Article

# Partial Lagrangian for Efficient Extension and Reconstruction of Multi-DoF Systems and Efficient Analysis Using Automatic Differentiation 

Takashi Kusaka ${ }^{1, *,+(\mathbb{C}}$ and Takayuki Tanaka ${ }^{2,+(\mathbb{D})}$<br>1 Independent Researcher, Sapporo 063-0867, Japan<br>2 Graduate School of Information Science and Technology, Hokkaido University, Sapporo 060-0814, Japan<br>* Correspondence: kusaka@frontier.hokudai.ac.jp<br>$\dagger$ These authors contributed equally to this work.

Citation: Kusaka, T.; Tanaka, T Partial Lagrangian for Efficient Extension and Reconstruction of Multi-DoF Systems and Efficient Analysis Using Automatic Differentiation. Robotics 2022, 11, 149 https://doi.org/10.3390/robotics 11060149

Academic Editor: Raffaele Di Gregorio

Received: 14 October 2022
Accepted: 6 December 2022
Published: 9 December 2022
Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.


Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/).


#### Abstract

In the fields of control engineering and robotics, either the Lagrange or Newton-Euler method is generally used to analyze and design systems using equations of motion. Although the Lagrange method can obtain analytical solutions, it is difficult to handle in multi-degree-of-freedom systems because the computational complexity increases explosively as the number of degrees of freedom increases. Conversely, the Newton-Euler method requires less computation even for multi-degree-of-freedom systems, but it cannot obtain an analytical solution. Therefore, we propose a partial Lagrange method that can handle the Lagrange equation efficiently even for multi-degree-of-freedom systems by using a divide-and-conquer approach. The proposed method can easily handle system extensions and system reconstructions, such as changes to intermediate links, for multi-degree-offreedom serial link manipulators. In addition, the proposed method facilitates the derivation of the equations of motion-by-hand calculations, and when combined with an analysis algorithm using automatic differentiation, it can easily realize motion analysis and control the simulation of multi-degree-of-freedom models. Using multiple pendulums as examples, we confirm the effectiveness of system expansion and system reconstruction with the partial Lagrangians. The derivation of their equations of motion and the results of motion analysis by simulation and motion control experiments are presented. The system extensions and reconstructions proposed herein can be used simultaneously with conventional analytical methods, allowing manual derivations of equations of motion and numerical computer simulations to be performed more efficiently.


Keywords: Lagrange equation; automatic differentiation; divide-and-conquer approach

## 1. Introduction

In dynamics, the Newton-Euler and Lagrange methods are well known for formulating and calculating the equations of motion and used for different purposes [1-4]. The Newton-Euler method is a computational procedure that can be used in multi-degree-offreedom (DoF) models with the systematic algorithm for one DoF at a time [5-7]. Therefore, it is a suitable method for obtaining numerical solutions, but not for analytical solutions. Conversely, the Lagrange method is a formulation procedure that yields analytical solutions that can be understood as physically meaningful terms. However, the amount of computation is enormous when analyzing multi-DoF systems. Therefore, it is generally used for the analysis of systems with fewer DoF.

Numerical solutions that can be computed with the Newton-Euler method are useful for implementing equipment that uses equations of motion, such as real-time feedback control of power assistance in human-machine coordination systems in the field of robotics $[8,9]$. However, when designing algorithms or conducting theoretical analysis of systems, the meaning of each physical term in the analytical solution obtained by the Lagrange method is extremely important. For example, it is used in the behavior analysis
and control of walking robots and drones [10-13]. We use these methods for different applications. In our past research, we used the Lagrange method for the analysis of human body motion modeled in four DoF and the design of control systems [14-16]. We also used the Newton-Euler method to analyze the dynamics of a 7-link (21-DoF) spinal column model for a wearable system [17,18].

A major problem with the Lagrange method is that the amount of computation increases explosively when the number of DoF increases, and the results become complicated. While computer algebra systems can be used to perform large-scale system analysis, manual calculations are extremely difficult to handle even with four DoF, and calculation errors are likely to occur. Because the equations of motion with multi-DoF are extremely complex, many techniques to generate them automatically by computer have been studied. In fact, the equations of motion for the robot arm can be obtained with mathematical processing software, such as Mathematica's Robotica [19] or MatLab's TMTDyn [20], and calculations such as the TMT method are used to optimize the computer calculations [21]. However, the results of large systems collectively are very complex and not reusable.

Therefore, we propose to solve this problem by introducing a partial Lagrangian and postural operator. The partial Lagrangian uses the divide-and-conquer approach [22,23] to divide the equations of motion, which become complex when the number of DoF increases, into the smallest units. The use of the partial Lagrangian reduces the computational load because the terms in the equations of motion for increasing DoF can be treated independently. In other words, a similar process of division can be performed for each DoF, and the results integrated to obtain an exact analytical solution. This has two advantages. First, when the system is extended or reconstructed, only the affected part of the system needs to be calculated, making this analysis method flexible in terms of system configuration. The second advantage is that the modularization of the calculation unit minimizes the burden of obtaining the equations of motion by manual calculation. The modularized calculation is also compatible with computer processing because it involves iterative calculations and the final integration of similar processes.

Studies analyzing multibody dynamics using the divide-and-conquer algorithm (DCA) include the generalized DCA [24], DCAe [25], and a study of sensitivity analysis using DCA [26]. The generalized DCA is an extension method of the DCA for modeling constrained multibody systems, the DCAe is a reconstruction of DCA for efficient handling of multibody dynamics, and the sensitivity analysis using DCA treats the DCA as a critical tool for efficient analysis of multibody dynamics. All of these are based on dividing the generalized force in a binary tree and applying the divide-and-conquer method. Therefore, mechanical constraints are important in all of these studies. In other past studies, such as those mentioned above, no method has focused on Lagrangian dividing. Since the proposed method applies to the Lagrangian (energy), which is abstract, it can be applied to any system that can be described by a Lagrangian that is linearly independent.

Recursive algorithms, such as an algorithm applying the Gibbs-Appel equation [27] and the harmony search algorithm [28], are also known to be very effective for complex systems, such as parallel robots. Our approach differs from recursive algorithms in that it exploits the linear independence of the Lagrangian for partitioning and reuse of computed results. As a benefit of partitioning, we can derive analytical solutions and gain computational efficiencies in computerized numerical solutions. Therefore, the final result of the proposed method is completely equal to the usual Lagrangian method and can be used without the need for a recursive algorithm.

We propose a numerical analysis method based on automatic differentiation as the optimal analysis method for the partial Lagrangian. Automatic differentiation is a method used for training neural networks [29-34] We also confirm that the proposed partial Lagrangian and automatic differentiation can be used to simulate multi-linked manipulators easily and that the system can be easily extended and reconstructed.

## 2. Methods

### 2.1. Partial Lagrangian

This section provides an overview of the partial Lagrangian. Specific examples are given in the next section. The usual Lagrangian requires the total kinetic and potential energy of the system, so the term explodes with increasing DoF. There is a known efficient method of recursive computation using the linear independence of the Lagrangian [35,36]. We extend this idea and consider how to design a more efficient system using the divide-and-conquer approach by organizing it in units of the partial Lagrangian.

