Feedback Control–Based Inverse Kinematics Solvers for a Nuclear Decommissioning Robot

Thomas Burrell* Allahyar Montazeri*[1] Stephen Monk* C. James Taylor*

** Engineering Department, Lancaster University, LA1 4YR, UK*

[1]Corresponding author: e-mail: a.montazeri@lancaster.ac.uk

Abstract: The article develops two novel feedback control–based Inverse Kinematics (IK) solvers. They are evaluated for a dual–manipulator mobile robotic system with application to nuclear decommissioning. The first algorithm has similarities to other feedback control based solvers, and borrows ideas from the Cyclic Coordinate Decent and the Jacobian Transpose methods. This yields a particularly straightforward algorithm with tunable Proportional–Integral–Derivative (PID) gains to determine performance. The second approach utilises a discrete–time state space modelling framework to solve the IK problem. Although the second solver is more complex to implement, preliminary simulation results for the case study example, show that it can converge quicker, and has improved immunity to the kinematic singularities that can occur in Jacobian based methods.

Keywords: Robot arms, Robot programming, Robotic equipment, Automation

1. INTRODUCTION

The global nuclear industry is expanding with new constructions, while many existing installations are approaching the end of their operating life. Hence, the need for nuclear decommissioning is increasing everywhere. Following the Fukushima disaster in 2011, Japan has chosen to close most of its 54 reactors. In Europe, more than 50 reactors are being closed down before 2025. Nuclear installations are built with resistant materials that have specific mechanical properties, and present various radioactive and chemical hazards. Their architecture is often complex and they were not necessarily designed with the decommissioning problem in mind.

In areas of significant contamination where the use of people is not always possible, remote–controlled robots provide an invaluable option for the safe retrieval of contaminated materials, whilst safeguarding the environment and minimizing radiation exposure to operators. Hence, in the last repairs (e.g. Bogue, 2011; Bloss, 2011).

For research into decommissioning using increased levels of autonomy, such as cutting and welding, Lancaster University has developed a dual–arm mobile robotic platform, namely a Brokk–40 demolition robot with caterpillar tracks, to which two seven–function hydraulically–actuated HydroLek–7W robotic manipulators have been attached (Bakari et al., 2007, Taylor & Seward, 2010). The resulting dexterous dual–arm system illustrated in Fig. 1 is potentially capable of achieving many manipulation tasks, by combining the strength of the hydraulic actuators with the cooperative work of dual grippers or other tools. However, with six degrees–of–freedom (DOF) determining the position and orientation of the end–effector, and the actuation of a tool nominally representing a 7th DOF, these manipulators are kinematically redundant. Furthermore, they have particularities in their geometry that exclude a closed–form analytic solution to the Inverse Kinematics (IK).

Hence, earlier studies into control of the device have used a reduced number of joints (Taylor & Robertson, 2013), or have applied the Jacobian transpose method (Buss, 2004) to solve the IK (Besset & Taylor, 2014). By contrast, the present article concerns the development of novel iterative IK algorithms that do not use the Jacobian matrix. Building on existing concepts in feedback control–based IK methods (see later citations), the first solver, called IK–PID, utilises a Proportional–Integral–Derivative algorithm. By contrast, the second approach uses a discrete–time state space modelling framework. These two complementary solvers have wide applicability but in this article they are evaluated for the HydroLek manipulator.

View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Lancaster E-Prints

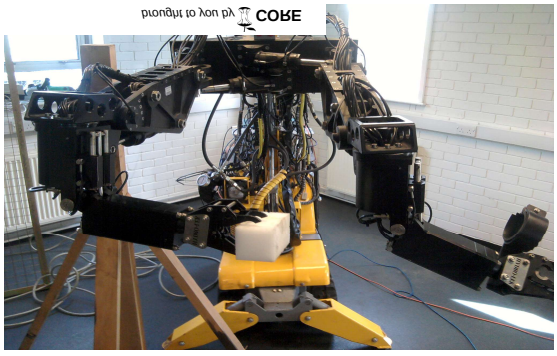

Fig. 1. Brokk–HydroLek Robotic platform.

Section 2 provides a brief overview of the nuclear robotics work currently being undertaken by the authors and thereby gives motivation for the case study. Section 3 describes conventional kinematic analysis, while section 4 develops the IK–PID algorithm with a focus on simplicity. Section 5 describes preliminary research into the second approach that focuses on speed. This is followed in section 6 by application of these two methods to the HydroLek, and comparison with the Jacobian approach, with the conclusions in section 7.

2. RESEARCH CONTEXT & CASE STUDY

With rapid start and stop, fast responses in general, and large torque–to–weight ratios, hydraulic robots are suitable for many applications. They are commonly employed by the construction and mining industries, in addition to the nuclear sector, where semi–automatic control systems are being adopted as a means of improving the efficiency, quality and safety of operations. In contrast to a typical machine driven by electric motors, however, hydraulic actuators generally have higher loop gains, wider bandwidths and lightly–damped, nonlinear dynamics (Merritt, 1976).

Particular challenges associated with such hydraulic systems include their friction characteristics, asymmetric actuation (Taylor & Robertson, 2013), hydraulic fluid compressibility (Sirouspour & Salcudean, 2001), valve saturations and dead-bands (Mohanty & Yao, 2011). Model uncertainties encompass the accumulation of oil contamination, potential leakages in the hydraulic circuit (Mohanty & Yao, 2011) and the changing viscosity of hydraulic fluid due to temperature variations (Kotzev et al., 1992). In the construction industry, automated prototypes include hydraulic manipulators for excavation and ground compaction (Shaban et al., 2008). For nuclear decommissioning, it is also necessary to take into account the large variety of items that have to be dismantled and the geometric changes that occur during the dismantling process (Taylor & Seward, 2010).

