Extended-configuration-space modelling: comparison, mapping to reduced order and real-time simulation of Lagrangian dynamics formulations of parallel manipulators

Abstract

This paper presents a method for the real-time simulation of Lagrangian formulation of the dynamics of parallel manipulators. A choice of generalised coordinates called the extended-configuration-space is introduced, which builds on the configuration, actuator and task-space coordinates. In particular, the proposed method presents the versatility to be transformed either into task-space or actuator-space models without sacrificing the parent models' computational advantage. A comparative numerical study illustrates the accuracy and efficiency of the proposed method. Simulation studies on a semi-regular Stewart platform manipulator (SRSPM) and the 6-RSS manipulator are presented to illustrate the utility of the proposed formulation.
Keywords: Lagrangian dynamics, parallel manipulators, extended-configuration-space, real-time simulation, root-tracking

1. Introduction

(sc:intro) Simulation of parallel manipulators is difficult, due to complexity in the dynamics modelling techniques, existence of singularities inside the workspace, stabilisation of the numerical methods used etc. Several approaches to formulate the dynamics model of parallel manipulators exists in the literature including Bond graph-based methods [1](damic2015dynamic), Kane's method [2](liu2000dynamics), virtual work [3](tsai2000solving), modified virtual work [4](wang1998new) etc. However, the Newton-Euler and the Euler-Lagrangian frameworks remain the most two popular methods used for the modelling and analysis of constrained mechanical systems. Although the recursive Newton-Euler algorithms are preferred in physics simulation environments [5](todorov2012mujoco), Euler-Lagrangian methods result in a set of equations compatible for motion simulation [6](saha1999dynamics). Further, Newton-Euler methods generally imply the need for distinct algorithms for inverse and forward dynamics, whereas the Lagrangian methods do not. The latter method also elucidates the differential-algebraic structure, which allows the study of controllability [7](choudhury2000singularity), design of non-linear controllers [8](murray1995nonlinear) etc. and hence deserve attention for the modelling and simulation of constrained mechanical systems. For an in-length discussion on this

topic, refer to [9](lewis2007worth).

Initial studies on the simulation of parallel manipulators, especially of the Stewart platform type manipulators shown in Fig. 1(fig:srspm), were conducted neglecting the leg masses to simplify the problem as done in [10](fichter1986stewart). Later, the effects of the leg inertias on the accuracy of the dynamics was studied in [11](ji1993study). It was concluded that the leg inertia had pronounced influence on the dynamic behaviour of the system, either when the payload mass is comparably high or when the end-effector moves fast. Since, several works have used the simplifications as mentioned above for simulation or development of control algorithms like the works [12, 13](li1997modeling, davliakos2008model) which neglect dynamic effects of the legs of SPM. Further, the works [14, 15](lee2003position, nguyen1993adaptive) restrain the workspace and velocity of the moving platform to assume the non-linear terms to be constant for the design of a position controller. Corroborating with [11](ji1993study), these works have observed that there were significant errors, especially while tracking sharp turns like corners. Moreover, in practical implementations, the models are often simplified and are compensated by the control algorithms implemented. However, even with a precise formulation of the dynamics model, real-time model-based control of the Lagrangian dynamics model of parallel manipulators remains a challenge [16](kuo2014experimental).

Dasgupta et. al. [17](dasgupta1998closed) regard task-space description as natural for parallel manipulators and joint-space for serial manipulators. The initial attempts for a task-space description of Lagrangian dynamics models for an SRSPM are presented in [18, 19](geng1992dynamic, liu1993singularities); a complete description is given only in [20](lebret1993dynamic). However, closed-form expressions of the linear and angular velocity Jacobian matrices of the legs were not obtained in this work. General closed-form dynamics expressions for the coefficient matrices in terms of the task-space variables was not given until [21](dasgupta1999general). The work also describes the computational advantage of simulation using a smaller set of variables to describe the system. Hence, task-space is suitable over actuator-space as the latter involves computation of the forward kinematic (FK) solutions at each iteration. Apart from task-space, configuration and actuator-spaces are other frequent choices for the formulations of equations of motion.

Parallel manipulators are prone to singularities within their workspaces called gain-type singularities which are influenced by choice of the generalised coordinates selected. These gain-type singularities of parallel manipulators, in turn, reflect in their dynamic behaviour. A detailed study on the effect of singularities on various Lagrangian dynamics formulations is presented in [22](MURALIDHARAN2018403). Therefore, the current work focusing on simulation meth-

ods, adopts the concept of safe working zone (SWZ) detailed in [23](murali2019computation) and assumes that the manipulator is working within this region for the entire duration of motion.

The advantages of choice of a redundant coordinate set for modelling constrained multi-body systems is studied in [24](ryan1990adams). A Newton-Euler framework utilising a highly redundant coordinate formulation is presented in [25](franchi1995highly). The work illustrates a better computational performance even without resorting to the development of any dedicated algorithm to exploit the sparse system of equations generated with the selection of highly redundant coordinates. More recently, Masarati et al. [26](masarati2014efficient) have offered a method to build a general-purpose multi-body analysis software using redundant coordinates. Further, [27](fumagalli2009real) has demonstrated the real-time computation capability of inverse dynamics of an 11 degree-of-freedom parallel manipulator exploiting the choice of redundant coordinates.

The current work deals with a specific set of redundant coordinates to formulate the equations of motion of parallel manipulators using the Lagrangian framework. The objective of this paper is to study this formulation called the extended-configuration-space modelling and understand its relative precedence in the simulation of parallel manipulators as compared to the configuration, actuator and task-space dynamics models. It is to be noted that the present method offers a computational advantage to systems with lower degrees of freedom and whereas, for systems with a large number of degrees of freedom it is advantageous to use recursive algorithms [6](saha1999dynamics). The main contributions of this paper can be summarised as:

- The extended configuration-space modelling and its interchangeability between models
- A comparative study of the proposed model with other popular models
- Numerical studies comparing the accuracy and computational time of various models
- The formulation and simulation of dynamics of the 6-RSS manipulator using the proposed model
- All the codes necessary for the simulation in C++ and Mathematica11.2 are made available in the project repository, https://github.com/akhilsathuluri/SRSPM_dynamics

The rest of the paper is organised as follows: all the required notations are given in Section 2(sc:notation) and the standard Lagrangian based dynamics formulation is illustrated in Section 3(sc:lagform). The extended-configuration-space dynamics model is discussed in Section 4(sc:extconf). The simulation and comparison of all the dynamics formulations is provided in Section 5(sc:sim). An example

simulation of the 6-RSS manipulator is given in Section 6(sc:rss). Finally, the conclusions and the future work are presented in Section 7(sc:conc). The appendix contain details of implementation in C++ and Mathematica11.2.

2. Basic definitions and notations

(sc:notation) The necessary set of assumptions and definitions are presented in the current section. Parallel manipulators fall under the classification of constrained mechanical systems. The architecture of parallel manipulators is characterised by their closed-loops and hence have both active, i.e., actuated and passive joints. The condition imposing all the closed-loops to remain intact at all times, gives rise to the loop-closure constraints, written as,

$$\boldsymbol{\eta}(\boldsymbol{q}) = \mathbf{0}, \text{ (eq:loopclosure)} \tag{1}$$

where $\boldsymbol{q} = [\boldsymbol{\theta}^\top, \boldsymbol{\phi}^\top]^\top$, and $\boldsymbol{\theta} \in \mathbb{R}^n$, $\boldsymbol{\phi} \in \mathbb{R}^m$ are the n active and m passive variables representing the respective joint angles of the manipulator. Since the loop-closure equations do not explicitly depend on time, they are scleronomic in nature [28](udwadia2007analytical) and derivative of Eq. (1) with respect to time can be written as,

$$\frac{d\boldsymbol{\eta}}{dt} = \boldsymbol{J}_{\boldsymbol{\eta q}}\dot{\boldsymbol{q}} = \mathbf{0}, \quad \text{where } \boldsymbol{J}_{\boldsymbol{\eta q}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{q}}. \tag{2}$$

Expanding Eq. (2) in terms of the active and passive joint variables is given as:

$$\boldsymbol{J}_{\boldsymbol{\eta\theta}}\dot{\boldsymbol{\theta}} + \boldsymbol{J}_{\boldsymbol{\eta\phi}}\dot{\boldsymbol{\phi}} = \mathbf{0}, \quad \text{where } \boldsymbol{J}_{\boldsymbol{\eta\theta}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\theta}}, \ \boldsymbol{J}_{\boldsymbol{\eta\phi}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\phi}}. \tag{3}$$

$$\dot{\boldsymbol{\phi}} = \boldsymbol{J}_{\boldsymbol{\phi\theta}}\dot{\boldsymbol{\theta}}, \quad \text{where } \boldsymbol{J}_{\boldsymbol{\phi\theta}} = -\boldsymbol{J}_{\boldsymbol{\eta\phi}}^{-1}\boldsymbol{J}_{\boldsymbol{\eta\theta}}. \tag{4}$$

The matrices $\boldsymbol{J}_{\boldsymbol{\eta q}}$, $\boldsymbol{J}_{\boldsymbol{\eta\phi}}$ are called as the configuration Jacobian matrix and the constraint Jacobian matrix. Numerical evaluation of these matrices is critical for the simulation of the equations of motion of the system. The following are the key assumptions under which the above formulation remains valid:

1. All the links of the manipulator are assumed to be rigid,
2. There is no friction, and all the joints are ideal,
3. The mass distribution of all the links is uniform.

2.1. Semi-regular Stewart platform manipulator (SRSPM)

The Gough-Stewart or the Stewart platform manipulator is a six degree-of-freedom parallel manipulator. Its construction involves a fixed platform connected to six linear actuators, also called legs, via universal or spherical joints. A moving platform is attached to the other ends of the linear actuators via spherical joints, as shown in Fig. 1(fig:srspm). Since its introduction in [29](stewart1965platform), it has attracted a large amount of research on the topics of kinematics, dynamics and control. The interest in this particular manipulator comes from its wide range of

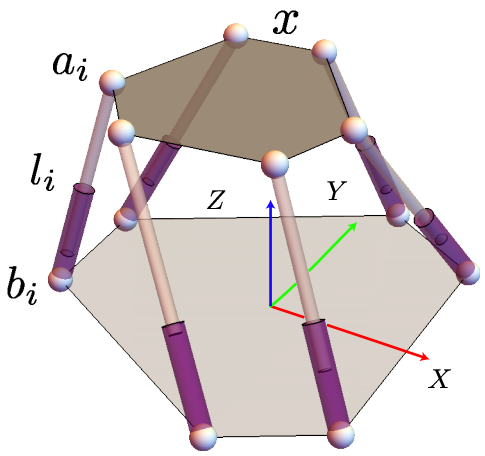


Figure 1: Architecture of a Semi-Regular Stewart Platform Manipulator

applications in automotive simulators [30, 31](freeman1995iowa, park2001development), machine tools [20](lebret1993dynamic), flight simulators [32](pradipta2013development) etc. The length of the legs (l_i) constitute the active variables ($\boldsymbol{\theta}$) and the configuration of the spherical joints attached to the rigid platform form the passive variables ($\boldsymbol{\phi}$). Whereas position and the orientation of the moving platform (end-effector) together constitute the task-space variable set ($\boldsymbol{x}$).

3. Lagrangian dynamics formulations of parallel manipulators

(sc:lagform) The current section provides brief descriptions of various dynamics formulations within the Lagrangian framework.

3.1. Configuration-space formulation

(sc:configspace) The set of actuated and passive joint variables together form the configuration-space or the joint-space of the manipulator. The equations of motion of parallel manipulators described in the configuration space can be derived using the constrained Lagrangian formulation and can be found in textbooks such as [33](ghosal2006robotics). The resulting equation of motion

is of the form:

$$\text{(eq:dynconfig)}\ \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{G} = \boldsymbol{Q}^{nc} + \boldsymbol{J}_{\boldsymbol{\eta q}}^{\top}\boldsymbol{\lambda}, \tag{5}$$

where, $\boldsymbol{M}$ is the generalised mass matrix, $\boldsymbol{C}$ is the combination of centripetal and Coriolis forces, $\boldsymbol{G}$ is the vector of gravitational and other potential forces, $\boldsymbol{Q}^{nc}$ is the vector of external forces, and $\boldsymbol{J}_{\boldsymbol{\eta q}}^{\top}\boldsymbol{\lambda}$ is the vector of constraint forces arising out of the loop-closure constraints Eq. (1). In the case of an SRSPM, the active variables representing the leg-lengths ($\boldsymbol{\theta}$) and the passive variables describing the configuration of the spherical joints at the fixed platform ($\boldsymbol{\phi}$), together form the configuration-space of the manipulator as shown in the Fig. 2. In this case, the Lagrangian multipliers $\boldsymbol{\lambda}$ can be

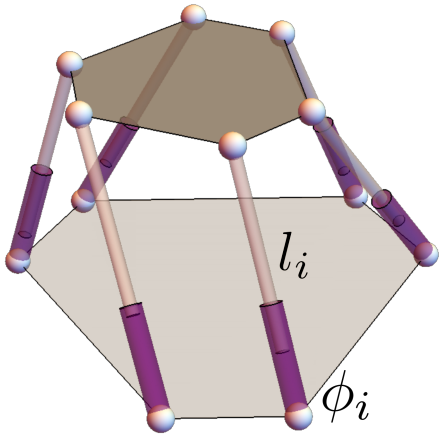


Figure 2: Configuration-space of the SRSPM

obtained by considering the derivative of Eq. (2) with respect to time as,

$$\boldsymbol{J}_{\boldsymbol{\eta q}}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{\boldsymbol{\eta q}}\dot{\boldsymbol{q}} = \boldsymbol{0}. \tag{6}$$

