Inverse Kinematics Problems with Exact Hessian Matrices

Kenny Erleben
University of Copenhagen
kenny@di.ku.dk

Sheldon Andrews
École de technologie supérieure
sheldon.andrews@etsmtl.ca

ABSTRACT

Inverse kinematics (IK) is a central component of systems for motion capture, character animation, motion planning, and robotics control. The field of computer graphics has developed fast stationary point solvers methods, such as the Jacobian transpose method and cyclic coordinate descent. Much work with Newton methods focus on avoiding directly computing the Hessian, and instead approximations are sought, such as in the BFGS class of solvers. This paper presents a numerical method for computing the exact Hessian of an IK system with spherical joints. It is applicable to human skeletons in computer animation applications and some, but not all, robots. Our results show that using exact Hessians can give performance advantages and higher accuracy compared to standard numerical methods used for solving IK problems. Furthermore, we provide code and supplementary details that allows researchers to plug-in exact Hessians in their own work with little effort.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; Motion processing;

KEYWORDS

inverse kinematics, Hessian, optimization

ACM Reference format:
Kenny Erleben and Sheldon Andrews. 2017. Inverse Kinematics Problems with Exact Hessian Matrices. In *Proceedings of MiG '17, Barcelona,Spain, November 8–10, 2017,* 6 pages.
DOI: 10.1145/3136457.3136464

1 INTRODUCTION

Inverse kinematics is the problem of computing the configuration (i.e., joint angles) for a kinematic chain, skeleton, or mechanism, such that an end effector will reach a prescribed goal. IK methods are fundamental for many robotics and computer graphics applications, as evidenced by literature on the topic dating back several decades [Craig 1989; Girard and Maciejewski 1985]. In robotics, the problem is usually phrased as a dynamic system which may lead to different schemes [Hsia and Guo 1991]. An overview of numerical methods used in computer graphics can be found in the report by Buss [2004].

Popular techniques for solving IK problems typically discount the use of exact Hessians, and prefer to rely on approximations of second-order derivatives. However, robotics work has shown that exact Hessians in 2D Lie algebra-based dynamical computations outperforms approximate methods [Lee et al. 2005]. Encouraged by these results, we examine the viability of exact Hessians for inverse kinematics of 3D characters. To our knowledge, no previous work in computer graphics has addressed the significance of using a closed form solution for the exact Hessian, which is surprising since IK is pertinent for many computer animation applications and there is a large body of work on the topic.

This paper presents the closed form solution in a simple and easy to evaluate geometric form using two world space cross-products. Compared to robotics work, we target joints with general Euler angles ordering and use the familiar homogeneous coordinates representation to describe the kinematics. This technical choice is based on the popularity of Euler angles in character animation applications, but the theory we present holds equally well for quaternion representation or other parameterizations of rotation angles. We consider the derivation and algorithm for the computation of the exact IK Hessian to be a novel theoretical contribution. Furthermore, we investigate numerically whether using the exact Hessian with a Newton's method type of approach improves the performance and accuracy for IK problems with many degrees of freedom and large displacements of end-effectors. Our results suggest that using exact Hessians can outperform using approximate Hessians, or even Hessian free methods, under certain conditions.

We supplement this paper with Python and MATLAB code allowing other researchers to harvest the benefits of using exact Hessians in their own IK solvers.

MiG '17, Barcelona,Spain
© 2017 ACM. 978-1-4503-5541-4/17/11...$15.00
DOI: 10.1145/3136457.3136464

2 NUMERICAL METHODS FOR IK

In Zhao and Badler [1994] a mathematical formalism is presented for solving IK as a constrained non-linear optimization problem.

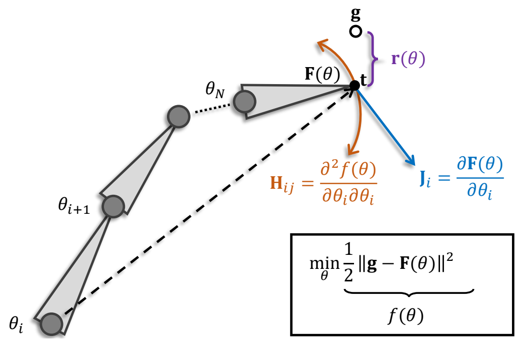
g
$\mathbf{r}(\theta)$
$\mathbf{F}(\theta)$
t
θ_N
θ_{i+1}
θ_i
$\mathbf{H}_{ij} = \frac{\partial^2 f(\theta)}{\partial\theta_i\partial\theta_i}$
$\mathbf{J}_i = \frac{\partial \mathbf{F}(\theta)}{\partial\theta_i}$
$\min_\theta \underbrace{\frac{1}{2}\|\mathbf{g} - \mathbf{F}(\theta)\|^2}_{f(\theta)}$

Figure 1: Conceptualization of the IK problem: find joint angles $\theta_0, \ldots, \theta_N$ that minimize the distance between end-effector position $F(\theta)$ and the goal position g. The distance is encoded by the residual $r(\theta)$. The non-linear optimization problem is solved using the gradient $\nabla f(\theta) = -J^T r$, and optionally the Hessian matrix H.

