Improved Computational Speed of System Dynamics for Cable-Driven Robots through Generalised Model Compilation

Yin Pok Chan[1], Ghasem Abbasnejad[1], Jonathan Eden[2] and Darwin Lau[1]

***Abstract*— The study of cable-driven parallel robots (*CDPRs*) has motivated numerous applications in recent years. Using a numerical CDPR modelling approach, the Cable-robot Analysis and Simulation Platform for Research (CASPR) allows simulations and analyses to be performed on arbitrary types of CDPRs. However, the inadequate computational speed of system dynamics hinders the use of CASPR in fast analyses and real-time control on CDPRs. In this paper, a generalised model compilation method is presented to improve the computational speed of system dynamics. By generating symbolic expressions of the system dynamics and compiling the variables into closed-form functions, improvements from 11 to 23 times in computational speed are achieved. Case studies in forward kinematics, inverse dynamics and workspace analysis are performed to demonstrate the benefits of the compilation, and the potential for extending the method to other robotic systems.**

I. INTRODUCTION

As a type of parallel robot that is actuated by cables rather than rigid links, cable-driven parallel robots (*CDPRs*) have gained much interest, due to their high payload-to-weight ratio [1], large maximum workspace [2], and bio-inspired nature [3]. These advantages induce a vast number of applications, from anthropomorphic musculoskeletal robots [4], exoskeletons [5] to high-speed manipulators [6], large-scale construction robots [7] and radio telescopes [8].

The key characteristic and challenge in the use of CDPRs is the constraint that cables can only pull and not push (*positive cable force*). This results in a range of unique challenges in the modelling and analysis of CDPRs. To consider these challenges, CDPR specific models [9] and analysis algorithms [10], [11] have been developed.

In existing CDPR studies, two different paradigms have been developed for the evaluation of system kinematics and dynamics: closed-form analytical solutions and numerical-based computation of the equations of motion (*EoM*). For the closed-form approach, the system model is typically derived each time for the specific robot. Although the use of a closed-form representation results in fast model dynamics computation, the approach is highly model specific. As a result, for different CDPR models, such as for different joints, number of links and cable-routing, the EoM must be re-derived. This process is time consuming and prone to introducing errors into the model and subsequent analyses.

[1]Yin Pok Chan, Ghasem Abbasnejad and Darwin Lau are with Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong {ypchan, darwinlau}@mae.cuhk.edu.hk, ghasem.abbasnejad@gmail.com

[2]Jonathan Eden is with the Melbourne School of Engineering, the University of Melbourne, Melbourne, Australia jpeden@student.unimelb.edu.au

In contrast, the numerical approach models the CDPR system dynamics by constructing an EoM recursively through numerical substitutions. In [12], a generalised kinematics and dynamics model for CDPRs was presented, which allowed arbitrary types of cable-driven robots to be modelled and analysed with ease. However, the convenience brought by such generalised model comes at the cost of higher computational time required to compute the system dynamics (*update of dynamics*) compared with the closed-form methods.

Motivated by the lack of a comprehensive software platform for the analysis of arbitrary types of CDPRs, the numerical model was used to develop the Cable-robot Analysis and Simulation Platform for Research (*CASPR*) [13], an open-source platform developed in MATLAB with a modular object-oriented programming (*OOP*) paradigm. The platform is capable of performing simulations and a wide range of analyses on arbitrary types of CDPRs, such as forward and inverse dynamics, forward and inverse kinematics, workspace analysis and control. The generality speeds up the development process by making the model derivation process automated and by abstracting the analyses. This facilitates the benchmarking of different CDPR algorithms and allows comparison to be made for different robot configurations.

However, one bottleneck of the numerical approach implemented in CASPR is the higher costs in the update of dynamics, due to the lack of closed-form solutions as well as overheads in the OOP structure. This is significant for applications requiring faster computational time, as CASPR may not be capable of meeting the speed requirements depending on the computing machine used. For example, in the control of CDPRs, a faster computation of system dynamics is beneficial, as it allows a higher flexibility in the complexity and type of control algorithm that can be tested. Moreover, the time needed to run procedures with repetitive updates of system dynamics such as workspace analysis, can be shortened with faster computational speed.