Substituting for $\ddot{\boldsymbol{q}}$ from Eq. 5, $\boldsymbol{\lambda}$ vector is obtained as:

$$\boldsymbol{\lambda} = -\boldsymbol{A}^{-1}\left(\dot{\boldsymbol{J}}_{\boldsymbol{\eta q}}\dot{\boldsymbol{q}} + \boldsymbol{J}_{\boldsymbol{\eta q}}\boldsymbol{M}^{-1}\boldsymbol{f}\right), \text{ where} \tag{7}$$

$$\boldsymbol{A} = \boldsymbol{J}_{\boldsymbol{\eta q}}\boldsymbol{M}^{-1}\boldsymbol{J}_{\boldsymbol{\eta q}}^{\top}, \tag{8}$$

$$\boldsymbol{f} = \boldsymbol{Q}^{nc} - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}. \tag{9}$$

Using Eq. (9), the forward dynamics model derived is of the form:

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}\boldsymbol{f} - \boldsymbol{M}^{-1}\boldsymbol{J}_{\boldsymbol{\eta q}}^{\top}\boldsymbol{A}^{-1}\left(\dot{\boldsymbol{J}}_{\boldsymbol{\eta q}}\dot{\boldsymbol{q}} + \boldsymbol{J}_{\boldsymbol{\eta q}}\boldsymbol{M}^{-1}\boldsymbol{f}\right). \tag{10}$$

For an in-depth discussion on the ranks of the Jacobian and the $\boldsymbol{A}$ matrices and their effects on the dynamic behavior of parallel manipulators, refer to [22](MURALIDHARAN2018403). The

necessary set of constraint equations are obtained by imposing the conditions of the rigidity of the moving platform adopted from [34](bandyopadhyay2006geometric), as explained below:

1. The initial six equations are derived by imposing a constraint on the relative distances between the vertices.
2. Rigid triangles are formed by placing a constraint on the distance between three consecutive vertices giving three more equations.
3. Ensuring all the planes of the above-formed triangles to be parallel, i.e., all the triangles are in the same plane result in three additional equations.

Together, they impose 12 constraint equations as functions of 18 variables, i.e., 6 active and 12 passive joint angle variables. For the complete formulation of the configuration-space dynamics model of SRSPM, refer to [35](anwar2012trajectory). The resulting coefficients $\boldsymbol{M}$ and $\boldsymbol{C}$ are 18×18 matrices, and the equations of motion are a set of 18 ordinary differential equations.

3.2. Actuator-space formulation

(sc:actspace) In the actuator-space formulation the number of generalised coordinates is exactly

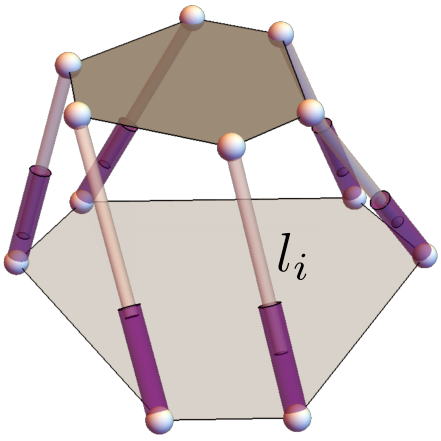


Figure 3: Actuator-space of the SRSPM

equal to the of degrees of freedom of the system as depicted in Fig. 3(fig:actspace). Therefore, the equations of motion in these coordinates does not involve the constraint Lagrangian term. Using the fact that constraint forces do not do any work, the actuator-space dynamics model can be derived by eliminating the Lagrangian multipliers, $\boldsymbol{\lambda}$.

$$\left(\boldsymbol{J}_{\boldsymbol{\eta q}}^{\top}\boldsymbol{\lambda}\right)^{\top}\dot{\boldsymbol{q}} = \boldsymbol{\lambda}^{\top}\boldsymbol{J}_{\boldsymbol{\eta q}}\dot{\boldsymbol{q}} = \boldsymbol{0}. \tag{11}$$

Further, from the equation Eq. (3),

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{q\theta}}\dot{\boldsymbol{\theta}}, \quad \text{where } \boldsymbol{J}_{\boldsymbol{q\theta}} = [\boldsymbol{J}_{\boldsymbol{\phi\theta}}\ \boldsymbol{I}_{6\times6}]. \tag{12}$$

Using the Eq. (12) defining the mapping between two variable sets $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{\theta}}$, the configuration-space system can be transformed into actuator-space formulation. This method is also referred to as the embedded Lagrangian technique [AS: find citation]. The resulting equations of motion are of the form,

$$\boldsymbol{M}_{\boldsymbol{\theta}}\ddot{\boldsymbol{\theta}} + \boldsymbol{C}_{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}} + \boldsymbol{G}_{\boldsymbol{\theta}} = \boldsymbol{Q}_{\mathrm{a}}^{nc}, \tag{13}$$

$\boldsymbol{M}_{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\boldsymbol{\theta}}$ and $\boldsymbol{G}_{\boldsymbol{\theta}}$ are analogues of the coefficient matrices derived in Eq. 5 for the configuration-space dynamics model. Since there are no closed-form solutions to the forward kinematics (FK) problem, a direct formulation of dynamics model in terms of the actuator-space variables is not feasible. Therefore, the model is formulated in a higher dimensional space, i.e., the configuration-space and is mapped to the actuator-space, resulting in an equivalent unconstrained dynamic system. The complete derivation of the actuator-space dynamics mode can be found in [36, 22, 37](nasa2011, MURALIDHARAN2018403, agarwal2016dynamic). In case of the SRSPM, this transformation results in the change of coordinates from 18 variables of configuration-space to 6 dimensional actuator-space. It should be noted that the formulation is dependent on the existence of the inverse of $\boldsymbol{J}_{\boldsymbol{\eta\phi}}$ matrix. The loss in rank of constraint Jacobian matrix is termed as the gain-type singularity which affects the mapping between the dynamics models.

Further, the mapping requires the computation of the passive joint variable values which involves solving the FK problem at every iteration of the simulation. Evaluating of the solutions to the FK problem is computationally intensive. To overcome this issue a Newton-Raphson based root-tracking algorithm is implemented in an intermediary step. The major advantage of this step is the ability to evaluate the values of passive joint angle variables using the constraint equations up to a required accuracy level.

3.3. Task-space formulation

(sc:taskspace) The task-space or the workspace variables describe the output space of a manipulator and are represented by $\boldsymbol{x}$ as shown in Fig. 4. The task-space generally consists of the Cartesian coordinates, $\{x, y, z\}$ and the Rodrigues' parameters, $\{c_1, c_2, c_3\}$ of the geometric centre, representing position and the orientation of the moving platform. The number of task-space variables are the same as the number of degrees of freedom and therefore there is no constraint term in the equations of motion and have a form similar to the actuator-space model:

$$\boldsymbol{M}_{\boldsymbol{x}}\ddot{\mathbf{x}} + \boldsymbol{C}_{\boldsymbol{x}}\dot{\mathbf{x}} + \boldsymbol{G}_{\boldsymbol{x}} = \boldsymbol{Q}_{\mathrm{x}}^{nc}, \tag{14}$$

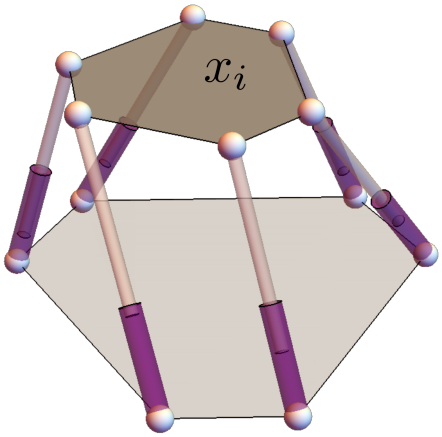


Figure 4: Task-space of the SRSPM

where $\boldsymbol{M_x}$, $\boldsymbol{C_x}$, $\boldsymbol{G_x}$ are the coefficient matrices of the task-space dynamics. Multiple ways to arrive at the Eq. (14) are explained below.

3.3.1. Direct task-space formulation

(sc:method1) The expressions of joint-space variables can be derived in terms of the task-space variables by solving the inverse kinematics (IK) problem of the manipulator. Therefore, linear and angular velocity Jacobian matrices can now be derived completely in terms of the task-space variables. A major drawback in this method is the resulting sizeable expressions of the derived Jacobian matrices. Moreover, it is computationally expensive to derive a closed-form expression for the $\boldsymbol{C}_x$ matrix using,

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{18}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{kj}}{\partial q_i}\right)\dot{q}_k. \tag{15}$$

Hence, the $\boldsymbol{C}_x$ is derived numerically from the closed-form expressions of the partial derivatives of the mass matrix, $\boldsymbol{M_x}$.

3.3.2. Indirect task-space formulation

(sc:method2) A much wiser way is to compute the coefficient matrices numerically at each step which involves computing the joint-space variables at each iteration of the simulation. The relation between configuration and task-space variables can be derived using the IK equations. Consider the time derivative of these IK equations,

$$\dot{\boldsymbol{q}} = \boldsymbol{J_{qx}}\dot{\boldsymbol{x}}, \quad \text{where } \boldsymbol{J_{qx}} = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}} \tag{16}$$

$$\text{and } \boldsymbol{q} = \boldsymbol{f}_{IK}(\boldsymbol{x}), \tag{17}$$

where $\dot{\boldsymbol{x}}$ are the task-space variables and $\boldsymbol{f}_{IK}$ represent the IK solutions. The expressions of the Jacobian matrix can be further simplified by splitting the configuration-space variables into active and passive joint coordinates which result in the following decomposition,

$$\boldsymbol{J}_{\boldsymbol{qx}} = \boldsymbol{J}_{\boldsymbol{q\theta}} \boldsymbol{J}_{\boldsymbol{\theta x}}. \tag{18}$$

Upon observation, it can be seen that using the decomposition in Eq. (18) is equivalent to mapping the actuator-space dynamics model to the task-space model using $\boldsymbol{J}_{\boldsymbol{\theta x}}$ Jacobian matrix. Therefore using the derived actuator-space model as the base model used in transformation to the task-space model leads to the evaluation of a smaller 6×6 matrix, $\boldsymbol{J}_{\boldsymbol{\theta x}}$ as opposed to the larger 18×18 matrix, $\boldsymbol{J}_{\boldsymbol{qx}}$. Resulting coefficient matrices of the task-space dynamics model is,

$$\begin{aligned}
\boldsymbol{M}_{\boldsymbol{x}} &= \boldsymbol{J}_{\boldsymbol{\theta x}}^{\top} \boldsymbol{M}_{\boldsymbol{\theta}} \boldsymbol{J}_{\boldsymbol{\theta x}}, \\
\boldsymbol{C}_{\boldsymbol{x}} &= \boldsymbol{J}_{\boldsymbol{\theta x}}^{\top} (\boldsymbol{C}_{\boldsymbol{\theta}} \boldsymbol{J}_{\boldsymbol{\theta x}} + \boldsymbol{M}_{\boldsymbol{\theta}} \dot{\boldsymbol{J}}_{\boldsymbol{\theta x}}), \\
\boldsymbol{G}_{\boldsymbol{x}} &= \boldsymbol{J}_{\boldsymbol{\theta x}}^{\top} \boldsymbol{G}_{\boldsymbol{\theta}} \boldsymbol{J}_{\boldsymbol{\theta x}}, \\
\boldsymbol{Q}_{\theta}^{nc} &= \boldsymbol{J}_{\boldsymbol{\theta x}}^{\top} \boldsymbol{Q}_{x}^{nc}.
\end{aligned}$$

This process is similar to the mapping done between configuration-space and actuator-space variables. Since most of the matrices are computed numerically at run-time, the dynamics model in Section 3.3.2(sc:method2) is computationally faster than the model described in Section 3.3.1(sc:method1) and all the other methods described above.

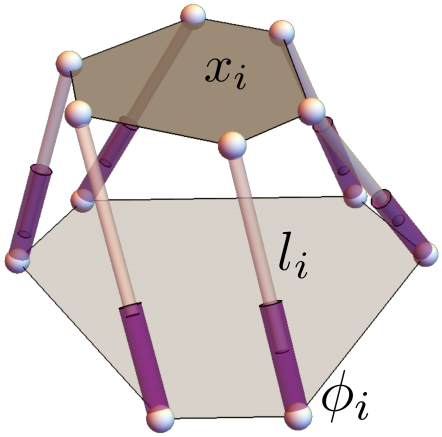
4. Extended-configuration-space formulation

(sc:extconf) As described in the introduction, there are several advantages to the usage of a redundant coordinate set. However, in general, the extra variables would mean enforcing additional constraints on the equations of motion leading to an increased complexity of the problem and computation of the constraint Lagrangian term. Consider an n degrees of freedom system described with m generalised coordinates. In such a case, an additional $m - n$ constraints are to be imposed on the system to completely determine its configuration. Even though through intelligent choice of variables it is possible to generate sparse coefficient matrices, it invariably results in a larger number of equations to be solved, increasing the size of the problem.

