Published in final edited form as:
J Robot Syst. 2005 November ; 22(11): 691–710.

From task parameters to motor synergies: A hierarchical framework for approximately-optimal control of redundant manipulators

Emanuel Todorov[1], **Weiwei Li**[2], and **Xiuchuan Pan**[1]

1 *Department of Cognitive Science, University of California San Diego*

2*Department of Mechanical and Aerospace Engineering, University of California San Diego*

Abstract

We present a hierarchical framework for approximately-optimal control of redundant manipulators. The plant is augmented with a low-level feedback controller, designed to yield input-output behavior that captures the task-relevant aspects of plant dynamics but has reduced dimensionality. This makes it possible to reformulate the optimal control problem in terms of the augmented dynamics, and optimize a high-level feedback controller without running into the curse of dimensionality. The resulting control hierarchy compares favorably to existing methods in robotics. Furthermore we demonstrate a number of similarities to (non-hierarchical) optimal feedback control. Besides its engineering applications, the new framework addresses a key unresolved problem in the neural control of movement. It has long been hypothesized that coordination involves selective control of task parameters via muscle synergies, but the link between these parameters and the synergies capable of controlling them has remained elusive. Our framework provides this missing link.

1 Introduction

The control of complex redundant systems is a challenging problem, of interest in both robotics and biological motor control. The nonlinear dynamics and high-dimensional state and control spaces of such systems prevent the use of many traditional methods for controller design. One way to tackle very complex problems is through divide-and-conquer strategies. Indeed, the most advanced control system known to date –the nervous system –appears to rely on such strategies. Sensorimotor control occurs simultaneously on many levels [1–3]. Lower-level circuits interact with the musculoskeletal system directly: they receive rich sensory input, and generate corresponding motor output before the rest of the brain has had time to react to that input. Higher-level circuits interact with an augmented plant, that consists of the lower levels and the musculoskeletal system. The lower levels perform a (not well understood) transformation, allowing higher levels to operate on increasingly more abstract and more goal-related movement representations [4].

Here we propose a hierarchical control scheme inspired by this general organization of the sensorimotor system, as well as by prior work on hierarchical control in robotics [5–7]. We focus on two-level feedback control hierarchies as illustrated in Fig 1. The low-level controller receives information about the plant state $\mathbf{x}$, and generates an abstract and more compact state representation $\mathbf{y}(\mathbf{x})$ that is sent to the high level. The high-level controller monitors task progress, and issues commands $\mathbf{v}(\mathbf{y})$ which in general specify how $\mathbf{y}$ should change. The job of the low-level controller is to compute energy-efficient controls $\mathbf{u}(\mathbf{v}, \mathbf{x})$ consistent with $\mathbf{v}$. Thus the low-level controller does not solve a specific sub-task (as usually assumed in hierarchical reinforcement learning [8,9]), but instead performs an instantaneous feedback transformation. This enables the high level to control $\mathbf{y}$ unencumbered by the full details of the plant.

While the proposed scheme is designed to resemble the sensorimotor system on a structural level, achieving functional resemblance is equally important. Functionally, sensorimotor control is best described as being near-optimal [10]. It may seem surprising that a hierarchical controller can closely approximate an optimal controller. But as we have shown elsewhere [11,12], optimal feedback controllers for redundant systems exhibit hierarchical organization similar to Fig 1, even when such organization is not imposed by design. This finding provides yet another motivation for the present scheme: if full-blown optimization on redundant tasks is known to yield hierarchical structure, it makes sense to restrict the optimization to an (appropriately chosen) family of hierarchical controllers.

The general idea that the brain monitors a small number of task parameters **y** instead of the full state **x**, generates abstract commands **v**, and maps them into muscle activations **u** using motor synergies, has been around for a long time [26,28]. A number of concrete models of end-effector control have been formulated in the context of reaching tasks [29–34]. The high-level state in such models is assumed to be hand position, the abstract command is desired velocity in hand space or in joint space, and the high-level controller is a simple positional servo. While these models are related to our work, in some sense they leave all the hard questions unanswered: it is unclear how the task parameters are actually controlled (i.e. what the corresponding muscle synergies are), and whether this choice of task parameters can yield satisfactory performance. We address these questions here.

Our framework is related in interesting ways to input-output feedback linearization [13,14] as well as to the operational space formulation [6] – which also fit in the general scheme in Fig 1. These methods yield linear dynamics on the high level, by cancelling the plant nonlinearities at the low level. However, many systems of interest cannot be linearized, and furthermore it is not clear that linearization is desirable in the first place. Suppressing the natural plant dynamics may require large control signals –which are energetically expensive, and also increase error in systems subject to control-multiplicative noise (which is a universal characteristic of biological movement [15–17]). In contrast, we summarize the plant dynamics on the high level and thus create opportunities for exploiting them. Recent work in biped locomotion [18] underscores the potential of such approaches. In general, our objective is dimensionality reduction rather than linearization. This is because we are pursuing optimality, and what makes optimal control hard is the curse of dimensionality. We are also pursuing neurobiological relevance, and it is clear that a number of behaviors (including locomotion [19] and arm movements [20]) are optimized with respect to the nonlinear musculoskeletal dynamics.

The rest of the paper is organized as follows. Section 2 summarizes our previous findings relating optimal feedback control to hierarchical control, introduces relevant concepts from biological motor control, and illustrates them with experimental data. This motivates the formal development of the new framework in Section 3. The relation to existing methodologies is elaborated in Section 4. Numerical comparisons to robotic control methods as well as to optimal control are presented in Section 5. Preliminary results have been reported in conference proceedings [21].

2 Biological motivation, and relation to optimal control

Optimal control models of biological movement have a long history, and provide satisfying computational accounts of many behavioral phenomena [10]. The majority of these models are formulated in open loop, and plan a desired trajectory while ignoring the role of online feedback. Desired trajectory planning implies that the neural processing in the mosaic of brain areas involved in online sensorimotor control does little more than play a prerecorded movement tape –which is unlikely [4]. Consequently, we and others [11,22–25] have focused

on optimal feedback control models, that predict not only average behavior but also the task-specific sensorimotor contingencies used to generate intelligent adjustments online. Such adjustments enable biological systems to "solve a control problem repeatedly rather than repeat its solution" [26], and thus afford remarkable levels of performance in the presence of noise, delays, internal fluctuations, and unpredictable changes in the environment. The latter models have significantly extended the range of phenomena addressed by open-loop optimization [10]. Some of the key results from this line of work are directly relevant to hierarchical control, as described next.

2.1 Minimal intervention, synergies, and uncontrolled manifolds

Our work on motor coordination [11,12] revealed that when the task is redundant, the optimal feedback controller exhibits hierarchical structure even though it is not specifically designed to be hierarchical. Fig 2 illustrates this finding in the context of a simple redundant task –where two state variables x_1, x_2 with dynamics $x_i(t+1) = ax_i(t) + u_i(t)(1+\varepsilon_i(t))$, $i = 1, 2$, have to be controlled so that $x_1 + x_2$ equals a given target T. The noise model is control-multiplicative; details can be found in [11]. The optimal controller pushes the system state towards the nearest point on the manifold of acceptable states ("redundant direction" in Fig 2). This clearly requires less control energy compared to pushing towards a predefined point (say $x_1 = x_2 = T/2$).

It also causes less variability in the direction that matters (compare state covariance to gray circle). Note however that variability in the redundant direction is increased –because the optimal controller is not correcting deviations in that direction. Analysis of optimal value functions for stochastic nonlinear control problems has shown that the structure in Fig 2 is a general property of optimal feedback controllers: deviations from the average behavior are corrected selectively, only when they compromise task performance [11]. We have called this the *minimal intervention* principle.

This is an important result because redundancy is ubiquitous in the motor system. The phenomenon of increased variability along redundant directions (illustrated below) has been observed in a wide range of behaviors [11], and has been quantified via the "uncontrolled manifold" method for comparing task-relevant and task-irrelevant variance [27]. It implies that the substantial yet structured variability of biological movements is not due to sloppiness, but on the contrary, is a signature of an exceptionally well-designed sensorimotor system.

The hierarchical nature of the optimal controller is evident in Fig 2: u_1, u_2 are coupled, and equal to some function of $x_1 + x_2$ rather than being functions of the individual state variables x_1, x_2. In the terminology of Fig 1, the high-level state is $y = x_1 + x_2$, the high-level control is $v = f(y)$, and the vector of actual control signals is $\mathbf{u} = [v; v]$. In more complex tasks our analysis has shown that the optimal control law still has this property, but for a high-level state $\mathbf{y}$ which may not be easy to guess. This is because $\mathbf{y}$ is related to the shape of the optimal value function (defined as the sum of future costs under an optimal control law) rather than the cost function, and the optimal value function is hard to compute. Nevertheless, these results [11,12] provide strong motivation for hierarchical approximations to optimal control in redundant tasks.

2.2 Empirical illustration of structured motor variability

Selective control of task-relevant parameters –which is a signature of both optimal feedback control and hierarchical control schemes –implies increased variability in task-irrelevant parameters. This phenomenon is illustrated here with data from the following experiment. Six subjects were asked to move repeatedly between three targets attached to their body (one target on each leg, and one on the left upper arm). Subjects held a 25cm wooden pointer in their right hand, and always used the same tip (TIP 1) to touch the center of the targets. We measured the position and orientation of the center of the pointer with a Polhemus Liberty sensor (240Hz

sampling rate), which allowed us to calculate the positions of both TIP 1 and TIP 2. The experimental setup is shown in Fig 3a. The goal of the experiment was to quantify the positional variance of the movement paths.

The data were analyzed as follows. The start and end times of each movement were found using a velocity threshold, applied when TIP 1 was within 10cm of a target. Outlier movements, that failed to reach within 10cm of their target, were eliminated. Each movement was resampled in space, at equal intervals with length 2% of the total path length for that movement (this resampling is needed to suppress temporal misalignment). The mean and covariance for each sample point were computed, separately for each subject and movement target. Positional variance was defined as the trace of the covariance matrix. The means and covariances were then averaged over subjects, and plotted in F**ig 3b** (±1 standard deviation ellipsoids are shown at 25% intervals). To facilitate comparison, we further averaged over the three targets and plotted the positional variances of the two tips of the pointer in Fig 3c.

The main result shown in Fig 3c nicely illustrates the typical structure of motor variability. The position of the task-relevant TIP 1 is less variable than the task-irrelevant TIP 2. Furthermore, both variances are smaller in the task-relevant portions of the movement (when the pointer approaches the target) compared to the middle of the movement. The latter effect is much stronger for the task-relevant TIP 1.

These observations make it clear that the nervous system is not using the classic robotics approach – which is to plan a trajectory in joint space, and then execute that trajectory using some combination of computed torque and servo control. This is not surprising. Since a joint-space trajectory plan contains no information regarding task relevance, the execution system would have no choice but to track faithfully all details of the plan –including the irrelevant ones. This would result in increased energy consumption, and increased errors due to control-dependent noise.

3 Hierarchical control framework

Key to our framework is the low-level controller $\mathbf{u}(\mathbf{v}, \mathbf{x})$ –whose design we address first, assuming that the high-level parameters $\mathbf{y}(\mathbf{x})$ and their desired dynamics have been given. We pay specific attention to the case when the controls $\mathbf{u}$ do not affect the high-level parameters $\mathbf{y}$ instantaneously. We then discuss how the desired $\mathbf{y}$-dynamics can be adapted to the natural plant dynamics, and how the high-level controller $\mathbf{v}(\mathbf{y})$ can be designed using optimal control techniques.

