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

MELISSA :

FIRST APPROACH OF MODEL BASED PREDICTIVE CONTROL OF SPIRULINA COMPARTMENT

Contract ESA-ESTEC/ADERSA
P.R.F. n° 132443

Technical Note 21.2

N. FULGET

December 1994

7, Bd du Maréchal Juin
B.P. 52
91371 VERRIERES-LE-BUISSON CEDEX
Tél. (1) 60.13.53.53
Télécopieur (1) 69.20.05.63

ADERSA

SUMMARY

1 - INTRODUCTION

During the first part of the study, the Spirulina Compartment has been considered in the following conditions of functioning :

- constant concentration of biomass : controlled with the action of the harvesting pump flow ;
- constant volume : controlled with the action of the injection pump ;
- control of the production of biomass with action on the light intensity.

The goal of the study was to test a predictive control law for the control of the production of biomass. In order to do that, the dynamic transfer between light intensity and biomass production has been identified. A simulator of the system, based on the results of identification, and on the knowledge of the physical behaviour, has been built. This analysis has allowed to underline some problems in the measurement system (biomass sensor, speed of production estimator, ...). The predictive control law has been tested in simulation (T.N. 21.1).

2 - NEW ORIENTATION OF THE STUDY

After a presentation of those results, it has seemed that the chosen conditions of functioning were not really adapted to the functioning of the whole system (interaction of all the compartments). Indeed, the incoming flow will be certainly given by the previous compartment, and won't be used to control the volume. So, the volume will be controlled with the harvesting pump flow, and the biomass concentration won't be controlled. As in the first configuration, the speed of production will be controlled with the light intensity.

In this configuration, the dynamic of the system is completely different. Another simulator has to be built. To do that, the knowledge model developped in the L.G.C.B. (Clermont Ferrand) is used, it is integrated in the Simulink Simulator and tuned according to some test protocols on the process.

3 - INTEGRATION OF THE KNOWLEDGE MODEL IN SIMULINK SIMULATOR

3.1. Presentation of the knowledge model

This model has been elaborated by J.F. Cornet (LGCB Clermont Ferrand) (TN 19.3), for simulation of culture of the cyanobacterium Spirulina Platensis cultivated in cylindrical and radially illuminated photobioreactors. Limitations by light and minerals (nitrate, sulfate) which may occur in the photobioreactor are taken into account. Nine compounds (including nitrate and sulfate as substrates are considered in the model. The physical equations are described in the TN 19.1 and TN 19.2.

The inputs of the model are the nine compounds concentrations in the incoming flow C_i^E, the dilution rate D, and the incident radiant flux F_r on the reactor.

For each compound, the conservation equation is given by :

$$\frac{dC_i^S}{dt} - \langle r_i \rangle - D\left(C_i^E - C_i^S\right) = 0$$

$\langle r_i \rangle$ is the mean volumetric reaction rate for the compound i calculated by the model of TN 19.2.

The outputs of the model are the concentrations of the nine compounds in the reactor : $\left\{C_i^S\right\}_{i=1,\ldots,9}$.

3.2. Interface of a Fortran model with Simulink

The simulator developped by J.F. Cornet (LGCB - Clermont Ferrand) is in Fortran. It is possible to interface it with Simulink. It is the best solution for at least two reasons :

- the speed of calculation is greater ;
- we don't have to translate in Matlab language, all the subroutines (risk of writing errors ...).

To interface it with Simulink, we first need the subroutine "deriv" (TN 19.3) which calculates the derivatives $\frac{dC_i^S}{dt}$ of each concentration in function of the input values $\left(D, F_r, \left\{C_i^E\right\}_{i=1,\ldots,9}\right)$ and of the concentrations in the reactor $\left\{C_i^S\right\}_{i=1,\ldots,9}$. It defines a non linear integro-differential system, which is directly integrated by Simulink. The Fortran subroutine photolab.f (which contains deriv ...) is compiled and linked with other fortran programs Simulink.f and photomexf.f (which defines an S. function). The instruction is :

fmex photomexf.f photolab.f simulink.f

It creates a mex.file photomexf.mex which is an S. function, and can be used in a Simulink scheme (see program in annex).

3.3. Test of the interface of Fortran Model with Simulink

When the integration of the fortran model in Simulink is done, we have to validate it. It's done by comparing the results obtained with the simulator of LGCB, and those obtained with Simulink (fig. 1) in the same conditions. We consider two cases of functioning that are described in the TN 19.3.

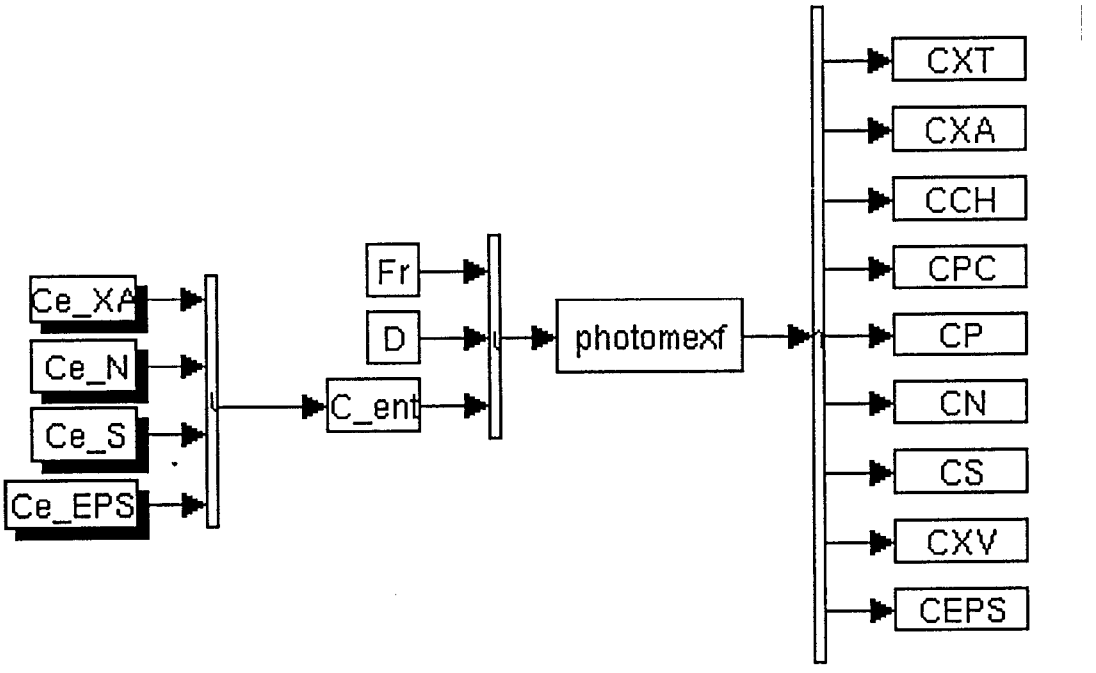

Figure 1

• First case :

Starting of a continuous culture after a batch cultivation period of 50 hours with a step in incident radiant energy flux from 50 W/m^2 to 100 W/m^2 after 250 hours of cultivation. The initial concentrations in the photobioreactor are respectively :

C_{XA} = .1 kg/m^3
C_{EPS} = .02 kg/m^3
C_N = .8 kg/m^3
C_S = .2 kg/m^3

The other concentrations are calculated from those one, according to the following equations :

$$C_{XV} = C_{XA}$$
$$C_{XT} = C_{XA} + C_{EPS}$$
$$C_{CH} = 0.01 * C_{XA}$$
$$C_{PC} = .162 * C_{XA}$$
$$C_{P} = .684 * C_{XA}$$

At a time of 50 hours, the reactor is supplied with an optimal dilution rate of 0.026 h^{-1} and with sustrates at the following concentrations in the incoming flow :

$$C_{e\_N} = .8\ kg/m^3$$
$$C_{e\_S} = .2\ kg/m^3$$
$$C_{e\_XA} = 0\ kg/m^3$$
$$C_{e\_EPS} = 0\ kg/m^3$$

The five other concentrations are given by the same equations as for the incoming flow, they are all equal to zero. At a time of 250 hours the incident radiant energy flux is increased from 50 W/m^2 to 100 W/m^2.

The figure 2 shows the results obtained with the simulator of LGCB (photosim), the figure 3 shows the results obtained with the Simulink simulator. They are exactly the same.