The choice of extended-configuration-space variables is a way to exploit the sparseness in the matrices while keeping the size of the problem small. Observing the earlier formulations in Sections 3.1(sc:configspace), 3.2(sc:actspace), 3.3(sc:taskspace), it is apparent that each method has its relative advantages and disadvantages. The primary computational burden in the configuration-space formulation is due to the calculation of velocity Jacobian matrices of the moving platform, whereas in the task-space it is due to the calculation of Jacobian matrices correspinding to the prismatic links. Description of the legs in the task-space model and the moving platform in the configuration-space model result in matrices with sizeable symbolic expressions which consume a significant amount of the computational time for evaluation. The set of generalised coordinates in the extended-configuration-space formulation represent each body of the system in its natural coordinates, i.e., describe all the elements with their associated variables as shown in Fig. 5(fig:extspace). Such a description would avoid posing one set of variables in terms of others and thereby reducing the complexity of the intermediate matrices involved.

As illustrated in the derivation of the actuator and task-space models in Section3.2 and 3.3, a dynamics model formulated in a higher dimension allows the flexibility of transformation between various variable sets. A distinct advantage of this specific choice of redundant variables means that the formulated model can be mapped to both actuator or task-space dynamics models. Therefore unlike [27](fumagalli2009real), the model not only remains sparse but the size of the problem also remains small, significantly reducing the computational time. Further, by the appropriate choice of variables to be mapped to, the constraint equations can be eliminated giving rise to the unconstrained form of the equations of motion.

4.1. Example formulation with SRSPM



(fig:extspace)

Figure 5: Extended-configuration-space of the SRSPM

As mentioned earlier, in the extended-configuration-space, every element of the manipulator is

described in its natural coordinates. For example, in the case of an SRSPM, it is easy to describe the top platform in the task-space coordinates whereas the legs in the joint-space. Following from this argument the set of active, passive and task-space variables together constitute the extended-configuration-space of the SRSPM manipulator.

$$\boldsymbol{q}_e = [\boldsymbol{x}^\top, \boldsymbol{\theta}^\top, \boldsymbol{\phi}^\top, \boldsymbol{\psi}^\top]^\top.$$

This particular choice of variables also result in a significant reduction of complexity in the formulation of the constraint equations. In the case of an SRSPM the constraints may be formulated by equating the positions of the vertices of the moving platform as traversed through each individual serial chain present along the legs,

$$\boldsymbol{\eta}(\boldsymbol{q}_e) = \mathbf{0}, \tag{19}$$

$$\boldsymbol{x} + \boldsymbol{R}_{tp}\boldsymbol{a}_{li} - \boldsymbol{a}_i = \mathbf{0}, \quad \text{where } i = 1, 2, \ldots 6, \tag{20}$$

and $\boldsymbol{x}$ is the position of the geometric centre $\{x, y, z\}$ and $\boldsymbol{R}_{tp}$ is the rotation matrix corresponding to the orientation of the moving platform; $\boldsymbol{a}_{li}$, $\boldsymbol{a}_i$ are the vertices of the moving platform described in the local and the global frames respectively. Such a formulation of constraint equations drastically simplifies the expressions corresponding the linear and angular velocity Jacobian matrices resulting in sparse coefficient matrices. Since there is no need to find the relations between different sets of variables (using FK or IK relations), it is easy to model the dynamics of a system using this method. Moreover, the derivation of the extended-configuration-space dynamics model is similar to deriving the configuration-space model. Hence, the form of the resulting equations of motion are equivalent to Eq. (5),

$$\boldsymbol{M}(\boldsymbol{q}_e)\ddot{\boldsymbol{q}}_e + \boldsymbol{C}(\boldsymbol{q}_e, \dot{\boldsymbol{q}}_e)\dot{\boldsymbol{q}}_e + \boldsymbol{G}(\boldsymbol{q}_e) = \boldsymbol{Q}_e^{nc} + \boldsymbol{J}_{\boldsymbol{\eta}\boldsymbol{q}_e}^\top \boldsymbol{\lambda}, \tag{21}$$

The resulting dynamics model consists of a set of 24 coupled non-linear differential equations, with the coefficient matrices as functions of the extended-configuration-space variables and their first-order time derivatives. However, note that this set of generalised coordinates do not convey any physical meaning as the prismatic joints are the only controllable variables.

In the case of an SRSPM, the 24 dimensional formulation enables multiple options to map the formulated model, either to actuator-space or to the task-space, both of which are only 6 dimensional. Such a mapping is similar to the transformation from 18 dimensional configuration-space to the actuator-space model of 6 dimensions. Further, mapping of the system is not possible to

the configuration-space model, as the Jacobian matrix, $\boldsymbol{J_{\eta x}}$ is rectangular and hence non-invertible. Therefore, the particular choice of extended-configuration-space provides the versatility of choosing a sub-space to be mapped to while retaining the sparseness of the matrices.

4.2. Extended configuration-space model mapped to task-space model

The works [20, 38](lebret1993dynamic, kim1998high) describe the importance of task-space dynamics in formulating controllers for various applications like machining and high-speed tracking. Further, task-space dynamics is preferred as it involves solving the simpler IK problem over the FK problem. Lee et. al. [14](lee2003position), describe the development of an IK based controller utilising the task-space dynamics model. The tasks-space dynamics model can be derived from the extended-configuration-space through the following mapping,

$$\{\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi}\} \mapsto \{\boldsymbol{x}\}. \tag{22}$$

Consider the derivative of the constraint equations in Eq. (20) with respect to time expanding it in terms joint and task-space coordinates.

$$\dot{\boldsymbol{q}} = -\boldsymbol{J}_{\boldsymbol{\eta q}}^{-1} \boldsymbol{J}_{\boldsymbol{\eta x}} \dot{\boldsymbol{x}}, \tag{23}$$

$$\text{where } \boldsymbol{J}_{\boldsymbol{\eta x}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{x}} \text{ and } \boldsymbol{J}_{\boldsymbol{\eta q}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{q}}. \tag{24}$$

From Eq. (24), it is apparent that the above mapping fails when $\boldsymbol{J_{\eta q}}$ is not a full ranked matrix. The condition where the configuration Jacobian matrix, $\boldsymbol{J_{\eta q}}$ is non-invertible is referred to as a configuration-space singularity, elaborated in [22]. The extended-configuration-space mapped to task-space model is abbreviated as extended-to-task-space model.

4.3. Extended configuration-space model mapped to actuator-space model

Despite the task-space dynamics model small and computationally faster, the actuator-space model is generally preferred for applications involving control and simulation of systems. The implementation of task-space dynamics requires the use of an external sensor to continuously identify the position and orientation of the moving platform. The actuated joints are controlled in the actuator-space dynamics model and hence their states are directly available. Therefore, the actuator-space is relevant in several practical applications as presented in [12, 39](li1997modeling,

su2004disturbance).

$$\{\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi}\} \mapsto \{\boldsymbol{\theta}\}. \tag{25}$$

The extended-configuration-space model provides a straightforward path for the formulation of an actuator-space model via transformation of the generalised coordinates. Let the set of task-space and passive joint angle variables together be denoted as,

$$\boldsymbol{q_x} = [\boldsymbol{x}^\top, \boldsymbol{\phi}^\top]^\top.$$

Consider the derivative of the constraint equations, Eq. (20), with respect to time,

$$\dot{\boldsymbol{q}} = -{\boldsymbol{J}_{\boldsymbol{\eta q_x}}}^{-1} \boldsymbol{J}_{\boldsymbol{\eta\theta}} \dot{\boldsymbol{\theta}}, \quad \text{where } \boldsymbol{J}_{\boldsymbol{\eta\theta}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\theta}} \text{ and } \boldsymbol{J}_{\boldsymbol{\eta q_x}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{q_x}}.$$

The Jacobian matrix $\boldsymbol{J}_{\boldsymbol{\eta q_x}}$ is responsible for the existance of the mapping and needs to be further studied to understand various conditions of singularity. Singularities play a significant role in determining the course of simulation of parallel manipulators. The coefficient matrices of the mapped models discussed in Sections 3.2(sc:actspace) and 3.3(sc:taskspace) are computed numerically at runtime via Jacobian matrices. Therefore, accuracy and computational speed are of major concern for dynamics models obtained through the transformation of coordinates. The extended-configuration-space mapped to actuator-space model is abbreviated as extended-to-actuator-space model.

5. Simulation and comparison between formulations

(sc:sim) This section presents the numerical simulation studies of the dynamics models using the SRSPM. As a convention, the local axis of the moving platform and the global axis fixed at its centre do not have any relative rotation between them. The moving-platform is initially at rest, i.e., $\dot{\boldsymbol{q}}(0) = \boldsymbol{0}$ with its geometric centre at a height, $z = 1.1\ m$. From the initial condition, the moving platform is let to fall freely under the influence of gravity. The Appendix A(app:a) contains details of the physical parameters of SRSPM used by the simulation models. The dynamics models are formulated and simulated in both Mathematica11.2 and C++. The simulations are allowed to run for a duration of 1 second on an AMD Ryzen 7 1800X, eight-core CPU[1] with 16 GB RAM and a clock speed of 3.6 GHz installed with Ubuntu 18.04.1 LTS.

[1]Note that only one core is used for simulating the free-fall scenario.

5.1. Comparison of coefficient matrices

The sizes[2] of the coefficient matrices reflect the density of the expressions contained within which has a direct implication on its compilation and execution time. The symbolic expressions for the coefficients are initially derived in Mathematica11.2. Later, the expressions are converted into C++ compatible code. Refer to the Appendix C(app:sim) for details. The sizes of the coefficient matrices $\boldsymbol{M}$ and $\boldsymbol{C}$ derived for the configuration-space model are given in Table 1. These coefficient matrices are functions of both passive and active joint variables.

Table 1: Size of the coefficient matrices used in configuration-space dynamics model of the manipulator

	Size (KB $\times 10^3$)	
Expression	C++	Mathematica
$\boldsymbol{M}$	0.900	13.369
$\boldsymbol{C}$	103.4	748.207

The matrices $\boldsymbol{M_\theta}$ and $\boldsymbol{C_\theta}$ cannot be obtained in closed-form as they involve expensive symbolic inverse computations. Therefore are computed numerically from the $\boldsymbol{M}$ and $\boldsymbol{C}$ matrices at each time step. An in-depth comparison of their relative time complexities can be found in [40](nag2019comparative). The sizes of relevant matrices used in the simulation of the actuator-space model are as shown in Table. 2(table:acttime).

Table 2: Size of the coefficient matrices used in dynamics modelled in the actuator-space of the manipulator

	Size (KB $\times 10^3$)	
Expression	C++	Mathematica
$\boldsymbol{J_{\eta\phi}}$	0.027	0.246
$\dot{\boldsymbol{J}}_{\boldsymbol{\eta\phi}}$	0.181	748.207
$\boldsymbol{J_{\eta\theta}}$	0.015	0.143
$\dot{\boldsymbol{J}}_{\boldsymbol{\eta\theta}}$	0.083	0.802

(table:acttime)

The inability in computing a closed-form expression for the $\boldsymbol{C_x}$ matrix using the method described in Section 3.3.1(sc:method1) becomes apparent from the sizes of the mass matrix and its partial derivatives given in Table 3(table:tasksize).

From Table 3(table:tasksize) and 1(table:configsize), it is evident that the latter has matrices of smaller size. Since the method given in Section 3.3.2(sc:method2) involves the numerical computation of the coefficient matrices from the coefficients of the actuator-space model, it is more practical

[2]Corresponds to the size of the expression computed using ByteCount function in Mathematica11.2 or the size

Table 3: size of the coefficient matrices used in dynamics modelled in the task-space of the manipulator

Expression	Size (KB $\times 10^3$)
$\boldsymbol{M}$	369.044
All the partial derivatives of $\boldsymbol{M}$	18531.400

to use this method. Further, the computation of solutions to the IK problem is simpler than that of the FK. Hence task-space models are easier to formulate and implement as compared to other formulations. Table 4(table:extconfsize) contains the sizes of the coefficient matrices of the extended-configuration-space model.

Table 4: Sizes of the coefficient matrices of extended-to-task-space dynamics model

	Size (KB)	
Expression	C++	Mathematica
$\boldsymbol{M}$	4.100	35.515
$\boldsymbol{C}$	15.500	100.984

From Table 1(table:configsize) and 4(table:extconfsize) the striking difference in the sizes of the coefficients in configuration-space and the extended-configuration-space is apparent. The sizes of the corresponding matrices differ at least by a magnitude of 10^3. This can be attributed to the sparsity of elements in the higher dimensional $\boldsymbol{M}$ and $\boldsymbol{C}$ matrices and are visualised using a matrix sparsity plot given in Fig. 6.

In Fig. 6, cells with "■" denote the presence of symbols, "■" denote the presence of constants and the "□" cells denote zero valued elements.

5.2. Comparison of computational time

As a pre-simulation step, all the formulated models are rewritten in state-space representation,

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}),$$

where $\boldsymbol{q}$ and $\boldsymbol{u}$ are variables correspond to states and actions of the system.

