Dynamic Programming in Reduced Dimensional Spaces: Dynamic Planning For Robust Biped Locomotion

Mike Stilman, Christopher G. Atkeson, James J. Kuffner, Garth Zeglin
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213, USA
robot@cmu.edu, cga@cs.cmu.edu, kuffner@cs.cmu.edu, garthz@ri.cmu.edu

***Abstract*— We explore the use of computational optimal control techniques for automated construction of policies in complex dynamic environments. Our implementation of dynamic programming is performed in a reduced dimensional subspace of a simulated four-DOF biped robot. We show that a computed solution to this problem can be generated and yield empirically stable walking that can handle various types of disturbances.**

***Index Terms*— dimensionality reduction, dynamic programming, biped locomotion**

I. Introduction

Our goal is to apply optimal control techniques that compute a policy for complex nonlinear systems. In this paper we focus on dynamic programming (DP) [20] [7]. Such techniques complement trajectory optimization techniques and can help avoid the poor local optima that make application of trajectory optimization difficult [13]. They also enable the use of model-based reinforcement learning, which will allow us to handle situations with uncertain models. In practical implementations we expect to refine the limit cycles that result from our computed policies using trajectory optimization techniques such as DIRCOL [19], and to evaluate the policy around the trajectory for local stability.

Previously, DP and related methods were implemented on small scale problems due to insufficient computational resources. In this work we strive to show that, under appropriate state space decomposition and model reduction, DP can currently be applied to significantly larger scale domains. Task subdivision is not uncommon in the domain of biped locomotion. [16] used a functional decomposition with separate controllers for speed, altitude and attitude. [3], [21] solve the locomotion problem analytically by formulating virtual constraints and mapping to a two dimensional submanifold. From a computational perspective, we map to low dimensional subspaces in order to increase the resolution of the state space grid under the constraints of memory and processing time.

In this paper, we manually perform a temporal decomposition of the problem into phases of single and double support (SS,DS). We show that DS on level ground can then be modeled fully and introduce a further model reduction of SS. The state space and model components are joined by mapping and the optimal policy is computed iteratively over the entire represented state space. The reductions performed are intended to be intuitive and illustrative of the practical effectiveness of DP through visual representations and empirical results.

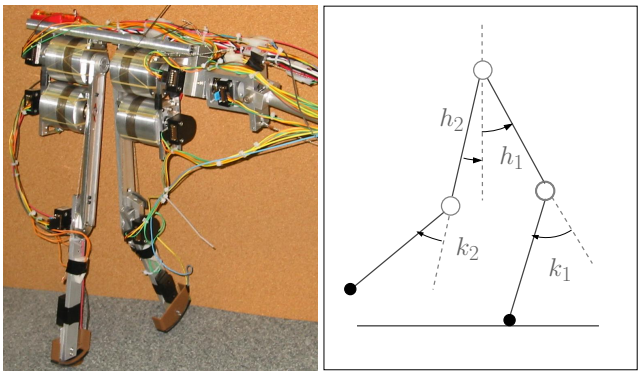


Fig. 1. The real biped with boom and planar model used in our work. The biped has point feet and no upper body.

Our future work will involve the application of reduced order modeling such as proper orthogonal decomposition [12] and truncated-balanced realization [15]. This will allow us to enforce stability constraints and prove robustness bounds for the computational methodology.

II. State Increment Dynamic Programming

A. Foundations

Our algorithm is a practical extension to grid-based, state-increment dynamic programming (DP). A typical DP problem consists of the triple $(\mathcal{S}, \mathcal{U}, \mathcal{C}[x(t), u(t), t])$.

- $\mathcal{S}$ represents the state space.
- $\mathcal{U}$ is the action space.
- $\mathcal{C}$ is the cost, a function of the state and action at each step.

For periodic tasks, such as walking, there is no goal state. In our work, $\mathcal{C}$ is a weighted difference between the horizontal torso velocity and a desired velocity, combined with an action penalty. α is the weight and γ is a discount factor.

$$\mathcal{C}[x(t), u(t), t] = \alpha(\dot{x}_{torso} - \dot{x}_{des})^2 + u^2 \quad (1)$$

$$V(x) = \sum_{k=1}^{\infty} \gamma^k \mathcal{C}[x(t), u(t), t] \quad (2)$$

Our task is to find an optimal mapping from states to actions, $\mathcal{P}(x)$, that minimizes $V(x)$. The performance criterion, $V(x)$ is a value function representing the cost-to-go from state x when following policy $\mathcal{P}$.