3.1 Low-level controller design

Consider the dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t)) + B(\mathbf{x}(t))\mathbf{u}(t) \tag{1}$$

where $\mathbf{x} \in \mathrm{R}^{n_\mathbf{x}}$ is the state vector, $\mathbf{u} \in \mathrm{R}^{n_\mathbf{u}}$ is the control vector, $\mathbf{a}(\mathbf{x})$ are the passive dynamics, and $B(\mathbf{x})\,\mathbf{u}$ are the control-dependent dynamics (assumed linear in the control). We are interested in controllers that achieve low cumulative cost, for a cost-per-step function of the form

$$\ell(t, \mathbf{x}(t), \mathbf{u}(t)) = q(t, \mathbf{x}(t)) + r(\mathbf{u}(t), \mathbf{x}(t)) \tag{2}$$

The high-level state vector $\mathbf{y} \in \mathrm{R}^{n_y}$, $n_\mathbf{y} \leq n_\mathbf{x}$, is a static function of the plant state:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{3}$$

The function **h** defining the high-level state **y** is supplied by the user. What we provide is a way to "control" **y**, that is, to generate controls **u** which affect **x** in such a way that the corresponding changes in **y** are as desired. We assume that **y** contains enough information to compute the state-dependent cost $q(t, \mathbf{x})$; in other words, there exists a function $\tilde{q}$ such that

$$\tilde{q}(t, \mathbf{h}(\mathbf{x})) = q(t, \mathbf{x}) \tag{4}$$

Other than that, the function **h** should be chosen to yield the lowest-dimensional representation which allows satisfactory control. Different choices of **h** will be explored later. The control cost $r(\mathbf{u}, \mathbf{x})$ cannot be represented exactly on the high level, because it typically contains independent contributions from all components of **u**. This is the reason why the proposed hierarchical scheme is only an approximation to optimal control.

Differentiating (3) w.r.t. t and using (1), the dynamics of **y** become

$$\dot{\mathbf{y}} = H(\mathbf{x})(\mathbf{a}(\mathbf{x}) + B(\mathbf{x})\mathbf{u}) \tag{5}$$

where $H(\mathbf{x}) = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian of the function **h**. Note that **y** will often contain more information than end-effector position.

Our design method seeks to create an augmented system with prescribed high-level dynamics

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}) + G(\mathbf{y})\mathbf{v} \tag{6}$$

where $\mathbf{v} \in \mathrm{R}^{n_{\mathbf{v}}}$ is the high-level control signal generated by some high-level controller. The functions **f**, G define the meaning of the high-level control **v**, as discussed later. For now we simply assume that **f**, G are some known functions.

The low-level controller design problem is the following: choose $\mathbf{u}(\mathbf{v}, \mathbf{x})$ so that the prescribed (6) and actual (5) **y**-dynamics are identical; when multiple solutions exist, use the control cost $r(\mathbf{u}; \mathbf{x})$ to resolve redundancy. The control **u** is thus defined at each time t as the solution to the following constrained optimization problem:

$$\begin{aligned} &\text{given } \mathbf{v} \text{ and } \mathbf{x}, \text{ find } \mathbf{u} \text{ that minimizes } r(\mathbf{u}, \mathbf{x}) \text{ subject to} \\ &H(\mathbf{x})B(\mathbf{x})\mathbf{u} = \mathbf{f}(\mathbf{y}) + G(\mathbf{y})\mathbf{v} - H(\mathbf{x})\mathbf{a}(\mathbf{x}) \end{aligned} \tag{7}$$

In addition to the above equality constraint, we can incorporate inequality constraints on **u**. The latter are particularly important in biological movement control where muscle activations are always non-negative. For a general $r(\mathbf{u}, \mathbf{x})$, problem (7) has to be solved numerically at each time step (using Sequential Quadratic Programming, or another efficient method). Since the solution is likely to be similar from one time step to the next, we can use the current solution to initialize the search for the next solution.

In practice, the control cost will normally have the quadratic form

$$r(\mathbf{u}, \mathbf{x}) = \frac{1}{2}\mathbf{u}^{\mathrm{T}} R(\mathbf{x})\mathbf{u} \tag{8}$$

where $R(\mathbf{x})$ is a symmetric positive-definite matrix (often proportional to the identify matrix). When $r(\mathbf{u}, \mathbf{x})$ is in the form (8), and control inequality constraints are absent, we can solve (7) explicitly. The solution relies on the following fact:

$$\text{the minimum of } \frac{1}{2}\mathbf{u}^{\mathrm{T}} W \mathbf{u} \text{ subject to } X\mathbf{u} = \mathbf{c} \text{ is } \mathbf{u} = X^{\dagger}_{W}\mathbf{c} \tag{9}$$

The weighted pseudo-inverse is defined as

$$X^{\dagger}_{W} = W^{-1} X^{\mathrm{T}} (X W^{-1} X^{\mathrm{T}})^{-1} \tag{10}$$

where W is symmetric positive-definite, and X has full row-rank. Note that $X^{\dagger}_{\mathrm{I}}$ is the Moore-Penrose pseudo-inverse $X^{\mathrm{T}} (XX^{\mathrm{T}})^{-1}$; in that case we use the shorthand $X^{\dagger}$.

Applying this to problem (7) for a cost in the form (8), and assuming that $H(\mathbf{x})B(\mathbf{x})$ has full row-rank (a condition which we will return to), the unique solution is

$$\mathbf{u}(\mathbf{v}, \mathbf{x}) = (H(\mathbf{x})B(\mathbf{x}))^{\dagger}_{R(\mathbf{x})}(\mathbf{f}(\mathbf{y}) + G(\mathbf{y})\mathbf{v} - H(\mathbf{x})\mathbf{a}(\mathbf{x})) \tag{11}$$

In case of inequality constraints on $\mathbf{u}$ we cannot obtain an explicit solution, and have to resort to numerical optimization. But the quadratic form (8) of the control cost allows us to use Quadratic Programming – which is numerically very efficient when the cost is convex.

Another form of cost function, that can be used on the low level to resolve redundancy, is

$$r(\mathbf{u}, \mathbf{x}) = \frac{1}{2}\mathbf{u}^{\mathrm{T}} R(\mathbf{x})\mathbf{u} + (\mathbf{a}(\mathbf{x}) + B(\mathbf{x})\mathbf{u})^{\mathrm{T}} \nabla_{\mathbf{x}} g(\mathbf{x}) \tag{12}$$

The new term encourages movement against the gradient of some potential function $g(\mathbf{x})$. With this $r(\mathbf{u}, \mathbf{x})$ problem (7) can still be solved explicitly, using the following fact:

$$\begin{aligned} &\text{the minimum of } \frac{1}{2}\mathbf{u}^{\mathrm{T}} W\mathbf{u} + \mathbf{u}^{\mathrm{T}}\mathbf{d} \text{ subject to } X\mathbf{u} = \mathbf{c} \text{ is} \\ &\mathbf{u} = \begin{bmatrix} X \\ N_X^{\mathrm{T}} W \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{c} \\ -N_X^{\mathrm{T}}\mathbf{d} \end{bmatrix} = X^{\dagger}_{W}\mathbf{c} - N_X (N_X^{\mathrm{T}} W N_X)^{-1} N_X^{\mathrm{T}}\mathbf{d} \end{aligned} \tag{13}$$

where N_X is a matrix whose columns form an orthonormal basis for the null-space of X. Note that solving the system $[X; N^{\mathrm{T}}_X W]\,\mathbf{u} = [\mathbf{c}; -N^{\mathrm{T}}_X\mathbf{d}]$ with a linear solver is numerically preferable to explicit computation of the pseudo-inverses in (13). A similar point has been made in [35] regarding velocity control methods.

Applying (13) to our problem in the case when control inequality constraints are absent, and suppressing the functional dependencies for clarity, we have

$$\mathbf{u} = (HB)^{\dagger}_{R}(\mathbf{f} + G\mathbf{v} - H\mathbf{a}) - N_{HB}(N_{HB}^{\mathrm{T}} R N_{HB})^{-1} N_{HB}^{\mathrm{T}} B^{\mathrm{T}} \nabla_{\mathbf{x}} g \tag{14}$$

In comparison to (11), the controller in (14) has an extra term acting in the redundant subspace where variations in $\mathbf{x}$ do not affect $\mathbf{y}$. Thus $g(\mathbf{x})$ can be used to modify the behavior of the controlled system in that subspace; the choice of $g(\mathbf{x})$ is presently left to the user.

An alternative way of computing $\mathbf{u}(\mathbf{v}, \mathbf{x})$ is to transform the above constrained optimization problem into an unconstrained one, and use a mixed cost that absorbs all constraints. Let $c(\mathbf{u}, \mathbf{x})$ be a potential function replacing any control inequality constraints: $c(\mathbf{u}, \mathbf{x}) = 0$ when $\mathbf{u}$ satisfies those constraints and $c(\mathbf{u}, \mathbf{x}) > 0$ when it does not. Then we can define $\mathbf{u}$ at each time t as the solution to the following unconstrained optimization problem:

$$\begin{aligned} &\text{given } \mathbf{v} \text{ and } \mathbf{x}, \text{ find } \mathbf{u} \text{ that minimizes} \\ &\lambda_1 \left\| \mathbf{f}(\mathbf{y}) + G(\mathbf{y})\mathbf{v} - H(\mathbf{x})\mathbf{a}(\mathbf{x}) - H(\mathbf{x})B(\mathbf{x})\mathbf{u} \right\|^2 + \lambda_2 c(\mathbf{u}, \mathbf{x}) + r(\mathbf{u}, \mathbf{x}) \end{aligned} \tag{15}$$

The constants λ_1, λ_2 set the relative importance of satisfying constraints versus minimizing energy. For large λ's the solution to this problem is numerically indistinguishable from the solution to the constrained problem, in the case when all constraints can be satisfied. However, the advantage of (15) is that it applies even when the constraints on $\mathbf{u}$ cannot be satisfied – which can happen when $H(\mathbf{x})B(\mathbf{x})$ becomes row-rank-defficient, or when the high-level control $\mathbf{v}$ calls for a low-level control $\mathbf{u}$ exceeding its allowed range.

3.2 Dynamic compatibility between levels of control

A potential problem with the above method is lack of "dynamic compatibility" between the two levels. This occurs when $\dot{\mathbf{y}}$ does not explicitly depend on $\mathbf{u}$, and leads to $H(\mathbf{x})B(\mathbf{x}) = 0$. For example, suppose $\mathbf{y}$ is end-effector position, $\mathbf{v}$ is end-effector velocity, and $\mathbf{u}$ is joint torque. This is problematic because torque cannot affect velocity instantaneously. However torque has a predictable effect when applied over time, suggesting that the "instantaneous" $\mathbf{a}$, B should be replaced with functions $\mathbf{a}$, B incorporating temporal prediction. Next we present a method for doing so.

Consider the discrete-time representation of system (1) with time step Δ:

$$\mathbf{x}(t+\Delta) = \mathbf{x}(t) + \Delta(\mathbf{a}(\mathbf{x}(t)) + B(\mathbf{x}(t))\mathbf{u}(t)) \tag{16}$$

We now analyze this system within a single step, from time t to time $t+\Delta$. Define the shortcut notation $\mathbf{x}_\tau = \mathbf{x}(t+\tau)$, $\tau \in [0, \Delta]$, so that $\mathbf{x}_0 = \mathbf{x}(t)$. The control term $\mathbf{b} = B(\mathbf{x}(t))\mathbf{u}(t)$ will be treated as constant, while the changes in $\mathbf{a}(\mathbf{x})$ will be captured to first order through the linearization

$$\mathbf{a}(\mathbf{x}_\tau) \approx \mathbf{a}(\mathbf{x}_0) + A(\mathbf{x}_\tau - \mathbf{x}_0), \quad A = \frac{\partial \mathbf{a}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_0} \tag{17}$$

Then we have continuous dynamics

$$\dot{\mathbf{x}}_\tau = A\mathbf{x}_\tau + (\mathbf{a}(\mathbf{x}_0) + \mathbf{b} - A\mathbf{x}_0) \tag{18}$$

where the term in the brackets is constant. In the case when x, a, b, $A = a'(x_0)$ are scalar we can solve this initial-value ODE analytically, and after some rearrangement obtain

$$x_\Delta = x_0 + \frac{\exp(\Delta a'(x_0)) - 1}{a'(x_0)}(a(x_0) + b) \tag{19}$$

This scalar result indicates that in the vector case we can expect a similar solution involving a matrix exponential and an inverse, and we can also expect complications when A is singular.