In Mathematica11.2 the equations of motion are integrated numerically with an Adams solver using the inbuilt NDSolve function, with the parameter, AccuracyGoal, set to 10^{-15}. More information regarding the simulation set up is given in Appendix C(app:sim). The time taken for free-fall

of the file containing the expression in C++

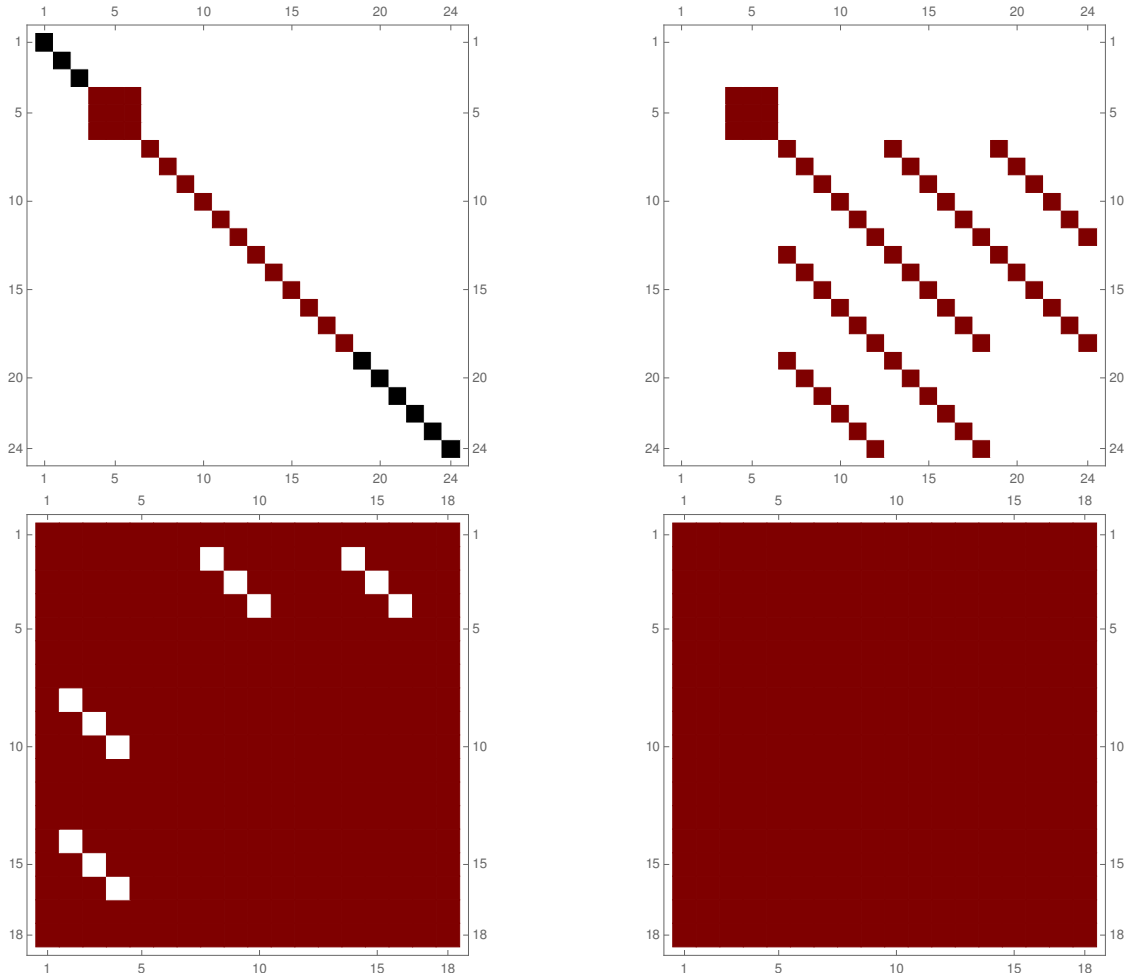
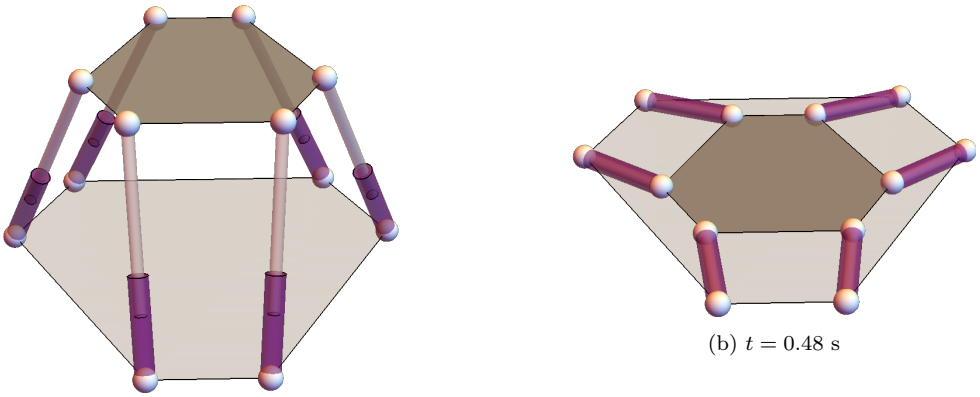


Figure 6: Visualising the sparsity of $\boldsymbol{M}$ and $\boldsymbol{C}$ matrices in the extended-configuration space vs. configuration-space formulations

simulation in each case is as shown in Table 5(time:mathematica).



(a) $t = 0$ s

(b) $t = 0.48$ s

(motion:act)

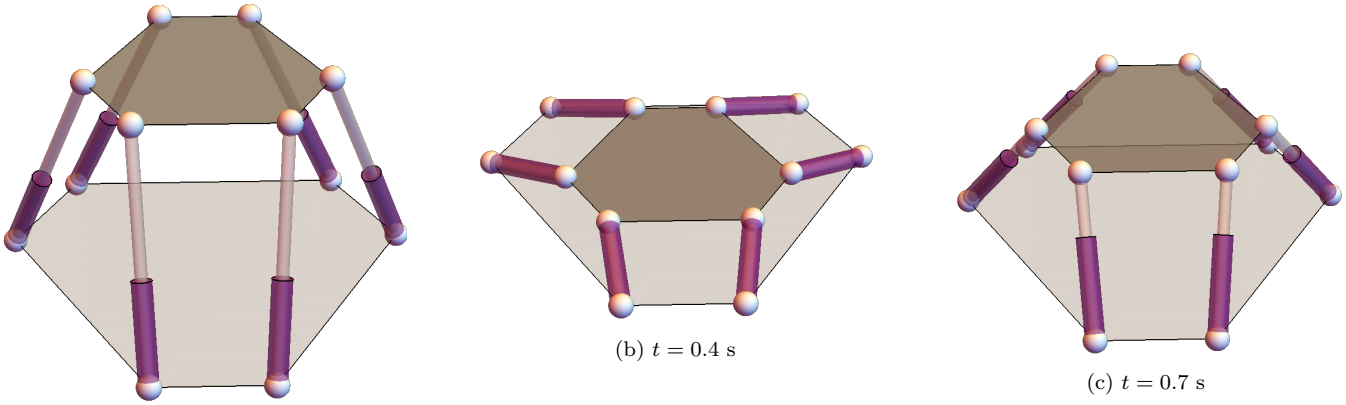
Figure 7: Motion of the manipulator when simulated in the actuator-space

As expected, the initial conditions, accuracy goal and the choice of numerical integrator have a significant effect on the simulation time and accuracy of the solutions obtained. During the simulation of actuator-space dynamics model, root-tracker fails to compute the FK solutions at around time $t = 0.48$ seconds, owing to the ill-conditioning of the $\boldsymbol{J}_{\boldsymbol{\eta\phi}}$ matrix corresponding to

Table 5: Computational time taken for the free-fall simulation in Mathematica11.2

Formulation	Real-time (s)	Simulation time (s)	Simulation time/Real-time (s)
Configuration-space	1.000	0.705	0.705
Actuator-space	0.480	0.337	0.717
Task-space	1.000	0.469	0.469
Extended-to-task-space	1.000	0.289	0.289
Extended-to-actuator-space	0.470	0.199	0.423

the achieved singular pose shown in Fig. 7(motion:act). Therefore, the simulation terminates at $t = 0.48$ seconds, whereas the other models are simulated for the entire duration of 1 second without encountering any singularity.



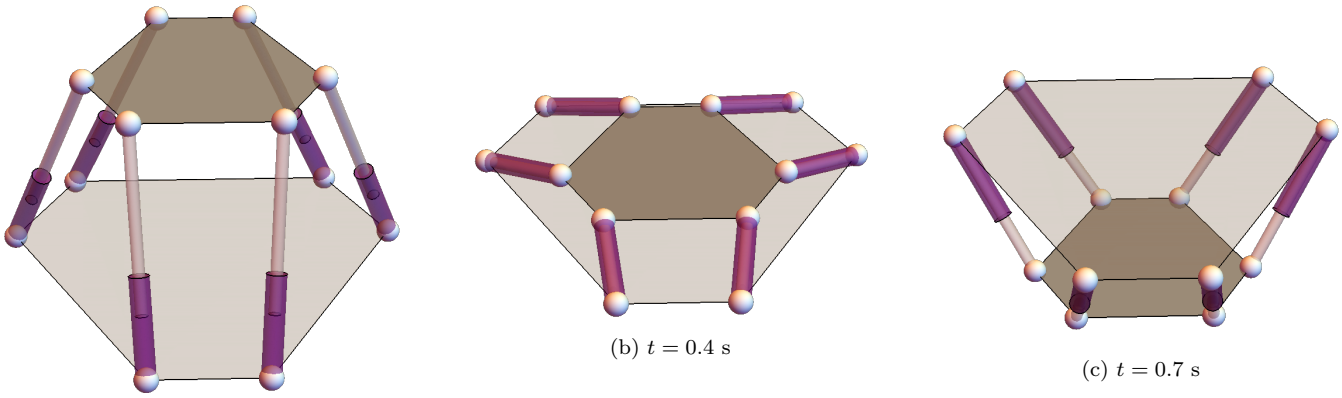
(a) $t = 0$ s (b) $t = 0.4$ s (c) $t = 0.7$ s

(motion:config)

Figure 8: Motion of the manipulator when simulated in the configuration-space

On the other hand, with the configuration-space model, at time $t = 0.48$ seconds, further motion is solely dependent on the solver, and the direction of movement cannot be precisely predicted. As shown in Fig. 8(motion:config). It is observed that the moving platform changes its direction upon reaching the singular pose. However, in the task-space simulation, the system does not encounter any singularity at time $t = 0.48$ seconds and proceeds usually, passing through the fixed platform, as depicted in Fig. 9(motion:etask). Detailed analysis of the behaviour of dynamics models at a singularity is elaborated in [22](MURALIDHARAN2018403).

Mathematica11.2 uses several optimisations beyond the control of the end-user and hence cannot be used as a benchmark for comparing different models. Therefore, all the dynamics models are ported to C++ benchmark them. The equations of motion rewritten in the state-space form are integrated using an explicit Runge-Kutta-Fehlberg method, an excellent general-purpose integrator. The duration of $t = 0$ seconds to $t = 1$ second is discretised into 100 equally spaced time-steps. For the actuator-space simulation, the accuracy goal of the custom Newton-Raphson based root-tracker is implemented, and the accuracy goal of the solutions is set to 10^{-15}. The details of the code and



(a) $t = 0$ s (b) $t = 0.4$ s (c) $t = 0.7$ s

(motion:etask)

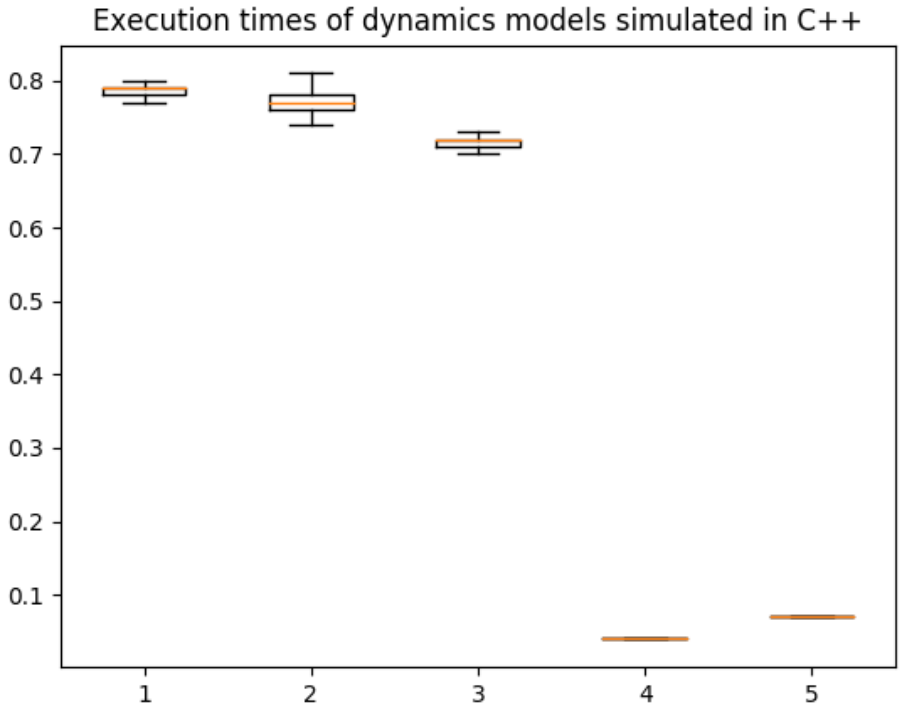
Figure 9: Motion of the manipulator when simulated in the extended-configuration-space

implementation can be found in the Github page of the project. The average time over 1000 runs per simulation is recorded and presented in Table 6(time:c++). Details regarding the compiler, compilation times of the codes, solvers libraries used can be found in the Appendix C(app:sim) [AS: These are from my laptop. Need to rerun codes on the lap PC]

Table 6: Computational time taken for the free-fall simulation in C++

Formulation	Real-time (s)	Simulation time (s)	Simulation time/Real-time (s)
Configuration-space	1.000	0.794	0.794
Actuator-space	0.480	0.777	1.619
Task-space	1.000	0.721	0.721
Extended-to-task-space	1.000	0.042	0.042
Extended-to-actuator-space	0.480	0.069	0.143

It is apparent from Table 6(time:c++) that the extended-to-task-space formulation takes the least amount of computational time. This reinforces the observations of the authors in [41](leger2007selection) that dynamics models with smaller coefficient matrices would not necessarily lead to faster simulation. In other words, even though the coefficient matrices in configuration-space are of size (18×18) as compared to extended-configuration-space, (24×24) size matrices, the later has a significantly lower computation time. The box plot corresponding to the data in Table 6(time:c++) is shown in Fig. 10(fig:boxplot).



