applied sciences

Article

Time-Optimal Trajectory Planning of Six-Axis Manipulators Based on the Improved Direct Collocation Method with FMU

Ziyao Xiong, Jianwan Ding * and Liping Chen

School of Mechanical Science& Engineering, Huazhong University of Science and Technology, Wuhan 430070, China; d201980206@hust.edu.cn (Z.X.); chenlp@hust.edu.cn (L.C.)

* Correspondence: dingjw@hust.edu.cn

Abstract: The trajectory planning method with dynamics is the key to improving the motion performance of manipulators. The optimal control method (OCM) is a key technology to solve optimal problems with dynamics. There are direct and indirect methods in OCM; indirect methods are difficult to apply to engineering applications, and so direct methods are widely applied instead. The direct collocation method (DCM) is a technology in OCM to transform an optimal control problem (OCP) to a nonlinear problem (NLP), so that plenty of solvers can be used directly. However, the general DCM, for which it has been found that the explicit form of the right-hand-side (RHS) functions of state equations of the complex system in the OCP is hard to derive, is limited to solving the OCP of three-axis manipulators. This paper proposes an improved DCM to solve the OCP of six-axis manipulators, which can find the solution of the time-optimal trajectory for the motion of six-axis manipulators based on the improved DCM. The proposed method derives the RHS equations implicitly by introducing a Functional Mock-up Unit (FMU), which simplifies the representation of the RHS equations as a black-box model, so that the DCM can be applied to the OCP of six-axis manipulators. A simulation case of a three-axis manipulator accomplished in a related study works as a reference compared with our improved method to verify the solution consistence between the DCM using the explicit RHS equations or using the implicit RHS equations, and the loss of computational efficiency is acceptable. In the meantime, a simulation solution and an experiment of six-axis manipulators, which is a novel advancement, are presented to validate the proposed method.

Keywords: FMU; Modelica; optimal control; six-axis manipulators; trajectory planning

check for updates

Citation: Xiong, Z.; Ding, J.; Chen, L. Time-Optimal Trajectory Planning of Six-Axis Manipulators Based on the Improved Direct Collocation Method with FMU. *Appl. Sci.* **2022**, *12*, 6741. https://doi.org/10.3390/app12136741

Academic Editor: José A.F.O. Correia

Received: 29 March 2022
Accepted: 30 June 2022
Published: 3 July 2022

1. Introduction

Industrial robotic manipulators constitute important modern manufacturing automatic equipment and play an important role in manufacturing. The trajectory planning method is the key to the manipulators' motion performance, and makes the control stable and agile by considering the robotic dynamics behavior [1]. The targets of trajectory planning are mainly time, energy, and jerk. They can increase productivity, save energy, and smooth the motion [2]. This paper talks about minimizing the time of trajectory planning of industrial robotic manipulators. There are some related research studies about this topic presented in this section.

A group of researchers used heuristics methods to solve these problems; for example, Buhai Shi and Jiaxiang Xu proposed an improved particle swarm optimization that transformed the problem into a parameter optimization problem of B-Spline [3]. G. Li et al. proposed an optimal trajectory planning method using a cubic polynomial curve [4], while H. Wang et al. applied a multi-segment high-order polynomial curve in the smooth trajectory planning of industrial manipulators [5]. H. Yu et al. applied an improved adaptive genetic algorithm with quintic polynomial function to realize the time-optimal trajectory planning problem [6]. B. Shi and H. Zeng applied an improved Hybrid-PSO algorithm with NURBS in Cartesian and cubic B-Spline in joint space to generate a smooth and time-optimal trajectory [7]. E. Barnett and C. Gosselin introduced the bisection algorithm to solve the trajectory planning problem of parallel manipulators along specified paths [8]. Another group of researchers solved the problem with optimal control methods; for example, Tie Zhang et al. used the method of J. E. Bobrow [9] to obtain the minimum-time trajectory along a specified path of the manipulators; moreover, they designed an input shaping algorithm to confirm the precision of the trajectory tracking [10].

However, all the above studies achieved the optimal trajectories in the state space of the system. The state space variables, such as joint positions, joint velocities, and joint accelerations of manipulators, cannot be applied as input to the real system directly. The inverse dynamics method, which usually refers to the feedback system, should be applied to transform the optimized states into the controls, such as joint torques, actuator voltage and so on. However, the controls of the system rely on the feedback system, which depends on the sample period, which may cause the system delay. We prefer the traditional optimal control method that can find the desired controls, and the result can be input to the system as feed-forward to obtain the expected motion of the system, and the performance of the system is going to be better in this way. The optimal control method (OCM) is a powerful technique for solving this kind of problem with dynamics. Related research about the optimal control is given in the following paragraphs.

J. Harzer et al. studied the OCM, solving the problem of high-oscillatory systems [11]. J. Wang et al. worked on the semi-linear parabolic optimal control problems [12]. M. Leomanni et al. worked on the time-optimal control problem for a chain of multidimensional integrators [13]. Meanwhile, K. Prag et al. reviewed the reinforcement learning approaches for adaptive data-driven optimal control frameworks, enabling the OCM with data-driven dynamics [14]. J. Zhao, Z. Yuan and S. Manna applied the OCM with data-driven dynamics in different fields, and obtained great results [15–17]. For further research, Z. Mei et al. proposed a feed-forward control method based on the hybrid inverse dynamic model that couples the data-driven model with the analytic model [18]. M. Schappler et al. realized the model-based joint impedance control on the arms of the Atlas robot [19]. J. Richalet et al. proposed a novel control strategy called model predictive control (MPC), and show its robustness [20]. C. E. Garcia et al. applied MPC in nonlinear systems, and proved it is highly effective by comparing it with the traditional feedback control method [21]. K. Czerwinski and M. Lawrynczuk proved that the dynamic matrix control algorithm, which belongs to MPC, can be used in the fast embedded system, with MPC satisfying the real-time requirements, and can achieve rapid response [22]. G. Ortiz-Torres et al. and F. D. J. Sorcia-Vazquez et al. presented a decentralized model predictive controller for different nonlinear systems [23,24]. All these studies offer a framework of control strategies based on OCM. Some studies are discussed below that use OCM to solve the trajectory planning problem of manipulators.

M. E. Kahn and B. Roth firstly worked on the minimum-time optimal control problem (OCP) for manipulators in 1971; their research [25] linearized the motion equations for the system so that they could simplify the highly nonlinear dynamics system and used the Pontryagin minimum principle to obtain an approximation of the minimum-time control.

J. E. Bobrow solved the minimum-time manipulator control problem, the path of which is specified, and the actuator torque limitations of which are known. This method finds the bang-bang switching structure and finds the optimal open-loop torques [9].

In 1988, M. Otter gave a non-academic, highly nonlinear model of a commercially available robot [26]. O Von Stryk used this model with a hybrid approach that combines a direct collocation method (DCM) and an indirect multiple shooting method to solve optimal point-to-point trajectory planning problems of robotic manipulators in the mid-1990s [27]. At this time, the optimal control method was firstly applied in three-axis robotic manipulators.

In 2010, Victor M. Becerra developed the DCM and published an open-source solver PSOPT [28]. The work of Stryk in 1988 was included in the example set of PSOPT. M. Kelly published an introduction to help others solve trajectory planning problems using DCM in 2016 [29].

It is said in [27] that the most difficult part in describing an OCP is to give the expression of the state equations of a complex dynamic system, which are generally called the right-hand-side (RHS) equations and written as:

$$\dot{x} = f(x, u, t), \tag{1}$$

where x and u refer to the states and the controls in a dynamic system; function f generally refers to the forward dynamics of the system.

As has been said in [27], the dynamics behavior of the manipulator is given either explicitly by the output of a symbolic computation system, or in an efficient implicit form of the RHS equations by the subroutine R3M2SI [30].

However, the RHS equations become more and more difficult to express as the degree of freedom of robotic manipulators increases. The work of [26] derived the explicit RHS equations of a three-axis manipulator; the work of [30] presents a general method to describe the RHS equations of manipulators wit ha tree structure in an implicit way, and the subroutine R3M2SI is an instance of this method applied to a three-axis manipulator. However, the RHS equations of six-axis manipulators are still tough to work out, and there is no research solving the OCP with the RHS equations of six-axis manipulators.

This research introduces Functional Mock-up Unit (FMU) to overcome the difficulty. Modelica is a modeling language, which builds models that can instead use the RHS equations. The Modelica model is defined as an FMU which can be regarded as a black-box function. Moreover, a standard interface criterion FMI (Functional Mock-up Interface) [31] is applied to call any specified variables in the Modelica models. Therefore, we can use the Modelica model to construct the dynamics of manipulators and call the model with FMI in an implicit form of the RHS equations. In this way, the OCP described in PSOPT with the RHS equations expressed by FMU can be solved. The solutions of the time-optimal trajectory planning problem of both three-axis and six-axis robotic manipulators are given in this paper.

The related paper flow is shown in Figure 1. The paper is organized as follows: The problem description of OCP is formulated in Section 2 for general manipulators. Section 3 introduces the improved DCM, and shows how to transcribe a continuous OCP into a finite-dimensional problem that can be solved by using an NLP algorithm. In Section 4, the simulation results of three-axis robotic manipulators named Manutec R3 model 2 and six-axis robotic manipulators named Manutec R3 model 1 in [26] are presented. Section 5 gives the experiments solutions, and Section 6 sums up the full paper.