To achieve faster computational speed, while preserving the advantages of the generalised approach, this paper proposes a new model compilation strategy that significantly speeds up system dynamics computation while preserving the generalised model formulation. First, using the generalised model, a set of variables with symbolic expressions is generated within MATLAB. Subsequently, the expressions are simplified and compiled into closed-form functions that can be used within CASPR when computing the system dynamics. Simulation results on a range of different CDPRs show a significant improvement in the computational speed compared with the existing generalised dynamics computations, from 11 to 23 times.

As a new mode to evaluate the dynamics within CASPR, this approach preserves the advantages of the numerical approach such that the entire family of CDPRs can be studied, while achieving the computational speeds comparable with the manually derived closed-form approach. The example case study shows that when using the new mode within the analysis of CDPRs, such as forward kinematics, inverse dynamics and workspace analysis, an improvement of 73%, 51% and 88%[1] were obtained, respectively. This improvement shows the significance in using the proposed method when performing analysis. Regarding the long compilation time needed beforehand, strategies are proposed to shorten the compilation time, to a maximum of 96%. Furthermore, this work represents a case-study on the potential to using the same approach for other types of robotic systems, such as mobile manipulators and robots of higher dimensions.

The remainder of the paper is organised as follows. Section II introduces the CDPR system dynamics and the existing CASPR methodology for system dynamics computation. Section III describes the proposed strategy and Section IV illustrates the improvement of computational speed with simulation results. Finally, Section V summarises the paper and its contributions.

II. UPDATE OF SYSTEM DYNAMICS IN CASPR

This section reviews the modelling of CDPRs and the existing CASPR methodology for calculating system dynamics.

A. System Dynamics of CDPRs

For a n-DoF (degrees-of-freedom) robot with m cables and p links, the system dynamics is given by the equation of motion (*EoM*)

$$M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}) + \mathbf{G}(\mathbf{q}) + \mathbf{w}_e = -L(\mathbf{q})^T \mathbf{f}, \qquad (1)$$

where $\mathbf{q} \in \mathbb{R}^n$ is the generalised coordinates (*pose*).

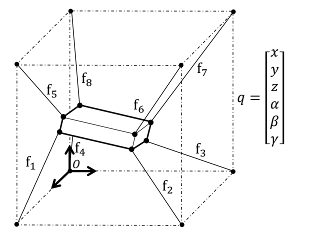
The LHS of the EoM (1) forms the system wrench, which consists of the mass-inertia matrix, $M(\mathbf{q}) \in \mathbb{R}^{n\times n}$, the centrifugal and Coriolis vector, $\mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}) \in \mathbb{R}^n$, the gravitational vector, $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^n$, and the external wrench applying on the system, $\mathbf{w}_e \in \mathbb{R}^n$. The system wrench is related to the cable force vector $\mathbf{f} \in \mathbb{R}^m$ by the joint-cable Jacobian matrix $L(\mathbf{q}) \in \mathbb{R}^{m\times n}$. Due to the property that cables can only exert pulling forces, the cable forces are constrained by a set of positive minimum and maximum feasible cable forces, f_{min} and f_{max}, such that

$$0 \le f_{min} \le f_i \le f_{max}, \quad i \in \{1, \ldots, m\}. \qquad (2)$$

B. Update Methodology in CASPR

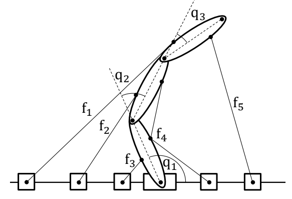
In CASPR, the complete system model is comprised within the `SystemModel` class, which can be further divided into two components: `SystemModelBodies` for rigid bodies, and `SystemModelCables` for cables, as shown in figure 2. Matrices in the EoM (M, $\mathbf{C}$, $\mathbf{G}$, L)

[1]Workspace analysis results obtained with the wrench-closure condition.



f1
f2
f3
f4
f5
f6
f7
f8
O
$q = \begin{bmatrix} x \\ y \\ z \\ \alpha \\ \beta \\ \gamma \end{bmatrix}$

(a) Example spatial single-link cable-driven robot (SCDR)



(b) Example multi-link cable-driven robot (MCDR)

