ASSOCIATION POUR LE DÉVELOPPEMENT DE L'ENSEIGNEMENT
ET DE LA RECHERCHE EN SYSTÉMATIQUE APPLIQUÉE

MELISSA

Memorandum of Understanding
ECT/FG/MMM/97.012

CONTRACT ESA-ESTEC/ADERSA
Purchase Order n° 171686 of 16.07.1997

TECHNICAL NOTE : 38.1

Transferability improvement of the Spirulina production control software for the present and future GPS structures

Version : 1
Issue : 0

J.-J. LECLERCQ

August 1999

10, rue de la Croix Martre
91873 PALAISEAU Cedex
Téléphone : (33) 1 60 13 53 53
Télécopie : (33) 1 69 20 05 63
Courriel : adersa@adersa.asso.fr

adersa

Document Change Log

Version	Issue	Date	Observation
0	0	June 1999	Draft
1	0	August 1999	Original version

TABLE OF CONTENTS

Abbreviations :
GPS : Global Purpose Station
DLL : Dynamic Link Library

1. INTRODUCTION

This study is needed by the requirements of the new GPS structure and by the development of the MELISSA project. Indeed, it becomes now necessary to improve the transferability of any control software so that it could be carried from the simulator (where it is build) into the software of the present or future GPS, with the fewest possible modifications. This attitude is applied for the first time to the Spirulina production control and will be generalized to the next control software's.

The following rules are applied for this work :

- the main control function communicates with the GPS software by means of arguments only;
- all the variables of the control function are local variables;
- the parameters values of the control which are modifiable by the user, such as the reference trajectory dynamic, are given in the GPS software;
- the parameters of the control which are under the responsibility of the authors of the first principles model or of the control, are embedded in the control function and are not accessible by mean of arguments.

Thanks to these rules, the software can be compiled :

- by the compiler of the existing GPS;
- or by an adequate compiler able to build a DLL for the new GPS structure.

At the meeting ESA/LGCB/ADERSA on May 12th 1999, it was decided to give an official name to a few software's :

name	software
LSPC	Light Spirulina Production Control
LRPC	Light Rhodobacter Production Control
SBQ	Spirulina Biomass Quality
MCS	Melissa Control Software (Level 2 of the hierarchical control)

So, according to this decision, the main function realizing the control of the Spirulina production by action on the light intensity is called now *lspc* .

The call of the control function *lspc* is done inside the function *control_spiru* of the source file *ctrlspir.c* so that to keep the organisation of the GPS.
The changes brought to the 97 upgraded software are detailed in the following chapters.

2. DESCRIPTION OF THE CONTROL FUNCTION *LSPC*

The function *lspc* realizes, in only one call, the control of the biomass production by action on the light intensity.

The communication of this function with the main programme of the GPS is done by means of arguments, exclusively : all the variables inside the function are local variables and none are global variables common with the main programme.

LSPC (PROD_SP2, CX, QE_SP2, FR, QE_MES, SM_SUP, CONS_SUP, VOL, FI, DT, LAMBDA, INIT, VAR_OUT)

These arguments are described in table 1.

Argument	v/p	Description
PROD_SP2	v	level2 production set point (g/h)
CX	v	biomass concentration (g/l)
QE_SP2	v	level2 flow rate set point (l/h)
FR	p	light intensity : measured or computed by the control (W/m2) . input argument : measured value of FR . output argument : computed by the control
QE_MES	v	measure of flow rate (l/h)
SM_SUP	p	production model output computed by the supervisor (g/h) . input argument : value at previous moment . output argument : value at present moment
CONS_SUP	p	production set point computed by the supervisor (g/h) . input argument : value at previous moment . output argument : value at present moment
VOL	v	volume of the reactor (l)
FI	v	illuminated surface fraction (no dimension)
DT	v	control period (h)
LAMBDA	v	dynamic of the reference trajectory (dimension less)
INIT	p	initialisation flag (when equal to 0) put to 1 by this programme
VAR_OUT[0]	p	level 1 production set point (g/h)
VAR_OUT[1]	p	level 1 flow rate set point (l/h)
VAR_OUT[2]	p	model output of the spirulina growth at next instant (l/h)

Table 1 : Arguments of the function *lspc*
v : numerical variable
p : pointer

At its first call, the function *lspc* has to be initialised by setting the value of *init* to 0 . This variable is put to 1 by the function itself to avoid next initialisation.

3. MODIFICATIONS OF THE FILE *CTRLSPIR.C*

3.1 *MODIFICATION OF control_spiru*

The call to the function *lspc* is done by the function *control_spiru* of the source file *ctrlspir.c* and replaces all the calculations directly connected to the control.

Hereafter are listed the previous and new function *control_spiru* .