Although relatively few articles concentrate on control design specifically for hydraulic manipulators, selected examples consider generalized predictive control (Kotzev et al., 1992), backstepping (Sirouspour & Salcudean, 2001), adaptive robust (Mohanty & Yao, 2011) and state–dependent parameter (SDP) design. In the latter regard, the present final author has utilised a true digital control approach, in which data–based model identification is followed by non–minimal state space control system design, using a digital, sampled–data standpoint throughout (Taylor et al., 2013). In the nonlinear case, the approach yields SDP control systems (Taylor et al., 2011), in which the model parameters are functionally dependent on measured variables, such as joint angles and velocities.

Practical examples of this approach have included vibro–lance ground compaction (Shaban et al., 2008) and, more recently, SDP control of the Brokk–HydroLek system (Taylor & Robertson, 2013). Present research by the authors using this robotic platform falls into four main categories: (i) computer vision systems; (ii) improved dynamic modelling so as to better address system nonlinearities; (iii) autonomous cutting and wielding case studies; and (iv) co-ordination between aerial robots and ground based nuclear robots.

The broad goal is to provide improved data collection and decision making. The demolition environment inside a nuclear reactor provides many obstacles for ground based robots trying to collect data. The environment is cluttered, there is little room to manoeuvre and off-the-shelf robots are generally large and lack mobility. Secondly, the reactors are tall, with heights that cannot be surveyed by ground based robots. Finally the environment is dynamic and information must be updated quickly. This is difficult for robots such as the BROKK, which are often stationary during work.

Hence, an important recent aspect of the research is the use of aerial vehicles such as multirotors. These aim to bypass some of the problems alluded to above and provide a richer information stream to the ground based robots. Although outside of the scope of the present article, this provides many research challenges including Simultaneous Localisation and Mapping (SLAM), SLAM in dynamic environments and multiple viewpoint vision systems, among other areas.

However, the present article focuses on improved IK solutions for robotic manipulators. For later reference, the kinematics of each HydroLek manipulator are described with the ubiquitous 'D-H' parameters, as shown in Table I. The joints 1–6 are known as the the azimuth yaw, shoulder pitch, elbow pitch, forearm role, wrist pitch and wrist role.

Table I D-H Parameters for HydroLek–7W manipulator.

Joint	θ (°)	d (mm)	a (mm)	α (°)	Joint Range (°)
1	θ_1	0	70.0	90	-27.07 to 40.60
2	θ_2	0	523.4	0	-10.32 to 65.64
3	θ_3	0	165.0	-90	-42.01 to 21.43
4	θ_4	-212.0	44.45	90	-63.94 to 114.97
5	$\theta_5 - 90$	0	184.0	-90	-81.52 to -4.16
6	$\theta_6 - 90$	284.8	0	0	-260.00 to 260.00

3. KINEMATICS

The above research relies heavily upon the use of arm type robots. Forward and inverse kinematics, and the associated trajectory planning, underpin all the movement of these types of robot and so are major topics of interest.

3.1 Forward Kinematics

Forward kinematics (FK) is the process of finding the position and orientation $\boldsymbol{s}$ of a point in a kinematic chain, called the end effector, given a known joint configuration $\boldsymbol{\theta}$. Here, $\boldsymbol{s}$ is a column vector containing the position of the end-effector $[x, y, z]$ and optionally its orientation. An example of this is the Cartesian coordinate system $[x, y, z, \phi, \theta, \psi]^T$, where $\phi, \theta,$ and ψ are yaw, pitch and roll respectively. Also, $\boldsymbol{\theta}$ is a vector of n joint configurations, $[\theta_1, \theta_2 \ldots, \theta_n]^T$, where θ_i can be the extension of a prismatic joint or the angular position of a revolute joint. In general, the relationship between $\boldsymbol{s}$ and $\boldsymbol{\theta}$ is a nonlinear function:

$$\boldsymbol{s} = f(\boldsymbol{\theta}) \qquad (1)$$

FK equations are relatively straightforward to derive as they are merely a sequence of transformations from the origin to the end-effector. As is well-known, Denavit and Hartenberg suggested a means of representing a kinematic chain as a series of homogenous transformation matrices that each represent

one link and one joint. The matrices are built by finding 4 parameters that describe a transformation of a coordinate system from one joint to the next (Table I). The 4 parameters for the i^{th} joint are: d_i, θ_i, a_i and α_i. For a revolute joint, θ_i is variable while d_i is fixed, while for a prismatic joint the reverse holds. In both cases there may be a constant offset depending on the zero positions of the joints. These parameters build up a transformation matrix $\boldsymbol{T}_i$ from the 1st to the ith joint:

$$\boldsymbol{T}_i = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & a_{i-1} \\ sin(\theta_i)\cos(\alpha_{i-1}) & cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & \sin(\alpha_{i-1})d_i \\ sin(\theta_i)\sin(\alpha_{i-1}) & cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

For a n-link kinematic chain, the transformation $\boldsymbol{T}_n$ from the base coordinate system to the end-effector coordinate system is determined from the product of all the individual transformations $\boldsymbol{T}_i$, for the current joint configuration $\boldsymbol{\theta}$:

$$\boldsymbol{T}_n = \prod_{i=1}^{n} \boldsymbol{T}_i(\theta_i) \quad (3)$$

$$\boldsymbol{T}_n = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & \mathrm{y} \\ r_{31} & r_{32} & r_{33} & \mathrm{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

The transformation of coordinate systems $\boldsymbol{T}_n$ can be decomposed into a single 3x3 rotation matrix and a single 3x1 translation. The translation is equal to the vector for the position of the end-effector $(x, y, z)^T$. The rotation matrix $\boldsymbol{r}$ is equivalent to a single 3-dimensional rotation representing the orientation of the end-effector. Euler proved that any single rotation can be decomposed into 3 individual rotations. One representation of this is yaw-pitch-roll, which are rotations about the original x, y, and z axes respectively. This completes the vector $\boldsymbol{s}$ to describe the state of the end effector. There are other ways to find a FK solution, however the above D-H notation is a common standard for a general kinematic chain.