B. Implementation

The state space of our robot is eight-dimensional, $(h_1, h_2, k_1, k_2, \dot{h}_1, \dot{h}_2, \dot{k}_1, \dot{k}_2)$. Due to the high dimensionality of the problem, we discretize $\mathcal{S}$ on a coarse grid. We assume that the grid represents the portion of continuous state space of interest. $V(x)$ is evaluated by means of multilinear interpolation of corner values in the containing cell [4]. In Section III we introduce a state space decomposition that makes this approach tractable.

The optimal value function $\mathbf{V}(x)$ over $\mathcal{S}$ is then computed iteratively. We evaluate a discrete set of actions from each state according to the expected results of each action in the system model $\hat{\mathbf{f}}$. The action with minimum expected return is selected. We repeatedly apply the following value update, as derived from Bellman's equation:

$$V_{k+1}(x) = \min_{u \in \mathcal{U}} \sum_{t_0}^{\epsilon} \gamma^k \mathcal{C}[x(t), u(t), t] + V_k(\hat{\mathbf{f}}(x, u, \epsilon)). \tag{3}$$

In state-increment DP [7], the interval ϵ of action execution is selected individually for each pair (x, u). It is defined as the minimum time interval required for any of the state indices to change by one increment. This ensures that $\mathbf{V}(x)$ is computed from the neighborhood (in memory) of x, while not using $V_k(x)$ itself in the interpolation.

C. Illustrative Example

Before applying DP to the 8-dimensional space of our biped, briefly consider a simpler problem. We fix the biped knee and suspend the hip to create a simple pendulum. The state space consists of the hip angle and velocity $(\theta, \dot{\theta})$. Suppose we want to keep the system at a constant energy (i.e. swinging at the natural period). Let

$$E(\theta, \dot{\theta}) = \frac{1}{2} I \dot{\theta}^2 - mgl \cos\theta, \tag{4}$$

$$\mathcal{C}_{pend}[s(t), u(t), t] = \alpha(E - E_{des})^2 + u^2, \tag{5}$$

and $\hat{\mathbf{f}}$ be the dynamic forward simulation. Directly applying the update rule (3), yields $\mathbf{V}(x)$ in less than a minute of computation. Fig. 2 demonstrates the shape of $\mathbf{V}(x)$. For each state, the associated policy $\mathcal{P}$ selects an action that is expected to minimize $\mathbf{V}(x)$ in the following state.

The bottom of the elliptical valley in Fig. 2 is the limit cycle for the pendulum task.

III. State Space Considerations

In the remainder of this paper, we relate the dynamically complex task of walking to the example in Section II(C). Unfortunately, representing and iterating over an eight dimensional grid is currently too expensive. We decompose the state space into a compact yet useful representation.

Our four-joint robot shares the periodic walking pattern that is commonly associated with bipeds. The pattern

