Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods

Samuel R. Buss*
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093-0112
sbuss@math.ucsd.edu

October 7, 2009

Note: This is an introduction that was originally written for a paper by Buss and Kim [7], but was subsequently separated out. This report is being made available via the internet — there are no plans to publish it.

Abstract

This is a introduction to the Jacobian transpose method, the pseudoinverse method, and the damped least squares methods for inverse kinematics (IK). The mathematical foundations of these methods are presented, with an analysis based on the singular value decomposition.

1 Introduction

A rigid multibody system consists of a set of rigid objects, called links, joined together by joints. Simple kinds of joints include revolute (rotational) and prismatic (translational) joints. It is also possible to work with more general types of joints, and thereby simulate non-rigid objects. Well-known applications of rigid multibodies include robotic arms as well as virtual skeletons for animation in computer graphics.

To control the movement of a rigid multibody it is common to use inverse kinematics (IK). For IK, it is presumed that specified points, called "end

*Supported in part by NSF grant DMS-0100589. Contact author: sbuss@ucsd.edu

effectors," on the links are are assigned "target positions." To solve the IK problem, we must find settings for the joint angles so that the resulting configuration of the multibody places each end effector at its target position. More general formulations of IK allow also orientation goals, or directional goals.

There are several methods for solving IK problems, coming originally from robotics applications. These include cyclic coordinate descent methods [43], pseudoinverse methods [45], Jacobian transpose methods [5, 46], the Levenberg-Marquardt damped least squares methods [41, 34], quasi-Newton and conjugate gradient methods [43, 49, 15], and neural net and artificial intelligence methods [19, 27, 36, 38, 20, 22, 40, 16].

The present paper focuses on applications of IK in computer graphics and real-time animation. There has already been extensive use of IK in computer graphics [18, 26, 25, 44, 23, 2, 1, 17, 24, 29, 39, 21, 37, 12]: the most common applications are animating humans or creatures by specifying the positions, and possibly the orientations, of their hands, feet and head. Our interests lie particularly in using target positions for end effectors to animate an entire multibody, and in methods that are robust and behave well in wide range of situations. As part of the robustness, we want the end effectors to track the target positions and to do a reasonable job even when the target positions are in unreachable positions. In this paper, we consider only first order methods and consider the following generic application: we presume a multibody has multiple end effectors and multiple target positions, given in real-time in an online fashion, and want to update the multibody configuration so as to dynamically track the target positions with the end effectors.

One might wonder why it is important to allow target positions to be unreachable. There are several reasons: First, it may be difficult to completely eliminate the possibility of unreachable positions and still get the desired motion. Second, if target positions are barely reachable and can be reached only with full extension of the links, then the situation is very similar to having unreachable targets. Unfortunately, the situation of target positions in unreachable positions is difficult to handle robustly. Many methods, such as the pseudoinverse or Jacobian transpose methods, will oscillate badly in this situation; however, (selectively) damped least squares methods can still perform well with unreachable target positions.

The outline of the paper is as follows. We first introduce the mathematical framework for the IK problem. We then discuss the Jacobian transpose method, the pseudoinverse method, the singular value decomposition, and the damped least squares (DLS) method. For an extension of the DLS methods to a method called *selectively damped least squares* (SDLS), see

Buss and Kim [7]. Nearly all the present paper is expository, but new aspects include the possibility of forming the Jacobian matrix with the target positions instead of the end effector positions. We attempt to explain the mathematical foundations clearly so as to elucidate the strengths and weaknesses of the various methods.

For simplicity and to keep the paper short, we do not consider any aspects of joint limits or avoiding self-collisions; rather, we will only consider the "pure" IK problem without joint limits and without self-collisions.

2 Preliminaries: forward kinematics and Jacobians

A multibody is modeled with a set of links connected by joints. There are a variety of possible joint types. Perhaps the most common type is a rotational joint with its configuration described by a single scalar angle value. Other joint types include prismatic (i.e., translational, or sliding) joints, screw joints, etc. For simplicity, we will discuss only rotational joints, but the algorithms and theory all apply to arbitrary joints. The key point is that the configuration of a joint is a continuous function of one or more real scalars; for rotational joints, the scalar is the angle of the joint.

The complete configuration of the multibody is specified by the scalars $\theta_1, \ldots, \theta_n$ describing the joints' configurations. We assume there are n joints and each θ_j value is called a *joint angle* (but, as we just said, could more generally represent a value which is not an angle). Certain points on the links are identified as *end effectors*. If there are k end effectors, their positions are denoted $\mathbf{s}_1, \ldots, \mathbf{s}_k$. Each end effector position $\mathbf{s}_i$ is a function of the joint angles. We write $\vec{\mathbf{s}}$ for the column vector $(\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k)^T$; this can be viewed as a column vector either with $m = 3k$ many scalar entries or with k many entries from $\mathbb{R}^3$.

The multibody will be controlled by specifying target positions for the end effectors. The target positions are also given by a vector $\vec{\mathbf{t}} = (\mathbf{t}_1, \ldots, \mathbf{t}_k)^T$, where $\mathbf{t}_i$ is the target position for the ith end effector. We let $\mathbf{e}_i = \mathbf{t}_i - \mathbf{s}_i$, the desired change in position of the ith end effector. We also let $\vec{\mathbf{e}} = \vec{\mathbf{t}} - \vec{\mathbf{s}}$.