3.2 Inverse Kinematics

Inverse Kinematics (IK) is the opposite process to FK in that the joint configuration is unknown and the objective is to reach a target position and orientation $\hat{\boldsymbol{s}}$. By inverting (1), this can be described as:

$$\boldsymbol{\theta} = f^{-1}(\boldsymbol{s}) \quad (5)$$

As f is nonlinear it is often difficult to find a closed-form solution to the above IK problem. Furthermore, for kinematically redundant chains, i.e. a robot arm with seven or more DOF, there can be multiple solutions for any $\boldsymbol{s}_t$. Analytical methods do not work for all kinematic chains and they become increasingly difficult to scale with increasing numbers of links. For this reason, a number of numerical methods have been developed. These methods most often use multiple iterations to converge on a solution, hence they are considerably slower than analytical methods.

Jacobian Methods

Jacobian methods are defined by their use of the Jacobian Matrix to solve the IK problem. The Jacobian Matrix is a linear approximation of how small changes in the joint configurations change the position and orientation of the end effector. It is defined as:

$$J(\boldsymbol{\theta})_{ij} = \left(\frac{\partial \boldsymbol{s}_i}{\partial \boldsymbol{\theta}_j}\right)_{ij} \quad (6)$$

The Jacobian matrix is a function of the 'present' joint configuration. The number of rows is the length of $\boldsymbol{s}$, i.e. 3 if only the position of the end-effector is required and 6 if both the position and orientation are used. The number of columns is equal to the number of joints that are being observed. Orin and Schrader (1984), for example, describe how to efficiently calculate the Jacobian for both revolute and prismatic joints.

The Jacobian can be used to calculate the FK as, by definition, it is instantaneously equal to a mapping of changes in the joint configurations to the position of the end-effector.

$$\dot{\boldsymbol{s}} = J(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \quad (7)$$

The IK problem is equivalent to finding a change in the joint configuration $\Delta\boldsymbol{\theta}$ that generates a desired change in the position $\Delta\boldsymbol{s}$. Evaluating the Jacobian for the current joint configurations, $J = J(\boldsymbol{\theta})$, and using equation (7), it is possible to estimate the effect of any change in the joint configuration on the position.

$$\Delta\boldsymbol{s} \approx J\Delta\boldsymbol{\theta} \quad (8)$$

The desired change in $\boldsymbol{s}$ is the difference, or error, between the current position and the target $\hat{\boldsymbol{s}}$. This error can be found as $\boldsymbol{e} = \hat{\boldsymbol{s}} - \boldsymbol{s}$. By inverting (8) and substituting $\boldsymbol{e}$ for $\Delta\boldsymbol{s}$, an equation for estimating IK is obtained, where the solution joint configuration $\boldsymbol{\theta} = \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.

$$\Delta\boldsymbol{\theta} \approx J^{-1}\boldsymbol{e} \quad (9)$$

Therefore, the IK problem seems to be solvable through inversion of the Jacobian matrix. However, the Jacobian has dimensions related to the number of degrees of freedom of the kinematic chain. In most cases this will mean the Jacobian is not square, hence it is not invertible. Another problem with this method is that even when the Jacobian is invertible, it may still be close to singular (Nakamura & Hanafusa, 1986). Under some joint configurations and target positions, a singularity (Gosselin & Angeles, 1990) would occur and a solution would not be found. For these reasons, other methods that use the Jacobian matrix have been developed. Such Jacobian methods have been reviewed by Buss (2004) and an example follows.

The Jacobian Transpose method, J^T is discussed by, for example, Wolovich & Elliott (1984). The inverse of the Jacobian is replaced by its transpose, with the addition of a linear scaling factor K. Wolovich and Elliott justify this using an observation by Paul (1981) that relates the joint torques and the force vectors using the Jacobian matrix transpose:

$$\boldsymbol{\tau} = J^T\boldsymbol{F} \quad (10)$$

where $\boldsymbol{\tau}$ is a vector of torques for each joint $(\tau_1, \ldots, \tau_n)^T$, and $\boldsymbol{F}$ is a vector of Cartesian forces $(F_x, F_y, \ldots)^T$. If the Cartesian force is treated as the force of a spring that is pulling the end

effector towards the goal, then it can be related to $\boldsymbol{e}$ by the equivalent of a spring constant α, as follows,

$$\Delta\boldsymbol{\theta} = \alpha J^T \boldsymbol{e} \tag{11}$$

where,

$$\alpha = \left(\frac{|J^T \boldsymbol{e}|}{|JJ^T \boldsymbol{e}|} \right)^2 \tag{12}$$

Evaluating (11) for the joint configuration $\boldsymbol{\theta}$ will produce a small $\Delta\boldsymbol{\theta}$, such that the new joint configuration $\boldsymbol{\theta} = \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$ will move the end-effector closer to the target location. Multiple iterations draw the end effector closer to the target. Fig. 2 represents this approach in the form of a feedback loop. Note that Wolovich & Elliott (1984) provide additional justification for why the transpose of the Jacobian is a valid replacement for the inverse.

