AN EMBEDDED SYSTEM IMPLEMENTATION OF A PREDICTIVE ALGORITHM FOR A BIOPROCESS

Florin Stîngă*, Marius Marian**, Valentin Kese***, Lucian Bărbulescu** and Emil Petre*
*Department of Automation and Electronics
**Department of Computers and Information Technologies
Faculty of Automation, Computers and Electronics
University of Craiova
***Softronic Group
Craiova, Romania
E-mail: [florin, epetre]@automation.ucv.ro,
E-mail: [marius.marian, lucian.barbulescu]@cs.ucv.ro,
E-mail: kesevalentin@gmail.com

KEYWORDS
Bioprocess, Model predictive control, embedded implementation.

INTRODUCTION

The modeling and control of biotechnological processes is an actual and challenging problem due to their complicated structure. It is well known that these processes are dealing with living organisms that evolve over a long time or at the smallest change in their environment can become highly sensitive. Therefore the bioprocesses are characterized by highly nonlinear and uncertain dynamics, and their mathematical model is complex (G. Bastin and D. Dochain 1990; O. Bernard et al. 2011; F. Mairet et al. 2011). One of these processes whose importance resides in strict environmental rules is the wastewater treatment process; these rules are imposed in order to limit the quantity of toxic matter released in industrial and urban effluents. Nevertheless, its main drawback is the production of carbon dioxide (CO2) and its easy destabilization to input variations. For CO_2 mitigation, a solution consist in the growth of some microalgae populations that by using light as source of energy are able to assimilate inorganic forms of carbon and to convert them into requisite organic substances for cellular functions, generating at the same time oxygen (O_2). In what concerns the control of these processes, during the last years, numerous control strategies were developed: linearizing feedback (I. Neria-González et al 2009), adaptive and robust-adaptive (D. Selisteanu et al. 2007), predictive control (S. Tebbani et al. 2014), so on. Moreover, the widespread use of the embedded systems, based on microcontrollers, offered the hardware support for testing and implementing such complex control algorithms.

The paper presents a solution for implementing a model predictive control (MPC) for a continuous photo-bioreactor used for the growth of some microalgae that have the ability to use CO_2 as carbon source and, together with the solar energy, to biosynthesize various components, generating O_2. A widely used software platform for modeling, simulating and then, getting the real-time code is MATLAB/Simulink environment with the Real-Time Workshop plug-in (a.k.a Automatic Code Generation) (MathWorks 2016). However, the use of this proprietary software, may be limiting due to their costs, the number of the necessary equivalent *embedded functions – microcontroller level* translate code, and, also by the classes of microcontrollers that are supported by this application.

The model-based predictive control has been adopted in industry as an effective control strategy due to its capabilities to generate an optimal control input, and also to tackle the constraints on states, outputs and inputs. The control strategies are based on solving on-line (or off-line), at each sampling instant, a mathematical optimization problem based on a dynamical model of the plant (M. Morari and J.H. Lee, 1999). Over the years, different predictive control strategies were proposed (E. F. Camacho and C. Bordons 2004), (Q. Mayne and. E. C. Kerrigan 2007), or for hybrid systems (M. Lazar 2006), directly related to the wider area of application were used. The complexity of the algorithm resides in a mathematical optimization problem for which the feasibility may be ensured at each sampling time. Large number of variables may be involved leading to considerable computation effort and larger times for the optimal solution. This disadvantage, in real time applications, can be overcome by efficient algorithms that provide the essential computational routines required. The implementation of the MPC strategy, consider that the optimization problem is solved by means of a Hildreth's procedure. The obtained C code is tested under the same considered hypothesis on real time-platforms.

MATHEMATICAL MODEL

We consider the following photoautotrophic process which describes the growth of the green alga *C.*

Proceedings 31st European Conference on Modelling and Simulation ©ECMS Zita Zoltay Paprika, Péter Horák, Kata Váradi, Péter Tamás Zwierczyk, Ágnes Vidovics-Dancs, János Péter Rádics (Editors)
ISBN: 978-0-9932440-4-9/ ISBN: 978-0-9932440-5-6 (CD)

reinhardtii in a photobioreactor under light limiting conditions (F. Stinga and. E. Petre 2016):

$$\begin{aligned} \dot{x}(t) &= f\left(x(t)\right) + g\left(x(t)\right)u \\ y(t) &= h\left(x(t)\right) \end{aligned} \tag{1}$$

where $u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T = \begin{bmatrix} D & G_{in}^{CO_2} \end{bmatrix}^T$,