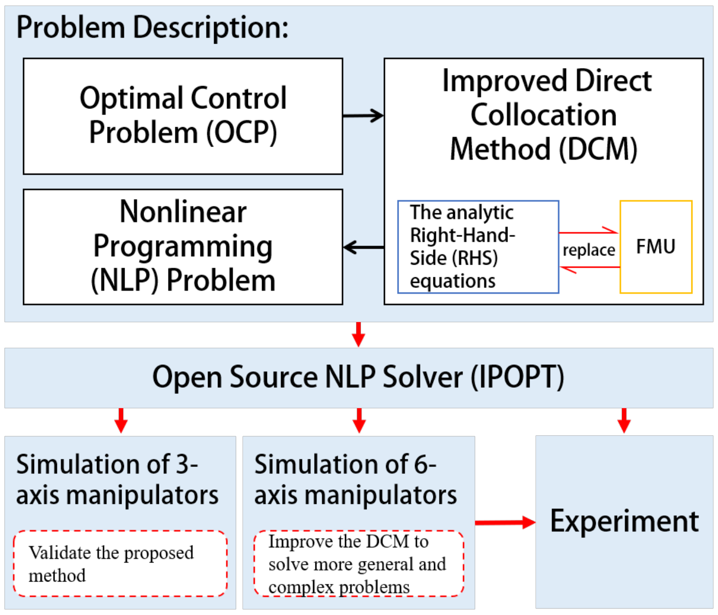

Problem Description:
Optimal Control Problem (OCP)
Improved Direct Collocation Method (DCM)
Nonlinear Programming (NLP) Problem
The analytic Right-Hand-Side (RHS) equations
replace
FMU
Open Source NLP Solver (IPOPT)
Simulation of 3-axis manipulators
Validate the proposed method
Simulation of 6-axis manipulators
Improve the DCM to solve more general and complex problems
Experiment

Figure 1. Paper flow chart.

2. Problem Statement

2.1. General Problem Formulation of Optimal Control

The minimum-time trajectory planning problem of manipulators can be described as an OCP. The general OCP formulation can be defined to find the control trajectories $\boldsymbol{u}(t), t \in [t_0, t_f]$ that minimize the following Bolza performance index:

$$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt, \tag{2}$$

subjected to the state equations (the RHS equations):

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), t \in [t_0, t_f], \tag{3}$$

the boundary conditions:

$$\boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) = 0, \tag{4}$$

and the inequality constraints:

$$\boldsymbol{C}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq 0, \tag{5}$$

where J is the index, which is the function we aim to optimize. Φ and L are the boundary cost function and process cost function. The performance index of the minimum-time trajectory planning problem discussed in this paper belongs to a boundary cost function, and the the process cost function equals zero. t_0 and t_f are the start time and the end time, respectively. $\boldsymbol{u}$ represents the controls in the system, and refers to the output torque of the joint actuators in the robotic dynamic system. $\boldsymbol{x}$ represents the states in the system, and refers to joint angles and joint velocities in the robotic dynamic system. $\dot{\boldsymbol{x}}$ represents the differential states, and refers to joint velocities and joint accelerations in the robotic dynamic system [27].

The minimum-time trajectory planning problem of robotic manipulators discussed in this paper is solved with the optimal control solver; the problem description is given in terms of the general OCP, as shown in the following section.

2.2. Minimum-Time Trajectory Planning Problem Description of Robotic Manipulators

For the minimum-time trajectory problem of industrial robots, the performance index is expressed as below:

$$J = t_f \rightarrow min. \tag{6}$$

The RHS equations can be derived from the dynamics of the system. The general dynamics equation of manipulators is expressed on this function:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{q} + \boldsymbol{v}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}), \tag{7}$$

where $\boldsymbol{q}$ indicates the states representing the relative angle between adjacent arms, $\boldsymbol{\tau}$ represents the controls representing the output torque of the joint actuators, $M(\boldsymbol{q})$ is the positive definite and symmetric matrix of the moment of inertia, $\boldsymbol{v}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the moment caused by Coriolis and centrifugal forces, and $\boldsymbol{g}(\boldsymbol{q})$ is the moment caused by gravitational forces.

After several transposition operations on (7), the RHS equations become:

$$\ddot{\boldsymbol{q}} = M^{-1}(\boldsymbol{q})[\boldsymbol{\tau} - v(\boldsymbol{q}, \dot{\boldsymbol{q}}) - g(\boldsymbol{q})]. \tag{8}$$

It is quite important to derive the RHS equations, because it helps to predict the states of the system. Finding methods to instead the explicit expression of (8) is also the main difficulty in solving the OCP.

As a result, the state and the state equations of robotic manipulators can be written as below:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{pmatrix}, \dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ M^{-1}(\boldsymbol{q})[\boldsymbol{\tau} - v(\boldsymbol{q}, \dot{\boldsymbol{q}}) - g(\boldsymbol{q})] \end{pmatrix}. \tag{9}$$

The boundary conditions $\boldsymbol{\phi}$ are taken into consideration as below:

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_0, \boldsymbol{q}(t_f) = \boldsymbol{q}_f, \tag{10}$$

$$\dot{\boldsymbol{q}}(t_0) = \dot{\boldsymbol{q}}_0, \boldsymbol{q}(t_f) = \dot{\boldsymbol{q}}_f. \tag{11}$$

Generally, $\boldsymbol{q}_0$ and $\boldsymbol{q}_f$ refer to the start and the end, and they are both constant and can be defined by users. The stationary boundary conditions are taken into consideration, i.e., $\dot{\boldsymbol{q}}_0 = \dot{\boldsymbol{q}}_f = 0$.

The inequality constraints can be categorized into the following two kinds:

- The control constraints on the torque voltage;

$$\mid \boldsymbol{u}_i(t) \mid \leq \boldsymbol{u}_{i,max}, i = 1...n. \tag{12}$$

- The state constraints on the angles and the angular velocities.

$$\mid \boldsymbol{q}_i(t) \mid \leq \boldsymbol{q}_{i,max}, i = 1...n. \tag{13}$$

$$\mid \dot{\boldsymbol{q}}_i(t) \mid \leq \dot{\boldsymbol{q}}_{i,max}, i = 1...n. \tag{14}$$

where i refers to the number of the robotic link, and n refers to the max number of the robotic link, which is equal to the degrees of freedom of robotic manipulators.

3. DCM with FMU

The OCP proposed in the last section is a kind of variational problem; the Pontryagin minimum principle offers an explicit solving method. However, when it comes to the engineering application, the related OCPs are too difficult to solve in this way. Therefore, the DCM is adopted here to transcribe a continuous OCP into a discrete NLP problem, so that a series of NLP solvers are available to be applied to solve the problem.

However, as said in last section, the RHS equations are still troublesome. In this section, a new method is proposed, where the FMU is applied to improve the DCM. The target DCM makes it possible to solve the problem with an NLP solver; the FMU can be applied to express the RHS equations of the complex system, such as six-axis robotic manipulators.

The collocation methods are divided into local and global methods according to the information used to approximate the RHS equations, which are the derivatives of the states in essence. While there are a lot of numerical methods using the local information to obtain the derivatives of a function, including the Euler method, trapezoidal method, Hermite–Simpson method and classical Runge–Kutta method, pseudospectral methods use, in contrast, the global information over samples of the whole domain of the function to approximate their derivatives at chosen points. Generally, the global pseudospectral methods offers a faster convergence. There are three kinds of pseudospectral methods, including the Legendre Pseudospectral Method (LPM), Gauss Pseudospectral Method (GPM) and Radau Pseudospectral Method (RPM). When applied to an OCP, the terminal of which is not completely free, the Legendre Pseudospectral Method allows better convergence [32–35]. As a result, this work applies the Legendre Pseudospectral Method.

The processes of the improved LPM method with FMU are shown in the following sections.

3.1. Original LPM Method Processes

This part shows how the LPM method transforms an OCP to an NLP problem.

3.1.1. Time Domain Transformation

The LPM method discretizes the states and the controls on the Legendre–Gauss–Lobatto (LGL) points (note that $P_N(T)$ is an N-order Legendre polynomial, and $\dot{L}_N(T)$ is the derivative of $L_N(T)$). Because the definition domain of Legendre polynomials is interval $[-1, 1]$, the LGL points can be given as

$$T_0 = -1, T_1, T_2, \ldots, T_{N-1}, T_N = 1, \tag{15}$$

where $T_1, T_2, \ldots, T_{N-1}$ are the roots of $\dot{L}_N(T)$. The transformation function between the real time interval $[t_0, t_f]$ and the definition interval $[-1, 1]$ are shown as follows:

$$T = \frac{2t - t_f - t_0}{t_f - t_0}, T \in [-1, 1]. \tag{16}$$

3.1.2. Collocation and Discretization

The collocation points are equivalent to the LGL points. Discretizing the states and the controls in every point to generate the discrete states $X_0, X_1, \ldots, X_N$ and the discrete controls $U_0, U_1, \ldots, U_N$. (N is the number of the collocation points). According to the Lagrange interpolation method, the approximations of the states and the controls are determined as below:

$$\boldsymbol{x}(T) \approx \boldsymbol{X} = \sum_{i=0}^{N} L_i(T)\boldsymbol{X}_i, \tag{17}$$

$$\boldsymbol{u}(T) \approx \boldsymbol{U} = \sum_{i=0}^{N} L_i(T)\boldsymbol{U}_i, \tag{18}$$

$L_i(T)$ is the basis function of Lagrange interpolation function, and is shown as below:

$$L_i(T) = \prod_{j=0, j\neq i}^{N} \frac{T - T_j}{T_i - T_j}. \tag{19}$$

In this way, differential operations in the state equations and the integral operation in the performance index are transformed into an algebraic operation, shown in the next step.