(fig:boxplot)

Figure 10: Box plot corresponding to the execution times of various dynamics formulations

The following are the measures taken to prevent evaluation of redundant code, ensure fast computation, code modularisation.

1. Compilation of the entire 18×18 dimensional $\boldsymbol{C}$ matrix is not feasible (with the computer used), and hence each element of the matrix is extracted as a separate file and is compiled independently. All the individual elements are evaluated, and the $\boldsymbol{C}$ matrix is reconstructed directly during runtime.
2. The coefficient matrices are functions of repetitive sine and cosine of the generalised coordinates. It is advised to pre-compute the values of the trigonometric functions to obtain a significant reduction in the computational time.
3. For the computation of the IK solutions, a cascaded substitution is done to prevent recomputing repetitive expressions.

Cascaded substitution for computing the IK solutions

1: The link lengths, $\boldsymbol{l}$ given the task-space variable data, $\boldsymbol{x}$ are computed in the first step.
2: The values of $\boldsymbol{\psi}$ given the link lengths, $\boldsymbol{l}$, and the task space data, $\boldsymbol{x}$ are computed next.
3: Finally, the variables $\boldsymbol{\phi}$ are evaluated using $\boldsymbol{x}$ and $\boldsymbol{\psi}$ values calculated in the above steps.

---

4. The matrix inversion operation is usually computationally more expensive than solving a linear equation. The same is true for all the inversions of the Jacobians and for computing

the second-order derivatives of the generalised coordinates, $\ddot{\boldsymbol{q}}_e$. However, the usage of the LinearSolve function in Mathematica11.2 instead of Inverse function has not shown any significant improvement in performance. The same observation remains valid for the simulations performed in C++. Factors like the size of the system, libraries and compilers used might affect the computational time.

5.3. Comparison of model accuracy

(sc:checks) Dynamical simulations are prone to deviate from the systems' actual evolution due to incorrect modelling, truncation errors, approximations in the numerical method used etc. Muralidharan et. al. [22](MURALIDHARAN2018403) have proposed numerical error measures to validate the mathematical consistency of the formulated dynamics model of a parallel manipulator by considering the norm of zeroth, first and second-order forms of the constraint equations. Instead of the norm, the absolute maximum of the error vector is used in this paper. The error measures generated using the data obtained from the free-fall simulation of the configuration-space model are described in this section.

1. Zeroth-order check on the loop-closure constraint (e_1) :
   This measure, shown in Fig. 11(e1plot), quantifies the deviation from the loop-closure equations, Eq. (1) with time,

$$e_1 = \max(|\boldsymbol{\eta}(\boldsymbol{q})|). \tag{26}$$

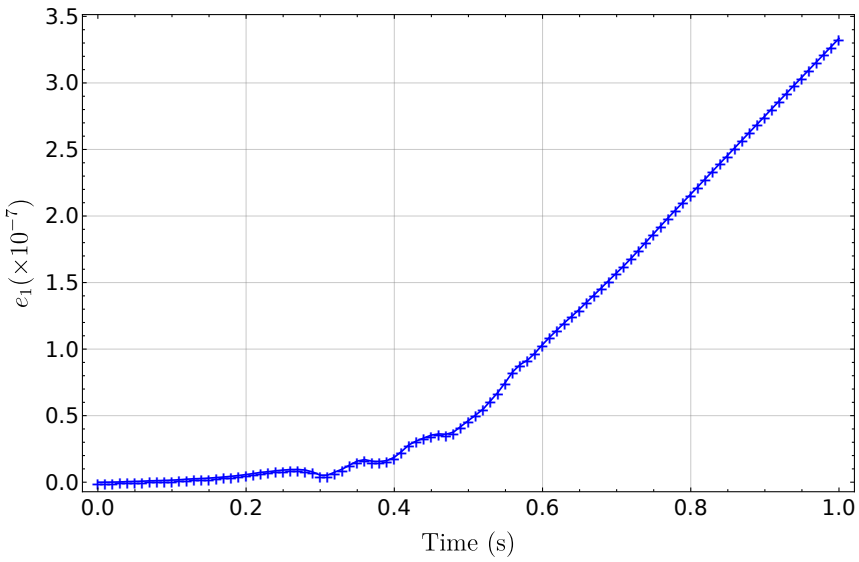


Figure 11: Absolute maximum error in loop-closure equations with time as given in Eq. (26)

2. First-order check on loop-closure constraint (e_2) :

   This verifies the consistency of the joint velocities with the loop-closure constraint in its Pfaffian form and is shown in Fig. 12(e2plot). Deviation from the first order derivative of the loop-closure constraints with respect to time,

$$e_2 = \max(|\boldsymbol{J}_{\boldsymbol{\eta q}}\dot{\boldsymbol{q}}|). \tag{27}$$

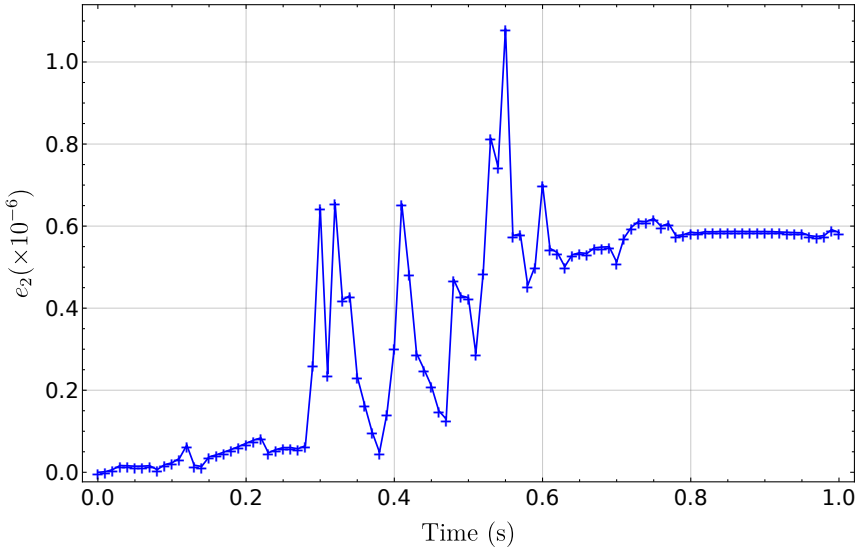


Figure 12: Absolute maximum of first order loop-closure equations with time as given in Eq. (27)

3. Second-order check on the loop-closure constraint (e_3) :

   The consistency of the generalised accelerations with the loop-closure constraint is verified by its second order derivative with respect to time and is shown in Fig. 13(e3plot).

$$e_3 = \max(|\boldsymbol{J}_{\boldsymbol{\eta q}}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{\boldsymbol{\eta q}}\dot{\boldsymbol{q}}|). \tag{28}$$

4. Conservation of energy check (en) :

   The total mechanical energy of the manipulator, $E(t)$ is given as,

$$E(t) = T(t) + V(t), \tag{29}$$

   where $T(t)$ and $V(t)$ are values of the kinetic and potential energy of the system at time t. The total energy of the system is expected to remain constant during the free-fall simulation.

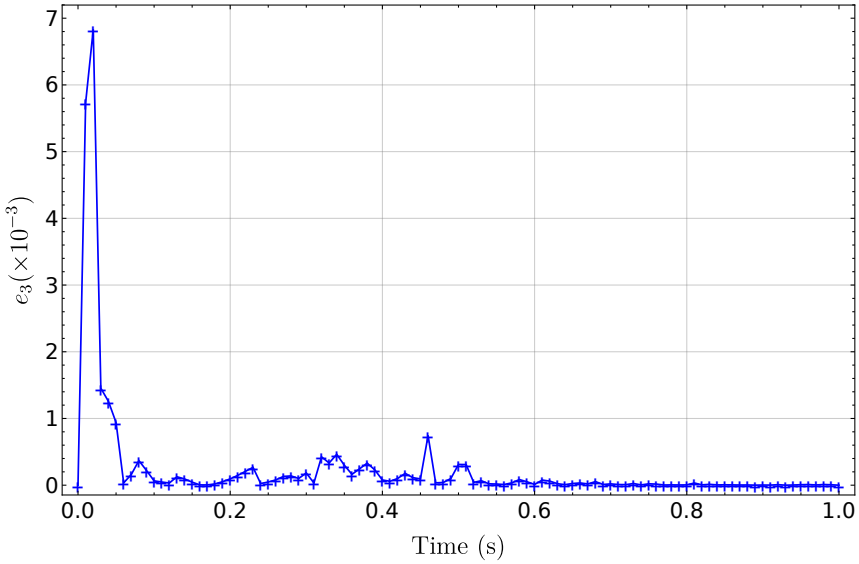


Figure 13: Absolute maximum of second order loop-closure equations with time as given in Eq. (28)

The percentage change in energy of the system is checked and is given as:

$$\delta E = \frac{E(t) - E(0)}{E(0)} \times 100,$$

where $E(0)$ is the total mechanical energy of the system at time $t = 0$ and is shown in Fig. 14(enplot).

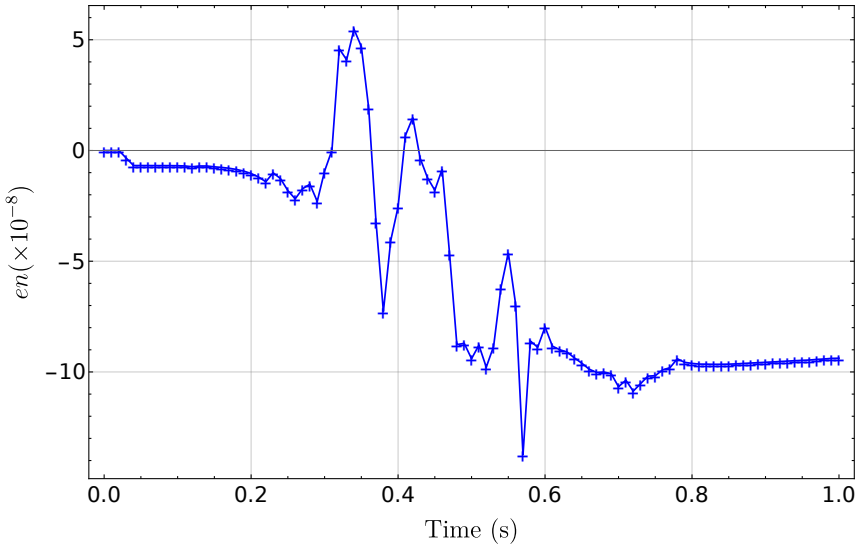


Figure 14: Percentage change in the total mechanical energy of the system with time as given in Eq. (29)

Similarly, all the dynamics models are validated for mathematical consistency, and the magnitude of all the errors are tabulated in Table 7(table:errortable).

From the values presented in 7(table:errortable), it is apparent that the order of error e_1 in the configuration-space model is 10^8 times that of the actuator-space model. The imposition of the loop-closure equations via root-tracker results in the generation of errors of smaller magnitude in

Table 7: Order of the magnitude of various error measures for different formulations

Formulation	e_1	e_2	e_3	en
Configuration-space	10^{-7}	10^{-6}	10^{-4}	10^{-8}
Actuator-space	10^{-15}	10^{-15}	10^{-13}	10^{-7}
Task-space	10^{-15}	10^{-14}	10^{-13}	10^{-6}
Extended-mapped-to-task-space	10^{-16}	10^{-21}	10^{-20}	10^{-8}
Extended-mapped-to-actuator-space	10^{-16}	10^{-15}	10^{-13}	10^{-7}

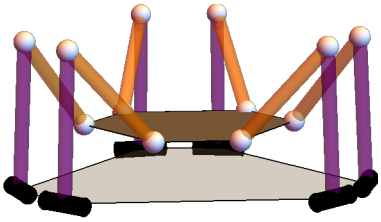
actuator-space. However, on the other hand, the extended-to-task-space formulation involves the computation of closed-form expressions corresponding to the IK solutions, which result in lower magnitudes of errors as shown in 7(table:errortable). The error e_1 is of the order 10^{-16} in the extended-to-task-space model as compared to 10^{-15} in the actuator-space. Hence the extended-configuration-space formulation is not only faster but also more precise than the conventional formulations. Therefore to obtain the best performance, the extended-to-task-space model should be chosen, and the joint torque requirements can be mapped to the task-space generalised forces.

5.4. A note on the mathematical consistency checks

The checks mentioned above ensure only the mathematically consistency of the formulations and do not necessarily validate the system modelled to being the actual system. Familiar sources for such discrepancy are a mismatch in the inertia or mass of elements modelled versus the physical system. These checks also fail to identify any representation errors. Say, if the local frame of the moving platform of the SRSPM is defined with a constant offset in its orientation from the global frame, in such cases, even if the simulation set-up is not consistent with the physical representation, the model remains mathematically consistent. In other words, modelling errors committed before the formulation of mass matrix continue being mathematically consistent and cannot be prevented by the described checks.