Fig. 1: Example CDPRs

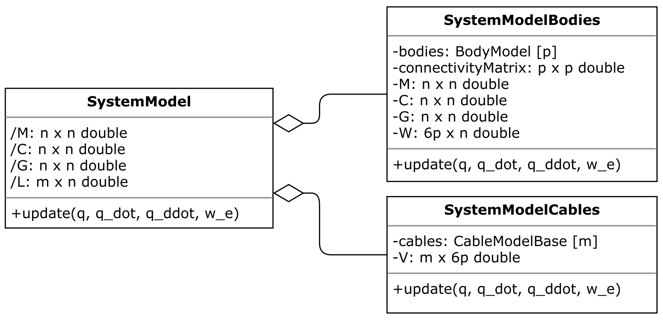


Fig. 2: Structure of System Dynamics Updates in CASPR

are calculated using the `update(q, q_dot, q_ddot, w_e)` function of `SystemModel`, in which the individual `update(...)` functions of the two components are called.

The flow of how system dynamics is updated in CASPR is shown in Algorithm 1. A brief description of the algorithm is provided below. Further details regarding the mathematical formulations are explained in [12], [13].

For each of the p rigid bodies, the *relative* kinematics such as the relative translation and rotation $\mathbf{r}_{rel}$ and R, respectively, is first updated. The relative kinematics of the bodies is then combined to obtain the *absolute* kinematics, such as the absolute joint position $\mathbf{r}_{OP}$ and rotation matrices R_{Ok}, which is used to construct the joint-body space Jacobian W. Using W, the absolute kinematics and the system parameters, body-space EoM variables M_b, $\mathbf{C}_b$ and $\mathbf{G}_b$ are computed. Joint-space EoM variables can then be obtained by

$$M = W^T M_b, \quad \mathbf{C} = W^T \mathbf{C}_b, \quad \mathbf{G} = W^T \mathbf{G}_b. \qquad (3)$$

After updating each of the m cable objects in `SystemModelCables`, the updated absolute kinematics, cable length vectors $\mathbf{l}(\mathbf{q})$ and cable routing matrix (CRM) [12] are used to determine the cable-body space Jacobian V.

The overall system joint-cable Jacobian L is then computed by multiplying the V and W Jacobians, i.e. $L = VW$, such that $\dot{\mathbf{l}} = VW\dot{\mathbf{q}} = L\dot{\mathbf{q}}$. As a result, matrices required in the EoM (M, $\mathbf{C}$, $\mathbf{G}$ and L) are all updated.

III. GENERALISED MODEL COMPILATION

This section illustrates the proposed generalised model compilation method (*compiled mode*) on CASPR. Motivated by the need for faster system dynamics computational speed, a hybrid of the closed-form and the numerical-based ap-

Algorithm 1 Update System Dynamics in CASPR (`update` function)

```
Input: q: pose, $\dot{\mathbf{q}}$: time derivative of pose, $\ddot{\mathbf{q}}$: second derivative of pose, $\mathbf{w}_e$: external wrench;
Output: $M$: mass-inertia matrix, $\mathbf{C}$: centrifugal and Coriolis vector, $\mathbf{G}$: gravitational vector, $L$: joint-cable Jacobian;
 1: // Rigid Body Update
 2: for each rigid body do
 3:     Update relative kinematics $\mathbf{r}_{rel}$, $R$;
 4: end for
 5: Compute absolute system kinematics $\mathbf{r}_{OP}$, $R_{Ok}$;
 6: Compute joint-body space Jacobian $W$;
 7: Compute body-space EoM variables $M_b$, $\mathbf{C}_b$, $\mathbf{G}_b$;
 8: Compute joint-space EoM variables $M$, $\mathbf{C}$, $\mathbf{G}$ with $W$, $M_b$, $\mathbf{C}_b$, $\mathbf{G}_b$;
 9: // Cable Update
10: for each cable do
11:     Update cable lengths $\mathbf{l}$ and cable segment vectors $\mathbf{s}$;
12: end for
13: Compute cable-body space Jacobian $V$ using CRM;
14: // System Jacobian
15: Compute joint-cable Jacobian $L$;
```

Algorithm 2 Update System Dynamics under Compiled Mode in CASPR

```
Input: Symbolic $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, $\mathbf{w}_e$;
Output: Update dynamics variables (e.g. $M$, $\mathbf{C}$, $\mathbf{G}$, $L$) using MATLAB function files;
 1: Declare symbolic input variables $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, $\mathbf{w}_e$
 2: Generate symbolic dynamics variables by passing symbolic $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, $\mathbf{w}_e$ into the update function
 3: Simplify symbolic dynamics variables;
 4: Compile symbolic dynamics variables into MATLAB function files by calling matlabFunction;
 5: Set update function to @compiledUpdate, in which compiled files are called to update system dynamics;
```

proach is developed. Utilising the CASPR generalised model, terms necessary in CDPR analysis are *compiled* into files with closed-form expressions. These files are then directly called to update the system dynamics without the recursive computations required by the numerical method.