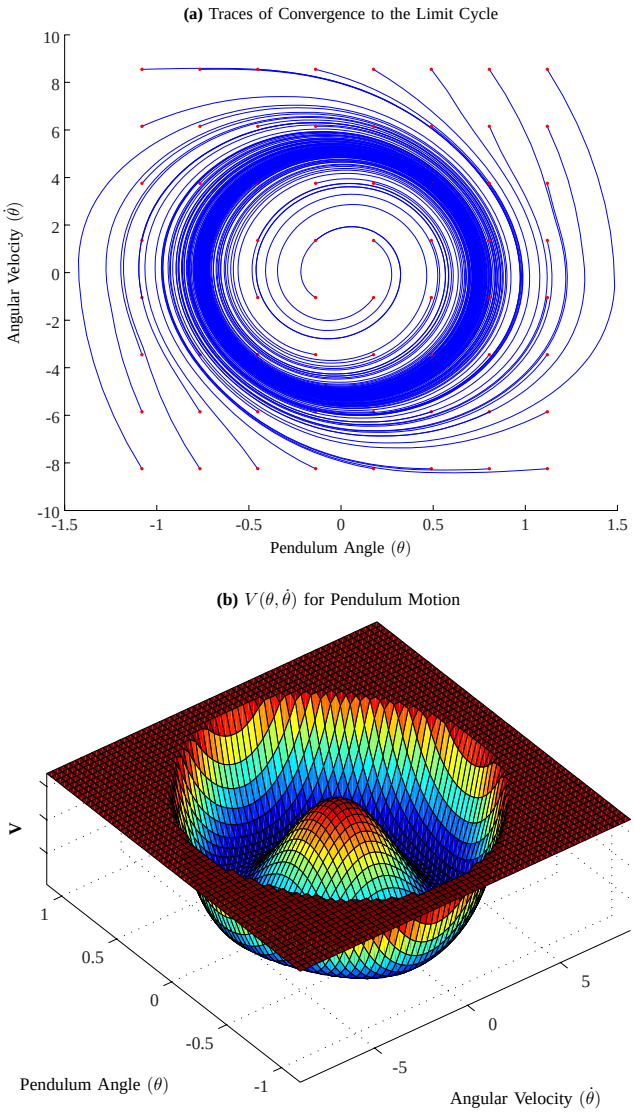


Fig. 2. (a) Traced trajectories of DP controlled motion from various points in the pendulum state space. (b) The pendulum value function shows a valley at the limit cycle.

consists of two alternating phases: Double Support(DS) and Single Support(SS). Since our algorithm takes into account both phases of motion we observe the following:

- Due to the underactuated nature of the robot, the successful completion of each phase relies highly on the system state when it enters the phase. For instance, when the robot enters single support, a minimal velocity of the stance leg is required for the hip to pass the stance foot. Likewise, after the swing foot makes ground contact, the distance between the feet cannot be changed throughout double support.
- There is an important distinction between the dynamics of DS and SS. The additional contact point during stance yields vastly different dynamic properties and action responses. For instance, a PCA analysis of the Jacobian for robot motion during SS would indicate that the swing knee torques have relatively little effect on the torso (COM) motion. Clearly, during DS, the

corresponding torques at the rear knee prescribe the acceleration of the COM, thereby creating a push-off.

In developing a compact state-action representation we seek features and actions that accurately approximate the behavior of the system. Any state representation that removes variables will be a slice in the state space. A useful representation should largely be invariant to changes in the excluded dimensions.

Though relevant features are not difficult to identify in each phase, the inconsistent dynamics of the system yield distinct selections of features that best represent DS and SS. The same is true for choices of actions. This observation points to the use of separate models and representations for each phase. In the context of a periodic system, however, DP cannot be applied separately to state space components. We need to establish a mapping between the components such that $\hat{\mathbf{f}}$ and therefore $V(\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u}, \epsilon))$ can be evaluated during DS and SS transitions.

IV. State Space Decomposition and Reduction

In this section, we detail our choices for state representation. An algorithmic procedure for making such choices remains an open problem and an interesting subject for future work.

A. Double Support: 5-Bar Linkage Reduction

Assuming a no-slip model of ground contact, with constant step length (d_f), we can represent the point foot contacts with ground hinges (given appropriate restrictions on velocity). As shown in Fig. 3(a) this model is equivalent to a 5-bar linkage. *Gruebler's Equation* indicates that the system has 2 degrees of freedom [8]. In fact, the entire system state can be described in terms of any two joints. For reasons of range and clarity, we chose k_1 and k_2. The remainder of the state can be resolved from the system constraints.

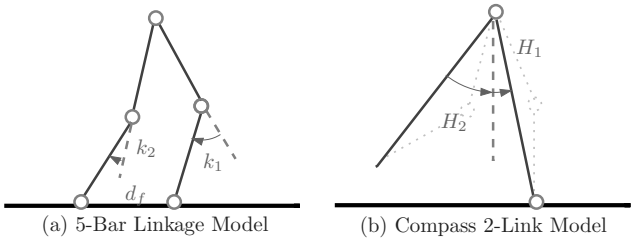
Due to the no-slip model, d_f cannot vary within a single pass through the DS phase and therefore adds no velocity component. Still, d_t may change after the robot completes the SS phase. The full state for DS is therefore a 5-tuple $(d_f, k_1, k_2, \dot{k}_1, \dot{k}_2)$. The DS Value Table (10x16x16x12x12) is used to model $V(x)$ at the respective resolutions: $(.02m, .075rad, .075rad, 1.25\frac{rad}{s}, 1.25\frac{rad}{s})/cell$.