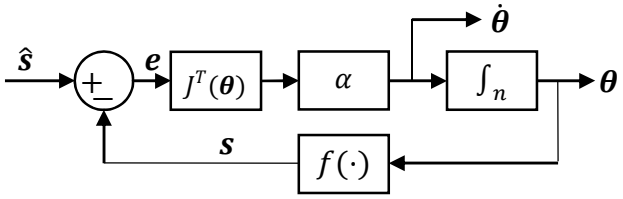

Fig. 2. Jacobian transpose algorithm represented in feedback control block diagram form.

The Jacobian Matrix can be evaluated symbolically from the forward kinematics function by software such as MATLAB and its symbolic toolbox. However the matrix determined in this way contains elements with multiple sine and cosine functions. This makes it relatively slow to evaluate the Jacobian and this step must be repeated for every iteration of the algorithm. This creates a computational bottleneck where it becomes difficult to improve the speed of the solution.

Cyclic Coordinate Decent

The CCD algorithm attempts to minimise the distance between the end effector and the target position by moving each joint in turn down the kinematic chain (Olsen & Petersen, 2011). After multiple loops through all joints, the position of the end effector will often converge onto the goal position. For a kinematic chain working in two dimensions this is straightforward, as each joint is rotated until the end effector lies on the line from the joint to the goal. Three dimensional space is more difficult: a minimal point must be generated within the plane of rotation. This point will be on a line that is both perpendicular to the plane and intersects the goal. This shadow of the goal allows CCD to be applied.

Unfortunately, CCD cannot be used to find a solution for the orientation of the end effector. Furthermore, it is necessary to partially calculate the FK multiple times per iteration so that the minimum distance between the end effector and goal can be determined. While the FK function is not as complex as the Jacobian matrix, it still contains multiple sine and cosine functions and can be slow to evaluate in some applications.

4. IK–PID ALGORITHM

The proposed IK–PID algorithm for solving the IK problem shares a similar structure to the CCD method. In this case, the joint positions are altered one at a time starting from the one closest to the end effector and looping through until a solution is found. However unlike CCD the movement of each joint does not minimise the error between the end effector and the goal. Instead the magnitude of the movement is calculated using a PID algorithm (13):

$$\Delta\theta_i = K_p * |\boldsymbol{e}(k)| + K_i * \Sigma(k) + K_d * \Delta\boldsymbol{e} \tag{13}$$

$$\Sigma(k) = \Sigma(k-1) + |\boldsymbol{e}(k)|$$

$$\Delta\boldsymbol{e} = |\boldsymbol{e}(k)| - |\boldsymbol{e}(k-1)|$$

where k increments after each joint is moved and the error is re-evaluated. In this case, K_p, K_i, and K_d are user defined gains. Unlike the Jacobian or CCD approaches, the IK–PID method of finding $\Delta\theta_i$ does not inherently return the direction of rotation. It is possible to find the preferred direction analytically but this would subsequently reintroduce most of the mathematics used to solve CCD. Alternatively, it is straightforward to solve the forward kinematics for both $+\Delta\theta_i$ and $+\Delta\theta_i$ to generate 3 sets of errors, i.e. no movement, rotation clockwise and rotation anticlockwise. The final angle is whichever of these yields the smallest error

For a practical robot each joint can only operate within a range. These limits are implemented as saturations, i.e. if the value θ_i surpasses either limit then it is set equal to that limit. The algorithm is considered to have failed to find a solution if the iteration limit is reached or if a full iteration passes with no movement. If at any point the error falls below an acceptable threshold then the algorithm should be stopped and a successful solution has been found. The approach is summarised as Algorithm I below.

IK–PID can be used to find a solution for just position or position and orientation of the end-effector. In the latter case the orientation must be scaled so that any orientation error is small compared to the position error. In this way the error will initially comprise mostly of the position error causing the end effector to move towards the goal. Once it is close to this goal, the error will comprise mostly of the orientation error and so the end effector will undergo small movements to align it.

The success and speed of the IK–PID algorithm relies upon the tuning of the gains K_p, K_i, and K_d. These three gains replace the need to calculate the Jacobian matrix or the need to find the minimal position on each iteration.

Of course, each robot will require a different set of gains for it to function optimally. For the examples considered by the authors, a poorly tuned algorithm may successfully find say 20% of reachable points, while improved tuning can improve this to close to 100%. However, since there are only three tuning gains, it is possible to optimise the algorithm by trial and error, each time testing a suitable number of test goals that are distributed through the reachable space.

This straightforward formulation and implementation could be of particular value in some application domains. To illustrate, animation has a much larger set of armature designers than robotics *per se*, and animators may find this algorithm simpler to use (e.g. when attempting to create natural systems using an underlying skeleton).

Algorithm I IK–PID

0: **Input-** Desired position $\hat{\boldsymbol{s}}$, tolerance
1: Define maximum number of iterations
2: Define gains: K_p, K_i, K_d
3: Calculate initial error, $\boldsymbol{e}_1 = \hat{\boldsymbol{s}} - \boldsymbol{s}_1$
4: **Loop For** max iterations
5: **Loop For** all i joints
6: Count: $k = k + 1$
7: Evaluate $\Delta\theta_i$,
8: $\theta_i = \theta_i + \Delta\theta_i$
9: **If** θ_i > maximum limit
10: θ_i = maximum limit
11: **End If**
12: Evaluate $\boldsymbol{s}_2 = f(\boldsymbol{\theta} + \theta_i)$, (1)
13: Evaluate $\boldsymbol{e}_2 = \hat{\boldsymbol{s}} - \boldsymbol{s}_2$
14: $\theta_i = \theta_i - 2\Delta\theta_i$
15: **If** θ_i > minimum limit
16: θ_i = minimum limit
17: **End If**
18: Evaluate $\boldsymbol{s}_3 = f(\boldsymbol{\theta} - \theta_i)$, (1)
19: Evaluate $\boldsymbol{e}_3 = \hat{\boldsymbol{s}} - \boldsymbol{s}_3$
20: Choose lowest |error|, update $\boldsymbol{\theta}$
21: **End Loop**
22: **End Loop**
23: **Output-** Joint angles $\boldsymbol{\theta}$, Any errors that occurred