$x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^T =$

$= \begin{bmatrix} X & C_{TIC} & C_{O_2} & y_{out}^{CO_2} & y_{out}^{O_2} \end{bmatrix}^T$,

$$f(x) = \begin{bmatrix} \langle \tau_x \rangle \\ -\langle \tau_{TIC} \rangle + N_{CO_2} \\ \langle \tau_{O_2} \rangle + N_{O_2} \\ -G_{out} - V_l N_{CO_2} \\ -G_{out} - V_l N_{O_2} \end{bmatrix}, \quad g(x) = \begin{bmatrix} -x_1 & 0 \\ C_{TIC,i} - x_2 & 0 \\ -x_3 & 0 \\ 0 & \frac{RT_a}{PV_g} \\ 0 & \frac{RT_a c_3}{PV_g} \end{bmatrix},$$

$$h(x) = \begin{bmatrix} x_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_4 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T$$

where X is the biomass concentration, C_{TIC} is total inorganic carbon concentration, and C_{O_2} is the dissolved oxygen concentration, $y_{out}^{CO_2}$ and $y_{out}^{O_2}$ are the molar fractions of CO_2 and O_2 in the outlet gas, D is the dilution rate, $C_{TIC,i}$ is the concentrations of TIC in the feed, R is the universal gas constant, V_g and V_l are the gas and liquid volume of the photobioreactor, P is the total pressure in the gas phase, T_a is the temperature, $G_{in}^{CO_2}$ and $G_{in}^{O_2}$ are the CO_2 and O_2 feeding flow rates, respectively, and G_{out} are the total output flow rate. Also, the expressions of the global volumetric rate (τ_x), the local value of the irradiance ($I(z)$), TIC consumption rate (τ_{TIC}), the oxygen production rate (τ_{O_2}), the CO_2 and O_2 mass transfer rates to the liquid phase (N_{CO_2} and N_{O_2}), the total output flow rate, the feeding flow rate of the oxygen, are expressed as in (G.A. Ifrim et al. 2013; S. Tebbani et al. 2015; F. Stinga and. E. Petre 2016).

According with the defined outputs and inputs of the system, the control objective is to maintain biomass concentration in the photobioreactor at certain setpoints and to minimize the quantity of CO_2 in the output flux, that is $y = \begin{bmatrix} X & y_{out}^{CO_2} \end{bmatrix}^T$ using the dilution rate and the feeding flow rate of CO_2 as manipulated variables, i.e. $u = \begin{bmatrix} D & y_{out}^{CO_2} \end{bmatrix}^T$. Therefore we have a multivariable control problem with two outputs and two inputs.

The MPC strategy requires solving the following constraint optimization problem, and applying the optimal control action based on the receding horizon strategy (L. Wang 2009):

$$\begin{aligned} &\min_{\Delta U} \left(\frac{1}{2} \Delta U^T H \Delta U + f^T \Delta U \right) \\ &subject\ to \left(M \Delta U \le N \right) \\ &\begin{cases} \tilde{x}(k+1) = \tilde{A}\tilde{x}(k) + \tilde{B}\Delta u(k) \\ \tilde{y}(k) = \tilde{C}\tilde{x}(k) \end{cases} \end{aligned} \tag{2}$$

where, the model of the system is expressed in the difference of the state and control variables in order to obtain an integral action in predictive formulation, and:

$$H = \Phi^T \bar{Q} \Phi + \bar{H} \tag{3}$$

$$f = \Phi_F \tilde{x} - \Phi_R R^* \tag{4}$$

with, $\Phi_F = \Phi^T \bar{Q} F$, $\Phi_R = \Phi^T \bar{Q}$,

$$F = \begin{bmatrix} \left(\tilde{C}\tilde{A}\right)^T & \left(\tilde{C}\left(\tilde{A}\right)^2\right)^T & \dots & \left(\tilde{C}\left(\tilde{A}\right)^{N_p}\right)^T \end{bmatrix}^T$$

$$\Phi = \begin{bmatrix} \tilde{C}\tilde{B} & o_{p\times m} & \dots & o_{p\times m} \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & \dots & o_{p\times m} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{C}\left(\tilde{A}\right)^{N_p-1}\tilde{B} & \tilde{C}\left(\tilde{A}\right)^{N_p-2}\tilde{B} & \dots & \tilde{C}\left(\tilde{A}\right)^{N_p-N_c}\tilde{B} \end{bmatrix} \tag{5}$$

$$\tilde{x}(k) = \begin{bmatrix} \Delta x(k)^T & y(k) \end{bmatrix}^T, \ \Delta x(k) = x(k) - x(k-1),$$

$$\tilde{A} = \begin{bmatrix} A_d & o_{p\times n}^T \\ C_d A_d & I_{p\times p} \end{bmatrix}, \ \tilde{B} = \begin{bmatrix} B_d \\ C_d B_d \end{bmatrix}, \ \tilde{C} = \begin{bmatrix} o_{p\times n} & I_{p\times p} \end{bmatrix}$$

$$\Delta u(k) = u(k) - u(k-1),$$

$$\Delta U = \begin{bmatrix} \Delta u(k) & \Delta u(k+1) & \dots & \Delta u(k+N_c-1) \end{bmatrix}^T$$

