DYNOPT - DYNAMIC OPTIMISATION CODE FOR MATLAB

*M. Čižniar**, *D. Salhi*†, *M. Fikar**, *M.A. Latifi*†

* Department of Information Engineering and Process Control,FCFT-STU
† Laboratoire des Sciences du Génie Chimique,CNRS-ENSIC

Abstract

This paper describes a MATLAB package for dynamic optimisation of processes. The package dynopt searches for profiles of decision variables which optimise a given performance index under specified constraints. The package uses the method of orthogonal collocations on finite elements and several case studies are successfully tested. The MATLAB source code is freely available at the package web page `http://www.kirp.chtf.stuba.sk/~fikar/research/dynopt/dynopt.htm`**.**

1 Introduction

The objective of dynamic optimisation is to determine, in open loop control, a set of decision variable time profiles (pressure, temperature, flow rate, current, heat duty, ...) for a dynamic system that optimise a given performance index (or cost functional or optimisation criterion)(cost, time, energy, selectivity, ...) subject to specified constraints (safety, environmental and operating constraints). Optimal control refers to the determination of the best time-varying profiles in closed loop control.

The numerical methods used to find a deterministic solution of dynamic optimisation problems can be grouped into two categories: indirect and direct methods. In this work only direct methods are considered. In this category, there are two strategies: sequential method and simultaneous method. The sequential strategy, often called control vector parameterisation (CVP), consists in an approximation of the control trajectory by a function of only a few parameters and leaving the state equations in the form of the original differential algebraic equation (DAE) system [7]. In the simultaneous strategy, both the control and state variables are discretised using polynomials (e.g., Lagrange polynomials) of which the coefficients become the decision variables in a much larger Nonlinear Programming problem (NLP) [2].

In this paper, the method of orthogonal collocation on finite elements is developed based on [2; 9]. It is implemented purely in MATLAB as a collection of M files without any MEX/DLL interface. For the solution of NLP, standard Optimisation Toolbox is used. Its aim is to provide a simple interface to dynamic optimisation. Is is suitable for typical problems in chemical and biochemical industries. When compared to lower (compilator based) programming languages, a typical solution time is longer. However, the total time of learning and simplicity makes rapid development and integration with other packages possible. The source code of the package is available free at the web pages of authors.

The outline of the paper is as follows. In the next section we sketch a general NLP formulation for optimal control problems using orthogonal collocation on finite elements method, which is implemented in the dynamic optimisation package (*dynopt*). Section 3 shows how to define a given control problem with *dynopt*. In section 4, we present some known examples from literature dealing with chemical reactors which are then solved and discussed in section 5.

2 NLP Formulation Problem

Consider the following general dynamic optimisation problem for $t \in [0, t_f]$ which comprises a Meyer-type cost functional, a process dynamic model described by a set of ordinary differential equations (ODE), and equality and inequality constraints.

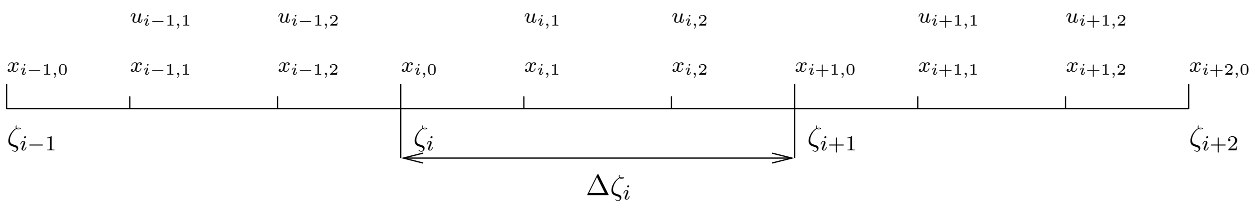

Figure 1: Finite-element collocation discretisation for state profiles, control profiles and element lengths

$$\min_{\boldsymbol{u}(t)} J[\boldsymbol{x}(t_f)] \tag{1}$$

such that

$$\begin{gathered}
\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[t, \boldsymbol{x}(t), \boldsymbol{u}(t)], \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\
\boldsymbol{h}[t, \boldsymbol{x}(t), \boldsymbol{u}(t)] = \boldsymbol{0} \\
\boldsymbol{g}[t, \boldsymbol{x}(t)\boldsymbol{u}(t)] \leq \boldsymbol{0} \\
\boldsymbol{x}(t)^L \leq \boldsymbol{x}(t) \leq \boldsymbol{x}(t)^U \\
\boldsymbol{u}(t)^L \leq \boldsymbol{u}(t) \leq \boldsymbol{u}(t)^U
\end{gathered}$$

where

$J[\boldsymbol{x}(t_f)]$ – objective function evaluated at final conditions,

$\boldsymbol{f}$ – vector of differential equations,

$\boldsymbol{h}$ – vector of equality constraints,

$\boldsymbol{g}$ – vector of inequality constraints,

$\boldsymbol{x}(t)$ – state vector,

$\boldsymbol{u}(t)$ – control vector,

$\boldsymbol{x}_0$ – process initial conditions,

${}^L,{}^U$ – lower and upper profile bounds.