3.1.3. The Transformation of RHS Equations

Once the estimation of the states comes out, the differential values of the states can be estimated as below:

$$\dot{\boldsymbol{x}}(T_k) \approx \dot{\boldsymbol{X}}_k = \sum_{i=0}^{N} \dot{L}_i(T_k)\boldsymbol{X}_i = \sum_{i=0}^{N} D_{ki}\boldsymbol{X}_i, \tag{20}$$

where $k = 0, 1, \ldots, N$ and the differential matrix D_{ki} represent the differential values of every basis function of the Lagrange interpolation function in every LGL point, as shown below:

$$D_{ki} = \begin{cases} \frac{P_N(T_k)}{P_N(T_i)(T_k - T_i)}, & i \neq k; \\ -\frac{N(N+1)}{4}, & i = k = 0; \\ \frac{N(4N+1)}{4}, & i = k = N; \\ 0, & otherwise; \end{cases} \tag{21}$$

Now, the RHS equations can be written as:

$$\sum_{i=0}^{N} D_{ki}\boldsymbol{X}_i - \frac{t_f - t_0}{2}\boldsymbol{f}(\boldsymbol{X}_k, \boldsymbol{U}_k, \tau_k) = 0, k = 0, 1, ..., N. \tag{22}$$

3.1.4. Integral Equations Transformation

The performance index after transformation using the Gauss–Lobatto integral method is shown as below:

$$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \frac{t_f - t_0}{2} \sum_{i=0}^{N} \omega_i L(\boldsymbol{X}_i, \boldsymbol{U}_i, T_i), \tag{23}$$

where ω_i is the integral weight, defined as:

$$\omega_i = \frac{2}{N(N+1)P_N^2(T_i)}. \tag{24}$$

3.1.5. The Constraints Transformation

The boundary conditions and the inequality constraints are discretized as below:

$$\boldsymbol{\phi}(\boldsymbol{X}_0, \boldsymbol{X}_N, t_0, t_f) \leq \boldsymbol{\delta}, \tag{25}$$

$$\boldsymbol{C}(\boldsymbol{X}_k, \boldsymbol{U}_k, T_k) \leq \boldsymbol{0}, k = 0, 1, ..., N, \tag{26}$$

There is no absolute equality between any values in a numerical calculation; therefore, the slack $\boldsymbol{\delta}$ is introduced to avoid system instability.

To sum up, the original OCP (2)–(5) transform into (23), (22), (25), and (26), respectively. However, Equation (22) still includes the right side of the RHS function $\boldsymbol{f}(\boldsymbol{X}_k, \boldsymbol{U}_k, \tau_k)$. Therefore, FMU is introduced to substitute the RHS equations in the next step.

3.2. The Substitution of RHS Equations with FMU

This step aims to rebuild (22) with an FMU as a substitution of the RHS equations. Modelica is used here to build FMU; FMI offers supports the interface to use FMU as the RHS equations.

Modelica is a language for physical modeling. The design approach builds on non-casual modeling with true equations and the use of object-oriented constructs to facilitate the reuse of modeling knowledge. As a result, it is convenient to write the dynamics equations in a freer form of which you do not have to guarantee there is one and only one variable on the left-hand side. With this property, the RHS equations can be derived implicitly by constructing the FMU of the robotic dynamics in Newton–Euler form.

The robot dynamics model is constructed by a Modelica based software Mworks supporting drag-and-drop modeling [36]. As a result, the dynamics in the Newton–Euler form that have clear physical significance can be expressed easily in Mworks. The model is shown in Figure 2.

Where the components tau1–tau6 represent the output torques of motors, the components inertia1–inertia6 represent the inertias of motors; the components r1–r6 represent the revolute joints in manipulators which shows the relationship of joint coordinates fixed on adjacent links; the components b1–b6 represent the link bodies in the manipulators which contain the iterative Newton–Euler dynamic formulation [37] of adjacent links. The outputs of components inertia 1–inertia6 are the output torques of gears, which are the input of components r1–r6.

The dynamics in Newton–Euler form are constructed by operations of dragging–dropping function blocks. The components' revolute contains information of kinematic forward derivation in the Newton–Euler method, and the component link body contains physical properties of the robotic body and information of dynamic backward derivation in the Newton–Euler Method.

The Modelica model can be transformed into FMU with FMI. The structure chart of FMU is shown in Figure 3.

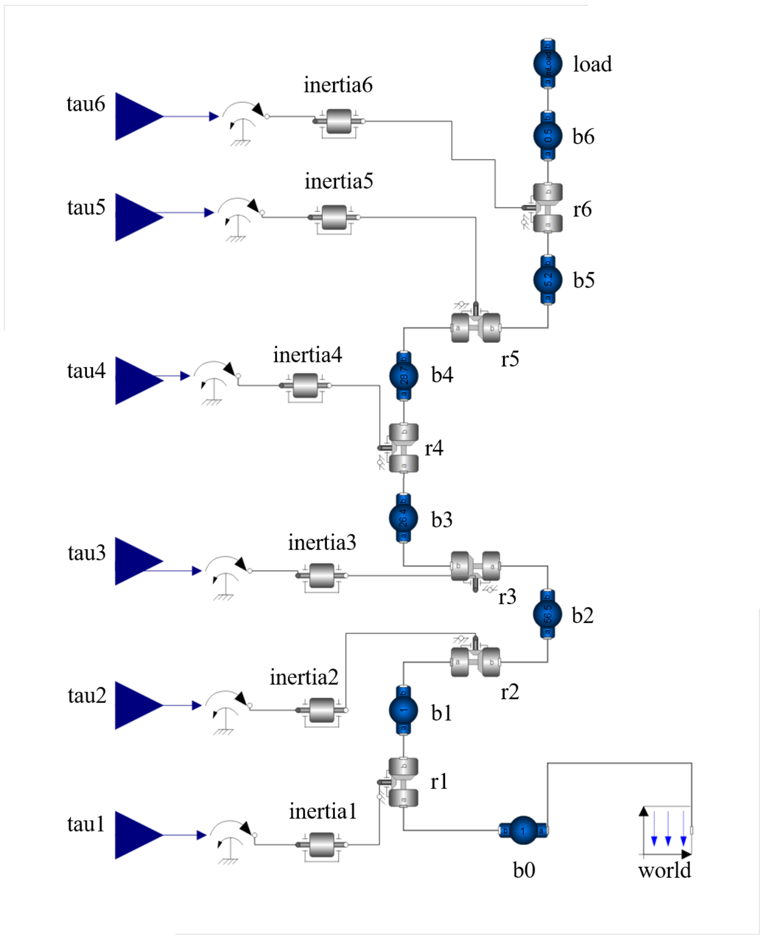


Figure 2. Modelica model of the manipulator.

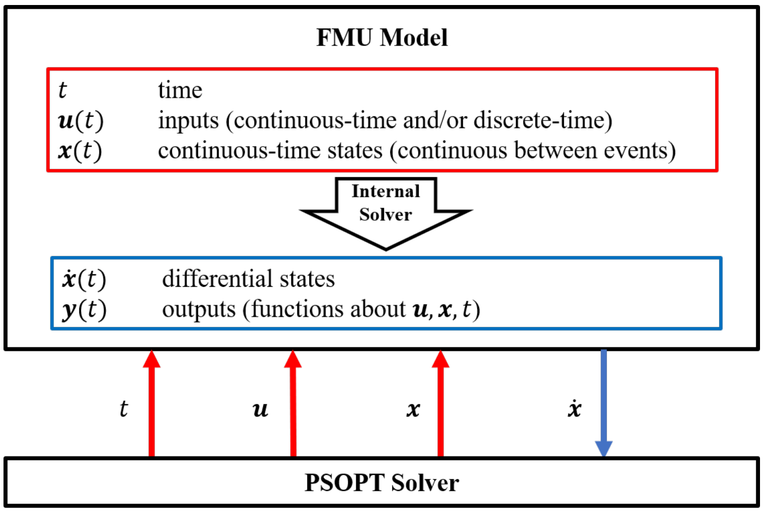

FMU Model
t time
$\boldsymbol{u}(t)$ inputs (continuous-time and/or discrete-time)
$\boldsymbol{x}(t)$ continuous-time states (continuous between events)
Internal Solver
$\dot{\boldsymbol{x}}(t)$ differential states
$\boldsymbol{y}(t)$ outputs (functions about $\boldsymbol{u}, \boldsymbol{x}, t$)
t
$\boldsymbol{u}$
$\boldsymbol{x}$
$\dot{\boldsymbol{x}}$
PSOPT Solver

Figure 3. FMU structure chart.

FMU is a black-box model used here to replace the analytic RHS equations; the input and output flows are presented in Figure 3. According to Figure 3, the FMU model, where the inputs are the present time t, the present controls $\boldsymbol{u}$, and the present states $\boldsymbol{x}$, can derive the present differential states $\dot{x}$ by the internal solver. Therefore, it can be applied to the OCP solver as the RHS equations, and the difficulty in expressing the RHS equations is overcome.