The joint angles are written as a column vector as $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^T$. The end effector positions are functions of the joint angles; this fact can be expressed as $\vec{\mathbf{s}} = \vec{\mathbf{s}}(\boldsymbol{\theta})$, or, for $i = 1, \ldots, k$, $\mathbf{s}_i = \mathbf{s}_i(\boldsymbol{\theta})$. The IK problem is to find values for the θ_j's so that

$$\mathbf{t}_i \;=\; \mathbf{s}_i(\boldsymbol{\theta}), \qquad \text{for all } i. \tag{1}$$

Unfortunately, there may not always be a solution to (1), and there may not be a unique (best) solution. Even in well-behaved situations, there may be no closed form equation for the solution.

We can, however, use iterative methods to approximate a good solution. For this, the functions $\mathbf{s}_i$ are linearly approximated using the Jacobian matrix. The Jacobian matrix J is a function of the $\boldsymbol{\theta}$ values and is defined by

$$J(\boldsymbol{\theta}) \;=\; \left(\frac{\partial \mathbf{s}_i}{\partial \theta_j}\right)_{i,j}.$$

Note that J can be viewed either as a $k \times n$ matrix whose entries are vectors from $\mathbb{R}^3$, or as $m \times n$ matrix with scalar entries (with $m = 3k$).

The basic equation for forward dynamics that describes the velocities of the end effectors can be written as follows (using dot notation for first derivatives):

$$\dot{\vec{\mathbf{s}}} = J(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}. \tag{2}$$

The Jacobian leads to an iterative method for solving equation (1). Suppose we have current values for $\boldsymbol{\theta}$, $\vec{\mathbf{s}}$ and $\vec{\mathbf{t}}$. From these, the Jacobian $J = J(\boldsymbol{\theta})$ is computed. We then seek an update value $\Delta\boldsymbol{\theta}$ for the purpose of incrementing the joint angles $\boldsymbol{\theta}$ by $\Delta\boldsymbol{\theta}$:

$$\boldsymbol{\theta} \;:=\; \boldsymbol{\theta} + \Delta\boldsymbol{\theta}. \tag{3}$$

By (2), the change in end effector positions caused by this change in joint angles can be estimated as

$$\Delta\vec{\mathbf{s}} \;\approx\; J\,\Delta\boldsymbol{\theta}. \tag{4}$$

The idea is that the $\Delta\boldsymbol{\theta}$ value should chosen so that $\Delta\vec{\mathbf{s}}$ is approximately equal to $\vec{\mathbf{e}}$, although it is also common to choose $\Delta\boldsymbol{\theta}$ so that the approximate movement $\Delta\vec{\mathbf{s}}$ in the end effectors (partially) matches the velocities of the target positions (see [45]). The update of the joint angles can be used in two modes: (i) Each simulation step performs a single update to the value of joint angles using equation (3), so that the end effector positions approximately follow the target positions. (ii) The joint angles are updated iteratively until a value of $\vec{\mathbf{s}}$ is obtained that is sufficiently close to a solution. It is also possible to use a hybrid of (i) and (ii), that is, using a small number of repeated updates using (3) so as to more accurately track the end effector positions.

The rest of this paper discusses strategies for choosing $\Delta\boldsymbol{\theta}$ to update the joint angles. In light of (4), one approach is to solve the equation

$$\vec{\mathbf{e}} \;=\; J\,\Delta\boldsymbol{\theta} \tag{5}$$

The entries in the Jacobian matrix are usually very easy to calculate. If the jth joint is a rotational joint with a single degree of freedom, the joint angle is a single scalar θ_j. Let $\mathbf{p}_j$ be the position of the joint, and let $\mathbf{v}_j$ be a unit vector pointing along the current axis of rotation for the joint. In this case, if angles are measured in radians with the direction of rotation given by the right rule and if the ith end effector is affected by the joint, then the corresponding entry in the Jacobian is

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} \;=\; \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j),$$

If the ith end effector is not affected by the jth joint, then of course $\partial \mathbf{s}_i / \partial \theta_j = 0$.

If the jth joint is translational, the entry in the Jacobian matrix is even easier to compute. Suppose the jth joint performs translation the direction of the unit vector $\mathbf{v}_j$, so that the the joint "angle" measures distance moved in the direction $\mathbf{v}_j$. Then if the ith end effector is affected by the jth joint, we have

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} \;=\; \mathbf{v}_j.$$

For more information, see Orin and Schrader [35] who discuss how to calculate the Jacobian matrix entries for different representations of joints and multibodies. The textbook [6, Ch. 12] also discusses a representation of rigid multibodies and how to calculate the Jacobian.

Calculating the Jacobian

for $\Delta\boldsymbol{\theta}$. In most cases, this equation cannot be solved uniquely. Indeed, the Jacobian J may not be square or invertible, and even if is invertible, just setting $\Delta\boldsymbol{\theta} = J^{-1}\vec{\mathbf{e}}$ may work poorly if J is nearly singular.

An alternate Jacobian. An alternate method for defining the Jacobian matrix is to let

$$J(\boldsymbol{\theta}) \;=\; \left(\frac{\partial \mathbf{t}_i}{\partial \theta_j}\right)_{i,j},$$

where the partial derivative is calculated using the formula for $(\partial \mathbf{s}_i / \partial \theta_j)$ with $\mathbf{t}_i$ substituted for $\mathbf{s}_i$. The meaning of $\partial \mathbf{t}_i / \partial \theta_j$ is that the target position is thought of as being attached to the same link as the ith end

effector. The intuition is that with this formulation of the Jacobian, we are trying to move the target positions towards the end effectors, rather than the end effectors towards the target positions.