First, the Lagrangian $\mathcal{L}$ of the $n$-DoF system is the difference between the total kinetic energy $K$ and the total potential energy $P$, as follows

$$
\begin{equation*}
\mathcal{L}=K-P \tag{1}
\end{equation*}
$$

Here, if we decompose each type of energy into DoF using the distributive property, we can transform it as follows.

$$
\begin{equation*}
\mathcal{L}=\sum_{i=1}^{n} K_{i}-\sum_{i=1}^{n} P_{i}=\sum_{i=1}^{n}\left(K_{i}-P_{i}\right)=\sum_{i=1}^{n} \mathcal{L}_{i} \tag{2}
\end{equation*}
$$

This $\mathcal{L}_{i}$ is defined as a partial Lagrangian. The subscript $i$ denotes the division into partial Lagrangians. An $n$-DOF system will have $n$ partial Lagrangians, corresponding to each DoF as $i=1,2, \cdots, n$. Since the Lagrangian $\mathcal{L}$ is linearly independent, the partial Lagrangian $\mathcal{L}_{i}$ for the $i$-th DoF consists of the partial kinetic energy $K_{i}$ and the partial potential energy $P_{i}$.

Considering the generalized coordinate $q_{k}$, the equation of motion with the Lagrangian is as follows:

$$
\begin{equation*}
\frac{d}{d t}\left(\frac{\partial}{\partial \dot{q}_{k}} \mathcal{L}\right)-\frac{\partial}{\partial q_{k}} \mathcal{L}=\tau_{k} \tag{3}
\end{equation*}
$$

where the subscript $k$ represents the $k$-th equation of motion and $k \leq n$ for the $n$-DoF system. $\tau_{k}$ represents the generalized force of $k$-th DoF since the above equation is the usual Lagrangian equation of motion. For simplicity, let the differential operator on the left-hand side be defined formally as $D_{k}=\frac{d}{d t} \frac{\partial}{\partial \dot{q}_{k}}-\frac{\partial}{\partial q_{k}}$. In other words, the equation of motion for the $k$-th $\operatorname{DoF}$ is $D_{k} \mathcal{L}=\tau_{k}$.

Here, considering the partial Lagrangian $\mathcal{L}_{i}$,

$$
\begin{equation*}
D_{k} \mathcal{L}=D_{k} \sum_{i=1}^{n} \mathcal{L}_{i}=\sum_{i=1}^{n}\left(D_{k} \mathcal{L}_{i}\right)=\sum_{i=1}^{n} \tau_{k i} \tag{4}
\end{equation*}
$$

because the order of the sum and derivative can be exchanged using term-wise differentiation. This $\tau_{k i}$ is defined as the partial generalized force.

Now consider the components of $\mathcal{L}$. If $i \geq k, \mathcal{L}_{k}$ contains $q_{i}$, but if $i<k, \mathcal{L}_{k}$ does not contain $q_{i}$. Therefore, the partial generalized force $D_{k} \mathcal{L}_{i}$ is as follows.

$$
D_{k} \mathcal{L}_{i}=\left\{\begin{array}{cc}
\tau_{k i} & (i \geq k)  \tag{5}\\
0 & (i<k)
\end{array}\right.
$$

This can be summarized as shown in Table 1. Looking at this table, the equation of motion $D_{k} \mathcal{L}=\tau_{k}$ using the Lagrangian method corresponds to calculating all of the entries in row k simultaneously. Therefore, the calculation explodes as the number of DoF increases. However, the partial Lagrangian $D_{k} \mathcal{L}_{i}=\tau_{k i}$ is equivalent to splitting this calculation and performing the calculation with respect to $\mathcal{L}_{i}$. In other words, the analysis is column-wise. Therefore, the final equivalent analytical solution is obtained by summing, but it can be computed by dividing the solution in order, starting from $i=1$.

In addition, the calculation results of $\mathcal{L}_{(i-1)}$ can be diverted for the calculation of $\mathcal{L}_{i}$, thus reducing the amount of the calculation. Furthermore, in the case of the Lagrangian equation $D_{k} \mathcal{L}=\tau_{k}$, if the robot's link is extended or the number of DoF is changed after the analytical solution is obtained, all calculations must be redone. Conversely, when the partial Lagrangian is used, the calculation is independent for each DoF, so the results of the root side calculation can be reused. Therefore, by managing the system in units of partial Lagrangian modules, system extensions, and changes can be handled in a prepared manner.

For example, to extend the system, simply add a new term regarding $\mathcal{L}_{(n+1)}$ as follows, and the summing part can be reused.

$$
\begin{align*}
\text { Current system: } \tau_{k} & =\sum_{i=1}^{n} D_{k} \mathcal{L}_{i}  \tag{6}\\
& \downarrow \\
\text { Extended system: } \tau_{k, \text { new }} & =\underbrace{\sum_{i=1}^{n} D_{k} \mathcal{L}_{i}}_{\text {Reusable term }}+\underbrace{D_{k} \mathcal{L}_{(n+1)}}_{\text {New term by the extension }} \tag{7}
\end{align*}
$$

In addition, system changes can be realized in the same way.

$$
\begin{gather*}
\text { Current system: } \tau_{k}=\sum_{i=1}^{n} D_{k} \mathcal{L}_{i, \text { old }}=\sum_{i=1}^{n-1} D_{k} \mathcal{L}_{i}+D_{k} \mathcal{L}_{n, \text { old }}  \tag{8}\\
\downarrow \\
\text { Changed system: } \tau_{k, \text { new }}=\sum_{i=1}^{n} D_{k} \mathcal{L}_{i, \text { new }}=\underbrace{\sum_{i=1}^{n-1} D_{k} \mathcal{L}_{i}}_{\text {Reusable term }}+\underbrace{D_{k} \mathcal{L}_{n, \text { new }}}_{\text {Replaced term }} \tag{9}
\end{gather*}
$$

Table 1. Partial Lagrangian vs. Lagrange method.

|  | Partial Lagrangian |  |  |  |  |  |  | Lagrange Method |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $\mathcal{L}_{1}$ | $\mathcal{L}_{2}$ | $\mathcal{L}_{3}$ | $\ldots$ | $\mathcal{L}_{i}$ | $\ldots$ | $\mathcal{L}_{n}$ | $\xrightarrow{\Sigma}$ | $\mathcal{L}$ |
| $D_{1}$ | $\tau_{11}$ | $\tau_{12}$ | $\tau_{13}$ | $\cdots$ | $\tau_{1 i}$ | $\cdots$ | $\tau_{1 n}$ | $\rightarrow$ | $\tau_{1}$ |
| $D_{2}$ | 0 | $\tau_{22}$ | $\tau_{23}$ | $\ldots$ | $\tau_{2 i}$ | $\ldots$ | $\tau_{2 n}$ | $\rightarrow$ | $\tau_{2}$ |
| $D_{3}$ | 0 | 0 | $\tau_{33}$ | $\ldots$ | $\tau_{3 i}$ | $\ldots$ | $\tau_{3 n}$ | $\rightarrow$ | $\tau_{3}$ |
| ! | : | : | : | $\ddots$ | $\vdots$ |  | : | ! | $\vdots$ |
| $D_{k}$ | 0 | 0 | 0 | $\ldots$ | $\tau_{k i}$ | $\ldots$ | $\tau_{k n}$ | $\rightarrow$ | $\tau_{k}$ |
| : | $\vdots$ | : | : |  | ; | $\because$ | : | ! | ; |
| $D_{n}$ | 0 | 0 | 0 | $\ldots$ | 0 | $\ldots$ | $\tau_{n n}$ | $\rightarrow$ | $\tau_{n}$ |

In summary, when the partial Lagrangian is generalized to $n$ DoF, the following procedure can be used to calculate the partition.