This work allows for general types of constraints. Furthermore, the optimization model can be used to derive a large spectrum of known methods for solving the IK problem.

Our work too takes as its outset the IK optimization model of Zhao and Badler [1994]. In fact, there exists many methods for solving the IK optimization problem. Engell-Nørregård and Erleben [2011] shows how the Jacobian Transpose (JT) method is equivalent to a steepest descent (SD) method without a line-search. Further, they show that damped least squares (DLS) is obtained from a Gauss-Newton (GN) approach, and the damping results in a Levenberg-Marquardt (LM) type method. Other quasi-Newton type methods are readily available such as memory limited BFGS [Zhao and Badler 1994] and a wide range of methods using iterative methods like preconditioned conjugate gradient (PCG) for solving the sub-system or approximate the Hessians using low-rank updates. Even cyclic-coordinate descent (CCD) [Wellman 1993] can be derived from using a matrix-splitting approach for solving the gradient equation that forms the basis for the JT method. Interestingly, solving the gradient equation with pseudo-inverses yields a system of equations similar to the Newton systems that form the basis for GN/LM type of methods. Hence, these quasi-Newton methods are linked to the JT method. Recent work by Harish et al. [2016] investigates how to parallelize a DLS approach for solving the IK problem. The work focuses on creating a parallel line-search that can help eliminate the need for the many iterations that JT method usually requires to converge for highly nonlinear motions with many degrees of freedom. This is expected from convergence theorems stating that JT/SD method has linear convergence rate [Nocedal and Wright 1999]. However, it has also been observed in practice [Callennec 2006]. Other work incorporates higher order information, mimicking the effect of a Hessian, by nonlinear conjugate gradients and adding non-smoothness allows a full coupling of joint limits too [Engell-Nørregård and Erleben 2009].

In Fedor [2003], three methods are revisited and evaluated for computer game usage: an algebraic method, Cyclic-Coordinate-Descent [Wellman 1993], and a Newton-Raphson method. The Newton-based method is used for complex manipulation and claimed to give the most realistic looking poses, but it is the slowest. However, joint limits were not dealt with in this work. Other formulations have been investigated for instance in Ho et al. [2005] the problem is solved using linear programming.

Finally, Zordan [2002] notes the importance of tuning parameters for numerical methods in order to achieve good performance, and this is a relevant step in our experiments.

3 THEORY

We begin by formally defining the IK problem of a serial chain mechanism. Without loss of generality, we ignore branched structures and closed loops in this formalism and limit our presentation to serial chains. Figure 1 conceptualizes many of the details we discuss in this section.

Let the *tool vector* in a serial chain be given by $\mathbf{t} \in \mathbb{R}^3$. We assume the serial chain is made of rigid links. The tool vector is placed in the last frame of the chain, which we call the end-effector frame. Note that we adopt robotics terminology throughout this section, and conceptually interpret $\mathbf{t}$ as the location of a tool being held by a robotic manipulator. However, in the context of computer graphics applications, $\mathbf{t}$ can be interpreted as a point with fixed position relative to a bone in a skeleton armature (e.g. an optical marker).

Each link describes a coordinate frame with respect to the parent link, or bone. The parent frame of the root link is the world coordinate system. Hence, a link is equivalent to a transformation of a point $\mathbf{p}$ in the current coordinate frame k to the coordinate frame of its parent $k-1$, and we write the transformation as

$$\begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_{k-1} = \mathbf{T}_k \begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_k . \tag{1}$$

The subscripts on the homogeneous vectors denote the frame of reference with the world coordinate system (WCS) assigned index $k = -1$.

The homogeneous transform matrix $\mathbf{T}_k$ is parameterized with the Euler angles α_k, β_k, and γ_k and position of the joint. We use homogeneous coordinates in deriving our equations and when we present our final results we implicitly homogenize the formulas. Also, our derivations use ZYZ Euler angle order to be specific and make derivations less dense. However, in the code and supplementary material we generalize to any Euler angle convention and extend our formulation to account for a translational moving root bone, which is important for working with motion capture data of human characters.

Given an N link serial chain and a tool vector $\mathbf{t}$ and known joint parameters, we wish to compute the tool vector position in the world coordinate system $\begin{bmatrix}\mathbf{t}^T & 1\end{bmatrix}^T_{WCS}$. This is called the end-effector position, and is given by

$$\begin{bmatrix}\mathbf{e}\\1\end{bmatrix} \equiv \begin{bmatrix}\mathbf{t}\\1\end{bmatrix}_{WCS} \equiv \mathbf{T}_0\,\mathbf{T}_1 \cdots \mathbf{T}_N \begin{bmatrix}\mathbf{t}\\1\end{bmatrix}_N . \tag{2}$$

The matrix concatenation in (2) defines the end effector function $\mathbf{F}(\theta)$ where $\theta = [\alpha_0\ \beta_0\ \gamma_0\ \alpha_1\ \beta_1\ \gamma_1\ \ldots \alpha_N\ \beta_N\ \gamma_N]^T$ is termed the joint angles or joint parameters.

