Modified Multiple Shooting Combined with Collocation Method in JModelica.org with Symbolic Calculations

Evgeny Lazutkin, Abebe Geletu, Siegbert Hopfgarten, Pu Li
Simulation and Optimal Processes Group, Institute for Automation and Systems Engineering,
Technische Universität Ilmenau, P.O.Box 10 05 65, 98684 Ilmenau, Germany.
(evgeny.lazutkin, abebe.geletu, siegbert.hopfgarten, pu.li)@tu-ilmenau.de

Abstract

This paper presents an efficient and a novel implementation of a combined multiple shooting and collocation (CMSC) algorithm for the solution of nonlinear optimal control problems. The implemented algorithm is a modification of the approach proposed in [17, 18]. The new implementation is done under the JModelica.org framework along with CasADi and Ipopt. The framework uses a symbolic pre-calculation of functions and derivatives. Besides the integration of various components of JModelica.org, Ipopt, and CasADi, the implementation facilitates simpler modeling of optimal control problems along with a choice of options for various linear algebra algorithms. The paper gives a description of the algorithm and elaborates the components of the framework. Numerical experimentations show that the new implementation is efficient in comparison with the published results of other authors.

Keywords: Nonlinear Optimal Control; Symbolic Automatic Differentiation; Nonlinear Programming; Multiple Shooting; Collocation

1 Introduction

Optimization methods nowadays play pivotal roles in engineering and industrial applications. Most engineering applications are dynamic by nature. Frequently, such dynamic processes have model equations involving large-scale nonlinear differential equations. Hence, the solution of large-scale optimal control problems is difficult to achieve by solving the equations of the optimality conditions. Therefore, the modern approach follows the "first discretize, then optimize" strategy. In this way, the optimal control problem will be transformed into a nonlinear programming problem (NLP). This facilitates the implementation of efficient and state-of-the-art NLP solvers to determine highly accurate approximate optimal solutions to the continuous optimal control problem.

The direct discretization of optimal control problems through the multiple shooting method was first proposed in [9]. On the other hand, the direct discretization of optimal control problems through collocation methods has been widely used for state constrained optimal control problems (e.g., [7, 10, 13]). The multiple shooting discretization results in block-structured matrices and facilitates easy parallelization of computations. However, the accuracy of the multiple shooting method can be highly augmented if it is combined with the collocation method. Therefore, recently, discretization of optimal control problems through a CMSC method is found to have enormous potentials for the solution of complex large-scale optimal control problems [17, 18].

In order to facilitate the industrial application of complex optimization algorithms, model-based optimization of dynamic systems is recently gaining greater momentum [1, 12]. The aim of this work is to implement the CMSC algorithm in the JModelica.org framework. We first divide the time horizon $[t_0, t_f]$ into subintervals (finite elements). Subsequently, we use multiple shooting and collocation methods to discretize the optimal control problem and to transform it into an NLP. In our implementation, we use precalculated derivatives, i.e., Jacobian matrix in symbolic form by means of CasADi [3, 4]. The nonlinear optimization problem is solved by using Ipopt (**I**nterior **p**oint **opt**imization solver, [19]). The total optimization tool-chain is provided in the JModelica.org framework [2].

The rest of the paper is organized as follows. Section 2 provides a general form of the considered optimal control problem. Section 3 presents the combined multiple shooting and collocation algorithm. Section 4 describes the implementation of the algorithm under the JModelica.org framework coupled with CasADi and Ipopt and section 5 presents case studies. The

comparative analysis in section 6 shows the high efficiency and viability of our implementation as compared to other implementations. The paper concludes in section 7 with a summary and future research work.

2 Problem definition

We consider the following general form of a nonlinear optimal control problem (NOCP):

$$\min_{u(t)} \left\{ J = E\left(x(t_f)\right) + \int_{t_0}^{t_f} L(x(t),u(t),t)dt \right\} \quad (1)$$

$$\text{subject to: } \dot{x}(t) = f(x(t),u(t),t),\ x(t_0) = x_0, \quad (2)$$

$$g(x(t),u(t),t) = 0, \quad (3)$$

$$x_{min} \leq x(t) \leq x_{max}, \quad (4)$$

$$u_{min} \leq u(t) \leq u_{max}, \quad (5)$$

$$t_0 \leq t \leq t_f, \quad (6)$$

where $x(t)^\top = (x_1(t),\ldots,x_n(t))$ and $u(t)^\top = (u_1(t),\ldots,u_m(t))$ are the state and control vectors, respectively. The functions f, h, and g are conveniently defined in appropriate function spaces. The dynamics of the process is described by (2) and we have algebraic equality constraints (3). In practical engineering or industrial application, state and control variables are usually bounded. Hence, (4) imposes lower and upper bounds on the states, while (5) defines bounds on the controls. The optimization variables (typically the controls) and state variables must satisfy the model equations, the state and control constraints and the boundary conditions. Furthermore, the optimization task is limited to a time horizon with an initial time t_0 and a fixed final time t_f. In this paper, time-optimal control problems will not be considered.