N_c is the control horizon, R^* is column vector with $p \cdot N_p$ elements of set points, N_p is the prediction horizon, $\bar{Q}$ is positive definite error weight matrix, $\bar{H}$ is a $(m \times N_c) \times (m \times N_c)$ diagonal weight matrix used as tuning parameter for closed loop performance, m is the number of inputs, p is the number of outputs, n is the number of states, $o_{p\times n}$ is a zero matrix, $I_{p\times p}$ is the identity matrix, (A_d, B_d, C_d) are the matrices of the discrete-time state-space representation of the initial system (the discrete-time model of the system was obtained by means of Euler approximation), and $(x(k), u(k), y(k))$ are the state, input and output variables of the discrete-time model of the system,

$$M = \begin{bmatrix} -\Phi \\ -\Omega_1 \\ \Phi \\ \Omega_1 \end{bmatrix},\ N = \begin{bmatrix} -Y_{\min} + Fx^a(k) \\ -\Delta U_{\min} + \Omega_2 u(k-1) \\ Y_{\max} - Fx^a(k) \\ \Delta U_{\max} - \Omega_2 u(k-1) \end{bmatrix}, \quad (6)$$

Ω_1 is a $(m \cdot N_c) \times (m \cdot N_c)$ lower triangular identity matrix, Ω_2 is a column matrix with N_c identity matrix $I_{m \times m}$, $\Delta U_{\min}$, $\Delta U_{\max}$, $Y_{\min}$ and $Y_{\max}$ are column vectors with $(m \cdot N_c)$ elements of $\Delta u_{\min}$, $\Delta u_{\max}$, $y_{\min}$ and, respectively $y_{\max}$.

Remark 1: For the considered control strategy we use a linearizing model of the initial system (1), defined around certain equilibrium points (see F. Stinga and. E. Petre 2016).

The constrained programming problem (2) can be solved by using the different methods, such as: the projection method, the primal-dual method, the penalty functions, and so on.

In our implementation, we consider a primal-dual method, based on an element-by-element search which minimizes the objective function, expressed by the Hildtreth algorithm. At each complete Hildreth iteration, the minimization of the objective function (2), was obtained by adaptation of a single component λ_i of the Lagrange multiplier vector λ, so that (D.G. Luenberger 1969):

$$\lambda_i^{k+1} = \max\left(0, \omega_i^{k+1}\right) \quad (7)$$

with

$$\omega_i^{k+1} = -\frac{1}{p_{ii}}\left[d_i + \sum_{j=1}^{i-1} p_{ij}\lambda_j^{k+1} + \sum_{j=i+1}^{n} p_{ij}\lambda_j^k \right] \quad (8)$$

where p_{ij} is the *ijth* element in the matrix $P = MH^{-1}M^T$ and d_i is the *ith* element in the vector $D = N + MH^{-1}F$.

The optimal solution is expressed by the following relation, taking into account that the $\lambda_i \geq 0$, the optimal command sequence applied to the controlled system is given by:

$$\Delta U = -H^{-1}\left(F + M^T\lambda\right) \quad (9)$$

A microcontroller implementation procedure of the above programming algorithm is described in the next section.

PREDICTIVE ALGORITHM IMPLEMENTATION

We started the effort of getting the C based implementation of the above predictive control algorithms taking into account two possible strategies.

The first strategy starts with the mathematical model of the system in MATLAB/Simulink and then the predictive control algorithm can be generated by means of: MATLAB functions, embedded MATLAB functions, and Simulink library blocks. In what concerns the actual implementation of the predictive control algorithms, one can use the computing capabilities of the associated language (matrix initialization and manipulations, plus large data manipulation). The Real Time Workshop (RTW) plug-in can then be used to generate the C language source code from the Simulink model.

There are a few important remarks concerning this first strategy. The above mentioned plug-in RTW is licensed under a commercial license therefore some limits may apply on its usage. In what concern the classes of microcontrollers for which C code can be automatically generated, the number of C compilers that are supported, how large is the integration of legacy code, what limits are concerning the model size, to what extent we have a compatibility of the model and the generated code, and many more.