3.1.1 Previous function *control_spiru*

```
/*-------------------------------------
      control programm
-------------------------------------*/

int control_spiru(ScreenViews active_reactor)
      {
      FILE *fp;
      int retval = 0;
      char buffer[300], buffer2[300];
      double  Fr1, Fr2, delfr;                                  /*in W/m2 */
      double  prod_ref,prod1,prod2,prod_max,prod_min;           /*in mg/h */
      double  qe_max, qe_min;                                   /*in l/h */
      double  dil;                                              /*in h-1 */
      double  cxa_moy , nit_moy;                                /*in mg/l */
      REACT   react;

    sprintf (buffer, "Accessing spiruline data ...");
    _outtext (buffer);

    acq_vars_spirulina();

    #ifndef __Monitoring
    display_status("Control running ...");
    #endif

      if(!(next_pfc_sp--))
                {
                /* control PFC algorithm */

                /* biomass concentration */
                cxa_moy=average_var(&cxa,10);
                nit_moy=average_var(&nitrate,10);

                /* production calculation  */
                production.sp=cxa_moy*qe_real.value;

                /* reactor state */
                react.Cno3=nit_moy;
                react.temp=temperature.value;
                react.Cxa=cxa_moy;

                /* flow and production constraints*/
                qe_max=qe_nom.value*(1+DQ);
                qe_min=qe_nom.value*(1-DQ);
                prod_max=qe_max*CXA_MAX;
                prod_min=qe_min*CXA_MIN;

                /* feasible production setpoint calculation*/
                cons_prod_real.sp=max(prod_min,min(prod_max,cons_prod_nom.value));

                /* real flow setpoint  and corresponding dilution rate*/
                qe_real.sp=qe_nom.value;
                if(cons_prod_real.sp/CXA_MAX>qe_nom.value)
                        {qe_real.sp=min(qe_max,cons_prod_nom.value/CXA_MAX);}
                if(cons_prod_real.sp/CXA_MIN<qe_nom.value)
                        {qe_real.sp=max(qe_min,cons_prod_nom.value/CXA_MIN);}
                dil=qe_real.sp/VOLUME_TOTAL;
```

```
                /* reference trajectory */
                prod_ref=cons_prod_real.sp-pow(LAMBDA,NHC)*(cons_prod_real.sp-
production.sp);

                /*first scenario */
                Fr1=Fr.value;
                react.Fr = Fr1;
                prod1=predimod(react,dil,NHC);

                /* second scenario */
                delfr=DFR*signe(cons_prod_real.sp-production.sp);
                Fr2=Fr1+delfr;
                react.Fr = Fr2;
                prod2=predimod(react,dil,NHC);

                /* Fr calculation */
                Fr.sp=Fr.value+(prod_ref-prod1)/(prod2-prod1)*delfr;

                /* constraints on Fr */
                Fr.sp=max(FR_MIN,min(FR_MAX,Fr.sp));

                /* light action sended to output of P100 controller */
                react.Fr=Fr.sp;
                lightcal(&react,CAL_ACT);

                /* pump setpoint sended to P100 controller */
                act_pompe.sp=qe_real.sp/cal_pump.value;

                /* model output calculation */
                prod_mod.sp = predimod(react,dil,1);

                next_pfc_sp=DT;
                }
        send_vars_spirulina();

    sprintf (buffer, " done\n");
    _outtext (buffer);

        return retval;
        }
```

3.1.2 New function *control_spiru*

```
/*-------------------------------------
        control programm
-------------------------------------*/

int control_spiru(ScreenViews active_reactor)
        {
        FILE *fp;
        int retval = 0;
        char buffer[300], buffer2[300];
        REACT   react;
        double  cx;                             /*in g/l */
        double  dt, fI, var_out[3];

    sprintf (buffer, "Accessing spiruline data ...");
    _outtext (buffer);
```

```
acq_vars_spirulina();

#ifndef __Monitoring
display_status("Control running ...");
#endif

    if(!(next_pfc_sp--))
            {
            /* control PFC algorithm */

            /* mean biomass concentration */
            cx=average_var(&cxa,10) / 1000.;

            /* illuminated surface fraction */
            fI = VOLUME_LIGHT / VOLUME_TOTAL;

            /* control period expressed in h */
            dt = DT / 60.;

            /* light spirulina production control */
            lspc(cons_prod_nom.value, cx, qe_nom.value,
                 &Fr.value, qe_real.value, &sm_sup, &cons_sup,
                 VOLUME_TOTAL, fI, dt, LAMBDA, &init_spir, var_out);

            /* light action sended to output of P100 controller */
            react.Fr=Fr.value;
            lightcal(&react,CAL_ACT);

            /* real flow rate set point */
            qe_real.sp = var_out[1];

            /* pump setpoint sended to P100 controller */
            act_pompe.sp=qe_real.sp/cal_pump.value;

            /* model output of spirulina growth */
            prod_mod.sp = var_out[2];

            /* feasible production setpoint */
            cons_prod_real.sp = var_out[0];

            next_pfc_sp=DT;
            retval = 1;
            }
    send_vars_spirulina();

sprintf (buffer, " done\n");
_outtext (buffer);

    return retval;
    }
```

3.2 *MODIFICATIONS OF init_vars_spirulina*

A few instructions of the function *init_vars_spirulina* have to be changed :

3.2.1 Previous instructions

```
/* Initialisation of the supervisor (for removal of the bias) V2.2 */
      prod = cxa.value * qe_nom.value;
      sm_sup = prod;
```

```
      cons_sup = prod;
```

3.2.2 New instruction

```
/* Initialisation of the supervisor (for removal of the bias) V3.0 */
      init_spir = 0.;
```

3.3 *CANCELLATION OF FUNCTIONS*

A few previous functions of *ctrlspir.c* that are now gathered in the source file *lspc.c* , have to be removed. These functions are :

- *model* (replaced by *dercx* in the source file *lspc.c*);
- *predimod* (replaced by *mod_spir* in the source file *lspc.c*).

NB : The mathematical functions *signe* , *min* and *max* are still necessary. The function of light calibration, *lightcal*, which is considered as a part outside the control, remains in the source file *ctrlspir.c* so that it could be modified when it is needed (if a change of equipment occurs, for example)

4. MODIFICATION OF THE INCLUDE *melissa.h*

The following parameters, that are modifiable by the control user, are defined in the include *melissa.h* :

- VOLUME_LIGHT : illuminated volume;
- VOLUME_TOTAL : total volume of the reactor;
- DT : control period;
- LAMBDA : reference trajectory dynamic.

The other parameters, that are modifiable by the persons in charge of the first principles model or of the control, are embedded in the source file *lspc.c* and have to be removed from *melissa.h* .