The alternate Jacobian may be used in place of the usual Jacobian in any of the algorithms discussed below. Our experience has been that this alternate can improve on the usual Jacobian in terms of reducing oscillation or overshoot when target positions are too far away to be reached by the end effectors. However, the drawback is that in some configurations, the alternative Jacobian can lead to "jerky" behavior. This is particularly true for rotational joints when the multibody's links are folded back on each other trying to reach a close target position.

Setting target positions closer. A recurring problem in tracking target positions, is that when the target positions are too distant, the multibody's arms stretch out to try to reach the target position. Once the multibody is extended in this way, it usually is near a singularity (that is, the Jacobian is very sensitive to small changes in joint angles), and the multibody will often shake or jitter, attempting unsuccessfully to reach the distant target. These effects can be reduced with DLS and SDLS algorithms, but are difficult to remove completely.

One technique to reduce this problem is to move the target positions in closer to the end effector positions. For this, we change the definition of $\vec{\mathbf{e}}$; instead of merely setting $\vec{\mathbf{e}} = \vec{\mathbf{t}} - \vec{\mathbf{s}}$, each component $\mathbf{e}_i$ in the vector $\vec{\mathbf{e}}$ has its length clamped to a specified maximum value. That is, we define

$$\mathbf{e}_i \;=\; \mathrm{ClampMag}(\mathbf{t}_i - \mathbf{s}_i, D_{\max}),$$

where

$$\mathrm{ClampMag}(\mathbf{w}, d) \;=\; \begin{cases} \mathbf{w} & \text{if } ||\mathbf{w}|| \le d \\ d\frac{\mathbf{w}}{||\mathbf{w}||} & \text{otherwise} \end{cases}$$

Here $||\mathbf{w}||$ represents the usual Euclidean norm of $\mathbf{w}$. The value $D_{\max}$ is an upper bound on how far we attempt to move an end effector in a single update step.

For damped least squares, clamping the magnitudes of $\vec{\mathbf{e}}_i$ in this way can reduce oscillation when target positions are out of reach. This has the advantage of allowing the use of a smaller damping constant; the smaller damping constant allows significantly quicker convergence to target positions.* When the end effectors are tracking continuously moving target

*Also for SDLS, [7] have found that clamping the magnitudes of $\vec{\mathbf{e}}_i$ in this way can effectively reduce oscillation when target positions are out of reach.

positions, the $D_{\max}$ distance should be at least several times larger than an end effector moves in a single update step. In our experience, setting $D_{\max}$ to be approximately half the length of a typical link works well.

For target positions that may jump discontinuously, we have used separate maximum values $D_{\max,i}$ for each i. After a discontinuous movement of the target positions (or when beginning a simulation of a continuously moving target), we initially set $D_{\max,i}$ to infinity. After the first simulation step, we let d_i be the amount by which the previous simulation step moved the ith end effector closer to its target position. Then, we let $D_{\max,i} = d_i + D_{\max}$, and use $D_{\max,i}$ to clamp the magnitude of $\mathbf{e}_i$.

3 The Jacobian transpose method

The Jacobian transpose method was first used for inverse kinematics by [5, 46]. The basic idea is very simple: use the transpose of J instead of the inverse of J. That is, we set $\Delta\boldsymbol{\theta}$ equal to

$$\Delta\boldsymbol{\theta} \;=\; \alpha J^T \vec{\mathbf{e}},$$

for some appropriate scalar α. Now, of course, the transpose of the Jacobian is not the same as the inverse; however, it is possible to justify the use of the transpose in terms of virtual forces. More generally, it can be shown that the following theorem holds [5, 46].

Theorem 1 *For all J and $\vec{\mathbf{e}}$, $\langle JJ^T\vec{\mathbf{e}}, \vec{\mathbf{e}}\rangle \geq 0$.*

Proof The proof is trivial: $\langle JJ^T\vec{\mathbf{e}}, \vec{\mathbf{e}}\rangle = \langle J^T\vec{\mathbf{e}}, J^T\vec{\mathbf{e}}\rangle = ||J^T\vec{\mathbf{e}}||^2 \geq 0$. □

The approximation (4) implies that, for sufficiently small $\alpha > 0$, updating the angles by equation (3) using $\Delta\boldsymbol{\theta} = \alpha J^T\vec{\mathbf{e}}$ will change the end effector positions by approximately $\alpha JJ^T\vec{\mathbf{e}}$. By Theorem 1, this has the effect of reducing the magnitude of the error vector $\vec{\mathbf{e}}$ if α is small enough.