We now have enough notation and terminology in place to define what we understand of the unconstrained inverse kinematics problem.

3.1 Non-linear optimization

Given a desired goal position $\mathbf{g} \in \mathbb{R}^3$, we seek to find the joint angles θ^* that minimize the distance between the goal position and the end-effector position

$$\theta^* = \arg\min_{\theta} \frac{1}{2} \left\| \begin{bmatrix}\mathbf{g}\\1\end{bmatrix} - \mathbf{F}(\theta) \right\|^2 \tag{3}$$

which defines the IK objective function $f(\theta)$. Joint angle limits are added easily using box constraints

$$\theta^* = \arg \min_{\mathbf{l} \le \theta \le \mathbf{u}} f(\theta) , \tag{4}$$

where $\mathbf{l} < \mathbf{u}$ are lower and upper bounds on the joint angles. We note that the IK problems in (3) or (4) are non-linear problems and require specialized iterative numerical methods that utilize the gradient, and possibly the Hessian, of $f(\theta)$. One may apply Newton's method in a root search setting, such that

$$\begin{bmatrix}\mathbf{g}\\1\end{bmatrix} - \mathbf{F}(\theta) = \mathbf{0} . \tag{5}$$

This defines a residual vector, $\mathbf{r}(\theta) = [\mathbf{g}^T\ 1]^T - \mathbf{F}(\theta)$, that is related to the minimization form by $f(\theta) \equiv \frac{1}{2}\|\mathbf{r}(\theta)\|^2$. Solving the residual equation (5) using an iterative Newton method implies solving the equation

$$\mathbf{J}\,\Delta\theta = \mathbf{r}(\theta) \tag{6a}$$

where $\mathbf{J} = \frac{\partial \mathbf{F}(\theta)}{\partial \theta}$ is the Jacobian matrix. A steepest descent method for solving the optimization problem in equation (3) iteratively updates the current iterate θ^k with the gradient $\nabla f(\theta^k)$ as the search direction and a step length α, to give the next iterate

$$\theta^{k+1} = \theta^k + \alpha \nabla f(\theta^k)\,. \tag{7}$$

The gradient of the IK minimization formulation is given as

$$\nabla f(\theta) = -\frac{\partial \mathbf{F}}{\partial \theta}^T \mathbf{r}(\theta) = -\mathbf{J}^T\mathbf{r}. \tag{8}$$

From equation (6a) and (8) it shows that an efficient method for computation of matrix $\mathbf{J}$ is desired.

Iterative solvers based on Newton's method use second derivative information contained in the Hessian matrix $\mathbf{H}$ by solving the linear system

$$\mathbf{H}\,\mathbf{p} = -\nabla f(\theta), \tag{9}$$

which is the form used in our experiments. Here, the descent direction $\mathbf{p}$ is used to update the current solution by means of a line search with step size α, such that $\theta \leftarrow \theta + \alpha\mathbf{p}$. The (i, j) entry of the Hessian is given by

$$\mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial\theta_j \partial\theta_i} = \mathbf{J}_j^T\mathbf{J}_i - \mathbf{K}_{i,j}^T\,\mathbf{r}\,, \tag{10}$$

where $\mathbf{K}_{i,j} = \frac{\partial \mathbf{J}_i}{\partial \theta_j}$. We use a subscript on $\mathbf{J}$ to denote respective columns of the matrix. Here, $\mathbf{K}$ is in fact a third order tensor. If we used tensor algebra notation and double contraction operator : then this can be written compactly as

$$\mathbf{H} = \mathbf{J}^T\mathbf{J} - \mathbf{K} : \mathbf{r}\,. \tag{11}$$

Previous work has argued to ignore the term $\mathbf{K} : \mathbf{r}$, leading to Gauss-Newton type methods, or using iterative techniques to approximate $\mathbf{H}$, such as the popular BFGS method. In this work, we examine computing the exact Hessian.

3.2 Computing the Jacobian

Consider that for the rotational degrees of freedom in a kinematic chain, the ith index of θ can be either the α, β or γ angle of the kth joint. That is,

$$\theta_i \in \{\alpha_k, \beta_k, \gamma_k\}\ . \tag{12}$$

The instantaneous change in the end-effector position due to rotation about joint axes α_k, β_k, or γ_k involves the cross-product of the joint axes in the world coordinate system. The IK Jacobian for each rotational DOF is therefore

$$\frac{\partial \mathbf{F}(\theta)}{\partial \alpha_k} = \left[\mathbf{k}\right]_{WCS} \times \Delta\mathbf{p}_{WCS}\,, \tag{13a}$$