As a result, two black-box functions are used to express the dynamics of manipulators; i.e., Equations (7)–(9) are rewritten in the FMU form below:

$$\boldsymbol{\tau} = \text{FMU}_{inverse}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, t), \tag{27}$$

$$\ddot{\boldsymbol{q}} = \text{FMU}_{forward}(\boldsymbol{\tau}, \boldsymbol{q}, \dot{\boldsymbol{q}}, t), \tag{28}$$

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{pmatrix}, \dot{\boldsymbol{x}}_{\text{FMU}} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \text{FMU}_{forward}(\boldsymbol{\tau}, \boldsymbol{q}, \dot{\boldsymbol{q}}, t) \end{pmatrix}. \tag{29}$$

By discretizing (29), we have:

$$\begin{cases} \boldsymbol{X}_i = \begin{pmatrix} \boldsymbol{q}(T_i) \\ \dot{\boldsymbol{q}}(T_i) \end{pmatrix}, \\ \dot{\boldsymbol{X}}_{\text{FMU},i} = \begin{pmatrix} \dot{\boldsymbol{q}}(T_i) \\ \text{FMU}_{forward}(\boldsymbol{\tau}(T_i), \boldsymbol{q}(T_i), \dot{\boldsymbol{q}}(T_i), T_i) \end{pmatrix}. \end{cases} \tag{30}$$

Then, the state equations are given by substituting (30) into (22); that is,

$$\left\| \sum_{i=0}^{N} D_{ki}\boldsymbol{X}_i - \frac{t_f - t_0}{2}\dot{\boldsymbol{X}}_{\text{FMU},i} \right\|_{\infty} \leq \delta, k = 0, 1, \ldots, N. \tag{31}$$

All in all, the FMU model of manipulators, which is built by using the Newton–Euler method, can be applied in the DCM to determine the value of the RHS equations in (22). $\dot{\boldsymbol{X}}_{\text{FMU},i}$ in (30), which is derived in the FMU, is used to obtain the values of the RHS equations in (22), which are the value of $\boldsymbol{f}(\boldsymbol{X}_k, \boldsymbol{U}_k, \boldsymbol{\tau}_k), k = 0, 1, \ldots, N$.

3.3. Final NLP Problem Description

After adding FMU into the LPM method, the final NLP problem formula of the time-optimal trajectory planning problem in terms of (2)–(5) is presented below:

Minimize:

$$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \frac{t_f - t_0}{2}\sum_{i=0}^{N} \omega_i L(\boldsymbol{X}_i, \boldsymbol{U}_i, T_i). \tag{32}$$

This is subjected to:

$$\left\| \sum_{i=0}^{N} D_{ki}\boldsymbol{X}_i - \frac{t_f - t_0}{2}\dot{\boldsymbol{X}}_{\text{FMU},i} \right\|_{\infty} \leq \delta, \tag{33}$$

$$\boldsymbol{\phi}(\boldsymbol{X}_0, t_0, \boldsymbol{X}_N, t_f) \leq \delta, \tag{34}$$

$$\boldsymbol{C}(\boldsymbol{X}_k, \boldsymbol{U}_k, T_k) \leq 0, k = 0, 1, ..., N. \tag{35}$$

This final formula of the problem statement makes it possible to solve the transformed OCP by NLP solvers. IPOPT is the NLP solver applied here.

The problem formula, which is transformed from the original OCP to the NLP problem with FMU, is listed in Table A1.

4. Simulation Results

This section gives two cases using the improved method proposed in the last section. The three-axis case results allow us to validate the improved method using the same case in the reference research [27]. Then, the six-axis case results are displayed.

4.1. The Manutec R3 Model 2

Manutec R3 is an industrial robot with 6 degrees of freedom; it is said that the first 3 d.o.f are responsible for the position and the last 3 d.o.f for the orientation of the tool center point frame [27]. For simplicity, the Manutec R3 model 2 (R3M2) focused on the first 3 d.o.f case. The relative physical parameters are displayed in [26], the robotic manipulator with no load is taken into consideration, and the terminal time t_f is set as a free variable to find the shortest working time. There are some conditions for this, as follows:

The initial state conditions, which give the positions of the start point and the end point and angular velocities on both points, are:

$$\boldsymbol{q}(0) = \begin{pmatrix} 0.00 \\ -1.50 \\ 0.00 \end{pmatrix}, \boldsymbol{q}(t_f) = \begin{pmatrix} 1.00 \\ -1.95 \\ 1.00 \end{pmatrix}, \tag{36}$$

$$\dot{\boldsymbol{q}}(0) = \boldsymbol{0}, \dot{\boldsymbol{q}}(t_f) = \boldsymbol{0}. \tag{37}$$

The boundary conditions give constant boundaries of the controls and the states. It is noted that the motors in the robot are modeled as ideal voltage–torque converters without any dynamic effects; in other words, there is a linear relationship between voltage and torque. The linear converter coefficient $\boldsymbol{LC}$ can be found in [26].

The controls boundaries of voltage $\boldsymbol{u}_{max}$, and the linear converter coefficient $\boldsymbol{LC}$ are shown as

$$u_{i,max} = 7.5(\mathrm{V}), i = 1, 2, 3. \tag{38}$$

$$LC[3] = [-126, 252, 72]^T (\mathrm{N} \cdot \mathrm{m/V}). \tag{39}$$

The torques boundaries $\boldsymbol{\tau}_{max}$ can be calculated by

$$\tau_{i,max} = LC[i] \times u_{i,max} (\mathrm{N} \cdot \mathrm{m}), i = 1, 2, 3. \tag{40}$$

The states' boundary conditions including the boundaries of positions $\boldsymbol{q}_{max}$ and angular velocities $\dot{\boldsymbol{q}}_{max}$ are as follows:

$$\begin{cases} q_{1,max} = 2.97(\mathrm{rad}), & \dot{q}_{1,max} = 3.00(\mathrm{rad/s}), \\ q_{2,max} = 2.01(\mathrm{rad}), & \dot{q}_{2,max} = 1.50(\mathrm{rad/s}), \\ q_{3,max} = 2.86(\mathrm{rad}), & \dot{q}_{3,max} = 5.20(\mathrm{rad/s}), \end{cases} \tag{41}$$

The LPM method is initialized with 20 and 100 nodes, and the initialization of the state $\widetilde{\boldsymbol{q}}_n$, $\dot{\widetilde{\boldsymbol{q}}}_n$ on each node is set as:

$$\widetilde{\boldsymbol{q}}_n = \boldsymbol{q}(0) + \frac{n-1}{N-1}(\boldsymbol{q}(t_f) - \boldsymbol{q}(t_0)), \tag{42}$$

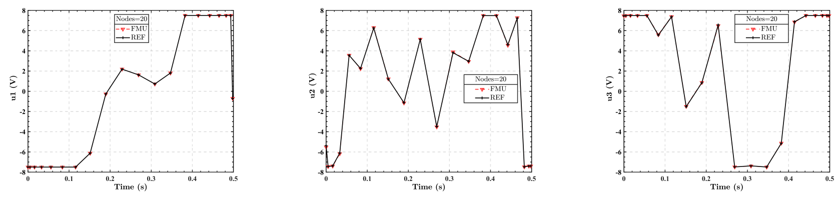
$$\dot{\widetilde{\boldsymbol{q}}}_n = \boldsymbol{q}(\dot{0}) + \frac{n-1}{N-1}(\dot{\boldsymbol{q}}(t_f) - \dot{\boldsymbol{q}}(t_0)). \tag{43}$$

The initial control $\widetilde{\boldsymbol{u}}_n$ on each node is defined as zero,

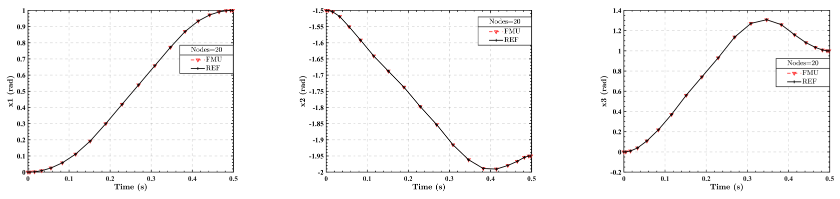
$$\widetilde{\boldsymbol{u}}_n(t) = \boldsymbol{0}. \tag{44}$$

where n represents the sequence number of the node, N represents the number of all nodes, $\widetilde{\boldsymbol{q}}_n(t)$ represents the initial angular position on the node n, and $\dot{\widetilde{\boldsymbol{q}}}_n(t)$ represents the initial angular velocity on the node n.

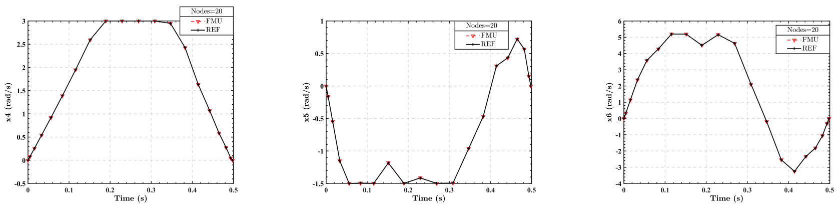
After setting the initialized conditions of the problem, the result solutions are shown in Figure 4 and Table 1. It should be noted that the REF model of R3M2 uses the analytic RHS equations, and the FMU model of R3M2 uses the black-box model of FMU to replace the analytic RHS equations.