In order to convert problem (1) into NLP problem, the orthogonal collocation on finite elements is used (Fig. 1). The optimal control problem (1) is then solved by complete parameterisation of both, the control and the state vectors [3; 9; 10]. That means that the control and state profiles on one time interval are approximated by linear combination of some basis functions (Lagrange polynomials (2), (3)).

$$\boldsymbol{x}_{K+1}(t) = \sum_{j=0}^{K} \boldsymbol{x}_{ij}\phi_j(t);\ \phi_j(t) = \prod_{k=0,j}^{K} \frac{(t - t_{ik})}{(t_{ij} - t_{ik})} \tag{2}$$
$$\text{in element } i,\ i = 1, \ldots, \text{NE}$$

$$\boldsymbol{u}_K(t) = \sum_{j=1}^{K} \boldsymbol{u}_{ij}\theta_j(t);\ \theta_j(t) = \prod_{k=1,j}^{K} \frac{(t - t_{ik})}{(t_{ij} - t_{ik})} \tag{3}$$
$$\text{in element } i,\ i = 1, \ldots, \text{NE}$$

Here $k = 0, j$ means that k starts from 0 and $k \neq j$, NE is the number of elements. Also $\boldsymbol{x}_{K+1}(t)$ is a $(K+1)$th order (deg $< K+1$) piecewise polynomial and $\boldsymbol{u}_K(t)$ is Kth order (deg $< K$) piecewise polynomial. The difference in orders is due to the existence of the initial conditions for $\boldsymbol{x}(t)$, for each element i. The times t_{ij} are given as roots of the Legendre polynomials on interval $[0, 1]$.

The problem (1) now becomes:

$$\min_{\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, \Delta\zeta_i} J(\boldsymbol{x}_f) \tag{4}$$

such that

$$\begin{gathered}
\boldsymbol{x}_{10} - \boldsymbol{x}_0 = \boldsymbol{0}, t_f - \sum_{i=1}^{NE} \Delta\zeta_i = 0 \\
\Delta\zeta \boldsymbol{r}_{ij} = \dot{\boldsymbol{x}}_{k+1}(t_{ij}) - \Delta\zeta_i \boldsymbol{f}(t_{ij}, \boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}) = \boldsymbol{0}, \\
i = 1, \ldots, \mathrm{NE} \quad j = 1, \ldots, K \\
\boldsymbol{x}_{i0} - \boldsymbol{x}_{K+1}^{i-1}(\zeta_i) = \boldsymbol{0}, \quad i = 2, \ldots, \mathrm{NE} \\
\boldsymbol{x}_f - \boldsymbol{x}_{K+1}^{NE}(\zeta_{NE+1}) = \boldsymbol{0} \\
\boldsymbol{u}_i^L \leq \boldsymbol{u}_K^i(\zeta_i) \leq \boldsymbol{u}_i^U, \quad i = 1, \ldots, \mathrm{NE} \\
\boldsymbol{u}_i^L \leq \boldsymbol{u}_K^i(\zeta_{i+1}) \leq \boldsymbol{u}_i^U, \quad i = 1, \ldots, \mathrm{NE} \\
\Delta\zeta_i^L \leq \Delta\zeta_i \leq \Delta\zeta_i^U \quad i = 1, \ldots, \mathrm{NE} \\
\boldsymbol{h}(t_{ij}, \boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}) = \boldsymbol{0},\ \boldsymbol{g}(t_{ij}, \boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}) \leq \boldsymbol{0} \\
\boldsymbol{x}_{ij}^L \leq \boldsymbol{x}_{K+1}(t_{ij}) \leq \boldsymbol{x}_{ij}^U,\ i = 1, \ldots, \mathrm{NE},\ j = 0, \ldots, K \\
\boldsymbol{u}_{ij}^L \leq \boldsymbol{u}_K(t_{ij}) \leq \boldsymbol{u}_{ij}^U,\ i = 1, \ldots, \mathrm{NE},\ j = 1, \ldots, K
\end{gathered}$$

where

i – refers to the element,

j – refers to the collocation point,

$\Delta\zeta_i$ – finite-element lengths,

$\boldsymbol{x}_0 = \boldsymbol{x}(0)$ – the value of the state at time $t = 0$,

$\boldsymbol{x}_f = \boldsymbol{x}(t_f)$ – the value of the state at the final time $t = t_f$,