$$\frac{\partial \mathbf{F}(\theta)}{\partial \beta_k} = \left[\mathbf{R}_Z(\alpha_k)\mathbf{j}\right]_{WCS} \times \Delta\mathbf{p}_{WCS}\,, \tag{13b}$$

$$\frac{\partial \mathbf{F}(\theta)}{\partial \gamma_k} = \left[\mathbf{R}_Z(\alpha_k)\mathbf{R}_Y(\beta_k)\mathbf{k}\right]_{WCS} \times \Delta\mathbf{p}_{WCS} \tag{13c}$$

where $\Delta\mathbf{p}_{WCS} = \left(\mathbf{e} - \left[\mathbf{t}_k\right]_{WCS}\right)$ is the vector difference between the end-effector position and the position of the joint origin in the world coordinate system, and we make use of the unit-axis vectors $\mathbf{i} = [1\ 0\ 0]^T$, $\mathbf{j} = [0\ 1\ 1]^T$, $\mathbf{k} = [0\ 0\ 1]^T$.

For the translation degrees of freedom $\{\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z\}$ of a moving root bone, it has the nice property that the bone vector lives in the world coordinate system and will be independent of any rotational degree of freedom in the system. Hence, the IK Jacobian is simply

$$\frac{\partial \mathbf{F}(\theta')}{\partial \mathbf{t}_x} = \mathbf{i}\,,\ \frac{\partial \mathbf{F}(\theta')}{\partial \mathbf{t}_y} = \mathbf{j}\,,\ \frac{\partial \mathbf{F}(\theta')}{\partial \mathbf{t}_z} = \mathbf{k}\,. \tag{14}$$

Further details about deriving (13) and (14) are provided in the supplementary material.

3.3 Computing the Hessian

We observe from equation (10) that we seek an efficient way to compute the third order tensor $\mathbf{K}$, which is defined as

$$\mathbf{K}_{i,j} \equiv \frac{\partial}{\partial\theta_j}\left(\frac{\partial \mathbf{F}}{\partial \theta_i}\right) = \mathbf{X}_0^{h-1}\frac{\partial \mathbf{T}_h}{\partial \theta_j}\mathbf{X}_{h+1}^{k-1}\frac{\partial \mathbf{T}_k}{\partial \theta_i}\mathbf{X}_{k+1}^{N}\begin{bmatrix}\mathbf{t}\\1\end{bmatrix}_N\ . \tag{15}$$

Here we assume $\theta_i \in \{\alpha_k, \beta_k, \gamma_k\}$ and $\theta_j \in \{\alpha_h, \beta_h, \gamma_h\}$ and the homogeneous matrix concatenation is defined as

$$\mathbf{X}_a^b \equiv \mathbf{T}_a, \mathbf{T}_{a+1}\cdots\mathbf{T}_{b-1}\mathbf{T}_b$$

for $a \le b$. We notice that (15) involves a recursive application of the transformation rules. The first derivative will change the position into a vector direction and apply a cross-product with the instantaneous joint axis from bone k. The second derivative will add a cross-product with the instantaneous joint axis from bone h. Hence, we may now immediately write up a world-coordinate system equation for computing $\mathbf{K}_{i,j}$.

Let the instantaneous joint axis of bone k be given by

$$\mathbf{v} = \begin{cases}\mathbf{k} & \text{if } \theta_i = \alpha_k\\ \mathbf{R}_Z(\alpha_k)\mathbf{j} & \text{if } \theta_i = \beta_k\\ \mathbf{R}_Z(\alpha_k)\mathbf{R}_Y(\beta_k)\mathbf{k} & \text{if } \theta_i = \gamma_k\\ \mathbf{0} & \text{otherwise}\end{cases} \tag{16}$$

and similarly for bone h by

$$\mathbf{w} = \begin{cases}\mathbf{k} & \text{if } \theta_j = \alpha_h\\ \mathbf{R}_Z(\alpha_h)\mathbf{j} & \text{if } \theta_j = \beta_h\\ \mathbf{R}_Z(\alpha_h)\mathbf{R}_Y(\beta_h)\mathbf{k} & \text{if } \theta_j = \gamma_h\\ \mathbf{0} & \text{otherwise}\end{cases}\ . \tag{17}$$

This gives

$$\mathbf{K}_{i,j} = \mathbf{w}_{WCS} \times (\mathbf{v}_{WCS} \times \Delta\mathbf{p}_{WCS}) = \mathbf{w}_{WCS} \times \mathbf{J}_i\,, \tag{18}$$

where $\mathbf{v}_{WCS} = \mathbf{R}_{WCS}^k\mathbf{v}$ and $\mathbf{w}_{WCS} = \mathbf{R}_{WCS}^h\mathbf{w}$ are the instantaneous joint axes transformed in the world coordinate system.

Finally, assuming the first three degrees of freedom are reserved for the root translation, then $\mathbf{K}_{i,j} = \mathbf{0}$ if $i \in \{0, 1, 2\}$ or $j \in \{0, 1, 2\}$.