It remains to decide how to choose the value of α. One reasonable way to try to minimize the new value of the error vector $\vec{\mathbf{e}}$ after the update. For this, we assume that the change in end effector position will be exactly $\alpha JJ^T\vec{\mathbf{e}}$, and choose α so as to make this value as close as possible to $\vec{\mathbf{e}}$. This gives

$$\alpha \;=\; \frac{\langle \vec{\mathbf{e}}, JJ^T\vec{\mathbf{e}}\rangle}{\langle JJ^T\vec{\mathbf{e}}, JJ^T\vec{\mathbf{e}}\rangle}.$$

4 The pseudoinverse method

The pseudoinverse method sets the value $\Delta\boldsymbol{\theta}$ equal to

$$\Delta\boldsymbol{\theta} \;=\; J^{\dagger}\vec{\mathbf{e}}, \tag{6}$$

where the $n \times m$ matrix $J^{\dagger}$ is the *pseudoinverse* of J, also called the *Moore-Penrose inverse* of J. It is defined for all matrices J, even ones which are not square or not of full row rank. The pseudoinverse gives the best possible solution to the equation $J\Delta\boldsymbol{\theta} = \vec{\mathbf{e}}$ in the sense of least squares. In particular, the pseudoinverse has the following nice properties. Let $\Delta\boldsymbol{\theta}$ be defined by equation (6). First, suppose $\vec{\mathbf{e}}$ is in the range (i.e., the column span) of J. In this case, $J\Delta\boldsymbol{\theta} = \vec{\mathbf{e}}$; furthermore, $\Delta\boldsymbol{\theta}$ is the unique vector of smallest magnitude satisfying $J\Delta\boldsymbol{\theta} = \vec{\mathbf{e}}$. Second, suppose that $\vec{\mathbf{e}}$ is not in the range of J. In this case, $J\Delta\boldsymbol{\theta} = \vec{\mathbf{e}}$ is impossible. However, $\Delta\boldsymbol{\theta}$ has the property that it minimizes the magnitude of the difference $J\Delta\boldsymbol{\theta} - \vec{\mathbf{e}}$. Furthermore, $\Delta\boldsymbol{\theta}$ is the unique vector of smallest magnitude which minimizes $||J\Delta\boldsymbol{\theta} - \vec{\mathbf{e}}||$, or equivalently, which minimizes $||J\Delta\boldsymbol{\theta} - \vec{\mathbf{e}}||^2$.

The pseudoinverse tends to have stability problems in the neighborhoods of singularities. At a singularity, the Jacobian matrix no longer has full row rank, corresponding to the fact that there is a direction of movement of the end effectors which is not achievable. If the configuration is *exactly* at a singularity, then the pseudoinverse method will not attempt to move in an impossible direction, and the pseudoinverse will be well-behaved. However, if the configuration is close to a singularity, then the pseudoinverse method will lead to very large changes in joint angles, even for small movements in the target position. In practice, roundoff errors mean that true singularities are rarely reached and instead singularity have to be detected by checking values for being near-zero.

The pseudoinverse has the further property that the matrix $(I - J^{\dagger}J)$ performs a projection onto the nullspace of J. Therefore, for all vectors $\boldsymbol{\varphi}$, $J(I - J^{\dagger}J)\boldsymbol{\varphi} = \mathbf{0}$. This means that we can set $\Delta\boldsymbol{\theta}$ by

$$\Delta\boldsymbol{\theta} \;=\; J^{\dagger}\vec{\mathbf{e}} + (I - J^{\dagger}J)\boldsymbol{\varphi} \tag{7}$$

for any vector $\boldsymbol{\varphi}$ and still obtain a value for $\Delta\boldsymbol{\theta}$ which minimizes the value $J\Delta\boldsymbol{\theta} - \vec{\mathbf{e}}$. This nullspace method was first exploited Liégeois [28], who used it to avoid joint limits. By suitably choosing $\boldsymbol{\varphi}$, one can try to achieve secondary goals in addition to having the end effectors track the target positions. For instance, $\boldsymbol{\varphi}$ might be chosen to try to return the joint angles back to rest positions [18]: this can help avoid singular configurations.

A number of authors (see [4]) have used the nullspace method to help avoid singular configurations by maximizing Yoshikawa's manipulability measure [48, 47]. Maciejewski and Klein [30] used the nullspace method for obstacle avoidance. A more sophisticated nullspace method, called the *extended Jacobian method*, was proposed by Baillieul [4]: in the extended Jacobian method a local minimum value of a function is tracked as a secondary objective. The nullspace method has also been used to assign different priorities to different tasks (see [10, 3]).

An algorithm for the pseudoinverse method can be derived as follows: From equation (5), we get the *normal equation*

$$J^T J \Delta\boldsymbol{\theta} \;=\; J^T \vec{\mathbf{e}}.$$

Then we let $\vec{\mathbf{z}} = J^T \vec{\mathbf{e}}$ and solve the equation

$$(J^T J)\Delta\boldsymbol{\theta} \;=\; \vec{\mathbf{z}}. \tag{8}$$

Now it can be shown that $\vec{\mathbf{z}}$ is always in the range of $J^T J$, hence equation (8) always has a solution. In principle, row operations can be used to find the solution to (8) with minimum magnitude; however, in the neighborhood of singularities, the algorithm is inherently numerically unstable.

When J has full row rank, then JJ^T is guaranteed to be invertible. In this case, the minimum magnitude solution $\Delta\boldsymbol{\theta}$ to equation (8) can be expressed as

$$\Delta\boldsymbol{\theta} \;=\; J^T (JJ^T)^{-1} \vec{\mathbf{e}}. \tag{9}$$

To prove this, note that if $\Delta\boldsymbol{\theta}$ satisfies (9), then $\Delta\boldsymbol{\theta}$ is in the row span of J and $J\Delta\boldsymbol{\theta} = \vec{\mathbf{e}}$. Equation (9) cannot be used if J does not have full row rank. A general formula for the pseudoinverse for J not of full row rank can be found in [6].