1. Dynamics: Differentiation of state variables as physical constraints.
2. Kinematics: Position $p_{i}$ and velocity $\dot{p}_{i}$ as geometric constraints.
3. Partial Lagrangian.
(a) Quadratic form: calculate $\boldsymbol{p}_{i}^{T} \boldsymbol{p}_{i}$ and $\dot{\boldsymbol{p}}_{i}^{T} \dot{\boldsymbol{p}}_{i}$ to find the energies. Calculate the partial energies: $K_{i}$ and $P_{i}$.
(b) Compute the partial Lagrangian: $\mathcal{L}_{i}$.
4. Find the partial generalized force: $D_{k} \mathcal{L}_{i}=\tau_{k i}$. If it is a multi-degree-of-freedom system, find the sum $\tau_{k}=\sum_{i=1}^{n} \tau_{k i}$.

This calculation procedure is performed in order from the smallest to the largest value of $i$, and finally, the equation of motion can be derived by computing the sum. Therefore, the complex multi-DoF Lagrangian equations can be obtained with relative ease using a divide-and-conquer approach. Censoring in the middle is equivalent to the equation of motion for a short robot arm, and it is also easy to add $i=n+1$ later.

### 2.2. Postural Operator for Hand Calculations

By using the partial Lagrangian, the Lagrangian is divided and treated independently with respect to each DoF. In this subsection, we define a posture operator for more efficient computations when dealing with partial Lagrangians. Let the attitude operator in the planar coordinates be a vector of length 1 , such that

$$
\boldsymbol{e}_{\theta}=\left[\begin{array}{c}
\cos \theta  \tag{10}\\
\sin \theta
\end{array}\right]
$$

This has the following trivial properties:

- Cancellation: $\boldsymbol{e}_{\theta}^{T} \boldsymbol{e}_{\theta}=1$
- Interference: $\boldsymbol{e}_{\theta_{12}}^{T} \boldsymbol{e}_{\theta_{1}}=\boldsymbol{e}_{12}^{T} \boldsymbol{e}_{1}=\cos \theta_{2}$
- Derivative: $\frac{d}{d t} \boldsymbol{e}_{\theta}=\left\{\begin{array}{cl}\dot{\theta} e_{\theta+\frac{\pi}{2}}=\dot{\theta} \boldsymbol{e}_{\theta}^{\prime} & \text { if } \theta=\theta(t) \\ 0 & \text { if } \theta=\text { const. }\end{array}\right.$
where $\theta_{12}=\theta_{1}+\theta_{2}$ and $\boldsymbol{e}_{\theta_{i}}=\boldsymbol{e}_{i}$ for notational simplicity. The phase difference of $+\frac{\pi}{2}$ is defined as $\boldsymbol{e}_{\theta+\frac{\pi}{2}}=\boldsymbol{e}_{\theta}^{\prime}$.

This is similar to the stationary phasor [37-39] in electrical engineering, which uses a polar form to improve the perspective of the formula expansion. Here, we treat it as a vector rather than a complex form. This also makes it intuitively consistent with the orthogonal form expression expansion. The description of motion in a two-dimensional plane is simplified by using the above properties. In the three-dimensional case, a similar argument can be made using a versor [40-43] with a magnitude of 1 as a rotation by quaternions or using a rotation matrix [44]. We deal with the two-dimensional case because time is used as the third axis for the visualization of the simulation results.

### 2.3. System Analysis Using Automatic Differentiation for Numerical Calculation

Next, we consider the numerical analysis using the partial Lagrangian. The advantage of the partial Lagrangian is that extensions and reconstructions of the system can easily be realized by splitting the Lagrangian. If the equations are transformed to the form of linear differential equations for each DoF for dynamics analysis, this is equivalent to the usual Lagrangian and loses the advantage of the split calculation.

An analytical method that can directly handle $D_{k} \mathcal{L}_{i}=\tau_{i} k$, which is a formal description using the partial Lagrangian, is desirable. Therefore, we use an analysis method that applies automatic differentiation. The solution of the equation of motion by the partial Lagrangian is obtained by generating a computational graph by automatic differentiation and performing a gradient calculation using the balance of the partial differential equation as a constraint. Applications to the analysis of partial differential equations other than neural networks are used in the fields of rigorous simulations, such as the finite element method analysis [45,46], dynamics calculations [47,48], and electronic circuit analysis [49-51].

An example of dynamics computed by automatic differentiation is shown in Appendix A. As indicated in the function in the Appendix, the program for generating a graph of differential equation calculations for automatic differentiation is written in the following flow.

1. Dynamics: The time evolution of the state variable is registered in the calculation graph as a constraint.
2. Kinematics: Register a geometric constraint on a calculation graph.
3. Register a partial Lagrangian in the computed graph
4. Register the partial generalized force, and if it is a multi-DoF, find the sum for each DoF.
5. Backpropagate and find the coefficients as the gradient of the state variable.

This can be described in exactly the same flow as the calculation algorithm for the partial Lagrangian method presented in Section 2.1. Therefore, by using automatic differentiation and calculating dynamics by the partial Lagrangian method, the calculation of the equations of motion can be realized using a divide-and-conquer approach. In addition, each calculation is divided and modularized, allowing the system to be reconstructed instantly.

## 3. Results

We present some examples and simulations to confirm the effect of the partial Lagrangian. For simplicity, we assume an n-link manipulator moving on a plane and describe it using the postural operator. Although the pendulum model is assumed for the simplicity of explanation, rigid body links can be treated in the same way as long as the Lagrangian is a linear sum. The link parameter for link i has a link length of $l_{i}$, mass $m_{i}$, and stiffness $k_{i}=0$. The link length is read as the total length when dealing with connections between links or the length to the center of gravity when dealing with the center of gravity, as appropriate.

### 3.1. Example of 1-DoF: Effect of Postural Operator

In the case of one $\operatorname{DoF}, \tau_{11}=D_{1} \mathcal{L}_{1}=D_{1} \mathcal{L}=\tau_{1}$; thus, the partial Lagrangian result is exactly equal to the usual Lagrangian method. Here, we use the single pendulum [52] as an example of a rotational joint. This calculation is very simple, but to confirm the effect of the postural operator and the flow of the partial Lagrangian processing, we show it in detail as an example.