We will now solve (18) in the vector case, by performing Euler integration in n smaller time steps of duration $d = \Delta/n$ each, and then taking the limit $n \to \infty$. Each step $k < n$ is in the form

$$\mathbf{x}_{d(k+1)} = (\mathrm{I} + dA)\mathbf{x}_{dk} + d(\mathbf{a}(\mathbf{x}_0) - A\mathbf{x}_0 + \mathbf{b}) \tag{20}$$

Applying this recursion n times, and noting that $\mathbf{x}_{dn} = \mathbf{x}_\Delta$, we have

$$\mathbf{x}_\Delta = (\mathrm{I} + dA)^n \mathbf{x}_0 + (\mathrm{I} + (\mathrm{I} + dA) + \cdots + (\mathrm{I} + dA)^{n-1})\, d\,(\mathbf{a}(\mathbf{x}_0) - A\mathbf{x}_0 + \mathbf{b}) \tag{21}$$

Using the identity $(\mathrm{I} + X + \cdots + X^{n-1})(X - \mathrm{I}) = (X^n - \mathrm{I})$ along with $dA\mathbf{x}_0 = ((\mathrm{I} + dA) - \mathrm{I})\,\mathbf{x}_0$, the above equality simplifies to

$$\mathbf{x}_\Delta = \mathbf{x}_0 + \Delta T(\Delta A,\ n)(\mathbf{a}(\mathbf{x}_0) + \mathbf{b}) \tag{22}$$

where the matrix function T is defined as

$$T(X,\ n) = \frac{1}{n}\left(\mathrm{I} + \left(\mathrm{I} + \frac{X}{n}\right) + \cdots + \left(\mathrm{I} + \frac{X}{n}\right)^{n-1}\right) \tag{23}$$

When X is invertible, T further simplifies to

$$T(X,\ n) = \left(\left(\mathrm{I} + \frac{X}{n}\right)^n - \mathrm{I}\right)X^{-1} \tag{24}$$

In the limit $n \to \infty$, the identity $\lim_{n\to\infty} (\mathrm{I} + X/n)^n = \exp(X)$ can be used to obtain

$$T(X,\ \infty) = (\exp(X) - \mathrm{I})X^{-1} \tag{25}$$

When X is singular we can approximate $T(X, \infty)$ by either adding $\epsilon \mathrm{I}$ to X and applying (25), or by using $T(X, n)$ evaluated with a large n. Numerically we have found that $T(X, n)$ converges even for singular X, although we do not have a closed-form solution in that case.

Switching back to continuous notation, we have the following modified integration scheme

$$\mathbf{x}(t + \Delta) = \mathbf{x}(t) + \Delta T(\Delta A,\ \infty)(\mathbf{a}(\mathbf{x}(t)) + B(\mathbf{x}(t))\mathbf{u}(t)) \tag{26}$$

which differs from Euler integration (16) only by T. This motivates the construction of the "predictive" dynamics $\mathbf{a}$, B from the following condition: Euler integration (16) with $\mathbf{a}$, B should be equivalent to the modified scheme (26) with the original $\mathbf{a}$, B. This condition is satisfied by

$$\begin{aligned} \mathbf{a}(\mathbf{x}) &= T(\Delta A,\ \infty)\mathbf{a}(\mathbf{x}) \\ B(\mathbf{x}) &= T(\Delta A,\ \infty)B(\mathbf{x}) \end{aligned} \tag{27}$$

Explicit integration of the functions $\mathbf{a}$, B resembles implicit integration of the original $\mathbf{a}$, B. To see the relationship to implicit integration, consider the implicit Euler scheme

$$\mathbf{x}_\Delta = \mathbf{x}_0 + \Delta(\mathbf{a}(\mathbf{x}_\Delta) + \mathbf{b}) \tag{28}$$

where $\mathbf{x}_\Delta$ is now the unknown in a non-linear equation. One way to approximate the solution is to linearize $\mathbf{a}$ around $\mathbf{x}_0$ as we did before, yielding

$$\mathbf{x}_\Delta = \mathbf{x}_0 + \Delta(\mathbf{a}(\mathbf{x}_0) + A\mathbf{x}_\Delta - A\mathbf{x}_0 + \mathbf{b}) \tag{29}$$

Moving $\Delta A\mathbf{x}_\Delta$ to the left hand side and left-multiplying by $(\mathrm{I} - \Delta A)^{-1}$, the solution is

$$\mathbf{x}_\Delta = \mathbf{x}_0 + \Delta\overline{T}(\Delta A)(\mathbf{a}(\mathbf{x}_0) + \mathbf{b}) \tag{30}$$

where the matrix function $\overline{T}$ is defined as

$$\overline{T}(X) = (\mathrm{I} - X)^{-1} \tag{31}$$

This is very similar to (22), except that $\overline{T}$ now replaces T. However (25) is preferable to (31), because it captures the exact solution of the linearized ODE (18) while the implicit Euler method relies on a finite-difference approximation. Note that implicit integration is in general

more stable and accurate, so the functions $\mathbf{a}$, B can be used instead of $\mathbf{a}$, B even when dynamic compatibility is not an issue. Their only drawback is the increased computational cost per step – which may be offset by the larger time steps that implicit integrators can safely take.

Here is a simple example of how (27) can enforce dynamic compatibility. Suppose p and v are the position and velocity of a 1D point with mass 1, and u is the applied force, so that $\dot{v} = u$. The plant state is $\mathbf{x} = [p; v]$. Let the high-level state be $y = p$. Then the dynamics functions are $\mathbf{a}(\mathbf{x}) = [v; 0]$ and $B = [0; 1]$. Therefore $H = \partial y / \partial \mathbf{x} = [1, 0]$, and so $HB = 0$. To apply (27) we first compute $A = \partial \mathbf{a} / \partial \mathbf{x} = [0, 1; 0, 0]$. Numerically $T(\Delta A; \infty)$ converges to the matrix $[1, \Delta/2; 0, 1]$. Thus the modified dynamics are $\mathbf{a}(\mathbf{x}) = \mathbf{a}(\mathbf{x})$ and $B = [\Delta / 2; 1]$, yielding $HB = \Delta / 2$. The fact that $HB \neq 0$ makes it possible to design the low-level controller using (11).

3.3 High-level dynamics and cost models

It is clear from (11) that as long as $H(\mathbf{x})B(\mathbf{x})$ has full row-rank, we can choose $\mathbf{f}$, G arbitrarily and thereby instantiate whatever high-level dynamics we desire. But what should we desire? One possibility is to choose $\mathbf{f}(\mathbf{y})$ linear and $G(\mathbf{y})$ constant – which yields linear $\mathbf{y}$-dynamics, and is related to feedback linearization as discussed below. However we believe that the $\mathbf{y}$-dynamics should mimic the $\mathbf{x}$-dynamics to the extent possible, so that the high-level controller can exploit the natural plant dynamics while operating on a lower-dimensional system. Matching the passive dynamics in (5) and (6) yields

$$\mathbf{f}(\mathbf{h}(\mathbf{x})) \approx H(\mathbf{x})\mathbf{a}(\mathbf{x}) \tag{32}$$

Matching the control-dependent terms in (5) and (6) yields

$$G(\mathbf{h}(\mathbf{x}))\mathbf{v}(\mathbf{u}, \mathbf{x}) \approx H(\mathbf{x})B(\mathbf{x})\mathbf{u} \tag{33}$$

where the function $\mathbf{v}(\mathbf{u}, \mathbf{x})$ remains to be defined. In addition, optimization of the high-level controller will be facilitated if the cost function can be captured on the high level. The state-dependent cost term $q(t, \mathrm{x})$ is by definition computable given $\mathbf{y} = \mathbf{h}(\mathbf{x})$, but we would also like (an approximation to) the control energy term $r(\mathbf{u}, \mathbf{x})$. This requires a function r such that

$$r(\mathbf{v}(\mathbf{u}, \mathbf{x}), \mathbf{h}(\mathbf{x})) \approx r(\mathbf{u}, \mathbf{x}) \tag{34}$$

Functions $\mathbf{f}$, G, r that approximately satisfy (32,33,34) can be constructed by choosing a suitable parameterization (basis functions, neural networks, etc.), collecting data, and fitting a model using supervised learning procedures. The dataset consists of observation pairs $(\mathbf{x}^{(k)}, \mathbf{u}^{(k)})$, where $\mathbf{u}^{(k)}$ is the signal generated by some existing controller when the plant is in state $\mathbf{x}^{(k)}$. Given $\mathbf{x}^{(k)}$ and $\mathbf{u}^{(k)}$, we can compute learning targets for $\mathbf{f}$, G, r by evaluating the right hand sides of (32,33,34). Thus the application of supervised learning is straightforward –as long as we define the function $\mathbf{v}(\mathbf{u}, \mathbf{x})$. This function corresponds to the meaning of the high-level control signals, and should be chosen using physical intuition. For example, if $\mathbf{u}$ is joint torque and $\mathbf{y}$ is end-effector position and velocity, $\mathbf{v}(\mathbf{u}, \mathbf{x})$ could be either end-effector acceleration or force. In the former case G will be constant and will not capture anything about plant dynamics. In the latter case G will be related to inverse inertia in end-effector space. Regardless of how we choose the function $\mathbf{v}$ after learning we will have $\mathbf{v}(\mathbf{u}(\mathbf{v}, \mathbf{x}), \mathbf{x}) \approx \mathbf{v}$. Note that whenever the functions $\mathbf{f}$, G change (through learning or otherwise) the low-level controller must be redesigned.

It is interesting to ask under what conditions (32,33,34) can be satisfied exactly. We focus on the passive dynamics (32), which do not depend on our choice of $\mathbf{v}(\mathbf{u}, \mathbf{x})$ and are also likely to

be the most worthwhile fitting. Since $\mathbf{x} \xrightarrow{\mathbf{h}} \mathbf{y}$ is a many-to-one mapping, a function **f** satisfying (32) does not always exist. The necessary and sufficient condition for the existence of **f** is

$$H(\mathbf{x}_1)\mathbf{a}(\mathbf{x}_1) = H(\mathbf{x}_2)\mathbf{a}(\mathbf{x}_2) \text{ whenever } \mathbf{h}(\mathbf{x}_1) = \mathbf{h}(\mathbf{x}_2) \tag{35}$$

Define the set $\mathcal{M}[\mathbf{x}_0] = \{\mathbf{x} : \mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{x}_0)\}$ and suppose **h** is such that $\mathcal{M}$ is a smooth manifold. This is the redundant (or uncontrolled) manifold where the plant state **x** can fluctuate without affecting the high-level state **y**. If the passive dynamics act within this manifold, i.e. $\mathbf{a}(\mathbf{x}) \in T_x\mathcal{M}[\mathbf{x}]$, then $H(\mathbf{x})\mathbf{a}(\mathbf{x}) = 0$ and (35) is trivially satisfied. Therefore (35) can only be violated by the component of the vector field **a** normal to $\mathcal{M}$. In particular, note that the row vectors of the matrix $H(\mathbf{x})$ span the subspace normal to $\mathcal{M}[\mathbf{x}]$, and so $H(\mathbf{x})\mathbf{a}(\mathbf{x})$ is a form of projection of the vector field in the normal subspace. For nonlinear $\mathbf{h}(\mathbf{x})$ this projection is difficult to understand because $H(\mathbf{x})$ varies with **x**, but in the linear case $\mathbf{h}(\mathbf{x}) = H\mathbf{x}$ condition (35) becomes

$$H(\mathbf{a}(\mathbf{x}_1) - \mathbf{a}(\mathbf{x}_2)) = 0 \text{ whenever } H(\mathbf{x}_1 - \mathbf{x}_2) = 0 \tag{36}$$

The latter condition has a simple interpretation: if the difference between two states lies within the redundant subspace, then the difference in the passive dynamics between these states should also lie within the redundant subspace.

3.4 High-level controller optimization

A high-level dynamics model together with an (approximate) high-level cost model lead to an optimal control problem formulated on the high level. This problem can be solved via standard techniques, or via a new implicit method (see below) that is specific to hierarchical control schemes.

While a survey of standard techniques for optimal controller design is beyond the scope of this paper, three points are in order. First, mapping the original control problem to the high level reduces dimensionality, and thereby facilitates the application of numerical methods for optimal control that suffer from the curse of dimensionality. Second, a high-level dynamics model is required in order to design the low-level controller, even if the high-level controller is built via model-free methods (such as proportional-derivative control, or reinforcement learning). Third, it may be advantageous to use iterative design: start with some high-level model, design the two-level control scheme and apply it, use the resulting data to fit a high-level model more compatible with the plant dynamics, and iterate. The need to iterate comes from the fact that the approximation (32,33,34) depends on what regions of the state and control spaces are visited, and that in turn depends on what controller is being used during data collection.