$\boldsymbol{h}$ – the equality constraint evaluated in time t_{ij},

$\boldsymbol{g}$ – the inequality constraint evaluated in time t_{ij},

$\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}$ – the collocation coefficients for the state and control profiles,

Problem (4) can be now solved by any standard nonlinear programming solver.

To solve this problem within MATLAB, we use the Optimization Toolbox which includes several programs for treating optimisation problems. In this case function *fmincon* was choosen. This can minimise/maximise a given objective function subject to nonlinear equality and inequality constraints. In order to use this function it was neccessary to create and program series of additional functions [13]. The resulting code is called *dynopt*. The use of this code is presented in next section.

3 Tutorial

This section presents a tutorial of to *dynopt*.

3.1 Problem Definition

Consider the following problem [5; 6; 8]

$$\min_{\boldsymbol{u}(t)} J = \int_0^{t_f} (x_1^2 + x_2^2 + 0.005u^2)\mathrm{d}t \tag{5}$$

such that

$$\begin{aligned} \dot{x}_1 &= x_2 & x_1(0) &= 0 \\ \dot{x}_2 &= -x_2 + u & x_2(0) &= -1 \\ 0 &\geq x_2 - 8(t-0.5)^2 + 0.5 & \forall t& \\ t_f &= 1 \end{aligned}$$

where

$x_1(t),\ x_2(t)$ – states,

$u(t)$ – control vector.

As the objective function is not in the Meyer form needed by *dynopt*, we define an additional differential equation

$$\dot{x}_3 = x_1^2 + x_2^2 + 0.005u^2, \quad x_3(0) = 0 \tag{6}$$

and rewrite the cost as

$$\min_{\boldsymbol{u}(t)} J = x_3(t_f) \tag{7}$$

3.2 User Functions Definitions

Step1: Process

```
function sys = process(t,x,flag,u)
switch flag,
   case 0
      sys = [x(2);
             -x(2)+u;
             x(1)^2+x(2)^2+0.005*u^2];
   case 1
      sys = [0 0 2*x(1);
             1 -1 2*x(2);
             0 0 0];
   case 2
      sys = [0 1 0.01*u];
   case 3
      sys = [0 0 0];
   case 4
      sys = [0;-1;0];
   otherwise
      error(['unhandled flag = ',num2str(flag)]);
end
```

dynopt optimises a given performance index evaluated at the final conditions subject to the constraints which can be evaluated at the initial conditions or over the full time interval or at the final conditions. Thus the input arguments of *objfun* and *confun* are as follows: `t` - scalar value representing t_{ij}, `x` - state vector evaluated at corresponding time t_{ij}, `u` - control vector evaluated at corresponding time t_{ij}. *objfun* should be defined as follows:

Step2: Objective Function

```
function [f,Dft,Dfx,Dfu] = objfun(t,x,u)

f=x(3);
Dft=[];
Dfx=[0;0;1];
Dfu=[];
```

The given constraints should be written in M-file `confun.m` as follows:

Step3: Constraints

```
function [c,ceq,Dct,Dcx,Dcu,Dceqt,Dceqx,Dcequ] = confun(t,x,u)

c=x(2)-8*(t-0.5)^2+0.5;
ceq=[];
Dct=[-16*t+8];
Dcx=[0;1;0];
Dcu=[];
Dceqt=[];
Dceqx=[];
Dcequ=[];
```

Step4: Optimisation After the problem has been defined by the above mentioned functions, user invokes the *dynopt* function as follows:

```
opt = optimset('LargeScale','off','Display','iter');
opt = optimset(opt,'GradObj', 'on','GradConstr','on');
opt = optimset(opt,'TolFun',1e-5);
opt = optimset(opt,'TolCon',1e-5);
opt = optimset(opt,'TolX',1e-5);

u = [13 4 -0.5 -2 -1.8 0 0.5 2 0 0];
time = 0.1*ones(10,1);
[x,fval,exitflag,output]=dynopt(1,2,4,2,time,1,u,[],[],'objfun','confun', ...
                          'process',opt);
```