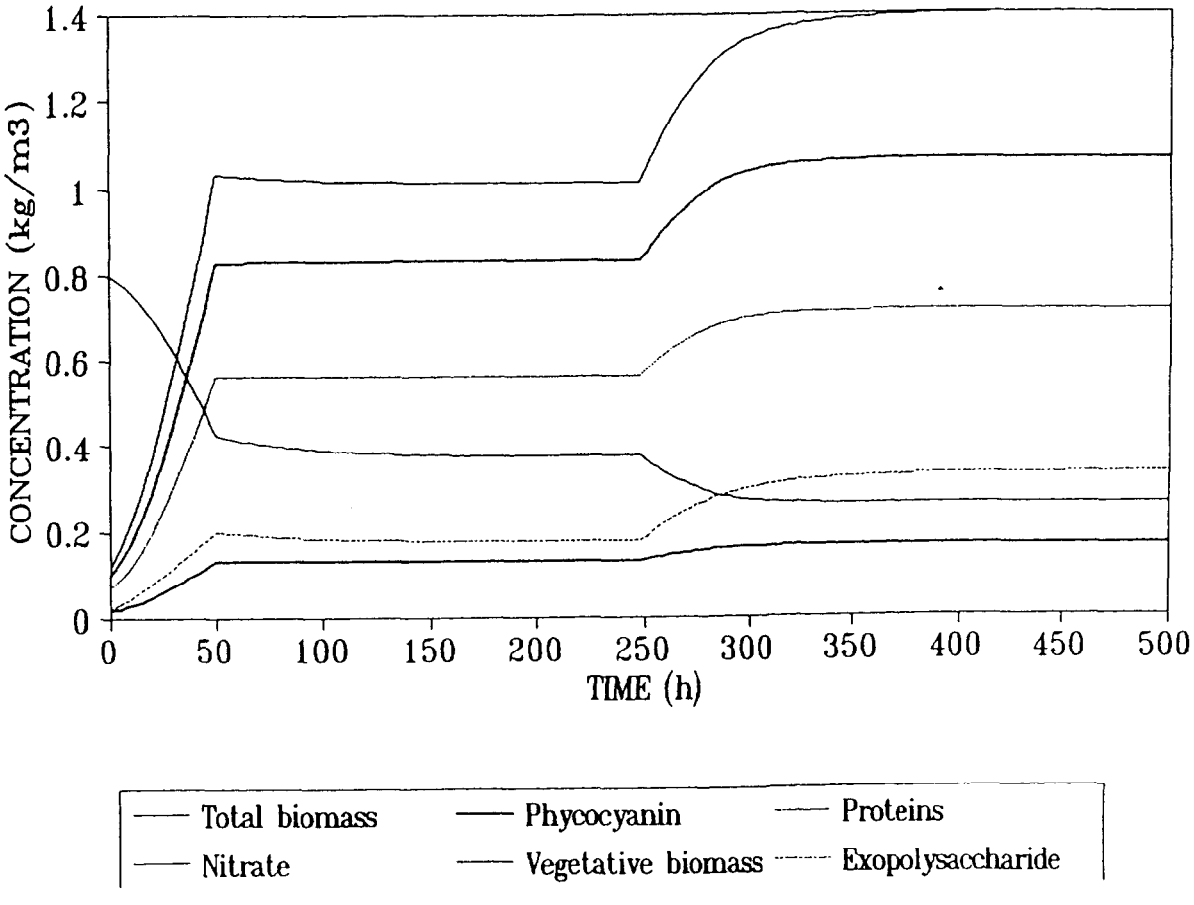

Figure 2 : Results with Photosim
(dil : 0 → .026 h^{-1}
F_r : 50 → 100 W/m^2)

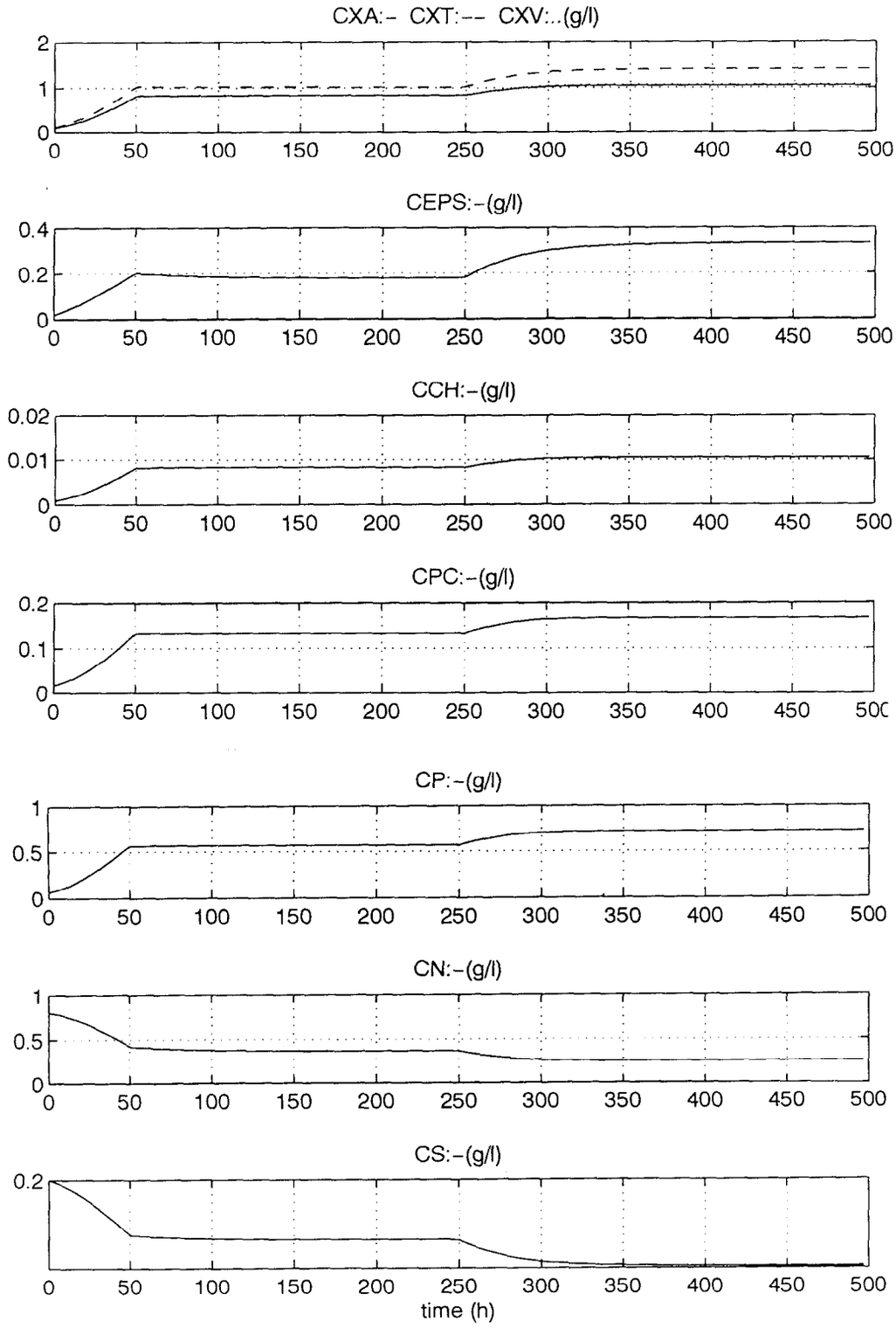

Figure 3 : Results with Simulink
(dil : 0 → .026 h^{-1}
F_r : 50 → 100 W/m^2)

• Second case :

The second case is the same as the first, except that the dilution rate is not optimal, its value is 0.05 h^{-1}. The results with both simulator can be compared (fig. 4 and 5).