In the following subsection, we represent the state during SS in terms of the hip-foot angles H_1, H_2. Equation (6) demonstrates the mapping from the stance leg angles in the DS model to those of the SS model. Analogously, we map swing leg angles, and derive angular velocities from the time derivatives of these equations.

$$H_1 = h_1 - \arcsin\left(\frac{l_{shin}\sin(k_1)}{\sqrt{l_{hip}^2 + l_{shin}^2 + 2l_{hip}l_{shin}\cos(k_1)}}\right) \tag{6}$$

B. Single Support: Compass Reduction

The added constraint during DS allowed us to represent the entire system state with only five variables. Though this is not possible during SS, we choose a compact state representation that is common in biped walking: the compass walker [5], [6], [9], [10], [18]. Our compass model assumes that k_1 and k_2 are constant, yielding a four-dimensional state space of hip angles and velocities $(H_1, H_2, \dot{H}_1, \dot{H}_2)$. In our representation H_1 and H_2 represent angles from the hip to the respective leg's foot. The SS value table (35x35x18x18) allows for resolutions of $(.046rad, .046rad, .50\frac{rad}{s}, .83\frac{rad}{s})/cell$.



(a) 5-Bar Linkage Model (b) Compass 2-Link Model

Fig. 3. The two models used in our state space representation.

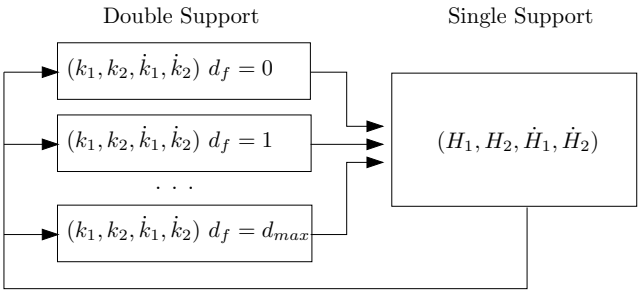


Fig. 4. Diagram of the decomposed state space. Since no actions directly connect stance components with different d_t, they are separated.

The model excludes k_1 and k_2 for separate reasons. The stance knee angle, k_1, has a small range in human walking.

k_2 on the other hand has a great deal of range. In fact, the final position of k_2 is directly related to d_f, the distance between the feet on contact. In contrast to the rest of the system, however, k_2 is well-actuated. The motion of k_2 can be prescribed in accordance with h_2 with little effect on the motion of the robot's COM or other system components. Prescribing desired positions for k_2, allows d_f to be controlled.

In our work, d_f is a dominant factor in expressing the preimage of the DS phase. The success of push-off largely depends on a *reasonable* distance between the feet. Though determining suitable distances was a task for our planner, we ensure that the compass state representation is sufficiently expressive in this dimension. In a transition state, the compass state is trivially mapped to the DS model as follows:

$$d_t = l_2\sin(H_2) - l_1\sin(H_1). \tag{7}$$

C. Complete State Space and Actions

The diagram in Fig. 4 represents the combined state space resulting from our decomposition. The transition to DS occurs when the swing leg makes contact with the ground. The model transitions to SS when the rear foot leaves the ground, violating a velocity constraint.

Transitions are a special case of motion in the state space. Generally, all motion is defined by the model $x' = \hat{\mathbf{f}}(x, u, \epsilon)$, where u is an action applied for time interval ϵ. In our formulation of dynamic programming, actions are discretized torques for each joint within the range of operation. We chose to discretize actions rather than numerically optimize in the continuous space, because doing so allows us to cache $\hat{\mathbf{f}}$. Consequently, (3) can be computed without additional model simulation.