In addition to standard techniques, the hierarchical framework allows a novel implicit method for optimal controller design. In this method, the passive dynamics are no longer modeled explicitly on the high level and the discrepancy between $\mathbf{f}(\mathbf{y})$ and $H(\mathbf{x})\mathbf{a}(\mathbf{x})$ is no longer compensated on the low level. Instead, the high-level controller is given online access to $H(\mathbf{x})\mathbf{a}(\mathbf{x})$ and is responsible for dealing with that term. The constraint in (7) simplifies to $H(\mathbf{x})B(\mathbf{x})\mathbf{u} = G(\mathbf{y})\mathbf{v}$, and the low-level controller (11) becomes

$$\mathbf{u}(\mathbf{v}, \mathbf{x}) = (H(\mathbf{x})B(\mathbf{x}))^{\dagger}_{R(\mathbf{x})}G(\mathbf{y})\mathbf{v} \tag{37}$$

When condition (35) is violated the input-output behavior of the augmented plant resulting from (37) is no longer autonomous in terms of **y**, but instead can depend on the specific value of **x** –because $H(\mathbf{x})\mathbf{a}(\mathbf{x})$ can vary while **y** remains constant. In that case $\mathbf{f}(\mathbf{y})$ can still be

computed as long as the plant state **x** is known. But how can we know **x** when $\mathbf{x} \xrightarrow{\mathbf{h}} \mathbf{y}$ is a many-to-one mapping? We cannot if all we are given is an arbitrarily chosen **y**. However, suppose that we reached **y** by initializing the plant in some state **x** (0), applying some high-level control sequence **v** (*t*), computing the low-level controls from (37), and integrating the plant dynamics. Then we always have a unique underlying state **x**. Implicit computation of **f** (**y**) can be used in conjunction with trajectory-based methods for optimal control –such as ODE methods [36], differential dynamic programming [37], or the iterative linear-quadratic-Gaussian method [38] we have developed (illustrated later). This modified version of our hierarchical scheme is likely to yield better performance, but comes at the price of not having an explicit high-level controller **v** (**y**) that is independent of the initial **x** and its trajectory. When condition (35) is satisfied, and an exact passive dynamics model **f** (**y**) = H (**x**) **a** (**x**) is available, the implicit method does not improve performance.

4 Relation to existing approaches

Hierarchical control is an appealing idea which has been pursued in a number of fields. Consequently, several existing approaches are related to ours. Some are related only in spirit (and are summarized first), while others allow formal comparisons developed in the sub-sections below.

In reinforcement learning, researchers have attempted to alleviate the curse of dimensionality via autonomous low-level control policies [8,9] that solve specific sub-problems over time (such as exiting a room in a maze). In contrast our low-level controller performs an instantaneous feedback transformation of plant dynamics, and is continuously driven by high-level commands. We believe this is more appropriate for control of complex redundant manipulators (and is a more plausible model of biological sensorimotor control) while hierarchical reinforcement learning is more appropriate for non-articulated “agents” solving navigation problems.

In control theory, there is increasing interest in hybrid systems [39–41] with continuous low-level dynamics and discrete event-driven high-level dynamics. In contrast, the high-level dynamics we instantiate are continuous. Consider for example a walking mechanism. Our framework can be applied to yield augmented dynamics in terms of the center-of-mass and feet positions. The resulting high level could be further decomposed via a hybrid systems approach, with ground contacts being the discrete events. Thus the two approaches are in some cases complementary, and a closer examination of the link between them is needed.

In robotics, examples of hierarchical control include the subsumption architecture [5] as well as virtual model control [42]. The latter is similar to our approach in terms of overall strategy –which is to design high-level feedback controllers without considering the full details of the plant. However virtual model control is a kinematic method: it maps forces produced by virtual actuators into real actuator commands, and does not exploit knowledge of plant dynamics. Another difference is that the high-level controller design is intuitive rather than model-based, and furthermore a model of high-level dynamics independent of the specific high-level controller is not available. Finally, virtual model control takes dynamic compatibility for granted and does not readily apply to biomechanical systems with higher-order dynamics.

4.1 Feedback linearization

There are interesting parallels between our method and feedback linearization (FL) –which is one of the few general methods for nonlinear control [13,14]. For comparison purposes, we first give a brief summary of input-output FL in the single-input-single-output case. Consider the system

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})u \\ y &= h(\mathbf{x}) \end{aligned} \tag{38}$$

where the control u is now called "input", the high-level state y is "output", the high-level control v is "external reference input", and u, y, v are for simplicity scalar. Define the Lie derivative of a function $h(\mathbf{x})$ with respect to a vector field $\mathbf{a}(\mathbf{x})$ as the directional derivative of h along $\mathbf{a}$:

$$L_{\mathbf{a}} h = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \mathbf{a}(\mathbf{x}) \tag{39}$$

System (38) is said to have relative degree r in a region D if for all $\mathbf{x} \in D$ we have

$$\begin{aligned} L_{\mathbf{b}} L_{\mathbf{a}}^{k} h(\mathbf{x}) &= 0 \quad \text{for all } k < r-1 \\ L_{\mathbf{b}} L_{\mathbf{a}}^{r-1} h(\mathbf{x}) &\neq 0 \end{aligned} \tag{40}$$

Then the low-level control law (called "state feedback control") given by

$$u(v, \mathbf{x}) = \frac{1}{L_{\mathbf{b}} L_{\mathbf{a}}^{r-1} h(\mathbf{x})} (v - L_{\mathbf{a}}^{r} h(\mathbf{x})) \tag{41}$$

yields linear y-dynamics of the form $y^{(r)} = v$, where $y^{(r)}$ is the r-th order time derivative of y.

To see the similarity to our method, note that definition (39) implies $L_a h(x) = H(\mathbf{x})\mathbf{a}(\mathbf{x})$ and similarly for L_b. Now, in the case $r = 1$ we have $L_a^{r-1} h = h$, and the control law (41) becomes

$$u(v, \mathbf{x}) = \frac{1}{H(\mathbf{x})\mathbf{b}(\mathbf{x})} (v - H(\mathbf{x})\mathbf{a}(\mathbf{x})) \tag{42}$$

which is identical to our control law (11) for prescribed high-level dynamics $f(y) = 0$, $G(y) = 1$ and control cost $R(\mathbf{x}) = 1$. The parallels between the two methods can be summarized as follows:

- The objective of FL is to make the $\mathbf{y}$-dynamics linear, in the form $\mathbf{y}^{(r)} = \mathbf{v}$, while our objective is to make it low-dimensional yet similar to the $\mathbf{x}$-dynamics. Our method can instantiate any desired $\mathbf{y}$-dynamics.
- The dynamic compatibility problem discussed above corresponds to a system with relative degree greater than one. FL handles this case by augmenting $\mathbf{y}$ with its first $r-1$ derivatives. We could do the same, but that may defeat the purpose of our method because dimensionality is increased. Instead, we prefer to use the predictive version $\mathbf{a}, B$ of plant dynamics.
- Most of FL theory is developed for systems with equal numbers of inputs and outputs, while redundant manipulators tend to have many more inputs than task-relevant outputs. Our method does not impose any such restriction. We deal with the mismatch in dimensionality by formulating an optimization problem –which yields a weighted pseudo-inverse solution.
- There is an important step in FL which was omitted in the above summary. It has to do with finding an explicit parameterization of the redundant manifold $\mathcal{M}$ defined above. Such a parameterization is analytically interesting, but does not facilitate the design of a high-level controller (and the low-level controller has already been designed).

4.2 Operational space formulation

Our framework also turns out to have a number of interesting similarities and differences with the operational space (OS) formulation [6] –which is another hierarchical scheme aiming to decouple task-level control from details of plant dynamics. For comparison purposes, we summarize the OS scheme here. Let $\mathbf{q}$ be the generalized coordinates of a redundant manipulator. The end-effector coordinates $\mathbf{p}$ are related to $\mathbf{q}$ by the forward kinematics function $\mathbf{p} = \mathbf{k}(\mathbf{q})$, with Jacobian $J(\mathbf{q}) = \frac{\partial \mathbf{k}(\mathbf{q})}{\partial \mathbf{q}}$. The equations of motion are

$$\tau = M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) \tag{43}$$

where $M(\mathbf{q})$ is the (symmetric positive-definite) generalized inertia matrix, and $\mathbf{n}$ are Coriolis, centripetal, gravitational, and viscoelastic forces. The end-effector inertia matrix is

$$\Lambda(\mathbf{q}) = \left(J(\mathbf{q})M(\mathbf{q})^{-1}J(\mathbf{q})^{\mathrm{T}}\right)^{-1} \tag{44}$$

The OS low-level controller maps desired end-effector accelerations $\ddot{\mathbf{p}}^*$ to generalized forces τ as

$$\tau = J(\mathbf{q})^{\mathrm{T}}\Lambda(\mathbf{q})(\ddot{\mathbf{p}}^* - \dot{J}(\mathbf{q})\dot{\mathbf{q}}) + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) \tag{45}$$

The OS scheme leaves the high-level controller unspecified – as long as that controller can issue desired end-effector accelerations. The high-level dynamics correspond to a point mass ($\ddot{\mathbf{p}} = \ddot{\mathbf{p}}^*$), therefore the operational space formulation is an instance of feedback linearization.

Let us now apply our method to the same plant. Note that our method is formulated using general first-order dynamics, and so the plant state contains both generalized coordinates and their derivatives. We have $\mathbf{x} = [\mathbf{q}; \dot{\mathbf{q}}]$, $y = [\mathbf{p}; \dot{\mathbf{p}}]$, and therefore

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \mathbf{k}(\mathbf{q}) \\ J(\mathbf{q})\dot{\mathbf{q}} \end{bmatrix} \tag{46}$$

The second row in the above equation comes from the identity

$$\dot{\mathbf{p}} = J(\mathbf{q})\dot{\mathbf{q}} \tag{47}$$

The Jacobian of $\mathbf{h}$ differs from the Jacobian of the forward kinematics function, and is given by

$$H(\mathbf{x}) = \begin{bmatrix} J(\mathbf{q}) & 0 \\ \frac{\partial}{\partial \mathbf{q}}(J(\mathbf{q})\dot{\mathbf{q}}) & J(\mathbf{q}) \end{bmatrix} \tag{48}$$

where the above partial derivative term can be written as $\dot{J}(\mathbf{q})$. This is because

$$\frac{\partial}{\partial \mathbf{q}}(J(\mathbf{q})\dot{\mathbf{q}}) = \frac{\partial}{\partial \mathbf{q}}\left(\frac{\partial \mathbf{k}}{\partial \mathbf{q}}\frac{d\mathbf{q}}{dt}\right) = \frac{\partial}{\partial \mathbf{q}}\left(\frac{d\mathbf{k}}{dt}\right) = \frac{d}{dt}\left(\frac{\partial \mathbf{k}}{\partial \mathbf{q}}\right) = \frac{d}{dt}(J(\mathbf{q})) = \dot{J}(\mathbf{q}) \tag{49}$$

From (43), the passive and control-dependent plant dynamics expressed in first-order form are

$$\mathbf{a}(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{q}} \\ -M(\mathbf{q})^{-1}\mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix}, \quad B(\mathbf{x}) = \begin{bmatrix} 0 \\ M(\mathbf{q})^{-1} \end{bmatrix} \tag{50}$$

Suppose the prescribed high-level dynamics correspond to a point mass ($\ddot{\mathbf{p}} = \mathbf{v}$) so that

$$\mathbf{f}(\mathbf{y}) = \begin{bmatrix} \mathbf{p} \\ 0 \end{bmatrix}, \quad G(\mathbf{y}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{51}$$

Let $R(x) = I$, so the control cost is $\frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{u}$. Then the constraint in (7) becomes

$$\begin{bmatrix} 0 \\ J(\mathbf{q})M(\mathbf{q})^{-1} \end{bmatrix}\mathbf{u} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \end{bmatrix} - \begin{bmatrix} J(\mathbf{q})\mathbf{q} \\ \dot{J}(\mathbf{q})\mathbf{q} - J(\mathbf{q})^{-1}\mathbf{n}(\mathbf{q}, \mathbf{q}) \end{bmatrix} \tag{52}$$