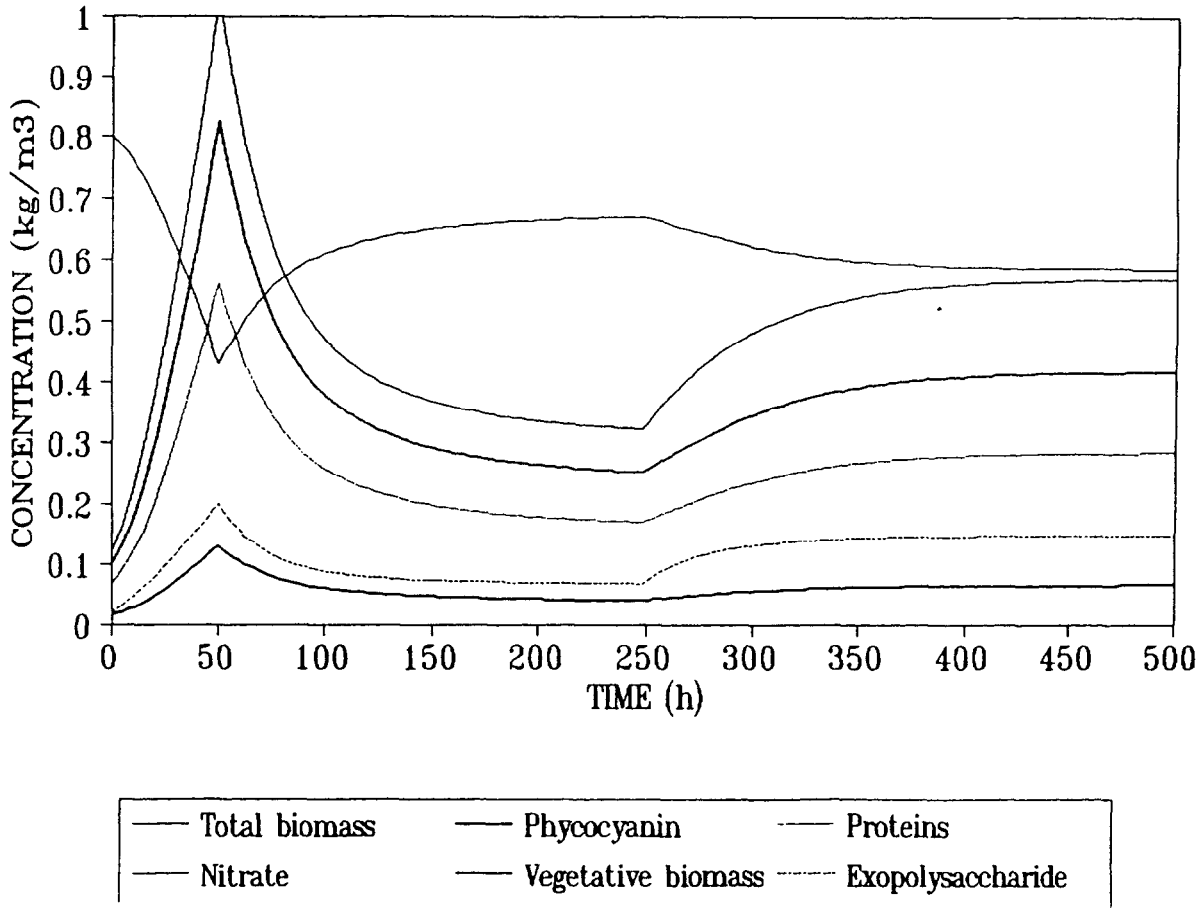

Figure 4 : Results with Photosim
(dil : 0 → 0.05 h^{-1}
F_r : 50 → 100 W/m^2)

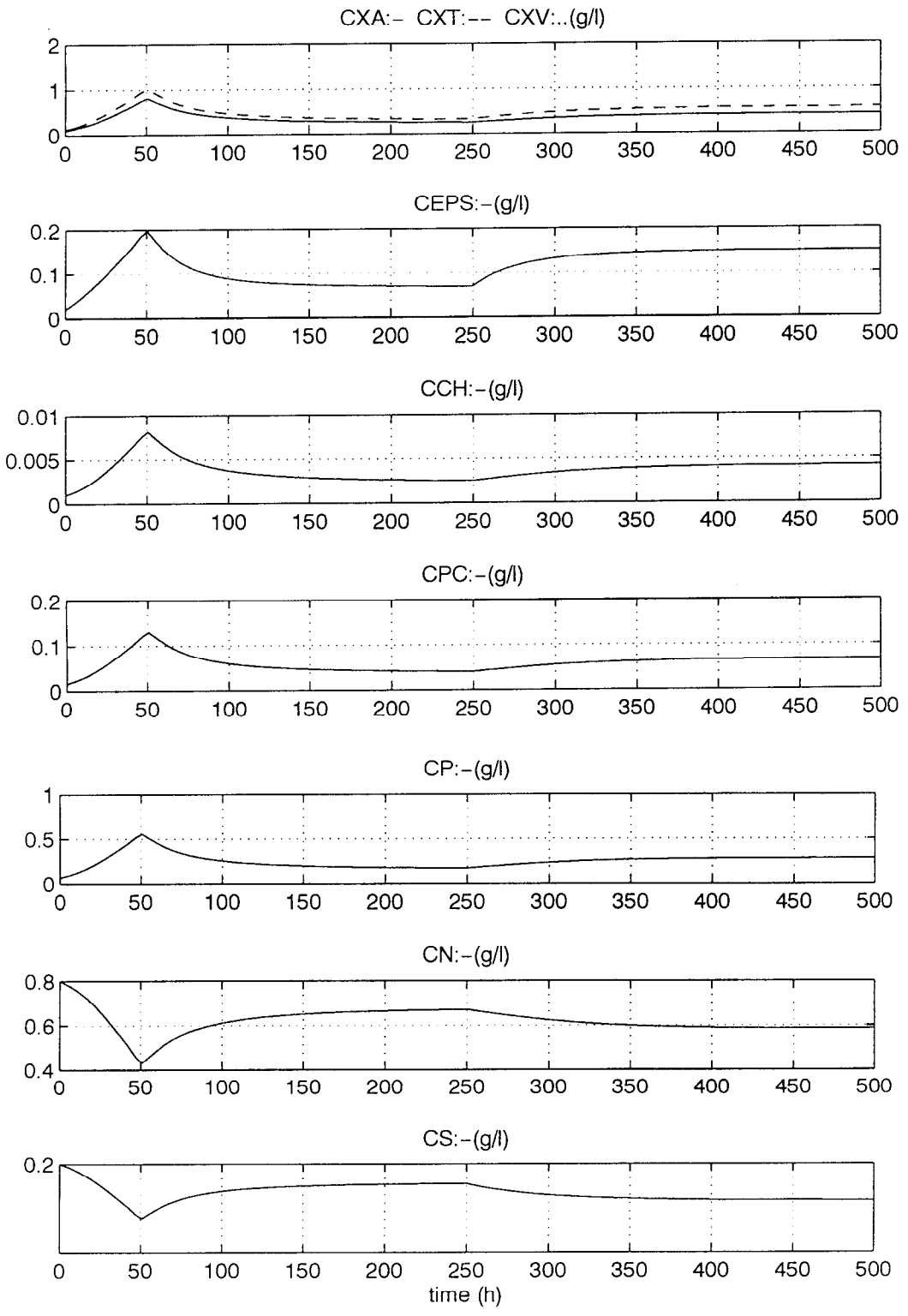

Figure 5 : Results with Simulink
(dil : 0 → 0.05 h^{-1}
F_r : 50 → 100 W/m^2)

On those two examples, we can verify that the behaviours are the same in Simulink and in fortran. It validates the interfacing. Then, this box can be integrated in a more complete scheme including the regulation of volume, and considering that the measured light is the light in the center of the reactor and not the incident radiant energy flux.

3.4. Relation $F_r = f(E_b)$

There exists a relation between the measured light in the center of the reactor E_b and the incident radiant flux F_r. It depends on the different concentrations. It has been elaborated by C. Binois (CNAM thesis 1994). The relation is given in function of the reactor radius.

The implementation of this relation in Simulink is done, considering that the inputs of this function are the concentrations of the different compounds in the reactor (only the concentrations C_{XA}, C_{PC} and C_{CH} are used in the calculations), and the measured light in the center of the reactor.

The program is given in the annex. It is called EB_FR.m.

3.5. Complete simulator of the process

The model "photomexf" of the Spirulina photosynthesis in integrated in a complete simulator of the reactor (figure 6). This model is considered with the following inputs :

- the light intensity measured in the center of the reactor (in W/m^2) ;
- the incoming flow (in l/h) ;
- the main concentrations in the incoming flow (C_{e_XA}, C_{e_N}, C_{e_S}, C_{e_EPS}).

