d'imbrication de 10.

Cette commande peut être utilisée pour modifier/actualiser les valeurs des variables d'un tableau. Elle surtout intéressante pour agir sur plusieurs actionneurs en séquence sans être obligé de dupliquer les instructions pour chaque actionneur. On autorise pour cela l'écriture suivante pour l'instruction de commande d'un actionneur :

Ctype '&&ctab [3]'

L'indice peut être remplacé par une variable %i : ex : Ctype '&&ctab[%i]'

<u>Exemple d'application :</u>

data &&Vme = V_HMZ11 V_HMZ14 V_HMZ21 V_HMZ24 V_HMZ31 V_HMZ34

do %i = 1 6

s_Rgate '&&Vme[%i]'

h 0.

enddo

Dans cet exemple les hauteurs d'ouverture des 6 vannes désignées par le tableau de chaines de charactères &&BME sont initialisées à 0.

3.3.4 Les instructions SELECTC…VALUEC…ENDSELECTC

Ces instructions permettent de spécifier les instructions à exécuter lorsqu'elles sont bordées par les commandes SELECTC … ENDSELECTC. La syntaxe est la suivante :

SELECTC (%var)

VALUEC (var_1)

(instructions)

VALUEC (var_2)

(instructions)

VALUEC (var_n)

(instructions)

ENDSELECTC

Les termes var_i déclarés par VALUEC sont des valeurs numériques. Un bloc d'instructions sous la commande VALUEC (vari) n'est exécuté que si var_i est égale à la valeur numérique contenue dans la variable %Var.

<u>Commentaires :</u>

- Le nombre de valeurs possibles prises par la variable %var est limité à 20.
- Le programme autorise des boucles SELECTC Imbriquées jusqu'à un niveau d'imbrication de 10.
- Il est possible de déclarer plus d'une valeur à la suite de la déclaration VALUEC. exemple : VALUEC (2 5 10) : les instructions sous cette déclaration sont exécutées si %var est égale à une des valeurs 2, 5, 10.

Ces commandes s'appliquent également si la variable numérique %var et remplacée par une variable de chaîne de caractères &cvar. Les valeurs numériques var_i sont alors remplacées par des chaines de caractères.

Exemple :

```
$ case (model1)

debug  0. 1.0  0.1

data &&cval =  petit  moyen  grand

do %i = 1 3
    set &ctmp = &&cval[%i]
    selectc( &ctmp)

    valuec ( petit)
        show &ctmp

    valuec ( moyen)
        show &ctmp

    valuec ( grand)
        show &ctmp

    endselectc

enddo
```

Le fichier .out produit les résultats suivants à chaque pas de temps de scrutation :

```
command SHOW :
temps :        0.802    CTMP  PETIT
temps :        0.802    CTMP  MOYEN
temps :        0.802    CTMP  GRAND
```

Remarque importante :

Les chevauchements entre blocs d'instructions sont interdits. Hydra opère un contrôle pour les blocs de même type, mais pas pour les blocs de types différents. Il appartient donc à l'utilisateur de vérifier la cohérence de son script pour éviter tout comportement imprévisible du code lors la phase d'exécution des calculs.

3.4 LES INSTRUCTIONS DE SUIVI DES FICHIERS DE CONTRÔLE

3.4.1 Commande DEBUG

Cette instruction est obsolète à partie de la version 3.10.3 d'hydra/Qgis : elle ne pose de difficulté avec les modèles construits dans des versions antérieures mais elle sera ignorée.

Elle est remplacée désormais par la meta commande $DEBUG (cf. §3.8.4).

3.4.2 Commandes VIEW_COMMAND ... ENDVIEW_COMMAND

Ces instructions servent à délimiter une portion de code dans un fichier de contrôle en phase d'exécution. Quand une instruction view_control est rencontrée Hydra affiche les commandes réellement exécutées et affiche en plus les résultats de chaque test comparatif. Cet affichage perdure jusqu'à ce que l'instruction endview_control est rencontrée.

Cet affichage est effectué uniquement aux pas de temps déclarés par la Meta commande $DEBUG. Exemple :

```
-------------------------------
Debug:temps d'affichage des commandes (heures)  :      3.053
 dans fichier :C:\Users\thierry.lepelletier\.hydra\gao_etude\data\regulation'
-------------------------------

command     4  VIEW_COMMAND
command     5  IF  (  Z NODA_24427 LT 30.55  )  THEN
True

command     6     QMV '94_OXII_VPO' DTREG 2.0
command     7     h 2
command     8     QMV '94_OXII_VPM' DTREG 2.0
command     9     h  0.
command    10     RGZ '94_OXII_RGQ_FICT' DTREG 2.0
command    11     qregul 0.0
command    12  ELSE
command    34  ENDIF
command    35  ENDVIEW_COMMAND
-------------------------------
```

Un test est effectué par la commande 5. . Il est trouvé valide par le programme . Par conséquent les lignes 6-11 sont exécutées. Les commandes sous l'instruction ELSE (commande 12) sont ignorées et donc non affichées.

On peut tout à fait définir plusieurs séquences d'affichage. Il suffit d'entourer chaque groupe d'instructions par les commandes View_command … Endview_command

3.4.3 Commande Show

L'instruction SHOW est définie comme suit :

SHOW x 'commentaire'

L'objet **x** peut être :

- un tableau %%tab ou &&ctab : toutes les variables du tableau tab sont affichées
- une variable simple %var ou &cvar
- une variable indicée d'un tableau : %%tab [2], %%tab[%i], &&ctab[3], &&ctab[%i].

Lors de l'exécution la valeur prise par la variable **x** est copiée dans le fichier .out tous les « dt_debug » pas de temps, accompagnée du commentaire contenu dans « commentaire » et du temps de calcul.

3.4.4 La commande WRITE

Cette commande permet d'écrire le contenu d'une variable à chaque pas de temps dans le fichier déclaré par le méta commande WRITE_EXT.

La commande s'écrit comme suit :

Write %var cid

Où cid désigne un identificateur de courbe. Deux formats sont acceptés :

- cid = 'xxx' : chaine de caractères entre quotes.
- cid = &&tab[3] ou &&ctab[%i] : indice ou variable indicée d'un tableau.

Le champ cid est obligatoire.

3.5 ACTIVATION D'UN RÉGIME TRANSITOIRE RAPIDE

La modification brutale de l'état de fonctionnement d'un actionneur peut générer des conditions initiales propices à la formation d'une intumescence lorsque l'option de calcul « AFFIN » est activée. Ce paramétrage est réalisé par l'intermédiaire du mot clé **'TREACS'** entre quotes, suivi de la durée en secondes que l'on souhaite imposer pour atteindre le nouvel état.

Dans l'exemple ci-dessous, le seuil du déversoir 'DE' est abaissé en 2 secondes de la position initiale à la nouvelle position zs=100. Cette manœuvre est commandée à partir de l'instant t=1h.

de 'DE ' 'treacs' 2.

If (time gt 1.) then (zs 100.)

Si le mot clé est omis, la même manœuvre sera exécutée, mais avec un temps d'exécution très supérieur, imposé par le programme, compatible avec un fonctionnement graduellement varié, mais incompatible avec la formation d'une intumescence.

Dans la version actuelle d'Hydra, 4 types d'objets reconnaissent le mot clé TREACS :

1. L'objet Weir,

2. L'objet L-Weir,
3. L'objet Pump,
4. L'objet Pump_deriv.
5. L'objet S_gate
6. L_gate

Avec ces objets on peut simuler par exemple :

- La génération d'une onde de rupture de barrage,
- La génération d'une intumescence à la suite d'une coupure brutale d'alimentation électrique d'une station de pompage.

3.6 OPÉRATIONS SUR LES VARIABLES TABLEAUX

Les tableaux sont de deux types :

- tableaux de valeurs numériques : %%tab
- tableaux de chaines de carctères : &&ctab

le programme alloue pour chaque tableau une taille maximum égal à 200, mais il conserve la taille définie dans la dernière instruction SET ot DATE rentrée.

Il est donc tout à fait licite de définir une taille dans une instruction et de rédéfinir cette taille pour la mêm variable tableau dans une autre nistruction.

!---- dans l'intr

Set %%tab = 1 2 3 4 5

3.7 LES META COMMANDES

Une méta commande est préfixée par le caractère « $ ». Elle est placée à l'extérieur d'un groupe de données de réseau et sa portée concerne l'ensemble des réseaux.