First, as a geometric constraint, the link end position and its velocity are as follows from the kinematics.

$$
\begin{align*}
& p_{1}=l_{1} e_{1}  \tag{11}\\
& \dot{p}_{1}=l_{1} \frac{d}{d t} e_{1}=l_{1} \dot{\theta}_{1} e_{1}^{\prime} \tag{12}
\end{align*}
$$

These quadratic forms are then obtained as inner products.

$$
\begin{align*}
& \boldsymbol{p}_{1}^{T} \boldsymbol{p}_{1}=l_{1}^{2} \underbrace{}_{\begin{array}{c}
\text { cancellation } \\
\boldsymbol{e}_{1}^{T} \boldsymbol{e}_{1}
\end{array}=l_{1}^{2}}  \tag{13}\\
& \dot{\boldsymbol{p}}_{1}^{T} \dot{\boldsymbol{p}}_{1}=l_{1}^{2} \dot{\theta}_{1}^{2} \underbrace{\boldsymbol{e}_{1}^{\prime T} \boldsymbol{e}_{1}^{\prime}}_{\text {cancellation }}=l_{1}^{2} \dot{\theta}_{1}^{2} \tag{14}
\end{align*}
$$

From these, the kinetic energy $K$ and potential energy $P$ can be calculated as follows.

$$
\begin{align*}
K_{1} & =\frac{1}{2} m_{1} \dot{\boldsymbol{p}}_{1}^{T} \dot{\boldsymbol{p}}_{1}=\frac{1}{2} m_{1} l_{1}^{2} \dot{\theta}_{1}^{2}  \tag{15}\\
P_{1} & =\frac{1}{2} k_{1} \boldsymbol{p}_{1}^{T} \boldsymbol{p}_{1}+m_{1} g y_{1}=m_{1} g \sin \theta_{1} . \tag{16}
\end{align*}
$$

The partial Lagrangian calculated using these is $\mathcal{L}_{1}=K_{1}-P_{1}=\mathcal{L}$. Therefore, its equation of motion can be obtained as $\tau_{11}=D_{1} \mathcal{L}_{1}=D 1 \mathcal{L}=\tau_{1}$. This should be equivalent to the equation of motion for an inverted pendulum in a general textbook.

To confirm this, we analyze the behavior of the equations of motion using automatic differentiation. For one link, the analysis can be performed using the sample of automatic differentiation shown in Appendix A. The link parameters are $m_{1}=3, k_{1}=0, l_{1}=2$, with an appropriate damping term $d_{1}=1$ to make the behavior easier to understand. The results are shown in Figure 1a. Posture control can also be simulated simply by writing a control input to the generalized forces of the equations of motion in the calculation graph. The results of the proportional-derivative (PD) control with a target angle of $110^{\circ}$ are shown in

Figure 1b. The PD gains were set to low values ( $K_{p 1}=200$ and $K_{d 1}=50$, respectively) to make the behavior easy to understand.

The results show how to calculate the partial Lagrangian by hand for the simplest case and analyze it by automatic differentiation. This is the minimum unit of the divide-andconquer approach for the Lagrangian, and from the next section, we will confirm that this procedure can be repeated according to Table 1 to obtain the desired equations of motion.


Figure 1. Motion analysis of the 1 -link system using automatic differentiation. (a) Damping oscillation; (b) PD control.

### 3.2. Example of 2-DoF: Effect of Divide-and-Conquer by Partial Lagrangian

In the previous chapter, we confirmed the reduction in computational complexity due to the postural operator $e_{\theta}$ on a 1-DoF example. Next, we will use a 2-DoF example to confirm this effect. According to Table 1, only $\tau_{12}$ and $\tau_{22}$ need to be added. As a simple example, we consider a double pendulum [53,54], which extends the 1-DoF example.

The same procedure as for the $1-\mathrm{DoF}$ is used to obtain the partial Lagrangian $\mathcal{L}_{2}$.
First, the geometric constraints are as follows.

$$
\begin{align*}
& p_{2}=p_{1}+l_{2} e_{12}=l_{1} \boldsymbol{e}_{1}+\underbrace{l_{2} e_{12}}_{\text {new information }}  \tag{17}\\
& \dot{p}_{2}=\dot{p}_{1}+l_{2} \frac{d}{d t} \boldsymbol{e}_{12}=l_{1} \dot{\theta}_{1} \boldsymbol{e}_{1}^{\prime}+\underbrace{l_{2} \dot{\theta}_{12} e_{12}^{\prime}}_{\text {new information }} \tag{18}
\end{align*}
$$

Then, their quadratic forms are

$$
\begin{align*}
\boldsymbol{p}_{2}^{T} \boldsymbol{p}_{2} & =l_{1}^{2} \underbrace{\boldsymbol{e}_{1}^{T} \boldsymbol{e}_{1}}_{\text {cancellation }}+l_{2}^{2} \underbrace{\boldsymbol{e}_{12}^{T} \boldsymbol{e}_{12}}_{\text {cancellation }}+l_{1} l_{2} \underbrace{e_{1}^{T} \boldsymbol{e}_{12}}_{\text {interference }} \\
& =l_{1}^{2}+l_{2}^{2}+l_{1} l_{2} \cos \theta_{2}  \tag{19}\\
\dot{\boldsymbol{p}}_{2}^{T} \dot{\boldsymbol{p}}_{2} & =l_{1}^{2} \dot{\theta}_{1}^{2} \underbrace{\boldsymbol{e}_{1}^{T} \boldsymbol{e}_{1}^{\prime}}_{\text {cancellation }}+l_{2}^{2_{1}^{2}} \dot{\theta}_{12}^{2} \underbrace{e_{12}^{\prime T} \boldsymbol{e}_{12}^{\prime}}_{\text {cancellation }}+l_{1} l_{2} \dot{\theta}_{1} \dot{\theta}_{12} \underbrace{e_{1}^{\prime T} \boldsymbol{e}_{12}^{\prime}}_{\text {interference }} \\
& =l_{1}^{2} \dot{\theta}_{1}^{2}+l_{2}^{2} \dot{\theta}_{12}^{2}+l_{1} l_{2} \dot{\theta}_{1} \dot{\theta}_{12} \cos \theta_{2} . \tag{20}
\end{align*}
$$

The kinetic and potential energies consisting of the link parameters for $i=2$ are

$$
\begin{align*}
K_{2} & =\frac{1}{2} m_{2} \dot{p}_{2}^{T} \dot{p}_{2}  \tag{21}\\
P_{2} & =\frac{1}{2} k_{2} \boldsymbol{p}_{2}^{T} \boldsymbol{p}_{2}+m_{2} g y_{2} \tag{22}
\end{align*}
$$

Since this partial Lagrangian is $\mathcal{L}_{2}=K_{2}-P_{2}$, the equations of motion are

$$
\begin{array}{lr}
\tau_{1}=\tau_{11}+\tau_{12}= & =\tau_{11}+D_{1} \mathcal{L}_{2} \\
\tau_{2}=\quad \tau_{22}= & D_{2} \mathcal{L}_{2} \tag{24}
\end{array}
$$

also using the result of $\tau_{11}$ with one DoF. In other words, the calculation was reduced by one previous result, $\tau_{11}$, in finding the equations of motion for two DoF. To obtain the equation of motion for $n$ DoF, only the $n$ columns of Table 1 must be calculated, and the calculations from columns 1 to $n-1$ are unnecessary because they can be reused. References [53,54] show that the equation of motion of a double pendulum is complicated to be solved during the process if it is obtained by the usual Lagrangian method; however, it can be described in a simple manner by the partial Lagrangian method. This simplifies the calculation even for an increased DoF and prevents calculation mistakes in manual calculations.