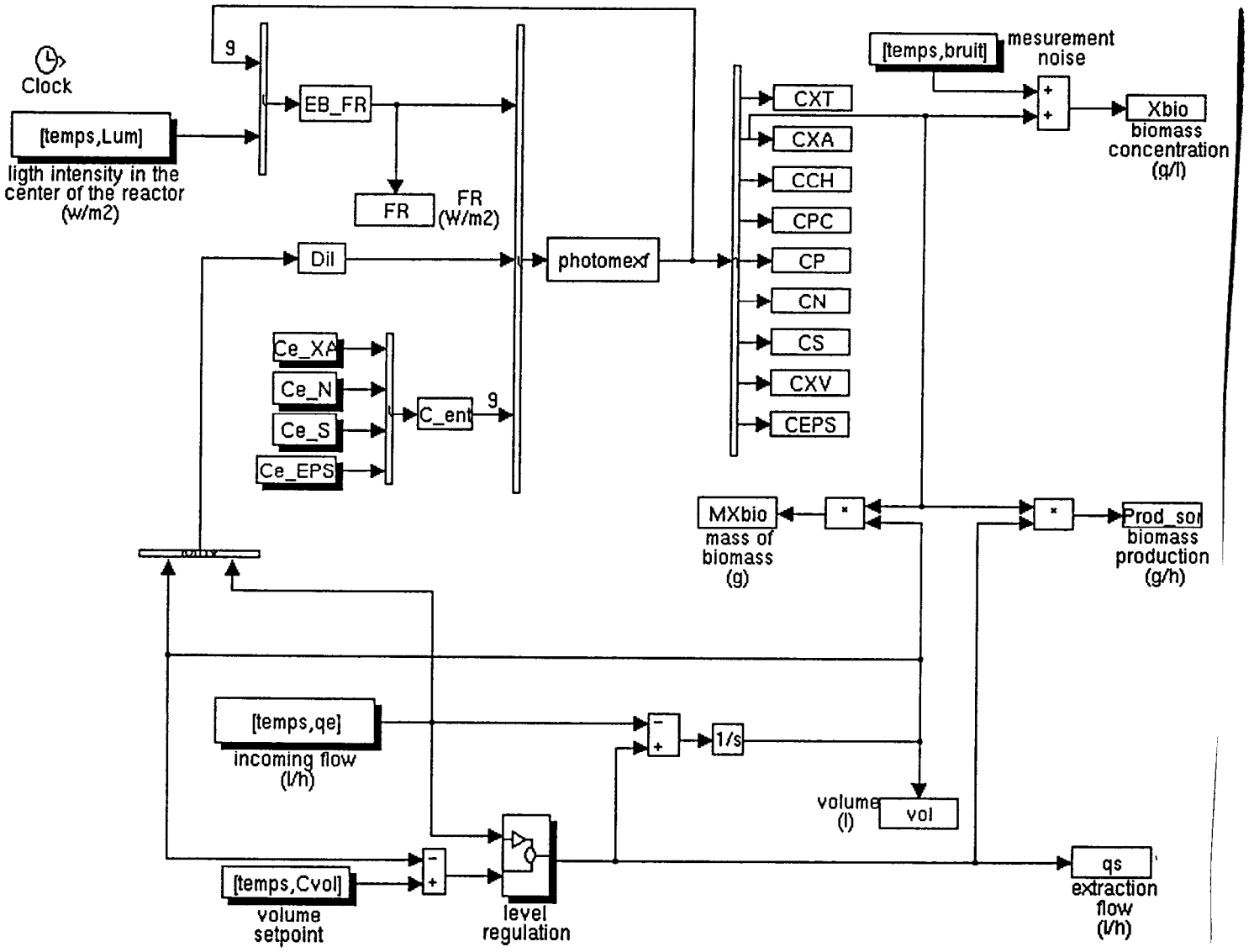

Figure 6

The level regulation is supposed to be realized by action of the extraction flow, in function of the incoming flow, and in function of the medium level (image of the volume).

The dilution rate is calculated with the incoming flow and the volume of the medium in the reactor.

The biomass concentration Xbio, which is measured is supposed to be equivalent to the active biomass C_{XA} with addition of a measurement noise.

the production of biomass is calculated with the biomass concentration measure and the extraction flow measure.

4 - CONCLUSION

The knowledge model developped by J.F. Cornet (LGCB Clermont-Ferrand - TN 19.i) is now integrated in our Simulink simulator.

We have to validate it with experimental data and after, we will be able to use it as a simulator of the process and as an internal model of our predictive control law.

This will be done in the next contract.

REFERENCES

BINOIS C., 1994. Automatisation d'un écosystème artificiel utilisé comme système de support vie. Première interaction modèle/système de contrôle. CNAM Thesis. September 1994.

CORNET J.-F., DUSSAP C.G., GROS J.-B., 1993a. Modelling of physical limitations in photobioreactors. Adaptation of the light energy transfer model to cylindrical geometries. ESA contract PRF 130-820, Technical Note 19.1.

CORNET J.-F., DUSSAP C.G., GROS J.-B., 1993b. Modelling of physical limitations in photobioreactors. Modelling of exopolysaccharide synthesis in cultures of *Spirulina platensis*. ESA contract PRF 130-820, Technical Note 19.2.

CORNET J.-F., DUSSAP C.G., GROS J.-B., 1993c. Modelling of physical limitations in photobioreactors. Applications to simulation and control of the Spirulina Compartment of the MELISSA artifical ecosystem. ESA contract PRF 130-820, Technical Note 19.3.