Hereafter is listed the new version of *melissa.h* :

```
#ifndef __melissa_def
#define __melissa_def

/*************************************************************************

      NAME              MELISSA.H

      AUTHOR            BINOIS C    (modified by FULGET N. ADERSA)

      DESCRIPTION
            General Declarations

      UPDATES
            20-09-95

*************************************************************************/
```

```
/*-------------------------------------
      constants for VARS
-------------------------------------*/
#define TAG     128
#define CMD     255
#define UNDEF   0
#define NB_SAMP 0xFF          /* number of samples stored in val[ ] */


/*-------------------------------------
      structure for variables
-------------------------------------*/

typedef struct _vars {
      char    name[9];        /* tag name              */
      char    file[12];       /* file name             */
      int     type;           /* rd_gps/set_cmd return code            */
      unsigned char tag_cmd;  /* TAG or CMD            */
      unsigned int dev_num;   /* controller number     */
      double  value;          /* current value         */
      int     i;              /* pointer on last value entered in val[] */
      double  val[NB_SAMP+1]; /* previous values       */
      double  min;
      double  max;
      double  sp;             /* set point for LOOP       */
      double  out;            /* out value for LOOP       */
      char    unit[5];        /* unit for analog values   */
      char    update;         /* ON when structure updated*/
} VARS;

/*-------------------------------------
      structure for opened GPS files
-------------------------------------*/

typedef struct _gps_file{
      char    file[15];           /* file name                */
      int     handler;            /* handler of gps file      */
      int     rank;               /* rank of the gps file     */
      struct  _gps_file   *next;  /* next opened gps file     */
} GPS_FILE;


/*-------------------------------------
      structure for reactor state
-------------------------------------*/

typedef struct  _react{
      double  Cxa;    /* in mg/l      */
      double  Cno3;   /* in mg/l      */
      double  temp;   /* in degre C   */
      double  press;  /* in           */
      double  Eb;     /* in W/m2      */
      double  Fr;     /* in W/m2      */
      double  rxa;    /* in mg/l/h    */
      double  rn;     /* in mg/l/h    */
      double  ro2;    /* in mg/l/h    */
} REACT;

/*-------------------------------------
      general constants
-------------------------------------*/
```

```
#define TSAMP_MAIN   60          /* sampling interval in secondes        */
#define SYNCHRO  1
#define ERROR_SPEED 0.01         /* max error for the model      */
#define VOLUME_LIGHT 3.900       /* illuminated volume (in l)    */
#define VOLUME_TOTAL 7.000       /* total volume (in l)          */

/*-------------------------------------
      Model control constants
-------------------------------------*/

#define CAL_FR  10
#define CAL_ACT  20


/*-------------------------------------
      Mathematical constants
-------------------------------------*/

#define PI      3.14159265359


/*-------------------------------------
      colours
-------------------------------------*/

#define BLACK   0
#define BLUE    1
#define GREEN   2
#define CYAN    3
#define RED     4
#define MAGENTA 5
#define BROWN   6
#define WHITE   7

/* =====================================
      Definitions for displaying
      multiples reactors
===================================== */

typedef enum _reactors { principal,
                    spirulina,
                    gspirulina,
                    igspirulina,
                    nitrifying,
                    gnitrifying,
                    ignitrifying,
                    reactor_3,
                    reactor_4,
                    help_1,
                    status,
                    filemanagement } ScreenViews;

/*-------------------------------------
      ADERSA constants
-------------------------------------*/

#define DT      30               /* sampling period of PFC       */
#define DT_NITROGEN     30       /* sampling period of PFC       */
#define LAMBDA  0.88             /* reference trajectory dynamic */

extern int SAMP_GLOB;
```

```
/* ================================
    output file definitions.
   ================================ */

#define CONFIG_FILE "melissa.cfg"

/*#define __Monitoring    /* Just for read only */

#endif
```

5. MODIFICATION OF THE INCLUDE *ctrlspir.h*

The modification of this include concerns only the declaration of the global variable *init_spir* . Its new version is given hereafter.

```
#ifndef __ctrlspir_def
#define __crtlspir_def
/*********************************************************************

NAME            CTRLSPIR.H

AUTHOR          Pedro Pons

DESCRIPTION     Headers and declarations for the spiruline controler.

UPDATES
      27-05-96

*********************************************************************/

#include <malloc.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#include "userdef.h"
#include "melissa.h"
#include "results.h"

int my_interrupt();

/*-------------------------------------
      variables declarations
-------------------------------------*/

VARS    cxa;            /* biomass concentration              */
VARS    nitrate;        /* nitrate concentration              */
VARS    cal_nitrate;    /* nitrate calibration switch         */

VARS    Eb;             /* light intensity in the reactor     */
VARS    Fr;             /* incident flux                      */
VARS    temperature;    /* temperature in the reactor         */
VARS    pH;             /* pH of culture                      */
VARS    act_pompe;      /* dilution pump action               */
VARS    act_lamp;       /* lamp action (dimensionless)        */
VARS    cal_pump;       /* calibration of pump (l/h)          */
```

```
/*********************   variables ADERSA   ******************/

VARS    cons_prod_nom;  /* nominal production setpoint          */
VARS    cons_prod_real; /* feasible production setpoint         */
VARS    qe_nom;         /* nominal flow setpoint                */
VARS    qe_real;        /* feasible flow setpoint               */
VARS    production;     /* measured production                  */
VARS    prod_mod;       /* model production                     */

double sm_sup;          /* model output (internal model) of the supervisor */
double cons_sup;        /* output of the supervisor */
double init_spir;       /* initialisation flag of spirulina control 'lspc'*/

int     next_pfc_sp;       /* next execution of PFC for spirulina reactor */
char    buffer[100];

/***********************************
      variables defined to get the history of
      the system for the input filters
***********************************/

extern double ALPHAFILTERcxa;
extern double ALPHAFILTERnitrate;
extern double ALPHAFILTEREb;
extern double ALPHAFILTERpH;

extern int SAMP_SPIRU;

int first_time_spiruline;

#endif
```