A second possible strategy involves first the design and then writing down the equivalent C source code for the predictive algorithm. We have opted for this second strategy after carefully considering the above points. The first reason was portability of source code written for different microcontroller/microprocessor architectures. We wanted to have a predictive algorithm C-based implementation that could be as easily portable as possible between different hardware architectures. Second reason was the usage preference of the authors for a free compiler such as GNU C compiler (GCC). Having said all these, we have continued to use the data sets generated by MATLAB/Simulink for testing the source code that we will present further on.

In our implementation we have designed and developed a mini library of C functions for common matrix manipulations such as: addition, subtraction, multiplication, computing the inverse or the transpose of a matrix. Furthermore, additional functions were written for initializing a matrix, copying a matrix, raising to a power, multiplication with a scalar or a vector.

Getting into the implementation of the predictive algorithm, we have to discuss a few aspects. In conformity with the mathematical model, we started with the calculation of the time-independent coefficients: H - as one can see in relation (3), Φ_R, Φ_F as in relation (4), $\left(\tilde{A}, \tilde{B}, \tilde{C}, A_d, B_d, C_d\right)$ defined in the above section, (Φ, F) according to relation (5), M and N as in relation (6).

```
#define EYE(sz, m) memset(m, 0, sizeof(m)); \
            for (i = 0; i < sz; i++) \
              m[i][i] = 1;
#define TRIL(sz, m) memset(m, 0, sizeof(m)); \
            for (i = 0; i < sz; i++) \
             for (j = 0; j <= i; j++) \
              m[i][j] = 1;
#define ZERO(m)    memset(m, 0 , sizeof(m));
#define INITM(sz, m1)  memset(m1, 0, sizeof(m1)); \
```

```
            for (i = 0; i < sz; i++) { \
             for (j = 0; j <= i; j++) {\
               m1[i][j] = -1; \
               m1[i+sz][j] = 1; \
             }\
            }
#define MAX(a, b)  (a > b ? (a) : (b))
double results[m], double DeltaU[Nc*m][1];
static double F[Np*p][x+p], Phi_F[Nc*m][x+p],
Phi_R[Nc*m][p], M[2*Nc*m+outcons*2*Np*p][Nc*m];
static double HInv[Nc*m][Nc*m],mHInv[Nc*m][Nc*m];
static double
mHInvMT[Nc*m][2*Nc*m+outcons*2*Np*p];
static double P[2*Nc*m + outcons*2*Np*p][2*Nc*m +
outcons* 2*Np*p];

double predc(double uLast[m], double r1[p], double
y[p], double dx[x])
{....
double A_tilda[x+p][x+p], B_tilda[x+p][m],
C_tilda[p][p+x], double C_tildaA_tildaP[p][x+p];
double Phi[np*p][nc*m], C_tildaA_tildaPB_tilda[p][m,
A_tildaP[x+p][x+p], PhiT[nc*m][np*p],
PhiTQ[nc*m][np*p], Phi_Phi[nc*m][nc*m]; dxy[n+p],
N[2*Nc*m+outcons*2*Np*p], f[Nc*m][1],
N[2*Nc*m+outcons*2*Np*p], Fdxy[Np*p][1],
eta[Nc*m][1], d[2*Nc*m+outcons*2*Np*p],
lambda[2*Nc*m + outcons*2*Np*m][1];
double lambda_p[2*Nc*m + outcons*2*Np*p][1],
lambda_m_lambda_p[2*Nc*m + outcons*2*Np*p][1],
lambda_m_lambda_pT[1][2*Nc*m + outcons*2*Np*p],
lambdaProd[1][1], a1, w;
int i, j, kk, km;

//Obtaining the vector F
copy(p, p+x, C_tilda, p, p+x, C_tildaA_tildaP, 0, 0);
   for (i = 0; i < Np; i++) {
multiply(p, p+x, p+x, p+x, C_tildaA_tildaP, A_tilda,
C_tildaA_tildaP);
copy(p, p+x, C_tildaA_tildaP, Np*p, x+p, F, i * p, 0);
copy(p, p+x, C_tildaA_tildaP, p, p+x, C_tildaA_tildaP,
0, 0); }

//Obtaining the matrix Φ
ZERO(Phi);
   for (i = 0; i < Np; i++) {
      for (j = 0; j < Nc; j++) {
         if (i < j) {
            break;
         } else if (i == j) {
multiply(p, p+x, p+x, m, C_tilda, B_tilda,