3.4 Other implementation details

A tree branched IK structure essentially consists of several serial chains that all share the same root and may have overlapping "trunks" of the tree structure in common. Such an IK structure is easily created by traversal over all bones $\mathcal{S}$ to identify end-effector bones, followed by a back-traversal from each end-effector to the root. This results in the end-effectors $\mathcal{L}$ and set of kinematic chains C. This process is run once at initialization, or whenever the tree-structure changes its connectivity or becomes re-rooted. A recursive traversal can then be used to perform forward kinematics and compute the world orientations and positions of all bones This step should be invoked prior to any Jacobian or Hessian computation. Also observe that in order to get the IK gradient one must use the result of the Jacobian in (8).

4 RESULTS

This section presents comparisons of tracking real motion data using quasi-Newton and exact Hessian-based methods. All experiments are performed on a Windows PC with Intel i7 2.80 GHz processor and 32 GB of memory.

A MATLAB implementation of the objective function, and Jacobian and Hessian computations is used in our evaluation. The exact Newton method uses the interior-reflective trust region algorithm [Coleman and Li 1996], and the quasi-Newton method used the BFGS algorithm [Broyden 1970; Fletcher 1970]. The nonlinear optimization is performed using the built-in *fminunc* function, or *fmincon* in the case of a constrained optimization. We also compare our results for unconstrained optimization with the LM algorithm, which uses the Hessian approximation $\mathbf{J}^T\mathbf{J}$ and is equivalent to the damped least squares method proposed in Buss [2004].

4.1 Motion capture data

We evaluate the performance of solving IK problems using motion examples from the HDM05 database [Müller et al. 2007]. This database provides skeletal motion and optical marker trajectories for a variety of motion classes. Examples of motions contained in this database are shown in Figure 2.

A total of 41 optical marker trajectories sampled at 120 Hz are used for tracking. We solve for the motion of a skeleton with 58 degrees of freedom (DOF). The skeleton initialization and forward kinematic updates are done using code provided with the database.

4.1.1 Unconstrained optimization. Figure 3 shows the convergence when performing the unconstrained optimization in (3) for selected frames of motion. The frames contain three types of motion: walking, kicking, and punching. In each case, the joint angles and root position of the previous frame of motion were used as the initial solution to the IK solver in order to give a reasonable initial estimate for both the gradient and Hessian since we found that both methods exhibited linear or sub-linear convergence when initialized with a configuration dissimilar to the current frame.

The average timings to solve for the examples shown in Figure 3 using $k_{max} = 100$ are 2.6 s for the quasi-Newton BFGS solver and 19.2 s for trust region with an exact Hessian with. In the latter case, the longer timing is due to the overhead of computing the matrix of second-order partial derivatives. However, using a stopping tolerance $f(\theta) < 10^{-2}$, we observe the average timings shown in Table 1. For Newton's method based algorithms it's clear that the exact Hessian gives a much lower residual with fewer iterations and less wall-clock time. However, the LM algorithm clearly gives the fastest performance when a good initial solution is provided.

Type	quasi-Newton	Exact Hessian	LM
Walking	5.30 s (24,383 calls)	0.59 s (7 calls)	0.25 s (34 calls)
Punching	6.05 s (49,333 calls)	2.54 s (226 calls)	0.49 s (167 calls)
Kicking	4.68 s (27,184 calls)	0.64 s (10 calls)	0.29 s (54 calls)

Table 1: Mean computation time & function calls per frame for motions in Figure 3 with stopping tolerance $f(\theta) < 10^{-2}$.

Motion seq.	quasi-Newton	Exact Hessian	Exact Hessian
(HDM05)	$k_{max} = 100$	$k_{max} = 10$	$k_{max} = 10$
dg_03-05_02	3.11 s / 36.1 cm	0.54 s / 2.5 cm	0.92 s / 0.3 cm

Table 2: The mean computation time and error per frame for solving a complete motion sequence with stopping tolerance $f(\theta) < 10^{-2}$ and prescribed maximum iterations k_{max}.

4.1.2 Constrained optimization. Figure 4 shows the convergence when performing the constrained optimization in (4), which accounts for joint limits. Compared to the unconstrained convergence shown in Figure 3, the constrained optimizer seems to give slightly performance in some instances, which is surprising. We hypothesize that the box constraints can sometimes help to avoid solutions where indefiniteness or singularities may occur, and this is a point for future investigation.

4.1.3 Initial solution. Here we set $\theta = \mathbf{0}$ as the initial solution and re-solve for the motion frames used in our previous analysis. As shown by Figure 5, the convergence across all of the tested methods becomes worse. Specifically, the trust region method with exact Hessian exhibits linear, or even sub-linear, convergence early on. The quasi-Newton and LM algorithms are also unable to achieve a low error solution in many of the example frames. The LM algorithm tends to get stuck in a local minimum when not initialized with a good solution.

The trust region with exact Hessian was the only method to consistently reach low error residual in instances of poor initial solution. We note that as progress is made towards a solution, the expected quadratic convergence behavior is observed.