6. PERFORMED TESTS

The function *lspc* is tested on the ADERSA's simulator of the spirulina reactor.
As the whole GPS cannot be compiled on the ADERSA 's computer, the call to *lspc* inside *control_spiru* is not compiled nor tested.
Nevertheless, the call to *lspc* is tested in the short main programme *tst_lspc* whose context is as near as possible of the one of *control_spiru* . The source file *tst_lspc.c* is given hereafter.

```
/*-------------------------------------
Main programme to test the call to 'lspc'
-------------------------------------

      tst_lspc.c

      June 1999
*/

#include <math.h>
#include <stdio.h>
#include "ctrlspir.h"

extern void lspc();

main()
```

```
        {
        FILE *fp;
        int retval = 0;
        char buffer[300], buffer2[300];
      REACT react;
        double  cx;                                  /*in g/l */
        double  dt, fI, var_out[3];
      short i;

init_spir = 0.;

    sprintf (buffer, "Accessing spiruline data ...");
    /* _outtext (buffer); */

      /* simulation of data acquisition */
      cons_prod_nom.value = 0.12;
      qe_nom.value = .07;
      Fr.value = 67.33;
      qe_real.value = .07;

      /* printing results to screen */
       printf("\t Fr cons_prod_real.sp qe_real.sp sm_sup cons_sup prod_mod.sp\n");
       for (i=0; i<=5; i++)
              {
              /* control PFC algorithm */

              /* mean biomass concentration */
              cx=1.36;

              /* illuminated surface fraction */
              fI = VOLUME_LIGHT / VOLUME_TOTAL;

              /* control period expressed in h */
              dt = DT / 60.;

              /* light spirulina production control */
              lspc(cons_prod_nom.value, cx, qe_nom.value,
                   &Fr.value, qe_real.value, &sm_sup, &cons_sup,
                   VOLUME_TOTAL, fI, dt, LAMBDA, &init_spir, var_out);

              /* light action sended to output of P100 controller */
              react.Fr=Fr.value;
              /* lightcal(&react,CAL_ACT); */

              /* real flow rate set point */
              qe_real.sp = var_out[1];

              /* pump setpoint sended to P100 controller */
              act_pompe.sp=qe_real.sp/cal_pump.value;

              /* model output of spirulina growth */
              prod_mod.sp = var_out[2];

              /* feasible production setpoint */
              cons_prod_real.sp = var_out[0];

      /* printing results to screen */
printf("%12.5e%12.5e%12.5e%12.5e%12.5e%12.5e\n",react.Fr,cons_prod_real.sp,qe_real.s
p,sm_sup,cons_sup,prod_mod.sp);

              }
       next_pfc_sp=DT;
```

```
        retval = 1;

    sprintf (buffer, " done\n");
    /* _outtext (buffer); */

        }
```

In order to compile *tst_lspc* on ADERSA 's computer, the 2 includes *userdef.h* and *results.h* were removed from *ctrlspir.h* .
The results of this test are saved in the Table 2.

i	Fr	cons_prod_real.sp	qe_real.sp	sm_sup	cons_sup	prod_mod.sp
0	8.307e+01	1.155e-01	7.700e-02	9.520e-02	1.155e-01	1.047e-01
1	1.656e+02	1.155e-01	7.700e-02	9.763e-02	1.179e-01	1.048e-01
2	3.145e+02	1.155e-01	7.700e-02	1.000e-01	1.194e-01	1.050e-01
3	3.924e+02	1.155e-01	7.700e-02	1.024e-01	1.194e-01	1.051e-01
4	3.998e+02	1.155e-01	7.700e-02	1.044e-01	1.194e-01	1.051e-01
5	3.999e+02	1.155e-01	7.700e-02	1.062e-01	1.194e-01	1.051e-01

Table 2 : Results of simulation with *tst_lspc*
(i is the counter of the *for* loop of *tst_lspc*)

7. EXAMPLE OF TEST TO PERFORM ON UAB 's COMPUTER

The aim is to check the integration of the new versions of *control_spiru* , *melissa.h* and *ctrlspir.h* in the GPS software without using the Spirulina reactor.

It should be possible to force the value of sensors to the values of *tst_lspc* :

- measured flow rate (`qe_real.value`) = 0.07 l/h at the beginning of the test;
- biomass concentration (`cx`) = 1360 mg/l (assumed constant all along the test);
- light controller action = 0.735 at the beginning of the test (corresponding to Fr_mes = 67.33 W/m^2).

The set point of nominal production (`cons_prod_nom.value`) has be put to 0.12 g/h and the nominal flow rate (`qe_nom.value`) to 0.07 l/h .

The results of the six first runs should be identical to those of the table 2.

8. CONCLUSION

The test proposed in the previous example was done at UAB and the results are correct (e-mail from Joan Albiol on July 9th 1999).
UAB realised four other tests whose initial conditions are detailed in Table 3.

The results of UAB are compared to those of ADERSA in the tables 4 to 7.