The overall method flow is presented in Algorithm 2. An example of the code needed for the compilation of the body-space mass-inertia matrix M_b is then shown in Code sample 1 with the resulting MATLAB function file `compiled_M_b.m` depicted in code sample 2.

Code Sample 1: Compile symbolic M_b to MATLAB function `compile_M_b.m`

```
matlabFunction(M_b, 'File', 'compile_M_b', ...
     'Vars', {q, q_dot, q_ddot, W_e});
```

Code Sample 2: Compiled file for M_b on a 2-link, 4-DoF MCDR

```
function out1 = compile_M_b(in1,in2,in3,in4)

q2 = in1(2,:);
q3 = in1(3,:);
q4 = in1(4,:);
t2 = cos(q2);
t3 = cos(q3);
t4 = sin(q3);
t5 = sin(q2);
t6 = sin(q4);
t7 = cos(q4);
t8 = t7.*2.18e2;
t9 = t8+7.9e1;
...
```

A. Selection of Variables for Compilations

As the number of variables to be compiled increases, more time is required for the compilation due to a growth in model complexity. It is thus important to maintain a balance between the desired improvement in computational speed and the compilation time. In CASPR, variables are selected to be compiled based on two main criteria: 1) frequency of usage in analyses; 2) complexity of update and compilation.

To reduce the system update computational time and shorten the compilation time, variables that are seldom used within the analyses are not compiled/updated in this mode. The body EoM variables M_b, $\mathbf{C}_b$ and $\mathbf{G}_b$ are necessary for joint interaction forces and moments analyses [10]. Additionally, the cable length vector $\mathbf{l}(\mathbf{q})$ is also crucial in both forward and inverse kinematics analyses. The computation of these variables in Algorithm 1 (lines 7, 11) is therefore compiled in advance such that the `update` operation can replace these lines with individual function calls to the compiled M_b, $\mathbf{C}_b$, $\mathbf{G}_b$ and $\mathbf{l}(\mathbf{q})$ variables. In contrast, some properties of individual rigid bodies and cables, such as the relative rotation matrices of individual links R, are seldom directly accessed in common analyses. These matrices are neither compiled nor updated in the new mode.

The complexity of updating and compiling variables must also be considered. Although the joint-space EoM variables M, $\mathbf{C}$ and $\mathbf{G}$ are frequently used in dynamics analyses, compilations of these variables are not preferred. The reason is that the symbolic expressions of M, $\mathbf{C}$, $\mathbf{G}$ are constructed by multiplying the symbolic W and the body EoM matrices. However, these variables can be obtained by multiplying the numerical results of W, M_b, $\mathbf{C}_b$ and $\mathbf{G}_b$. This multiplication takes comparable computational time to the compiled operations. As a result, instead of compiling separate files for the joint-space EoM variables, the body-space EoM variables and W are compiled and are used to calculate M, $\mathbf{C}$ and $\mathbf{G}$.

B. Simplification of Variables

Direct compilation from symbolics can require a long time, in particular on CDPRs with high DoF. Other than selecting variables with the above criteria, matrices are simplified before compilation to reduce the compilation time. Common symbolic programming softwares such as

TABLE I: Comparison of the Computational Time and Compilation Time on a 2-link, 4-DoF MCDR

	Computational Time (ms)	Compilation Time (s)
Without Simplification	0.16	2058.1
With Simplification	0.14	90.8

Mathematica [x] and Maple [x] provide functions for simplification of symbolics. For better integration with CASPR, in the compiled mode, simplification is performed with the `simplify` function included in the Symbolic Math Toolbox of MATLAB [x]. Code Sample 3 shows the code for simplifying M_b, with simplification steps specified.