The optimal control problem (1) - (6) is an infinite-dimensional problem. Since real-world applications have very complicated structure including nonlinearity and high dimensionality, indirect methods, like methods based on the Pontryagin's principle, are not suitable. Therefore, here the NOCP will be directly discretized by using a combined multiple shooting and collocation (CMSC) method and thereby it will be transformed to a finite-dimensional optimization problem.

3 An improved multiple shooting with collocation framework

For the multiple shooting and collocation discretization scheme, first the time interval $[t_0,t_f]$ is divided into appropriate shooting intervals $[t_i,t_{i+1}], i = 0,\ldots,N-1$. Then, on each shooting interval $[t_i,t_{i+1}]$, collocation nodes $t_i = t_{i0} < t_{i1} < \ldots, t_{iN_c} = t_{i+1}$ are defined, so that each state variable $x_k(t)$ is approximated by the polynomial

$$\hat{x}_k(t) = \sum_{j=0}^{N_c} x_{ij}^{(k)} l_{ij}(t), \quad (7)$$

where $l_{ij}(t)$ are the Lagrange polynomials

$$l_{ij}(t) = \prod_{\substack{s=0 \\ s\neq j}}^{N_c} \left[\frac{t-t_{is}}{t_{ij}-t_{is}}\right], j = 1,\ldots,N_c, i = 1,\ldots,N, \quad (8)$$

and the variables $x_{ij}^{(k)}, j = 1,\ldots,N_c$, represent the state values corresponding to the state variable $x_k(t), k = 1,\ldots,n$, at the collocation points on $[t_i,t_{i+1}]$ with the property that $x_k(t_{ij}) = \hat{x}_k(t_{ij}) = x_{ij}^{(k)}$. Hence, the combined multiple shooting and collocation scheme transforms the NOCP into an NLP with the additional constraints

$$h_{i+1}^k = \hat{x}_k(t_{(i+1)0}), i = 0,\ldots,N-1; k = 1,\ldots,n \quad (9)$$

to be imposed along with the discretized constraints (2)-(5). The equations (9) guarantees the continuity of the state trajectories at the end point of each shooting interval. In the following $h_{i+1}^x = \left(h_{i+1}^1,\ldots,h_{i+1}^n\right)^\top$ and $\hat{x}_{i+1}(h_i^x,v_i,t_{i+1}) = \left(\hat{x}_1(t_{(i+1)0}),\ldots,\hat{x}_n(t_{(i+1)0})\right)^\top$ are used for the sake of brevity. The expression $\hat{x}_{i+1}(h_i^x,v_i,t_{i+1})$ indicates that the state variables are dependent on the initial state h_i^x, the controls v_i and the end time point $t_{(i+1)0} = t_{i+1}$ on the interval $[t_i,t_{i+1}]$. The resulting nonlinear optimization problem can be written as

$$\min_{h_0^x,\ldots,h_N^x,v_0,\ldots,v_{N-1}} \left\{ E\left(h_N^x\right) + \sum_{i=0}^{N-1} L(h_i^x,x_i,v_i) \right\} \quad (10)$$

$$\text{subject to: } h_{i+1}^x - \hat{x}_{i+1}(h_i^x,v_i,t_{i+1}) = 0, \quad i = 0,\ldots,N-1, \quad (11)$$

$$G(h_i^x,x_i,v_i) = 0,\ i = 0,\ldots,N-1, \quad (12)$$

$$h_0^x - x_0 = 0, \quad (13)$$

$$x_{min} \leq h_i^x \leq x_{max}, \quad (14)$$

$$u_{min} \leq v_i \leq u_{max}, \quad (15)$$

In problem (10) - (15) on the i-th shooting subinterval, h_i^x represents parameterized initial conditions for the vector of state variables, $\hat{x}_i$ is the state on the collocation point at the end of interval i, the vector x_i consists of all coefficients of the collocation polynomials corresponding to the states on the i-th interval and v_i are