5. STATE SPACE BASED SOLVER

The feedback control analogy is taken further to develop a new discrete-time state space approach to IK. This (potentially) more rigorous approach is similarly an attempt to break the computational bottleneck that arises from updating the Jacobian matrix for each incremental robot configuration. The Jacobian transpose method in Fig. 2 uses the transpose of the Jacobian to map the position error $\boldsymbol{e}$ into a vector of joint rates $\Delta\boldsymbol{\theta}$. These are integrated to determine incremental joint angles, with the error and the Jacobian updated for each iteration.

By contrast, the hypothesis investigated below is that a static mapping can be used in the place of the Jacobian, to guide the end effector to its desired position. Such IK mapping is made difficult by two main factors. Firstly, the effect of a rotation or extension of any joint in the chain will have a different effect upon the position of the end effector, depending on the configuration of both preceding and subsequent joints. Secondly, the mapping is between a 3 or 6–dimension vector representing coordinate errors in end effector position, and an *n* vector with length equal to the number of joints. Moving any single joint may or may not have an effect parallel to any individual error dimension. These issues define the requirements of the proposed mapping system, i.e. it must map both the error input and the current state of the chain into a new joint configuration at each iteration.

To fulfil these demands, a discrete time feedback controller is introduced, as shown in Fig. 3 and the following equations:

$$\boldsymbol{G}(z): \begin{matrix} \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{e}(k) \\ \Delta\boldsymbol{\theta}(k) = \boldsymbol{C}\boldsymbol{x}(k) \end{matrix} \quad (14)$$

$$\boldsymbol{\theta}(k) = \boldsymbol{\theta}(k-1) + \Delta\boldsymbol{\theta}(k) \quad (15)$$

where the transfer matrix $\boldsymbol{G}(z)$ is represented in state space form in which $\boldsymbol{x}$ is the state vector and $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are matrices that form the mapping. Note that $\boldsymbol{A} \in \mathbb{R}^{n\,x\,n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\,x\,3}$ when the error only includes position, while $\boldsymbol{B} \in \mathbb{R}^{n\,x\,6}$ when the error contains position and orientation, and $\boldsymbol{C} \in \mathbb{R}^{n\,x\,n}$.

In this formulation each of the states in $\boldsymbol{x}$ corresponds to a single joint. Although these state variables are *not* direct representations of a physical property of the joints, the state vector represents the configuration of the manipulator. Note that $\boldsymbol{B}$ in equation (14) adds each error component that is input to one or more of the states at each iteration, i.e. it maps the input to the state. Finally, $\boldsymbol{A}$ maps the state to the next state, satisfying the cross-dependence between joint configurations.

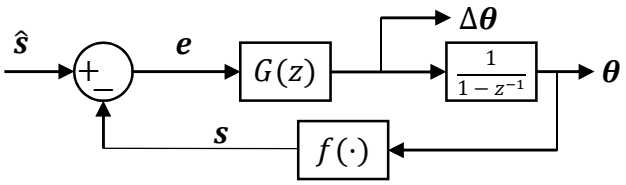

Fig. 3. Proposed Inverse Kinematics Control System.

Because $\boldsymbol{G}(z)$ forms a mapping from error to joint velocities, the problem becomes a matter of finding values for the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$, such that all desired positions $\hat{\boldsymbol{s}}$ can be reached by the end-effector (see later). There are some examples of similar control systems being used to solve the IK problem in the literature. Examples include Sciavicco & Siciliano (1987) and Pechev (2008). However, most existing approaches still involve an evaluation of the Jacobian matrix.

At the implementation stage, the new approach is summarised as Algorithm II. The main loop, lines 3-17, represents the feedback shown in Fig. 3. This is limited to a maximum number of iterations. Much like existing Jacobian methods, convergence is not guaranteed for all inputs, due to phenomena such as singularities. An iteration limit stops the program from entering an infinite loop. This value is arbitrary and, therefore, it can be set as high as is practical.

Algorithm II State space IK solver

0: **Input-** Desired position $\hat{\boldsymbol{s}}$, tolerance
1: Define maximum number of iterations (*k*)
2: **Loop For** max *k* iterations
3: Evaluate the error $\boldsymbol{e} = \hat{\boldsymbol{s}} - \boldsymbol{s}$
4: **If** |error| is < tolerance
5: **Break Loop**
6: **End If**
7: Evaluate $\boldsymbol{G}(z)$**,** (14)
8: Evaluate $\boldsymbol{\theta}(k)$**,** (15)
9: **For** all i joints
10: **If** θ_i > maximum limit
11: θ_i = maximum limit
12: **Else If** θ_i < minimum limit
13: θ_i = minimum limit
14: **End If**
15: **End Loop**
16: Evaluate $\boldsymbol{s} = f(\boldsymbol{\theta})$, (**1**)
17: **End Loop**
18: **Output-** Joint angles $\boldsymbol{\theta}$, Any errors that occurred

Lines 4-6 of Algorithm II check the magnitude of the error at each iteration. If the error falls below a given tolerance then the algorithm will stop. In some cases fine positioning is required and, if the tolerance is low, the algorithm would take a long time to reach a result. In other cases, more general coarse positioning is adequate, and a larger tolerance is used to reduce the number of iterations. Finally, the joints on most robots are unable to rotate through a full 360° or extend indefinitely. In reality they operate within a range with an upper and lower limit. There are multiple ways to apply these limits although perhaps the most straightforward is saturation. Hence, in a similar manner to Algorithm I, if the calculated joint position passes either of its limits then it is instead set equal to that limit.

