RST Digital Controller (C^{++})
Régulateur numérique RST(programme)

The C++ program for the implementation of RST digital controllers is presented. These programs have been written by Adaptech [http://www.adaptech.com] and they can be freely used provided that the author (Adaptech) is mentionned. Note that WinREG softaware (Adaptech) automatically generates the C^{++} code for the designed controller

On donne dans ce qui suit une présentation des programmes de base pour la mise en œuvre des régulateurs R-S-T. Ces programmes élaborés par Adaptech [Adaptech (2001a)] sont dans le domaine public et peuvent être utilisés librement (sous réserve de la mention de l'auteur). A noter que le logiciel WinREG engendre automatiquement le programme en C^{++} du régulateur calculé.

A.8.1 Programme en C^{++} pour le régulateur R-S-T (exemple)

```
//===============================================
// rst.H
// Description : Algorithme du Régulateur RST
// Auteur: ADAPTECH
// Création du fichier : 14/02/2000
//===============================================

//-----------------------------------------------------------------------------------------------
// Fonction : init_stockage
// Cette fonction doit être appelée une fois en début de
// programme afin d'initialiser les vecteurs de stockage et
// composante continue du régulateur
// Variables d'entrée : yinit = valeur initiale de la mesure
//                      uinit = valeur initiale de la commande
```

```
//                          cinit = valeur initiale de la consigne
//-------------------------------------------------------------------------------------------

void init_stockage ( float yinit, float uinit, float cinit);

//-------------------------------------------------------------------------------------------
// Fonction : execute_rst
// Cette fonction doit être appelée à chaque période d'échantillonnage Te
// Variables d'entrée : consigne = consigne à l'instant t
//                          mesure = sortie du procédé à l'instant t
// Variable de sortie : commande à appliquer à l'instant t
//-------------------------------------------------------------------------------------------

float execute_rst (float consigne, float mesure);

//=============================================
// Dans le cas où il est nécessaire de minimiser le temps entre
// l'acquisition de la mesure et l'envoi de la commande, on peut
// utiliser la décomposition en 3 fonctions présentée ci-dessous:
//=============================================

void precalcul_rst (void);

float execalcul_rst (float consigne, float mesure);

//=============================================
// Variables d'entrée : consigne = consigne à l'instant t
//                          mesure = sortie du procédé à l'instant t
// Variable de sortie : commande à appliquer à l'instant t
//=============================================

void postcalcul_rst(float commande);

//=============================================
// Variable d'entrée : commande appliquée à l'instant t
// A chaque période d'échantillonage Te, on doit donc réaliser
// l'enchaînement présenté ci-après :
//          ...
//       precalcul_rst();
//          ... Acquisition de la mesure (et calcul de la consigne)
```

```
//      execalcul_rst(consigne, mesure);
//         ... Envoi de la commande
//      postcalcul_rst(commande);
//         ...
//================================================

//================================================
// rst.c
// Description : Algorithme du Régulateur RST
// Equations de base:   yrt = Bm(q-1)*ur(t) - Am(q-1)*yrt(t-1)
//                   S(q-1)*u(t) = T(q-1 )*yr(t) - R(q-1)*y(t)
// Auteur : ADAPTECH
// Création du fichier : 14/02/2000
//================================================

#include "rst.H"

#define Rdeg 2
float R[Rdeg+1] = {   1.332245,  -1.905430,   0.781800 };

#define Sdeg 3
float S[Sdeg+1] = {   1.000000,   0.511808,  -0.904179,  -0.607629 };

#define Tdeg 2
float T[Tdeg+1] = {   0.666840,  -0.659073,   0.200848 };

#define Amdeg 2
float Am[Amdeg+1] = { 1.,  -0.988352,   0.301194 };

#define Bmdeg 1
float Bm[Bmdeg+1] = { 0.187539,   0.125304 };

// Vecteurs utilisés pour le stockage des signaux :
// L'indice i correspond au temps t-i, t étant l'instant actuel
// par exemple, ut[1] représente u(t-1), yt[3] représente y(t-3)

float yt[Rdeg+1], ut[Sdeg+1], yreft[Tdeg+1];
float yrt[Amdeg+1], urt[Bmdeg+1];

float compo_continu;      // Composante continue du régulateur
```

```
float prepa_cmde;           // Valeur intermédiaire pour le calcul

//-----------------------------------------------------------------------------------------------
// Fonction: init_stockage
// appelée une fois en début de programme afin d'initialiser
// les vecteurs de stockage et la composante continue du régulateur
//-----------------------------------------------------------------------------------------------

void init_stockage ( float yinit, float uinit, float cinit)
{
    int j;
    float sigmaR, sigmaT, sigmaS;

    sigmaR = 0.;
    for (j = 0; j <= Rdeg; j++)
    {
        yt[j] = yinit;
        sigmaR += R[j];
    }
    sigmaS = 0.;
    for (j = 0; j <= Sdeg; j++)
    {
        ut[j] = uinit;
        sigmaS += S[j];
    }
    sigmaT = 0.;
    for (j = 0; j <= Tdeg; j++)