DOI 10.3384/ECP14096999

parameterized control variables. The discretization of the states uses the Radau collocation method. Using the function G, equation (12) represents the discretized form of the equations (2, 3). The control variables are usually parameterized as piecewise constant functions. E.g., in each subinterval $[t_i, t_{i+1}], i = 0, \ldots, N-1$, the control variables are assumed to take constant values and the state trajectories will be approximated by the collocation method. The unique feature of CMSC lies in the fact that it utilizes the advantages of both multiple shooting and collocation methods. Detailed discussions on multiple shooting and collocation methods are found in [8, 9, 16].

In the objective (10) the variable x does not appear exclusively as an optimization variable. Hence, after appropriate aggregation of variables, at each step of the optimization algorithm, we can solve for x in terms of (h, v), so that we have $x(h, v)$. In this way the equality constraints (12) can be eliminated by reducing the number of optimization variables. However, in this work, constraints on the coefficients of the collocation polynomials are not considered inside the collocation intervals, but only at the end-points of the intervals.

Therefore, the problem (10) - (15) can be equivalently written as

$$\min_{h,v} F(h, x(h, v), v) \tag{16}$$

$$\text{subject to :} \quad H(h, v) = 0 \tag{17}$$

$$x_{min} \leq h \leq x_{max} \tag{18}$$

$$u_{min} \leq v \leq u_{max}, \tag{19}$$

where H in equation (17) provides a compact representation of the equations (11) - (13). Consequently, to solve the nonlinear optimization problem (16) - (19) we use the state-of-the-art optimization solver Ipopt [19]. At each iteration of Ipopt, the nonlinear model equations are solved by using a local Newton algorithm along with a linear algebra solver for the determination of the Newton-steps to determine state variables for given values of h and v. Future investigations will consider the implementation of a globalized Newton method with appropriate linear algebra solvers.

All function values and gradients, required in the optimization framework, are pre-calculated and made available in symbolic form. Symbolic derivatives are calculated by using CasADi and stored in matrices or vectors to facilitate faster accessability. For this, it is needed to compute the sensitivities $\frac{\partial F}{\partial v}, \frac{\partial F}{\partial h}, \frac{\partial H}{\partial v}$, and $\frac{\partial H}{\partial h}$ in symbolic representations. Further details are found in [17, 18].

The computational framework is summarized graphically in Fig. 1 and Algorithm 1 provides a description in general terms. More detailed discussions on the CMSC method are found in [17, 18].

Algorithm 1 : A general CMSC

1: **Input**: Time horizon, number of subintervals, number of state and control variables, lower and upper bounds for the control and state variables, model equations, objective function, initial guesses, optimizer options.
2: Initialization of the NOCP
3: Define continuity and path constraints
4: Initialize the n-point collocation for each subinterval
5: Compute states at collocation points and sensitivities
6: Compute objective function and gradient
7: Compute constraint function values and Jacobian
8: Solve the equations of the Karush-Kuhn-Tucker (KKT) optimality conditions
9: **if** KKT condition not satisfied **then**
10: GOTO 3
11: **else**
12: STOP.
13: **end if**
14: **Output**: optimal state and control trajectories, optimal objective function value, number of iterations and CPU time.

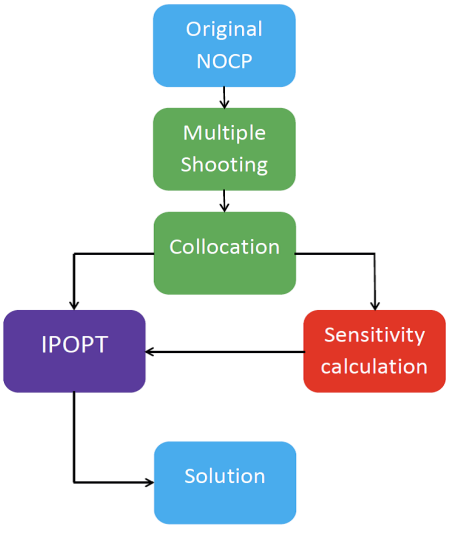

Figure 1: : Combined multiple shooting and collocation (CMSC) framework

Algorithm 1 is general for CMSC from the work of [17, 18]. This work implements a modified and improved version of Algorithm 1. Hence, Algorithm 2 provides a modified CMSC.