ANNEX

programs for the simulator :

```
      SUBROUTINE PHOTOLAB(F,CI,Y)
C
C*********************************************************************
C*    PROGRAMME PRINCIPAL DE SIMULATION D'UN PHOTOBIOREACTEUR         *
C*    CYLINDRIQUE POUR LA CULTURE DE SPIRULINA PLATENSIS.            *
C*                  ____________________________________             *
C*                                                                   *
C*                           PHOTOSIM                                *
C*                                                                   *
C*                                                                   *
C*                                          J-F. CORNET.             *
C* LABORATOIRE DE GENIE CHIMIQUE BIOLOGIQUE, UNIVERSITE BLAISE PASCAL. *
C*                                       T.N. 19.3, 1993.            *
C*********************************************************************
C*                                                                   *
C*    CE PROGRAMME PERMET LA SIMULATION DU CALCUL DES CONCENTRATIONS  *
C*    DANS UN PHOTOBIOREACTEUR CYLINDRIQUE A ECLAIRAGE RADIAL, SOIT:  *
C*                                                                   *
C*                                                                   *
C*    LES CONCENTRATIONS PRISES EN COMPTE AINSI QUE LES EQUATIONS     *
C*    DU MODELE SONT DEFINIES DANS LA NOTE TECHNIQUE TN 19.2.         *
C*    LES ESPECES SUIVANTES SONT CONSIDEREES:                        *
C*                                                                   *
C*    XA : BIOMASSE ACTIVE                                           *
C*    EPS: EXOPOLYSACCHARIDE                                         *
C*    XT : BIOMASSE TOTALE                                           *
C*    XV : BIOMASSE VEGETATIVE                                       *
C*    CH : CHLOROPHYLLE                                              *
C*    PC : PHYCOCYANINE                                              *
C*    P  : PROTEINES                                                 *
C*    N  : NITRATE                                                   *
C*    S  : SULFATE                                                   *
C*                                                                   *
C*                                                                   *
C*    ATTENTION!!!                                                   *
C*    ___________                                                    *
C*                                                                   *
C*    LE MODELE PEUT PRENDRE EN COMPTE LES LIMITATIONS PAR LA LUMIERE, *
C*    LES NITRATES, ET LES SULFATES, MAIS PEUT DONNER UNE SOLUTION    *
C*    VIABLE POUR LA BIOMASSE VEGETATIVE EN CONTINU ALORS QU'IL N'Y A *
C*    PLUS D'AZOTE OU DE SOUFRE DANS LE REACTEUR. EN REALITE, DANS CE *
C*    CAS, IL Y A LESSIVAGE DU REACTEUR CAR LA BIOMASSE VEGETATIVE NE *
C*    SE DIVISE PLUS BIEN QU'ELLE SYNTHETISE DES RESERVES            *
C*    INTRACELLULAIRES.                                              *
C*                                                                   *
C*    CE PROGRAMME UTILISE DES VECTEURS DIMENSIONNES A N=9, CE QUI    *
C*    REPRESENTE LES 9 EQUATIONS INTEGRO-DIFFERENTIELLES DU MODELE    *
C*    DONNE DANS LA NOTE TECHNIQUE TN 19.2; DANS L'ORDRE:            *
C*    rXT, rXA, rCH, rPC, rP, rN, rS, rXV, rEPS. LES DEUX VARIABLES   *
C*    RXA ET REPS SONT DES VARIABLES INTERNES AU SUBROUTINE DERIV ET  *
C*    REPRESENTENT LA BIOMASSE ACTIVE ET LE POLYSACCHARIDE EN        *
C*    LIMITATION LUMIERE SEULE.                                      *
C*                                                                   *
C*********************************************************************
C
C     declarations
      INTEGER i
      DOUBLE PRECISION Y(9),CI(11),F(9),CI1(11)
C
      COMMON/PHOTO1/CI1
C
      do 1 i=1,11
         CI1(i) = CI(i)
 1    continue
```

```
C
      CALL DERIV(Y,F)
C
      return
      end

CDEB  deriv
      SUBROUTINE DERIV(Y,F)
C**********************************************************************
C     SUBROUTINE CONTENANT LES DERIVEES DU SYSTEME D'EQUATIONS INTEGRO-
C     DIFFERENTIELLES DU MODELE (TN 19.2) DANS LE VECTEUR F(N).
C     LES CONCENTRATIONS DE CHAQUE ESPECE SONT FOURNIES DANS LE VECTEUR
C     Y(N).
C
C     CE SUBROUTINE FAIT APPEL A UNE FUNCTION RF(G,D) CALCULANT LES
C     RACINES D'UNE EQUATION PAR LA METHODE DE REGULA FALSI, ET A UNE
C     FUNCTION G(Z) OU SE TROUVE DEFINIE LA FONCTION.
C**********************************************************************
CFIN
C
C     declarations
      EXTERNAL G,RF
      INTRINSIC DSINH,DCOSH,DSQRT
      DOUBLE PRECISION G,RF,DSINH,DCOSH,DSQRT
      DOUBLE PRECISION Y(9),F(9),CI(11)
      DOUBLE PRECISION RT,WIV,PAR(16)
      DOUBLE PRECISION EA,ES,ALPHA,DELTA
      DOUBLE PRECISION PAS,XI,XS,XG,XD
      DOUBLE PRECISION ROL,R1,R2
      DOUBLE PRECISION SA,SB,SJXA,SJEPS
      DOUBLE PRECISION MUXA,MUEPS
      DOUBLE PRECISION RXA,REPS,REPS1,REPS2
      DOUBLE PRECISION GAMMA,A,PE,D

      COMMON/PHOTO1/CI
      COMMON/PHOTO2/PAR
      COMMON/PHOTO3/ALPHA,DELTA,RT

      RT=.045d0
      WIV=1.d0

      PAR(1)=150.d0
      PAR(2)=200.d0
      PAR(3)=.01d0
      PAR(4)=.162d0
      PAR(5)=.684d0
      PAR(6)=.45d0
      PAR(7)=1.852d0
      PAR(8)=20.d0
      PAR(9)=750.d0
      PAR(10)=5.3d-3
      PAR(11)=2.5d-4
      PAR(12)=.15d0
      PAR(13)=.55d0
      PAR(14)=.516d0
      PAR(15)=.022d0
      PAR(16)=.049d0

      EA=PAR(1)/(PAR(3)+PAR(4))*(Y(3)+Y(4))
      ES=PAR(2)*Y(8)
      ALPHA=DSQRT(EA/(EA+ES))
      DELTA=DSQRT(EA*(EA+ES))
```

```
C     RECHERCHE DU RAYON UTILE ECLAIRE.

C     LOCALISATION DES RACINES PAR LE THEOREME DE ROLLE ET CALCUL DES
C     RACINES PAR LA METHODE DE REGULA FALSI.
C_____________________________________________________________________

      PAS=RT/10.d0
      XI=1d-5
      XS=1d-5+PAS
 10   ROL=G(XI)*G(XS)
      IF(ROL.GT.0.d0) THEN
        IF(XS.GE.RT) THEN
          R1=1d-5
          R2=1d-5
          GOTO 20
        ENDIF
        XI=XS
        XS=XS+PAS
        GOTO 10
      ELSE
        XG=XI
        XD=XS
        R1=RF(XG,XD)
      ENDIF
      IF(XD.GE.RT) THEN
        R2=R1
        R1=1d-5
        GOTO 20
      ENDIF
      XI=XD
      XS=XD+PAS
 50   ROL=G(XI)*G(XS)
      IF(ROL.GT.0.d0) THEN
        IF(XS.GE.RT) THEN
          R2=R1
          R1=1d-5
          GOTO 20
        ENDIF
        XI=XS
        XS=XS+PAS
        GOTO 50
      ELSE
        XG=XI
        XD=XS
        R2=RF(XG,XD)
 20   ENDIF


C     CALCUL DES INTEGRALES DONNANT RXA ET REPS1.
C_____________________________________________________________________

      SA=1d-5
      SB=R1
      CALL SIMPSON(SA,SB,SJXA,SJEPS)
      MUXA=2.d0*PAR(6)*SJXA/(R1*R1)
      MUEPS=2.d0*PAR(7)*SJEPS/(R1*R1)
      SA=R2
      SB=RT
      CALL SIMPSON(SA,SB,SJXA,SJEPS)
      MUXA=MUXA+2.d0*PAR(6)*SJXA/(RT**2-R2**2)
      MUEPS=MUEPS+2.d0*PAR(7)*SJEPS/(RT**2-R2**2)
      GAMMA=WIV*((R1/RT)**2+((RT**2-R2**2)/RT**2))
      RXA=MUXA*GAMMA*Y(4)
      REPS1=MUEPS*GAMMA*Y(4)
```

```
C     CALCUL DE REPS2 PAR L'APPROCHE BIOCHIMIQUEMENT STRUCTUREE
C     (voir TN 19.2).
C-----------------------------------------------------------------------

      A=4.d0*CI(1)*ALPHA*DSINH(DELTA*RT)/(RT*(DCOSH(DELTA*RT)+
     *ALPHA*DSINH(DELTA*RT)))
      PE=1.222d-5*A+1.267d0
      REPS2=(29.33d0*(PE*2.874d0-3.568d0)*RXA/23.096d0)/
     *(3.33d0-PE*1.92d0)


C     CALCUL DE REPS PAR LA MOYENNE ARITHMETIQUE DE REPS1 ET REPS2
C-----------------------------------------------------------------------

      REPS=(REPS1+REPS2)/2.d0


C     DERIVEES DES 9 ESPECES CONSIDEREES PAR LE MODELE (voir TN 19.2).
C-----------------------------------------------------------------------

      D=CI(2)

C     BIOMASSE TOTALE
      F(1)=D*(CI(3)-Y(1))+RXA+REPS

C     BIOMASSE ACTIVE
      F(2)=D*(CI(4)-Y(2))+RXA*(Y(6)/(PAR(10)+Y(6)))*(Y(7)/(PAR(11)+
     *Y(7)))

C     CHLOROPHYLLE
      F(3)=D*(CI(5)-Y(3))+PAR(3)*RXA*(Y(6)/(PAR(10)+Y(6)))*(Y(7)/
     *(PAR(11)+Y(7)))

C     PHYCOCYANINE
      F(4)=D*(CI(6)-Y(4))+PAR(4)*RXA*((Y(6)/(PAR(10)+Y(6)))*(Y(7)/
     *(PAR(11)+Y(7)))-((PAR(10)/(PAR(10)+Y(6)))+(PAR(11)/