4.1.4 Reconstructing motion sequence. We evaluate the IK solvers when reconstructing a complete motion sequence contained more than several thousand frames. The average timing and error per frame can be found in Table 2, where the error is measured as the sum of the difference between the real and reconstructed marker positions (in cm). A video of the reconstructed motions, with side-by-side comparison to the original motion from the HDM05 database, can be found in the supplementary material.

The exact Newton solver with $k_{max} = 10$ gives low error when fitting to marker data and qualitatively the best reconstruction. The exact method typically required fewer than 4 iterations to achieve the prescribed error of $f(\theta) < 10^{-2}$. The quasi-Newton BFGS method with $k_{max} = 10$ does a very poor job of reconstructing the motion, and for some frames the residual is quite high. Increasing the maximum iteration count to $k_{max} = 100$ improved the reconstruction, but still does not achieve the low error of the exact method, even through the computational time is comparable.

4.1.5 Tuning the optimization. Here we delve into some specific details about the trust region and BFGS algorithm used in our experiments. While the BFGS method gave consistently similar rates of convergence, we found that the trust region method was particularly sensitive to selection of parameters related to the PCG algorithm that is used to solve the trust region subproblem.

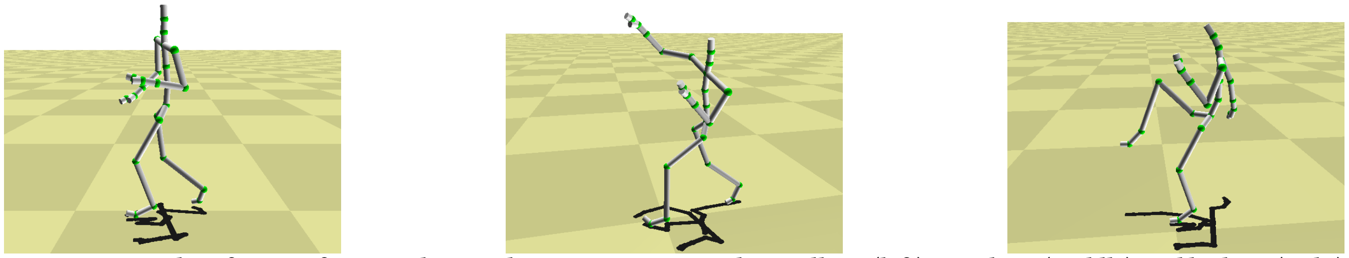
Figure 2: Exemplary frames of motion data used in our comparison: slow walking (left), punching (middle), and kicking (right).

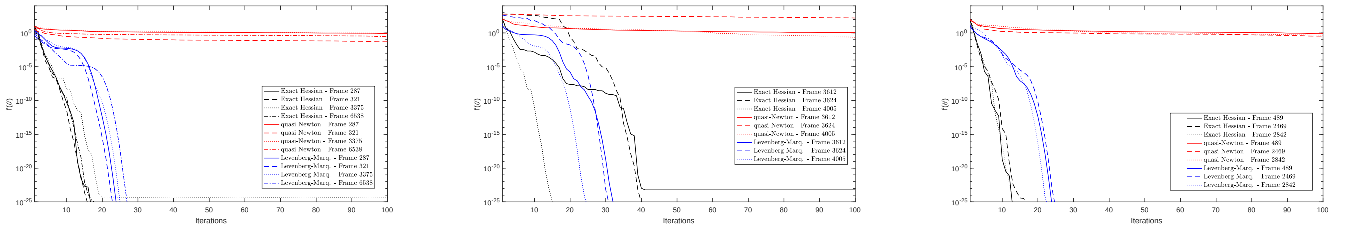

Figure 3: Convergence of the unconstrained optimization in (3) **for frames selected from the *dg_03-02_01* motion in the HDM05 database. Three types of motion are used: walking (left), punching (middle), and kicking (right).**

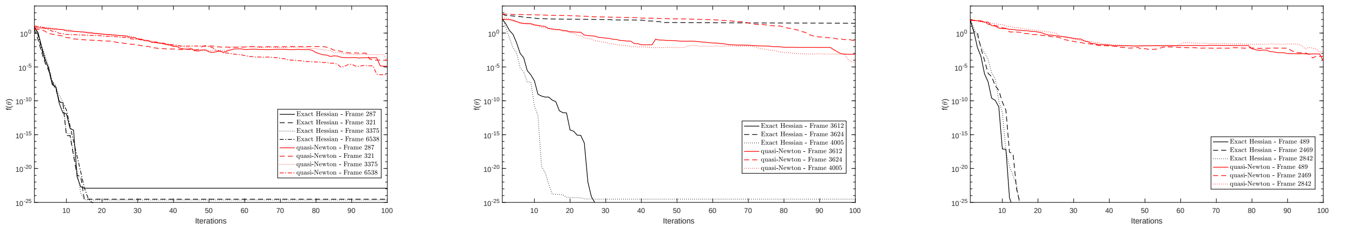

Figure 4: Convergence of the constrained optimization in (4) **for frames selected from the *dg_03-02_01* motion in the HDM05 database. Three types of motion are used: walking (left), punching (middle), and kicking (right).**