Rather than direct simplification of large matrices, simplification is performed on the elements required in constructing the final matrices. For example, rotation matrices R_{Ok} are the basic elements of matrices such as the Jacobian W [12] and are hence simplified before the construction and simplification of the Jacobian. This approach results in a number of small and quick simplifications taking place instead of a single large and slower simplification. Table I shows the comparison of the computational time for system update and the compilation time, with and without simplification on a 2-link, 4-DoF multi-link cable-driven robot (*MCDR*). The compilation time is observed to be significantly higher without simplification, with trivial difference in computational time for system update.

Code Sample 3: Simplification of the body-space mass-inertia matrix, M_b

```
obj.M_b = simplify(obj.M_b, 'Step', 20);
```

IV. CASE STUDIES

In this section, the proposed method is demonstrated by comparing simulation results on different CDPRs using CASPR. The CDPRs used in this paper are a 3-DoF planar robot, the 6-DoF IPAnema1 spatial robot [1], the 2-link, 4-DoF BioMuscular Arm (*BM-Arm*) and a 2-link MCDR with 2 spherical joints. These robots are chosen based on their differences in links, DoFs, and complexity in system dynamics calculations. Simulations are conducted on a hardware with the Intel Core i5-6500 CPU @ 3.20GHz and 8.00GB of RAM, using MATLAB R2017a (64-bit).

Section IV-A compares the computational time under the previous *default mode* and the new *compiled mode*. Case studies in forward kinematics (FK), inverse dynamics (ID), and workspace analysis are performed. Details of the results are shown in Table II to V. Section IV-B discusses the factors affecting the time required for the compilation process on the tested models with results displayed in Table VI.

A. Computational time

1) Forward Kinematics: The forward kinematics (*FK*) problem looks to find the pose $\mathbf{q}$ given a set of cable lengths $\mathbf{l}$. To solve FK problems, system dynamics may have to be updated more than once, depending on the solving algorithm. In this paper, the least square method [14] is used.

TABLE II: Forward Kinematics Computation Times

	CDPR Models			
FK	**Planar**	**BM-Arm**	**IPAnema1**	**2S-MCDR**
Update				
- *Default* (ms)	1.64	2.80	2.81	3.37
- *Compiled* (ms)	0.13	0.14	0.14	0.16
- *Improvement* (%)	92.38	94.99	95.05	95.16
Time Step				
- *Default* (ms)	10.26	14.78	17.48	17.83
- *Compiled* (ms)	3.60	3.59	4.31	3.96
- *Improvement* (%)	64.88	75.68	75.36	77.81

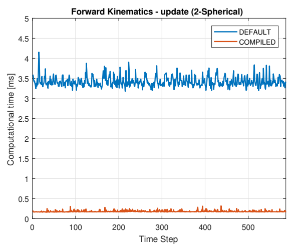
This case study compares the FK computational time under the two modes. Two types of computational time are recorded: 1) time to run the `update` function; and 2) time to run the whole time step. In all FK simulations, the CDPRs are required to run on a trajectory that involves movements in all DoFs. For example, a 10-second parabolic blend trajectory between the coordinates $\mathbf{q_s} = [0.3, 0.5, 0.0]^T$ and $\mathbf{q_e} = [0.4, 0.3, 0.1]^T$ is used on the 3-DoF planar robot, where $\mathbf{q} = [x, y, \theta]^T$ represents the x and y position, and the orientation of the planar robot, respectively.

Table II shows the time results of the FK analysis and the improvements made by the compiled mode. Using the default mode, the time for calling the `update` function alone takes 1.6 ms to 3.4 ms. After compilations, the time reduces by an average of 94.4%, with a range of time from 0.13 ms to 0.16 ms. Figure 3a shows that the computational speed of the `update` function has a significant improvement at every time step along the tested trajectory.

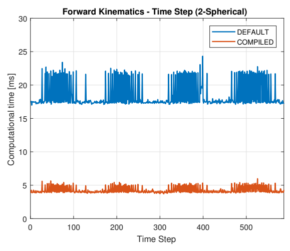
Figure 3b shows the time needed for each complete time step along the trajectory. The improvement ranges from around 65% to 78%. It is important to note that the `update` function is called multiple times when solving the FK. The improvement contributed by one call of the function, which range from 1.5 ms to 3.2 ms is also multiplied accordingly. The absolute differences under the two modes hence range from 6.7 ms to 13.9 ms.