C_tildaA_tildaPB_tilda); }
else {
matPower(x+p, A_tilda, i-j, A_tildaP);
multiply(p, p+x, p+x, p+x, C_tilda, A_tildaP,
C_tildaA_tildaP);
multiply(p, p+x, p+x, m, C_tildaA_tildaP, B_tilda,
C_tildaA_tildaPB_tilda); }
copy(p, m, C_tildaA_tildaPB_tilda, Np*p, Nc*m, Phi, i *
p, j * m);}
copy(p, m, C_tildaA_tildaPB_tilda, Np*p, Nc*m, Phi, i *
p, j * m);
      }
   }

//Obtaining the matrix H
transpose(Np*p, Nc*m, Phi, PhiT);
multiply(Nc*m, Np*p, Np*p, Np*p, PhiT, Q, PhiTQ);
multiply(Nc*m, Np*p, Np*p, Nc*m, PhiTQ, Phi,
Phi_Phi);
multiply(Nc*m, Np*p, Np*p, x+p, PhiTQ, F, Phi_F);
for (i = 0; i < Nc*m; i++) {
      for (j = 0; j < p; j++) {
         Phi_R[i][j] = Phi_F[i][x + j];
      }
   }
EYE(Nc*m, eyeNcIn);
scalarMultiply(1, Nc*m, Nc*m, eyeNcIn, eyeNcInDiv);
add(Nc*m, Nc*m, Phi_Phi, eyeNcInDiv, H);

//Obtaining the matrix M
M13(Nc*m, M13);
   if (outcons) {
      for (i = 0; i < Np * p; i++) {
         for (j = 0; j < Nc * m; j++) {
M[2 * Nc * m + i][j] = -Phi[i][j];
M[2 * Nc * m + Np * p + i][j] = Phi[i][j];}}}

//Obtaining the matrix P
inverse(Nc*m, H, HInv);
scalarMultiply(-1, Nc*m, Nc*m, HInv, mHInv);
transpose(2*Nc*m + outcons*2*Np*p, Nc*m, M, MT);
multiply(Nc*m, Nc*m, Nc*m, 2*Nc*m +
outcons*2*Np*p,HInv, MT, HInvMT);
multiply(2*Nc*m + outcons*2*Np*p, Nc*m, Nc*m,
2*Nc*m + outcons*2*Np*p, M, HInvMT, P);

//Obtaining the vector f
memcpy(dxy, dx, x * sizeof(double));
memcpy(dxy+x, y, p * sizeof(double));
multiplyMatVec(Nc*m, x+p, x+p, Phi_F, dxy,
Phi_Fdxy);
multiplyMatVec(Nc*m, p, p, Phi_R, r1, Phi_Rr1);
sub(Nc*m, 1, Phi_Fdxy, Phi_Rr1, f);
multiplyMatVec(Np*p, x+p, x+p, f, dxy, Fdxy);

//Obtaining the vector N
for (j = 0; j < m; j++)
         {
      for (i = 0; i < Nc; i++)
            {
         N[i + j*Nc] = -umin[j] + uLast[j];
         N[i + j*Nc + m*Nc] = umax[j] - uLast[j]; }
         }

   if (outcons) {
      for (j = 0; j < p; j++)
         {
         for (i = 0; i < Np; i++)
         {
N[2*i + j + 2*Nc*m] = -ymin[j] + Fdxy[i*p+j][0];
N[2*i + j + (2*Nc*m+p*Np)] = ymax[j] -
Fdxy[i*p+j][0];}
         }}
```

```
//Implementation of the Hildreth's algorithm
multiply(Nc*m, Nc*m, Nc*m, 1, mHInv, f, eta);
   kk = 0;
   for (i = 0; i < 2*Nc*m + outcons*2*Np*p; i++) {
      sum = 0;
      for (j=0; j < Nc*m; j++) {
         sum += M[i][j] * eta1[j][0];
      }
      if (sum > N[i]) {
         kk = 1;
         break;
      }
   }
if (kk == 0) {
     for(i=0;i<m;i++)
     {
        results[i]=eta1[i][0];
     }}
   multiply(Nc*m, Nc*m, Nc*m, 1,HInv, f1, HInvf);
   multiply(2*Nc*m + outcons*2*Np*p, Nc*m, Nc*m, 1,
M, HInvf, d);
   for (i = 0; i < 2*Nc*m + outcons*2*Np*p; i++) {
      d[i][0] += N[i];
   }
   ZERO(lambda);
for (km = 0; km < 100; km ++) {
copy(2*Nc*m + outcons*2*Np*p, 1, lambda, 2*Nc*m +
outcons*2*Np*p, 1, lambda_p, 0, 0);
  for (i = 0; i < 2*Nc*m + outcons*2*Np*p; i++) {
         w = -P[i][i] * lambda[i][0] + d[i][0];
         for (j = 0; j < 2*Nc*m + outcons*2*Np*p ; j++)
{
            w += P[i][j] * lambda[j][0];
         }
         lambda[i][0] = MAX(0, -w / P[i][i]);
      }
sub(2*Nc*m + outcons*2*Np*p, 1, lambda, lambda_p,
lambda_m_lambda_p);
transpose(2*Nc*m + outcons*2*Np*p, 1,
lambda_m_lambda_p, lambda_m_lambda_pT);
multiply(1, 2*Nc*m + outcons*2*Np*p, 2*Nc*m +
outcons*2*Np*p, 1, lambda_m_lambda_pT,
lambda_m_lambda_p, lambdaProd);
      a1 = lambdaProd[0][0];
      if (a1 < 10e-8) {
         break;
      }}
multiply(Nc*m, 2*Nc*m + outcons*2*Np*p, 2*Nc*m +
outocns*2*Np*p, 1, mHInvMT, lambda,
mHInvMTlambda);
add(Nc*m, 1, eta1, mHInvMTlambda, DeltaU);
for (i=0; i<m;i++)
{
   results[i] = DeltaU[i][0];
}
return 0;
```