     *(PAR(11)+Y(7)))))

C     PROTEINES
      F(5)=D*(CI(7)-Y(5))+PAR(5)*RXA*((Y(6)/(PAR(10)+Y(6)))*(Y(7)/
     *(PAR(11)+Y(7)))-PAR(13)*(PAR(11)/(PAR(11)+Y(7))))

C     NITRATE
      F(6)=D*(CI(8)-Y(6))-PAR(14)*RXA*(Y(6)/(PAR(10)+Y(6)))*(Y(7)/
     *(PAR(11)+Y(7)))

C     SULFATE
      F(7)=D*(CI(9)-Y(7))-PAR(15)*RXA*(Y(6)/(PAR(10)+Y(6)))*(Y(7)/
     *(PAR(11)+Y(7)))-PAR(16)*REPS*(Y(6)/(PAR(10)+Y(6)))*(Y(7)/
     *(PAR(11)+Y(7)))

C     BIOMASSE VEGETATIVE
      RXV=RXA*(((Y(6)/(PAR(10)+Y(6)))*(Y(7)/(PAR(11)+Y(7))))+((Y(4)/
     *(PAR(12)+Y(4)*Y(4)))*((PAR(10)/(PAR(10)+Y(6)))+(PAR(11)/
     *(PAR(11)+Y(7))))))
      F(8)=D*(CI(10)-Y(8))+RXV

C     EXOPOLYSACCHARIDE
      F(9)=D*(CI(11)-Y(9))+REPS*(Y(6)/(PAR(10)+Y(6)))*(Y(7)/
     *(PAR(11)+Y(7)))+(RXA+REPS-RXV)*((PAR(10)/(PAR(10)+Y(6)))+
     *(PAR(11)/(PAR(11)+Y(7))))
```

```
      RETURN
      END



CDEB  rf
      DOUBLE PRECISION FUNCTION RF(XG,XD)
C**********************************************************************
C     SOUS-PROGRAMME RESOLVANT L'EQUATION G(X)=0 PAR LA METHODE REGULA-
C     FALSI.
C**********************************************************************
CFIN
C
C     declarations
      INTEGER I,ITMAX
      EXTERNAL G
      DOUBLE PRECISION G
      DOUBLE PRECISION XG,XD,X
      DOUBLE PRECISION YD,YG,Y
      DOUBLE PRECISION EPS
C
      PARAMETER (EPS=.001,ITMAX=100)
C
      YD=G(XD)
      YG=G(XG)
      DO 40 I=1,ITMAX
        X=(YD*XG-XD*YG)/(YD-YG)
        Y=G(X)
        IF(Y*YD)10,10,20
 10       YG=Y
          XG=X
          GO TO 30
 20       YD=Y
          XD=X
 30       IF(ABS(XD-XG).LT.EPS) GO TO 50
            IF(ABS(Y).LT.EPS) GO TO 60
 40   CONTINUE
 50   RF=(XD+XG)/2.d0
      RETURN
 60   RF=X
      RETURN
      END



CDEB  g
      DOUBLE PRECISION FUNCTION G(Z)
C**********************************************************************
C     DEFINITION DE LA FONCTION DONT ON CHERCHE UNE RACINE.
C**********************************************************************
CFIN
C
C     declarations
      INTRINSIC DCOSH,DSINH
      DOUBLE PRECISION DCOSH,DSINH
      DOUBLE PRECISION Z
      DOUBLE PRECISION CI(11)
      DOUBLE PRECISION RT,ALPHA,DELTA
C
      COMMON/PHOTO1/CI
      COMMON/PHOTO3/ALPHA,DELTA,RT
C
```

```
      G=RT*2.d0*DCOSH(DELTA*Z)/(Z*(DCOSH(DELTA*RT)+
     *ALPHA*DSINH(DELTA*RT)))- 1.d0/CI(1)
C
      RETURN
      END



CDEB  simpson
      SUBROUTINE SIMPSON(SA,SB,SJXA,SJEPS)
C*********************************************************************
C     SUBROUTINE DE CALCUL D'INTEGRALE PAR LA METHODE DE SIMPSON.
C*********************************************************************
CFIN
C
C     decalarations
      EXTERNAL SYXA,SYEPS
      DOUBLE PRECISION SYXA,SYEPS
      DOUBLE PRECISION SA,SB,SJXA,SJEPS
      DOUBLE PRECISION SP,SX
      INTEGER N,I,K
C
      IF(SA.EQ.SB) THEN
        SJXA=0.d0
        SJEPS=0.d0
        RETURN
      ENDIF
      N=5
      SP=(SB-SA)/dble(N)
      SX=SA
      SJXA=-SYXA(SX)
      SJEPS=-SYEPS(SX)
      SX=SB
      SJXA=SJXA+SYXA(SX)
      SJEPS=SJEPS+SYEPS(SX)
      DO 10 I=0,N-1
        SX=SA+DBLE(I)*SP
        SJXA=SJXA+2.d0*SYXA(SX)
        SJEPS=SJEPS+2.d0*SYEPS(SX)
 10   CONTINUE
      DO 20 K=0,N-1
        SX=SA+K*SP+SP/2.d0
        SJXA=SJXA+4.d0*SYXA(SX)
        SJEPS=SJEPS+4.d0*SYEPS(SX)
 20   CONTINUE
      SJXA=SJXA*SP/6.d0
      SJEPS=SJEPS*SP/6.d0
      RETURN
      END



CDEB  syxa
      DOUBLE PRECISION FUNCTION SYXA(SX)
C*********************************************************************
C     FONCTION A INTEGRER CONCERNANT LA BIOMASSE ACTIVE.
C*********************************************************************
CFIN
C
C     declarations
      INTRINSIC DCOSH,DSINH
      DOUBLE PRECISION DCOSH,DSINH
```

```
      DOUBLE PRECISION SX,PJ
      DOUBLE PRECISION CI(11),PAR(16),ALPHA,DELTA,RT
C
      COMMON/PHOTO1/CI
      COMMON/PHOTO2/PAR
      COMMON/PHOTO3/ALPHA,DELTA,RT
C
      PJ=RT*CI(1)*2.d0*DCOSH(DELTA*SX)/(SX*(DCOSH(DELTA*RT)+
     *ALPHA*DSINH(DELTA*RT)))
      SYXA=SX*PJ/(PAR(8)+PJ)
      RETURN
      END



CDEB  syeps
      DOUBLE PRECISION FUNCTION SYEPS(SX)
C*********************************************************************
C     FONCTION A INTEGRER CONCERNANT L'EXOPOLYSACCHARIDE.
C*********************************************************************
CFIN
C
C     declarations
      INTRINSIC DCOSH,DSINH
      DOUBLE PRECISION DCOSH,DSINH
      DOUBLE PRECISION SX,PJ
      DOUBLE PRECISION CI(11),PAR(16),ALPHA,DELTA,RT
C
      COMMON/PHOTO1/CI
      COMMON/PHOTO2/PAR
      COMMON/PHOTO3/ALPHA,DELTA,RT
C
      PJ=RT*CI(1)*2.d0*DCOSH(DELTA*SX)/(SX*(DCOSH(DELTA*RT)+
     *ALPHA*DSINH(DELTA*RT)))
      SYEPS=SX*PJ/(PAR(9)+PJ)
      RETURN
      END
```

```
C=======================================================================
C     PHOTOMEXF.F
C
C     This is a FORTRAN representation of the Spirulina system.
C     It works in conjunction with the the file SIMULINK.F.
C     To compile this system, use the syntax:
C          fmex simomex.f simulink.f
C
C=======================================================================
C  Written:
C       Ned Gulley         Apr 28, 1992
C  Overwritten:
C       Nathalie Fulget    Nov 22, 1993
C       Copyright (c) 1990-93 by The MathWorks, Inc.
C       All Rights Reserved
C=======================================================================
C
C=======================================================================
      SUBROUTINE SIZES(SIZE)
C     .. Array arguments ..
      INTEGER*4         SIZE(*)
C=======================================================================
C     Purpose:
C       Function to set size vector
C     Input arguments:
C       None
C     Output arguments:
C       SIZE      a vector of model sizes
C     Description:
C       Sizes returns a vector which determines model characteristics.
C       This vector contains the sizes of the state vector and other
C       parameters. More precisely,
C       SIZE(1)   number of continuous states
C       SIZE(2)   number of discrete states
C       SIZE(3)   number of outputs
C       SIZE(4)   number of inputs
C       SIZE(5)   number of discontinuous roots in the system
C       SIZE(6)   set to 1 if the system has direct feedthrough of its
C                 inputs, otherwise 0
C=======================================================================
C     .. Parameters ..
      INTEGER*4        NSIZES
      PARAMETER        (NSIZES=6)
C     .. Local scalars ..
C     .. Local arrays ..
C     .. Executable statements ..
      SIZE(1) = 9
      SIZE(2) = 0
      SIZE(3) = 9
      SIZE(4) = 11
      SIZE(5) = 0
      SIZE(6) = 0

      RETURN
      END

      SUBROUTINE INITCOND(X0)
C     .. Array arguments ..
      REAL*8            X0(*)
C=======================================================================
C     Purpose:
C       Function to set initial condition vector
C     Input arguments:
C       None
```

```
C     Output arguments:
C       X0       a vector of initial conditions
C=======================================================================
C     .. Local scalars ..
C     .. Local arrays ..
C     .. Executable statements ..
      X0(2) = 0.1
      X0(6) = 0.8
      X0(7) = 0.2
      X0(9) = 0.02
      X0(1) = X0(2)+X0(9)
      X0(3) = 0.01*X0(2)
      X0(4) = 0.162*X0(2)
      X0(5) = 0.684*X0(2)
      X0(8) = X0(2)

      RETURN
      END

      SUBROUTINE DERIVS(T, X, U, DX)
C     .. Scalar arguments ..
      REAL*8           T
C     .. Array arguments ..
      REAL*8           X(*), U(*), DX(*)
C=======================================================================
C     Purpose:
C       Function to return derivatives
C     Input arguments:
C       T        time
C       X        state vector
C       U        input vector
C     Output arguments:
C       DX       state vector derivatives
C     Remark:
C       The state vector is partitioned into continuous and discrete
C       states. The first states contain the continuous states, and
C       the last states contain the discrete states.
C=======================================================================
C     .. Local scalars ..
C     .. Local arrays ..
C     .. Executable statements ..
