KINEMATIC CALCULATION FOR MODULAR RECONFIGURABLE COOPERATING ROBOTIC SYSTEMS

Dominic Weber, Canlong Ma, Martin Wahle, Tim Detert, Burkhard Corves
Department of Mechanism Theory and Dynamics of Machines
Kackerstr. 16-18, 52072 Aachen, Germany
dominic.weber@rwth-aachen.de, canlong.ma@rwth-aachen.de, martin.wahle@rwth-aachen.de, detert@igm.rwth-aachen.de, corves@igm.rwth-aachen.de

ABSTRACT
The field of reconfigurable and cooperative robotics is attracting increasingly widespread attention in industry. The reasons for this are the many advantages of such systems over traditional fixed configuration single arm robots, such as their adaptability to changing tasks and superior stiffness properties. This paper presents a solution scheme for a computational calculation of the joint variables and Jacobian of arbitrarily configured modular cooperating robots for the task of object integrated manipulation. The system architecture is defined and a calculation procedure which takes into consideration possible kinematic redundancy as well as actuator constraints is implemented and then validated on a representative example.

KEY WORDS
Robot kinematics, reconfigurable, modular, cooperating, redundancy.

1. Introduction
The field of reconfigurable and cooperative robotics is attracting increasingly widespread attention in industry. These systems can consist of multiple cooperating arm modules that can each consist of multiple subsystem modules (see Fig. 4). The ability to reconfigure the modules results in a system that is highly adaptive and has superior performance compared to fixed configuration systems.

At the core of any model of a robot lies the kinematic model describing the relation of tool center point (TCP) movement and movement of the drives and is thus of central importance. Kinematic modeling of Modular Reconfigurable Cooperating Robots (MRCRs) needs to be valid not just for a particular system configuration, but for arbitrary combinations of modules that together form the system. This paper demonstrates an approach for the calculation of MRCRs engaged in object-integrated manipulation by first defining the system architecture and then determining the inverse kinematics solution of the system and its Jacobian matrix in MATLAB. The validity of the results is demonstrated in a dynamic simulation using SimMechanics.

A finite number of relatively simple subsystem modules are commonly used in the majority of industry applications. Therefore, an approach that uses their already well-established solutions, usually readily available in analytical form, as building blocks for the kinematic solution of the entire system is to be developed. To achieve this, the system architecture must be defined. System architectures are often defined hierarchically, as in [1]. Some system architectures break up the manipulator arm into regional and local segments [2].

Closed form kinematic solution approaches as well as iterative ones will now be reviewed. Tabandeh proposes two algorithms, one based on genetic algorithms and the other on the joint reflection operator, for calculating the Inverse Kinematic Problem (IKP) of serial modular reconfigurable robots [3]. Both algorithms solve for multiple solutions even when no closed form solution exists, and do not require knowledge of the number of solutions. IKPs can also be solved by employing a Newton Raphson algorithm using the Jacobian, which is obtainable numerically from the forward kinematics. This approach can also be extended to account for redundant manipulators by using Jacobian pseudo inverse [4], [5], singularity robust inverse [6], [7], or extended Jacobian [8] solutions. Other solution approaches use optimization algorithms such as Lagrange Multiplier Methods [9].

As opposed to the iterative solutions listed above, Ahlers presents a modular approach for solving the IKP in closed form, combining the forward and inverse kinematics of simple modules [10]. Rocha et al extend the modular idea to using entire sections of kinematic chains that can even form closed chains [11].

For parallel systems or cooperating systems the Jacobian can be assembled from the Jacobians of the individual arms that form the system. Simple examples of this can be seen in [12] and [13]. These approaches of combining arm Jacobians are shown only for particular configurations, however, not for a universal case, the latter of which will be shown in this paper.

Furthermore, some IKP calculation methods mentioned in this section are limited to certain types of configurations. Also, some of the authors do not demonstrate how the methods can be applied to systems of generic, i.e., arbitrary, configuration. In contrast, this paper aims to provide a generic, automated approach for solving the
IKP and determining the Jacobian of systems consisting of arbitrary configuration.

2. Framework for the System Architecture

The kinematic calculation methods deal with problems fulfilling the following requirements: The calculation methods shall be universal and shall allow for reconfigurability of the system, potential redundancies, and shall be extensible. In this context, universality means the applicability to any type of module and any combination thereof (i.e. serial combination, or parallel combination via the manipulated object). Reconfigurability encompasses interchanging modules, changing base frame and grasping poses, as well as link dimensions. The redundancies to be covered can be both kinematic or actuator related.

These first three requirements result in a system that has the following parameters and configuration options in Fig. 1. It should be noted here that although individual arms can be actuator deficient, systems that are actuator deficient as a whole are not covered in this paper.