The parameters list of the ***predc*** function includes: $u(k-1), y(k), r(k),$ and $\Delta x(k)$.

This function is called iteratively in the *main()* function. It involves calculating the value of several parameters that change at every step based on state, input and output variables updates.

For defining constraints on the output variables the parameter *outcons* is used. The algorithm considers implicit constraints on the command variable. The complexity of the considered problem is influenced by two elements. First, it is the size of the two horizons: prediction and control, and second, by the number of system variables involved plus the number and type of constraints (hard or soft).

The final section of the function describes the implementation of the Hildreth's algorithm. In fact, the relations (7), (8), and (9) are translated in C level code. Also, the predictive control algorithm applying the principle of receding horizon strategy, such that only first m elements of ΔU are taken into account to form incremental optimal control. The rest of the elements are discarded, and at the next sampling instant a new control sequence is computed.

RESULTS

The considered bioprocess is a photoautotrophic growth of the green alga C. *reinhardtii* in a photobioreactor under specifying lighting conditions (see G.A. Ifrim et al 2013).

Also, for predictive algorithm the following input data are used (F. Stinga and E. Petre, 2016):

$$\tilde{A}=\begin{pmatrix} -0.03632 & 0 & 0 & 0 & 0 \\ -0.00049 & -0.09753 & 0 & 0.25117 & 0 \\ 0.00054 & 0 & -0.95000 & 0 & 1.08729 \\ 0 & 0.00969 & 0 & -4.36893 & -2.98859 \\ 0 & 0 & 0.18353 & -0.08390 & -4.14908 \end{pmatrix}$$

$$\tilde{B}=\begin{pmatrix} -0.42468 & 0 \\ 0.00036 & 0 \\ -0.00092 & 0 \\ 0 & 138.72761 \\ 0 & 38.84373 \end{pmatrix}, \quad \tilde{C}=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

$n=5, \quad m=2, \ p=2 \quad , \quad T=0.3\,h. \quad ,$

$\bar{Q}=\begin{bmatrix} 100\cdot I_{N_p\times N_p} & 1\cdot I_{N_p\times N_p} \end{bmatrix} \quad , \quad \bar{H}=1\cdot I_{mNc\times mNc} \quad ,$

$\Delta u_{\min}=\{0,0\} \ , \ \Delta u_{\max}=\{0.1,0.03\} \ , \quad y_{\min}=\{0,0\},$

$y_{\max}=\{0.7,0.06\}.$

The hardware platforms used for the implementation and testing of the described algorithm can be summarized as follows:

- PC platform: 2.4 GHz Intel i5 microprocessor and 4 GB of RAM memory, and two different microcontroller-based configurations:
- Atmel ARM Cortex M4 SAM4S16C: a 32-bit MCU, 120 MHz, 2KB of cache, 128 KB of embedded SRAM, 1 MB embedded flash memory, single-precision FPU;

- Atmel ARM Cortex M7 ATSAME70-XPLD Evaluation Board: a 32-bit MCU, 300 MHz, single- and double-precision FPU.

The software configurations include the following packages:

- 32-bit Windows 7 operating system
- MATLAB/SIMULINK 7.9.0
- Code::Blocks IDE 16.01
- GCC compiler v6.2
- AVR Atmel Studio IDE v6.2

Running the algorithm on different configurations hardware/software led to the following results. First, the simulation data obtained in MATLAB/Simulink was used as reference for all tests. The considered process outputs obtained after running the implemented algorithm on the Atmel ARM Cortex M7 ATSAME70-XPLD Evaluation Board and MATLAB/Simulink ($N_p = N_c = 5$) are presented in Figures 1 and 2.