3.7.1 La méta commande $CASE

L'instruction est la suivante

$ CASE (Cres1)

{ groupe d'instructions relatives au réseau Cres1 }

$ CASE (Cres2)

{ groupe d'instructions relatives au réseau Cres1 }

$ CASE (Cresn)

{ groupe d'instructions relatives au réseau Cresn }

Le groupe d'instructions à la suite de la meta-command CASE est exécuté si le paramètre entre parenthèses correspond à un des noms de réseaux listés dans le fichier SCENARIO.NOM. Ce fichier est produit par HYDRA au début de chaque nouvelle simulation.

Chaque groupe de commandes à la suite de l'instruction $CASE (Cresi } ne s'applique qu'aux entités du réseau Cresi jusqu'à la déclaration d'une autre commande CASE

Cette procédure alternative permet de conserver les mêmes structures de fichiers de commande, que l'on travaille en mode « Réseau simple » ou en mode « Complexe ».

Contrairement aux versions antérieures d'Hydra un nom d'élément est défini sans ambiguïté car il se rapporte à un réseau donné : il n'est donc plus nécessaire de préfixer ce nom par celui du réseau.

Seule restriction : les instructions ne peuvent pas mélanger les noms de réseaux différents

Commentaires importants :

- Aucune instruction ne peut être définie à l'extérieur d'un groupe préfixé par la commande CASE, même dans le cas d'un seul réseau.
- Les variables définies dans un groupe « CASE » ne sont visibles que dans ce groupe. Les variables de même nom peuvent être définies dans deux groupes différents. Elles sont considérées comme distinctes dans le programme.
- Le programme vérifie le nom de tous les réseaux défies par CASE. Un message d'erreur est produit si un réseau n'appartient pas au projet, même s'il n'est pas activé dans le scénario de calcul.
- Les commandes peuvent être partagées entre plusieurs fichiers. Un même réseau peut être défini dans plusieurs commandes CASE différentes. Le programme cherche toutes les commandes d'un même réseau et les concatène en une même suite d'instructions.
- La première instruction en début de chaque fichier doit être la commande CASE.

Attention : la nouvelle gestion des méta-commande est différente de celles appliquées dans les versions antérieures d'Hydra ; les fichiers de contrôle doivent être adaptés en conséquence.

3.7.2 La méta commande $READ_EXT

Définition : lecture d'un fichier externe de courbe x(t) pour une exploitation par le module de régulation.

Cette commande doit être déclarée en début de fichier avant la première méta commande CASE.

L'instruction est la suivante :

$READ_EXT

{nom-de-fichier}

Le fichier se compose d'une série de courbes x(t) au format Hydra simple ou Excel

Dans la version actuelle :

- Le nombre maximum de courbes =200
- Le nombre maximum de points par courbe = 1000

On souhaite extraire d'une des courbes de ce fichier la valeur x_k correspondant au temps t_k

Pour accéder à une valeur de ce fichier il faut utiliser la fonction *get_ext* définie au chapitre 5.

3.7.3 La méta commande $WRITE_EXT

Définition : Cette méta commande active l'écriture des valeurs de variables simples dans un fichier externes à chaque pas de temps.

Cette commande doit être déclarée en début de fichier avant la première méta commande CASE.

L'instruction est la suivante :

$WRITE_EXT

{nom-de-fichier}

Le fichier de sortie est au format Hydra multi colonnes

%

Date ; nom1 ; nom2 ; ---------- ; nomn

{aaammjj hh :mn ; x1 ; x2 ; ----- ; xn}

L'écriture du contenu d'une variable dans ce fichier est activée via la commande *write suivi du nom de la variable.*

3.7.4 La méta commande $DEBUG

Cette méta commande doit apparaitre à l'extérieur d'une méta commande CASE ; elle s'applique à l'ensemble des modèles pilotés par le calcul. Elle s'écrit comme suit :

$DEBUG

tdebh tfinh dth [DUMP]

tdebh : début du temps de scrutation en heures relatives/date0

tfinh : fin du temps de scrutation en heures relatives/date0

dth : intervalle de temps entre 2 scrutations

DUMP : mot clé optionnel

Le suivi de code est écrit dans chaque fichier X-Gi.out aux pas de temps successifs :

tdebh, tdebh+dth, tdebh+2*dth + …. tfinh-dt.

L'option DUMP restitue la totalité des instructions lues en phase de lecture des données et attribue un numéro à chaque instruction comme suit :

```
**********************************
  commandes du fichier de controle ( si option dump) : C:\Users\thierry.lepelletier\.hydra\gao_e
**********************************

command    4  VIEW_COMMAND
command    5  IF  (  Z NODA_24427 LT 30.55  )  THEN
command    6     QMV '94_OXII_VPO' DTREG 2.0
command    7     H 2
command    8     QMV '94_OXII_VPM' DTREG 2.0
command    9     H  0.
command   10     RGZ '94_OXII_RGQ_FICT' DTREG 2.0
command   11     QREGUL 0.0
command   12  ELSE
command   13    IF  (  Z NODA_24427 LT 31.05  )  THEN
command   14     QMV '94_OXII_VPO' DTREG 2.0
command   15     H 2.
command   16     QMV '94_OXII_VPM' DTREG 2.0
command   17     H  0.
command   18     RGZ '94_OXII_RGQ_FICT' DTREG 2.0
command   19     QREGUL 0.4
command   20      ELSE
command   21     QMV '94_OXII_VPO' DTREG 2.0
command   22     H 2.
command   23     QMV '94_OXII_VPM' DTREG 2.0
command   24     H  0.
command   25     RGZ '94_OXII_RGQ_FICT' DTREG 2.0
command   26     QREGUL 0.8
command   27      ENDIF
```

Ces numéros servent dans la phase exécution des calculs à mieux se repérer sur le suivi des instructions réellement appliquées par le programme en fonction des tests conditionnels définis dans les fichiers script de contrôle.

4 LES SOUS PROGRAMMES

4.1 PRINCIPES

4.1.1 Défintiion

Un sous-programme est un bloc de commandes encapsulé par les instructions :

SUBROUTINE *subr (arg1, … argn)*

……..

ENDSUBR

Un sous-programme est obligatoirement rattaché à un réseau ; il n'est donc accessible que par le programme principal également rattaché à ce réseau ou par un autre sous-programme rattaché au même réseau.

Les sous programmes sont paramétrés dans des fichiers scripts déclarés par le mot clé : *F_REGUL. **Ils sont déclarés obligatoirement à la suite de tous les autres fichiers de contrôle**

Les instructions sont les mêmes que celles utilisées par le programme principal et les variables sont de même nature, à la différence près que ces variables sont locales, c'est-à-dire qu'elles ne sont accessibles qu'à l'intérieur du sous-programme.

L'appel à un sous-programme est réalisé à l'aide de la commande : CALL *subr (arg1, …, argn)*. Cette commande peut être insérée n'importe où dans le programme principal ou dans un sous-programme. Les arguments contenus dans la liste appelés par la commande CALL sont des variables de type %, %%, &, &&.

4.1.2 Exemple d'agencement

Après lecture des fichiers scripts les instructions se présentent comme suit dans le cas de deux réseaux :

$CASE (res1)

{liste d'instructions du programme principal attaché au réseau res1}

$CASE (res2)

{liste d'instructions du programme principal attaché au réseau res2}

$CASE (res1)

SUBROUTINE subr11 ( )

{liste d'instructions du sous-programme subr11 attaché au réseau res1}

END

```
SUBROUTINE subr12 ()
{liste d'instructions du sous-programme subr12 attaché au réseau res1}
ENDSUBR

$CASE (res2)
SUBROUTINE subr21 ()
{liste d'instructions du sous-programme subr21 attaché au réseau res2}
END
SUBROUTINE subr22 ( )
{liste d'instructions du sous-programme subr22 attaché au réseau res2}
END
```

Dans cet exemple les sous-programme subr11 et subr12 ne peuvent être appelés que par le programme principal du réseau res1.

De même les sous-programme subr21 et subr22 ne peuvent être appelés que par le programme principal du réseau res2.

4.1.3 Intérêt présenté par le sous-programme

Le sous-programme élargit la notion de fonction : il peut être appelé plusieurs fois par le programme principal sans devoir retaper à chaque fois les listes d'instructions répétitives.

Cette possibilité contribue également à clarifier considérablement l'écriture de gros fichiers de contrôle (tels que les fichiers développés pour l'application GAO) et à segmenter les blocs d'instructions en unités distinctes.