Algorithm 2 : A modified CMSC

1: **Input**: Time horizon, number of subintervals, number of state and control variables, lower and upper bounds for the control and state variables, model equations, objective function, initial guesses, optimizer options, number of collocation points in subintervals, user-defined options, e.g., choice of a linear algebra algorithms, etc.
2: Calculate in symbolic form the Jacobian of the system and all directional derivatives.
3: Corresponding to the number of subintervals, states/controls, and collocation points, construct derivative matrices
4: Initialize and construct symbolic representations of the objective function and corresponding gradient.
5: Initialize and construct symbolic representation of the constraint functions and corresponding Jacobian.
6: Give an initial guess for the optimization variables
7: Set options for Ipopt to provide Hessian approximation, tolerance of the solution, and other user specified options.
8: Call the Ipopt solver
9: Call plotting routine and save the results.

The improvements provided in Algorithm 2 are:

- the symbolic pre-calculation to obtain representations for the values of the objective function, constraints, and corresponding sensitivities (Jacobians and derivatives),
- facilitate the use of several linear algebra algorithms to give the user a choice of options and improve the efficiency of computations,
- implementation on JModelica.org platform for a wider public accessibility.

Section 4 discusses the software implementation of Algorithm 2.

4 Framework and software components

In the framework shown in Fig. 2, the NOCP is modeled under JModelica.org using a Python script. Then the problem is discretized using our CMSC with the help of CasADi to obtain an NLP. Subsequently, CasADi is again invoked to generate symbolic expressions for the derivatives. Finally, JModelica.org invokes Ipopt to solve the NLP by using the precalculations. All problem definitions and our custom implementations are done using the Python programming language.

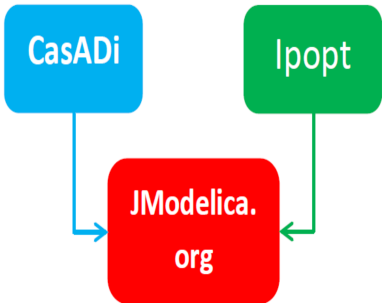

Figure 2: : Software framework

In the next subsections, we give a brief review on the software components.

4.1 JModelica.org

JModelica.org [2] is a Modelica-based open source software platform for modeling and solving optimal control problems and implementation of user-developed algorithms. JModelica.org was first developed at the Department of Automatic Control, Lund University. Currently, it enjoys active support from the industry (Modelon AB) as well as academic and research institutions. Since JModelica.org is extensible through user-designed algorithms, we have decided to implement our algorithm under this framework.
Among the salient features of JModelica.org are:

- support for mixed-language programming mode,
- easy and smoother integrability of custom numerical libraries,
- support for object-oriented modeling based on Modelica,
- wider public access owing to simpler user interfaces.

4.2 CasADi

CasADi is an open source software library for symbolic automatic differentiation of functions [5]. CasADi uses computer algebra techniques to implement the forward and adjoint automatic differentiation techniques and facilitate the pre-computation of gradients and Jacobian of objective functions and constraints, respectively.

DOI 10.3384/ECP14096999

One of the major features of CasADi is its high integrability to widely available open source numerical libraries and optimization solvers. CasADi is written using C++ programming language. However, it can be interfaced to numerical solvers based on various programming language implementations, e.g., C, C++, Python, FORTRAN, etc. The recent integration of CasADi to the JModelica.org platform [4] is one proof for its high integrability and applicability for the solution of complex optimal control problems. Therefore, in our implementation, we have exploited the potential of CasADi for symbolic pre-computation of derivatives in order to improve the efficiency of computations.

4.3 Ipopt

Ipopt is a software implementation of an interior point algorithm coupled with a filter line-search algorithm [19]. Ipopt is a state-of-the-art solver for large-scale NLP problems. Consequently, it facilitates the efficient solution of large-scale optimal control problems using the "first discretize, then optimize" approach.

In general, Ipopt can be used to solve NLP problems of the form

$$\min_{x \in R^n} f(x) \tag{20}$$

subject to :

$$g_{min} \leq g(x) \leq g_{max}, \tag{21}$$

$$x_{min} \leq x \leq x_{max}, \tag{22}$$

where $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function, and $g(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is the constraint function. The vectors x_{min} and x_{max} are the bounds on the variables x, and the vectors g_{min} and g_{max} denote the lower and upper bounds on the constrained function $g(x)$. Furthermore, equality constraints can also be stated in the above formulation by setting the corresponding components of g_{min} and g_{max} to the same value. The functions $f(x)$ and $g(x)$ can be nonlinear and non-convex. Theoretically, $f(x)$ and $g(x)$ are required to be twice continuously differentiable. However, Ipopt is capable of working with first order information, so that Hessian matrices can be approximated numerically.