The pseudoinverse method is widely discussed in the literature but it often performs poorly because of instability near singularities. The (selectively) damped least squares methods have much superior performance.

5 Damped least squares

The damped least squares method avoids many of the pseudoinverse method's problems with singularities and can give a numerically stable method of selecting $\Delta\boldsymbol{\theta}$. It is also called the Levenberg-Marquardt method and was first used for inverse kinematics by Wampler [41] and Nakamura and Hanafusa [34].

The damped least squares method can be theoretically justified as follows (see [42]). Rather than just finding the minimum vector $\Delta\boldsymbol{\theta}$ that gives a best solution to equation (5), we find the value of $\Delta\boldsymbol{\theta}$ that minimizes the quantity

$$||J\Delta\boldsymbol{\theta} - \vec{\mathbf{e}}||^2 + \lambda^2||\Delta\boldsymbol{\theta}||^2,$$

where $\lambda \in \mathbb{R}$ is a non-zero damping constant. This is equivalent to minimizing the quantity

$$\left|\left|\begin{pmatrix} J \\ \lambda I \end{pmatrix} \Delta\boldsymbol{\theta} - \begin{pmatrix} \vec{\mathbf{e}} \\ \mathbf{0} \end{pmatrix}\right|\right|.$$

The corresponding normal equation is

$$\begin{pmatrix} J \\ \lambda I \end{pmatrix}^T \begin{pmatrix} J \\ \lambda I \end{pmatrix} \Delta\boldsymbol{\theta} \;=\; \begin{pmatrix} J \\ \lambda I \end{pmatrix}^T \begin{pmatrix} \vec{\mathbf{e}} \\ \mathbf{0} \end{pmatrix}.$$

This can be equivalently rewritten as

$$(J^TJ + \lambda^2 I)\Delta\boldsymbol{\theta} \;=\; J^T\vec{\mathbf{e}}.$$

It can be shown (by the methods of section 6 below) that $J^TJ + \lambda^2 I$ is non-singular. Thus, the damped least squares solution is equal to

$$\Delta\boldsymbol{\theta} \;=\; (J^TJ + \lambda^2 I)^{-1}J^T\vec{\mathbf{e}}. \tag{10}$$

Now J^TJ is an $n \times n$ matrix, where n is the number of degrees of freedom. It is easy to show that $(J^TJ + \lambda^2 I)^{-1}J^T \;=\; J^T(JJ^T + \lambda^2 I)^{-1}$. Thus,

$$\Delta\boldsymbol{\theta} \;=\; J^T(JJ^T + \lambda^2 I)^{-1}\vec{\mathbf{e}}. \tag{11}$$

The advantage of equation (11) over (10) is that the matrix being inverted is only $m \times m$ where $m = 3k$ is the dimension of the space of target positions, and m is often much less than n.

Additionally, (11) can be computed without needing to carry out the matrix inversion, instead row operations can find $\vec{\mathbf{f}}$ such that $(JJ^T + \lambda^2 I)\vec{\mathbf{f}} = \vec{\mathbf{e}}$ and then $J^T\vec{\mathbf{f}}$ is the solution.

The damping constant depends on the details of the multibody and the target positions and must be chosen carefully to make equation (11) numerically stable. The damping constant should large enough so that the solutions for $\Delta\boldsymbol{\theta}$ are well-behaved near singularities, but if it is chosen too large, then the convergence rate is too slow. There have been a number of methods proposed for selecting damping constants dynamically based on the configuration of the articulated multibody [34, 14, 15, 31, 11, 13, 8, 9, 33, 32].

6 Singular value decomposition

The singular value decomposition (SVD) provides a powerful method for analyzing the pseudoinverse and the damped least squares methods. In addition, we shall use the SVD to design a selectively damped least squares method in [7]. Let J be the Jacobian matrix. A *singular value decomposition* of J consists of expressing J in the form

$$J \;=\; UDV^T,$$

where U and V are orthogonal matrices and D is diagonal. If J is $m \times n$, then U is $m \times m$, D is $m \times n$, and V is $n \times n$. The only non-zero entries in the matrix D are the values $\sigma_i = d_{i,i}$ along the diagonal. We henceforth assume $m \leq n$. Without loss of generality, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$.

Note that the values σ_i may be zero. In fact, the rank of J is equal to the largest value r such that $\sigma_r \neq 0$. For $i > r$, $\sigma_i = 0$. We use $\mathbf{u}_i$ and $\mathbf{v}_i$ to denote the ith columns of U and V. The orthogonality of U and V implies that the columns of U (resp., of V) form an orthonormal basis for $\mathbb{R}^m$ (resp., for $\mathbb{R}^n$). The vectors $\mathbf{v}_{r+1}, \ldots, \mathbf{v}_n$ are an orthonormal basis for the nullspace of J. The singular value decomposition of J always exists, and it implies that J can be written in the form

$$J \;=\; \sum_{i=1}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \;=\; \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T. \tag{12}$$

The transpose, D^T, of D is the $n \times m$ diagonal matrix with diagonal entries $\sigma_i = d_{i,i}$. The product DD^T is the $m \times m$ matrix with diagonal entries $d_{i,i}^2$. The pseudoinverse, $D^\dagger = (d_{i,j}^\dagger)$, of D is the $n \times m$ diagonal matrix with diagonal entries

$$d_{i,i}^\dagger \;=\; \begin{cases} 1/d_{i,i} & \text{if } d_{i,i} \neq 0 \\ 0 & \text{if } d_{i,i} = 0. \end{cases}$$