    {
        yreft[j] = cinit;
        sigmaT += T[j];
    }

// Calcul de la composante continue à prendre en compte dans la sommation,
// avant la division par S0:
compo_continu = (sigmaR-sigmaT) * yinit + sigmaS * uinit;

//-----------------------------------------------------------------------------------------------
// Fonction execute_rst
// appelée à chaque période d'échantillonnage
```

```
//-------------------------------------------------------------------------------------------

float execute_rst (float consigne, float mesure)
{
    float commande;
    float yref;    // Sortie du modèle de référence de poursuite

    precalcul_rst();

    yref = calc_ref(consigne);
    commande = calc_cmde(yref, mesure);

    decal_trajref(consigne, yref);
    decal_obser(yref, mesure, commande);

    return commande;
}
//==== VARIANTE en 3 PHASES optimisant les temps de traitement ====
//  - précalcul_rst
//  - execalcul_rst =>  /calc_ref       /calc_cmde
//  - poscalcul_rst =>  /decal_trajref  /decal_obser
//=================================================

//-------------------------------------------------------------------------------------------
// Fonction: precalcul_rst
// appelée à chaque période d'échantillonnage
//-------------------------------------------------------------------------------------------

void precalcul_rst (void)
{
    float somme;
    int             j;

    somme = compo_continu;
    for (j=1; j<=Tdeg; j++)
    {  somme += T[j]*yreft[j]; }
    for (j=1; j<=Rdeg; j++)
    {  somme -= R[j]*yt[j]; }
    for (j=1; j<=Sdeg; j++)
    {  somme -= S[j]*ut[j]; }
```

```
    prepa_cmde = somme;
    return;
}

//-------------------------------------------------------------------------------------------
// Fonction execalcul_rst
// appelée à chaque période d'échantillonnage
// après l'acquisition des mesures et avant l'envoi de la commande
//-------------------------------------------------------------------------------------------

float execalcul_rst (float consigne, float mesure)
{
    float commande;

    yt[0] = mesure;
    urt[0] = consigne;
    yreft[0] = calc_ref(consigne);
    commande = calc_cmde(yreft[0], mesure);
    return commande;
}

//-------------------------------------------------------------------------------------------
// Fonction: calc_ref
// utilisée par la fonction execalcul_rst
//-------------------------------------------------------------------------------------------

float calc_ref (float nvcons)

// Calcul de la sortie du modèle de référence, selon :
//      yrt = Bm . urt - Am* yrt(t-1)
//  ATTENTION : la valeur calculée n'est pas stockée dans yrt[0]

{
    int j;
    float somme;
    somme = Bm[0] * nvcons;
    for (j = 1; j <=Bmdeg; j++)
    {   somme += Bm[j] * urt[j]; }
    for (j = 1; j <=Amdeg; j++)
    {   somme -= Am[j] * yrt[j]; }
```

```
    return(somme);
}

//-------------------------------------------------------------------------------------------
// Fonction: calc_cmde
// utilisée par la fonction execalcul_rst
//-------------------------------------------------------------------------------------------

float calc_cmde (float yref, float mesure)
{
    float somme;

    somme = prepa_cmde;
    somme -= R[0]*mesure;
    somme += T[0]*yref;
    somme = somme / S[0];

    return somme;
}

//-------------------------------------------------------------------------------------------
// Fonction postcalcul_rst
// appelée à chaque période d'échantillonnage
// après l'envoi de la commande
//-------------------------------------------------------------------------------------------

void postcalcul_rst (float commande)
{
    decal_trajref(urt[0], yreft[0]);
    decal_obser(yreft[0], yt[0], commande);
    return;
}

//-------------------------------------------------------------------------------------------
// Fonction: decal_trajref
// utilisée par la fonction postcalcul_rst
//-------------------------------------------------------------------------------------------

void decal_trajref (float consigne, float yref)
{
```

```
    int j;
    for (j=Bmdeg; j>1; j--)
    { urt[j] = urt[j-1]; }
    for (j=Amdeg; j>1; j--)
    { yrt[j] = yrt[j-1]; }

    urt[1] = consigne;
    yrt[1] = yref;
}

//-----------------------------------------------------------------------------------------------
// Fonction: decal_obser
// utilisée par la fonction postcalcul_rst
//-----------------------------------------------------------------------------------------------

void decal_obser(float yref, float mesure, float commande)
{
    int j;
    for (j=Rdeg; j>1; j--)
    { yt[j] = yt[j-1]; }
    for (j=Sdeg; j>1; j--)
    { ut[j] = ut[j-1]; }
    for (j=Tdeg; j>1; j--)

    { yreft[j] = yreft[j-1]; }

    yt[1] = mesure;
    ut[1] = commande;
    yreft[1] = yref;
}

//================ Fin de l'algorithme RST ==================
```

A.8.2. Notes et indications bibliographiques

Plus de détails sur les programmes et leur implémentation effective sur différentes cibles (automates programmables, régulateur programmable, carte microcontroleur, PC superviseur, S.N.C.C.) sont disponibles dans :

Adaptech (2001a) : *Guide d'intégration du régulateur R-S-T sur cibles programmables,* Adaptech, 4 Rue de la Tour de l'Eau, Saint Martin d'Hères, France.