Dans le projet GAO le programme principal se contente de lire les données fournies par le superviseur et de se brancher sur les sous programmes spécifiques gérant les unités fonctionnelles. Les développements des blocs des différents sous programmes peuvent être effectués par des équipes travaillant en parallèle, avec les gains de productivité qui en découlent.

Un autre intérêt des sous programmes, une fois développés et testés, concerne la possibilité de les lire en format déjà compilé, ce qui évite de les retraduite en code compilé à chaque nouvelle exécution de calcul (cette option reste à implémenter).

4.1.4 Règles d'utilisation

1. un sous-programme est nécessairement attaché à un réseau : il ne peut pas être partagé entre des réseaux différents.

2. Les variables utilisées par le sous-programme sont locales et reconnues uniquement par le sous-programme.

3. Un sous-programme peut contenir une liste non nulle d'arguments. Ces derniers sont définis par l'instruction CALL appelante. Le nombre et les types d'arguments doivent correspondre entre la commande CALL et le sous-programme.

4. Les appels à des sous programmes successifs sont possibles. Le nombre de sous programmes ainsi emboités peut être quelconque. Cette architecture est gérée par un concept de « pile » adapté au langage du fichier de contrôle.

5. Les instructions présentes dans un sous-programme sont les mêmes que celles du programme principal.

6. L'écriture du code nécessaire à l'implémentation des sous-programmes est conçue pour minimiser les modifications apportées au code existant.

4.2 DESCRIPTION DES INSTRUCTIONS

4.2.1 Appel à un sous-programme

CALL Sp (arg1 arg2 …. argn)

Sp : nom de la routine (24 caractères max.)

Nombre maximum d 'arguments =20.

Arg est un champ défini par un des 3 types suivants :

- variable scalaire : %var
- tableau de scalaires : %%tab
- variable caractère : &cvar
- tableau de chaine de 24 caractères : &&ctab

Chaque variable doit être défini préalablement à l'appel CALL par une instruction de type DATA.

Les valeurs des arguments peuvent être indifféremment définies lors de l'appel CALL ou calculées/recalculées dans la routine.

4.2.2 Déclaration d'un sous-programme

SUBROUTINE Sp (arg1 arg2 …. argn)

Le nombre et le type d'arguments doivent être identiques avec ceux de l'instruction CALL

Dans les versions antérieures il était nécessaire de préciser la taille maximum des tableaux à l'aide du délimiteur [ ]. Désormais ce délimiteur n'est plus nécessaire, car la taille maximum des tableaux est définie par le programme.

Pour des raisons de compatibilité le délimiteur [ ] est toujours admis mais il n'a plus d'utilité. Il est donc indifférent d'écrire :

%%tab [15] &&Ctab[100]

ou : %%tab &&Ctab

les tailles de tableaux entre crochets sont ignorées.

4.2.3 Déclaration de fin d'un sous-programme

ENDSUBR Sp

4.2.4 Déclaration de variables d'actionneurs dans une commande CALL

Il est possible de déclarer en arguments des variables caractères contenant des noms d'actionneurs. La syntaxe de l'exemple suivant est admise :

```
 Data &&ctab =  act1  act2 act3
Call sp1 (&&ctab)
-----------------------------
Subroutine sp1 ( &&ctab1 )
Data %%zz  0  0  0  0  0
Rgz  &&ctab1[3)
Close                                  ! fermeture de la vanne act3
Calcul %dim = #DIM( &&ctab1)      ! lecture de la dimension du tableau ctab
do %k = 1 %dim
  Set  %%zz(k) = zam &&ctab[%k]
! stockage des niveaux d'eau amont de chaque actionneur dans le tableau zz.
end do
```

On note au passage la récupération de la taille du tableau &&ctab1 via la fonction #DIM

4.3 EXEMPLE

Le sous programme développé ci-après a pour fonction de réguler les débits d'un groupe de pompes à vitesse variable. Il ajuste à chaque pas de temps le débit de chaque pompe pour satisfaire la cote de régulation souhaité à l'aide d'une alogirithme de régulation PID.

Le fichier de contrôle se compose d'un programme principal décrivant un cas test et du sous programme SREGUL_POMPE qui peut être utilisé pour des groupes de pompages différents.

Cet exemple utilise des fonctionnalités avancées du module de cotôlée, notamments les variables tableaux en liaison avec les instructions DO ...ENDDO.

Programme principal

```
$ case (model1)

debug  0. 5.0  0.001

!  cas test de regulation d'un groupe de 3 pompes

data sequence
&&pompe0   %%zstart  %%zstop  %%qregul
3
p1            101        100       0
p2           101.2       100.2     0
p3           101.4       100.4     0

data %npompe0 = 3


data %e_last = 0.
data %ecum_last = 0.
data %%pid0 = 5. 0.  400

data %%qi = 0 0 0
data %zregul = 100.5

set %zbache = Z bache
set %dtlag = DTLAG

call sregul_pompe(&&pompe0 %npompe0 %%zstop %zbache %zregul  %%pid0 %e_last %ecum_last  %dtlag %%qregul )

do  %i = 1  %npompe0
    set %qregul = %%qregul[%i]
    show %qregul

    qmp  '&&Pompe0[%i]'
    qregulp %qregul

enddo
```

Sous-programme *Sregul_Pompe*

```
subroutine SREGUL_POMPE ( &&pompe[10] %npompe  %%zstop[10] %z %zregul  %%pid[3] %e_last %ecum_last  %dtlag %%qregul[10]

!   regulation PID  d'un goupe de pompage pour satisfaire une cote de regulation zregul
!   zsart : cote de detection niveau bas d'une pompe
!   zstop  : cote de detection niveau haut d'une pompe
!   z : cote d'eau dans la bache de pompage
!   pid(3) coefficents PID de refulation du groupe de pompage
!   e_last  :ecert z-zregul au dernier pas de temps
!   ecum_last : ecart dumule : int (e*dt)
!   qregul : nouvelle cote de regulation de chaque pompe
!
data %%alp = 0 0 0 0 0 0 0 0 0 0
calcul %e=%z-%zregul
calcul %dt=%dtlag*3600.
calcul %de = %e -%e_last
calcul %ecum_last = %ecum_last + %e*%dt
calcul %dqtot = %%pid[1]*%e + %%pid[2]*%ecum_last + %%pid[3]*%de/%dt
set %e_last = %e
set %qtot=0.

do %i=1 %npompe
    set %qi = qliais &&pompe[%i]
    calcul %qtot= %qtot +%qi
enddo

calcul %qregul = %dqtot+%qtot
show %qtot
show %qregul

set %np =0.
do %i =1 %npompe
    set %zstop= %%zstop[%i]
    if (%z lt. %zstop ) then
       set %%alp[%i] =0.0
    else
       set %%alp[%i] =1.0
       calcul %np = %np+1
    endif
enddo

if (%np eq 0) then
    do %i =1 %npompe
      set %%qregul[%i]=0.
    enddo
else
    show %dqtot
    do %i =1 %npompe
        calcul %%qregul[%i] = %qregul*%%alp[%i]/%np
        show %np
        show %%alp[%i]
        show %%qregul[%i]
    enddo
endif
endsubr SREGUL_POMPE
```

5 EXEMPLES DE FICHIER DE CONTROLE

5.1 EXEMPLE 1

Pilotage d'une consigne de débit d'un module RGQ en fonction du temps

```
RGQ   'VAL2 _ ANTOB92_RQ2'
IF   ( TIME  LT  3.0 )   THEN  ( QREGUL   0.5 )
IF   ( TIME  GT  3.0 )   THEN  ( QREGUL   4.0 )
IF   ( TIME  GT  4.0 )   THEN  ( QREGUL   1.0 )
```

Avec ces instructions :

QREGUL = 0.5 si TIME < 3.0

QREGUL = 4.0 si 3.0 < TIME < 4.0

QREGUL = 1.0 si TIME > 4.0

5.2 EXEMPLE 2

Pilotage d'une consigne de débit d'un module RGQ en fonction d'une détection de niveau dans le réseau au point FRE57

```
L_Rgate   'ANTOB92_RQ2'
IF   ( Z  FRES57  LT  38.5 )   THEN   ( QREGUL   0.5 )
IF   ( Z  FRES57  GT  40.0 )   THEN   ( QREGUL   4.0 )
```

Le débit de consigne à travers la vanne RQ2 sera positionné à :

- 4.0 m^3/s en cas de détection du dépassement de cote 40.0 au point FRES57
- 0.5 m^3/s en cas de détection de cote inférieure à 38.5 au point FRES57

5.3 EXEMPLE 3

Consignation en position fixe d'une vanne normalement régulée

```
L_Rgate   'RQ2'
IF   ( TIME  GT  0.5  )   THEN   ( ZS   35 )
```

La vanne RQ2 sera immobilisée à la cote 35 pour t > 0.5 heure. Elle sera régulée selon les paramètres définis par défaut dans le modèle pour t < 0.5 heure.