6. Dynamics formulation and simulation of **6**-RSS manipulator

(sc:rss) This section demonstrates the utility of having a fast and accurate Lagrangian dynamics model. The 6-RSS manipulator is a 6 degree-of-freedom parallel robot similar to the SRSPM. Its architecture is shown in Fig. 15(fig:rssman). Since the construction of moving and fixed platforms of the SRSPM and 6-RSS manipulator are identical, the axes conventions remain the same. A rotary joint is present at each of the vertices of the fixed platform with its central axis tangential to the circumcircle of the fixed platform. The x-axis is aligned with the rotary joint axis and z-axis



(fig:rssman)

Figure 15: The 6-RSS manipulator

along the links' length. Similar to the SRSPM, the set of joint and task-space variables together constitute the extended-configuration-space of the 6-RSS manipulator.

$$\boldsymbol{q}_e = [\boldsymbol{x}^\top, \boldsymbol{\phi}^\top, \boldsymbol{\theta}^\top]^\top,$$

where $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ and $\boldsymbol{x}$ are the active, passive and task-space variables respectively. A mathematical model is developed following the process as provided in Section 4(sc:extconf). The details of the numerical checks described in Section 5.3(sc:checks) are given in Fig. 17(rss:check). The forward dynamics for 0.6 seconds free-fall simulation of the top platform implemented in C++ is completed in 0.546 seconds by the extended-to-task-space dynamics model.

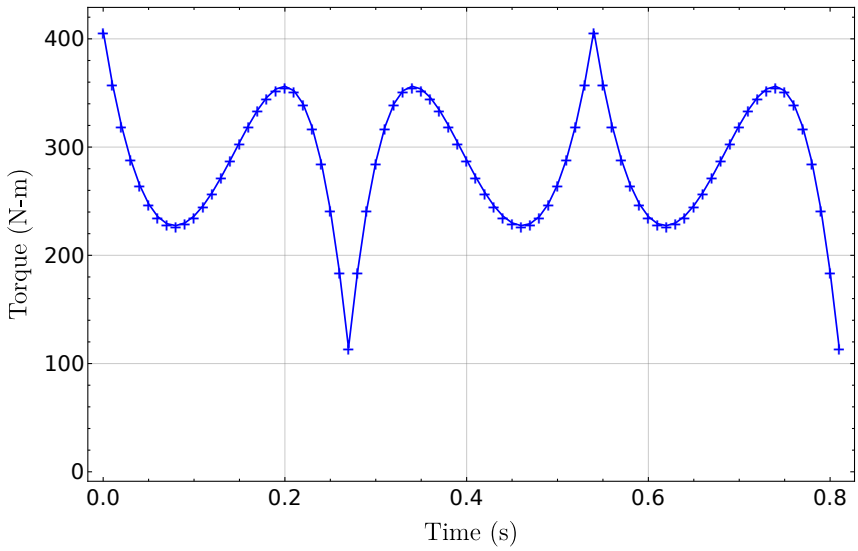
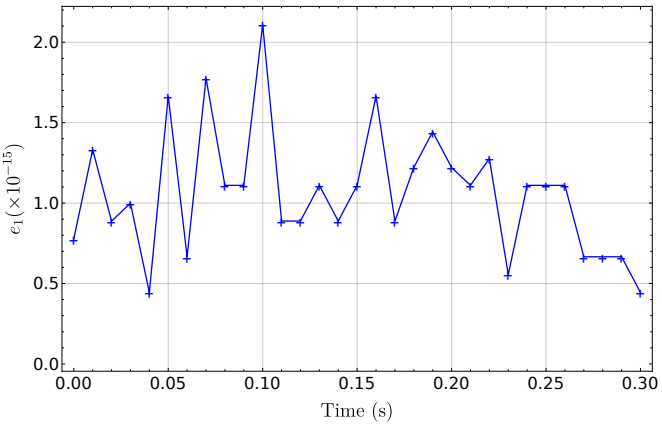
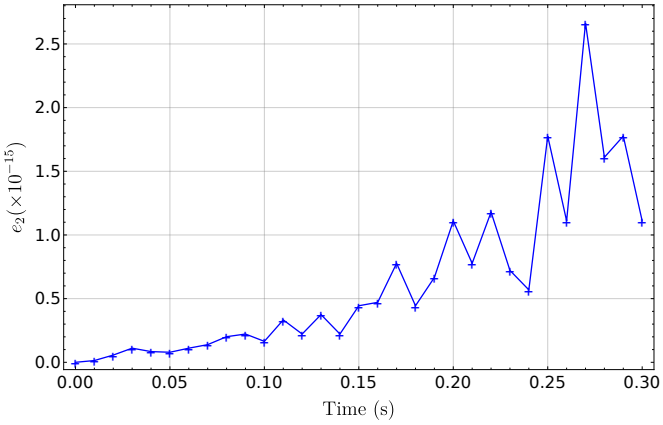


Figure 16: Torque profile for the heave motion requirement of the moving platform

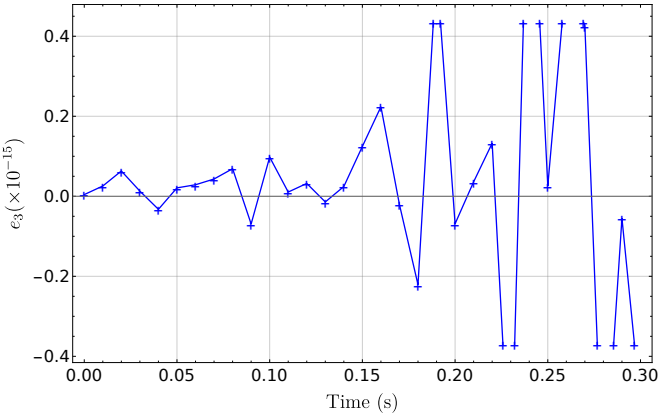
This example is presented to illustrate the role of the dynamics model in decision making of design specifications. The moving platform of the manipulator is required to perform a prescribed heave motion. Inverse dynamics analysis is performed to compute the torque requirements to select the compatible motors.



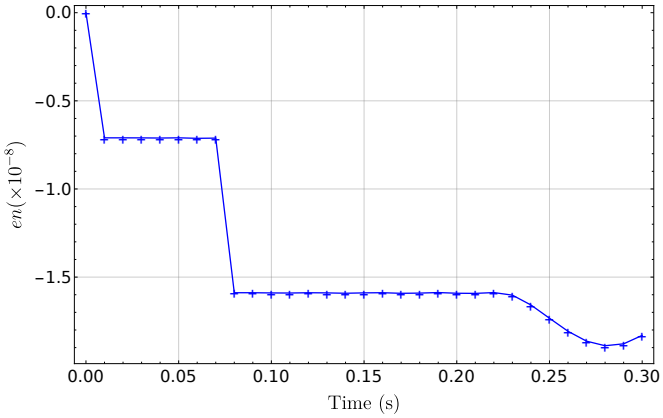
(a) Absolute maximum error in loop-closure equations with time



(b) Absolute maximum of first order loop-closure equations with time



(c) Absolute maximum of first order loop-closure equations with time



(d) Percentage change in the total mechanical energy of the system with time as given in Eq.(29)

Figure 17: Validation of the extended-to-actuator-space mathematical model

6.1. Inverse dynamics simulation for a required heave motion

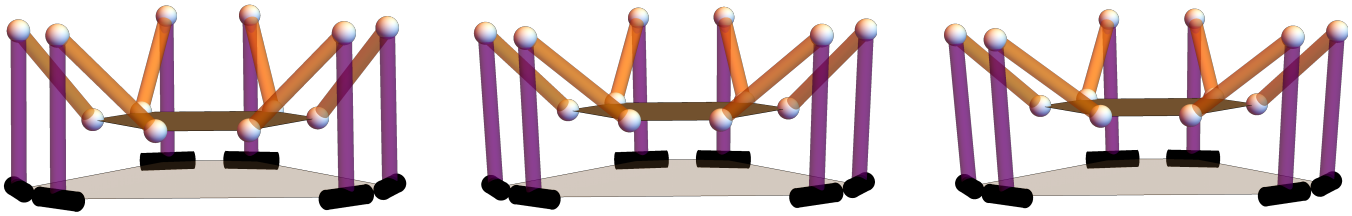
The moving platform is required to produce a heave motion with constraints on its peak acceleration as $0.8g\ m/s^2$ in the z direction, with the following terminal conditions,

$$\boldsymbol{x}_i = [0, 0, 0.3, 0, 0, 0]^\top \quad \boldsymbol{x}'_i = [0, 0, 0, 0, 0, 0]^\top$$
$$\boldsymbol{x}_f = [0, 0, 0.4, 0, 0, 0]^\top \quad \boldsymbol{x}'_f = [0, 0, 0, 0, 0, 0]^\top,$$

where $\boldsymbol{x}_i, \boldsymbol{x}_f$ are the initial and final poses and $\dot{\boldsymbol{x}}_i, \dot{\boldsymbol{x}}_f$ are the initial and final velocities of the moving platform given in task-space coordinates. The extended-to-actuator-space model is used for the inverse dynamics (ID) simulations to compute the motor torques. Since the requirements are the torques of the motors, the model is mapped to actuator-space; the same formulation can also be mapped to task-space if required. The manipulator takes around 0.168 s to complete the ID simulation for a real-time of 0.3 s. The obtained torque profile is shown in Fig. 16(fig:heave). The manipulator parameters used for the simulation are given in Appendix A(app:a). The torque profile shows that for the achieved configurations during the motion, the motor always supplies the positive

torque. Since the initial and final poses have a zero velocity constraint, the manipulator decelerates as it reaches these positions and hence a peak is observed in the torque profile. A small segment of the manipulator moving up is shown in the motion profile given in Fig. 18(motionrss). The Fig. 16(fig:heave) establishes theoretical bounds on the motor torque requirement for the assumed task providing essential information in making design decisions.

[AS: Impose a sine wave corresponding to the path and then put manipulator on it to illustrate the heave motion]



(a) $t = 0.1$ s (b) $t = 0.2$ s (c) $t = 0.3$ s

Figure 18: Configurations of the manipulator during heave motion

7. Conclusion

(sc:conc) A detailed study of the extended-configuration-space dynamics model is presented in this paper. The versatility of the model through transformation between dynamic models is demonstrated. The idea of exploiting a sparse system while keeping the size of the problem small is also illustrated. Numerical studies are presented to compare the proposed model with-space, actuator-space and task-space models. The extended-configuration-space models are not only computationally efficient but also accurate among all the models.

Fast and accurate Lagrangian dynamics models are crucial for designing complex non-linear controllers. Especially, fast dynamics formulations are essential in high-frequency applications. Moreover, the model is used for deciding design specification for a prescribed heave motion of the 6-RSS manipulator. The proposed dynamics model opens several possibilities in design optimisation with dynamics considerations and real-time model predictive control of parallel manipulators.

8. Acknowledgements

[AS: TBD] [AS: The appendix and the references are to be rewritten]

Appendix A. Physical parameters of the manipulators

(app:a)

Table A.8: Parameters of the SRSPM used to solve the FK problem

Parameter	Symbol	Value	Unit
Circumradius of the base platform	r_b	1	m
Circumradius of the moving platform	r_t	0.5803	m
Angular spacing between the adjacent pair of legs of the fixed platform	γ_b	0.2985	rad
Angular spacing between the adjacent pair of legs of the moving platform	γ_t	0.6573	rad
Length of the sliding link of the prismatic joint	$l_{bi}, i = (1, \ldots, 6)$	0.5	m
Length of the fixed link of the prismatic joint	$l_{ai}, i = (1, \ldots, 6)$	1.5	m

Table A.9: Mechanical parameters of the 6-RSS manipulator used in the simulation of the dynamics model

Parameter	Value
r_b	1 (m)
r_t	0.6 (m)
γ_b	0.8080 (rad)
γ_t	0.4040 (rad)
I_{tpxx}	1.1717 (kgm^2)
I_{tpyy}	1.1717 (kgm^2)
I_{tpzz}	2.3431 (kgm^2)
I_{lxx}	0.0283 (kgm^2)
I_{lyy}	0.0283 (kgm^2)
I_{lzz}	6.325×10^{-6} (kgm^2)
I_{rxx}	0.1764 (kgm^2)
I_{ryy}	0.1764 (kgm^2)
I_{rzz}	$5.400\times10^{-6}(kgm^2)$
m_p	200.2033 (kg)
$m_{li}, i = (1, \ldots, 6)$	0.5060 (kg)
$m_{ri}, i = (1, \ldots, 6)$	0.4320 (kg)
$l_{li}, i = (1, \ldots, 6)$	0.8200 (m)
$l_{ri}, i = (1, \ldots, 6)$	0.7000 (m)

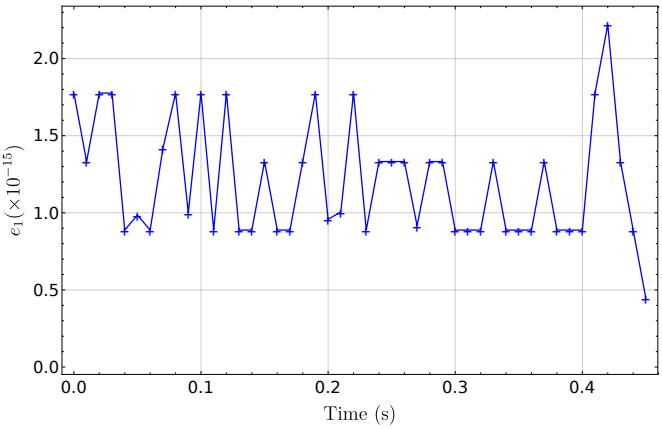
(tab:rssmanipulator)