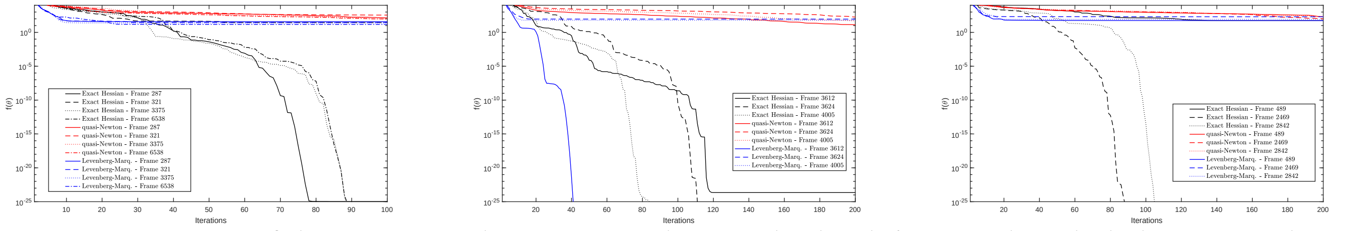

Figure 5: Convergence of the unconstrained optimization when initialized with $\theta = 0$. Both methods demonstrate slower convergence compared to initializing with the joint angles from the previous frames. In particular, the exact Newton method demonstrates linear convergence early on. Note that $k_{max} = 200$ was used to generate some of these plots since more iterations were required to observe convergence behavior.

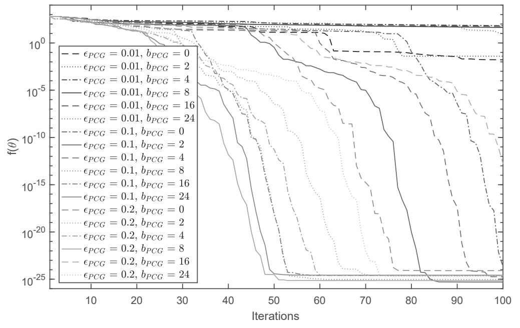

Figure 6: Evaluating PCG stopping tolerance ϵ_{PCG} and preconditioner bandwidth b_{PCG}.

The default Jacobi, or diagonal, preconditioner is simple and efficient to compute. However, it assumes that the Hessian is diagonally dominant, which is not guaranteed in our case. We found that using a larger bandwidth for the preconditioning matrix helped convergence in some instance Furthermore, we found that a lower stopping tolerance for the PCG iterations could sometimes lead to needless exploration of regions of the solution space, especially when the Hessian matrix was ill conditioned or indefinite. This occurred most often when the residual was large (e.g. $\|r(\theta)\| > 1$). This is not unexpected since our objective function is essentially a sum of squared errors function, and the Hessian will only be positive definite near the final solution. In such cases, setting a low function tolerance was detrimental and gave linear or sub-linear rate of convergence, similar to what we observed in Figure 5.

We tune these parameters empirically by sweeping the function tolerance $\epsilon_{PCG} \in [0.01, 0.1, 0.2]$ and preconditioner bandwidth $b_{PCG} \in [0, 2, 4, 8, 16, 24]$. Figure 6 shows the convergence when running this sweep for a selected frame of motion. We found that $\epsilon_{PCG} = 10^{-1}$ and $b_{PCG} = 24$ give overall the best performance, and therefore we use these values in all of our experiments.

We also observed that the sub-solver sometimes required many iterations to converge. This is not unexpected, since there is no guarantee that the Hessian matrix is well-conditioned. In fact, sometimes the condition number of the Hessian can become quite large (larger than 10^5). We therefore give the PCG solver adequate opportunity to solve the sub-problem by setting the maximum iterations to M^2, where M is the number of skeletal degrees of freedom.

Finally, in cases where the Hessian matrix is indefinite, we found that applying the technique described in Higham [1988] to compute a "nearby" semi-positive definite Hessian matrix was useful. For the results in Figure 5, this helped to improve convergence when the initial solution was not close to the true solution.

5 DISCUSSION AND CONCLUSION

The computation of exact Hessians for solving IK problems is often dismissed based on conventional pragmatism, but rarely examined in detail. In this paper and the accompanying supplementary material, we have presented a mathematical framework for computing exact Hessians which is useful for computer graphics researchers. We have also evaluated the benefits of including exact Hessians based on experimentation with real-world motion capture data.

We observe from our convergence plots in Figure 3 and Figure 4 that with fewer than 10 iterations, the exact method usually achieves several orders of magnitude better accuracy than the quasi-Newton method. For Newton's method class of algorithms, the added information from the exact Hessian results in very few required iterations to achieve a specified accuracy, and in many cases the low number of iterations outweighs the per iteration cost of computing the exact Hessian. Hence, one gets a faster overall numerical method.