As the complexity of the system increases, which means an increase in number of links, cables, and/or DoFs, it is observed that the computational time of the `update` function also shows an increasing trend under both of the modes. However, the increase is much more significant in the case of the default mode. In contrast, the function under compiled mode scales well as the complexity of robots increases. This result shows the high scalability of the proposed strategy.

2) Inverse Dynamics: In inverse dynamics (ID) analysis, a set of cable forces that satisfies the EoM (1) and the force bound (2) is determined at each time step to track a desired trajectory, where the CDPR actuation redundancy makes the solving of ID non-trivial. In this paper, ID is solved using the `quadprog` function from the optimisation toolbox of

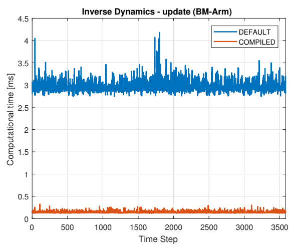


(a) Update Function

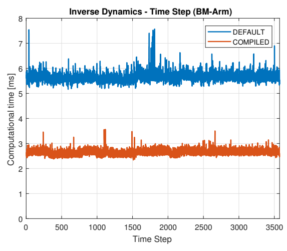


(b) Complete Time Step

Fig. 3: 2-Spherical MCDR FK Computation Time



(a) Update Function



(b) Complete Time Step

Fig. 4: Inverse Dynamics analysis on BM-Arm

TABLE III: Computation Time in Inverse Dynamics Analysis

	CDPR Models			
ID	**Planar**	**BM-Arm**	**IPAnema1**	**2S-MCDR**
Update				
- *Default* (ms)	1.72	3.00	2.95	3.54
- *Compiled* (ms)	0.11	0.14	0.15	0.20
- *Improvement* (%)	93.35	95.37	94.92	94.35
Time Step				
- *Default* (ms)	4.20	5.69	5.71	6.17
- *Compiled* (ms)	2.43	2.58	2.72	2.81
- *Improvement* (%)	42.17	54.65	52.35	54.49

TABLE IV: WCC Workspace Analysis Computation Time

	CDPR Models			
WCC	**Planar**	**BM-Arm**	**IPAnema1**	**2S-MCDR**
Update				
- *Default* (ms)	1.63	2.86	2.79	3.46
- *Compiled* (ms)	0.11	0.15	0.12	0.19
- *Improvement* (%)	93.17	94.66	95.73	94.57
Grid Point				
- *Default* (ms)	1.94	3.31	2.99	4.00
- *Compiled* (ms)	0.38	0.57	0.28	0.71
- *Improvement* (%)	80.41	82.91	90.75	82.14

MATLAB with the *interior-point-convex* algorithm.

Similar to the FK analysis, both the computational time of the `update` function and a complete time step is recorded. Time results and the improvements are presented in Table III. It is observed that the time needed solving the system dynamics shows similar improvements as in the FK analysis, with an average of 94.5%.

Improvement is also seen in the case of the complete time step. Figure 4b shows the computational time of a time step under the two modes, along a trajectory running on the BM-Arm. The compiled mode shows a steady improvement of around 54.7% along the trajectory. Other CDPRs also show improvements with an average of 50.9%. The results under compiled mode also present good scalability, with only approximately 0.4 ms increase as the tested robot changed from the planar robot to the 2S-MCDR.

It is observed that the improvement in terms of percentage, is not as drastic as that in the FK analysis. The reason is that the `update` function is only called once at each time step when solving ID. Moreover, time is needed for the solving of the non-trivial ID problem, but the time needed by the `quadprog` function to solve the quadratic program is not affected by the compiled update functions. The impact made by the compiled mode is hence less than that in the FK case.

3) Workspace Analysis: In this case study, wrench-closure condition (WCC) and wrench-feasible condition (WFC) are used to perform the workspace analysis. Both conditions are checked at 200 points in the feasible space of $\mathbf{q}$ (*grid points*), at which system dynamics is updated. The computational time for the `update` function, and the time needed for the complete analysis at each grid point are compared.