The pseudoinverse of J is equal to

$$J^\dagger \;=\; VD^\dagger U^T.$$

Thus,

$$J^\dagger \;=\; \sum_{i=1}^{r} \sigma_i^{-1} \mathbf{v}_i \mathbf{u}_i^T. \tag{13}$$

The damped least squares method is also easy to understand with the SVD. The matrix $JJ^T + \lambda^2 I$ is equal to

$$JJ^T + \lambda^2 I \;=\; (UDV^T)(VD^TU^T) + \lambda^2 I \;=\; U(DD^T + \lambda^2 I)U^T.$$

The matrix $DD^T + \lambda^2 I$ is the diagonal matrix with diagonal entries $\sigma_i^2 + \lambda^2$. Clearly, $DD^T + \lambda^2 I$ is non-singular, and its inverse is the $m \times m$ diagonal matrix with non-zero entries $(\sigma_i^2 + \lambda^2)^{-1}$. Then,

$$J^T(JJ^T + \lambda^2 I)^{-1} \;=\; (VD^T(DD^T + \lambda^2 I)^{-1}U^T \;=\; VEU^T,$$

where E is the $n \times m$ diagonal matrix with diagonal entries equal to

$$e_{i,i} \;=\; \frac{\sigma_i}{\sigma_i^2 + \lambda^2}.$$

Thus, the damped least squares solution can be expressed in the form

$$J^T(JJ^T + \lambda^2 I)^{-1} \;=\; \sum_{i=1}^{r} \tfrac{\sigma_i}{\sigma_i^2+\lambda^2} \mathbf{v}_i \mathbf{u}_i^T. \tag{14}$$

Comparison of equations (13) and (14) makes clear the relationship between the pseudoinverse and damped least squares methods. In both cases, J is "inverted" by an expression $\sum_i \tau_i \mathbf{v}_i \mathbf{u}_i^T$. For pseudoinverses, the value τ_i is just σ_i^{-1} (setting $0^{-1} = 0$); whereas for damped least squares, $\tau_i = \sigma_i/(\sigma_i^2 + \lambda^2)$. The pseudoinverse method is unstable as σ_i approaches zero; in fact, it is exactly at singularities that σ_i's are equal to zero.

For values of σ_i which are large compared to λ, the damped least squares method is not very different from the pseudoinverse since for large σ_i, $\sigma_i/(\sigma_i^2 + \lambda^2) \approx 1/\sigma_i$. But, when σ_i is of the same order of magnitude as λ or smaller, then the values σ^{-1} and $\sigma_i/(\sigma_i^2 + \lambda^2)$ diverge. Indeed, for any $\lambda > 0$, $\sigma_i/(\sigma_i^2 + \lambda^2) \to 0$ as $\sigma_i \to 0$. Thus, the damped least squares method tends to act similarly to the pseudoinverse method away from singularities and effectively smooths out the performance of pseudoinverse method in the neighborhood of singularities.

7 Selectively damped least squares

For the material that used to be in this section, see Buss and Kim [7].

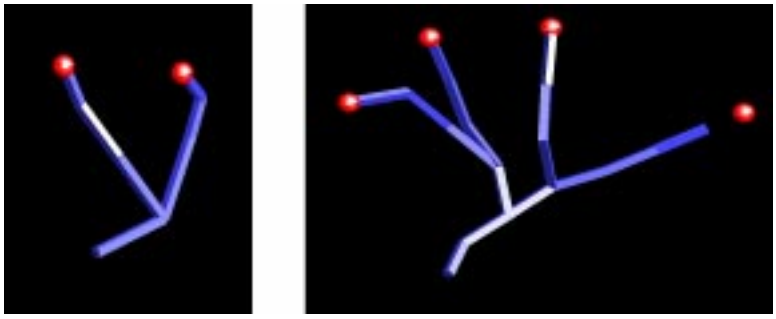

Figure 1: The Y and double-Y shapes. The end effectors are at the ends of the branches; the red balls indicate the target positions.

8 Experimental results and recommendations

For the rest of the material that used to be in this section, plus the results of additional experiments, see Buss and Kim [7].

To compare the IK algorithms, we implemented the "Y"-shaped and "double-Y" shaped multibodies pictured in figure 1. The first has seven links with two end effectors and the latter has 16 links with 4 end effectors. We let the target positions (the red balls in the figures) move in sinusoidally varying curves in and out of reach of the multibodies. The target positions moved in small increments (just large enough to still look visually smooth), and in each time step we updated the joint angles once.[†] Since joint angles were updated only once per time step, the end effectors tracked the target positions only approximately, even when the target positions were within reach. We visually inspected the simulations for oscillations and tracking abilities. We also measured the accuracy of the tracking over a period of hundreds of simulation steps.

The Jacobian transpose had the advantage of being fast, but of poor quality. Its quality was poor for the Y shape and extremely poor for the double-Y shape. However, in other simultations, we have seen the Jacobian transpose method work well for a system with a single end effector.

The pseudoinverse method worked very poorly whenever the target positions were out of reach, and we do not recommend its use unless joint

[†]All our software, including source code, is available from the web page `http://math.ucsd.edu/~sbuss/ResearchWeb/ikmethods`. Short movie clips of the Jacobian transpose, the pure pseudoinverse method, the DLS and the SDLS methods are also available there.

angles are severely clamped with ClampMaxAbs.