Test number	Cx	Fr_initial	cons_prod_nom	qe_nom
1	0.660	67.33	0.120	0.07
2	0.800	10.00	0.090	0.07
3	1.500	200.0	0.020	0.05
4	1.400	400.0	0.010	0.07

Table 3 : Initial conditions of the supplementary tests done by UAB

TEST 1

i	Fr		cons_prod_real.sp		qe_real.sp		prod_mod.sp	
	UAB	ADERSA	UAB	ADERSA	UAB	ADERSA	UAB	ADERSA
0	175.1	176.6	0.1155	0.1155	0.0770	0.0770	0.0508	0.0512
1	304.6	307.3	0.1155	0.1155	0.0770	0.0770	0.0508	0.0513
2	384.0	385.3	0.1155	0.1155	0.0770	0.0770	0.0508	0.0513
3	399.7	399.7	0.1155	0.1155	0.0770	0.0770	0.0508	0.0513
4	400.0	400.0	0.1155	0.1155	0.0770	0.0770	0.0508	0.0513
5	400.0	400.0	0.1155	0.1155	0.0770	0.0770	0.0508	0.0513

Table 4 : Comparison of results UAB/ADERSA for test 1

TEST 2

i	Fr		cons_prod_real.sp		qe_real.sp		prod_mod.sp	
	UAB	ADERSA	UAB	ADERSA	UAB	ADERSA	UAB	ADERSA
0	74.5	74.8	0.090	0.090	0.0700	0.0700	0.0560	0.0562
1	193.6	194.8	0.090	0.090	0.0700	0.0700	0.0560	0.0564
2	326.7	328.3	0.090	0.090	0.0700	0.0700	0.0560	0.0564
3	391.9	392.4	0.090	0.090	0.0700	0.0700	0.0560	0.0565
4	400.0	400.0	0.090	0.090	0.0700	0.0700	0.0560	0.0565
5	400.0	400.0	0.090	0.090	0.0700	0.0700	0.0560	0.0565

Table 5 : Comparison of results UAB/ADERSA for test 2

TEST 3

i	Fr		cons_prod_real.sp		qe_real.sp		prod_mod.sp	
	UAB	ADERSA	UAB	ADERSA	UAB	ADERSA	UAB	ADERSA
0	10.0	10.0	0.0225	0.0225	0.0450	0.0450	0.0675	0.0674
1	10.0	10.0	0.0225	0.0225	0.0450	0.0450	0.0675	0.0674
2	10.0	10.0	0.0225	0.0225	0.0450	0.0450	0.0675	0.0674
3	10.0	10.0	0.0225	0.0225	0.0450	0.0450	0.0675	0.0674
4	10.0	10.0	0.0225	0.0225	0.0450	0.0450	0.0675	0.0674
5	10.0	10.0	0.0225	0.0225	0.0450	0.0450	0.0675	0.0674

Table 6 : Comparison of results UAB/ADERSA for test 3

TEST 4

i	Fr		cons_prod_real.sp		qe_real.sp		prod_mod.sp	
	UAB	ADERSA	UAB	ADERSA	UAB	ADERSA	UAB	ADERSA
0	10.0	10.0	0.0315	0.0315	0.0630	0.0630	0.0882	0.0879
1	10.0	10.0	0.0315	0.0315	0.0630	0.0630	0.0882	0.0879
2	10.0	10.0	0.0315	0.0315	0.0630	0.0630	0.0882	0.0879
3	10.0	10.0	0.0315	0.0315	0.0630	0.0630	0.0882	0.0879
4	10.0	10.0	0.0315	0.0315	0.0630	0.0630	0.0882	0.0879
5	10.0	10.0	0.0315	0.0315	0.0630	0.0630	0.0882	0.0879

Table 7 : Comparison of results UAB/ADERSA for test 4

For these four supplementary tests, the distances between the results are quite correct too : they are less than 1 % for the 2 first tests and less than 0.4 % for the 2 last ones.

REFERENCES

CORNET J.F., DUSSAP C.G., GROS J.B. "Modelling of physical limitations in photobioreactors. Adaptation of the light energy transfer model to cylindrical geometry" ESA contract PRF 130820, 1993, TN 19.1 .

FULGET N. "Study for the non linear Base Model predictive Control of Spirulina compartment using knowledge model". ESA/ESTEC contract PRF 142356, October 1995, TN 24.2.

LECLERCQ J.J. "Final validation of the production control of the 7 litres photoautotrophic compartment". ESTEC contract 171686 of 16/07/1997, May 1998, TN 38.2 .

PONS P., ALBIOL J., GODIA F. "General Purpose Station. Software upgrade 97". ESTEC contract 11549/95/NL/FG, April 1998, TN 37.9 .

RICHALET J. "Pratique de la commande prédictive". HERMES Edition, 1993.

ANNEX 1

Modification of the reference trajectory dynamic LAMBDA

The reference trajectory dynamic, LAMBDA, is a parameter, at the disposal of the control user, which allows to shorten or increase the closed loop time response. The general rule is : the smaller this parameter is (between 0 and 1), the shorter the closed loop time response is and the less robust the control is.

When it was tested on the 7 litres reactor in October 1997, it can be seen that the manipulated variable, Fr, had a tendency to oscillate after a set point step. It was because the parameter LAMBDA was too small (LAMBDA was equal to 0.75).

Simulations of the closed loop system show that the optimal value of lambda is 0.88 : the closed loop time response is nearly the same but the behaviour of the manipulated variable is smoother . Figures A1.1 and A1.2 give an example of this behaviour when control and process are mismatched : in figure 1, LAMBDA = 0.75 and in figure 2, LAMBDA = 0.88 .

This value of LAMBDA = .88 is now fixed in the new version of *melissa.h* .