5.4 EXEMPLE 4

Réactivation d'un mode de régulation d'une vanne consignée en position fixe

RGQ 'ANTOB92-RQ2'

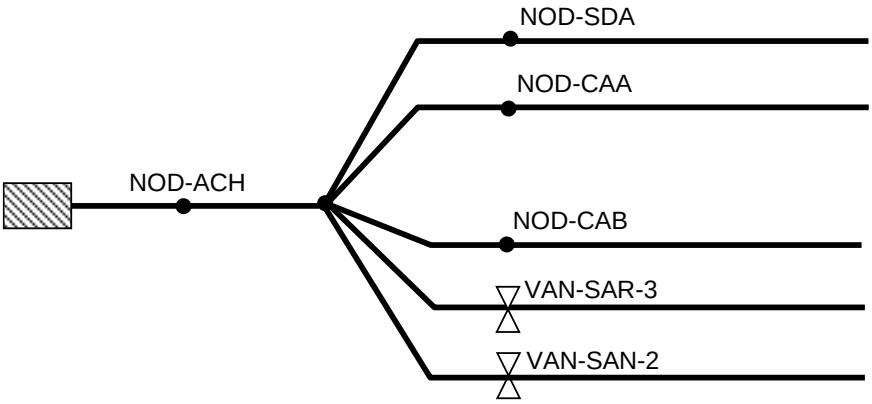
IF (TIME GT 10) THEN (QREGUL "DEFAUT")

5.5 EXEMPLE 5

Utilisation de variables locales : application à la régulation du débit à Achères

On veut réguler les débits par les vannes VSAR-3 et VSAN-2 de façon à limiter le débit d'entrée à Achères à 25 m^3/s, sachant que les débits dans SDA, CAA et CAB ne sont pas régulés.

La consigne de débit est supposée la même pour le SAR et le SAN :



L'algorithme de pilotage peut s'écrire ainsi :

```
! régulation des vannes du score
! qregul =( 25 – QSDA – QCAB – QCAA) /2 pour chaque vanne du SAR et du SAN
SET    %QCONS =   25  ! débit de consigne
SET    %QSDA =  QAM  NOD-SDA
SET    %QCAA =  QAM  NOD-CAA
SET    %QCAB =  QAM  NOD-CAB
CALCULN     %Qregul  =    (25. - %QSDA – %QCAB – %QCAA) /2

RGB    'VAN-SAR-3'
IF    ( TIME  GT  -999.  )    THEN    ( QREGUL  %Qregul )
RGB    'VAN-SAN-2'
```

```
IF   ( TIME  GT  -999. )   THEN   ( QREGUL  %Qregul )
! fin de l'algorithme
```

5.6 EXEMPLE 6

Utilisation de variable tableaux : fermeture d'une vanne en cas de dépassement de volume de remplissage de bassin

```
! définition de la courbe de remplissage
SET   %%1  =  10   12       14        16        ! cote bassin
SET   %%2  =   0   10000    15000     30000     ! volume bassin
SET   %1  =  Z  NOD-BAS1
! calcul du volume de remplissage correspondant à la cote %1
CALCUL   %2  =  NTERP  (  %1   %1%1   %%2   )
! fermeture de la vanne VAL1_VAN3 si volume de remplissage supérieur à 20000 m3
VA   'VAN3'
IF   ( %2  GT  20000 )   THEN   ( H   0  )
```

5.7 EXEMPLE 7

Utilisation des instructions IF THEN END

```
IF ( TIME LT 3.0 ) THEN
  RGQ  'VALANTOB'
  QREGUL 0.5
ENDIF
IF ( TIME GT 3.0 ) THEN
  IF ( Z FRES57 LT 38.5 ) THEN
    RGQ  'VALANTOB'
    QREGUL 0.5
  ENDIF
  IF ( Z FRE57 GT 40.0 ) THEN
    RGQ  'VALANTOB'
    QREGUL 4.0
  ENDIF

ENDIF
```

5.8 EXEMPLE 8 :UTILISATION DE L'INSTRUCTION DO… ENDDO

```
data &&Vme =  V_HMZ11  V_HMZ14  V_HMZ21  V_HMZ24  V_HMZ31  V_HMZ34
data %qtot = 25.

do %i = 1  6
    s_Rgate  '&&Vme[%i]'
    h  0.

enddo

IF ( %QTot LT 9.0 ) then
   set %Ndecm = 2
elseif ( %QTot LT 11.0 ) then
   set %Ndecm = 3
elseif ( %QTot LT 14.0 ) then
   set %Ndecm = 4
elseif ( %QTot LT 15.0 ) then
   set %Ndecm = 5
else
   set %Ndecm = 6
endif

calculn %q = %qmemb/%Ndecm

do %i = 1  %Ndecm
   s_Rgate  '&&Vme[%i]'
   Qregul  %q

enddo
```

5.9 EXEMPLE 9 : MANIPULATION DE VARIABLES TABLEAUX

```
set %%tab1 = 1 2 4 8 10 12
set %%tab2 = 11 12 14 18 20 22

set %%tinc = 1 2 3 4 5 6

calcul %%tab3 = %%tab1 + %%tab2

show %%tab3

do %i = 1 6
    set %var = %%tab1[%i]
    calcul %%tab1[%i] = %%tab1[%i] +%%tinc[%i]
enddo

show %%tab1

set %%tab1 = 10 11 12
show %%tab1

set %%tab1[2] = 1
show %%tab1

data %%tab4 = 2 2 2
calcul %%tab1 = %%tab1 * %%tab4
show %%tab1
```

On obtient les résultats suivants :

```
command SHOW :
temps :        0.429    TAB3
                         1        12.000
                         2        14.000
                         3        18.000
                         4        26.000
                         5        30.000
                         6        34.000
temps :        0.429    TAB1
                         1         2.000
                         2         4.000
                         3         7.000
                         4        12.000
                         5        15.000
                         6        18.000
temps :        0.429    TAB1
                         1        10.000
                         2        11.000
                         3        12.000
temps :        0.429    TAB1
                         1        10.000
                         2         1.000
                         3        12.000
temps :        0.429    TAB1
                         1        20.000
                         2         2.000
                         3        24.000
```

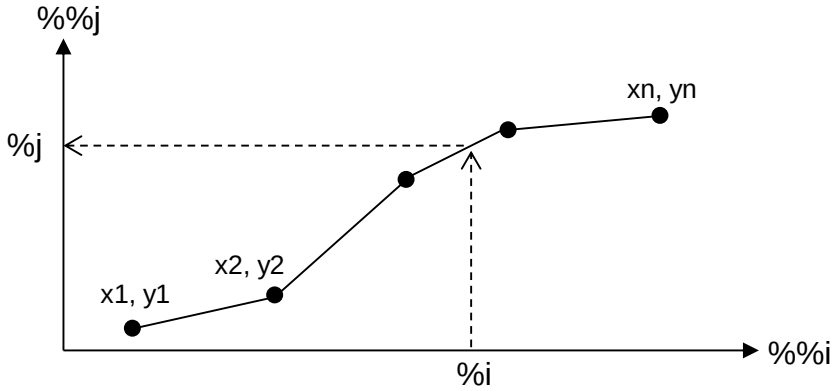
6 ANNEXE 1 : FONCTIONS DISPONIBLES DANS LA BIBLIOTHEQUE DU MODULE DE CONTROLE

6.1 MOT CLÉ : NTERP

Définition : interpolation linéaire

Syntaxe : CALCUL %varj = #NTERP (%%tabi %%tabj var)

La variable %j est calculée par interpolation linéaire :



Commentaires :

les tableaux %%tabi et %%tabj doivent avoir la même dimension.

les valeurs du tableau %%tabi doivent être croissantes.

Les formats suivants sont reconnus par **var** : %vari %tab[3] tab[%i] valeur-numérique

6.2 MOT CLÉ : DATEVALJ

Définition : calcul de la date exprimée en nombre de jours écoulés depuis le 1er janvier.