We confirm that this result is correct by simulation with automatic differentiation. The simulation can take full advantage of the divide-and-conquer effect of the partial Lagrangian. In the example program shown in Appendix A (Listing A1), to obtain $\mathcal{L}_{2}$, only two lines of the geometric constraint need to be rewritten according to the system. Therefore, the idea of division of the process in the partial Lagrangian corresponds to the program, and the same calculation results obtained using the complicated equations of motion can also be obtained as an iteration of this module.

The results of the 2-DoF example are shown in Figure 2a as damped oscillations and that of adding the PD control to the joints are shown in Figure 2b. The link parameters are $m_{1}=3, k_{1}=0, l_{1}=2, d_{1}=1$ for the first joint and $m_{2}=1, k_{2}=0, l_{2}=1, d_{2}=1$ for the second. Target values of $\theta_{1 \text { ref }}=80^{\circ}$ and $\theta_{2 \text { ref }}=30^{\circ}$ were used for PD control, and its gains are $K_{p 1}=100, K_{p 2}=50, K_{d 1}=50$, and $K_{d 2}=20$. It can be shown that the desired behavior can be analyzed even in the case of multi-DoF.

(a)

(b)

Figure 2. Motion analysis of a two-link system using automatic differentiation. (a) Damping oscillation; (b) PD control.

### 3.3. Example of 3-DoF and Changing System Construction

First, we show an example of a triple pendulum [55,56] that extends the 2-DOF example. Then, we look at the system reconstruction with partial Lagrangian when the second link is changed from a rotational joint to a linear motion joint. The calculation of $\mathcal{L}_{3}$ is an iteration of Equations (8)-(17) and is abbreviated here. The required equations of motion are as follows.

$$
\begin{array}{lrr}
\tau_{1}= & \tau_{11}+\tau_{12}+\tau_{13} & =\tau_{11}+\tau_{12}+D_{1} \mathcal{L}_{3} \\
\tau_{2}= & \tau_{22}+\tau_{23} & \tau_{22}+D_{2} \mathcal{L}_{3} \\
\tau_{3}= & \tau_{33} & D_{3} \mathcal{L}_{3} \tag{27}
\end{array}
$$

Since the only new term that needs to be calculated is the term related to $\mathcal{L}_{3}$, we can extend the link easily even by hand calculation, by using the postural operator as in the case up to the 2-DoF example. References $[55,56]$ show that the Lagrangian becomes very complex with three DoF; thus, this approach has the advantage of performing the partitioning calculation with the partial Lagrangian.

This behavior is confirmed by simulation with automatic differentiation. The results of the damped oscillation are shown in Figure 3a, and the results with the PD control are shown in Figure 3b. The target angles for PD control are $\theta_{1 r e f}=125^{\circ}, \theta_{2 \text { ref }}=-30^{\circ}$, and $\theta_{3 \text { ref }}=-30^{\circ}$, and its gains are $K_{p 1}=1000, K_{p 2}=1000, K_{p 3}=500, K_{d 1}=200, K_{d 2}=200$, and $K_{d 3}=100$. As with the 2-DoF case, this simulation can be handled simply by modifying two lines of the geometric constraint for $\mathcal{L}_{3}$; thus, the implementation cost is very small.

(a)

(b)

Figure 3. Motion analysis of the 3-link system using automatic differentiation. (a) Damping oscillation; (b) PD control.

Next, as an example of system reconstruction, a simulation of a system in which the second link is changed to a linear joint is shown in Figure 4a,b. The second link is controlled to shorten its length by the PID control, and the other joints are the damped vibrations. The initial length of the second link is $l_{2}=1$ and its target length is $l_{2 \text { ref }}=0.1$. The results show that the desired behavior can be achieved even when the intermediate link is changed as an example of the system reconstruction. As mentioned above, in the automatic differentiation program, the simulation can be realized merely by rewriting the time-varying parameters of the geometric constraint.

(a)

(b)

Figure 4. System reconstruction example (link 2 is changed from rotational joint to linear slider). (a) Side view; (b) Front view.

## 4. Discussion

In this section, we discuss the advantages of the proposed partial Lagrangian. As mentioned above, the purpose of the usual Lagrangian is to derive an analytical solution; thus, its application to multiple degrees of freedom is difficult. However, by dividing it into partial Lagrangians, it can be handled relatively easily even with multiple degrees of freedom, and can be used to compute numerical solutions. In this section, we discuss computational complexity and scalability by considering extensions up to 10 links, as shown in Figure 5.


Figure 5. Experimental results of a large-scale model using the partial Lagrangian . (a) Damping oscillation by the 10-DoF pendulum; (b) computational cost of automatic differentiation (all simulations were performed by Intel Core i5-10400 CPU @2.90 GHz).

### 4.1. Computational Advantages of Partial Lagrangian

The usual Lagrangian is generally difficult to use with the multi-DoF system because the number of terms explodes with increasing degrees of freedom. The partial Lagrangian uses the divide-and-conquer approach to divide the computation of the Lagrange equation into its smallest module, $D_{k} \mathcal{L}_{i}=\tau_{i k}$. This makes it relatively easy to obtain an exact analytical solution eventually by repeating the same simple procedure, even if the number of DoF is increased.

We consider the advantages of the partial Lagrangian from two perspectives: the derivation of the equations of motion by hand calculations and computer simulations.

First, for manual calculations, the introduction of the postural operator together with the partial Lagrangian simplifies the calculation with respect to rotational joints and significantly reduces the number of calculations compared to those that would otherwise be required. In addition, divide-and-conquer by the partial Lagrangian prevents calculation errors due to manual calculations because the number of calculations is small for each module.

Second, for computer simulations, the gradient calculation of the calculation graph by automatic differentiation was used as an efficient method to process the formal description of the divide-and-conquer approach by the partial Lagrangian, $D_{k} \mathcal{L}_{i}=\tau_{i k}$. Automatic differentiation is a different method of analysis from mathematical and numerical differentiation. Automatic differentiation can solve the inefficiency of mathematical differentiation and the accuracy problem of numerical differentiation. It is compatible with partial Lagrangians because it can describe partial differential equations directly and perform the calculation of the balance. The partial Lagrangian allows for efficient simulation by iterating the modularized process through automatic differentiation and integrating it at the necessary stages.