Step5: Interpretation of Results The results returned by *dynopt* in `x` contain optimal values of $\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}$, and eventually of $\Delta\zeta_i$ and can be interpreted by example in a graph. This can be done by using additional functions *tuxprint* and *tcceqprint* which prepare plotable profiles for control variables, state variables, and also for constraints. In our case these two functions can be invoked as follows:

```
[time,state,control] = tuxprint(x,4,2,10,1,3,1000);
[tp,cp,ceqp] = tcceqprint(2,x,4,2,10,1,3,'confun',1000);
```

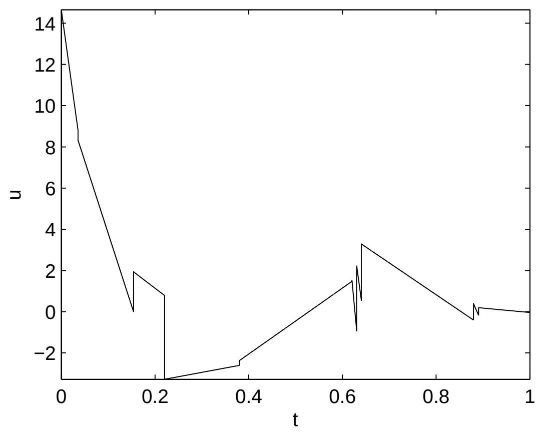

Figure 2: Optimal trajectory found for tutorial problem - control

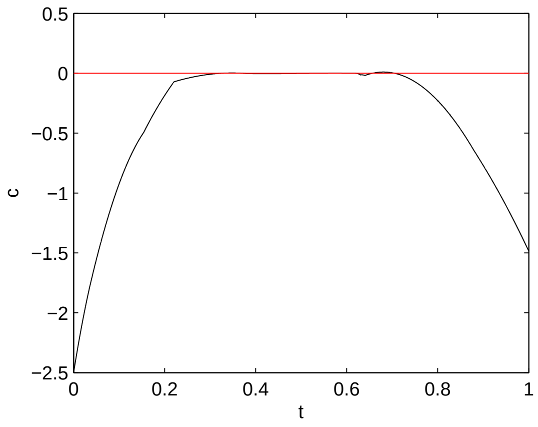

Figure 3: Optimal trajectory found for tutorial problem - constraint

Optimal control and the constraint are shown in Fig. 2 and Fig. 3, respectively. Note, that the constraint is satisfied in collocation points only – this is given by the orthogonal collocation approach.

4 Case Studies

In this section we present the examples from literature solved by *dynopt*.

4.1 Car Optimisation

Consider a following minimum time problem [4; 9; 12]:

$$\min_{\boldsymbol{u}(t)} J = t_f \tag{8}$$

such that

$$\begin{aligned} &\dot{x}_1 = u && x_1(0) = 0 && x_1(t_f) = 0 \\ &\dot{x}_2 = x_1 && x_2(0) = 0 && x_2(t_f) = 300 \\ &u \in [-2, 1] \end{aligned}$$

where

$x_1(t)$ – velocity,

$x_2(t)$ – distance,

$u(t)$ – control variable (acceleration).

4.2 Tubular Reactor

Consider a tubular reactor with parallel reactions $A \to B$, $A \to C$ [4; 9; 12]:

$$\max_{\boldsymbol{u}(t)} J = x_2(t_f) \tag{9}$$

such that

$$\begin{aligned} \dot{x}_1 &= -(u + 0.5u^2)x_1 & x_1(0) &= 1 \\ \dot{x}_2 &= ux_1 & x_2(0) &= 0 \\ u &\in [0, 5] & t_f &= 1 \end{aligned}$$

where

$x_1(t)$ – dimensionless concentration of A,

$x_2(t)$ – dimensionless concentration of B,

$u(t)$ – control variable.

4.3 Batch Reactor

Consider a batch reactor [4; 12] the consecutive reactions $A \rightarrow B \rightarrow C$:

$$\max_{\boldsymbol{u}(t)} J = x_2(t_f) \tag{10}$$

such that

$$\begin{aligned} \dot{x}_1 &= -k_1 x_1^2 & x_1(0) &= 1 \\ \dot{x}_2 &= k_1 x_1^2 - k_2 x_2 & x_2(0) &= 0 \\ k_1 &= 4000e^{(-\frac{2500}{T})} & k_2 &= 620000e^{(-\frac{5000}{T})} \\ T &\in [298, 398] & t_f &= 1 \end{aligned}$$

where

$x_1(t)$ – concentration of A,

$x_2(t)$ – concentration of B,

T – temperature (control variable).