The two figures present the evolution of the biomass concentration X and of the molar fraction of CO_2 , respectively, in two possible scenarios: with constraints on input variables (the blue line) and with constraints both on input and output variables (the red line).

Remark 2: Also, for the considered inputs of the system (the dilution rate and the feeding flow rate of CO_2), data set obtained for MATLAB environment and those generated by the microcontrollers, using ***predc*** function, coincide.

The tests yielded different execution times that are available in Table 1. The results are calculated by averaging the values returned by the real-time systems.

Obviously, the results obtained are influenced by the benchmark system. Large horizons of prediction and control, and also hard constrains both on command and output may lead to ever-increasing execution times.

These include the use of hardware-specific dedicated digital signal processing software libraries or improving the code for matrix manipulation. Another approach would be to take advantage of the compiler's levels of code optimization. Yet another approach would be to handle optimally the memory usage based on the specifics of the hardware platform used.

In these scenarios the data memory usage vary from 50.2% to 72.4%, instead the program memory usage is 8.2%. Certainly, the higher value from the data mamory usage is generated by the considered prediction and control horizons, and, also by the number of imposed constrained.

CONCLUSIONS

The paper presented an implementation of a C-based optimal control algorithm. The code was tested on two real time-platforms with different technical specifications. The obtained results are provided in order to demonstrate the effectiveness of the proposed implementation. We intend to continue working on optimizing the C implementation taking into account the specifics of target hardware platforms and also the software libraries available for the microcontrollers.

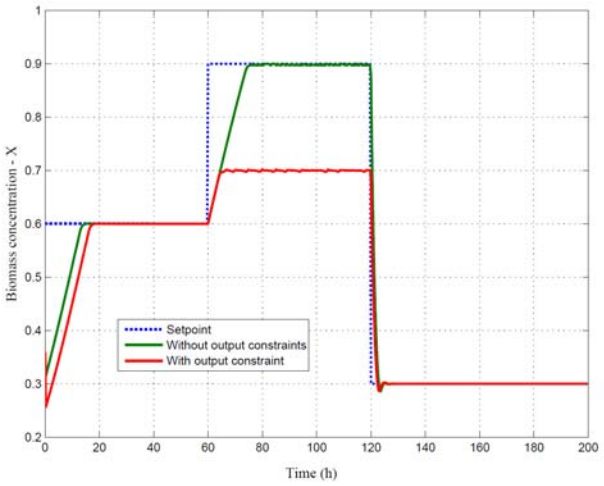

Figure 1: The time evolution of the first considered output of the system

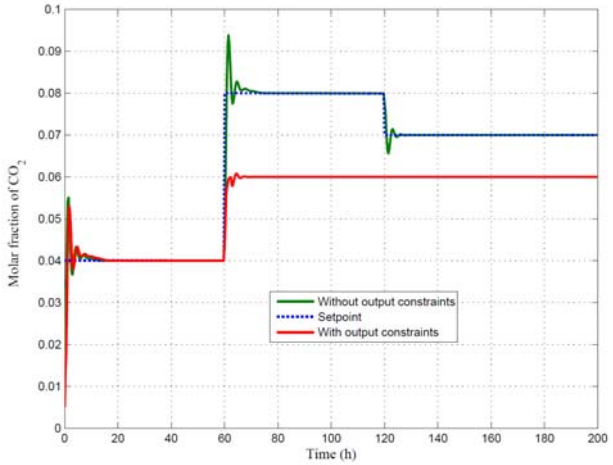

Figure 1: The time evolution of the second considered output of the system

TABLE I. THE EXECUTION TIMES

#	Implementation platform	N_p	N_c	Constraints	Measured time (sec.)
1	Atmel ARM Cortex M4	5	5	Only input	15×10^{-3}
				Both input and output	57.3×10^{-3}
2		8	5	Only input	38.2×10^{-3}
				Both input and output	133×10^{-3}
3	Atmel ARM Cortex M7	5	5	Only input	2.13×10^{-3}
				Both input and output	2.23×10^{-3}
4		8	5	Only input	2.25×10^{-3}
				Both input and output	5.27×10^{-3}

ACKNOWLEDGMENT

This work was supported by UEFISCDI, project BIOCON no. PN-II-PT-PCCA-2013-4-0070, 269/2014.