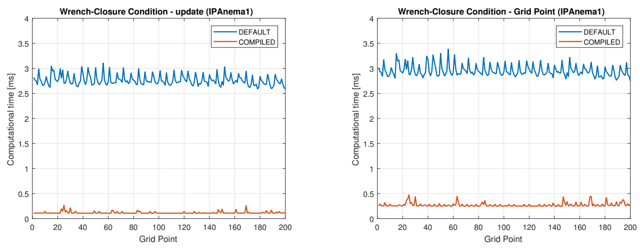
The WCC considers the CDPR's ability to produce arbitrary wrench, given unbounded positive cable forces. In this paper, the combinatorics null space method [15] is used for WCC analysis, with the results depicted in Table IV. The compiled `update` function shows substantial improvement with an average of 94.5% and good scalability.

In the case of the complete analysis at a grid point, speed improvements are observed with a range of 80.4% to 90.8%. Comparing to the FK and ID case studies, the higher percentage of improvement is due to the larger proportion of the `update` function costs in the whole calculation cycle. In WCC analysis, the time taken by the `update` function at each grid point has an average of 87.5% under the default mode, meaning that the improvement made by the compiled mode has a significant effect on the whole process.

In contrast to WCC, which allows unbounded cable forces, WFC is defined as the ability of the CDPR to produce a certain set of wrenches, given the cable force bounds (2). The force constraints result in a higher complexity in calculations and checking of conditions than in WCC analysis.

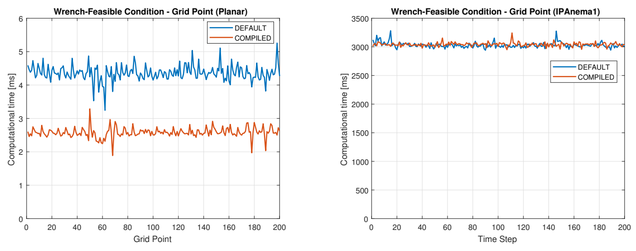
Table V shows the time results in workspace analysis with WFC. The time used at each grid point is significantly higher than that with WCC. On the IPAnema1 robot, the time needed for the calculations at each grid point is around 3 seconds, which is over a thousand times than that in WCC.

Despite the improvement of the compiled `update` function, the computational time change of the whole process is insignificant, especially for robots with higher complexity. Consider the results shown in figure 6a and 6b. On the planar



(a) Update Function (b) Complete Grid Point

Fig. 5: WCC IPAnema1 Workspace analysis



(a) Planar Robot (b) IPAnema1

Fig. 6: WFC Computational Time Analysis

TABLE V: WFC Workspace Analysis Computation Time

	CDPR Models			
WFC	**Planar**	**BM-Arm**	**IPAnema1**	**2S-MCDR**
Update				
- *Default* (ms)	1.74	3.12	3.24	4.01
- *Compiled* (ms)	0.13	0.23	0.26	0.35
- *Improvement* (times)	92.76	92.76	91.97	91.25
Grid Point				
- *Default* (ms)	4.31	25.00	3037.85	2678.31
- *Compiled* (ms)	2.60	22.19	3013.27	2651.78
- *Improvement* (%)	39.70	11.24	0.81	1.36

TABLE VI: Compilation Time on different CDPRs

Compilation Time (s)	**Planar**	**BM-Arm**	**IPAnema1**	**2S-MCDR**
- Without Simplification	5.1	2058.1	1183.8	23167.2
- With Simplification	4.9	90.8	158.2	784.4
- Improvement (%)	3.36	95.59	86.64	96.61

robot, approximately 39.7% of the time needed is reduced, while on the IPAnema1 robot, no observable improvement is found. This is a result of other calculations such as the evaluation of the workspace conditions occupying most of the computational time for these cases.

B. *Compilation time*

The time needed for compilation on the tested CDPRs is presented in Table VI. As explained in Section III, simplification shortens the compilation time by reducing the complexity of the compilation variables. The effect is found to be significant in this case study, with a maximum improvement of 96.6% on the 2S-MCDR.

As the complexity of the system increases, the compilation time increases accordingly. In particular, the time on the 2S-MCDR is significantly higher than the other robots, although the number of links and DoF are the same as the BM-Arm and IPAnema1, respectively. It is hence shown that the high complexity of the rotation matrices on spherical joints potentially leads to a large rise in compilation time.

Moreover, it is observed that simplification is more effective as the complexity of the system increases. On the planar robot, the reduction in compilation time is insignificant. However, drastic reductions of over 86% are found on the other 3 robots with higher DoF. This result shows the particular importance of simplification on complex systems.