As an example of this, a simulation in which the model is extended to 10 links is shown in Figure 5a. Here, we consider Figure 5b as an efficiency improvement by dividing it into partial Lagrangians. By dividing, computers can compute independent components
using multi-threading. The order of computational amount of $n$-DoF without dividing can be estimated as the number of terms as follows. The $\mathcal{O}$ means Landau symbol.


The fact that the computational cost is $\mathcal{O}\left(n^{4}\right)$ in the analysis of general dynamics is stated in the reference [26] and supports this result.

On the contrary, using partial Lagrangian, the generation of the computational graph and the gradient calculation (i.e., torque calculation) with multi-threading is shown below.

$$
\underbrace{n}_{\begin{array}{c}
\text { Max number of dividing }  \tag{29}\\
\text { into partial Lagrangian }
\end{array}} \times \underbrace{\left(n^{2}+2 n\right)}_{\text {Number of terms in } \tau_{k i}}=\mathcal{O}\left(n^{3}\right)
$$

As shown above, the division into partial Lagrangians allows the linearly independent parts to be computed in parallel, lowering the order of computation from $\mathcal{O}\left(n^{4}\right)$ to $\mathcal{O}\left(n^{3}\right)$. These computational amounts are consistent with the results of the actual simulation shown in Figure 5b.

### 4.2. Application of Partial Lagrangian as an Extension or Restructuring of the System

Another advantage of the partial Lagrangian is that it can handle system extensions and reconstructions in units of divided modules. This is due to the linear independence of each column of Table 1. The pendulum examples shown in Section 3 show the flow of the system extension from one to three DoF. The system extended to $10-\mathrm{DoF}$ to evaluate the computational amount in the previous section is an extension of the 3DoF model with additional link parameters, all computed with the same partial Lagrangian module.

With the partial Lagrangian, the calculations required for energy and partial derivatives are about half of those of the usual Lagrangian. This makes it easier to handle the multi-DoF system than the usual Lagrangian, but the difficulty increases with the DoF number. Moreover, if the middle link is changed, it is necessary to go back that far in Table 1. However, in the case of simulation with automatic differentiation, energy calculation and partial differentiation are performed automatically, so system reconstruction, such as the intermediate link changes, can be performed merely by rewriting the geometric constraint part. Therefore, similar to the conclusion of the computational advantage, the divide-and-conquer approach using the partial Lagrangian and module-by-module processing using an automatic differentiation algorithm allows for the easy dynamics analysis of complicated systems and their configuration changes with only a formal description of the partial Lagrangian.

## 5. Conclusions

We proposed the partial Lagrangian to handle the Lagrange equation efficiently using a divide-and-conquer approach. The partial Lagrangian makes it possible to handle extensions and reconstructions of the system easily to obtain analytical solutions, which was not previously possible. In addition, the introduction of the postural operator together with the partial Lagrangian facilitates the derivation of the equations of motion by hand calculation. The division of computations by the partial Lagrangian and reduction of computational complexity by the postural operator reduce the computational cost of manual calculations of energy and partial derivatives, even with multi-DoF.

Furthermore, we proposed a numerical method of computing the partial Lagrangian using automatic differentiation for simulation. Since automatic differentiation calculates the energy and partial derivatives, it is compatible with the partial Lagrangian formal description, and the system designer needs only to describe the geometric constraints to analyze the motion using the partial Lagrangian's divide-and-conquer approach.

We confirmed that the division into partial Lagrangians allows us to take advantage of multi-threading, reducing the computational complexity that normally requires $\mathcal{O}\left(n^{4}\right)$ to $\mathcal{O}\left(n^{3}\right)$. Even when extended to the multi-DoF system, the numerical solution can be obtained efficiently by repeatedly calling the partial Lagrangian module. As an example of a multi-DoF, we confirmed its effectiveness with a 10-DoF pendulum model.

By replacing or combining the proposed method with conventional systems, the system extensions and simulations can be realized more efficiently than previously possible in the field of design and control using analytical solutions of the conventional equations of motion. However, because the partial Lagrangian uses the linear independence of each DoF to split its computation, the split may not work well if its assumptions change. Several recent studies have shown that recursive algorithmic solutions are also effective in parallellinked systems $[27,28]$. Therefore, our future work will include applications to systems other than the serial link manipulators treated in this study, such as closed-link systems and parallel-link systems.

Author Contributions: Conceptualization, T.K.; investigation, T.K.; methodology, T.K.; supervision, T.T.; validation, T.K. and T.T.; writing-original draft, T.K.; writing-review and editing, T.T. All authors have read and agreed to the published version of the manuscript.
Funding: This research received no external funding
Institutional Review Board Statement: Not applicable.
Informed Consent Statement: Not applicable.
Data Availability Statement: Not applicable.
Conflicts of Interest: The authors declare no conflict of interest.

## Appendix A. A Sample Analysis of Partial Lagrangian Using Automatic Differentiation

A sample analysis of an edited differential equation with automatic differentiation is shown below. We begin with a sample program for the case of one degree of freedom, implemented using PyTorch [57]. In item 0, we first define the constraints for each state variable and time evolution on the computational graph. In item 1, define the geometric constraints for each system. This is the only part that needs to be rewritten when the system changes. In item 2, the partial Lagrangian is defined on the computational graph. The equations of motion with friction are used here to make the behavior easier to understand in the sample graph. In item 3, the constraints of the equations of motion are defined on the computational graph as edited differential equations using partial Lagrangians. Item 4 performs the backpropagation of the computed graph. In item 5, finally, the dynamicsaware inertia term $I$ is obtained as a coefficient of $\ddot{\theta}$ : acc, and the state variables updated by backpropagation are stored as internal variables.

As a precaution, since the Lagrangian is an edited differential equation, the Lagrangian must be registered in the computational graph as functions of $\theta, \omega$, and $t$. Therefore, the time evolution constraint is written in the program so that the variable theta is a function of omega and dt .

The computational graph generated by this sample is shown in Figure A1. In practice, programmers can analyze partial differential equations automatically without dealing directly with this complex graph, but only by writing algorithms similar to the sample code.


Figure A1. Example of a generated calculation graph of a partial Lagrangian (reading downward: forward calculations as inverse dynamics; reading upward: backward calculations as dynamics).