The first row is always satisfied because of (47). Thus the effective constraint on $\mathbf{u}$ is given by the second row of the above equation. Replacing the general $\mathbf{u}$, $\mathbf{v}$ with their specific to this example values τ, $\ddot{\mathbf{p}}^*$, the solution produced by our method is

$$\tau = \left(J(\mathbf{q})M(\mathbf{q})^{-1}\right)^{\dagger}\left(\ddot{\mathbf{p}}^* - \dot{J}(\mathbf{q})\mathbf{q} + J(\mathbf{q})M(\mathbf{q})^{-1}\mathbf{n}(\mathbf{q}, \mathbf{q})\right) \tag{53}$$

To make the relationship to the operational space formulation more explicit, we omit the dependence for clarity, and observe that the definition of the weighted pseudo-inverse implies $(JM^{-1})^{\dagger} = MJ_M{}^{\dagger}J^{\mathrm{T}}\Lambda = MJ_M^{\dagger}$. The two methods can then be rewritten as

$$\begin{aligned} &\text{operational space}: \quad \tau = MJ_M^{\dagger}(\ddot{\mathbf{p}}^* - \dot{J}\mathbf{q}) + \mathbf{n} \\ &\text{hierarchical control}: \quad \tau = MJ_M{}^{\dagger}{}_2(\ddot{\mathbf{p}}^* - \dot{J}\mathbf{q}) + \left(JM^{-1}\right)^{\dagger}\left(JM^{-1}\right)\mathbf{n} \end{aligned} \tag{54}$$

The overall form is very similar, but there are two differences. First, the torques $\mathbf{n}(\mathbf{q}, \mathbf{q})$ are fully cancelled in the OS method, while our method only cancels the component acting in end-effector space. Second, the Jacobian pseudo-inverses being used are weighted differently (see next section). Note that when J is invertible, both formulas reduce to the inverse dynamics (43), and map desired accelerations to forces. To see this, differentiate (47) to obtain $J(\mathbf{q})\ddot{\mathbf{q}} = \ddot{\mathbf{p}} - \dot{J}(\mathbf{q})\mathbf{q}$.

Despite the similarity, our framework is more general than the operational space formulation –in two ways that are obscured by this example. First, it can handle systems other than second-order, without any modification. Indeed we will later illustrate the control of a third-order system (a model of the human arm). Second, our framework is not constrained to point-mass high-level dynamics in the form (51), but can be used with other high-level models that are better adapted to the natural plant dynamics. Below we will see an example where adapting the high-level dynamics makes a difference in terms of performance.

4.3 Kinematic redundancy elimination

A number of kinematic planning methods that map desired end-effector velocities into joint velocities have been studied. Such methods start from the identity (47) and look for ways to invert $J(\mathbf{q})$. The majority of available methods lead to solutions of the form

$$\dot{\mathbf{q}} = J(\mathbf{q})_{W(\mathbf{q})}^{\dagger}\dot{\mathbf{p}}^* \tag{55}$$

and can be interpreted as minimizing $\frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}}W(\mathbf{q})\dot{\mathbf{q}}$ subject to the constraint (47). We will explore numerically three such pseudo-inverses:

$$\begin{aligned}&\text{Moore–Penrose: } W(\mathbf{q}) = \mathrm{I}\\ &\text{dynamically consistent: } W(\mathbf{q}) = M(\mathbf{q})\\ &\text{impedance control: } W(\mathbf{q}) = \mathbf{I} - \frac{\partial}{\partial \mathbf{q}}\left(J(\mathbf{q})^{\mathrm{T}}\hat{\mathbf{f}}\right),\ \hat{\mathbf{f}} = \left(J(\mathbf{q})^{\mathrm{T}}\right)^{\dagger}(\mathbf{q}_0 - \mathbf{q})\end{aligned} \tag{56}$$

The Moore-Penrose pseudo-inverse minimizes joint velocity, while the "dynamically consistent" one minimizes kinetic energy [43]. The pseudo-inverse labeled "impedance control" minimizes a quantity that does not have a name, but has an integrability property which yields repeatable joint motion [44]. In our implementation of that method $\hat{\mathbf{f}}$ is the end-effector force resulting from joint stiffness I and joint displacement $\mathbf{q}_0 - \mathbf{q}$, where $\mathbf{q}_0$ is the configuration illustrated in Fig 4a.

Note that the dynamically-consistent pseudo-inverse $J_M^{\dagger}$ also appears in the operational space formulation (despite the fact that the OS controller is driven by desired accelerations rather than desired velocities). Instead of $J_M^{\dagger}$, our method as shown in (54) involves the pseudo-inverse $J_{M^2}^{\dagger}$. If one were to apply (55) using our pseudo-inverse, that would correspond to minimizing $\frac{1}{2}\mathbf{q}^{\mathrm{T}}M(\mathbf{q})^2\mathbf{q} = \frac{1}{2}\|M(\mathbf{q})\mathbf{q}\|^2$ – which is the vector norm of the generalized momentum. However we hesitate to propose this or any other method for kinematic redundancy elimination, because we do not believe that redundancy elimination during planning is a good idea in the first place.

Mapping desired end-effector velocities to joint velocities has traditionally been used for open-loop planning. In contrast, our method makes it possible to perform closed-loop control with end-effector velocity as the high-level control signal. All we have to do is redefine $\mathbf{y}$ as $\mathbf{y} = \mathbf{p}$, while the plant is still $\mathbf{x} = [\mathbf{q}; \dot{\mathbf{q}}]$. Now $\mathbf{h}(\mathbf{x}) = \mathbf{k}(\mathbf{x})$ and so $H(\mathbf{x}) = [J(\mathbf{q}), 0]$. The prescribed high-level dynamics are $\mathbf{f}(\mathbf{y}) = 0$, $G(\mathbf{y}) = \mathrm{I}$, and the plant dynamics are the same as in (50). In this case we have a dynamic compatibility problem: $H(\mathbf{x})B(\mathbf{x}) = 0$, and so in order to apply our method, the functions $\mathbf{a}$, B have to be replaced with their predictive form $\mathbf{a}$, B.

5 Numerical simulations

5.1 Comparison to robotic control methods

Our method was compared to existing methods in Robotics on a family of robotic manipulators (Fig 4a), which were simulated with the Matlab Robotics Toolbox [45]. The manipulators had between 2 and 10 hinge joints and moved in the plane.

The kinematics were scaled so that all links of a given manipulator were equal in length, and the end-effector (filled circle) was 1m away from the base in the shown configuration. The dynamics were also scaled, so that all links had the same mass and the sum of all link masses was 1kg. Material density was kept constant, and set so that a 1m cylindrical link with diameter 0.1m had mass 1kg. The link diameter for each manipulator was computed given the constraint on total mass and the fixed material density. Link inertia was then found assuming uniform density.

We compared the three kinematic methods given in (56) and the two dynamics methods given in (54), as follows. A database of joint positions $\mathbf{q}$ and joint velocities $\dot{\mathbf{q}}$, along with desired end-effector accelerations $\ddot{\mathbf{p}}^*$, was generated randomly. For each manipulator, the database had 50,000 entries sampled uniformly from $q_i \in [-\pi, \pi]$, $\dot{q}_i \in [-1, 1]$, $\ddot{p}_i^* \in [-10, 10]$. Each method was used to compute a torque τ that yields end-effector acceleration $\ddot{\mathbf{p}}^*$ when the plant is in state $[\mathbf{q}; \dot{\mathbf{q}}]$. The dynamic methods (54) perform that computation directly. Using the kinematic methods (56) is more complicated, but possible. We first computed the current end-effector velocity $\dot{\mathbf{p}}$ from (47),

and the desired velocity at a time $\Delta = 0.01$s later as $\mathbf{p}*(\Delta) = \mathbf{p} + \Delta\dot{\mathbf{p}}*$. Then $\mathbf{p}*(\Delta)$ was mapped to $\mathbf{q}*(\Delta)$ using each kinematic method, the desired joint acceleration was computed as $\ddot{\mathbf{q}}* = (\mathbf{q}*(\Delta) - \mathbf{q}) / \Delta$, and used in the inverse dynamics (43) to obtain the torque τ. Despite this complication, both kinematic and dynamic methods achieved the desired end-effector acceleration in the absence of noise.

We compared two indices of performance, averaged over the entire database for each manipulator. The first index was control energy (Fig 4b). In the non-redundant case (dof=2) all methods are identical, but as redundancy increases we see clear differences. The dynamic methods outperform the kinematic ones, and in particular our method (hierarchical control) outperforms all others. The second index is the error in end-effector acceleration, caused by control-multiplicative noise injected in the computed torque: $\tau_i^{\text{actual}} = (1 + 0.1\varepsilon_i)\tau_i^{\text{computed}}$, where $\varepsilon_i \sim N(0, 1)$. The results in Fig 4c are not identical to Fig 4b, despite the fact that noise is proportional to torque, because the nonlinear mapping from torques to end-effector accelerations makes a difference. But the rank order of the five methods remains unchanged.

The above simulations are encouraging – especially since they do not re‡ect the full power of our method. This is because the version (54) of the method was derived for trivial high-level dynamics (51), not adapted to the plant dynamics. The next example illustrates the advantage of such adaptation. We defined an optimal control problem with the following total cost:

$$\| \mathbf{p}(T) - \mathbf{p}* \|;^2 + w\Delta \sum_{t=1}^{T-1} \| \tau(t) \|^2 \tag{57}$$

where $\mathbf{p}*$ is the target of a reaching movement executed with the 2-dof manipulator, $T = 500$ is the number of time steps, step duration is $\Delta = 0.001$s (resulting in a 0.5s movement), and $w = 0.0001$ is the weight for the control cost. Targets were 0.3m away from the starting position, in eight equally spaced directions. This simulation also included gravity (downward direction in the figures).

We compared the performance of our method, with $\mathbf{y} = [\mathbf{p}; \dot{\mathbf{p}}]$, applied to trivial high-level dynamics (51) versus high-level dynamics adapted to the plant. The latter were obtained by fitting a linear dynamical system model to data from the former set of simulations. The dataset contained entries of the form $(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}, \mathbf{v}, r)$, where $\mathbf{v}$ is the end-effector force $\mathbf{v} = J(\mathbf{q})^{\dagger}\tau$, and r is the control cost $r = w\Delta\|\tau\|^2$. There were 4,000 such entries.

The model fitting was based on the equations

$$\begin{aligned} \ddot{\mathbf{p}} &= F[\mathbf{p}; \dot{\mathbf{p}}] + G\mathbf{v} \\ r &= \mathbf{v}^{\mathrm{T}} R \mathbf{v} \end{aligned} \tag{58}$$

These equations are linear in the unknown matrices F, G, R, and therefore can be solved in a least squares sense via linear regression. The fit accounted for 87% of the variance in $\ddot{\mathbf{p}}$ and 95% of the variance in r, despite the nonlinear dynamics of the manipulator. Once F, G, R were computed, the adapted high-level dynamics could be defined as (58) by replacing $\mathbf{v}$ with $\mathbf{v}$. The fit also yielded a control cost model $\mathbf{v}^{\mathrm{T}}R\mathbf{v}$ on the high level. In case of the trivial dynamics, that cost model was in the same form but with $R = w\Delta\mathrm{I}$. Thus in both cases we have linear high-level dynamics, and quadratic cost

$$\| \mathbf{p}(T) - \mathbf{p}* \|^2 + \sum_{t=1}^{T-1} \mathbf{v}(t)^{\mathrm{T}} R \mathbf{v}(t) \tag{59}$$

which corresponds to a linear-quadratic optimal control problem. The optimal high-level controller in each case was found with standard LQR techniques, and the resulting two-level control scheme was applied to the manipulator.

Results are shown in Fig 5. For the trivial dynamics (before learning) the end-effector paths were straight as expected. For the adapted dynamics (after learning) the paths became systematically curved. As a result of this strategy, the total cost (57) was reduced by 23%. The improvement was largely due to the adapted controller's ability to take advantage of gravity.