5 Case studies

To investigate the performance of the modified method, the following two case studies are considered.

5.1 A nonlinear batch-reactor

The system has two state variables $x_1(t)$ and $x_2(t)$ (corresponding to concentrations of two species) and one control variable $u(t)$ (reaction temperature). The objective of this benchmark problem is to achieve a maximum product output of $x_2(t_f)$. The NOCP is formulated as follows:

$$\max_{u(t)} x_2(t_f) \tag{23}$$

subject to :

$$\dot{x}_1(t) = -\left(u(t) + \frac{u^2(t)}{2}\right) \cdot x_1(t), \tag{24}$$

$$\dot{x}_2(t) = u(t) \cdot x_1(t), \tag{25}$$

$$x_1(t_0) = 1, \tag{26}$$

$$x_2(t_0) = 0, \tag{27}$$

$$0 \leq \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \leq 1, \tag{28}$$

$$0 \leq u(t) \leq 5, \tag{29}$$

$$0 \leq t \leq 1. \tag{30}$$

The objective function in equation (23) describes the amount of output of the second species at the final time. This example has been also studied in [6, 14, 17, 18].

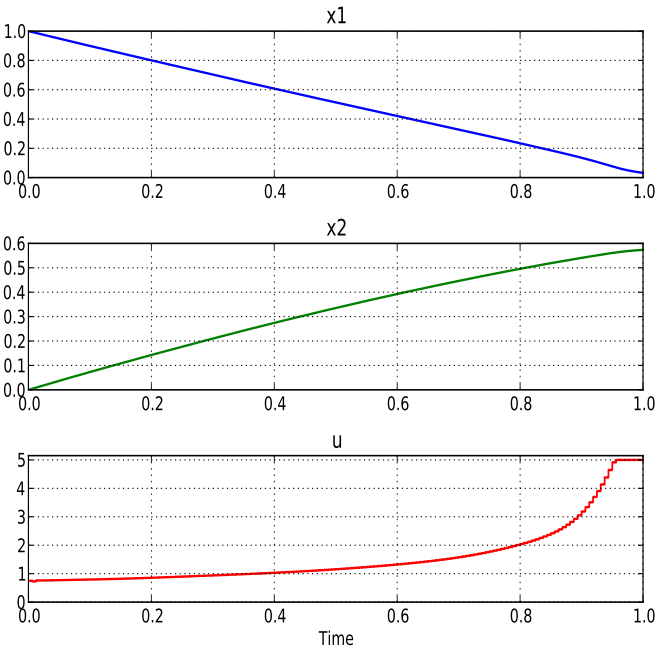

Figure 3: : Optimal solution of the NOCP (23) - (30)

The results shown in the Fig. 3 is obtained by using Algorithm 2 with 160 subintervals. In the next section we will present a comparative analysis of results

obtained from our Algorithm 2 and similar solution methods of other authors.

5.2 A satellite control problem

A nonlinear optimal control problem of a rigid satellite initially undergoing a tumbling motion is considered. The problem data are given as follows [17]:

- $I_1 = 10^6, I_2 = 833333, I_3 = 916677$ represent principal moments of inertia, respectively
- $T_{1s} = 550, T_{2s} = 50, T_{3s} = 550$ are time constants
- Initial states: $x_1(0) = 0,\ x_2(0) = 0,\ x_3(0) = 0,$ $x_4(0) = 1, x_5(0) = 0.01, x_6(0) = 0.005,$ $x_7(0) = 0.001$
- Time horizon: $[t_0, t_f] = [0, 100]$
- Fixed terminal state: $x_{t_f}^\top = (0.70106, 0.0923, 0.56098, 0.43047, 0, 0, 0)$