The discrete action space $\mathcal{U}$ of the robot forms an interesting contrast to the state space. In the case of SS, our state is fully represented in terms of hip angles and velocities $H, \dot{H}$ Since the robot cannot be actuated at the feet, we are left with a one-dimensional $\mathcal{U}$ (a choice of hip torque). This is no different from the acrobot domain [2]. We sample $\mathcal{U}$ in the range $\pm 1.8Nm$ at $.6Nm$ intervals (17 actions).

In DS, however, the additional contact point adds significantly to the potential for actuation. The hip and both knee joints can accelerate the COM, yielding a three-dimensional action space. In practice, we fix the hip action to zero Nm to gain more significant resolution in knee torques. The resulting action space is two dimensional with each knee torque in the range $\pm 5.4Nm$ sampled at $.514Nm$ intervals (7x7).

The entire model $\hat{\mathbf{f}}$ has dimension equal to the product of the state and action spaces. It therefore contains $398640 \times 49 + 396900 \times 17 = 24810660$ cells.

V. Results

A. Generated Policy

In order to compute a policy, we first constructed a full planar simulation of the robot that defined $\hat{\mathbf{f}}$. Applying the dynamic programming method of Section II to the compact state space described in Section IV we automatically generated a global value function and policy in 11hr. of computation. Of these, 7hr. were dedicated to model caching. The resulting value function exhibits characteristics of convergence to a limit cycle similar to those in Fig. 2(a). We can see the progressive stabilization of several trajectories from different starting states to periodic locomotion in Fig. 5(a).

In order to visualize progress towards the limit cycle, consider Fig. 5(b-c). The shapes of slices in $\mathbf{V}(x)$ illustrate some of the automatically detected features in the locomotion space. Although this representation is simply a slice in the space, the relative values of neighboring states indicate preferable directions of motion. For instance, during push-off, the rear leg extends (decreasing the angle k_2) in order to accelerate the center of mass. The robot enters a singularity if the knee is fully extended. In Fig. 5(b), we see that the preferred angular velocity ($\dot{k_2}$) shows substantial increase as we approach the singularity. Experimentally, we see that the policy selects states with larger knee angles and smaller velocities. This corresponds to low values of $\mathbf{V}(x)$.

Fig. 5(c) yields analogous insight into swing. Clearly for the stance leg (H_1), position and velocity are inversely related. Before the COM crosses the stance foot ($H_1 > 0$), the velocity ($\dot{H}_1$) must be negative. Notice the narrow range of velocities handled by the policy in the vicinity of $H_1 = 0$. Other slices of $\mathbf{V}(H_1, H_2, \dot{H}_1, \dot{H}_2)$ that vary $(H_2, \dot{H}_2)$ exhibit the same narrow range. DP finds that no actions can significantly adjust $\dot{H}_1$ during SS. This consequence of underactuation verifies our earlier claims that the preimage of single support imposes a strict range on $\dot{H}_1$.

As the COM passes the stance foot with a small negative velocity, the angle H_1 becomes slightly negative. The slice in Fig. 5(d) focuses on a positive $\dot{H}_2$. Notice that the value function guides H_2 to a narrow corner as H_1 passes 0. For this slice, this is the computed desired angle of ground contact at which the controller applies negative torque to H_2. Hence $\dot{H}_2$ decreases and leaves this slice of $\mathbf{V}(x)$.

Further analysis of the optimal value function indicates a similar capacity of DP to automatically compute the features of our complex dynamic space. The computed policy locates a limit cycle through the space and calculates the actions that drive the system towards this cycle.

B. Successful Performance Under Error

In testing our approach, we constructed a 3D simulation of the biped and boom. The extended simulation contained small lateral perturbations and a frictional ground contact model. We introduced two simple PD-servo controls to constrain the lateral motion of the hips and move the swing knee during the SS phase. All significant aspects of the robot dynamics were handled by the policy computed using DP on a planar model. Following this policy, the robot successfully entered a stable walking gait on a flat surface.

To further test the response of our DP policy under error and uncertainty, we applied disturbances to the robot and environment models. Table II summarizes our results and Table I describes the physical parameters of our robot.