Finally, extensibility of the methods refers to extending the ability of the code to calculate a larger variety of subsystems than the ones initially implemented. Based on the above requirements the chosen coding approach is one of object-oriented programming. Every subsystem and every arm are implemented as objects that have attributes and methods. This approach enables the addition of new modules with little coding effort and allows for a well-structured integration of kinematic data due to the simple information flow shown in Fig. 2. Any combination of modules can thus be handled in a unified way.

At the robot level, the arm joint variables and Jacobians are processed to formulate the kinematics for the entire MRCR system. To calculate the arm kinematics data regarding the desired path trajectory is required. The arm kinematics is determined from the subsystem kinematics. To calculate subsystem kinematics kinematic data about their configuration, e.g. link lengths is needed. The arrows indicate this information flow.

![Figure 1. System Parameters and Configuration Options](image1)

![Figure 2. Information flow between Modules](image2)

![Figure 3. Flowchart for Calculation Process](image3)
3. Kinematic Solution Strategy

Fig. 3 summarises the kinematic solution approach, in which the System IKP is derived from the Forward Kinematic Problem (FKP) solutions of the manipulator arms. There are numerical and analytical approaches for determining arm joint variables and Jacobians, but the numerical methods are the main focus of this paper, as they are universally applicable. The system Jacobian, however, is determined analytically, as this simple method is efficient and retains its universal applicability (see Fig. 3).

3.1 Arm Kinematics

3.1.1 Arm Inverse Kinematic Problem

To determine the joint variables of a manipulator arm, the following equation is solved numerically, where \( f_{\text{forward}} \) represents the arm FKP function, \( q \) represents the joint variables vector, and \( x_{\text{desired}} \) the desired path trajectory at the arm grasping point in global coordinates.

\[
f_{\text{forward}}(q) - x_{\text{desired}} = 0
\]  

(3.1)

This was solved iteratively using linear extrapolations of \( \Delta q \) with the arm Jacobian \( J \), as shown in (3.2), but can also be solved by using standard optimization solvers.

\[
\Delta q = J(q_{-1})^{-1}(x(q_{-1}) - x_{\text{desired}})
\]  

(3.2)

The initial value for \( q \) can be obtained from actuator sensor data at the initial position or from an approximate first assumption.

For kinematically redundant systems, the inverse of the Jacobian is not defined as the Jacobian is not square. Here, the Moore-Penrose pseudo inverse is used and thus a solution which minimizes the 2-norm of \( \Delta q \) is obtained. Other pseudo inverse definitions are possible, for example formulations avoiding singularities or obstacles in the path [4].

The general solution procedure at (3.2) finds only one of the possibly several solutions to the IKP, however successive solutions along the tracked path are not likely to jump in the joint space as actuator limits constrain the solution, and hence a single-solution algorithm is sufficient for many applications.

3.1.2 Arm Jacobian

If the Jacobians of the subsystem modules are available in global coordinates and in dx/dq form, as opposed to its inverse, they can be analytically merged into a single arm Jacobian. An analytical method similar to this was used to determine the system Jacobian (discussed in Section 3.2). In this paper, the arm Jacobian is however determined numerically by the use of finite differencing techniques. As opposed to the analytical method, for which rotation matrices between the global and subsystem base frames must be available, the numerical method is universally applicable. The arm Jacobian is calculated in dx/dq form.

3.2 System Kinematics

As the problem being discussed is that of object-integrated manipulation, the IKP solution of one arm does not affect that of another, and thus the system \( q \) vector is formed by sequentially appending the \( q \) vectors of the subsystems in a single vector. The system Jacobian, which is, in general, only available in dx/dq form, can therefore also be formed by appending the arm Jacobians. To achieve this, however, the arm Jacobians must first be transformed such that they refer not to their respective arm grasp points but to a single common point on the object [14]. \( T_{\text{gp, obj}} \) represents a skew symmetrical matrix of the position vector between a grasp point (‘gp’) and the object frame (‘obj’). \( T \) is a transformation matrix, \( v \) a velocity vector and \( \omega \) an angular velocity vector. The object is assumed to be a rigid body.

Expressing the grasp point velocities in terms of the object frame velocities gives:

\[
\begin{bmatrix}
  v_{\text{gp}} \\
  \omega_{\text{gp}}
\end{bmatrix}
= \begin{bmatrix}
  I \\
  0
\end{bmatrix}
\begin{bmatrix}
  v_{\text{obj}} \\
  \omega_{\text{obj}}
\end{bmatrix}
= \begin{bmatrix}
  \lambda
\end{bmatrix}
\begin{bmatrix}
  v_{\text{obj}} \\
  \omega_{\text{obj}}
\end{bmatrix}
\]  

(3.3)

Now the system Jacobian elements can be derived:

\[
\dot{q} = J_{\text{arm}}^{-1} \begin{bmatrix}
  v_{\text{gp}} \\
  \omega_{\text{gp}}
\end{bmatrix}
= \begin{bmatrix}
  J_{\text{system, obj}} \\
  \lambda
\end{bmatrix}
\begin{bmatrix}
  v_{\text{obj}} \\
  \omega_{\text{obj}}
\end{bmatrix}
\]  

(3.4)

Then they can be appended into a single system matrix:

\[
J_{\text{system}} = \begin{bmatrix}
  J_{\text{system, obj}} \\
  \vdots
\end{bmatrix}
\]  

(3.5)

The system Jacobian is therefore calculated without the need to first determine and then differentiate the FKP of the system.