V. Conclusion

A generalised model compilation method that improves the computational speed of CDPR system dynamics was presented. The method was implemented on CASPR by modifying the existing structure of the numerical model and compiling symbolic expressions of the system dynamics into closed-form functions. Strategies regarding the selection and simplification of compilation variables were discussed to improve compilation performance. To illustrate the improvement contributed by the compilations, case studies on forward kinematics, inverse dynamics and workspace analysis were performed. Despite the long compilation time in particular for robots of high DoF, the improvements on the computational speed of the system dynamics were extremely significant and required when performing intensive CDPR analysis algorithms. The successful implementation on CASPR shows the method's potential to be extended to other systems with similar numerical system models.

References

[1] A. Pott, H. Mütherich, W. Kraus, V. Schmidt, P. Miermeister, and A. Verl, "Ipanema: A family of cable-driven parallel robots for industrial applications," in *Cable-Driven Parallel Robots*, ser. Mech. and Mach. Sci., vol. 12, pp. 119–134.

[2] N. Riehl, M. Gouttefarde, S. Krut, C. Baradat, and F. Pierrot, "Effects of non-negligible cable mass on the static behaviour of large workspace cable-driven parallel mechanisms," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 2193–2198.

[3] S. Ma, S. Hirose, and H. Yoshinada, "CT ARM-I: coupled tendon-driven manipulator model i-design and basic experiments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1992, pp. 2094–2100.

[4] S. Wittmeier, C. Alessandro, N. Bascarevic, K. Dalamagkidis, D. Devereux, A. Diamond, M. Jäntsch, K. Jovanovic, R. Knight, H. G. Marques, P. Milosavljevic, B. Mitra, B. Svetozarevic, V. Potkonjak, R. Pfeifer, A. Knoll, and O. Holland, "Toward anthropomimetic robotics: Development, simulation, and control of a musculoskeletal torso," *J. Artif. Life*, vol. 19, no. 1, pp. 171–193, 2013.

[5] Y. Mao and S. K. Agrawal, "Design of a cable-driven arm exoskeleton (CAREX) for neural rehabilitation," *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 922–931, 2012.

[6] S. Kawamura, W. Choe, S. Tanaka, and S. R. Pandian, "Development of an ultrahigh speed robot FALCON using wire drive system," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1995, pp. 215–220.

[7] P. Bosscher, R. L. Williams II, L. S. Bryson, and D. Castro-Lacouture, "Cable-suspended robotic contour crafting system," *Autom. Constr.*, vol. 17, no. 1, pp. 45–55, 2007.

[8] R. Nan, D. Li, C. Jin, Q. Wang, L. Zhu, W. Zhu, H. Zhang, Y. Yue, and L. Qian, "The five-hundred-meter aperture spherical radio telescope (fast) project," *International Journal of Modern Physics D*, vol. 20, no. 06, pp. 989–1024, 2011.

[9] S. Rezazadeh and S. Behzadipour, "Tensionability conditions of a multi-body system driven by cables," in *Proc. ASME Int. Mech. Eng. Congress and Exposition*, 2007, pp. 1369–1375.

[10] D. Lau, D. Oetomo, and S. K. Halgamuge, “Inverse dynamics of multilink cable-driven manipulators with the consideration of joint interaction forces and moments,” *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 479–488, 2015.

[11] ——, “Wrench-closure workspace generation for cable driven parallel manipulators using a hybrid analytical-numerical approach,” *J. Mech. Des.*, vol. 133, no. 7, pp. 071 004/1–7, 2011.

[12] ——, “Generalized modeling of multilink cable-driven manipulators with arbitrary routing using the cable-routing matrix,” *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1102–1113, 2013.

[13] D. Lau, J. Eden, Y. Tan, and D. Oetomo, “CASPR: A comprehensive cable-robot analysis and simulation platform for the research of cable-driven parallel robots,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 3004–3011.

[14] D. Lau, “Initial length and pose calibration for cable-driven parallel robots with relative length feedback,” 07 2018, pp. 140–151.

[15] B. Ouyang and W.-W. Shang, “A new computation method for the force-closure workspace of cable-driven parallel manipulators,” *Robotica*, vol. 33, no. 3, pp. 537–547, 2015.