Syntaxe : CALCUL %varj = #DATEVALJ (%%tabi var)

%%tabi est un tableau de 5 valeurs définissant une date calendaire :

%%tabi(1) : année

%%tabi(2) : mois

%%tabi(3) : jour

%%tabi(4) : heure

%%tabi(5) : minute

Var est un temps exprimé en jours.

%varj est un temps exprimé en jours, égal à la date %%tabi décalée du temps **var** modulo 365.

Les formats suivants sont reconnus par **var** : %vari %tab[3] tab[%i] valeur-numérique

Exemple : si %%tabi = (2007 12 25 00 00) et %vari = 10 alors la valeur retournée dans %varj est égale à : 4.

6.3 MOT CLÉ : MIN

Définition : calcul de la valeur minimum d'une liste de valeurs

Syntaxe : CALCUL %varj = #MIN (var1 var2 ..... varn)

n doit au plus égal à 10.

Les formats suivants sont reconnus par **vari** : %vari %tab[3] tab[%i] valeur-numérique

6.4 MOT CLÉ : MAX

Définition : calcul de la valeur maximum d'une liste de valeurs

Syntaxe : CALCUL %varj = #MAX (%var1 %var2 ..... %varn)

n doit au plus égal à 10.

Les formats suivants sont reconnus par **vari** : %vari %tab[3] tab[%i] valeur-numérique

6.5 MOT CLÉ : MOY

Définition : calcul de la valeur moyenne d'une liste de valeurs

Syntaxe : CALCUL %varj = #MOY (%var1 %var2 ..... %varn)

n doit au plus égal à 10.

M Les formats suivants sont reconnus par **vari** : %vari %tab [3] tab[%i] valeur-numérique

6.6 MOT CLÉ : ABS

Définition : calcul de la valeur absolue d'une grandeur

Syntaxe : CALCUL %varj = #ABS (%var1)

6.7 MOT CLÉ : LEC_ DATE0

Définition : lecture de la date origine de la simulation hydraulique.

Syntaxe : CALCUL %%tabi = #LEC_DATE0 ( )

Tabi est une variable tableau de rang 5. l'utilisation de cette fonction suppose que le mot clé *TIME soit déclaré dans le fichier X.CMD. le programme lit les paramètres de la ligne inscrite sous ce mot clé est les stocke dans le tableau %%tabi :

%%tabi(1) : année

%%tabi(2) : mois

%%tabi(3) : jour

%%tabi(4) : heure

%%tabi(5) : minute

6.8 MOT CLÉ : EXTRACT

Cette commande est devenue obsolète avec les nouvelles possibilités d'affectation offertes par la commande SET. On peut en effet écrire :

SET %val =%%tab [%i]. Elle n'est plus disponible.

6.9 MOT CLÉ: PID

Définition : calcul de régulation en mode PID.

Syntaxe :

CALCUL %dQ = #PID (%Z %Zcons %Time %Time1 %e1 %ecum1 %k1 %k2 %k3)

Les variables %Zcons, %k1, %k2, %k3 peuvent être des valeurs numériques.

la variable dQ est calculée comme suit :

e=z-zcons

de=e-e1

dt= (time – time1)*3600.

ecum=ecum1+e*dt

dQ= k1*e +k2*ecum +k3*de/dt

Cette fonction actualise également les variables %Time1 %e1 %ecum1 :

%Time1 = Time

%e1 = e

%ecum1 = ecum

6.10 MOT CLÉ : GET_EXT

Définition :lecture de la valeur x(t) d'une courbe contenue dans le fichier déclaré par la méta commande READ_EXT

Syntaxe :

CALCUL %x = GET_EX T(%timeh , cc)

%timeh : varaiable désignant le temps relatif en heures / date0

Cc: identificateur de courbe défini dans le fchier déclaré par la meta commande READ_EXT.

Les formats suivants sont reconnus pour **Cc** : &ci &&ctab [3] &&ctab[%i] chaine-de-caractères

(24 caractères max).

Exemple :

```
$ READ_EXT
{nom-de-fichier}
$CASE(reseau)
------------------
Data &&ctab  = c1  c2  c3
Do %i = 1  3
  Set %timeh = TIME
  Calcul  %x = #get_ext ( %timeh  &&ctab[%i] )
 -----------------------------
Enddo
```

6.11 MOT CLÉ : IRSTART

Définition : detection du mode de calcul

Syntaxe :

CALCUL %istrart = #IRSTART()

La fonction retourne la valeur :

0 : si le calcul est effectué avec démarrage à froid

1 : si le calcul est effectué avec démarrage à chaud

6.12 MOT CLÉ : LEC_CDATE0

Définition : lecture de la date origine de la simulation hydraulique.

Syntaxe : CALCUL &date = #LEC_DATE0 ( )

La variable caractère &cdate contient la date de début de la simulation sous la forme :

aaaammjr:hhmn

6.13 MOT CLÉ : DIFF_DATE

Définition : calcul de l'intervalle de temps en heures séparant deux variables de type « date » :

Syntaxe : CALCUL %diff_date = #DIFF_DATE (&cdate1 &cdate2)

Les deux variables cdate1 et cdate2 contiennet obligatoirement des chaines de caractéres au formatt "date" : aaaammjr:hhmn

6.14 MOT CLÉ : ADD_HR_TO_DATE

Définition : calcul d'une nouvelle date calculée en additionnant un nombres d'heures à une date donnée :

Syntaxe : CALCUL &cdate2 = #ADD_hr_TO_DATE(&cdate1 %dth)

Exemples :

```
calcul &cdate = #lec_cdate0( )
show &cdate

set %dth = 24.
calcul &cdate1 = #add_hr_to_date (&cdate %dth)
show &cdate1

calcul %dth1 = #diff_date (&cdate  &cdate1)
show  %dth1
```

Le fichier .out produit les résultats suivants à chaque pas de temps de scrutation :

```
temps :        0.802    CDATE  20230210:0210
temps :        0.802    CDATE1  20230211:0210
temps :        0.802    DTH1        24.000
temps :        0.802    YES  YES
```

6.15 MOT CLÉ : INSTANCIATION

Définition : instanciation d'une variable à partir de données définies dans un tableau de chaîne de caractères et d'un tableau de valeurs numériques.

Syntaxe : CALCUL %valx = #INSTANCIATION (&&ctab %%val)

Les deux tableux ont la même dimension Ndim.

. Le programme opére une boucle sur le tableau &&ctab et recherche lindice i tel que :

&&ctab(i) = "valx"

Il instancie alors la variable %valx comme suit :

%valx = %%val(i)

Cette fonction a été développé pour le projet GAO.

Exemple :

```
$ case (model1)

debug  0. 5.0  0.5

! test de la fonction instanciation

data sequence
&&pomp          %%qpomp
P1                1.
P2                2.
P3                3.

call subtest( &&pomp  %%qpomp )


subroutine subtest(&&pilotage[10] %%pilotage[10])

calcul %P1 = #instanciation  ( &&pilotage  %%pilotage)
show %p1

calcul %P2 = #instanciation  ( &&pilotage  %%pilotage)
show %p2

calcul %P3 = #instanciation  ( &&pilotage  %%pilotage)
show %p3

endsubr  subtest
```

Le programme affiche les résultats suivants à chaque pas de temps de scrutation :

```
command SHOW :
temps :       0.784    P1        1.000
temps :       0.784    P2        2.000
temps :       0.784    P3        3.000
```

6.16 MOT CLE : DIM

Définition : lecture de la dmiension d'un tableau %% ou &&

Syntaxe : CALCUL %dim = #DIM (%%val)

La varible %dim récupère la dimension du tableau %%val.

Cette fonction est utile dans un sous programme et dans une boucle DO pour traiter la totalité des élements d'un tableau.