The damped least squares method worked substantially better than the Jacobian transpose method, although it is somewhat slower. We attempted to set the damping constant λ so as to minimize the average error of the end effectors's positions, but at the point where the error was minimized, there was a lot of oscillation and shaking. Thus, we had to raise the damping constant until unwanted oscillation became very rare (but at the cost of accuracy in tracking the target positions).

We also implemented a version of the DLS method which uses the ClampMag method to clamp the components of the $\vec{\mathbf{e}}$ vector: this method is called DLS$'$. The advantage of the DLS$'$ method is that the clamping of $\vec{\mathbf{e}}$ reduces oscillation and shaking, and thus a lower damping constant can be used. The lower damping constant allows the multibody to more agressively move towards the target positions.

The runtimes for two different methods are described in the table below. Runtimes are in microseconds and were measured with custom `C++` code on a 2.8GHz Pentium. The DLS$'$ runtime is not reported, but is very close to that of DLS. For the Y-shape, the Jacobian matrix is 6×7, for the double-Y, it is 12×16.

Shape	Jacobian Transpose	DLS
Y	1.1 μs	2.2 μs
Double-Y	6.5 μs	18.5 μs

We conclude with some recommendations. First, the Jacobian transpose performed poorly in our tests, but we have seen it work well in situations where there is a single end effector. For these applications, the Jacobian transpose is fast and easy to implement. For multiple end effectors, the DLS or DLS$'$ methods can be used. For controlled situations where a damping constant can be set ahead of time, the DLS$'$ method gives good performance and relatively easy implementation. For recommendations relating to the use of selectively damped least squares, see [7].

A final recommendation that applies to any method is that it is almost always a good idea to clamp the maximum angle change in a single update to avoid bad behavior from unwanted large instantaneous changes in angles.

References

[1] N. I. Badler, K. H. Manoochehri, and G. Walters, *Articulated*

figure positioning by multiple constraints, IEEE Computer Graphics and Applications, 7 (1987), pp. 28–38.

[2] P. Baerlocher and R. Boilic, *Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures*, PhD thesis, Ecole Polytechnique Federale de Lausanne, 2001.

[3] P. Baerlocher and R. Boulic, *Task-priority formulations for the kinematics control of highly redundant articulated structures*, in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, 1998.

[4] J. Baillieul, *Kinematic programming alternatives for redundant manipulators*, in Proc. IEEE International Conference on Robotics and Automation, 1985, pp. 722–728.

[5] A. Balestrino, G. De Maria, and L. Sciavicco, *Robust control of robotic manipulators*, in Proceedings of the 9th IFAC World Congress, Vol. 5, 1984, pp. 2435–2440.

[6] S. R. Buss, *3-D Computer Graphics: A Mathematical Introduction with OpenGL*, Cambridge University Press, 2003.

[7] S. R. Buss and J. S. Kim, *Selectively damped least squares for inverse kinematics*. Typeset manuscript, April 2004. Draft available at `http:/math.ucsd.edu/~sbuss/ResearchWeb`. Submitted for publication.

[8] S. K. Chan and P. D. Lawrence, *General inverse kinematics with the error damped pseudoinverse*, in Proc. IEEE International Conference on Robotics and Automation, 1988, pp. 834–839.

[9] S. Chiaverini, *Estimate of the two smallest singular values of the jacobian matrix: Applications to damped least-squares inverse kinematics*, Journal of Robotic Systems, 10 (1988), pp. 991–1008.

[10] ——, *Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators*, IEEE Transactions on Robotics and Automation, 13 (1997), pp. 398–410.

[11] S. Chiaverini, B. Siciliano, and O. Egeland, *Review of damped least-squares inverse kinematics with experiments on an industrial robot manipulator*, IEEE Transactions on Control Systems Technology, 2 (1994), pp. 123–134.

[12] K.-J. Choi and H.-S. Ko, *On-line motion retargetting*, Journal of Visualization and Computer Animation, 11 (2000), pp. 223–235.

[13] C. Y. Chung and B. H. Lee, *Torque optimizing control with singularity-robustness fir kinematically redundant robots*, Journal of Intelligent and Robotic Systems, 28 (2000), pp. 231–258.

[14] A. S. Deo and I. D. Walker, *Robot subtask performance with singularity robustness using optimal damped least squares*, in Proc. IEEE International Conference on Robotics and Automation, 1992, pp. 434–441.

[15] ——, *Adaptive non-linear least squares for inverse kinematics*, in Proc. IEEE International Conference on Robotics and Automation, 1993, pp. 186–193.

[16] A. D'Souza, S. Vijayakumar, and S. Schaal, *Learning inverse kinematics*, in Proc. IEEE IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, 2001, pp. 298–303.

[17] M. Girard, *Interactive design of 3D computer-animated legged animal motion*, IEEE Computer Graphics and Applications, 7 (1987), pp. 39–51.

[18] M. Girard and A. A. Maciejewski, *Computational modeling for the computer animation of legged figures*, Computer Graphics, 19 (1985), pp. 263–270. SIGGRAPH'85.

[19] R. Grzeszczuk and D. Terzopoulos, *Automated learning of muscle-actuated locomotion through control abstraction*, in Proc. ACM SIGGRAPH'95, New York, 1995, ACM Press, pp. 63–70.