At this juncture, it should be pointed out that Algorithms I and II both take a desired position and return a single joint configuration that will place the end effector there. Further control systems are clearly required to move the robot and its actuators in a suitable manner. A starting joint configuration can be picked arbitrarily or the arm might be placed in a known position close to the goal.

The present article includes only limited discussion on the 'tuning' of $G(z)$ (see section 6). In part for space limitations and in part because it is the subject of the authors' ongoing research, algorithmic methods of tuning, and an analysis of the convergence and stability properties of the approach using control theoretic methods are omitted. These results will be reported in future articles. Instead, the present article focuses on a simulation study relating to the nuclear decommissioning robot, with the algorithm tuned numerically using a particle swarm method (Montazeri et al. 2008).

6. CASE STUDY

To demonstrate the efficacy of both approaches, they are applied to the right hand side HydroLek manipulator.

6.1 IK-PID Tuning

First the IK-PID algorithm is applied to the arm. Initial basic tuning of the algorithm is achieved by altering the gain K_p while K_i and K_d are set to 0. This is tested against a single goal position of (-10 50 -20), with the error shown in Fig. 4. Under these conditions, K_p is initially set high, which results in the joints moving to their limits and the arm stalling. As the gain is gradually lowered, the goal is successfully achieved when $K_p = 0.0847$. Lowering K_p further means that more iterations are required to reach the same goal.

The second stage of tuning requires the algorithm to reach all points in the workspace. To evaluate this, 1000 randomly distributed points within the workspace are used. The algorithm is tested against each in turn, with the result being it either succeeds or fails to reach the point. With $K_p = 0.02$ as above, for example, the algorithm is successful on 960 out of 1000 points.

A trial and error approach was used to optimise all three gains for the maximum number of successes, for which $K_p = 0.012$, $K_i = 0.0000001$ and K_d=0.08. With these values the algorithm is successful on 994 out of 1000 goals. The 6 points that fail are on the periphery of the workspace and, in fact, three existing Jacobian methods have been tested with the same 1000 points and these fail on between 1 and 8 points depending on the method used.

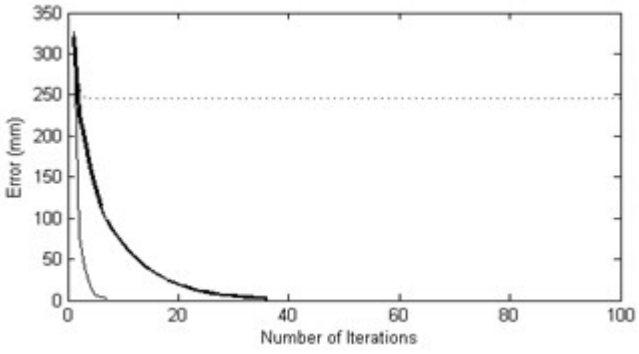

Fig. 4. End–effector errors with $K_I = K_d = 0$ and K_P = 0.0848 (dashed), 0.0847 (thin) and 0.02 (thick).

6.2 State Space Solver Tuning

Following a similar approach to the above, the state space based algorithm is tuned initially using a single reference point of (-10 50 -20). In this case, however, there are many more than three parameters to tune. In fact, the $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ matrices for a 6-DoF kinematic chain result in a total of 90 elements to tune. For the purposes of the present article, user experience and trial and error experimentation is again utilised, and yields the following matrices:

$$\boldsymbol{A} = -0.5\boldsymbol{I}^{6x6},\ \boldsymbol{B} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \boldsymbol{C} = -0.05\boldsymbol{I}^{6x6}$$

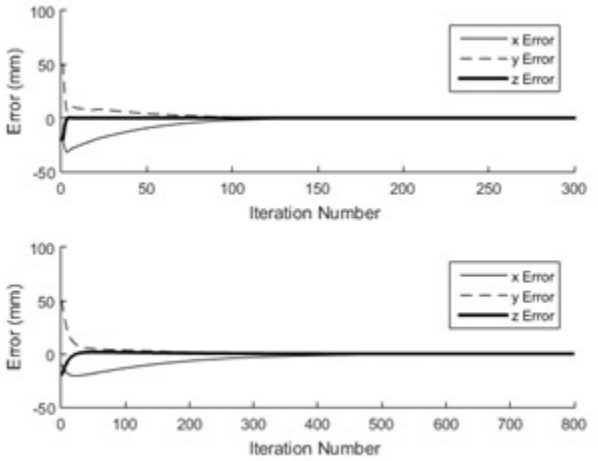

Fig. 5. The error response of the state space solver for a goal position of (-10, 50, -20), showing the initial trial and error tuning (upper subplot) and optimised (lower) systems.

Using these values it is possible to find the goal position of (-10, 50, -20) in ~100 iterations, as illustrated by Fig. 5. However, for the same 1000 distributed points as in section 6.1, this system can only reach 665 configurations. In order to reach more positions the matrices are subsequently

optimised using Particle Swarm Optimisation. Such optimisation is the subject of on-going research by the authors and, for brevity, details of the approach are omitted. Nonetheless, using standard software tools in MATLAB, the optimised system is able to reach 993 out of 1000 positions, albeit over a longer timeframe of ~350 iterations, as illustrated by Fig. 5 (for an illustrative single realisation) and Fig. 6.