Appendix B. Numerical verification of various dynamics models

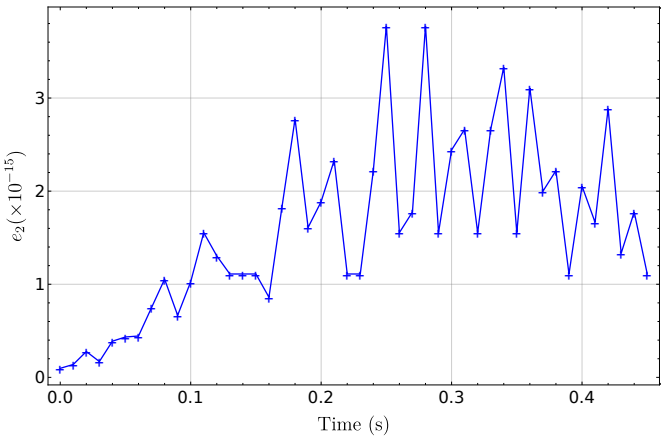
(app:checks)

Appendix B.1. Actuator-space model verification

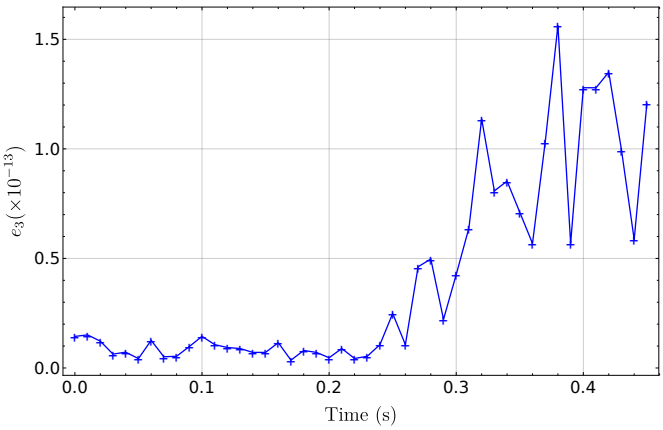
The errors corresponding to the verification of the mathematical dynamics model are presented in Fig. B.19(act:check)



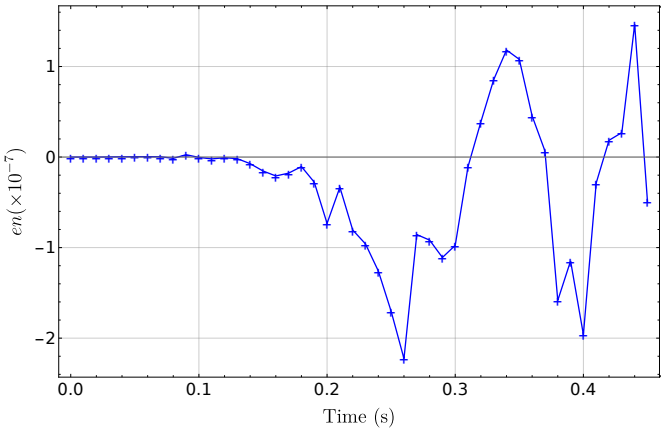
(a) Absolute maximum error in loop-closure equations with time



(b) Absolute maximum of first order loop-closure equations with time



(c) Absolute maximum of first order loop-closure equations with time

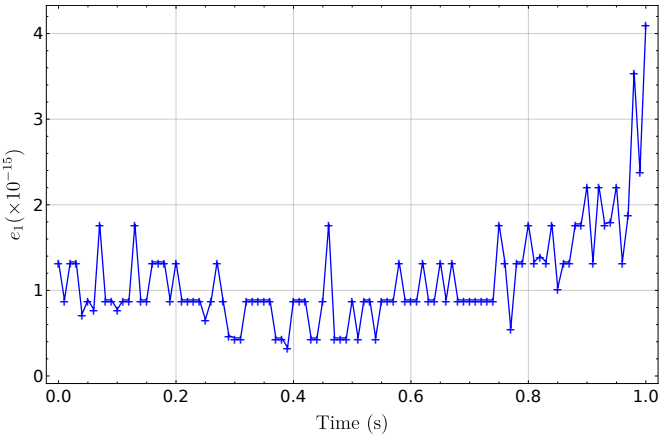


(d) Percentage change in the total mechanical energy of the system with time as given in Eq.(29)

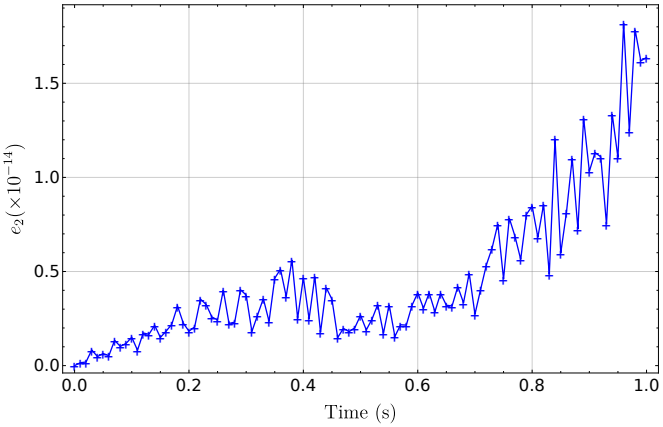
Figure B.19: Validation of the actuator-space dynamics model

Appendix B.2. Task-space model verification

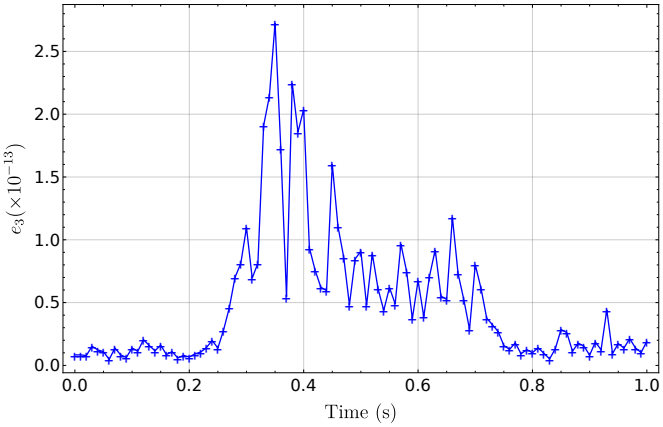
The errors corresponding to the verification of the mathematical dynamics model are presented in Fig. B.20(task:check)



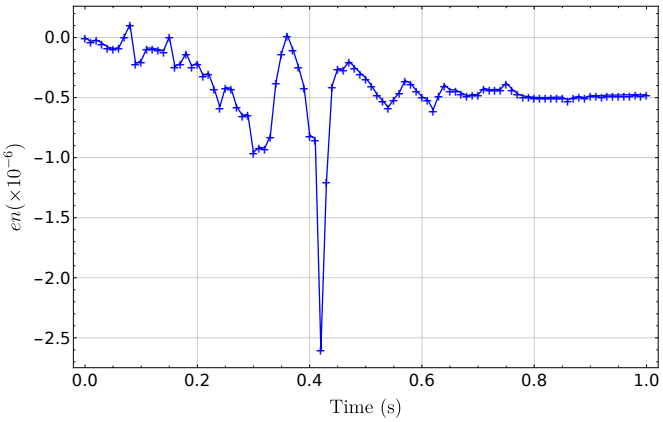
(a) Absolute maximum error in loop-closure equations with time



(b) Absolute maximum of first order loop-closure equations with time



(c) Absolute maximum of first order loop-closure equations with time

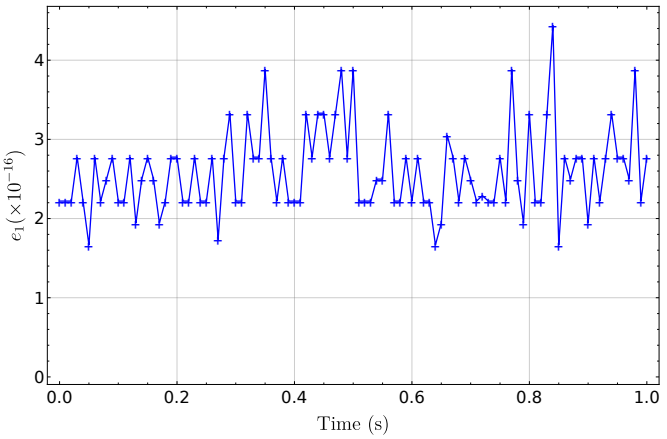


(d) Percentage change in the total mechanical energy of the system with time as given in Eq.(29)

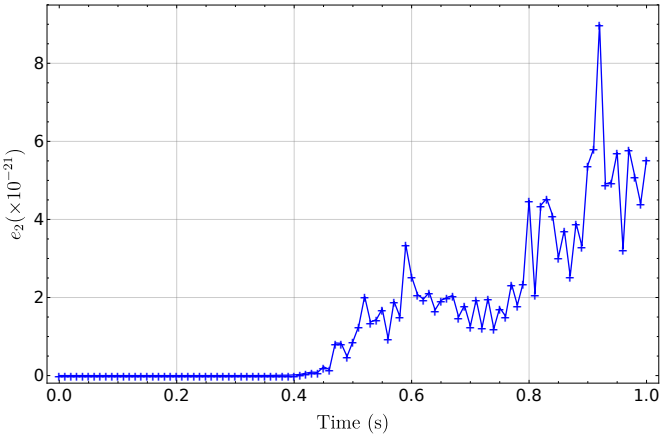
Figure B.20: Validation of the task-space dynamics model

Appendix B.3. Extended-configuration mapped to task-space model verification

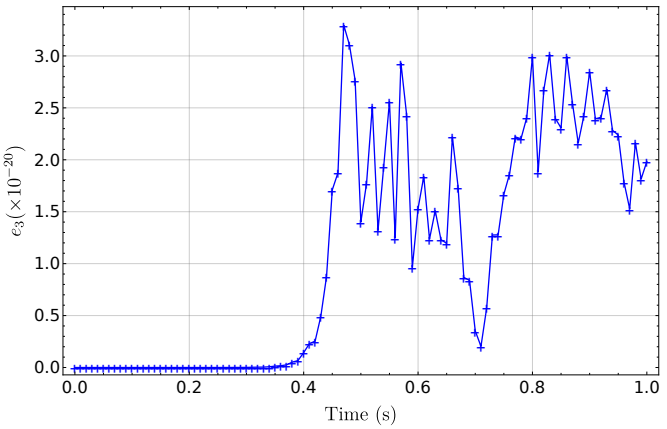
The errors corresponding to the verification of the mathematical dynamics model are presented in Fig. B.21(exttask:check)



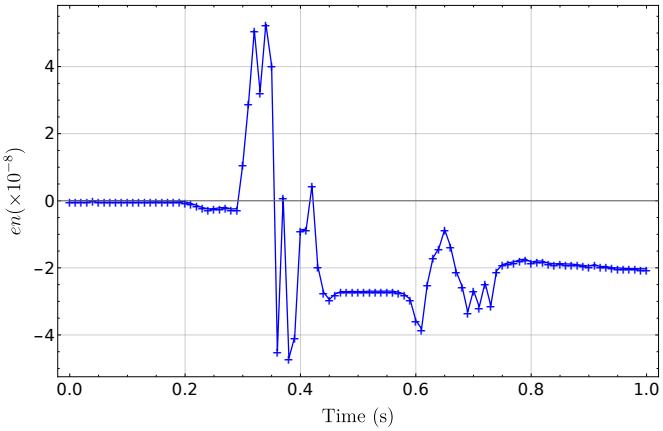
(a) Absolute maximum error in loop-closure equations with time



(b) Absolute maximum of first order loop-closure equations with time



(c) Absolute maximum of first order loop-closure equations with time

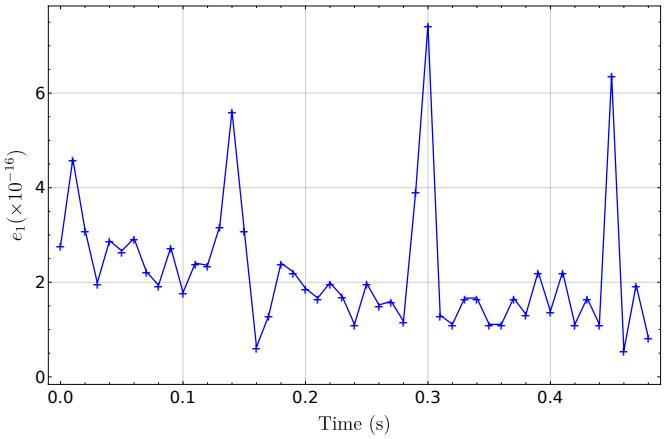


(d) Percentage change in the total mechanical energy of the system with time as given in Eq.(29)