C
      CALL PHOTOLAB(DX,U,X)


      RETURN
      END

      SUBROUTINE DSTATES(T, X, U, XNEW)
C     .. Scalar arguments ..
      REAL*8           T
C     .. Array arguments ..
      REAL*8           X(*), U(*), XNEW(*)
C=======================================================================
C     Purpose:
C       Function to perform discrete state update
C     Input arguments:
C       T        time
C       X        state vector
C       U        input vector
C     Output arguments:
C       XNEW     next state values
C     Remark:
C       The state vector is partitioned into continuous and discrete
```

```
C       states. The first states contain the continuous states, and
C       the last states contain the discrete states.
C=======================================================================
C     .. Local scalars ..
C     .. Local arrays ..
C     .. Executable statements ..

      RETURN
      END

      SUBROUTINE OUTPUT(T, X, U, Y)
C     .. Scalar arguments ..
      REAL*8           T
C     .. Array arguments ..
      REAL*8           X(*), U(*), Y(*)
      INTEGER i
C=======================================================================
C     Purpose:
C       Function to return continuous outputs
C     Input arguments:
C       T        time
C       X        state vector
C       U        input vector
C     Output arguments:
C       Y        output vector
C     Remark:
C       The state vector is partitioned into continuous and discrete
C       states. The first states contain the continuous states, and
C       the last states contain the discrete states.
C=======================================================================
C     .. Local scalars ..
C     .. Local arrays ..
C     .. Executable statements ..

      DO 1 i=1,9
         Y(i) = X(i)
 1    CONTINUE

      RETURN
      END

      SUBROUTINE DOUTPUT(T, X, U, Y)
C     .. Scalar arguments ..
      REAL*8           T
C     .. Array arguments ..
      REAL*8           X(*), U(*), Y(*)
C=======================================================================
C     Purpose:
C       Function to return discrete outputs
C     Input arguments:
C       T        time
C       X        state vector
C       U        input vector
C     Output arguments:
C       Y        output vector
C     Remark:
C       The state vector is partitioned into continuous and discrete
C       states. The first states contain the continuous states, and
C       the last states contain the discrete states.
C       This procedure is called only if it is a sample hit.
C=======================================================================
C     .. Local scalars ..
C     .. Local arrays ..
C     .. Executable statements ..
```

```
      RETURN
      END

      SUBROUTINE TSAMPL(T, X, U, TS, OFFSET)
C     .. Scalar arguments ..
      REAL*8           T,TS,OFFSET
C     .. Array arguments ..
      REAL*8           X(*), U(*)
C=======================================================================
C     Purpose:
C       Function to return the sample and offset times
C     Input arguments:
C       T        time
C       X        state vector
C       U        input vector
C     Output arguments:
C       TS       sample time
C       OFFSET   offset time
C=======================================================================
C     .. Local scalars ..
C     .. Local arrays ..
C     .. Executable statements ..

      RETURN
      END


      SUBROUTINE SINGUL(T, X, U, SING)
C     .. Scalar arguments ..
      REAL*8           T
C     .. Array arguments ..
      REAL*8           X(*), U(*), SING(*)
C=======================================================================
C     Purpose:
C       Function to return singularities
C     Input arguments:
C       T        time
C       X        state vector
C       U        input vector
C     Output arguments:
C       SING     singularities
C=======================================================================
C     .. Local scalars ..
C     .. Local arrays ..
C     .. Executable statements ..

      RETURN
      END
```

```
      SUBROUTINE MEXFUNCTION(NLHS, PLHS, NRHS, PRHS)
C     .. Scalar arguments ..
      INTEGER          NLHS, NRHS
C     .. Array arguments ..
      INTEGER          PLHS(*), PRHS(*)
C=======================================================================
C   Purpose:
C      Glue routine for making FORTRAN MEX-file systems and blocks
C   Synopsis:
C      [sys,x0] = usersys(t,x,u,flag)
C   Arguments:
C   Description:
C      This file should be linked with the subroutines that return
C      derivatives, outputs, discrete states and sample times.
C      Use a syntax of the form:
C                fmex usersys.f simulink.f
C      where usersys.f is the name of the user-defined function.
C  Algorithm:
C      Usersys is a MEX-file
C=======================================================================
C  Written:
C      Aleksandar Bozin  Mar 03, 1992
C  Modifications, bug fixes and extensions:
C      Ned Gulley        Apr 29, 1992
C      Copyright (c) 1990-93 by The MathWorks, Inc.
C      All Rights Reserved
C=======================================================================
C     .. Parameters ..
      INTEGER*4       NSIZES
      PARAMETER       (NSIZES=6)
      REAL*8          HUGE
      PARAMETER       (HUGE=1.0E+33)
C     .. Local scalars ..
      INTEGER*4        I, X0, T, U, X, SYS, SIZEOUT, FLAG
C     .. Local arrays ..
      INTEGER*4        LSIZE(NSIZES)
      REAL*8           DSIZE(NSIZES)
C     .. External subroutines ..
      EXTERNAL         MXCOPYPTRTOREAL8, MXCOPYREAL8TOPTR, MEXERRMSGTXT
      EXTERNAL         SIZES, INITCOND
      EXTERNAL         DERIVS, DSTATES, OUTPUT, DOUTPUT, SINGUL
      EXTERNAL         TNEXT, DCOPY
C     .. External functions ..
      LOGICAL          SAMPLHIT
      EXTERNAL         SAMPLHIT
      INTEGER*4        MXCREATEFULL, MXGETPR
      EXTERNAL         MXCREATEFULL, MXGETPR
      DOUBLE PRECISION MXGETSCALAR
      EXTERNAL         MXGETSCALAR
      INTEGER*4        MXGETM, MXGETN
      EXTERNAL         MXGETM, MXGETN
C     .. Intrinsic functions ..
      INTRINSIC        IABS, MAX0, MIN0, DBLE
C     .. Scalars in common ..
      REAL*8           CURHIT, NXTHIT
C     .. Arrays in common ..
      INTEGER*4        SIZE(NSIZES)
C     .. Common blocks ..
      COMMON           /SIMLAB/CURHIT, NXTHIT, SIZE
C     .. Executable statements ..

C
C     Check validity of arguments
C
```

```
      IF (NRHS .EQ. 0) THEN
         IF (NLHS .GT. 2) THEN
            CALL MEXERRMSGTXT('Too many output arguments.')
         ENDIF
C
C     Special case FLAG=0, return sizes and initial conditions
C
         PLHS(1) = MXCREATEFULL(NSIZES, 1, 0)
         SIZEOUT = MXGETPR(PLHS(1))
         DO 100 I = 1,NSIZES
            LSIZE(I) = 0
  100    CONTINUE
         CALL SIZES(LSIZE)
         DO 150 I = 1,NSIZES
            SIZE(I) = LSIZE(I)
            DSIZE(I) = DBLE(LSIZE(I))
  150    CONTINUE
         CALL MXCOPYREAL8TOPTR(DSIZE, SIZEOUT, NSIZES)
         IF (NLHS .GT. 1) THEN
            PLHS(2) = MXCREATEFULL(SIZE(1)+SIZE(2), 1, 0)
            X0 = MXGETPR(PLHS(2))
            CALL INITCOND(%VAL(X0))
         ENDIF
         CURHIT = -HUGE
         NXTHIT = HUGE
         RETURN
      ENDIF
C
C     Right hand side arguments
C
      IF ((NLHS .GT. 2) .OR. (NRHS .NE. 4)) THEN
         CALL MEXERRMSGTXT('Wrong number of input arguments.')
      ENDIF
C
C     Check for correct dimensions of input arguments
C
      M = MXGETM(PRHS(4))
      N = MXGETN(PRHS(4))
      IF ((M .NE. 1) .OR. (N .NE. 1)) THEN
         CALL MEXERRMSGTXT('Flag must be a scalar variable.')
      ENDIF
      FLAG = INT(MXGETSCALAR(PRHS(4)))
      IF (FLAG .EQ. 0) THEN
         IF (NLHS .GT. 2) THEN
            CALL MEXERRMSGTXT('Too many output arguments.')
         ENDIF
C
C        Special case FLAG=0, return sizes and initial conditions
C
         PLHS(1) = MXCREATEFULL(NSIZES, 1, 0)
         SIZEOUT = MXGETPR(PLHS(1))
         DO 200 I = 1,NSIZES
            LSIZE(I) = 0
  200    CONTINUE
         CALL SIZES(LSIZE)
         DO 250 I = 1,NSIZES
            SIZE(I) = LSIZE(I)
            DSIZE(I) = DBLE(LSIZE(I))
  250    CONTINUE