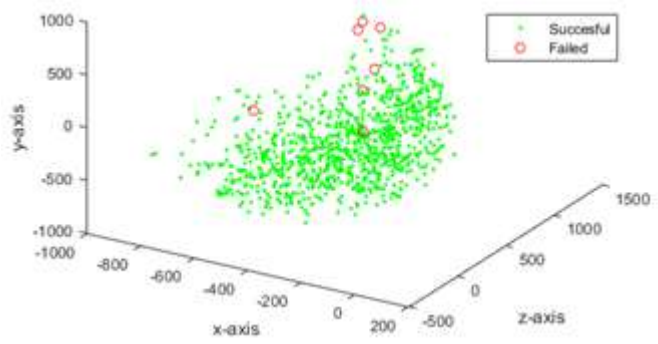

Fig. 6. Reachable workspace for the manipulator, with O indicating positions for which the state space solver failed.

6.3 *Evaluation of Timing*

One of the most important measures for any IK algorithm is the time it takes to find a solution. As IK is usually just a precursor to a more sophisticated control algorithm for actuating the robot, it is desirable to minimise the amount of processing time. In order to compare various IK methods, a random sample of 1000 points without orientation is created, all within the workspace of the HydroLek right hand side manipulator. Each algorithm is timed to produce a solution for the 1000 points. In this study, the two algorithms introduced above are compared to the Jacobian Pseudo-Inverse $J^{\dagger}$, Jacobian transpose J^{T} and Damped Least Squares (DLS) approaches (Meldrum et al., 1991a,b; Wolovich & Elliott, 1984). All five tested algorithms failed to reach at least one goal position, hence these positions were removed from the comparative tests.

Note that the same kinematics function is used for each of the algorithms and this has been optimised to run as quickly as possible. The same process was used to optimise the evaluation of the Jacobian to facilitate a fair comparison. The time for each algorithm is measured three times in MATLAB on a desktop PC with minimal background applications. The results shown in Table II show the average time for each algorithm to reach all 1000 points, with the tests repeated using 4 different tolerances.

Table II Comparison of the speed of different IK algorithms.

Method	Average time per tolerance (ms)			
	5mm	2mm	1mm	0.1mm
$J^{\dagger}$	0.823	1.115	1.368	1.727
J^{T}	0.951	1.327	1.602	2.391
DLS	1.091	1.524	1.780	2.538
IKPID	16.258	16.231	16.147	N/A
State Space	0.604	0.763	0.875	1.310

Table II shows that the IK-PID performs poorly in speed tests, which is to be expected as it must perform the FK function 12 times per iteration. However, the new state space based algorithm is significantly faster than any of the Jacobian based methods that it has been tested against to date. In particular, its average solution time is approximately 65% of the time of the Jacobian Pseudo-Inverse method, the next fastest method.

6.4 *Evaluation of Singularities*

A singularity is a position in which the Jacobian matrix loses rank and it becomes impossible to find an IK solution using the Jacobian methods. This often happens when one or more joints are aligned. For an initial evaluation of this issue, consider an arm in the x-y plane with 3 links each of length 2 units (Fig. 7). The first joint connects the first link to the origin, while the next two joints connect the next link to the end of the last. The end effector is at the end of the third link. Each joint can rotate ±60° with the 0° positions when all joints are aligned with the x axis. The D-H parameters are shown in Table III.

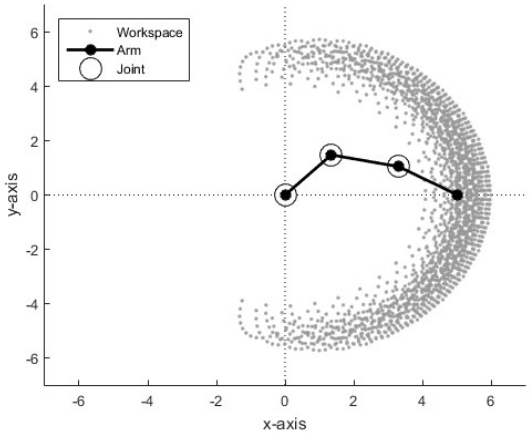

Fig. 7 Illustrative 3DoF Planar arm reaching the coordinate (5, 0). Grey dots represent the extent of the reachable space.

Table III D-H Parameters for 3 DoF planar arm.

Joint	θ (°)	d	a	α (°)	Joint Range (°)
1	θ_1	0	2	0	-60 to 60
2	θ_2	0	2	0	-60 to 60
3	θ_3	0	2	0	-60 to 60

Using equation (10), $\Delta\boldsymbol{\theta}$ is evaluated for the first iteration of the Jacobian Pseudo-Inverse method. The end effector is initially at the position (6, 0) and a desired position of (5, 0) is required, which is in line with the manipulator.

Evaluate the Jacobian:

$$\mathbf{J} = \begin{bmatrix} 0 & 0 & 0 \\ 6 & 4 & 2 \end{bmatrix}$$

Evaluate α:

$$\alpha = \left(\frac{\left| \begin{bmatrix} 0 & 6 \\ 0 & 4 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -\mathbf{1} \\ \mathbf{0} \end{bmatrix} \right|}{\left| \begin{bmatrix} 0 & 0 & 0 \\ 6 & 4 & 2 \end{bmatrix} \begin{bmatrix} 0 & 6 \\ 0 & 4 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -\mathbf{1} \\ \mathbf{0} \end{bmatrix} \right|} \right)^{\mathbf{2}} = 0$$

Therefore:

$$\Delta\boldsymbol{\theta} = 0 * J^{T} * \begin{bmatrix} -1 \\ 0 \end{bmatrix} = [0 \quad 0 \quad 0]^{T}$$

As there is no change in the joint angles after the first iteration, the subsequent iterations will have the same result. In this case the Jacobian transpose method has failed to find a solution. However, it is clear that a solution is possible as the position (5, 0) is within the workspace. Applying the state space algorithm to this robot and tuning it accordingly, it is possible to determine a solution, i.e. the joint angles are equal to 47.6°, -60.0°, and -18.8° respectively. This configuration is shown in Fig. **7** where it can clearly be seen that the end effector has arrived at the correct position.