REFERENCES

G. Bastin and D. Dochain, On-line estimation and adaptive control of bioreactors, New York, NY: Elsevier, 1990.

O. Bernard, "Hurdles and challenges for modelling and control of microalgae for CO2 mitigation and biofuel production," *J. Process Control*, vol. 21, pp. 1378–1389, 2011.

S. Tebbani, F. Lopes, R. Filali, D. Dumur, and D. Pareau, "Nonlinear predictive control for maximization of CO_2 bio-fixation by microalgae in a photobioreactor," *Bioprocess Biosyst. Eng*., vol. 37, no. 12, pp. 83–97, 2014.

F. Mairet, O. Bernard, M. Ras, L. Lardon, and J.P. Steyeret, "Modeling anaerobic digestion of microalgae using ADM1," *Bioresource Technology*, vol. 102, pp 6823–6829, 2011.

I. Neria-González, A.R. Dominguez-Bocanegra, J. Torres, R. Maya-Yescas, and R. Aguilar-López, "Linearizing control based on adaptive observer for anaerobic continuous sulphate reducing bioreactors with unknown kinetics," *Chem. Biochem. Eng. Q.*, vol. 23, no. 2, pp. 179–185, 2009.

D. Selişteanu, E. Petre, and V. Răsvan, "Sliding mode and adaptive sliding mode control of a class of nonlinear bioprocesses," *Int. J. Adapt. Contr. & Signal Process*, vol. 21, no. 8-9, 2007, pp. 795-822.

M. Morari and J.H. Lee, "Model predictive control: past, present and future", in *Computers and Chemical Engineering*, 23(4-5), pp. 667-682, 1999.

MathWorks, Real Time Workshop, avaible online at https://www.mathworks.com/products/simulink-coder.html, 2016.

M. Lazar, Model predictive control of hybrid systems: stability and robustness, Phd. Thesis, 2006.

E. F. Camacho and C. Bordons, Model Predictive Control, 2nd edition, Springer, 2004.

Q. Mayne and. E. C. Kerrigan, "Tube-based robust nonlinear model predictive control", in Proceedings of the 7th IFAC Symposium – NOLCOS2007, Praetoria, 2007.

L. Wang, Model predictive control system design and implementaion using Matlab, Springer-Verlag London, 2009.

D.G. Luenberger, Optimization by vector space methods, John Wiley & Sons, Inc. , 1969.

G.A. Ifrim et al., "Multivariable feedback linearizing control of Chlamydomonas reinhardtii photoautotrophic growth process in a torus photobioreactor," *Chemical Eng. Journal*, vol. 218, pp. 191–203, 2013.

F. Stîngă and E. Petre, Predictive and feedback linearizing control of chlamydomonas reinhardtii photoautrotrophic growth process, Proceedings of the 30th European Conference on Modelling and Simulation, pp. 361-367, Regensburg, Germany, 2016.

FLORIN STÎNGĂ was born in Craiova, Romania. He received the B. Eng., M.S. and Ph.D. degrees in system engineering, all from University of Craiova, in 2000, 2003 and 2012. Currently, he is Lecturer in the Department of Automatic Control at the Faculty of Automation, Computers and Electronics, Craiova. His researches interested are in hybrid dynamical systems and predictive control.

MARIUS MARIAN received his Engineering degree in system and engineering from the University of Craiova (Romania), in 1998, and Ph.D. degree in information system security from Politecnico di Torino (Italy) in 2003. Currently, he is Lecturer in the Department of Computers and Information Technology at the Faculty of Automation, Computers and Electronics, University of Craiova. His research interests are in computer education, embedded systems and computer security.

KESE VALENTIN received the B.Eng. and M.Sc. diploma in system engineering from University of Craiova, in 1998 and 2008 respectively. Currently, he is an R&D system engineer at Softronic Group Craiova. His research interests are in microcontrollers, hardware, real-time systems, predictive control.

LUCIAN BĂRBULESCU was born in Craiova, Romania. He received the B. Eng. and M.S. degrees in computer science from the University of Craiova, in 2003 and 2005 and the Ph.D. degree in system engineering from the University "Lower Danube" of Galați in 2013. Currently, he is Lecturer in the Department of Computers and Information Technology at the Faculty of Automation, Computers and Electronics, University of Craiova. His research interests are in software systems for satellite dynamics.

EMIL PETRE received the Engineering degree in Automatic Control and Computers in 1977 and Ph.D. degree in Automatic Control in 1997 from University of Craiova, Romania. Since 1981 he is with the University of Craiova. Currently, he is Professor at the Department of Automatic Control, and from 2011, Dr. Petre serves as director of this department. His research interests include nonlinear systems, adaptive control, estimation and identification, control of bioprocesses, and neural control.