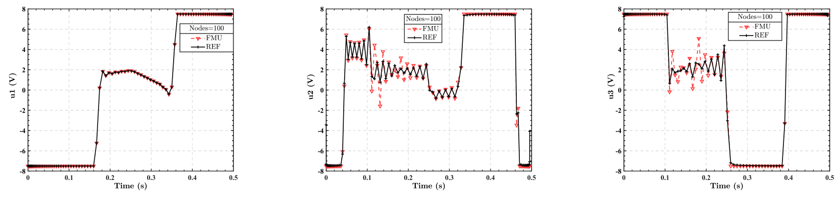
(**a**) $u - t$



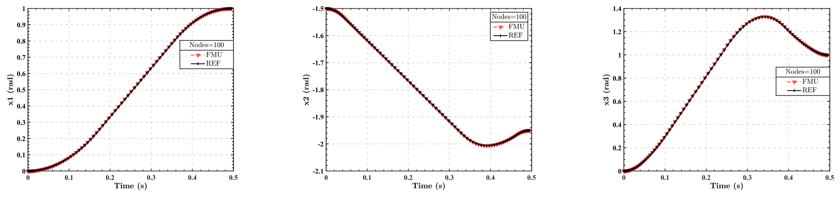
(**b**) $q - t$



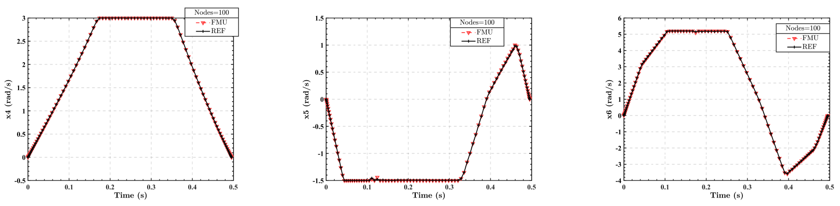
(**c**) $\dot{q} - t$



(**d**) $u - t$



(**e**) $q - t$



(**f**) $\dot{q} - t$

Figure 4. The solutions of the R3M2. Subgraphs (**a**–**c**) are the solutions using LPM method with 20 nodes; Subgraphs (**d**–**f**) are the solutions using LPM method with 100 nodes. Subgraphs (**a**,**d**) print the curve of the input voltage of joint actuators. Subgraph (**b**,**e**) print the curve of the joint positions. Subgraph (**c**,**f**) print the curve of the joint velocities. The red curves with triangles ($- \bigtriangledown -$) refer to the solutions of the improved method with FMU. The black curves with plus signs ($- + -$) refer to the reference solution.

Table 1. Comparison of results of the improved method with FMU and the reference research.

Model(R3M2)	Nodes of LPM Method	Total Cost Time of CPU (s)	Performance Index (Terminal Time t_f) (s)
REF	20 100	2.221 196.933	0.498 0.495
FMU	20 100	2.144 125.136	0.498 0.495

Figure 4 displays the solution of the trajectory planning problem; the figures of desire actuator current, joint position and joint velocity are given in the time domain. A different number of time nodes are considered. From Figure 4, the results of the improved method with FMU coincide with the results of the reference research completely when the LPM method with 20 nodes is applied. When it comes to applying the LPM method with 100 nodes, the results of both are still highly consistent. What is more, from Table 1, the optimized performance indexes are all the same. Although the proposed method needs extra time to analyze FMU, which is a black-box function, so that the value of FMU can be applied in the DCM as the RHS equation, the total cost time of the CPU of the proposed method and the reference research is of the same order of magnitude. Furthermore, the decrease in nodes in the LPM method has little impact on the result of performance index, but can greatly improve the efficiency of calculation. This property can help us to obtain good initial values or analyze the structure of results quickly.

To sum up, the proposed method in this work is not only validated, but also proved to be highly efficient.

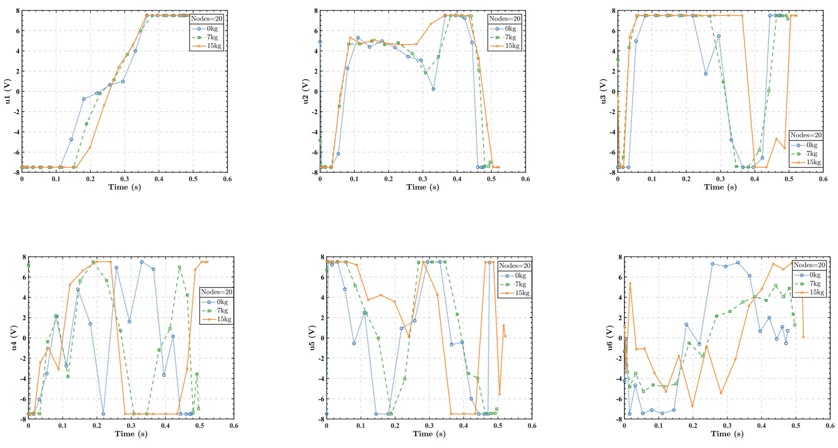
4.2. The Manutec R3 Model 1

The Manutec r3 model 1 (R3M1) is a kind of six-axis robotic manipulator constructed by Otter and Türk. However, the dynamics of the model are not presented in the literature. The proposed method in this work fills this gap. The exact boundary conditions and constraints are displayed in Table A2 to solve this problem.

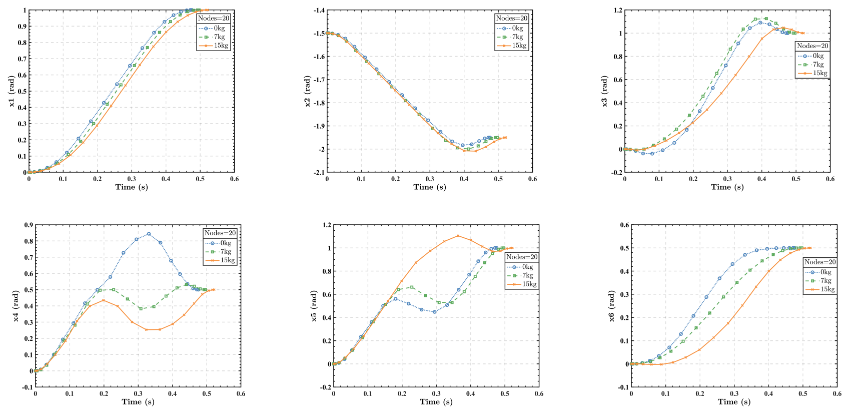
The terminal time t_f, the number of nodes, the discrete initial states and the discrete initial controls are set as usual. The loads are set to 0 kg, 7 kg, and 15 kg, respectively. The results of the proposed method are displayed in Figure 5 and Table 2, and the impact of different loads is analyzed.

Table 2. Comparison of results of the improved method with different loads.

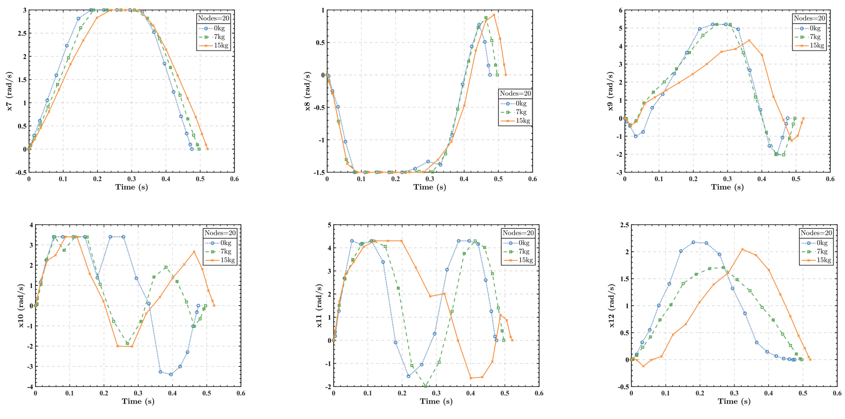
Loads of R3M1	Nodes of LPM Method	Total Cost Time of CPU (s)	Performance Index (Terminal Time) (s)
0 kg	20 100	16.644 468.226	0.476 0.475
7 kg	20 100	8.144 383.118	0.497 0.496
15 kg	20 100	15.319 469.191	0.522 0.521