5.2 Application to arm movements, and comparison to optimal control

Here we apply our method to a model of the human arm (Fig 6a). The details of this model can be found elsewhere [46], so we will keep the description brief. The skeletal kinematics and dynamics are identical to the above 2-dof robotic manipulator (apart from parameter settings), but the muscle actuators make this model more complex: the dynamics become third-order, and the controls become constrained. The muscles acting on the human arm in the horizontal plane can be organized into six actuator groups: shoulder flexors (SF) and extensors (SX), elbow flexors (EF) and extensors (EX), bi-articular flexors (BF) and extensors (BX). The tension produced by muscle i depends on its physiological cross-sectional area PCSA_i and activation state a_i, as well as the muscle length and velocity. The substantial length-and-velocity dependence is illustrated in Fig 6a, for maximal activation $a = 1$. This surface is based on the Virtual Muscle model [47], which provides a state-of-the-art fit to a range of physiological data. Tensions are multiplied by moment arms to yield joint torques. Moment arms are posture-dependent (Fig 6a), in a way consistent with experimental data. They also determine the mapping from joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ to muscle lengths and velocities. Muscle activation states have first-order low-pass filter dynamics $\dot{a}_i = (u_i - a_i) / \tau$. The controls u_i are constrained to the range $[0, 1]$. The complete system has a 10D state vector $\mathbf{x} = [q_1; q_2; \dot{q}_1; \dot{q}_2; a_1; \cdots a_6]$ and a 6D control vector $\mathbf{u} = [u_1; \cdots u_6]$.

The task is reaching: the arm starts from rest at $(q_1 = \pi/4; q_2 = \pi/2)$, and has to reach a specified target in 0.5s, with minimal control energy. The cost-per-step is

$$s(t) \| \mathbf{p}(t) - \mathbf{p}^* \|^2 + \mathbf{u}(t)^{\mathrm{T}} R \mathbf{u}(t) \tag{60}$$

where $s(t) = 1$ when $t > 0.4$s and 0 otherwise (we want the hand to remain stationary at the target for the last 0.1s). The diagonal matrix R has entries proportional to PCSA_i –because larger muscles generate more force per unit activation, and therefore consume more metabolic energy. Targets are arranged in a circle with 15cm radius around the start position.

We designed two hierarchical controllers, for $\mathbf{y} = \mathbf{p}$ and $\mathbf{y} = [\mathbf{p}; \dot{\mathbf{p}}]$ respectively. Because of the third-order dynamics, our method relied on the predictive functions $\mathbf{a}$, B in both cases. The control inequality constraints made the explicit solutions (11,33) inapplicable, and therefore the low-level controller used Quadratic Programming at each time step to compute $\mathbf{u}(\mathbf{v}, \mathbf{x})$. Instead of modeling the high-level dynamics, here we used the implicit method described above (which yields a high-level feedback controller). In addition to the two hierarchical controllers, we also computed the optimal feedback controller in each case, using our iLQG method [38].

Fig 6b shows deterministic and stochastic hand paths for each controller. Stochastic trajectories were simulated by corrupting each control signal with 100% control-multiplicative noise. The differences on the level of kinematics are small, although they increase (especially for the $\mathbf{y} = \mathbf{p}$ controller) when noise is added. Note that although the $\mathbf{y} = [\mathbf{p}; \dot{\mathbf{p}}]$ controller appears to be more accurate than the optimal controller, that is not the case – the optimal controller allows more variability during the movement but reaches the target more accurately. Note also that

the controllers can sustain such a large amount of noise because we are not modeling sensory delays and noise here. Fig 6c shows the (open-loop) muscle activations. Each subplot is one muscle-controller combination. The horizontal axis corresponds to movement direction, while the vertical axis is time during the movement (increasing downward). Dark means higher activation. We now see a much more clear distinction between the two hierarchical controllers. The muscle activations found by the $\mathbf{y} = [\mathbf{p}; \mathbf{p}]$ controller are quite similar to the optimal muscle activations, and furthermore resemble many features of experimentally observed muscle activations (but that is beyond the scope of the paper). On the other hand, the $\mathbf{y} = \mathbf{p}$ controller misses the elaborate temporal pattern of muscle activation, although it still activates the appropriate muscles.

Apart from demonstrating the power and generality of the proposed method, these results are important with regard to prior models of end-effector arm movement control [29–34]. Our method completes these models by going all the way to muscle activations. It also reveals a problem. The relatively poor performance of the $\mathbf{y} = \mathbf{p}$ controller suggests that controlling hand position through instantaneous velocity commands is not a good idea –because such commands are too far removed from the muscles that have to carry them out. In contrast, high-level commands related to hand force rather than velocity afford hierarchical control that is much closer to optimal.

5.3 Internal dynamics and sensorimotor synergies

Here we study a tracking task executed with a redundant linear plant. Linearity allows us to look more closely at the structure of the hierarchical controller, and compare it to the structure of the corresponding optimal controller. We also explore the behavior within the redundant subspace where the high-level controller cannot act.

The plant (Fig 7a) consists of two points (H,E) with mass 1, driven by three linear muscle-like actuators that can both push and pull. Actuator i has activation state ai and generates force $a_i - bq_i - k(q_i - \bar{q}_i)$, where $b = 10$, $k = 10$ are intrinsic damping and stiffness, q_i is the length of the actuator, and $\bar{q}_i = 0.3$ is the resting length of the spring. We can think of point H as the "hand", and point E as the "elbow" of a telescopic linear "arm". At each time step we are given a target position p^*. The task is to track the target with the hand. The state vector is $\mathbf{x} = [q_1; q_2; \dot{q}_1; \dot{q}_2; a_1; a_2; a_3; 1; p*]$. The constant 1 is needed to implement the spring model, and p^* is included in the state because it varies over time. Note that q_3 is not included in the state because $q_3 = q_1 + q_2$. The activation dynamics are $\dot{a}_i = (u_i - a_i) / \tau$ with time constant 40msec. The control vector is $\mathbf{u} = [u_1; u_2; u_3]$. Tracking is formulated as an optimal control problem with cost-per-step $\mathbf{x}^T Q\mathbf{x} + \mathbf{u}^T R\mathbf{u}$. We set $R = 10^{-5}$, and $Q = H^T H$ where $H = [1, 1, 0, 0, 0, 0, 0, 0, -1]$. This encodes the tracking error $(q_1 + q_2 - p^*)^2$, where $q_1 + q_2$ is hand position. Representing $p^*(t)$ as Brownian motion, the problem becomes LQG and can be solved with standard methods.

The optimal control law is $\mathbf{u} = L^o\mathbf{x}$, where L^o is a 3x9 matrix of optimal feedback gains.

The high-level state is defined as the instantaneous tracking error:

$$y = q_1 + q_2 - p* \tag{61}$$

Note that $y = H\mathbf{x}$, with H as defined above. The high-level dynamics and cost are $\dot{y} = v$ and y^2 respectively. Because of the linearity of the plant, the mapping from v to $\mathbf{u}$ is now linear: $\mathbf{u} = Kv$, where K is computed from (11). The optimal high-level control law is also linear: $v = gy$, where the gain g is found via LQG techniques. Then the hierarchical controller is

$$\mathbf{u} = L\,\mathbf{x}, \quad \text{where } L = KgH \tag{62}$$

To compare the structure of the hierarchical and optimal controllers, we applied singular value decomposition to the optimal L^o and found that it has singular values (447.6, 1.9, 0) – which is essentially rank 1. So we can write the optimal controller as

$$\mathbf{u} = L^o\mathbf{x}, \quad \text{where } L^o \approx K^o g^o H^o \tag{63}$$

and K^o and H^o are the left and right singular vectors of L^o corresponding to the largest singular value. The elements of vectors H and H^0 (normalized to unit length) are compared in Fig 7b (left), and K and K^0 are compared in Fig 7b (right). The similarity in both cases is striking. Note that of these four vectors, the only one we set manually was H. Then K was computed automatically by our method, while K^o and H^o fell out of (non-hierarchical) optimal feedback control. In summary, the structure of the hierarchical and optimal feedback controllers can be is very similar.

This linear “arm” model is interesting because the augmented plant has internal dynamics: point E is free to move without affecting the task outcome. Of course it cannot move arbitrarily, since the actuators have stiffness and damping that couple points E and H. But the high-level controller has no means of monitoring and correcting the trajectory of E, because it sees only the trajectory of H. To explore the internal dynamics, we applied both the hierarchical and optimal control laws, with 100% control-multiplicative noise. Fig 7c,d show the trajectories of points H and E. While the “hand” H tracks the target reliably, the “elbow” E exhibits substantial variability for both controllers. The lack of control over E may seem undesirable, but in fact there is nothing undesirable about it –or else the optimal controller would not allow E to ‡uctuate. If we really want E to follow a certain trajectory, we should be honest about that fact and encode it in the performance criterion. Since the performance criterion for this task is not affected by E, we cannot blame our controller for failing to accomplish something we did not ask for. On the contrary, we should be pleased that it closely resembles the optimal controller.

6 Discussion

We presented a general framework for hierarchical feedback control of redundant systems. The design of the proposed control hierarchy involves: (**i**) specifying a set of high-level parameters and their desired dynamics; (**ii**) designing a low-level feedback controller which yields an augmented plant with the specified input-output behavior; (**iii**) designing a high-level feedback controller that solves the original control problem but operates on a simplified system. Our focus was on automating the design of the two feedback controllers, as well as the construction of high-level dynamics that mimic the plant dynamics. This provided a way of controlling the specified high-level parameters.

The choice of appropriate high-level parameters is presently an open question, although a few relevant comments can be made. One extreme case would be to use a scalar performance index as the only high-level parameter. Then the high-level commands will essentially say “perform better”, but will not contain information as to how to perform better. Clearly such commands are not useful to the low-level controller. The other extreme would be to include all the state information on the high level (perhaps transformed nonlinearly), but that would not in general lead to a simplification of the controller design problem. So we seek a middle ground, where the high level state contains task-related parameters that enter in the computation of the performance index, and perhaps other closely related parameters. The application to reaching movements, for example, revealed a benefit of including velocity as a high-level parameter, even though the state-dependent cost was only a function of position.

Another issue deserving closer examination is the internal dynamics. In the above numerical examples we did not attempt to shape that dynamics, although our framework provides a mechanism for doing so by specifying a potential function over the plant state. Yet we saw that both the structure and the behavior of the hierarchical controller can be very similar to the optimal controller. This issue is relevant not only in our framework, but in all methods for hierarchical control. Velocity control methods that yield integrable solutions [44,48] eliminate the internal dynamics altogether, by enforcing a one-to-one mapping from plant states to high-level (end-effector) states. However, such strategies are suboptimal and unlikely to be used by the nervous system. An alternative approach is to somehow choose high-level parameters that result in asymptotically stable internal dynamics (corresponding to the notion of "minimum phase" in feedback linearization). Asymptotic stability is of course appealing, but its relationship to optimal performance is unclear; how to achieve such stability without unnecessary waste of control energy is also unclear. Since we already know that optimal controllers for redundant tasks are in essence hierarchical, further analysis of their structure can illuminate the choice of both the high-level state and the potential function used to shape the internal dynamics.

Acknowledgements

This work was funded by US National Institutes of Health Grant R01-NS045915. E. Todorov is a Sloan Research Fellow.