7. CONCLUSIONS

This article has developed two algorithms for solving the Inverse Kinematics (IK) problem, one based on a straightforward PID control algorithm and the other using a discrete-time state space based framework. The IK–PID algorithm is extremely straightforward to tune and implement, making it an ideal choice for non-experts who may be daunted by the complexity of Jacobian based methods. While it is unable to match the speed of performance of other approaches, the simulation study in this article suggests that it could match the success rate of other algorithms, i.e. in finding the solution.

In this article, the state space based algorithm is presented in its preliminary state. This algorithm is more complex than IK–PID and requires longer and more sophisticated tools to tune. However, simulation results to date suggest that it is demonstrably quicker than other commonly used IK methods, and has improved robustness against singularities. However, further study into singularity robustness is required before generalised conclusions can be made. Other future work includes the formalisation of a tuning strategy for the new state space algorithm and consideration of additional case studies, i.e. using different robots.

ACKNOWLEDGEMENTS

The authors are grateful to the National Nuclear Laboratory and Nuclear Decommissioning Authority.

REFERENCES

Bakari, M., Zeid, L. & Seward, D.W., 2007. Development of a multi-arm robot for nuclear decommissioning tasks. International Journal of Advanced Robotic Systems, 4, pp. 387-406.

Besset, P. & Taylor, C.J., 2014. Inverse kinematics for a redundant robotic manipulator used for nuclear decommissioning. UKACC International Conference Control, Loughborough, UK. Paper ID: 31.

Bloss, R., 2011. How do you decommission a nuclear installation? Call in the robots. Industrial Robot: An International Journal, 37, pp. 133-136.

Bogue, R., 2011. Robots in the nuclear industry: a review. Industrial Robot: An International Journal, 38, pp. 113-118.

Buss S. R., 2004. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation, vol. 17, p. 16.

Gosselin C. & Angeles J., 1990. Singularity analysis of closed-loop kinematic chains. IEEE Transactions on Robotics and Automation, 6, pp. 281-290.

Kotzev, A., Cherchas, D., Lawrencet, P. & Sepehrit, N., 1992. Generalized predictive control of a robotic manipulator with hydraulic actuators. Robotica, 10, pp. 447-495.

Meldrum D., Rodriguez G., & Franklin G., 1991a. An order (N) recursive inversion of the Jacobian for an N-link serial manipulator. Proceedings IEEE International Conference on Robotics and Automation, pp. 1175-1180.

Meldrum D., Rodriguez G., and Franklin G., 1991b. Efficient control with an order (n) recursive inversion of the jacobian for an n-link serial manipulator. American Control Conference, pp. 2039-2044.

Merritt, E., 1976. Hydraulic Control Systems. New York: John Wiley.

Mohanty, A. & Yao, B., 2011. Indirect adaptive robust control of hydraulic manipulators with accurate parameter estimates. IEEE Transactions on Control Systems Technology, 9, pp. 567-575.

Montazeri A., Poshtan J., Yousefi-Koma A., 2008. The use of particle swarm to optimize the control system in a PZT laminated plate. Smart Materials and Structures, 17(4).

Nakamura, Y. & Hanafusa H., 1986. Inverse kinematic solutions with singularity robustness for robot manipulator control. Journal of dynamic systems, measurement, and control, 108, pp. 163-171.

Olsen, A. & Petersen, H., 2011. Inverse kinematics by numerical and analytical cyclic coordinate descent. Robotica, 29(4), pp. 619-626.

Orin D. E. & Schrader W. W., 1984. Efficient computation of the Jacobian for robot manipulators. The International Journal of Robotics Research, 3, pp. 66-75.

Paul R. P., 1981. Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators. MIT Press.

Pechev A. N., 2008. Inverse kinematics without matrix inversion. IEEE International Conference on Robotics and Automation, pp. 2005-2012.

Sciavicco L. & Siciliano B., 1987. A dynamic solution to the inverse kinematic problem for redundant manipulators. Proceedings IEEE International Conference on Robotics and Automation, pp. 1081-1087.

Shaban, E., Ako, S., Taylor, C. & Seward, D., 2008. Development of an automated verticality align ment system for a vibro-lance. Automation in Construction, 17(5), pp. 645-655.

Sirouspour, M. & Salcudean, S., 2001. Nonlinear control of hydraulic robots. IEEE Transactions on Robotics and Automation, 17(2), pp. 173-182.

Taylor, C.J., Chotai, A. & Burnham, K., 2011. Controllable forms for stabilising pole assignment design of generalised bilinear systems. Electronics Letters, 47(7), pp. 437-439.

Taylor, C.J. & Robertson, D., 2013. State-dependent control of a hydraulically-actuated nuclear decommissioning robot. Control Engineering Practice, 21(12), pp. 1716-1725.

Taylor, C.J. & Seward, D., 2010. Control of a dual-arm robotic manipulator. Nuclear Engineering International, August, 55, pp. 24-26.

Taylor, C.J., Young, P. & Chotai, A., 2013. True Digital Control: Statistical Modelling and Non-Minimal State Space Design. s.l.:John Wiley and Sons.

Wolovich W. A. & Elliott H., 1984. A computational technique for inverse kinematics. 23rd IEEE Conference on Decision and Control, 1984, pp. 1359-1363.