TABLE I

Robot and Dynamic Model Parameters

	Torso	Thigh	Shin
Mass (kg)	2.97	.64	.22
Length (m)	.01	.2	.22
		Hips	Knees
Torque Limits (Nm)		± 1.8	± 5.4

TABLE II

Perturbations Preserving Stable Walking

Random noise was sampled at the controller rate of .001s

Accepted Disturbance	Range
Torso Mass	$.01kg \leftrightarrow 3.5kg$
Thigh Mass (each thigh)	$.01kg \leftrightarrow 1.2kg$
Shin Mass (each shin)	$.12kg \leftrightarrow .26kg$
Ground Incline Up	$< 3.0^o$
Ground Incline Down	$< 2.4^o$
Uniform Random Grnd. Height	$\pm 1cm$
Uniform Random Hip Sensor Noise	$\pm .15rad$
Uniform Random Knee Sensor Noise	$\pm .3rad$
Uniform Random Hip Torque Noise	$\pm 1.0Nm$
Uniform Random Knee Torque Noise	$\pm 1.5Nm$

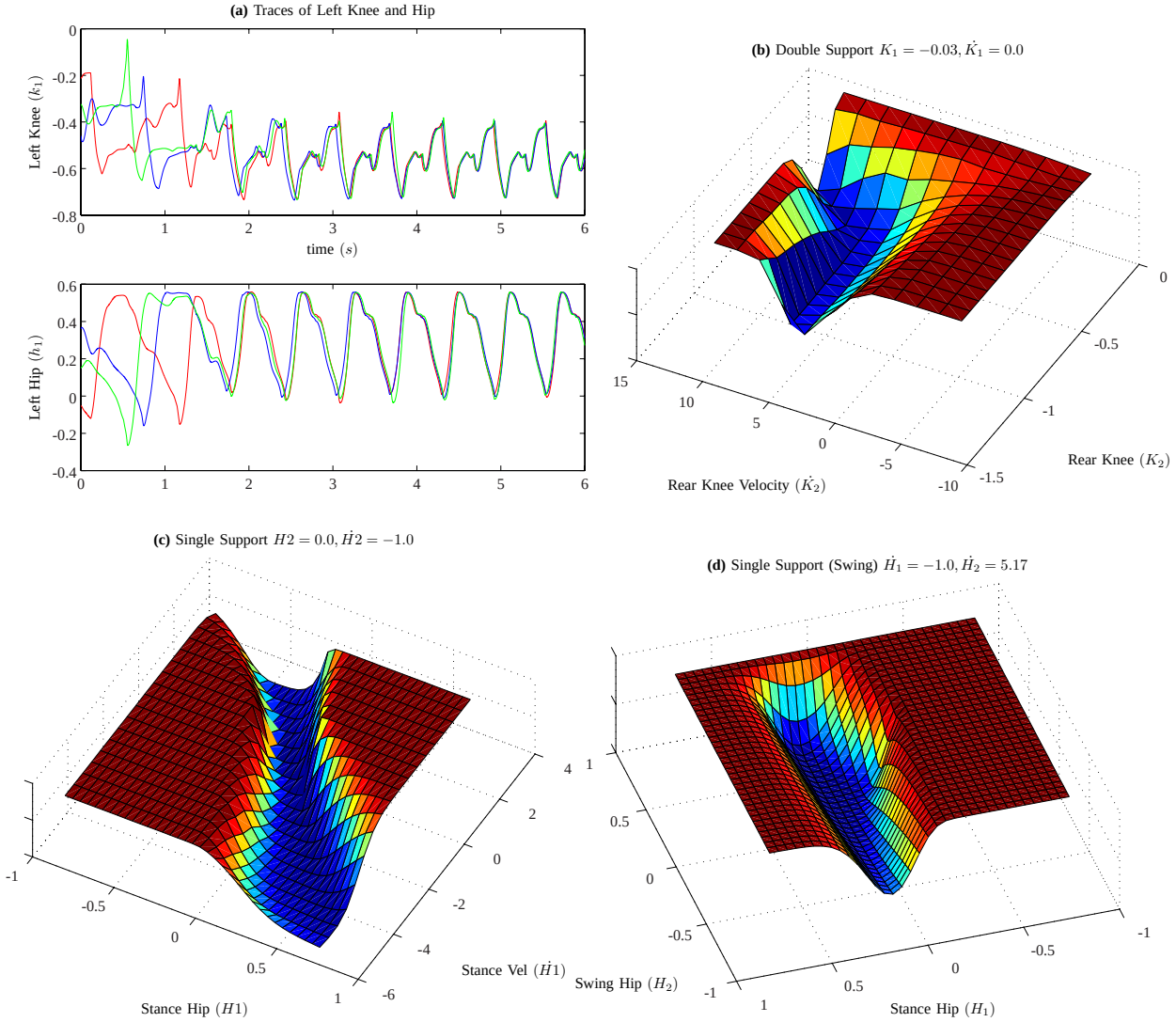


Fig. 5. (a) Traces of robot trajectories converging to the limit cycle when following the DP policy. (b) Selected slices of the computed value function. The vertical axis represents the value, s.t. among neighboring states lower values are preferred by the policy. One can also see the relative resolution of the grid for different state space components.

The disturbances in Table II can be categorized into two groups: correlated error and random, uncorrelated noise. Clearly in the case of random error, the system responded with relative ease. For both sensor readings and applied torques, the ranges successfully dealt with were well beyond observed noise. We observed that at the limits of the error bounds, the system exhibited large offsets from the limit cycle. Still, within the given range, the policy was able to walk forward successfully.

Correlated error is a more substantial challenge. In the simple case of modeling error, $\hat{\mathbf{f}}$ incorrectly represents the forward model of the state. DP selects actions based on the expected resulting state. Therefore, the action choices do not produce the expected motion towards the limit cycle. Consequently, the system can drift away from the attractor trajectory. The key advantage of DP is that we consider a large volume of state space. The wide set of acceptable states allows the actual trajectory to be distinct from the expected limit cycle. Our results show that when the torso mass varies between $1-120\%$, or the hip masses $1-200\%$ the unaltered policy enforces a lasting periodic limit cycle.