The aim of the optimal control is to determine the torques that bring the satellite to rest in the specified time $t_f = 100$, while minimizing the performance index. The NOCP is defined as follows [17]:

$$\min_{u(t)} \left\{ \|x(t_f) - x_f\|^2 + \frac{1}{2}\int_{t_0}^{t_f} \|u\|^2 dt \right\} \tag{31}$$

$$\text{subject to: } \dot{x}_1 = \frac{1}{2}(x_5x_4 - x_6x_3 + x_7x_2) \tag{32}$$

$$\dot{x}_2 = \frac{1}{2}(x_5x_3 + x_6x_4 - x_7x_1) \tag{33}$$

$$\dot{x}_3 = \frac{1}{2}(-x_5x_2 + x_6x_1 - x_7x_4) \tag{34}$$

$$\dot{x}_4 = -\frac{1}{2}(x_5x_1 + x_6x_2 + x_7x_3) \tag{35}$$

$$\dot{x}_5 = \frac{(I_2 - I_3)x_6x_7 + T_{1s}u_1}{I_1} \tag{36}$$

$$\dot{x}_6 = \frac{(I_3 - I_1)x_7x_5 + T_{2s}u_2}{I_2} \tag{37}$$

$$\dot{x}_7 = \frac{(I_1 - I_2)x_5x_6 + T_{3s}u_3}{I_3}. \tag{38}$$

Here $x^\top(t) = (x_1(t), \ldots, x_7(t))$ is the state vector and $u^\top(t) = (u_1(t), u_2(t), u_3(t))$ is the control vector of the torques acting for the respective body-principle axes. The model equations (32) - (35) are the kinematic equations associated to the orientation and (36) - (38) are the dynamic equations associated to the motion of the satellite. The state variables $x_1(t)$ - $x_4(t)$ are the Euler parameters and $x_5(t)$ - $x_7(t)$ are the angular rates.

To solve this problem using Algorithm 2, we divide the time horizon into 50 subintervals and use a 3-point collocation on each subinterval. This leads to an NLP with 507 variables and 357 constraints. The results of our implementation are depicted in Figs. 4 - 6.

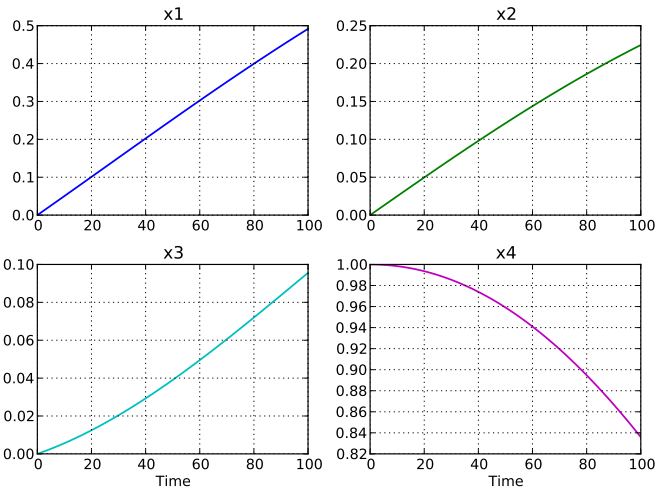

Figure 4: : Optimal states x_1 - x_4

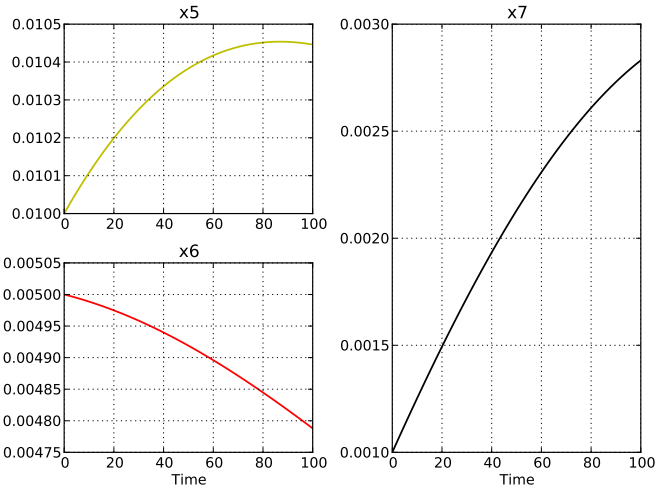

Figure 5: : Optimal states x_5 - x_7

It takes 291 milliseconds of CPU time for the computation and the obtained optimal value of the objective function is 0.463968. In contrast, the optimal value of the objective function in [17] is 0.468287 obtained in 531 milliseconds of CPU time.