References

1. Sherrington, C. The integrative action of the nervous system. New Haven: Yale University Press; 1906.
2. Bernstein, N. Dexterity and its development. In: Latash, M.; Turvey, M., editors. Dexterity and its development. Lawrence Erlbaum; 1996.
3. Loeb G, Brown I, Cheng E. A hierarchical foundation for models of sensorimotor control. Exp Brain Res 1999;126(1):1–18. [PubMed: 10333003]
4. Kalaska, J.; Sergio, L.; Cisek, P. Cortical control of whole-arm motor tasks. In: Glickstein, M., editor. Sensory guidance of movement: Novartis Foundation Symposium. Chichester, UK: John Wiley and Sons; 1998.
5. Brooks R. A robust layered control-system for a mobile robot. IEEE Journal of Robotics and Automation 1986;2(1):14–23.
6. Khatib O. A unified approach to motion and force control of robotic manipulators: The operational space formulation. IEEE Journal of Robotics and Automation 1987;RA-3(1):43–53.
7. Pratt J, Chew C, Torres A, Dilworth P, Pratt G. Virtual model control: An intuitive approach for bipedal locomotion. International Journal of Robotics Research 2001;20(2):129–143.
8. Precup, D.; Sutton, R. Advances in Neural Information Processing. 10. 1998. Multi-time models for temporally abstract planning.
9. Dietterich T. Hierarchical reinforcement learning with the maxq value function decomposition. Journal of Artificial Intelligence Research 2000;13:227–303.
10. Todorov E. Optimality principles in sensorimotor control. Nature Neuroscience 2004;7(9):907–915.
11. Todorov E, Jordan M. Optimal feedback control as a theory of motor coordination. Nature Neuroscience 2002;5(11):1226–1235.
12. Todorov, E.; Jordan, M. A minimal intervention principle for coordinated movement. In: Becker, S.; Thrun, S.; Obermayer, K., editors. Advances in Neural Information Processing Systems. 15. Cambridge, MA: MIT Press; 2002. p. 27-34.
13. Isidori, A. Nonlinear Control Systems. London: Springer-Verlag; 1995.
14. Khalil, H. Nonlinear Systems. New Jersey: Prentice-Hall; 2002.
15. Sutton G, Sykes K. The variation of hand tremor with force in healthy subjects. Journal of Physiology 1967;191(3):699–711. [PubMed: 6051798]
16. Schmidt R, Zelaznik H, Hawkins B, Frank J, Quinn J. Motor-output variability: a theory for the accuracy of rapid notor acts. Psychol Rev 1979;86(5):415–451. [PubMed: 504536]

17. Todorov E. Cosine tuning minimizes motor errors. Neural Computation 2002;14(6):1233–1260. [PubMed: 12020444]

18. Collins A, Ruina A, Tedrake R, Wisse M. Efficient bipedal robots based on passive-dynamic walkers. Science 2005;307:1082–1085. [PubMed: 15718465]

19. Anderson F, Pandy M. Dynamic optimization of human walking. J BiomechEng 2001;123(5):381–390.

20. Sabes P, Jordan M, Wolpert D. The role of inertial sensitivity in motor planning. JNeurosci 1998;18 (15):5948–5957. [PubMed: 9671681]

21. Li, W.; Todorov, E.; Pan, X. Hierarchical optimal control of redundant biomechanical systems. 26th Annual IEEE Conference on Engineering in Biology and Medicine; 2004.

22. Loeb G, Levine W, He J. Understanding sensorimotor feedback through optimal control. Cold Spring Harb Symp Quant Biol 1990;55:791–803. [PubMed: 2132855]

23. Hoff, B. A computational description of the organization of human reaching and prehension. Ph.D. Thesis, University of Southern California; 1992.

24. Kuo A. An optimal control model for analyzing human postural balance. IEEE Transactions on Biomedical Engineering 1995;42:87–101. [PubMed: 7851935]

25. Shimansky Y, Kang T, He J. A novel model of motor learning capable of developing an optimal movement control law online from scratch. Biological Cybernetics 2004;90:133–145. [PubMed: 14999480]

26. Bernstein, N. The Coordination and Regulation of Movements. Pergamon Press; 1967.

27. Scholz J, Schoner G. The uncontrolled manifold concept: identifying control variables for a functional task. Exp Brain Res 1999;126(3):289–306. [PubMed: 10382616]

28. Latash, M. On the evolution of the notion of synergy. In: Gantchev, G.; Mori, S.; Massion, J., editors. Motor Control, Today and Tomorrow. Sofia: Academic Publishing House “Prof. M. Drinov”; 1999. p. 181-196.

29. Hinton G. Parallel computations for controlling an arm. Journal of Motor Behavior 1984;16(2):171–194. [PubMed: 14713664]

30. Pellionisz A. Coordination: a vector-matrix description of transformations of overcomplete cns coordinates and a tensorial solution using the moorepenrose generalized inverse. Journal of Theoretical Biology 1984;110:353–375. [PubMed: 6503306]

31. Meyer D, Abrams R, Kornblum S, Wright C, Smith J. Optimality in human motor performance: Ideal control of rapid aimed movements. Psychological Review 1988;95:340–370. [PubMed: 3406245]

32. Bullock D, Grossberg S. Neural dynamics of planned arm movements: emergent invariants and speed-accuracy properties during trajectory formation. Psychological Review 1988;95:49–90. [PubMed: 3281179]

33. Hoff B, Arbib M. Models of trajectory formation and temporal interaction of reach and grasp. J Mot Behav 1993;25(3):175–192. [PubMed: 12581988]

34. Torres E, Zipser D. Reaching to grasp with a multi-jointed arm. i. computational model. Journal of Neurophysiology 2002;88(5):2355–2367. [PubMed: 12424277]

35. English J, Maciejewski A. On the implementation of velocity control for kinematically redundant manipulators. IEEE Trans on Systems, Man, and Cybernetics, Part A 2000;30:233–237.

36. Bryson, A.; Ho, Y. Applied Optimal Control. Massachusetts: Blaisdell Publishing; 1969.

37. Jacobson, D.; Mayne, D. Differential Dynamic Programming. New York: Elsevier; 1970.

38. Todorov, E.; Li, W. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In Proceedings of the American Control Conference; 2005.

39. Branicky M, Brokar V, Mitter S. A unified framework for hybrid control: Model and optimal control theory. IEEE Transactions on Automatic Control 1998;43:31–41.

40. Sastry S. Hybrid control in air traffic management systems. IEEE Conference on Decision and Control 1995;34

41. Hedlund S, Rantzer A. Optimal control of hybrid systems. IEEE Conference on Decision and Control 1999;43

42. Pratt, J.; Pratt, G. Intuitive control of a planar bipedal walking robot. IEEE International Conference on Robotics and Automation (ICRA ’98); 1998.

43. Bruyninckx H, Khatib O. Gauss' principle and the dynamics of redundant and constrained manipulators. IEEE Int Conf Robotics Automation 2000:2563–2568.

44. Mussa-Ivaldi A, Hogan N. Integrable solutions of kinematic redundancy via impedance control. Int J Robotics Research 1991;10:481–491.

45. Corke P. A robotics toolbox for matlab. IEEE Robotics and Automation Magazine 1996;3:24–32.

46. Li, W.; Todorov, E. Iterative linear-quadratic regulator design for nonlinear biological movement systems. 1st International Conference on Informatics in Control, Automation and Robotics; 2004.

47. Brown I, Cheng E, Loeb G. Measured and modeled properties of mammalian skeletal muscle. ii. the effects of stimulus frequency on force-length and force-velocity relationships. JMuscle ResCell Motil 1999;20(7):627–643.

48. Baillieul J. Avoiding obstacles and resolving kinematic redundancy. Proc IEEE Int Conf Robotics and Automation 1986:1698–1704.

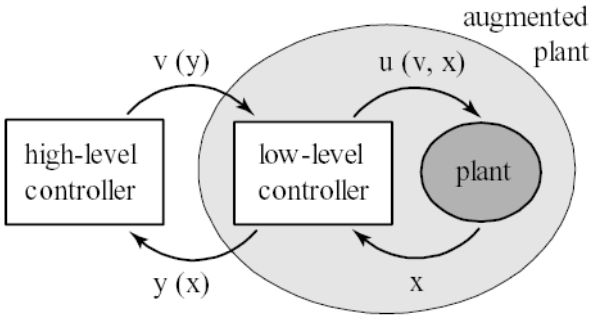

Figure 1.
schematic illustration of the proposed framework.

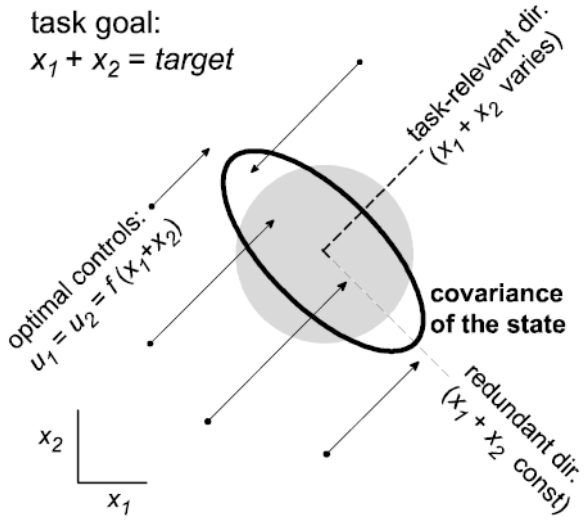

Figure 2.
properties of optimal feedback controllers in redundant tasks (from ref. 10)

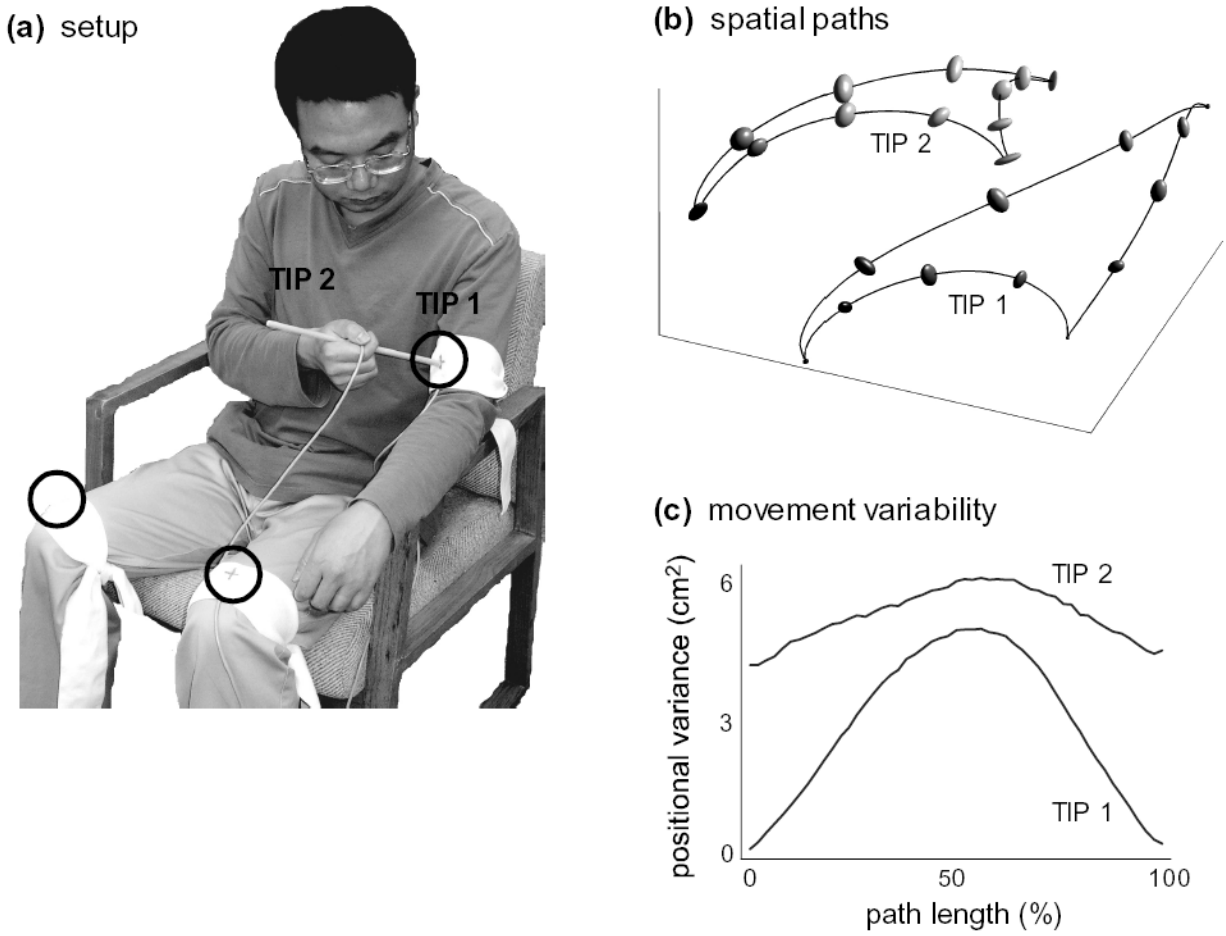

Figure 3.
experimental illustration of increased variability in redundant dimensions.