         CALL MXCOPYREAL8TOPTR(DSIZE, SIZEOUT, NSIZES)
         IF (NLHS .GT. 1) THEN
            PLHS(2) = MXCREATEFULL(SIZE(1)+SIZE(2), 1, 0)
            X0 = MXGETPR(PLHS(2))
            CALL INITCOND(%VAL(X0))
```

```
         ENDIF
         CURHIT = -HUGE
         NXTHIT = HUGE
         RETURN
      ENDIF
C
C     Error checking (can be omitted for speed but may cause
C     segmentation faults if called with the wrong sizes of
C     arguments)
C
      IF ((NLHS .GT. 1) .OR. (NRHS .NE. 4)) THEN
         CALL MEXERRMSGTXT('Too many output arguments.')
      ENDIF
C
C     Time parameter
C
      M = MXGETM(PRHS(1))
      N = MXGETN(PRHS(1))
      IF ((M .NE. 1) .OR. (N .NE. 1)) THEN
         CALL MEXERRMSGTXT('Time must be a scalar variable.')
      ENDIF
C
C     State vector
C
      M = MXGETM(PRHS(2))
      N = MXGETN(PRHS(2))
      IF ((M*N) .NE. SIZE(1)+SIZE(2)) THEN
         CALL MEXERRMSGTXT('State vector of wrong size.')
      ENDIF
C
C     Input vector
C
      M = MXGETM(PRHS(3))
      N = MXGETN(PRHS(3))
      IF ((M*N) .NE. SIZE(4)) THEN
         CALL MEXERRMSGTXT('Input vector of wrong size.')
      ENDIF
C
C     Check flag value and return appropriate vector
C
      T = MXGETPR(PRHS(1))
      X = MXGETPR(PRHS(2))
      U = MXGETPR(PRHS(3))
      IF (IABS(FLAG) .GT. 5) THEN
         CALL MEXERRMSGTXT('Not a valid flag number.')
      ENDIF
      GOTO (400,500,600,700,800) IABS(FLAG)
C
C     Flag is 1 or -1, return state derivatives
C
  400 CONTINUE
         PLHS(1) = MXCREATEFULL(SIZE(1), 1, 0)
         SYS = MXGETPR(PLHS(1))
         CALL DERIVS(%VAL(T), %VAL(X), %VAL(U), %VAL(SYS))
         GOTO 900
C
C     Flag is 2 or -2, return discrete states
C
  500 CONTINUE
         PLHS(1) = MXCREATEFULL(SIZE(2), 1, 0)
         SYS = MXGETPR(PLHS(1))
         IF (SAMPLHIT(%VAL(T))) THEN
            CALL DSTATES(%VAL(T), %VAL(X), %VAL(U), %VAL(SYS))
         ELSE
```

```
            CALL DCOPY(%VAL(X), SIZE(1)+1, %VAL(SYS), SIZE(2))
         ENDIF
         GOTO 900
C
C     Flag is 3, return system outputs
C
  600 CONTINUE
         PLHS(1) = MXCREATEFULL(SIZE(3), 1, 0)
         SYS = MXGETPR(PLHS(1))
         IF (SAMPLHIT(%VAL(T))) THEN
            CALL DOUTPUT(%VAL(T), %VAL(X), %VAL(U), %VAL(SYS))
         ENDIF
         CALL OUTPUT(%VAL(T), %VAL(X), %VAL(U), %VAL(SYS))
         GOTO 900
C
C     Flag is 4, return next time interval for update
C
  700 CONTINUE
         PLHS(1) = MXCREATEFULL(1, 1, 0)
         SYS = MXGETPR(PLHS(1))
         IF (SIZE(2) .GT. 0) THEN
            CALL TNEXT(%VAL(T), %VAL(X), %VAL(U), %VAL(SYS))
         ENDIF
         CALL MXCOPYREAL8TOPTR(NXTHIT, SYS, 1)
         GOTO 900
C
C     Flag is 5, return the values of the system root functions
C
  800 CONTINUE
         PLHS(1) = MXCREATEFULL(SIZE(5), 1, 0)
         SYS = MXGETPR(PLHS(1))
         CALL SINGUL(%VAL(T), %VAL(X), %VAL(U), %VAL(SYS))
         GOTO 900
C
C     Last card of MEXFUNCTION
C
  900 CONTINUE
      RETURN
      END

      REAL*8 FUNCTION HITFCN(TS, OFFSET, T)
C     .. Scalar arguments ..
      REAL*8           TS, OFFSET, T
C=======================================================================
C     Description:
C       Function to calculate the next sample time
C     Input arguments:
C       T        time
C       OFFSET   offset time
C       TS       sample time
C     Remark:
C       This function should not be called by the user.
C=======================================================================
C     .. Parameters ..
      INTEGER*4       NSIZES
      PARAMETER       (NSIZES=6)
      REAL*8          HUGE
      PARAMETER       (HUGE=1.0E+33)
C     .. Local scalars ..
      REAL*8          NSAMPL
C     .. Intrinsic functions ..
      INTRINSIC        AINT
C     .. Executable statements ..
```

```
      NSAMPL = (T-OFFSET)/TS
      HITFCN = OFFSET+(1.0+AINT(NSAMPL+1.0E-13*(1.0+NSAMPL)))*TS
      RETURN
      END

      SUBROUTINE TNEXT(T, X, U, TN)
C     .. Scalar arguments ..
      REAL*8           T
C     .. Array arguments ..
      REAL*8           X(*), U(*), TN(*)
C=======================================================================
C     Description:
C       Function to return the next sample time
C     Input arguments:
C       T        time
C       X        state vector
C       U        input vector
C     Output arguments:
C       TN       a scalar which contains the next sample time
C     Remark:
C       This function should not be called directly by the user.
C=======================================================================
C     .. Parameters ..
      INTEGER*4       NSIZES
      PARAMETER       (NSIZES=6)
      REAL*8          HUGE
      PARAMETER       (HUGE=1.0E+33)
C     .. Local scalars ..
      REAL*8          TS, OFFSET
C     .. External subroutines ..
      EXTERNAL         TSAMPL
C     .. External functions ..
      DOUBLE PRECISION HITFCN
      EXTERNAL         HITFCN
C     .. Scalars in common ..
      REAL*8           CURHIT, NXTHIT
C     .. Arrays in common ..
      INTEGER*4        SIZE(NSIZES)
C     .. Common blocks ..
      COMMON           /SIMLAB/CURHIT, NXTHIT, SIZE
C     .. Executable statements ..

      CALL TSAMPL(T, X, U, TS, OFFSET)
      IF (CURHIT .EQ. -HUGE) THEN
         CURHIT = HITFCN(TS, OFFSET, T-TS)
      ELSE
         CURHIT = NXTHIT
      ENDIF
      NXTHIT = HITFCN(TS, OFFSET, T)
      TN(1) = NXTHIT

      RETURN
      END

      LOGICAL FUNCTION SAMPLHIT(T)
C     .. Scalar arguments ..
      REAL*8           T
C=======================================================================
C     Purpose:
C       Function to check whether it is a sample hit or not
C     Input arguments:
C       T        time
C     Description:
C       This functions returns a boolean variable which can be used
```

```
C       to decide whether a sample hit occured or not.
C     Remark:
C       Called internally, but can be used by the user as well.
C=======================================================================
C     .. Parameters ..
      INTEGER*4       NSIZES
      PARAMETER       (NSIZES=6)
C     .. Scalars in common ..
      REAL*8          CURHIT, NXTHIT
C     .. Arrays in common ..
      INTEGER*4       SIZE(NSIZES)
C     .. Common blocks ..
      COMMON          /SIMLAB/CURHIT, NXTHIT, SIZE
C     .. Executable statements ..

      IF (T .EQ. CURHIT) THEN
         SAMPLHIT = .TRUE.
      ELSE
          SAMPLHIT = .FALSE.
      ENDIF

      RETURN
      END

      SUBROUTINE DCOPY(X, OFFSET, Y, N)
C     .. Scalar arguments ..
      INTEGER         OFFSET, N
C     .. Array arguments ..
      REAL*8          X(*), Y(*)
C=======================================================================
C     Purpose:
C       Function to copy one array into another
C     Input arguments:
C       X       input array
C       OFFSET  an index of the first element to be copied
C       N       number of elements to be copied
C     Output arguments:
C       Y       a vector containing a copy of the input vector
C=======================================================================
C     .. Local scalars ..
      INTEGER         I, IB
C     .. Executable statements ..

      IB = OFFSET
      DO 100 I = 1,N
          Y(I) = X(IB)
          IB = IB+1
  100 CONTINUE

      RETURN
      END
```

```
function [sys,x0]=C_ent(t,x,u,flag)


if flag==0
   sys =[0 0 9 4 0 1];
   x0=[];
elseif flag==3
   sys(1)=u(1)+u(4);
   sys(2)=u(1);
   sys(3)=0.01*u(1);
   sys(4)=0.162*u(1);
   sys(5)=0.684*u(1);
   sys(6)=u(2);
   sys(7)=u(3);
   sys(8)=u(1);
   sys(9)=u(4);
else
   sys=[];
end
```

```
function [sys,x0]=EB_FR(t,x,u,flag)


if flag==0
   sys =[0 0 1 10 0 1];
   x0=[];
elseif flag==3
   Ea=871;
   Es=200;
   zg=0;
   zpc=u(4)/u(2);
   zch=u(3)/u(2);
   za=zpc+zch;

   Rb=0.0095;
   R3=0.0115;
   R2=0.02585;
   R1=0.0302;
   R=0.048;

% Calcul de alpha
   alpha=sqrt(za*Ea/(za*Ea+Es*(1+zg)));

% Calcul de delta
   delta=(za*Ea+Es*(1+zg))*u(2)*alpha*R;

   FR3=u(10)*Rb/(pi*R3);

   delta3=delta*R2/R;
   z3=R3/R2;

   FR2=FR3/(2*alpha)*(cosh(delta3)+alpha*sinh(delta3))/sinh(delta3*z3)*z3;

   FR1=FR2*R2/R1;
   z1=R1/R;

   FR=FR1/(2*alpha)*(cosh(delta)+alpha*sinh(delta))/sinh(delta*z1)*z1;

   sys(1)=FR;
else
   sys=[];
end
```