6 Comparative analysis

In this section we use the problem (23) - (30) for the purpose of comparison. Tables 1 - 4 present results obtained from similar, but different optimization methods. Note, that these results have been obtained using different computational platforms. Nevertheless, some ideas can be gained by observing the results obtained.

DOI 10.3384/ECP14096999

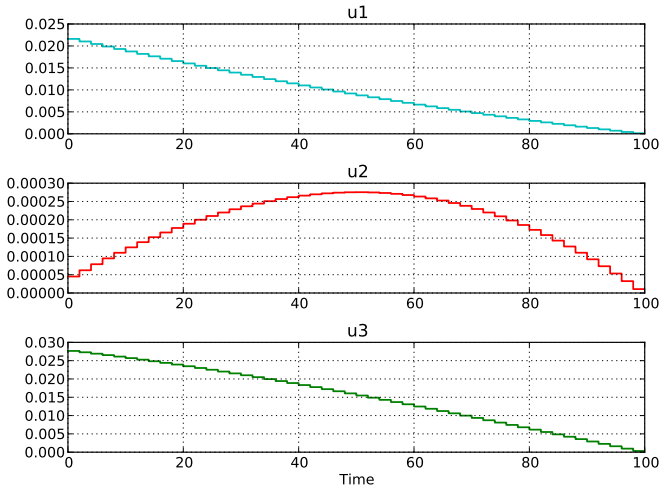

Figure 6: : Optimal controls $u_1(t), u_2(t), u_3(t)$

However, our modified CMSC algorithm is comparable to the collocation algorithm suggested in [15], since the implementation in [15] and of our modified CMSC are both done on the same type of computer with 3.2 GHz CPU frequency.

Table 1: CPU time (in milliseconds)

Number of subintervals	Modified CMSC	CMSC [17]	Collocation [15]	Multiple Shooting [14]	
				Unc., Sp.	Con., Co.
5	4	188	4	4	3
10	6	290	8	6	5
20	8	350	12	9	9
40	12	480	20	12	17
80	24	547	48	24	57
160	40	735	102	58	341
320	81	NaN	268	159	2717

In Tables 1 and 2
- Unc.: uncondensed SQP,
- Con.: condensed SQP,
- Sp.: Block structured QP solver using MA57,
- Co.: Matrix condensing and dense QP solver qpOASES [11].

Table 2: Number of iterations

Number of subintervals	Modified CMSC	CMSC [17]	Collocation [15]	Multiple Shooting [14]	
				Unc., Sp.	Con., Co.
5	14	16	17	7	7
10	16	21	24	9	9
20	19	21	21	9	9
40	22	25	20	10	10
80	27	23	23	10	11
160	18	23	23	12	12
320	14	27	27	12	14

As shown in Tables 1 and 2, the modified CMSC method performs more efficient, both in terms of CPU time and number of iterations, as compared to pure simultaneous (collocation) method. This facilitates the future use of Algorithm 2 for the model predictive control scheme on longer prediction horizons.

Considering Table 3, for the discretization using 5 subintervals, the collocation scheme of [15] provides lower function values as compared to Algorithm 2, but the CPU time of the modified CMSC method is better than the one reported in [17].

Table 3: Comparative results from discretization (5 subintervals)

	Objective function	Number of optimization variables	Number of constraints
Modified CMSC	0.56817	17	12
CMSC by [17]	0.56817	18	12
Collocation algorithm by [15]	0.57302	101	86
Multiple shooting by [14]	0.56838	18	10

Table 4: Comparative results from discretization (160 subintervals)

	Objective function	Number of optimization variables	Number of constraints
Modified CMSC	0.57354	482	322
CMSC by [17]	0.57354	483	322
Collocation algorithm by [15]	0.57354	3046	2566
Multiple shooting by [14]	0.57354	483	320

As shown in Table 4, our Algorithm 2 shows a higher performance in terms of CPU time as compared to all presented algorithms, still obtaining the the same objective function value as reported by other authors.

7 Conclusion and future work

This paper presents the first prototype of a modified combined multiple shooting and collocation method. The major difference from the original version of the CMSC approach in [17, 18] consists in using precalculated derivatives and their symbolic representation. That is, in every iteration of the optimization algorithm, sensitivities are automatically available without further calculations. The optimizer is provided with symbolic derivatives which are to be evaluated at the given iterate. This leads to accurate results with speedup of the overall computation time.