4.4 Plug Flow Reactor

Consider a catalytic plug flow reactor [4; 12] with the following reactions:
$A \leftrightarrow B \rightarrow C$

$$\max_{\boldsymbol{u}(t)} J = 1 - x_1(t_f) - x_2(t_f) \tag{11}$$

such that

$$\begin{aligned} \dot{x}_1 &= u(10x_2 - x_1) & x_1(0) &= 1 \\ \dot{x}_2 &= -u(10x_2 - x_1) - (1 - u)x_2 & x_2(0) &= 0 \\ u &\in [0, 1] & t_f &= 12 \end{aligned}$$

where

$x_1(t)$ – mole fraction of A,

$x_2(t)$ – mole fraction of B,

$u(t)$ – fraction of type 1 catalyst.

4.5 CSTR

Consider the following problem of continuously stirred tank reactor (CSTR) [1; 6; 11]

$$\begin{aligned}\max_{\boldsymbol{u}(t)} J = \int_0^{0.2} \big(5.8(qx_1 - u_4) - 3.7u_1 - 4.1u_2 \\ + q(23x_4 + 11x_5 + 28x_6 + 35x_7) - 5.0u_3^2 \\ - 0.099\big)\,\mathrm{d}t \end{aligned} \tag{12}$$

such that

$$\begin{aligned}
\dot{x}_1 &= u_4 - qx_1 - 17.6x_1x_2 - 23x_1x_6u_3 \\
\dot{x}_2 &= u_1 - qx_2 - 17.6x_1x_2 - 146x_2x_3 \\
\dot{x}_3 &= u_2 - qx_3 - 73x_2x_3 \\
\dot{x}_4 &= -qx_4 + 35.2x_1x_2 - 51.3x_4x_5 \\
\dot{x}_5 &= -qx_5 + 219x_2x_3 - 51.3x_4x_5 \\
\dot{x}_6 &= -qx_6 + 102.6x_4x_5 - 23x_1x_6u_3 \\
\dot{x}_7 &= -qx_7 + 46x_1x_6u_3 \\
\boldsymbol{x}(0) &= [0.1883\ 0.2507\ 0.0467\ 0.0899\ 0.1804\ 0.1394\ 0.1046]^{\mathrm{T}} \\
q &= u_1 + u_2 + u_4 \\
0 &\le u_1 \le 20 \\
0 &\le u_2 \le 6 \\
0 &\le u_3 \le 4 \\
0 &\le u_4 \le 20 \\
t_f &= 0.2
\end{aligned}$$

where

$x_1(t) - x_7(t)$ – states,

$u_1(t) - u_4(t)$ – controls.

Again, the cost function can be rewritten to the Meyer form by introducing a new state defined by the integral function with its initial value equal to zero.

5 Results and Discussion

For all the examples, a piecewise linear profile for the control variable was used. We have used 2 time elements (example 1), 4 time elements (examples 2–4), and 10 elements (examples 5, 6) and precision 10^{-5} (if not stated otherwise).

The first problem (Section 4.1) consists of starting and stoping a car in minimum for a fixed distance (300 units). This problem was treated by [2; 9; 10] and the optimal value of 30 time units was reported. By using 4 collocation points for state variables and 2 collocation points for control variable we obtained the same value of performance index as from the literature. Optimal control and state profiles are shown in Fig. 4, Fig. 5, and Fig. 6.

The second problem (Section 4.2) is a tubular reactor control problem where the state variable x_2 at final time has to be maximised. This problem was treated by [4; 9; 12] and the optimal value (0.57353) was reported by [4; 9] and optimal value (0.57284) was given by [12].

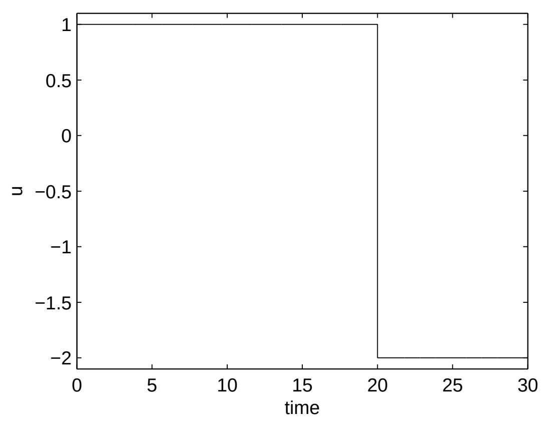

Figure 4: Optimal trajectory found for Example 4.1 - control (acceleration)