Exemple :

```
Subroutine sub1 (%%tabin  %%tabout)
Calcil %dim = #DIMn(%%tabin)
do %k = 1 %dim
Calcul %%tabout[%k] = %%tabin[%k] +%k
enddo
```

6.17 MO CLÉ : LEC_CDATE

Définition : conversion d'un tableau de valeurs numériques en variable caractère de type « date »

Syntaxe : CALCUL &date = #LEC_CDATE (%%Tab)

%%Tab : tableau de 5 valeurs définies comme suit :

%%tabi(1) : année

%%tabi(2) : mois

%%tabi(3) : jour

%%tabi(4) : heure

%%tabi(5) : minute

La variable caractère &cdate contient la date de début de la simulation sous la forme :

aaaammjr:hhmn

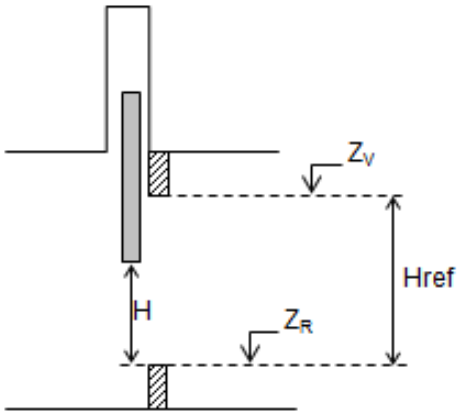
7 ANNEXE 2 : PARAMETRES MODIFIABLES POUR CHAQUE ACTIONNEUR

7.1 S_GATE

Vanne d'exploitation le long d'un collecteur. Généralement on impose une condition d'ouverture ou de fermeture totale.

Mode de pilotage	*Manuel*		*Régulation*
Paramètre modifiable	Zseuil	H	/
Valeur minimale	ZR	0	/
Valeur maximale	Zv	Href	/
Mot clé PARAM	Zs	H	/
Réglage par défaut	H = Href		

Href est calculée à partir de des cotes radier et voûte de la vanne définies dans le modèle de référence : Href = Zv - Z_R



Le réglage par défaut peut être réactivé à tout moment par l'instruction :

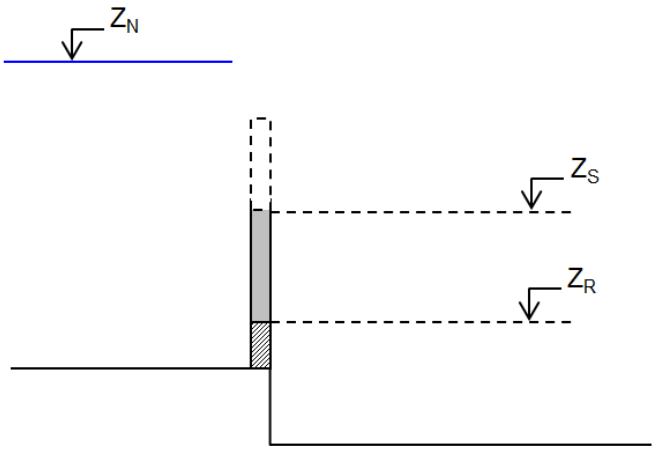
(H ''DEFAUT'')

7.2 S_RWEIR

Cette singularité s'utilise surtout dans le domaine fluvial pour la régulation d'un plan d'eau en amont de l'ouvrage.

Mode de pilotage	*Manuel*			*Régulation*
Paramètre modifiable	Z_S		Largeur seuil	Z_N
Valeur minimale	Z_R		0	Pas de limite
Valeur maximale	Pas de limite		Pas de limite	Pas de limite
Mot clé PARAM	ZS		BS	ZREGUL
Réglage par défaut		Régulation du plan d'eau à $Z_N$ref		

$Z_N$ref est la valeur de Z_N définie dans le modèle de référence.



Le réglage par défaut peut être réactivé à tout moment par l'instruction :

(ZREGUL ''DEFAUT'')

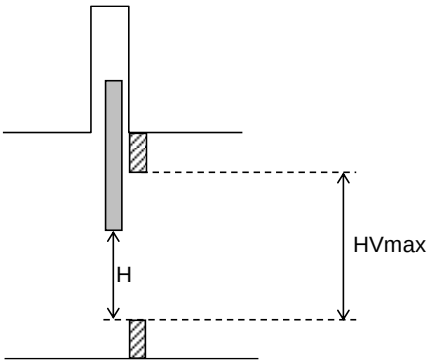
7.3 S_RGATE

Vanne plate en ligne le long d'un collecteur régulable en débit ou en cote

Mode de pilotage	*Manuel*			*Régulation*	
Paramètre modifiable	Z seuil	H		Z_C = cote consigne de régulation	Q_C = débit de consigne au travers de la vanne
Valeur minimale		0		Pas de limite	Pas de limite
Valeur maximale	Zmax	HVmax		Pas de limite	Pas de limite
Mot clé PARAM	Zs	H		ZREGUL	QREGUL
Réglage par défaut				Régulation en cote ou en débit comme défini dans le modèle de référence (le « ou » est exclusif)	

3 modes de régulation possibles :

- IREGUL=1 : Débit de consigne QREGUL
- IREGUL=2 : Cote de consigne ZREGUL en un point donné, défini dans l'élément
- IREGUL=0 : Position fixe de vanne (mode manuel)



Le changement de mode par le fichier de contrôle est soumis aux restrictions suivantes :

Mode initial	Changement de Mode via le fichier de contrôle		
	0	1	2
0	-	interdit	interdit
1 : ZREGUL	autorisé	-	autorisé
2 : QREGUL	autorisé	interdit	interdit

en mode régulation de débit, la vanne s'efface en cas de dépassement de la cote critique du niveau d'eau en amont de la vanne : la consigne de débit n'est plus suivie.

7.4 L_PUMP_DERIV

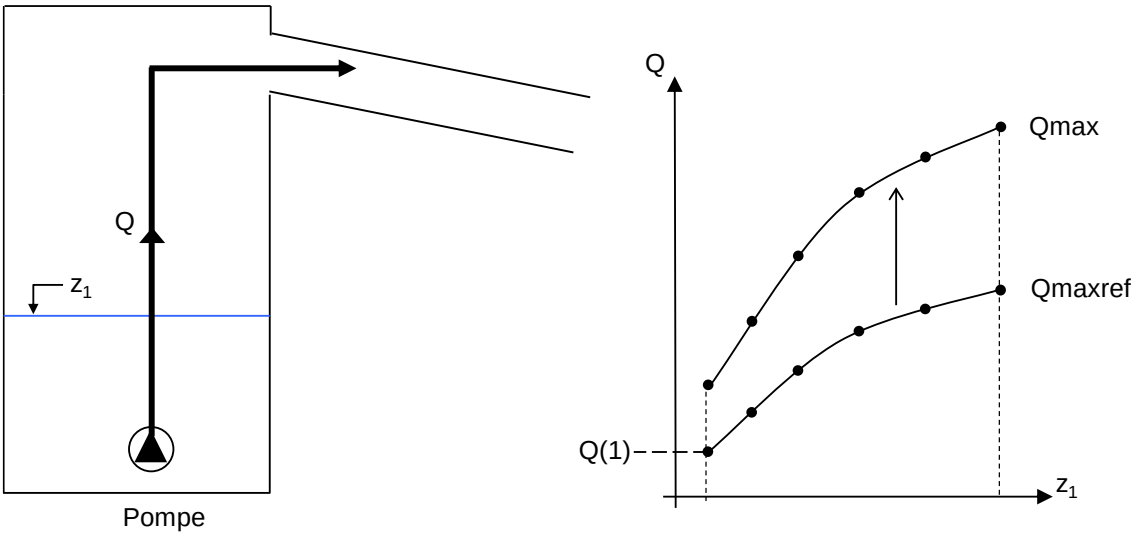
Cette singularité est une liaison latérale bi nodale de type dérivation : le débit pompé n'est fonction que du niveau d'eau dans la bâche. Elle est généralement utilisée avec une vanne contrôlant l'entrée d'eau dans la bâche.

Mode de pilotage	*Manuel*		*Régulation*
Paramètre modifiable	Qmax	Zd	/
Valeur minimale	0	Pas de limite	/
Valeur maximale	Pas de limite	Pas de limite	/
Mot clé PARAM	QPOMP	ZPOMP	/
Réglage par défaut	Qmax = Qmaxref ou Zd = Zdref		

<u>Qmax est le débit maximum de la pompe.</u>

Qmaxref est calculé à partir de la courbe de pompage α(z) définie dans le modèle de référence. Qmaxref = Qini* α(zNP) avec NP : nombre de points définissant la courbe α (z).

Lorsque le mot clé QPOMP est spécifié, la nouvelle courbe de pompage est déduite de celle du modèle de référence par homothétie de rapport Qmax / Qmaxref.