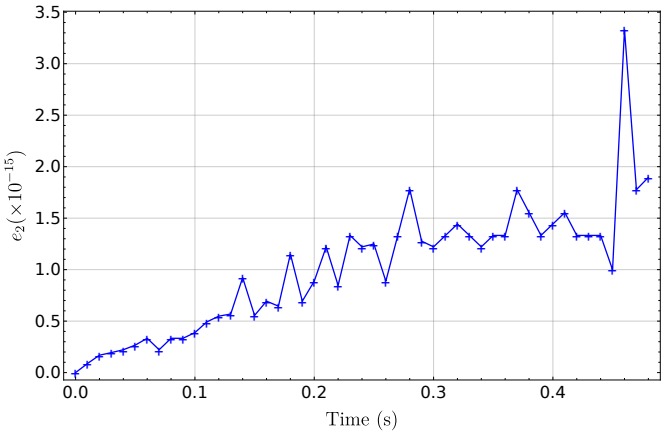
Figure B.21: Validation of the extended-to-task-space dynamics model

Appendix B.4. Extended-configuration mapped to actuator-space model verification

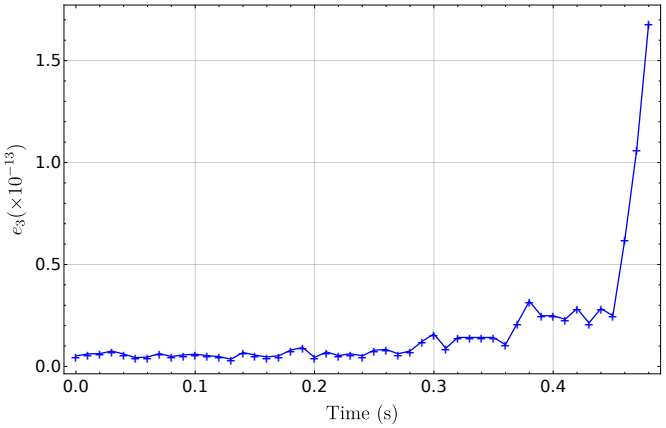
The errors corresponding to the verification of the mathematical dynamics model are presented in Fig. B.22(extact:check)



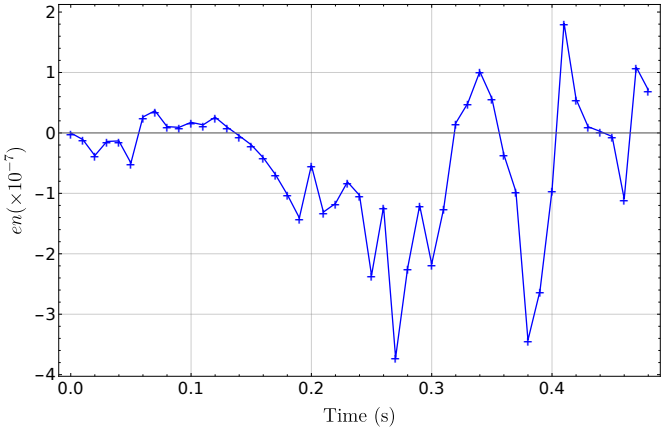
(a) Absolute maximum error in loop-closure equations with time



(b) Absolute maximum of first order loop-closure equations with time



(c) Absolute maximum of first order loop-closure equations with time



(d) Percentage change in the total mechanical energy of the system with time as given in Eq.(29)

Figure B.22: Validation of the extended-to-actuator-space dynamics model

Appendix C. Additional information for execution of simulations

(app:sim)

This section presents the implementation details for the simulations performed in both C++ and Mathematica11.2.

Appendix C.1. Porting files from Mathematica11.2 to C++

FileTemplateApply and CForm functions

Appendix C.2. Simulation in Mathematica11.2

The average time to solve the differential equations in Mathematica11.2 is recorded by using the RepeatedTiming function. The Compile command is used to convert all of the coefficient matrices

into compiled functions, which optimises all the expressions for real-valued input, cutting down their evaluation time significantly, from the order of days to seconds.

Appendix C.3. Simulation in C++

[AS: Appendix contains information about the solvers used. fsl_odeiv2_step_rkf45].

[AS: Begin appendix] Linear algebra package Eigen [42](eigenweb), is used for matrix manipulations. Numerical solvers of GSL, GNU Scientific Library [43](Gough:2009:GSL:1538674) are used for numerical integration and solving linear equations. After experimenting with a few compilers, it is found that the clang++ [44](LLVM:CGO04) compiler has an advantage in compilation time, reducing the testing time. However, its execution time is almost the same as the g++ compiler [45](stallman2009 Refer to [AS: Appendix] for more details on various compilers tested. Execution time is further improved by using code optimisation flags during compilation. Comparison of multiple compiler flags and their corresponding performance can be found in the [AS: Appendix]. As expected, the maximum reduction in execution times is obtained with the -Ofast optimisation flag which enables several optimisations including -ffast-math. [AS: End appendix] [AS: Appendix: The GNU solver, gsl_linalg_complex_LU_solve is used for solving linear equations].

References

[1] V. Damic, M. Cohodar, Dynamic analysis of Stewart platform by bond graphs, Procedia Engineering 100 (2015) 226–233.

[2] M.-J. Liu, C.-X. Li, C.-N. Li, Dynamics analysis of the Gough-Stewart platform manipulator, IEEE Transactions on Robotics and Automation 16 (1) (2000) 94–98.

[3] L.-W. Tsai, Solving the inverse dynamics of a Stewart-Gough manipulator by the principle of virtual work, Journal of Mechanical Design 122 (1) (2000) 3–9.

[4] J. Wang, C. M. Gosselin, A new approach for the dynamic analysis of parallel manipulators, Multibody System Dynamics 2 (3) (1998) 317–334.

[5] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033.

[6] S. K. Saha, Dynamics of Serial Multibody Systems Using the Decoupled Natural Orthogonal Complement Matrices, Journal of Applied Mechanics 66 (4) (1999)

986–996. arXiv:https://asmedigitalcollection.asme.org/appliedmechanics/article-pdf/66/4/986/5466543/986_1.pdf, doi:10.1115/1.2791809.
URL https://doi.org/10.1115/1.2791809

[7] P. Choudhury, A. Ghosal, Singularity and controllability analysis of parallel manipulators and closed-loop mechanisms, Mechanism and machine theory 35 (10) (2000) 1455–1479.

[8] R. M. Murray, Nonlinear control of mechanical systems: A lagrangian perspective, IFAC Proceedings Volumes 28 (14) (1995) 349–360.

[9] A. D. Lewis, Is it worth learning differential geometric methods for modeling and control of mechanical systems?, Robotica 25 (6) (2007) 765–777.

[10] E. F. Fichter, A Stewart platform-based manipulator: general theory and practical construction, The International Journal of Robotics Research 5 (2) (1986) 157–182.

[11] Z. Ji, Study of the effect of leg inertia in Stewart platforms, in: IEEE International Conference on Robotics and Automation, IEEE, 1993, pp. 121–126.

[12] D. Li, S. Salcudean, Modeling, simulation, and control of a hydraulic Stewart platform, in: IEEE International Conference on Robotics and Automation, Vol. 4, IEEE, 1997, pp. 3360–3366.

[13] I. Davliakos, E. Papadopoulos, Model-based control of a 6-dof electrohydraulic Stewart-Gough platform, Mechanism and machine theory 43 (11) (2008) 1385–1400.

[14] S.-H. Lee, J.-B. Song, W.-C. Choi, D. Hong, Position control of a Stewart platform using inverse dynamics control with approximate dynamics, Mechatronics 13 (6) (2003) 605–619.

[15] C. C. Nguyen, S. S. Antrazi, Z.-L. Zhou, C. E. Campbell Jr, Adaptive control of a Stewart platform-based manipulator, Journal of Robotic systems 10 (5) (1993) 657–687.

[16] Y.-L. Kuo, T.-P. Lin, C. Y. Wu, Experimental and numerical study on the semi-closed loop control of a planar parallel robot manipulator, Mathematical Problems in Engineering 2014 (2014).

[17] B. Dasgupta, T. Mruthyunjaya, Closed-form dynamic equations of the general Stewart platform through the Newton–Euler approach, Mechanism and machine theory 33 (7) (1998) 993–1012.

[18] Z. Geng, L. S. Haynes, J. D. Lee, R. L. Carroll, On the dynamic model and kinematic analysis of a class of Stewart platforms, Robotics and autonomous systems 9 (4) (1992) 237–254.

[19] K. Liu, F. Lewis, G. Lebret, D. Taylor, The singularities and dynamics of a Stewart platform manipulator, Journal of Intelligent and Robotic Systems 8 (3) (1993) 287–308.

[20] G. Lebret, K. Liu, F. L. Lewis, Dynamic analysis and control of a Stewart platform manipulator, Journal of Robotic Systems 10 (5) (1993) 629–655.

[21] B. Dasgupta, P. Choudhury, A general strategy based on the Newton-Euler approach for the dynamic formulation of parallel manipulators, Mechanism and machine theory 34 (6) (1999) 801–824.

[22] V. Muralidharan, T. K. Mamidi, S. Guptasarma, A. Nag, S. Bandyopadhyay, A comparative study of the configuration-space and actuator-space formulations of the lagrangian dynamics of parallel manipulators and the effects of kinematic singularities on these, Mechanism and Machine Theory 130 (2018) 403 – 434.

[23] M. K. Karnam, A. Baskar, R. A. Srivatsan, S. Bandyopadhyay, Computation of the safe working zones of planar and spatial parallel manipulators, Robotica 38 (5) (2020) 861–885. doi:10.1017/S0263574719001139.

[24] R. Ryan, Adams—multibody system analysis software, in: Multibody systems handbook, Springer, 1990, pp. 361–402.

[25] C. Franchi, A highly redundant coordinate formulation for constrained rigid bodies, Meccanica 30 (1) (1995) 17–35.

[26] P. Masarati, M. Morandini, P. Mantegazza, An efficient formulation for general-purpose multibody/multiphysics analysis, Journal of Computational and Nonlinear Dynamics 9 (4) (2014).

[27] A. Fumagalli, P. Masarati, Real-time inverse dynamics control of parallel manipulators using general-purpose multibody software, Multibody System Dynamics 22 (1) (2009) 47–68.

[28] F. E. Udwadia, R. E. Kalaba, Analytical dynamics: a new approach, Cambridge University Press, 2007.

[29] D. Stewart, A platform with six degrees of freedom, Proceedings of the Institution of Mechanical Engineers 180 (1) (1965) 371–386.

[30] J. Freeman, G. Watson, Y. Papelis, T. Lin, A. Tayyab, R. Romano, J. Kuhl, The iowa driving simulator: An implementation and application overview, Tech. rep., SAE Technical Paper (1995).

[31] M. K. Park, M. C. Lee, K. S. Yoo, K. Son, W. S. Yoo, M. C. Han, Development of the pnu vehicle driving simulator and its performance evaluation, in: IEEE International Conference on Robotics and Automation, Vol. 3, IEEE, 2001, pp. 2325–2330.

[32] J. Pradipta, M. Klünder, M. Weickgenannt, O. Sawodny, Development of a pneumatically driven flight simulator Stewart platform using motion and force control, in: IEEE/ASME International Conference on Advanced Intelligent Mechatronics, IEEE, 2013, pp. 158–163.

[33] A. Ghosal, Robotics: Fundamental Concepts and Analysis, Oxford University Press, New Delhi, 2006.

[34] S. Bandyopadhyay, A. Ghosal, Geometric characterization and parametric representation of the singularity manifold of a 6–6 Stewart platform manipulator, Mechanism and Machine Theory 41 (11) (2006) 1377–1400.

[35] S. M. Anwar, S. Bandyopadhyay, Trajectory-tracking control of semi-regular Stewart platform manipulator, in: Proceedings of the 15th National Conference on Machines and Mechanisms,(NaCoMM 2011), 2012, pp. 446–454.

[36] C. Nasa, S. Bandyopadhyay, Trajectory-tracking control of a planar 3-$\underline{\text{R}}$RR parallel manipulator with singularity avoidance, in: 13th World Congress in Mechanism and Machine Science, Guanajuato, Mexico, 2011, pp. 19–25.

[37] A. Agarwal, C. Nasa, S. Bandyopadhyay, Dynamic singularity avoidance for parallel manipulators using a task-priority based control scheme, Mechanism and Machine Theory 96 (2016) 107–126.

[38] N.-I. Kim, C.-W. Lee, High speed tracking control of Stewart platform manipulator via enhanced sliding mode control, in: IEEE International Conference on Robotics and Automation, Vol. 3, IEEE, 1998, pp. 2716–2721.

[39] Y. X. Su, B. Y. Duan, C. H. Zheng, Y. Zhang, G. Chen, J. Mi, Disturbance-rejection high-precision motion control of a Stewart platform, IEEE Transactions on Control Systems Technology 12 (3) (2004) 364–374.

[40] A. Nag, S. Bandyopadhyay, A comparative study of the Configuration-Space and Actuator-Space forward dynamics of closed-loop mechanisms using the Lagrangian formalism, in: Machines, Mechanism and Robotics, Springer, 2019, pp. 95–106.

[41] M. Léger, J. McPhee, Selection of modeling coordinates for forward dynamic multibody simulations, Multibody System Dynamics 18 (2) (2007) 277–297.

[42] G. Guennebaud, B. Jacob, et al., Eigen v3, http://eigen.tuxfamily.org (2010).

[43] B. Gough, GNU Scientific Library Reference Manual - Third Edition, 3rd Edition, Network Theory Ltd., 2009.

[44] C. Lattner, V. Adve, LLVM: A compilation framework for lifelong program analysis and transformation, in: CGO, San Jose, CA, USA, 2004, pp. 75–88.

[45] R. M. Stallman, G. DeveloperCommunity, Using the gnu compiler collection: A gnu manual for gcc version 4.3. 3, CreateSpace, Paramount, CA (2009).