(**a**) $u - t$

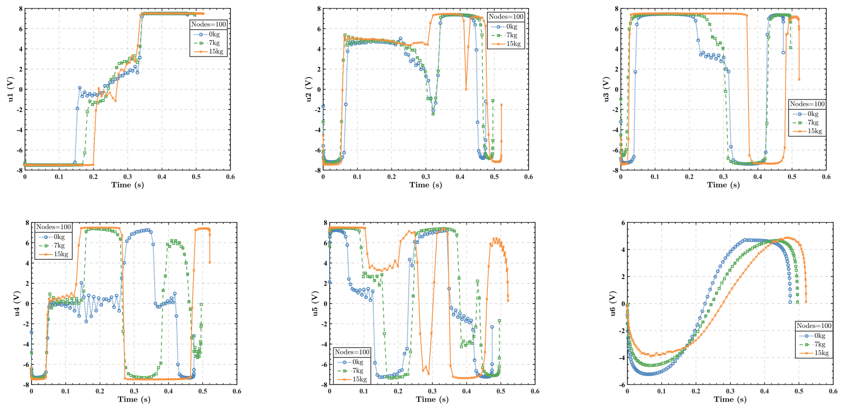


(**b**) $q - t$

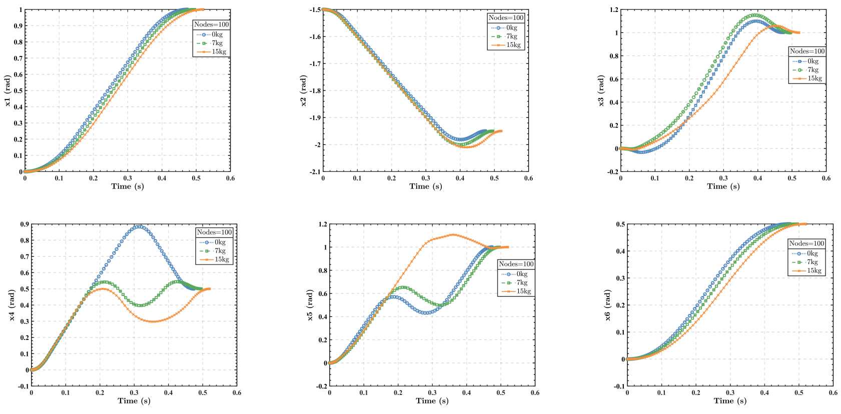


(**c**) $\dot{q} - t$

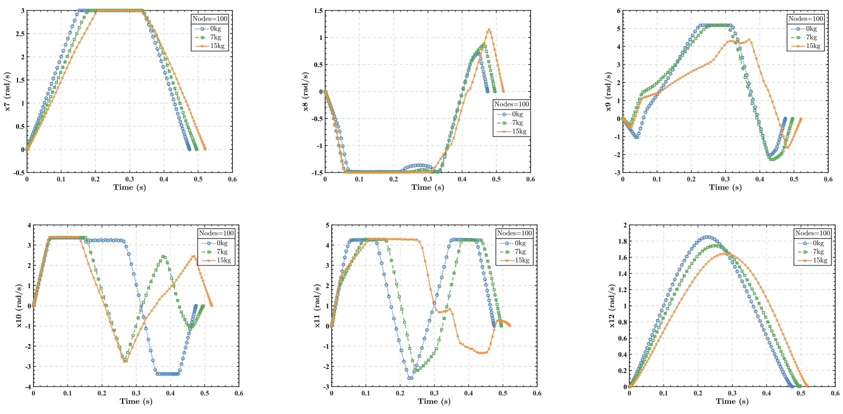
Figure 5. *Cont.*



(**d**) $u - t$



(**e**) $q - t$



(**f**) $\dot{q} - t$

Figure 5. The solutions of the R3M1. Subgraphs (**a**–**c**) are the solutions using LPM method with 20 nodes; subgraphs (**d**–**f**) are the solutions using LPM method with 100 nodes. Subgraphs (**a**,**d**) and print the curve of the input voltage of joint actuators. Subgraphs (**b**,**e**) print the curve of the joint positions. Subgraphs (**c**,**f**) print the curve of the joint velocities. The blue curves with circles $(-\text{o}-)$ refer to the solutions with 0 kg load. The green curves with squares $(-\square-)$ refer to the solutions with 7 kg load. The orange curves with crosses $(-\text{x}-)$ refer to the solutions with 15 kg load.

Figure 5 not only has the same content as Figure 4, but also displays the results of the R3M1 manipulators with different loads. It is found that, with the increase in the load, the optimized terminal time t_f increases. Joints 4 and 5 are influenced by loads greatly.

What is more, the oscillating behavior of the controls caused by the discrete property is common for a collocation method. Therefore, a better approximation of the controls can be found by applying a huge number of nodes in the LPM method, or by taking the switching structure into account to transform the original one-stage problem to a multi-stage problem [38]. According to Table 2, the calculation time of the first plan increases exponentially. However, the switching structure can be generally figured out even with the LPM method with fewer nodes. Therefore, the second plan has broader application prospects.

5. Experiment

ROCR6, a type of six-axis robotic manipulator from the company Si Valley in China, is applied in the experiment. The related MDH parameters and kinetic parameters of the robotic manipulators are listed in the Tables 3 and A3.

Table 3. MDH parameters.

Link Number	Link Length a (m)	Link Offset D (m)	Link Twist α (rad)	Joint Angle θ (Rad)
1	0.1223	0	0	0
2	0	0	$\pi/2$	$-\pi/2$
3	0	-0.270	0	0
4	0.1233	-0.253	0	$-\pi/2$
5	0.1071	0	$\pi/2$	0
6	0.0991	0	$-\pi/2$	0

In the next step, the improved DCM is applied to solve the problem of the time-optimal trajectory of the manipulator. The OCP description is given as Table A4.

It is noted that the state equations referring to (30) can be expressed after constructing the FMU of ROCR6. The unit of $\boldsymbol{q}$ is rad, the unit of $\dot{\boldsymbol{q}}$ is rad/s, and the unit of $\boldsymbol{u}$ is A. The output torques of actuators are proportional to the current $\boldsymbol{u}$, where the proportionality coefficient is 101 N·m/A.

The resulting minimum time is 0.903 s. The resulting trajectory of the improved PSOPT is input into ROCR6, which was compared with another two cases in this experiment to prove the validity of the result. One is inputting the default trajectory built into the manipulator and another is inputting the same path with a shorter time. The comparison results are shown in the next section.

5.1. Comparison with the Default Trajectory

The default trajectory applies the maximum velocity trajectory planning method. As Figure 6 shows, the joint position of each joint is given, and the right side of the figure identifies the joint number. The different maximum velocity trajectories of each joint are shown as red lines in Figure 6. The experiment shows that the maximum velocity can differ from 0.163 rad/s to 1.501 rad/s. The lower bound of the maximum velocity is not fixed, but there is an interruption caused by self-collision in the upper bound case, which means a bigger upper bound of more than 1.501 rad/s is not allowed in the system. However, the OCP case achieves a maximum velocity more than 1.501 rad/s. This shows that the dynamics of the system are taken into consideration properly, and realizes better dynamic performance.

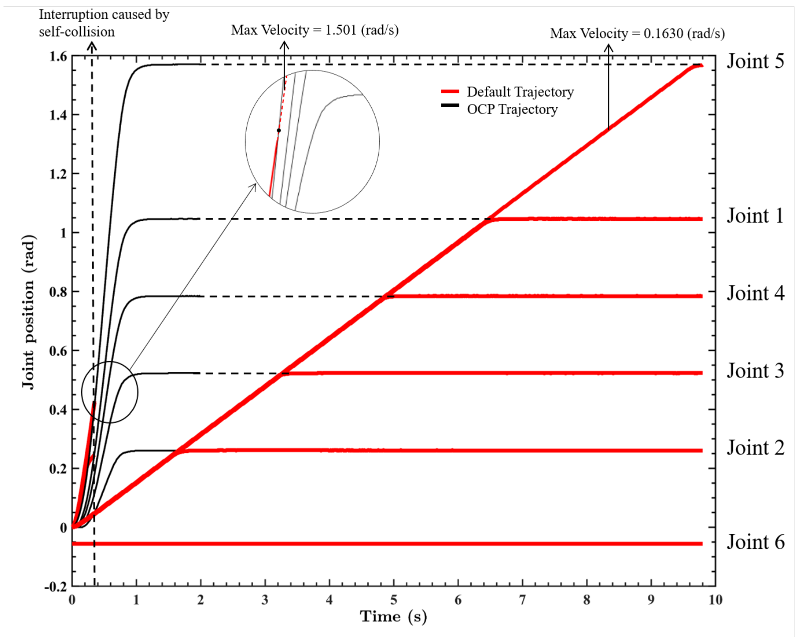


Figure 6. The OCP trajectory and default trajectory with different maximum velocity in the experiment.

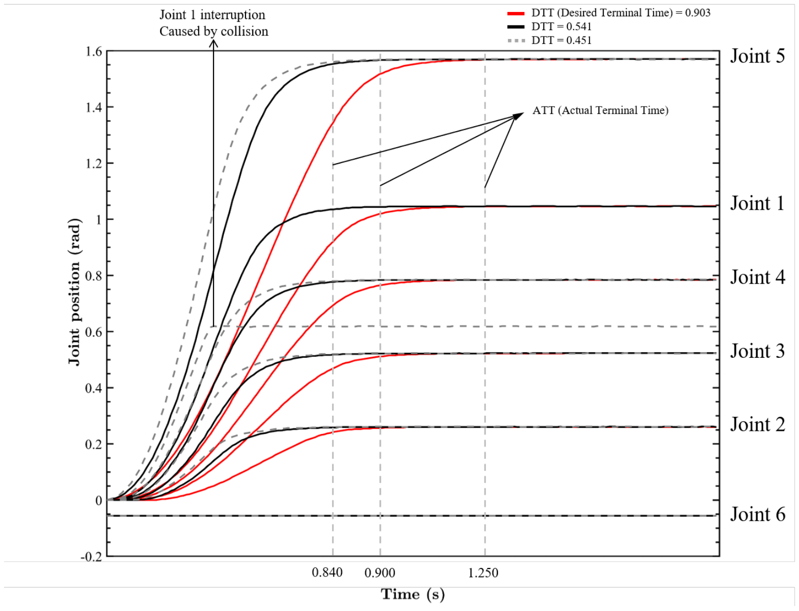
5.2. Comparison with the OCP Path with Shorter DTT

Several cases with a trajectory Desired Terminal Time (DTT) shorter than or equal to 0.903 s are taken into consideration in the experiment, including 0.451 s, 0.541 s and 0.903 s. The experiments are executed three times for each case. The Actual Terminal Time (ATT) is shown in Table 4. The executive time increased by 81.08%, 64.51% and 31.79%. Because the model is not exactly the same as the experimental robotic manipulator, ATT is always longer than DTT. As shown in Figure 7a, the joint position of each joint in each experiment is given, and we find that the shorter the time, the higher the slope of the trajectory. What is more, when DTT = 0.451 s, the trajectory of joint 1 is interrupted half way because of self-collision, which means the trajectory with a time shorter than 0.451 s is not allowed.