Listing A1. Example of the partial Lagrangian module.

```
# O. State variables and their constraints of time evolution for AD
(_theta,_omega,_acc) = state
acc = torch.tensor(_acc, requires_grad=True)
omega = torch.tensor(_omega, requires_grad=True) + acc * self.dt
theta = torch.tensor(_theta, requires_grad=True) + omega *self.dt
# 1. Geometric constraints and their derivatives
x = self.l * torch.cos(theta)
y = self.l * torch.sin(theta)
dx = torch.autograd.grad(x, self.dt, create_graph=True)
dy = torch.autograd.grad(y, self.dt, create_graph=True)
# 2. (Partial) Lagrangian with friction
K = 0.5*self.m * (dx[0]**2+dy[0]**2)
P}=0.5*\mathrm{ self.k * (x**2+y**2) + self.m*9.8*y
R = 0.5*self.c * (dx[0]**2+dy[0]**2)
L}=\textrm{K}-\textrm{P
# 3. Lagrange Equation of Motion
dLdth = torch.autograd.grad(L, theta, create_graph=True)
dLdomega = torch.autograd.grad(L, omega, create_graph=True)
dLdt = torch.autograd.grad(dLdomega, self.dt, create_graph=True)
dRddth = torch.autograd.grad(R, omega, create_graph=True)
Iacc = dLdth[0] - dRddth[0]
T = dLdt[0] - Iacc # balance
# 4. Backpropagation
T.backward(retain_graph=True)
# 5. Convert force to acc
I = acc.grad
acc = float(Iacc/I)
omega += acc * self.dt
state = (theta, omega, acc)
return state
```