Consider the shin mass and the ground incline. In our tests, these were the least versatile components. Although both permit a significant error margin, ($\approx 6^o$ incline range), they appear relatively small. We attribute this largely to the fact that these components were entirely unmodeled during policy acquisition. While various joint angles are considered, the ground is always assumed to be flat. Furthermore, shin positions during SS are determined by an ad-hoc controller. Therefore optimal behavior for the shins is not achieved. In future work, we should consider developing appropriate measures for incorporating these components into the policy.

Perhaps the most interesting experimental result is the potential for handling a 200% thigh mass. One might surmise that thigh mass is another insignificant component of the robot dynamics. Based on our experiments, that conclusion is far from the truth. While testing the controller on the unaltered model, the DS phase composed 2% of the

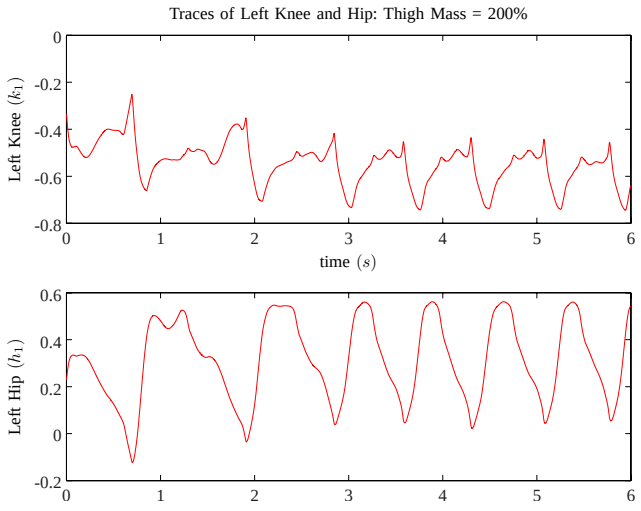


Fig. 6. Trace of hips and knees as in Fig. 5(a). Here, hip masses are both set to 200% of the actual robot. Notice the longer period and in particular more prominent peaks in the knee plot which indicate the DS push-off.

walk cycle. Altering the thigh mass increased the presence of this phase to 7%. Such a change was not brought about by any other disturbance. We conclude that during DS, our policy covers a wide range of states and corrects for this correlated shift in the cycle.

VI. Future Work

The results from the previous section indicate the potential for applying reduced dimensional dynamic programming towards control in complex state spaces. With this work as a baseline we can now explore numerous avenues of research.