3.3 Solution with Actuator Constraints

Joint variables are subject to constraints imposed by physical actuator bounds, i.e. angle/displacement and rate limits, which must be accounted for during the solution scheme. The most interesting case here is that of kinematically redundant arms, as individual actuators can be forcibly set to a \( q \) or \( \dot{q} \) bound value if these are exceeded, while possibly still meeting the requirement of tracking the path trajectory in the task space. The redundancy handling algorithm that is utilised in this paper checks for \( q \) and \( \dot{q} \) exceedance after each Jacobian extrapolation is completed. If an actuator exceeds one of these bounds, the corresponding joint value is forced to the bound it has exceeded and kept there for the duration of that time step, whereas the other joint values are calculated as shown in 3.1.1.

Due to the use of the pseudo inverse and the enforcement of the actuator constraints the magnitudes of the \( \dot{q} \) values
will be prevented from being very large. Therefore inverse kinematic singularities, near which $q$ becomes very large, are avoided.

If the system contains kinematically redundant arms, the system Jacobian is formed by using the pseudo inverse of the arm Jacobian, which does not correspond to a solution that forces the joint variables onto the bounds. The deviation from a $\min\|q\|$ solution while bounds are being enforced must therefore be taken into consideration in order to obtain an accurate system Jacobian. This remains as an issue for future development [14].

4 Validation

The validation of the kinematics results of the MKCM code was performed on an example scenario. This example scenario was chosen to be representative of all the different configuration possibilities mentioned in Fig. 4.

Specifically, the configurations of the arms are as follows:
- Arm 1: XYZ-CBB-S (9 DOF)
- Arm 2: XYZ-R (4 DOF)
- Arm 3: 3RUU-S (6 DOF).

The arms each have unique base attachment poses and grasping poses on the object. The shortest link length is 0.2m and the longest is 6m. Fig. 4 shows the three arms and where they grasp the object.

The chosen path consists of 500 path points spaced at 200 Hz and is approximately 8 m long. Firstly the kinematic results were evaluated, and subsequently the example scenario was modeled and simulated.

![Figure 4. Example Configuration: Three Manipulator Arms](image)

### 4.1 Kinematic Results

The kinematic validation of the results is here shown for one of the revolute joints on arm 1 (Fig. 4) as an example. If the physical actuator bound bandwidth is changed, then one can see clearly in Fig. 5 how the extrapolation algorithm for redundant arms enforces the bounds. As can be seen, the bounds successfully constrain the solution. In future versions of this code it would however be desirable to implement the algorithm in such a way that the resulting $q$ and $q'$ solutions are smoother over time. This may, for example, be achieved by enforcing the bounds in incremental steps while the solution is approaching a bound.

When the $q$ values obtained from the numerical solver are substituted back into $f_{\text{forward}}(q)$, a comparison of desired and calculated operational space coordinates is possible. The residual difference between these values has an error on the order of 10^{-4} meters or radians, which is deemed to be acceptable for most applications. Only one extrapolation iteration was used here, however using five iterations decreases the error to an order of magnitude of 10^{-15} meters or radians [15].
4.2 Dynamic Simulation

A dynamic simulation was carried out with SimMechanics in MATLAB to demonstrate that the system is accurately controllable using the kinematic results. For the dynamic simulation, the SimMechanics model of the example scenario is controlled by a PD-Controller. The results of the PD controlled dynamic simulation show good agreement with the desired object poses. The difference between the desired path trajectory of the object and its actual path trajectory is on the order of 10^-4 m [15]. This is deemed to be reasonably accurate, given that the dimensions of some of the robot links are up to several metres in length in this fictional example. It has therefore been demonstrated that the system is accurately controllable using elementary control methods.

5. Conclusion

This paper presents a new framework addressing the problem of calculating the inverse kinematics of modular reconfigurable cooperating robots of generic configuration that is universally applicable. This covers systems consisting of an arbitrary number of arms that can each consist of an arbitrary number of serial, parallel or hybrid configuration subsystems. Furthermore, calculation methods were developed and implemented in MATLAB. Results were then presented and it was shown that these results validate the calculation methods. The kinematic results showed good agreement with the desired path. At the end of the project, the results were successfully implemented in a dynamic model with a controller to demonstrate their validity and the potential ability to be used in an industry application. In future potential improvements to computational efficiency and the redundancy algorithms are to be investigated.

References