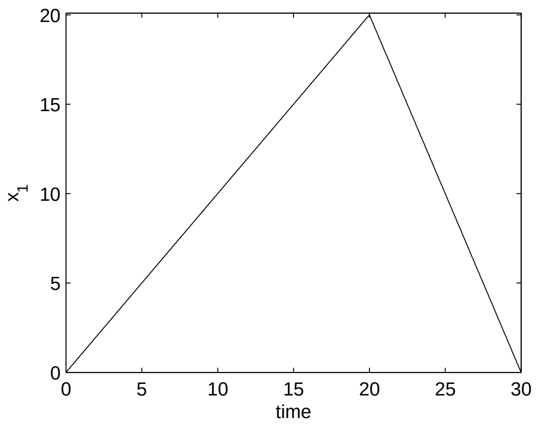

Figure 5: Optimal trajectory found for Example 4.1 - velocity

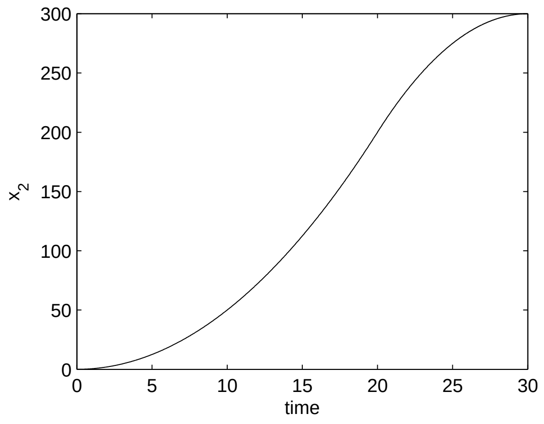

Figure 6: Optimal trajectory found for Example 4.1 - distance

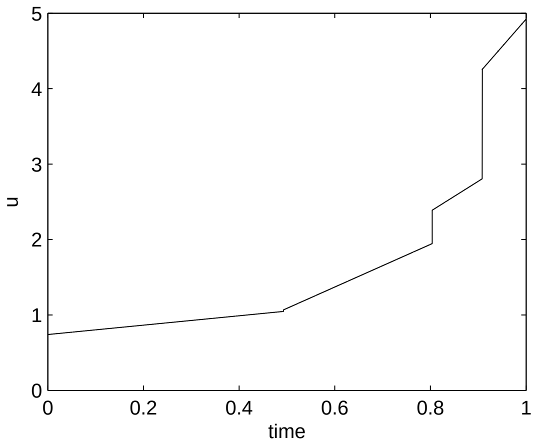

Figure 7: Optimal trajectory found for Example 4.1 - control

By using 4 collocation points for state variables and 2 collocation points for control variable we obtained the value of performance index 0.5729. Optimal control trajectory is presented by Fig. 7.

The objective in problem described in Section 4.3 is to obtain the optimal temperature profile that maximizes x_2 at the end of a specified time. The problem was solved by [9; 12] and the reported optimum (0.610775) was found by [9] and (0.61045) obtained by [12]. We were able to obtain the value of 0.6107 by applying 3 collocation ponits for state variables and 2 collocation points for control variable. Optimal temperature profile is presented by Fig. 8.

Optimisation of problem in Section 4.4 has also been analysed. This problem was solved by [9; 12] and the optima (0.476946, 0.47615) were found. Value of the performance index obtained for this example using *dynopt* is 0.4770. In this case 4 collocation points for state variables and 2 collocation points for control variables were used. Optimal control trajectory is presented by Fig. 9.

Maximisation problem in Section 4.5 was treated by [1; 6; 11]. Four control variables of a chemical reactor are optimised in order to obtain maximum economic benefit. Reported optimal value (21.757) was obtained using CVP method inplemented in *DYNO* [6] and also in other references. For this example, 4 collocation points for state variables and 2 collocation points for control variables were choosen and an optimum was found at value 21.8003. Note however, that better value can be explained by the use of linear control profiles compared to piece-wise constant approximations in references.