First, we observe that even the decomposed state space forms a grid of substantial size. In fact, the dimensionality of the cached model ($\hat{\mathbf{f}}$) is the product of the state and action spaces. Consequently our approach was implemented on a coarse grid. Fig. 5 shows that significant sectors of this grid are not valid states. Methods such as [11], [14] that disregard these components would yield greater detail in the space that is relevant to the policy.

Alternatively, we can pursue the use of regression to depart from the grid-based architecture. The two models, the segmented value function and policy can all be approximated by means of locally weighted regression (LWR) [1] and similar methods. Our architecture is complementary to these techniques as it creates a significantly smaller space to be represented. In fact, by relating our ideas to [17] we can bootstrap the process of learning the LWR model on the application of our DP solution. In doing so, we can safely increase the effectiveness of our policy with reasonable training on the robot.

In a broader perspective, we are interested in developing automated means for decomposing complex dynamic state spaces. Analyzing principal components is a first step in this direction. However, computing the necessary coupling between phases of control as well as reasoning about the motion of lesser components remains a challenging and interesting domain. Further research in related directions will allow us to implement inherently robust techniques such as dynamic programming on a wider set of complex robotic applications.

VII. Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant Numbers ECS-0325383, CNS-0224419, and DGE-0333420.

References

[1] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11(1-5):75–113, 1997.

[2] G. Boone. Efficient reinforcement learning: Model-based acrobot control. In *ICRA*, 1997.

[3] G.; Aoustin Y.; Plestan F.; Westervelt E.R.; Canudas-De-Wit C.; Grizzle J.W. Chevallereau, C.; Abba. Rabbit: a testbed for advanced control theory. *IEEE Control Systems Magazine, 23(5), 57-79*, 2003.

[4] Scott Davies. Multidimensional triangulation and interpolation for reinforcement learning. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 1005. The MIT Press, 1997.

[5] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman. The simplest walking model: Stability, complexity, and scaling. *ASME Journal of Biomechanical Engineering, Vol.120 No.2 pp. 281-288 April*, 1998.

[6] A. Goswami, B. Thuilot, and B. Espiau. A study of the passive gait of a compass-like biped robot: symmetry and chaos. *International Journal of Robotics Research Vol. 17, No. 12*, 1998.

[7] Robert E. Larson. *State Increment Dynamic Programming*. American Elsener Publishing Company, New York., 1968.

[8] M.T. Mason. *Mechanics of Robotic Manipulation*. MIT Press, 2001.

[9] T. McGeer. Passive dynamic walking, international journal of robotics research. *International Journal of Robotics Research, 9(2), 62–82*, 1990.

[10] Seiichi Miyakoshi and Gordon Cheng. Ballistic walking by compass-like biped walker - exploiting physical dynamics in achieving human-like walking. In *Proceedings of 5th International Conference on Climbing and Walking Robots (CLAWAR2002) pp. 445–452., 1-86058-380-6.*, 2002.

[11] A. W. Moore. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In L. Birnbaum and G. Collins, editors, *Machine Learning: Proceedings of the Eighth International Workshop*. Morgan Kaufmann, 1991.

[12] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control, 26(1), 17-32*, 2003.

[13] J. Morimoto, G. Zeglin, and C.G. Atkeson. Minimax differential dynamic programming: Application to a biped walking robot. In *IROS*, 2003.

[14] Remi Munos and Andrew W. Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *IJCAI*, pages 1348–1355, 1999.

[15] L.M.; Phillips, J.; Silveira. Poor man's tbr: a simple model reduction scheme. In *Design, Automation and Test in Europe Conference and Exhibition*, 2004.

[16] Marc H. Raibert. *Legged Robots that Balance.* MIT Press, 1986.

[17] Jeff G. Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 1047. The MIT Press, 1997.

[18] M.W. Spong and G. Bhatia. Further results on control of the compass gait biped. In *IROS 2003, Las Vegas, Nevada, October 27-30*, 2003.

[19] O. Stryk. Numerical solution of optimal control problems by direct collocation. *http://www.sim.informatik.tu-darmstadt.de/sw/dircol/dircol.html*.

[20] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[21] J.W. Koditschek D.E. Westervelt, E.R. Grizzle. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control, 48(1), 42–56*, 2003.