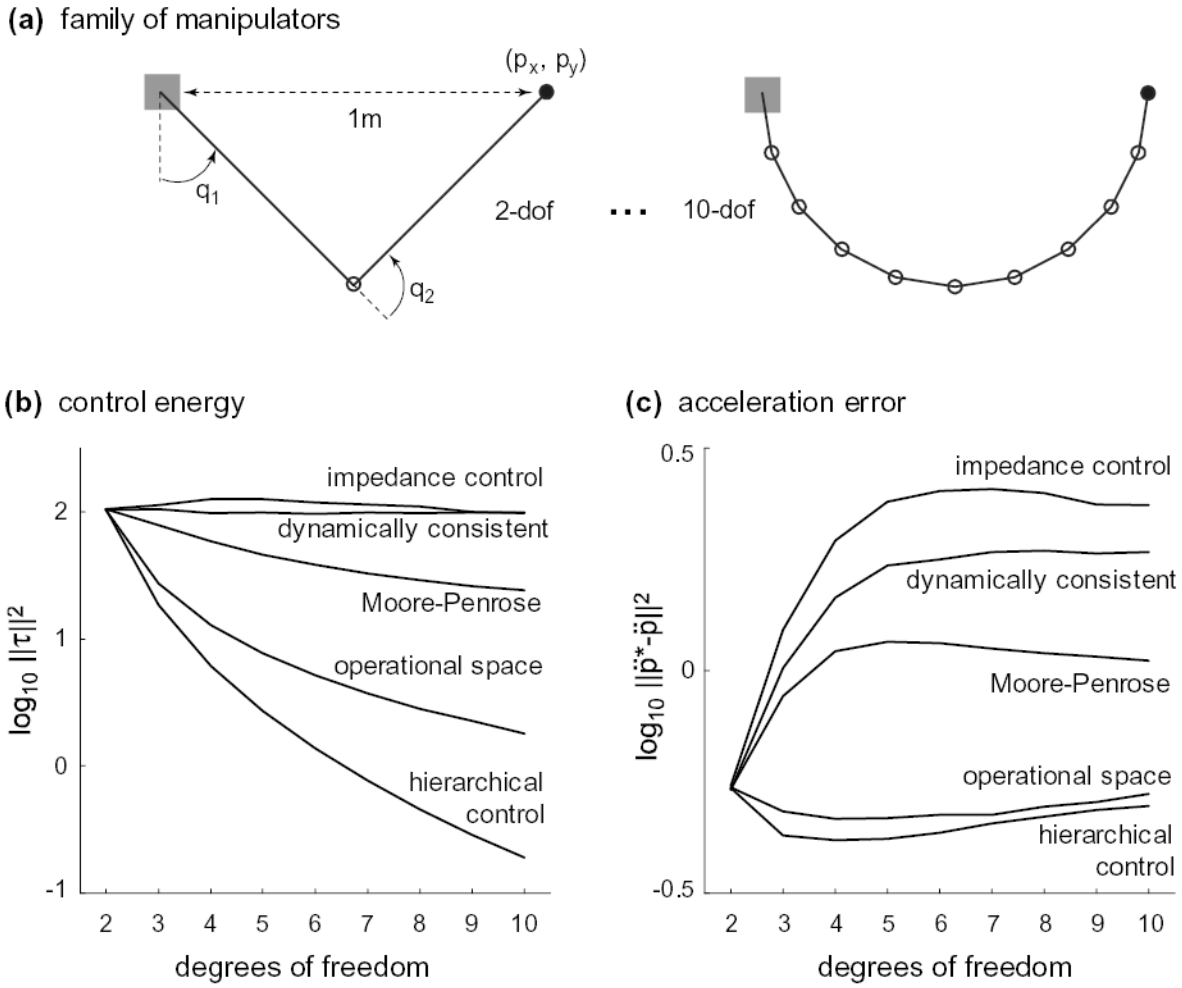

Figure 4.
comparison of control methodologies on a family of robotic manipulators.

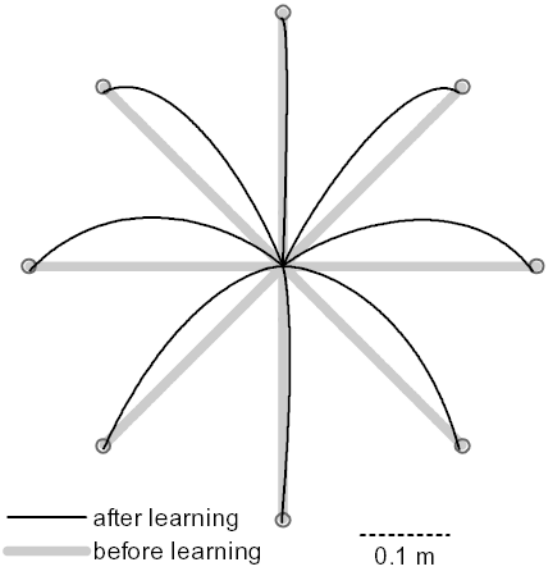

Figure 5.
effects of adapting the high-level dynamics to the plant dynamics.

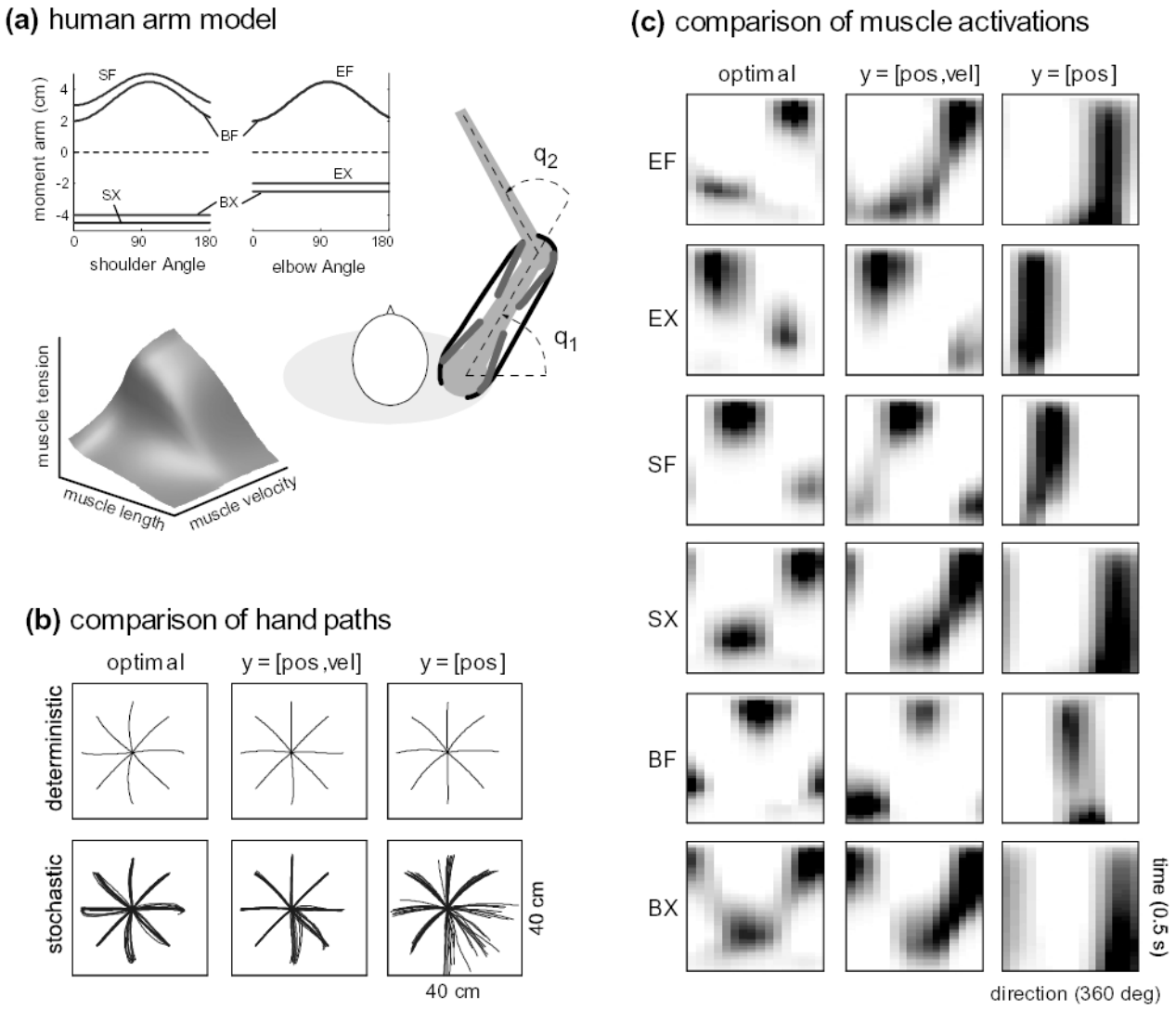

Figure 6.
comparison to the optimal controller in human reaching.

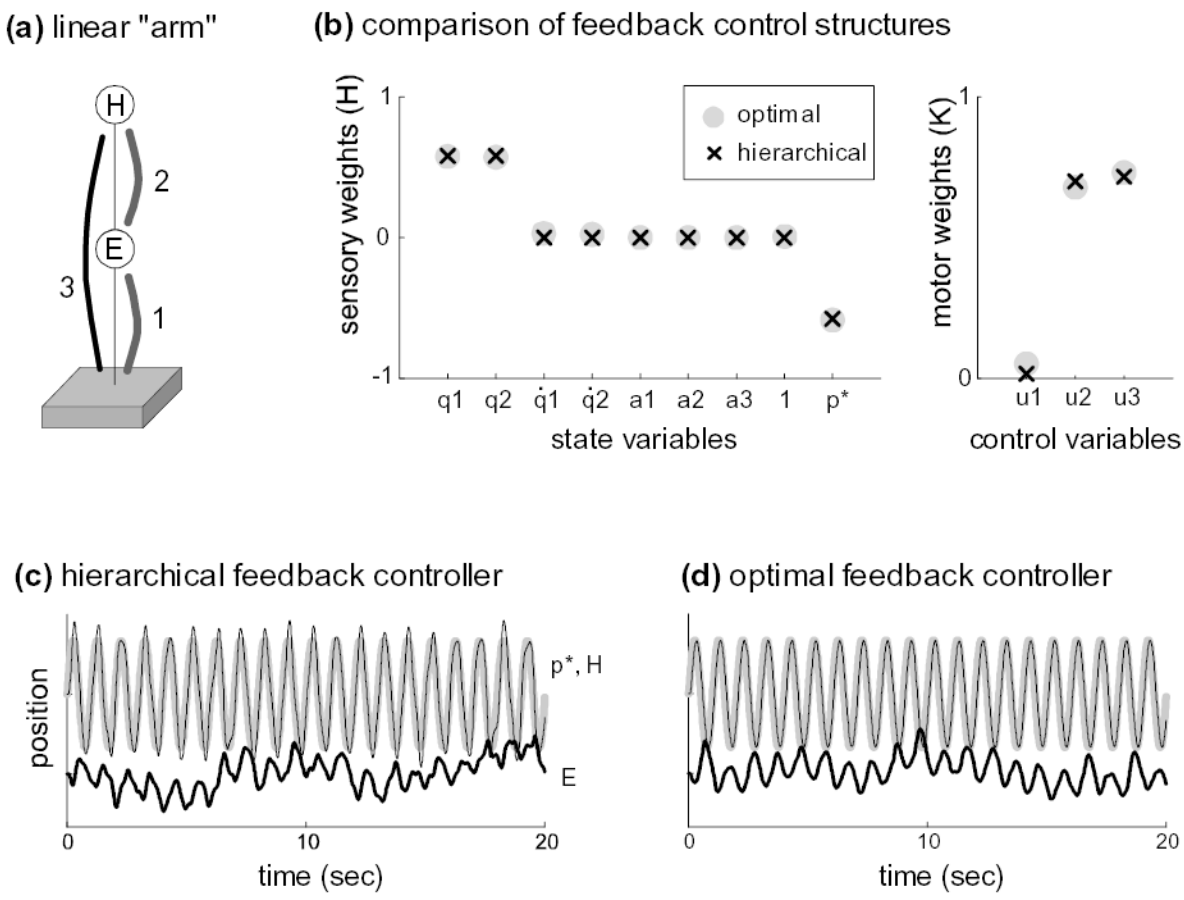

Figure 7.
internal dynamics and sensorimotor synergies.