However, the LM algorithm achieves comparable convergence at much less computational cost, but as Figure 5 illustrates convergence to a low error solution is dependent on having a good initial solution. All methods benefit when temporal coherence can be exploited (i.e. using joint angles from the previous frame of motion), but only the exact Hessian method consistently give good accuracy when a poor initial solution was provided. In our experiments, the Hessian approximation methods could often not reach the same level of accuracy in these instances, even after more than 200 iterations.

We note that our MATLAB implementation is unoptimized, and variables shared during computation of the Jacobian and Hessian are recomputed. Computing these just once and reusing their value would certainly improve computational performance. Performance could also be improved by realizing a parallelized implementation of the Jacobi and Hessian computations. Currently, a single thread is used to compute matrices and minimize the objective function. We note that although computing the Hessian has quadratic time complexity, the for-loops in this algorithm are straightforward to parallelize. Hence, a parallel numerical method with exact Hessians will have low computational time constants. In fact, the speedup factor of computing the Hessian is likely to be in favor of a GPU as the fill-in of the Hessian is one order of magnitude larger than the data on which its computation depends on. Hessian free and approximate methods do not have this potential for speed-up.

As noted in Section 4.1.5, the exact Newton method regularly encounters solutions where the Hessian matrix is indefinite and we proposed using a technique to enforce semi-positive definiteness. Future work will investigate the conditions under which this happens and how to avoid them. A hybrid approach that switches between exact and approximate modes would also be an interesting research trajectory to explore. For instance, choosing a gradient-based, quasi-Newton, or exact Newton method based-on performance expectations given the current solution and residual.

ACKNOWLEDGMENTS

Thanks to Jernej Barbič and Yili Zhao for releasing their ASF/AMC viewer, which we use to visualize some of our results.

REFERENCES

Charles George Broyden. 1970. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics* 6, 1 (1970), 76–90.

S.R. Buss. 2004. *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods.* Technical Report.

B. Le Callennec. 2006. *Interactive Techniques for Motion Deformation of Articulated Figures Using Prioritized Constraints.* Ph.D. Dissertation. École Polytechnique Fédérale de Lausanne (EPFL).

Thomas F Coleman and Yuying Li. 1996. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on optimization* 6, 2 (1996), 418–445.

John J. Craig. 1989. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Morten Engell-Nørregård and Kenny Erleben. 2009. A Projected Non-linear Conjugate Gradient Method for Interactive Inverse Kinematics.. In *MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling.*

Morten Engell-Nørregård and Kenny Erleben. 2011. Technical Section: A Projected Back-tracking Line-search for Constrained Interactive Inverse Kinematics. *Comput. Graph.* 35, 2 (April 2011), 288–298. https://doi.org/10.1016/j.cag.2010.12.011

Martin Fedor. 2003. Application of inverse kinematics for skeleton manipulation in real-time. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics.* ACM Press, New York, NY, USA, 203–212.

Roger Fletcher. 1970. A new approach to variable metric algorithms. *The computer journal* 13, 3 (1970), 317–322.

Michael Girard and A. A. Maciejewski. 1985. Computational Modeling for the Computer Animation of Legged Figures. *SIGGRAPH Comput. Graph.* 19, 3 (July 1985), 263–270. https://doi.org/10.1145/325165.325244

Pawan Harish, Mentar Mahmudi, Benoît Le Callennec, and Ronan Boulic. 2016. Parallel Inverse Kinematics for Multithreaded Architectures. *ACM Trans. Graph.* 35, 2, Article 19 (Feb. 2016), 13 pages. https://doi.org/10.1145/2887740

Nicholas J Higham. 1988. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications* 103 (1988), 103–118.

Edmond S. L. Ho, Taku Komura, and Rynson W. H. Lau. 2005. Computing inverse kinematics with linear programming. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology.* ACM, New York, NY, USA, 163–166.

T. C. Hsia and Z. Y. Guo. 1991. New inverse kinematic algorithms for redundant robots. *Journal of Robotic Systems* 8, 1 (1991), 117–132.

Sung-Hee Lee, Junggon Kim, F. C. Park, Munsang Kim, and J. E. Bobrow. 2005. Newton-Type Algorithms for Dynamics-Based Robot Movement Optimization. *IEEE Transactions on Robotics* 21, 4 (Aug 2005), 657–667. https://doi.org/10.1109/TRO.2004.842336

M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. 2007. *Documentation Mocap Database HDM05.* Technical Report. Universität Bonn.

Jorge Nocedal and Stephen J. Wright. 1999. *Numerical optimization.* Springer-Verlag, New York. xxii+636 pages.

Chris Wellman. 1993. *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation.* Master's thesis. Simon Fraser University.

Jianmin Zhao and Norman I. Badler. 1994. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Trans. Graph.* 13, 4 (1994), 313–336.

Victor B Zordan. 2002. *Solving computer animation problems with numeric optimization.* Technical Report. Georgia Institute of Technology.