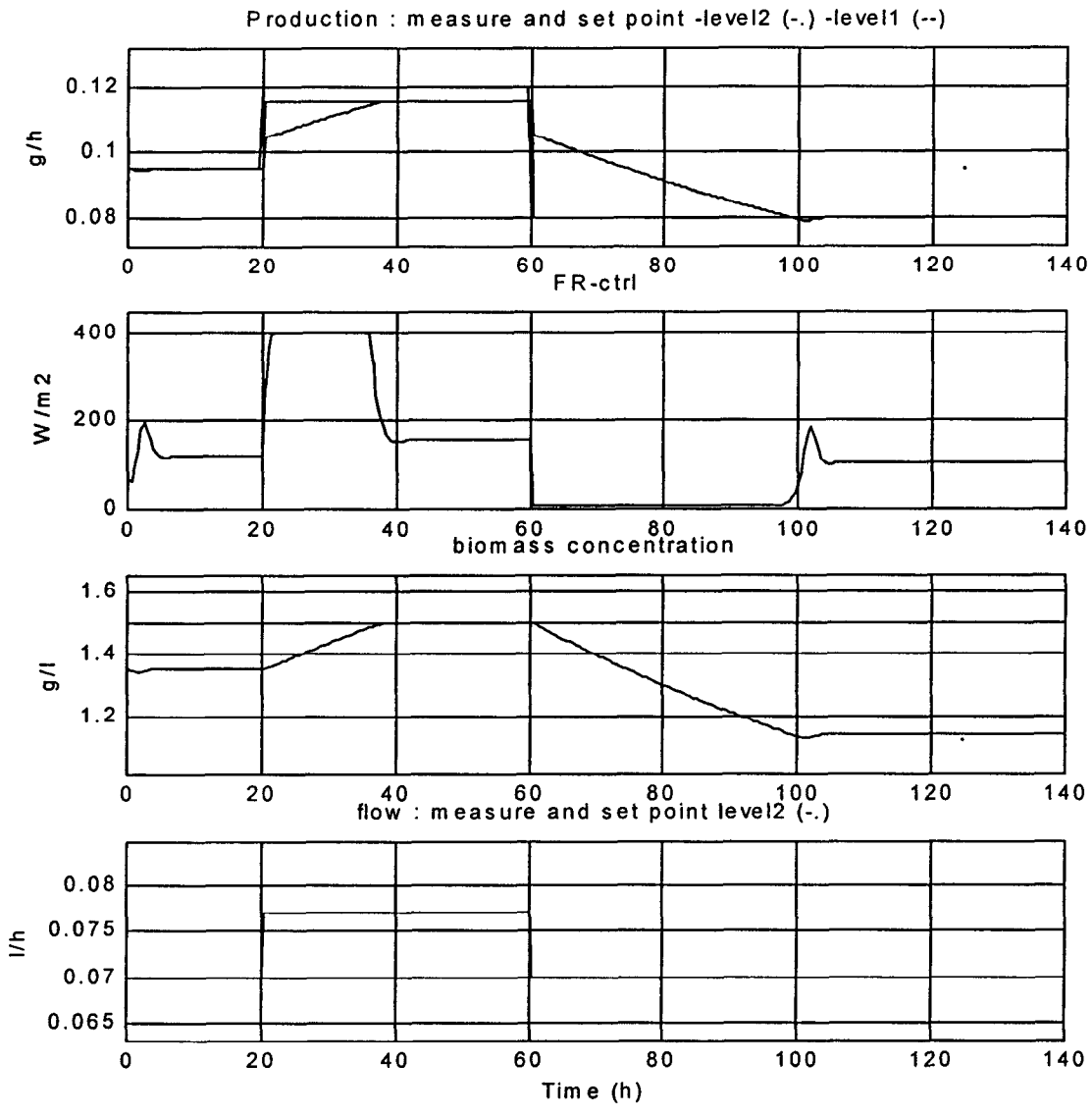

Figure A1.1 : Mismatched controller : Fr_process = Fr_ctrl - 50

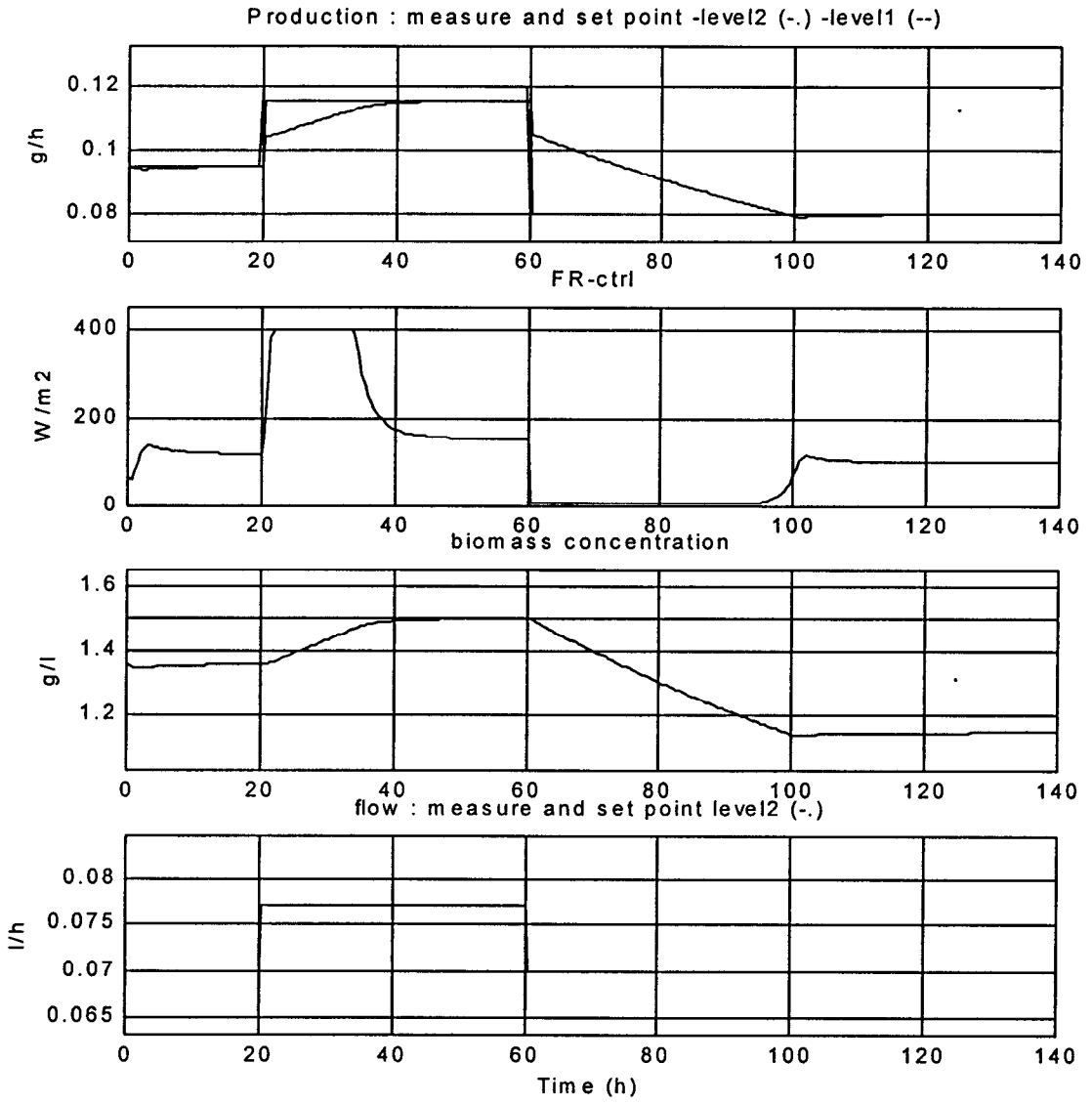

Figure A1.2 : Mismatched controller : Fr_process = Fr_ctrl - 50

ANNEX 2

Source file of the Light Spirulina Production Control

LSPC.C
V3.0

```
/*
   LSPC.C   Light Spirulina Production Control
   Version 3.0

   AUTHOR : FULGET N. (ADERSA)
   September 1995

   Modified by LECLERCQ JJ (ADERSA)
   . Elimination of the static bias according to TN 38.2 (October 1997)
   . Improvement of the transferability according to TN 38.1 (June 1999)


-------------------------------------------------------------------------
   Function:
      Non linear PFC control of Spirulina production by light
   Synopsis:
      LSPC(PROD_SP2,CX,QE_SP2,FR,QE_MES,SM_SUP,CONS_SUP,
           VOL,FI,DT,LAMBDA,INIT,VAR_OUT)
      (v) : numerical value
      (p) : pointer
      PROD_SP2 (v):level2 production set point (g/h)
      CX       (v):biomass concentration (g/l)
      QE_SP2   (v):level2 flow rate set point (l/h)
      FR       (p):light intensity : measured or computed by the control (W/m2)
                   . input argument : measured value of FR
                   . output argument : computed by the control
      QE_MES   (v):measure of flow rate (l/h)
      SM_SUP   (p):production model output computed by the supervisor (g/h)
                   . input argument : value at previous moment
                   . output argument : value at present moment
      CONS_SUP (p):production set point computed by the supervisor (g/h)
                   . input argument : value at previous moment
                   . output argument : value at present moment
      VOL      (v):volume of the reactor (l)
      FI       (v):illuminated surface fraction (no dimension)
      DT       (v):control period (h)
      LAMBDA   (v):dynamic of the reference trajectory (dimension less)
      INIT     (p):initialisation flag (when equal to 0)
                   put to 1 by this programme
      VAR_OUT[0]  :level 1 production set point (g/h)
      VAR_OUT[1]  :level 1 flow rate set point (l/h)
      VAR_OUT[2]  :model output of the spirulina growth at next instant (l/h)
   Description:
      . Computation of level 1 production and flow rate set points, VAR_OUT[0]
      and VAR_OUT[1], functions of level 2 set points, PROD_SP2 and QE_SP2.
      . Computation of the Manipulated Variable at current instant n, FR(n),
      function of level 1 production and flow rate set points,
      of measured flow rate, QE_MES, and
      of measured ligth intensity measured at previous time, FR(n-1).
*/

#include "math.h"

/* prototype */
void lspc( );
double mod_spir( );
double dercx( );
double signe( );
double min( );
double max( );
```

```
#ifndef ADERSA
void lspc(double prod_sp2, double cx, double qe_sp2, double *Fr, double qe_mes,
          double *sm_sup, double *cons_sup, double vol, double fI,
          double dt, double lambda, double *init, double var_out[3])