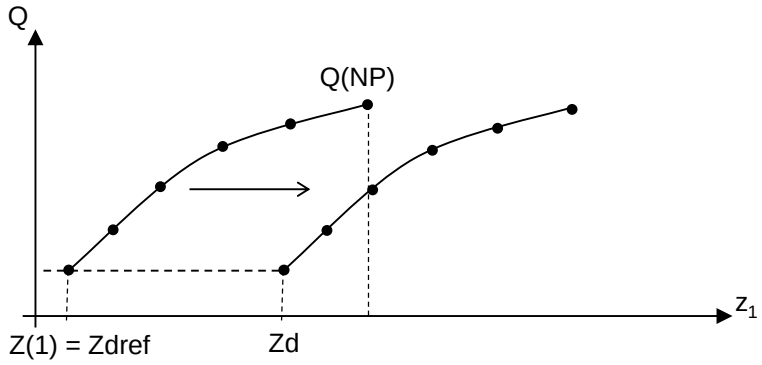
Le réglage par défaut peut être réactivé à tout moment par l'instruction :

(QPOMP ''DEFAUT'')

<u>Zdref est la cote démarrage de la pompe.</u>

Zdref est calculée à partir de la courbe de pompage α(z) définie dans le modèle de référence. Zdref = Z1 (premier point de la courbe).

Lorsque le mot clé ZPOMP est spécifié la nouvelle courbe de pompage est déduite de celle du modèle de référence par translation horizontale de valeur Zd - Zdref.



Le réglage par défaut peut être réactivé à tout moment par l'instruction :

(ZPOMP ''DEFAUT'')

QREGULP est un débit de consigné de régulation, compris entre 0. et Qmax.

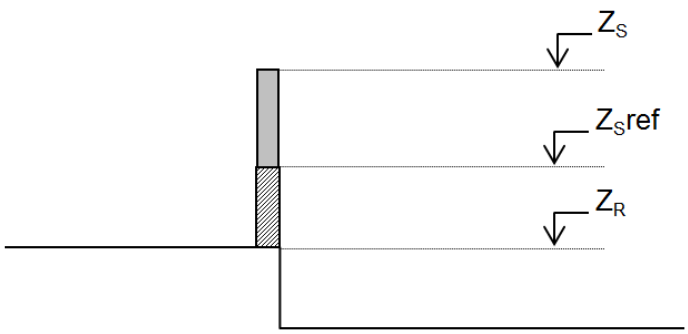
La pompe s'ajuste immédiatement à ce débit de consigne, sans délai de rattrapage.

7.5 L_WEIR

Cette singularité est une liaison latérale bi nodale de type maillage : le débit échangé est fonction des niveaux d'eau aux nœuds amont et aval.

Mode de pilotage	*Manuel*		
Paramètre modifiable	Z_S		Hauteur seuil/radier
Valeur minimale	Z_R		0
Valeur maximale	Pas de limite		Pas de limite
Mot clé PARAM	ZS		H
Réglage par défaut	Zs = $Z_S$ref		

$Z_S$ref est la valeur de Z_S définie dans le modèle de référence.



Le réglage par défaut peut être réactivé à tout moment par l'instruction :

(ZS ''DEFAUT'')

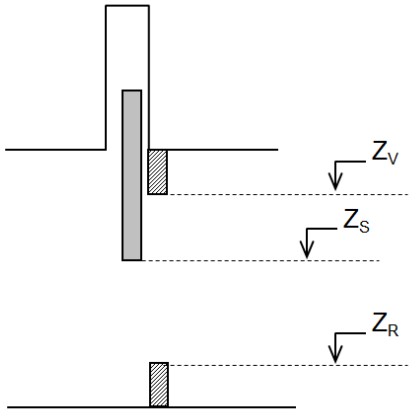
7.6 L_GATE

Singularité la plus répandue dans les ouvrages d'assainissement. Elle est utilisée comme maillage d'exploitation.

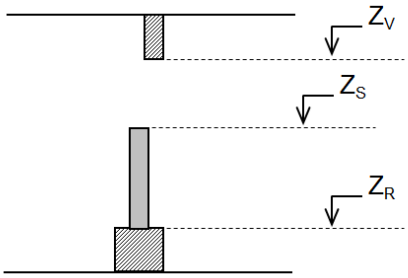
C'est une liaison latérale bi nodale de type maillage, comme la singularité LDEV.

Mode de pilotage	*Manuel*		
Paramètre modifiable	Z_S		Hauteur seuil/radier
Valeur minimale	Z_R		0
Valeur maximale	Z_V		Pas de limite
Mot clé PARAM	ZS		H
Réglage par défaut	$Z_S = Z_S$ref		

$Z_S$ref est la valeur de Z_S définie dans le modèle de référence.



Vanne à effacement par le haut



Vanne à effacement par le bas

Le réglage par défaut peut être réactivé à tout moment par l'instruction :

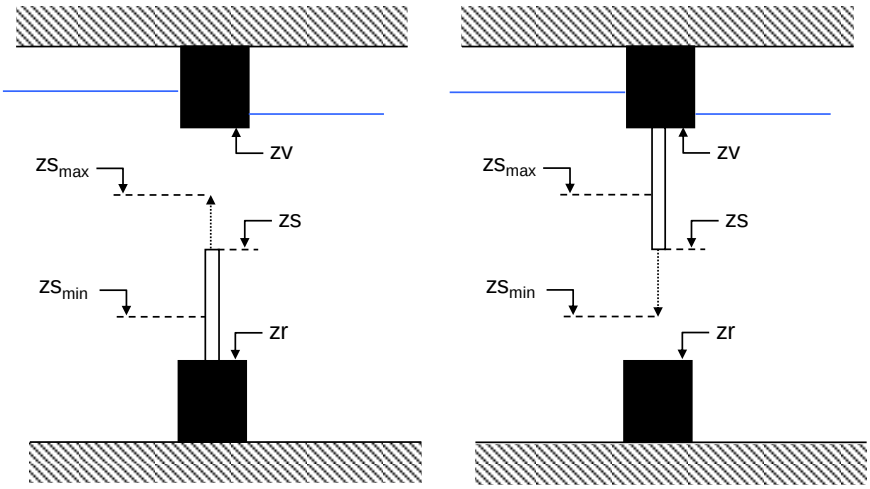
(ZS ''DEFAUT'')

7.7 L_RGATE

C'est le mode de pilotage le plus répandu pour les vannes mobiles.

Mode de pilotage	*Manuel*			*Régulation*	
Paramètre modifiable	Z seuil	H		Z_C = cote consigne de régulation	Q_C = débit de consigne au travers de la vanne
Valeur minimale		0		Pas de limite	Pas de limite
Valeur maximale	Zmax	HVmax		Pas de limite	Pas de limite
Mot clé PARAM	Zs	H		ZREGUL	QREGUL
Réglage par défaut				Régulation en cote ou en débit comme défini dans le modèle de référence (le « ou » est exclusif)	

Z_{Smin} et Z_{Smax} sont définies dans le modèle de référence.



Vanne à effacement par le bas

Vanne à effacement par le haut

Le réglage par défaut est le réglage défini dans le modèle de référence. Il peut être réactivé à tout moment par l'instruction :

(ZREGUL ''DEFAUT'')

Mode initial	Changement de Mode via le fichier de contrôle		
	0	1	2
0	-	interdit	interdit
1 : ZREGUL	autorisé	-	autorisé
2 : QREGUL	autorisé	interdit	interdit

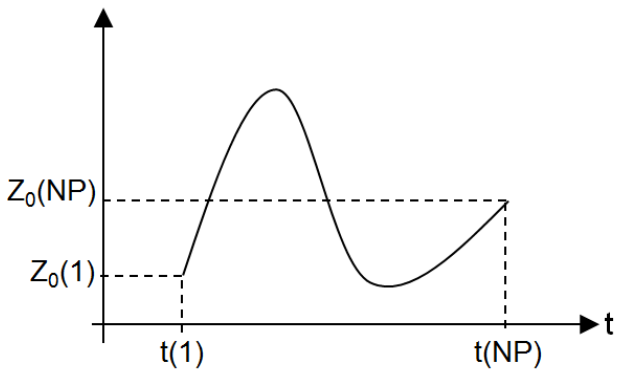
Note : en mode régulation de débit, si la cote critique est atteinte en amont immédiat, la vanne s'efface pour laisser passer du débit et tenter ainsi de faire redescendre le niveau d'eau : la consigne de débit n'est plus suivie. . Cette disposition a été étendue pour le mode régulation en cote, afin de sécuriser le pilotage en cas de contrôle de niveau d'eau en aval de la vanne.

7.8 CL_ZT

Cette singularité est de type liaison uninodale. Elle est utilisée pour spécifier un niveau d'eau imposé de cours d'eau à l'exutoire d'un ouvrage en contact avec le milieu récepteur.