[20] R. Grzeszczuk, D. Terzopoulos, and G. Hinton, *NeuroAnimator: Fast neural network emulation and control of physics-based models*, in Proc. ACM SIGGRAPH'98, New York, 1998, ACM Press, pp. 9–20.

[21] J. Hodgkins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien, *Animating human athletics*, in Proc. ACM SIGGRAPH'95, New York, 1995, ACM Press, pp. 71–78.

[22] M. I. Jordan and D. E. Rumelhart, *Forward models: supervised learning with a distal teacher*, Cognitive Science, 16 (1992), pp. 307–354.

[23] G. T. Ke, *Solving inverse kinematics constraint problems for highly articulated models*, Master's thesis, University of Waterloo, 2000. Tech. Rep. CS-2000-19.

[24] J. U. Korein and N. I. Badler, *Techniques for generating the goal-directed motion of articulated structures*, IEEE Computer Graphics and Applications, 2 (1982), pp. 71–81.

[25] J. Lander, *Making kine more flexible*, Game Developer, 5 (1998).

[26] ——, *Oh my God, I inverted kine!*, Game Developer, 5 (1998).

[27] G. G. Lendaris, K. Mathia, and R. Sacks, *Linear hopfield networks and constrained optimization*, IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics, 29 (1999), pp. 114–118.

[28] A. Liégeois, *Automatic supervisory control of the configuration and behavior of multibody mechanisms*, IEEE Transactions on Systems, Man, and Cybernetics, 7 (1977), pp. 868–871.

[29] A. A. Maciejewski, *Dealing with the ill-conditioned equations of motion for articulated figures*, IEEE Computer Graphics and Applications, 10 (1990), pp. 63–71.

[30] A. A. Maciejewski and C. A. Klein, *Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments*, International Journal of Robotic Research, 4 (1985), pp. 109–117.

[31] ——, *The singular value decomposition: Computation and applications to robotics*, International Journal of Robotic Research, 8 (1989), pp. 63–79.

[32] R. V. Mayorga, N. Milano, and A. K. C. Wong, *A simple bound for the appropriate pseudoinverse perturbation of robot manipulators*, in Proc. IEEE International Conference on Robotics and Automation, vol. 2, 1990, pp. 1485–1488.

[33] R. V. Mayorga, A. K. C. Wong, and N. Milano, *A fast procedure for manipulator inverse kinematics evaluation and pseudoinverse robustness*, IEEE Transactions on Systems, Man, and Cybernetics, 22 (1992), pp. 790–798.

[34] Y. Nakamura and H. Hanafusa, *Inverse kinematics solutions with singularity robustness for robot manipulator control*, Journal of Dynamic Systems, Measurement, and Control, 108 (1986), pp. 163–171.

[35] D. E. Orin and W. W. Schrader, *Efficient computation of the Jacobian for robot manipulators*, International Journal of Robotics Research, 3 (1984), pp. 66–75.

[36] E. Oyama, N. Y. Chong, A. Agah, T. Maeda, and S. Tachi, *Inverse kinematics learning by modular architecture neural networks with performance prediction networks*, in Proc. IEEE International Conference on Robotics and Automation, 2001, pp. 1006–1012.

[37] C. B. Phillips and N. I. Badler, *Interactive behaviors for bipedal articulated figures*, Computer Graphics, 25 (1991), pp. 359–362.

[38] A. Ramdane-Cherif, B. Daachi, A. Benallegue, and N. Lévy, *Kinematic inversion*, in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002, pp. 1904–1909.

[39] H. Rijpkema and M. Girard, *Computer animation of knowledge-based human grasping*, Computer Graphics, 25 (1991), pp. 339–348. SIGGRAPH'91.

[40] G. Tevatia and S. Schaal, *Inverse kinematics for humanoid robots*, in Proc. IEEE International Conference on Robotics and Automation, vol. 1, 2000, pp. 294–299.

[41] C. W. Wampler, *Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods*, IEEE Transactions on Systems, Man, and Cybernetics, 16 (1986), pp. 93–101.

[42] C. W. Wampler and L. J. Leifer, *Applications of damped least-squares methods to resolved-rate and resolved-acceleration control of manipulators*, Journal of Dynamic Systems, Measurement, and Control, 110 (1988), pp. 31–38.

[43] L.-C. T. Wang and C. C. Chen, *A combined optimization method for solving the inverse kinematics problem of mechanical manipulators*, IEEE Transactions on Robotics and Automation, 7 (1991), pp. 489–499.

[44] C. Welman, *Inverse kinematics and geometric constraints for articulated figure manipulation*, Master's thesis, Simon Fraser University, September 1993.

[45] D. E. Whitney, *Resolved motion rate control of manipulators and human prostheses*, IEEE Transactions on Man-Machine Systems, 10 (1969), pp. 47–53.

[46] W. A. Wolovich and H. Elliot, *A computational technique for inverse kinematics*, in Proc. 23rd IEEE Conference on Decision and Control, 1984, pp. 1359–1363.

[47] T. Yoshikawa, *Dynamic manipulability of robot manipulators*, Journal of Robotic Systems, 2 (1985), pp. 113–124.

[48] ——, *Manipulability of robotic mechanisms*, International Journal of Robotics Research, 4 (1985), pp. 3–9.

[49] J. Zhao and N. I. Badler, *Inverse kinematics positioning using nonlinear programming for highly articulated figures*, ACM Transactions on Graphics, 13 (1994), pp. 313–336.