#else
void lspc(prod_sp2, cx, qe_sp2, Fr, qe_mes,
          sm_sup, cons_sup, vol, fI, dt, lambda, init, var_out)
double   prod_sp2, cx, qe_sp2, *Fr, qe_mes,
         *sm_sup, *cons_sup, vol, fI, dt, lambda, *init, var_out[3];
#endif


{
        double  nhc=5.;         /* coincidence point (in dt) */
        double  dFr=5.;         /* radiant flux increment (in W/m2)  */
        double  Fr_min=10.;     /* min constraint on FR (in W/m2) */
        double  Fr_max=400.;    /* max constraint on FR (in W/m2) */
        double  dq=.1;          /* flow variation (dimensionless) */
        double  cx_min=0.5;     /* min constraint on CX (in g/l) */
        double  cx_max=1.5;     /* max constraint on CX (in g/l) */


/* internal variables declaration */
   double   prod, dil, prod_ref, delFr;
   double   Fr1, Fr2, prod1, prod2, prod_mod;
   double   qe_max , qe_min , prod_max , prod_min ;
   double   cons_prod1 , cons_qe1;
   double   cons_prod0 , cons_prod0_min , cons_prod0_max;
   double   dprod_min , dprod_max;

/* level 2 control */
   /* 1_ flow and production constraints */
   qe_max = qe_sp2*(1+dq);
   qe_min = qe_sp2*(1-dq);
   prod_max = qe_max*cx_max;
   prod_min = qe_min*cx_min;
   /* 2_ feasible production setpoint calculation */
   cons_prod1 = max(prod_min,min(prod_max,prod_sp2));
   /* 3_ real flow setpoint  and corresponding dilution rate */
   cons_qe1 = qe_sp2;
   if (cons_prod1/cx_max > qe_sp2 )
     cons_qe1 = min(qe_max,cons_prod1/cx_max);
   if (cons_prod1/cx_min < qe_sp2 )
     cons_qe1 = max(qe_min,cons_prod1/cx_min);

/* computation of dilution and biomass production */
   dil = cons_qe1/vol;
   prod = cx * qe_mes;

/* PFC_supervisor to cancel static bias of the scenario method
   Internal model of this PFC : first order without delay
                                unit static gain
                                trbo = trbf of the embeded system */
   /* 1_ computation of the internal model output of this PFC */
   if (*init < 0.5)
   {
     *sm_sup = prod;
     *cons_sup = prod;
     *init = 1.;
   }
   *sm_sup = lambda * *sm_sup + (1. - lambda) * *cons_sup;
```

```
    /* 2.1_ Computation of the output of the supervisor 'cons_prod0' */
    cons_prod0 = cons_prod1 - prod + *sm_sup;
    /* 2.2_ Transfer of the constraints 'Fr_min' and 'Fr_max' on 'cons_prod0' */
    dprod_max = mod_spir(cx,Fr_max,dil,vol,dt,nhc,fI) - prod;
    cons_prod0_max = prod + dprod_max / (1.-pow(lambda,nhc));
    dprod_min = mod_spir(cx,Fr_min,dil,vol,dt,nhc,fI) - prod;
    cons_prod0_min = prod + dprod_min / (1.-pow(lambda,nhc));
    cons_prod0 = max(cons_prod0_min,min(cons_prod0_max,cons_prod0));

/* reference trajectory */
    prod_ref = cons_prod0 - pow(lambda,nhc)*(cons_prod0 - prod);

/* first scenario */
    Fr1 = *Fr;
    prod1 = mod_spir(cx,Fr1,dil,vol,dt,nhc,fI);

/* second scenario */
    delFr = dFr*signe(cons_prod0 - prod);
    Fr2 = Fr1 + delFr;
    prod2 = mod_spir(cx,Fr2,dil,vol,dt,nhc,fI);

/* computation of the Manipulated Variable 'Fr' */