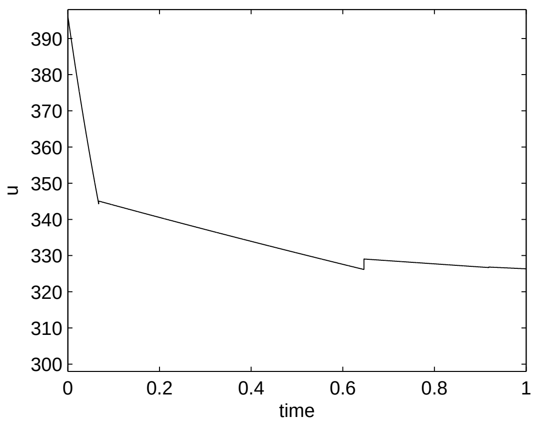

Figure 8: Optimal trajectory found for Example 4.2 - control

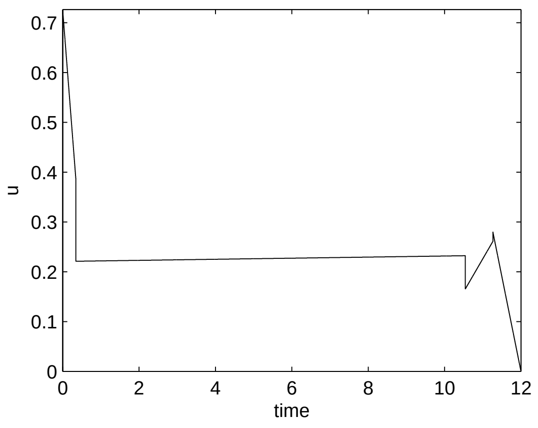

Figure 9: Optimal trajectory found for Example 4.3 - control

Note, that all the results obtained by orthogonal collocation on finite elements method implemented within MATLAB-*dynopt* are only local in nature, since NLP solvers are only based on necessary conditions for optimality.

6 Conclusion

The orthogonal collocation on finite elements has been developed and implemented within MATLAB environment. It has been tested on a few examples from the literature dealing with chemical reactors. The examples were chosen to illustrate the ability of the *dynopt* package to treat the problems of varying levels of difficulty.

The package *dynopt* is able to obtain results in a good agreement with the referenced works. From the user point of view, it is implemented in MATLAB and thus aimed for a large base of its users. Its main purpose is to make possible a rapid development and testing of dynamic optimisation problems and incorporation to higher level problems.

It must be noted that optima obtained are only local in nature. Our future work will be devoted to global optimisation problems where the package can provide local results for the global problems.

Acknowledgments

The authors are pleased to acknowledge the financial support of the Scientific Grant Agency of the Slovak Republic under grant No. 1/0135/03. Part of the work has been completed under the French – Slovak program Stefanik (Dynamic and Global Optimisation of Processes) and within the framework of the European Social Fund (Development of Human Resources for Research and Progress in Automation, JPD 3 2004/1-056).

References

[1] M-S. G. E. Balsa-Canto, J. R. Banga, A. A. Alonso, and V. S. Vassiliadis. Dynamic optimization of chemical and biochemical processes using restricted second-order information. *Computers chem. Engng.*, 25(4-6):539–546, 2001. 8, 9

[2] J. E. Cuthrell and L. T. Biegler. On the optimization of differential-algebraic process systems. *AIChE Journal*, 33:1257–1270, 1987. 1, 8