## References

1. de Jesus Rubio, J.; Humberto Perez Cruz, J.; Zamudio, Z.; Salinas, A.J. Comparison of two quadrotor dynamic models. IEEE Lat. Am. Trans. 2014, 12, 531-537.
2. Ali, S. Newton-Euler Approach for Bio-Robotics Locomotion Dynamics: From Discrete to Continuous Systems. Ph.D. Thesis, Ecole des Mines de Nantes, Nantes, France, 2011; p. 251.
3. Sciavicco, L.; Siciliano, B.; Luigi, V. Lagrange and Newton-Euler dynamic modeling of a gear-driven robot manipulator with inclusion of motor inertia effects. Adv. Robot. 1995, 10, 317-334. [CrossRef]
4. Robotic Systems Lab, ETH Zurich Robot Dynamics Lecture Notes. 2017. Available online: https://ethz.ch/content/dam/ ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2017/RD_HS2017script.pdf (accessed on 20 September 2022).
5. De Luca, A.; Ferrajoli, L. A modified newton-euler method for dynamic computations in robot fault detection and control. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12-17 May 2009; pp. 3359-3364. ISSN 1050-4729. [CrossRef]
6. Buondonno, G.; De Luca, A. A recursive Newton-Euler algorithm for robots with elastic joints and its application to control. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September-2 October 2015; pp. 5526-5532. [CrossRef]
7. Li, X.; Nishiguchi, J.; Minami, M.; Matsuno, T.; Yanou, A. Iterative calculation method for constraint motion by extended newton-euler method and application for forward dynamics. In Proceedings of the 2015 IEEE/SICE International Symposium on System Integration (SII), Nagoya, Japan, 11-13 December 2015; pp. 313-319. [CrossRef]
8. Hirata, Y.; Iwano, T.; Tajika, M.; Kosuge, K. Motion Control of Wearable Walking Support System with Accelerometer Based on Human Model; Advances in Human-Robot Interaction; IntechOpen: London, UK, 2009.
9. Benaddy, A.; Labbadi, M.; Bouzi, M. Adaptive Nonlinear Controller for the Trajectory Tracking of the Quadrotor with Uncertainties. In Proceedings of the 2020 2nd Global Power, Energy and Communication Conference (GPECOM), Izmir, Turkey, 20-23 October 2020; pp. 137-142. [CrossRef]
10. Rameez, M.; Khan, L.A. Modeling and dynamic analysis of the biped robot. In Proceedings of the 2015 15th International Conference on Control, Automation and Systems (ICCAS), Busan, South Korea, 13-16 October 2015; pp. 1149-1153. ISSN 20937121. [CrossRef]
11. Nicotra, M.M.; Garone, E. Control of Euler-Lagrange systems subject to constraints: An Explicit Reference Governor approach. In Proceedings of the 2015 54th IEEE Conference on Decision and Control (CDC), Osaka, Japan, 15-18 December 2015; pp. 1154-1159. [CrossRef]
12. Su, B.; Gong, Y. Euler-Lagrangian modeling and exact trajectory following controlling of Ballbot-like robot. In Proceedings of the 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, Macao, 5-8 December 2017; pp. 2325-2330. [CrossRef]
13. Al-Shuka, H.F.N.; Corves, B.; Zhu, W.H. Dynamics of Biped Robots during a Complete Gait Cycle: Euler-Lagrange vs. Newton-Euler Formulations; Research Report; School of Control Science and Engineering, Shandong University: Jinan, China, 2019. Available online: https:/ /hal.archives-ouvertes.fr/hal-01926090 (accessed on 19 September 2022).
14. Kusaka, T.; Tanaka, T.; Kaneko, S.; Suzuki, Y.; Saito, M.; Kajiwara, H. Assist Force Control of Smart Suit for Horse Trainers Considering Motion Synchronization. Int. J. Autom. Technol. 2009, 3, 723-730.
15. Kusaka, T.; Tanaka, T.; Kaneko, S.; Suzuki, Y.; Saito, M.; Kajiwara, H. Smart suit for horse trainers-power and skill assist based on semi-active assist and energy control. In Proceedings of the 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Montreal, QC, Canada, 6-9 July 2010; pp. 509-514. ISSN 2159-6255. [CrossRef]
16. Kusaka, T.; Tanaka, T.; Kaneko, S.; Suzuki, Y.; Saito, M.; Seki, S.; Sakamoto, N.; Kajiwara, H. Assist force control of Smart Suit for horse trainer considering motion synchronization and postural stabilization. In Proceedings of the 2009 ICCAS-SICE, Fukuoka, Japan, 18-21 August 2009; pp. 770-775.
17. Tsuchiya, Y.; Imamura, Y.; Tanaka, T.; Kusaka, T. Estimating Lumbar Load During Motion with an Unknown External Load Based on Back Muscle Activity Measured with a Muscle Stiffness Sensor. J. Robot. Mechatron. 2018, 30, 696-705. [CrossRef]
18. Tsuchiya, Y.; Kusaka, T.; Tanaka, T.; Matsuo, Y. Wearable Sensor System for Lumbosacral Load Estimation by Considering the Effect of External Load. In Proceedings of the Advances in Human Factors in Wearable Technologies and Game Design; Advances in Intelligent Systems and Computing; Ahram, T.; Falcão, C., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 160-168. ._16. [CrossRef]
19. Nethery, J.; Spong, M. Robotica: A Mathematica package for robot analysis. IEEE Robot. Autom. Mag. 1994, 1, 13-20.
20. Sadati, S.H.; Naghibi, S.E.; Shiva, A.; Michael, B.; Renson, L.; Howard, M.; Rucker, C.D.; Althoefer, K.; Nanayakkara, T.; Zschaler, S.; et al. TMTDyn: A Matlab package for modeling and control of hybrid rigid-continuum robots based on discretized lumped systems and reduced-order models. Int. J. Robot. Res. 2021, 40, 296-347.
21. Sadati, S.M.H.; Naghibi, S.E.; Naraghi, M. An Automatic Algorithm to Derive Linear Vector Form of Lagrangian Equation of Motion with Collision and Constraint. Procedia Comput. Sci. 2015, 76, 217-222. [CrossRef]
22. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. Introduction to Algorithms, 3rd ed.; The MIT Press: Cambridge, MA, USA; London, UK, 2009.
23. Knuth, D.E. The Art of Computer Programming; Addison-Wesley Pub. Co.: Reading, MA, USA, 1968.
24. Poursina, M.; Anderson, K.S. An extended divide-and-conquer algorithm for a generalized class of multibody constraints. Multibody Syst. Dyn. 2013, 29, 235-254. [CrossRef]
25. Critchley, J.; Binani, A.; Anderson, K. Design and Implementation of an Efficient Multibody Divide and Conquer Algorithm. In Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Las Vegas, NV, USA, 4-7 September 2007. [CrossRef]
26. Mukherjee, R.M.; Bhalerao, K.D.; Anderson, K.S. A divide-and-conquer direct differentiation approach for multibody system sensitivity analysis. Struct. Multidiscip. Optim. 2008, 35, 413-429. [CrossRef]
27. Zahedi, A.; Shafei, A.M.; Shamsi, M. On the dynamics of multi-closed-chain robotic mechanisms. Int. J. Non-Linear Mech. 2022, 147, 104241. [CrossRef]
28. Mazare, M.; Taghizadeh, M. Geometric Optimization of a Delta Type Parallel Robot Using Harmony Search Algorithm. Robotica 2019, 37, 1494-1512.
29. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic Differentiation in Machine Learning: A Survey. J. Mach. Learn. Res. 2018, 18, 43.
30. Neidinger, R.D. Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming. SIAM Rev. 2010, 52, 545-563. [CrossRef]
31. Naumann, U. The Art of Differentiating Computer Programs; Software, Environments, and Tools, Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2011. [CrossRef]
32. Harrison, D. A Brief Introduction to Automatic Differentiation for Machine Learning. arXiv 2021, arXiv:2110.06209.
33. Al Seyab, R.; Cao, Y. Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation. J. Process Control 2008, 18, 568-581. [CrossRef]
34. Griewank, A.; Walther, A. Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, 2nd ed.; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2008. [CrossRef]
35. Hollerbach, J.M. A Recursive Lagrangian Formulation of Maniputator Dynamics and a Comparative Study of Dynamics Formulation Complexity. IEEE Trans. Syst. Man Cybern. 1980, 10, 730-736.
36. Luh, J.Y.; Walker, M.W.; Paul, R.P. On-line computational scheme for mechanical manipulators. Trans. ASME J. Dyn. Syst. Meas. Control 1980, 102, 69-76. [CrossRef]
37. Fox, H.; Bolton, W. Mathematics for Engineers and Technologists; Butterworth-Heinemann: Oxford, UK, 2002.
38. Rawlins, J.C.; Fulton, S.R.B.A.C. Basic AC Circuits; Newnes: Boston, MA, USA, 2000.
39. Robbins, A.; Miller, W.C. Circuit Analysis: Theory and Practice; DELMAR Cengage Learning: Clifton Park, NY, USA, 2013.
40. McCarthy, J.M. An Introduction to Theoretical Kinematics; MIT Press: Cambridge, MA, USA, 1990.
41. Vicci, L. Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation; Technical Report; UNC: Chapel Hill, NC, USA, 2001.
42. Shoemake, K. Quaternions. 1994. Available online: https:/ /web.archive.org/web/20200503045740/http:/ /www.cs.ucr.edu/ $\sim \mathrm{vbz} /$ resources/quatut.pdf (accessed on 11 September 2022).
43. Dam, E.B.; Koch, M.; Lillholm, M. Quaternions, Interpolation and Animation; Technical Report; Datalogisk Institut, Kobenhavns Universitet: Copenhagen, Denmark, 1998.
44. Bar-Itzhack, I. New Method for Extracting the Quaternion from a Rotation Matrix. J. Guid. Control. Dyn. 2000, 23, 1085-1087. [CrossRef]
45. Ma, F.; Liu, W.; Ma, T. Automatic differentiation application on stochastic finite element. In Proceedings of the 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering, Changchun, China, $24-26$ August 2010; Volume 2, pp. 219-221. ISSN 2159-6034. [CrossRef]
46. Jeßberger, J.; Marquardt, J.E.; Heim, L.; Mangold, J.; Bukreev, F.; Krause, M.J. Optimization of a Micromixer with Automatic Differentiation. Fluids 2022, 7, 144.
47. Enciu, P.; Gerbaud, L.; Wurtz, F. Automatic Differentiation Applied for Optimization of Dynamical Systems. IEEE Trans. Magn. 2010, 46, 2943-2946.
48. Kaheman, K.; Brunton, S.L.; Kutz, J.N. Automatic Differentiation to Simultaneously Identify Nonlinear Dynamics and Extract Noise Probability Distributions from Data. arXiv 2020, arXiv:2009.08810.
49. Feldmann, P.; Melville, R.; Moinian, S. Automatic differentiation in circuit simulation and device modeling. In Proceedings of the 1992 IEEE/ACM International Conference on Computer-Aided Design, Santa Clara, CA, USA, 8-12 November 1992; pp. 248-253. [CrossRef]
50. Toshiji, K. Generalization of Circuit Simulator by Automatic Differentiation. IEEJ Trans. Electron. Inf. Syst. 2004, 124, 404-410. [CrossRef]
51. Christoffersen, C. Implementation Of Exact Sensitivities in a Circuit Simulator Using Automatic Differentiation. In Proceedings of the 20th European Conference on Modelling and Simulation, Bonn, Germany, 28-31 May 2006. [CrossRef]
52. Ueding, M. Lagrange Examples, 2013. Available online: https:/ / martin-ueding.de/posts/lagrange-examples / (accessed on 19 September 2022).
53. Assencio, D. The Double Pendulum: Lagrangian Formulation-Diego Assencio. Available online: https://diego.assencio.com/ ?index=1500c66ae7ab27bb0106467c68feebc6 (accessed on 19 September 2022).
54. Weisstein, E.W. Double Pendulum—From Eric Weisstein's World of Physics; Wolfram Research, Inc.: Champaign, IL, USA, 2018. Available online: https:/ / scienceworld.wolfram.com/physics/DoublePendulum.html (accessed on 19 September 2022).
55. Harman, Nick N. Motion of a Triple Rod Pendulum. Available online: https:/ / www.authorea.com/users/259349/articles /4124 91-motion-of-a-triple-rod-pendulum (accessed on 19 September 2022).
56. Jake VanderPlas, Triple Pendulum CHAOS! \| Pythonic Perambulations. Available online: https://jakevdp.github.io/blog/2017 /03/08/triple-pendulum-chaos/ (accessed on 19 September 2022).
57. Automatic Differentiation Package-torch.autograd-PyTorch 1.12 Documentation. Available online: https:/ / pytorch.org/docs/ stable/autograd.html (accessed on 21 September 2022).