    *Fr += (prod_ref - prod1)/(prod2 - prod1)*delFr;

/* constraints on Fr */
    *Fr = max(Fr_min,min(Fr_max,*Fr));

/* model output of the spirulina growth at next moment */
    prod_mod = mod_spir(cx,*Fr,dil,vol,dt,1.,fI);

/* transfer of output arguments */
    *cons_sup = cons_prod0;
    var_out[0] = cons_prod1;
    var_out[1] = cons_qe1;
    var_out[2] = prod_mod;
    return;
}
/* ---  MOD_SPIR  ------------------------------------------------------------
    Function:
          Integration with Euler method
    Synopsis:
          MOD_SPIR(CX,FR,DIL,VOL,DT,NHC,FI,PROD)

*/
double mod_spir(cx,Fr,dil,vol,dt,nhc,fI)

double cx, Fr, dil , vol , dt , nhc , fI;

{
    double v, dv, vout , prod;
    int k;

    v = cx;
    for (k=1 ; k <= (int)nhc; k++)
        {
        dv = dercx(v,Fr,dil,fI);
        vout =v + dt *dv;
        v=vout;
        }
    prod=vout*dil*vol;
    return (prod);
}
```

```
/* ---   DERCX  ------------------------------------------------------------

   Function:
         Computation of the derivative of the biomass concentration
   Synopsis:
         DERCX(CX,FR,DIL,FI,DVT);

*/
double dercx(cx,Fr,dil,fI)

double cx, Fr, dil, fI;

{
        double  zpc=.135;       /* biotic mass fraction of phycocyanins */
        double  zch=0.0085;     /* biotic mass fraction of clorophylles */
        double  zg=0.;          /* biotic mass fraction of glycogen */
        double  Ea=871.;        /* global absorption mass coefficient (m2/kg) */
        double  Es=167.;        /* global scattering mass coefficient (m2/kg) */
        double  Kj=20;          /* half saturation constant for radiant energy
                                   avalaible (W/m2) */
        double  muM=0.54;       /* maximum growth rate (1/h) */
        double  RT=0.048;       /* radius of the cylindrical bioreactor */
        double  R1=0.0302;
        double  R2=0.02585;
        double  Fmin=1.;        /* minimum radiant energy flux (W/m2) */
        double  jstep=0.01;     /* integrative dimensionless step */

        double  z0, kstep, sX;
        double  dcxdt;
        double  alpha, delta, pijz;
        double  z, rx;
        double  za;


        /*  general parameters ----------------------------------*/
        za=zpc+zch;
        alpha=sqrt(za*Ea/(za*Ea+(1+zg)*Es));
        delta=(za*Ea+(1+zg)*Es)*cx*alpha*RT;

        /*
        Computation of rx
        Integration interval : [z0, 1]
        This interval is divided into 'nstep = 1/jstep ' equal parts
        Integration : rectangle method
        */
        z0 = 1.e-6 / RT;
        kstep = (1. - z0) * jstep;
        sX = 0.;
        for (z=z0; z<1.; z+=kstep)
          {
          if(( z<= R2/RT)||(z >= R1/RT))
            {
            pijz = 2*Fr/z*cosh(delta*z)/(cosh(delta)+alpha*sinh(delta));
            if (pijz>=Fmin)
              sX += z * pijz / (Kj+pijz);
            }
          }
        rx = 2. * muM * cx * sX * fI * zpc * kstep;

        dcxdt = -dil*cx + rx ;
```

```
        return (dcxdt);
}
□
```