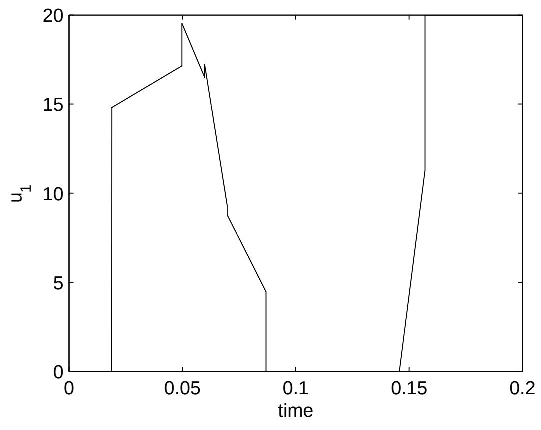

Figure 10: Optimal trajectory found for Example 4.5 - control 1

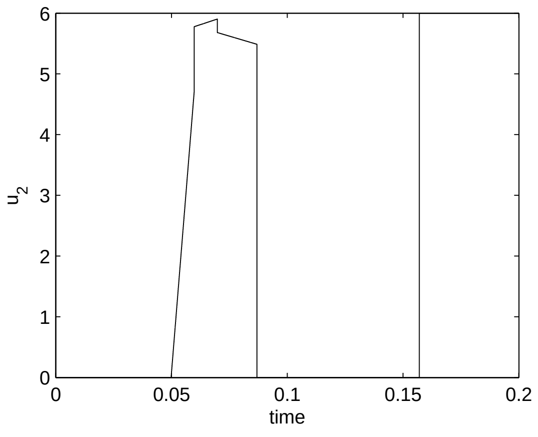

Figure 11: Optimal trajectory found for Example 4.5 - control 2

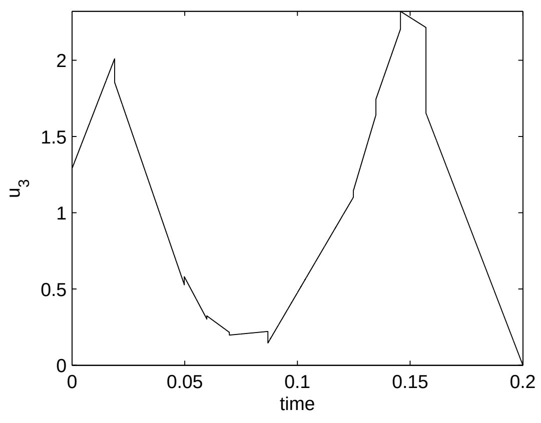

Figure 12: Optimal trajectory found for Example 4.5 - control 3

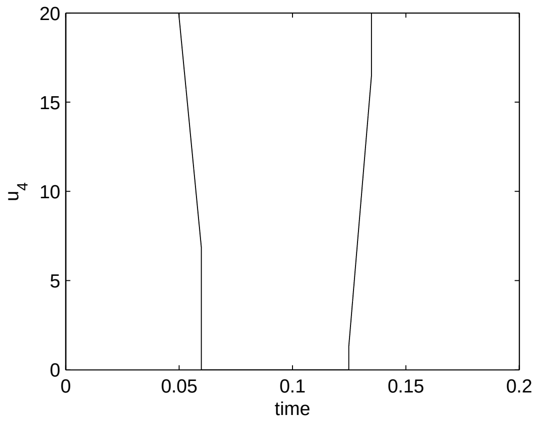

Figure 13: Optimal trajectory found for Example 4.5 - control 4

[3] J. E. Cuthrell and L. T. Biegler. Simultaneous optimization and solution methods for batch reactor control profiles. *Computers chem. Engng.*, 13(1/2):49–62, 1989. 2

[4] S.A. Dadebo and K.B. McAuley. Dynamic optimization of constrained chemical engineering problems using dinamic programming. *Computers chem. Engng.*, 19:513–525, 1995. 6, 7, 8

[5] W. F. Feehery. *Dynamic Optimisation with Path Constraints.* PhD thesis, MIT, 1998. 4

[6] M. Fikar and M. A. Latifi. User's guide for FORTRAN dynamic optimisation code DYNO. Technical Report mf0201, LSGC CNRS, Nancy, France; STU Bratislava, Slovak Republic, 2002. 4, 8, 9

[7] C. J. Goh and K. L. Teo. Control parametrization: A unified approach to optimal control problems with general constraints. *Automatica*, 24:3–18, 01 1988. 1

[8] D. Jacobson and M. Lele. A transformation technique for optimal control problems with a state variable inequality constraint. *IEEE Trans. Automatic Control*, 5:457–464, 1969. 4

[9] J. S. Logsdon and L. T. Biegler. Accurate solution of differential-algebraic optimization problems. *Chem. Eng. Sci.*, (28):1628–1639, 1989. 1, 2, 6, 8, 9

[10] J. S. Logsdon and L. T. Biegler. Decomposition strategies for large-scale dynamic optimization problems. *Chem. Eng. Sci.*, 47(4):851–864, 1992. 2, 8

[11] R. Luus. Application of dynamic programming to high-dimensional non-linear optimal control problems. *Int. J. Control*, 52(1):239–250, 1990. 8, 9

[12] J. Rajesh, K. Gupta, H.S. Kusumakar, V.K. Jayaraman, and B.D. Kulkarni. Dynamic optimization of chemical processes using ant colony framework. *Computers and Chemistry*, 25:583–595, 2001. 6, 7, 8, 9

[13] M. Čižniar, M. Fikar, and M.A. Latifi. *User's Guide for MATLAB Dynamic Optimisation Code DYNOPT*, 2005. 3

M. Čižniar, M. Fikar
Department of Information Engineering and Process Control,
Faculty of Chemical and Food Technology, STU
Radlinského 9, 83102 Bratislava
fax : +421 2 52496469 and e-mail : miroslav.fikar@stuba.sk

D. Salhi, M. A. Latifi
Laboratoire des Sciences du Génie Chimique
CNRS-ENSIC, B.P. 451, 1 rue Grandville, 54001 Nancy Cedex
fax : +33 (0)3 83 17 53 26 and e-mail : latifi@ensic.inpl-nancy.fr

Európsky sociálny fond