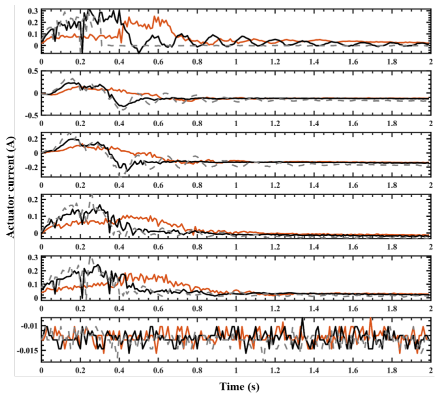
Table 4. Table of DTT and ATT in the experiments.

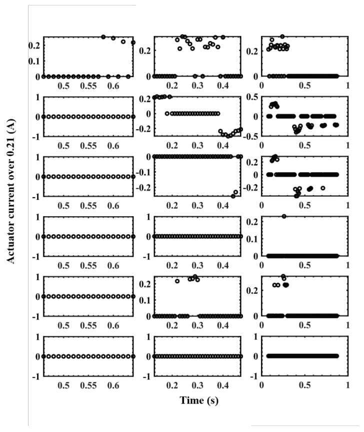
DTT (s)	Order	ATT (s)	Rate	
	1	1140	26.25%	
0.903	2	1180	30.68%	31.79%
	3	1250	38.43%	
	1	0.840	55.27%	
0.541	2	0.930	71.90%	64.51%
	3	0.900	66.36%	
	1	0.820	81.82%	
0.903	2	0.790	75.17%	81.08%
	3	0.840	86.25%	

The current of each joint actuator is shown in Figure 7b, and the current over 0.21 A is marked in Figure 7c. It is found that the current of the standard OCP trajectory is stable; only four sampling points in joint 1 are over the rated maximum. As a result, the motion is quick and smooth without too much vibration. However, the current of the OCP path with a shorter DTT becomes not stable any more, and the period of current over rated maximum increases greatly, which causes violent vibration. It was found that the case with the same path and shorter time requires higher voltage to achieve the trajectory, which may go beyond the nominal voltage and cause a vibration. The default case takes a much longer time to finish the motion, and the voltage curve is more smooth, which means the capabilities of the manipulator are not fully utilized. As the result, the validity of the proposed method is proved.



(**a**)





(**b**) (**c**)

Figure 7. Experiment solutions of the OCP path with shorter DTT. In (**a**,**b**), the DDT of 0.903 s, 0.541 s and 0.451 s are refered to the red solid line, the black solid line and the gray dash line respectively. In (**c**), the DTT of the first column is 0.903 s, the DTT of the second column is 0.541 s, and the DTT of the third column is 0.451 s.

6. Conclusions

An improved method based on the traditional collocation method with FMU is proposed in this paper. This proposed method is validated by comparing the results to the simulation accomplished in PSOPT, and is proved to be highly efficient. At the same time, it is demonstrated that this method is available to solve the six-axis robotic manipulator problem, which was not solved with the traditional method before. By considering different loads in the six-axis case, there is an obvious impact on joints 4 and 5, which makes the input voltage of the actuator shift between the minimum and maximum values more frequently. A motors' life might therefore be affected. The experiments show that the OCP trajectory can stimulate the performance of the robotic manipulator by achieving a larger maximum of velocity, and extend the motors' life by controlling the currents under the rated maximum. Moreover, the calculation time increases exponentially when applying the LPM method with more nodes. Therefore, applying the switching structure, each stage of which uses the LPM method with fewer nodes, is more useful to eliminate the oscillating behavior of the controls.

For further research, we can study the effects of other different numerical integral methods besides LPM in the DCM on the solution, and study how to improve the proposed method so that it can be applied for real-time implementations. These studies may have great value.

Author Contributions: Conceptualization, Z.X. and J.D.; methodology, J.D. and Z.X.; software, Z.X.; validation, J.D. and Z.X.; formal analysis, Z.X.; investigation, Z.X.; resources, L.C. and J.D.; data curation, Z.X.; writing—original draft preparation, Z.X.; writing—review and editing, J.D.; visualization, Z.X.; supervision, L.C.; project administration, L.C. and J.D.; funding acquisition, L.C.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China under grant number 2018YFB1701702 and by the National Key R&D Program of China under grant number 2019YFB1706501.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article. The data presented in this study can be requested from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Problem Formulas List.

		Problem Formula
OCP	Performance Index	$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt,$
	The State Equations	$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t), t \in [t_0, t_f],$
	Boundary Conditions	$\boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) = 0,$
	Inequality Constraints	$\boldsymbol{C}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq 0.$
NLP	Performance Index	$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \frac{t_f - t_0}{2} \sum_{i=0}^{N} \omega_i L(\boldsymbol{X}_i, \boldsymbol{U}_i, T_i),$
	The State Equations	$\left\Vert \sum_{i=0}^{N} D_{ki}\boldsymbol{X}_i - \frac{t_f - t_0}{2} \boldsymbol{f}(\boldsymbol{X}_k, \boldsymbol{U}_k, \tau_k) \right\Vert_\infty \leq \delta, k = 0, 1, \ldots, N,$
	Boundary Conditions	$\boldsymbol{\phi}(\boldsymbol{X}_0, \boldsymbol{X}_N, t_0, t_f) \leq \delta,$
	Inequality Constraints	$\boldsymbol{C}(\boldsymbol{X}_k, \boldsymbol{U}_k, T_k) \leq 0, k = 0, 1, \ldots, N,$
NLP with FMU	Performance Index	$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \frac{t_f - t_0}{2} \sum_{i=0}^{N} \omega_i L(\boldsymbol{X}_i, \boldsymbol{U}_i, T_i),$
	The State Equations	$\left\Vert \sum_{i=0}^{N} D_{ki}\boldsymbol{X}_i - \frac{t_f - t_0}{2} \dot{\boldsymbol{X}}_{\mathrm{FMU},i} \right\Vert_\infty \leq \delta, k = 0, 1, \ldots, N,$
	Boundary Conditions	$\boldsymbol{\phi}(\boldsymbol{X}_0, \boldsymbol{X}_N, t_0, t_f) \leq \delta,$
	Inequality Constraints	$\boldsymbol{C}(\boldsymbol{X}_k, \boldsymbol{U}_k, T_k) \leq 0, k = 0, 1, \ldots, N,$

Table A2. The OCP constraint conditions of the Manutec R3 Model 1.

Vars \ Num	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
$q(t_0)$	0.00	−1.50	0.00	0.00	0.00	0.00
$q(t_f)$	1.00	−1.95	1.00	0.50	1.00	0.50
$\dot{q}(t_0)$	0.00	0.00	0.00	0.00	0.00	0.00
$\dot{q}(t_f)$	0.00	0.00	0.00	0.00	0.00	0.00
u_{max}	7.50	7.50	7.50	7.50	7.50	7.50
q_{max}	2.97	2.01	2.86	3.63	2.27	3.15
$\dot{q}_{max}$	3.00	1.50	5.20	3.40	4.30	3.70
LC	−126	252	72	−24.8	21.4	−8.6

Table A3. Kinetic parameters.

Vars \ Num		Link 1	Link 2	Link 3	Link 4	Link 5	Link 6
Mass (kg)		1.6895	3.3798	1.4040	0.7111	0.7111	0.1043
Mass Center w.r.t. MDH Coordinates (m)	X	0.0001	−0.2324	−0.1710	0.0001	0.0001	−0.0003
	Y	−0.0005	0.0000	0.0001	−0.0142	−0.0241	0.0046
	Z	−0.0132	−0.2331	0.0226	−0.0081	−0.0081	−0.0535
Inertia Tensor w.r.t. Mass Center (kg·m^2)	$I11$	0.0053	−0.1045	0.0028	0.0033	0.0011	−0.0002
	$I12$	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
	$I13$	0.0000	0.1125	−0.0209	0.0000	0.0000	0.0000
	$I22$	0.0051	−0.1732	0.0803	0.0002	0.0022	−0.0002
	$I23$	0.0000	0.0000	0.0000	0.0001	0.0002	0.0000
	$I33$	0.0018	−0.0544	0.0584	0.0013	0.0011	0.0003

Table A4. The OCP Description.