This preliminary investigation shows that the implemented algorithm has a competitive performance compared to other similar investigations. There is also a potential for parallel implementation of the proposed algorithm, whereby constraints to be considered on the coefficients of the collocation polynomials inside the shooting intervals. Furthermore, the modified CMSC method will be implemented to work directly with Modelica models along with a choice of local and global nonlinear equation solvers. Hence, the proposed framework will be refined to make it highly transparent, so that end-users can solve various types of applied optimal control problems. In addition, the algorithm will be extended to handle nonlinear model predictive control (NMPC) problems.

8 Acknowledgements

This work has been supported by Model Driven Physical Systems Operation project (MODRIO) by ITEA2, No. 11004, and by the German BMBF (BMBF Förderkennzeichen: 01IS12022H).

References

[1] Åkesson, J.: Optimica—An Extension of Modelica Supporting Dynamic Optimization. 6th International Modelica Conference, March 3 - 4, 2008, Bielefeld, Germany, pp. 57-66.

[2] Åkesson, J., Årzén, K.-E., Gåfvert, M., Bergdahl, T., Tummescheit, H.: Modeling and optimization with Optimica and JModelica.org-Languages and tools for solving large-scale dynamic optimization problems. Computers and Chemical Engineering, 34(2010)11, pp. 1737-1749.

[3] Andersson, J., et. al.: Dynamic optimization with CasADi. In Proceedings of the 51st IEEE Conference on Decision and Control, Maui (USA), 2012.

[4] J. Andersson, J., Casella, F., Diehl, M.: Integration of CasADi and JModelica.org. 8th International Modelica Conference, Dresden, 2011. DOI:10.3384/ecp1106321

[5] Andersson, J., Åkesson, J., Diehl, M.: CasADi: A Symbolic Package for Automatic Differentiation and Optimal Control. Lecture Notes in Computational Science and Engineering, Vol. 87, Springer, 2012, pp. 297-307.

[6] Bachmann, B., Ochel, L., Ruge, V., Gebremedhin, M., Fritzson, P., Nezhadali, V., Eriksson, L., Sivertsson, M.: Parallel Multiple-Shooting and Collocation Optimization with OpenModelica. Proceedings of the 9th International Modelica Conference, Munich, pp. 659-668, 2012.

[7] Bartl, M., Li, P., Biegler, L. T.: Improvement of state profile accuracy in nonlinear dynamic optimization with the quasi-sequential approach. AIChE Journal, 57(2011)8, pp. 2185-2197.

[8] Biegler, L. T. Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. SIAM, 2010.

[9] Bock, H. G., Plitt, K. J. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems, Prepr. 9th IFAC World Congress, Budapest, 1984, pp. 242-247.

[10] Cuthrell, J. E., Biegler, L. T.: Simultaneous optimization and solution methods for batch reactor control profiles. Comput. Chem. Eng. 13(1989), pp. 49-62.

[11] Ferreau, H. J.: Model predictive control algorithms for applications with millisecond timescales. PhD Thesis, KU Leuven, 2011.

[12] Franke, R.: Formulation of dynamic optimization problems using Modelica and their efficient solution. 2nd International Modelica Conference, March 18 - 19, DLR, Oberpfaffenhofen, Germany, pp. 315 - 323.

[13] Hong, W., Wang, S., Li, P., Wozny, G., Biegler, L. T.: A quasi-sequential approach to large-scale dynamic optimization problems. AIChE Journal, 52(2006)1, pp. 255-268.

[14] Kirches, C., Wirsching, L., Bock, H. G., Schlöder, J. P.: Efficient Direct Multiple Shooting for Nonlinear Model Predictive Control on Long Horizons. J. Process Control, 22(2012), pp. 540-550.

[15] Magnusson, F.: Collocation Methods in JModelica.org. Master Thesis, Lund University, February 2012.

[16] Russel, R. D., Shampine, L. F.: A Collocation Method for Boundary Value Problems, Numerical Mathematic, 19(1971), pp. 1-28, Springer.

[17] Tamimi, J.: Development of the Efficient Algorithms for Model Predictive Control of Fast Systems. PhD Thesis, Technische Universität Ilmenau, VDI Verlag, 2011.

[18] Tamimi, J., Li, P.: A Combined approach to nonlinear model predictive control of fast systems. J. Process Control, 20(2010)9, pp. 1092-1102.

[19] Wächter, A., Biegler, L.T.: On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming, Mathematical Programming, Ser. A, 106(2006)1, pp. 25-57.