Mode de pilotage	*Manuel*	*Régulation*
Paramètre modifiable	Z_0	/
Valeur minimale	Pas de limite	/
Valeur maximale	Pas de limite	/
Mot clé PARAM	Z0	/
Réglage par défaut	$Z_0 = Z_0$ref	

Z_0ref est la valeur de Z_0 définie dans le modèle de référence par la courbe $Z_0(t)$.



Le réglage par défaut peut être réactivé à tout moment par l'instruction :

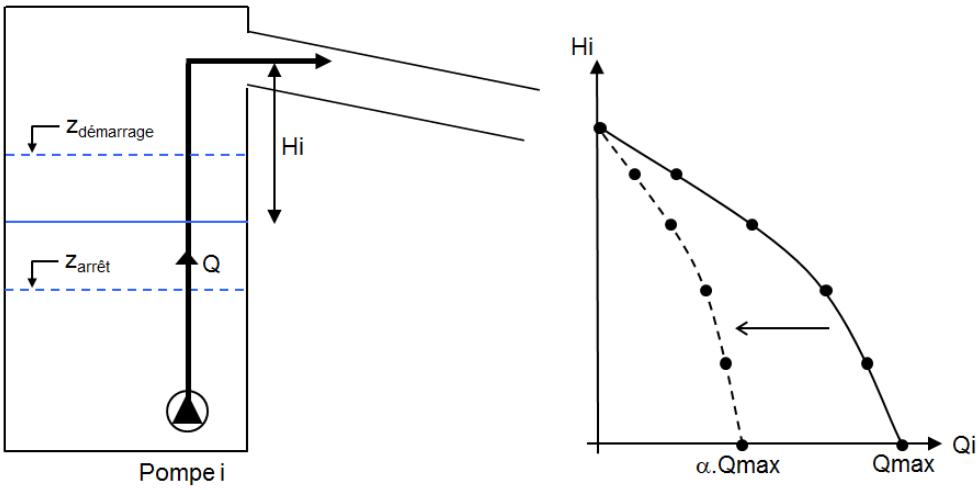
(Z0 ''DEFAUT'')

7.9 L_PUMP

Cette singularité est une liaison latérale de type maillage : le débit pompé dépend de la charge hydraulique entre le point d'aspiration et le point de relèvement.

Le pompage est modélisé par un groupe de pompes en parallèle, définie chacune par :

- un niveau haut de début de pompage,
- un niveau bas de fin de pompage,
- une courbe caractéristique de pompage Qi (H), où H est la charge hydraulique.



Les paramètres ajustables par le fichier de contrôle sont :

- FQPOMP : fraction du débit nominal pompé par le groupe de pompage.
- DZPOMP: translation des cotes de démarrage/arrêt de chaque pompe.
- ZPSTART : cote de démarrage des pompes (niveau haut)
- ZPSTOP : cote d'arrêt des pompes (niveau bas)
- QREGULP : débit total de régulation du groupe de pompage.

Ces deux paramètres s'appliquent simultanément à toutes les pompes.

Mode de pilotage	*Manuel*		*Régulation*
Paramètre modifiable	Facteur de réduction du pompage	Translation niveaux marche/arrêt pompes	Débit de régulation de la pompe
Valeur minimale	0	0	0
Valeur maximale	Pas de limite	Pas de limite	Pas de limite
Mot clé PARAM	FQPOMP	DZPOMP	QREGULP

Réglage par défaut	1	0	/

Mode de pilotage	*Manuel*		
Paramètre modifiable	cote de démarrage du pompage	cote de démarrage du pompage	
Valeur minimale	0	0	
Valeur maximale	Pas de limite	Pas de limite	
Mot clé PARAM	ZPSTART	ZPSTOP	
Réglage par défaut	1	0	

Remarque :

La commande QREGULP est utilisée pour réguler un niveau d'eau dans la bâche amont de la pompe. Il est généralement calculé à l'aide du sous-programme « Sregul_Pompe » listé au §4.3 dans lequel est codé un algorithme de régulation PID.

7.10 CL_Q0

Cette singularité est de type liaison uninodale. Elle correspond à l'objet « constant inflow ». Elle est utilisée pour spécifier un débit d'injection

Mode de pilotage	*Manuel*	*Régulation*
Paramètre modifiable	Q0	/
Valeur minimale	Pas de limite	/
Valeur maximale	Pas de limite	/
Mot clé PARAM	Q0	/
Réglage par défaut	Q = Q0_ref	

Q0ref est la valeur de Q0 définie dans le modèle de référence.

Le réglage par défaut peut être réactivé à tout moment par l'instruction :

(Q0 ''DEFAUT'')

7.11 CASIER -STORAGE

On peut imposer une cote dans un casier via le fichier de contrôle :

Mode de pilotage	*Manuel*	*Régulation*
Paramètre modifiable	Cote initiale	/
Valeur minimale	Pas de limite	/
Valeur maximale	Pas de limite	/
Mot clé PARAM	ZINI	/
Réglage par défaut	Zini = Zini_ref	

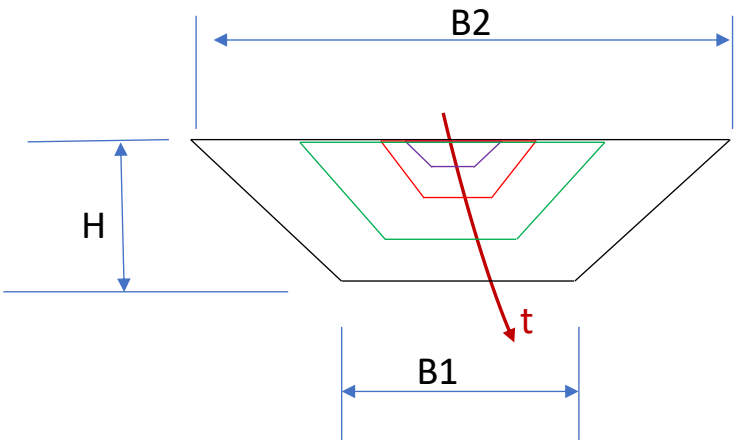
Zini_Ref est la valeur définie dans l'élément.

Commentaire : la cote est actualisée dans le casier à chaque fois que cette instruction est rencontrée. Si on souhaite seulement initialiser cette cote en débit de calcul il faut programmer le fichier de contrôle en conséquence.

7.12 LSTK

le paramètre modifiable est le coefficient alp : facteur de réduction de la géométrie de la liaison.

Cette fonctionnalité a été ajoutée pour simuler une brèche de forme trapézoïdale à travers un cordon dunaire. Le modèle d'ouverture de brèche est le suivant :



La géométrie de la liaison totalement ouverte est définie par les trois paramètres (B1, B2, H).

Le coefficient « alp » permet réduire la géométrie (B'1, B'2, H') de l'ouverture comme suit :

H' = alp x H

B'1 = alp x B1

B'2 = alp x B2.

Mode de pilotage	*Manuel*	*Régulation*
Paramètre modifiable	alp	/
Valeur minimale	0.	/
Valeur maximale	1.	/
Mot clé PARAM	ALP	/
Réglage par défaut	Alp=1.	

Par défaut alp =1.

Si on souhaite modéliser l'ouverture progressive d'une brèche qui commence à s'ouvrir au temps t0, il faut définir les instructions suivantes dans le fichier de contrôle :

LSTK ' cid'

IF (TIME LT *t0*) then (ALP 0.)

IF (TIME LT t0) and (TIME LT t1) then (ALP *alp1*)

IF (TIME LT t1) and (TIME LT t2) then (ALP *alp2*)

IF (TIME LT tn-1) and (TIME LT tn) then (ALP *alpn)*

IF (TIME GT *tn*) then (ALP *1.)*

Les paramètres (cid, t1----- tn, alp1---------alpn) sont des valeurs numériques définies par l'utilisateur pour décrire la loi temporelle d'ouverture de la brèche.

Les temps sont exprimés en heures relativement à la date origine de la simulation.

7.13 CL_DLAT

Cette singularité est de type liaison uninodale. Elle est utilisée pour spécifier une nouvelle cote de seuil.

Mode de pilotage	*Manuel*	*Régulation*
Paramètre modifiable	zs	/
Valeur minimale	zs0	/
Valeur maximale	Pas de limite	/
Mot clé PARAM	zs	/
Réglage par défaut	zs = zs0	