Performance Index	$J = t_f,$						
The State Equations	$\left\Vert \sum_{i=0}^{N} D_{ki}X_i - \frac{t_f - t_0}{2}\dot{X}_{FMU,i} \right\Vert_\infty \leq \delta, k = 0, 1, \ldots, N,$						
Constraint Conditions	**Vars \ Num**	**Joint 1**	**Joint 2**	**Joint 3**	**Joint 4**	**Joint 5**	**Joint 6**
	$q(t_0(\text{rad}))$	0.00	0.00	0.00	0.00	0.00	0.00
	$q(t_f)(\text{rad})$	1.07	0.26	0.52	0.78	1.57	0.00
	$\dot{q}(t_0)(\text{rad/s})$	0.00					
	$\dot{q}(t_f)(\text{rad/s})$	0.00					
	$u_{max}(\text{A})$	0.21					
	$q_{max}(\text{rad})$	2.67					
	$\dot{q}_{max}(\text{rad/s})$	1.74					
	$LC(\text{N}\cdot\text{m/A})$	101					

References

1. Shin, K.; McKay, N. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Trans. Autom. Control* **1986**, *31*, 491–500. [CrossRef]
2. Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Path Planning and Trajectory Planning Algorithms: A General Overview. In *Motion and Operation Planning of Robotic Systems*; Springer: Cham, Switzerland, 2015; pp. 3–27.
3. Shi, B.; Xu, J. Time-Optimal Trajectory Planning of Industrial Robot based on Improved Particle Swarm Optimization Algorithm. In Proceedings of the 2020 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020; pp. 3683–3688.
4. Li, G.; Wang, Y. Industrial Robot Optimal Time Trajectory Planning Based on Genetic Algorithm. In Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 4–7 August 2019.
5. Wang, H.; Wang, H.; Huang, J.; Zhao, B.; Quan, L. Smooth point-to-point trajectory planning for industrial robots with kinematical constraints based on high-order polynomial curve. *Mech. Mach. Theory* **2009**, *139*, 84–293. [CrossRef]
6. Yu, H.; Meng, Q.; Zhang, J. Time-optimal trajectory planning of robot based on improved adaptive genetic algorithm. In Proceedings of the 2018 Chinese Control And Decision Conference (CCDC), Shenyang, China, 9–11 June 2018.

7. Shi, B.; Zeng, H. Time-Optimal Trajectory Planning for Industrial Robot based on Improved Hybrid-PSO. In Proceedings of the 2021 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021.
8. Barnett, E.; Gosselin, C. A Bisection Algorithm for Time-Optimal Trajectory Planning Along Fully Specified Paths. *IEEE Trans. Robot.* **2021**, *37*, 131–145. [CrossRef]
9. Bobrow, J.E.; Dubowsky, S.; Gibson, J.S. Time-Optimal Control of Robotic Manipulators Along Specified Paths. *Int. J. Robot. Res.* **1985**, *4*, 3–17. [CrossRef]
10. Zhang, T.; Zhang, M.; Zou, Y. Time-optimal and Smooth Trajectory Planning for Robot Manipulators. *Int. J. Control Autom. Syst.* **2020**, *44*, 521–531. [CrossRef]
11. Harzer, J.; Schutter, J.D.; Diehl, M. Efficient Numerical Optimal Control for Highly Oscillatory Systems. *IEEE Control Syst. Lett.* **2022**, *6*, 2719–2724. [CrossRef]
12. Wang, J.; Lu, Z.; Cai, F.; Feng, Y. Fully Discrete Interpolation Coefficients Mixed Finite Element Methods for Semi-Linear Parabolic Optimal Control Problem. *IEEE Access* **2022**, *10*, 2169–3536. [CrossRef]
13. Leomanni, M.; Costante, G.; Ferrante, F. Time-Optimal Control of a Multidimensional Integrator Chain With Applications. *IEEE Control Syst. Lett.* **2022**, *6*, 2371–2376. [CrossRef]
14. Prag, K.; Woolway, M.; Celik, T. Toward Data-Driven Optimal Control: A Systematic Review of the Landscape. *IEEE Access* **2022**, *10*, 32190–32212. [CrossRef]
15. Zhao, J. Data-Driven Adaptive Dynamic Programming for Optimal Control of Continuous-Time Multicontroller Systems With Unknown Dynamics. *IEEE Access* **2022**, *10*, 41503–41511. [CrossRef]
16. Yuan, Z.; Cortes, J. Data-Driven Optimal Control of Bilinear Systems. *IEEE Control Syst. Lett.* **2022**, *6*, 2479–2484. [CrossRef]
17. Manna, S.; Mani, G.; Ghildiyal, S.; Stonier, A.A.; Peter, G.; Ganji, V.; Murugesan, S. Ant Colony Optimization Tuned Closed-Loop Optimal Control Intended for Vehicle Active Suspension System. *IEEE Access* **2022**, *10*, 53735–53745. [CrossRef]
18. Mei, Z.; Chen, L.; Ding, J. Feed-forward control of elastic-joint industrial robot based on hybrid inverse dynamic model. *Adv. Mech. Eng.* **2021**, *13*, 16878140211038102.
19. Schappler, M.; Vorndamme, J.; Todtheide, A.; Conner, D.C.; von Stryk, O.; Haddadin, S. Modeling, Identification and Joint Impedance Control of the Atlas Arms. In Proceedings of the 2015 IEEE-RAS 15th International Conference on Humanoid Robots, Seoul, Korea, 3–5 November 2015; pp. 1052–1059.
20. Richalet, J.; Rault, A.; Testud, J.L.; Papon, J. Model Predictive Heuristic Control: Applications to Industrial Processes. *Automatica* **1977**, *14*, 413–428. [CrossRef]
21. Garcia, C.E.; Prett, D.M.; Morari, M. Model Predictive Control: Theory and Practice–A Survey. *Automatica* **1987**, *25*, 335–348. [CrossRef]
22. Czerwinski, K.; Lawrynczuk, M. Dynamic Matrix Control Algorithm Implementation on ARM Cortex-R5 MCU: Performance Analysis. *IFAC PapersOnLine* **2018**, *51*, 330–335. [CrossRef]
23. Ortiz-Torres, G.; Castillo, P.; Sorcia-Vázquez, F.D.; Rumbo-Morales, J.Y.; Brizuela-Mendoza, J.A.; De La Cruz-Soto, J.; Martínez-García, M. Fault Estimation and Fault Tolerant Control Strategies Applied to VTOL Aerial Vehicles With Soft and Aggressive Actuator Faults. *IEEE Access* **2020**, *8*, 10649–10661. [CrossRef]
24. Sorcia-Vázquez, F.D.J.; Garcia-Beltran, C.D.; Valencia-Palomo, G.; Brizuela-Mendoza, J.A.; Rumbo-Morales, J.Y. Decentralized robust tube-based model predictive control: application to a four-tank system. *Rev. Mex. Ing. Quim.* **2019**, *19*, 1135–1151. [CrossRef]
25. Kahn, M.E.; Roth, B. The near-minimum-time control of open-loop articulated kinematic chains. *Trans. ASME J. Dyn. Syst. Meas. Control* **1971**, *93*, 164–172. [CrossRef]
26. Otter, M.; Türk, S. DFVLR Models 1 and 2 of the Manutec R3 Robot. *DFVLR-Mitteilung* **1988**, *88*, 1–46.
27. Stryk, O.V.; Schlemmer, M. Optimal Control of the Industrial Robot Manutec r3. In *Computational Optimal Control*; Bulirsch, R., Kraft, D., Eds.; Birkhäuser Verlag: Basel, Switzerland, 1994; pp. 367–382.
28. Becerra, V.M. Solving complex optimal control problems at no cost with PSOPT. In Proceedings of the 2010 IEEE International Symposium on Computer-Aided Control System Design, Yokohama, Japan, 8–10 September 2010; pp. 1391–1396.
29. Kelly, M. An introduction to Trajectory Optimization: How to do your own Direct Collocation. *SIAM Rev.* **2016**, *59*, 849–904. [CrossRef]
30. Brandl, H.; Johanni, R.; Otter, M. A Very Efficient Algorithm for the Simulation of Robots and Similar Multibody Systems without Inversion of the Mass Matrix. *IFAC Proc. Vol.* **1986**, *19*, 95–100. [CrossRef]
31. Widl, E.; Müller, W.; Elsheikh, A.; Hörtenhuber, M.; Palensky, P. The FMI++ library: A high-level utility package for FMI for model exchange. In Proceedings of the 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), Berkeley, CA, USA, 20 May 2013.
32. Xu, S.; Li, S.; Cheng, B. Theory and application of Legendre pseudo-spectral method for solving optimal control problem. *Control Decis.* **2014**, *29*, 2113–2120.
33. Fahroo, F.; Ross, I.M. Advances in Pseudospectral Methods for Optimal Control. In Proceedings of the AIAA Guidance, Navigation and Control Conference, Honolulu, HI, USA, 20–23 August 2008.
34. Fahroo, F.; Ross, I.M. Pseudospectral Methods for Infinite-Horizon Nonlinear Optimal Control Problems. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Francisco, CA, USA, 5–9 January 2005.
35. Gong, Q.; Fahroo, F.; Ross, I.M. Spectral Algorithm for Pseudospectral Methods in Optimal Control. *J. Guid. Control Dyn.* **2008**, *31*, 460–471. [CrossRef]

36. Mattsson, S.E.; Elmqvist, H. Modelica—An International Effort to Design the Next Generation Modeling Language. *IFAC Proc. Vol.* **1997**, *30*, 151–155. [CrossRef]
37. Craig, J.J. Manipulator Dynamics. In *Introduction to Robotics: Mechanics and Control*, Horton, M.J., Ed.; Addison-Wesley Longman: Petaluma, CA, USA, 1989; pp. 165–200.
38. Stryk, O.V. Numerical Solution of Optimal Control Problems by Direct Collocation. In *Optimal Control. ISNM International Series of Numerical Mathematics*; Bulirsch, R., Miele, A., Stoer, J., Well, K., Eds.; Birkhäuser: Basel, Switzerland, 1993; Volume 111, pp. 129–143.